Supporting Intellectual Work through Rendering and Review

Lars Yde

11 April 2001
# Contents

1 Introduction and background ........................................... 5  
  1.1 Acknowledgements ......................................................... 7

2 Purpose, methodology and overview .................................... 8  
  2.1 Introduction ................................................................. 8  
  2.2 The purpose of PeerView .................................................. 8  
  2.3 Methodology ................................................................. 10  
  2.4 Supporting Intellectual Work Through Artifact Rendering and Group Review ........................................ 11  
  2.5 Overview ................................................................. 12  
  2.6 Description of the Prototype ............................................ 13  
    2.6.1 System Overview ......................................................... 13  
    2.6.2 Architecture ........................................................... 13  
    2.6.3 The PeerView Client .................................................. 13  
    2.6.4 The PeerView Server .................................................. 15  
    2.6.5 Design Rationale ....................................................... 15  
  2.7 Implementation Overview ............................................ 16  
    2.7.1 The User Interface .................................................. 16  
    2.7.2 Communication and Data Distribution ................................ 16  
  2.8 Potential Uses for PeerView ........................................... 17  
  2.9 Earlier Work ............................................................... 18  
  2.10 Future Plans .............................................................. 19  
  2.11 Conclusion .............................................................. 20

3 PeerView - background and details ..................................... 21  
  3.1 Analysis ................................................................. 21  
  3.2 Design ................................................................. 25  
    3.2.1 Interface design .................................................... 25  
    3.2.2 Functionality design .............................................. 27
3.2.3  Class design ................................................. 28
3.3  Implementation .............................................. 29
   3.3.1  The scalable desktop .................................... 30
   3.3.2  Message and progress bar ................................ 32
   3.3.3  Communications layer .................................... 33
   3.3.4  Group directory .......................................... 34
   3.3.5  Discussion forum ........................................ 35
   3.3.6  Artifact updating ........................................ 35

4  Experimental evaluation of PeerView .......................... 39
   4.1  Introduction ................................................. 39
   4.2  The experiment .............................................. 39
   4.3  Discussion .................................................. 40
       4.3.1  Experimental findings .................................. 41
   4.4  User evaluation ............................................. 43

5  Implications of PeerView 1.0 for future systems ............. 44
   5.1  Introduction ................................................. 44
   5.2  Design analysis ............................................. 44
       5.2.1  Introduction ............................................. 44
       5.2.2  Interface ................................................ 45
       5.2.3  Functionality ............................................ 46
   5.3  Implementation analysis .................................... 47
       5.3.1  Introduction ............................................. 47
       5.3.2  Implementation language and technologies ............. 47
       5.3.3  Representation of properties and messages ............. 49
       5.3.4  Implementation of the distributed architecture ....... 50

6  InSiter ......................................................... 52
   6.1  Introduction ................................................. 52
   6.2  User audience ............................................... 52
   6.3  Design objectives ......................................... 52
   6.4  Interface design ............................................ 54
       6.4.1  Introduction ............................................. 54
       6.4.2  Screen designs ......................................... 55
       6.4.3  Design rationale ....................................... 60
   6.5  Architecture ................................................. 63
6.5.1 Information architecture ........................................ 64
6.5.2 Distributed architecture ........................................ 69
6.6 System requirements .................................................. 69

7 Conclusion ............................................................... 71
7.1 Results and contributions ........................................... 71
7.2 Lessons learned ....................................................... 72
7.3 Future work ............................................................ 73

A Design documentation .................................................. 74
A.1 Use cases .............................................................. 74
   A.1.1 Add documents .................................................... 74
   A.1.2 Remove documents ............................................... 75
   A.1.3 Open group directory dialog box ............................... 76
   A.1.4 Update group directory ......................................... 76
   A.1.5 Submit group directory ......................................... 77
   A.1.6 Create group ...................................................... 78
   A.1.7 Delete group ...................................................... 78
   A.1.8 Edit group ......................................................... 79
   A.1.9 Join group ......................................................... 79
   A.1.10 Select document from visible documents box ............... 79
   A.1.11 Zoom to overview ............................................... 80
   A.1.12 Centre document ............................................... 80
A.2 Class diagrams ....................................................... 80
A.3 Class diagram for package clientapp ................................ 81
A.4 Class diagram for package serverapp ............................... 82
A.5 Class diagram for package peerviewmisc .......................... 82

B Evaluation experiment materials .................................... 83
B.1 Task lists ............................................................ 84
   B.1.1 Danish version ..................................................... 84
   B.1.2 English version of task list for rank-and-file members .... 88
   B.1.3 English version of task list for group creator ............... 90
B.2 Transcript ............................................................. 92
   B.2.1 Danish version ..................................................... 92
   B.2.2 English version .................................................... 96
B.3 Questionnaires ......................................................... 100
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.3.1 Danish version</td>
<td>100</td>
</tr>
<tr>
<td>B.3.2 English version of PeerView questionnaire for first participant</td>
<td>107</td>
</tr>
<tr>
<td>B.3.3 English version of PeerView questionnaire for second participant</td>
<td>108</td>
</tr>
<tr>
<td>B.3.4 English version of PeerView questionnaire for third participant</td>
<td>109</td>
</tr>
<tr>
<td>C Source code</td>
<td>112</td>
</tr>
<tr>
<td>C.1 Introduction</td>
<td>112</td>
</tr>
<tr>
<td>C.2 List of listings</td>
<td>112</td>
</tr>
<tr>
<td>D PeerView help system</td>
<td>481</td>
</tr>
<tr>
<td>D.1 Introduction</td>
<td>482</td>
</tr>
<tr>
<td>D.2 Add documents</td>
<td>483</td>
</tr>
<tr>
<td>D.3 Remove documents</td>
<td>485</td>
</tr>
<tr>
<td>D.4 Panorama</td>
<td>486</td>
</tr>
<tr>
<td>D.5 Discussion forum</td>
<td>489</td>
</tr>
<tr>
<td>D.6 Group directory</td>
<td>491</td>
</tr>
<tr>
<td>D.7 Preferences</td>
<td>493</td>
</tr>
<tr>
<td>D.8 Using a divider bar</td>
<td>496</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction and background

Software development is difficult. This observation comes not only from my personal experience, but seems to be universally acknowledged by practitioners and researchers in the field alike [11, 51, 19]. Nor is it a particularly novel or sensational insight: numerous manuscripts and much research have recognized it, addressed it and, in many cases, proffered some panacea for remediating it. The problem persists, though, with no signs of abating. Software systems continue to be late, continue to frustrate their intended users and continue to be flawed or even dysfunctional [21]. The most important reason for what seems to be a perennial problem is probably the inherent complexity of software development with its complicated interplay of technological, social and psychological factors. The difficulty in managing this complexity is severely compounded by the intangibility of software engineering’s subject matter. Contrary to other branches of engineering which are typically concerned with the transformation of physical matter in one form or another, software engineering is concerned with the creation of symbolic artifacts. In other words, software engineers have no immediate sensory perception of the product they fashion because it is, in essence, a purely intellectual construct. In his much quoted 1987 article [11], Fred Brooks states it so:

“The reality of software is not inherently embedded in space. Hence, it has no ready geometric representation in the way land has maps, silicon chips have diagrams, computers have connectivity schematics . . . In spite of progress in restricting and simplifying the structures of software, they remain inherently unvisualizable, and thus do not permit the mind to use some of its most powerful conceptual tools. This lack not only impedes the process of design within one mind, it severely hinders communication among minds.”

Instead of direct observation, we rely on tools (compilers, debuggers, code analyzers, benchmarks, etc.) to help us gauge and probe the end product of a software development process, namely the code itself\(^1\). Such external measurement and experimentation can only process the syntactic and (formal) semantic properties of code. It does not yield information about its “sanity”, that is, whether it is sensible relative to its problem domain or whether it makes

\(^1\)A similar argument can be made for other phases, such as design and analysis but I will focus on program code for now.
logical sense\textsuperscript{2}. To that end, a developer must still rely on human review conducted by herself or by others, either informally or systematized as code inspection meetings, walkthroughs or the like [40]. The value of review and scrutiny of program code and other symbolic artifacts, particularly by others than the author herself, should not be underestimated. Mathematicians have known and applied this knowledge for centuries through the institutionalized practice of proving the logical consistency of their ideas and then subjecting those proofs to scrutiny by other mathematicians. Software engineering has rediscovered the value of this mechanism and has accumulated statistical evidence to support it. For example, in [30] Humphrey writes that

“...if you were to inspect a software product that contained 100 defects, how many would you expect to find? We will call the percentage of the defects found the yield of the review or inspection. Again, there are no published data but my experience at IBM was that inspections typically yielded between 60 percent to 80 percent. Data from another organization support this with a reported 68% yield for one large operating system.”

These characteristics of being intangible and subject to improvement through review is of course not exclusive to software. All artifacts of intellectual work seem to share them, consider for example coauthoring on scientific articles, proof-reading by editors, paper submissions by students to teachers, and so on. Software development is, however, conducted on an industrial scale and produces systems whose complexity dwarfs that of most other human artifacts. It might therefore be the area that could benefit the most from technologies, for example in the form of software tools, that could make software artifacts perceptible to its developers and thus more susceptible to both traditional and entirely new forms of inspection and review as the artifacts were being developed. Ideally, such tools would render a representation of a set of artifacts and the process by which they are brought about and would provide facilities for the generation and accumulation of review material concerning that representation. That is, in its fully fledged incarnation, the support tool envisioned would provide a both spatial and temporal rendition of a software project and facilities for communicating about it.

In Chapters 2 and 3 of this thesis, I present and discuss a prototype for such a tool, called PeerView, which was implemented by me in the latter half of 2000. The first part of that presentation will be an overview of this prototype in the form of an article coauthored with Jyrki Katajainen, which appeared as [62] and has been submitted to an international journal\textsuperscript{3}. It is followed by a more thorough discussion of the material it introduces. In Chapter 5, I analyze the design, implementation and development process of the PeerView project, and discuss what lessons can be derived from it. Next, in Chapter 6, I apply those lessons to the design of a more ambitious system that is to succeed and in part build on the PeerView prototype. This system, which has been given the working title \textit{InSiter}, will be implemented if the necessary resources can be made available, and will hopefully then be subjected to extensive usability testing. The thesis concludes with a brief summary and look at results achieved and experience garnered as well as a perspective on the future of this work.

\textsuperscript{2}The “sanity” of a design, a piece of code or other symbolic information might be called its \textit{natural semantics} to distinguish it from formalized descriptions of its semantics.

\textsuperscript{3}Another, somewhat shorter article appeared in the February issue of Distributed Systems Online [61]
1.1 Acknowledgements

Thanks to my supervisor Jyrki Katajainen who provided guidance and ample room for discussion, both technical and otherwise, and who helped ensure that i’s were dotted and t’s crossed by proof-reading the draft manuscript. Also, through financial support from grant 9801749 (project Performance Engineering), he made it possible for me to focus on thesis writing, not making ends meet. Thanks also to Morten Nielsen who gave useful feedback on late drafts of the manuscript which helped improve the final version. The three participants in the experimental evaluation — Paul Broholm Abrahamsen, Thomas Hyldahl and Rikke Iversen — provided valuable oral and written feedback by allowing me to tape record our communication during the experiment and by subsequently completing a questionnaire. I would like to thank them also for their assistance. The Department of Computing at the University of Copenhagen (DIKU) proved helpful by providing me with office facilities and funds for external consultancy on portions of the thesis manuscript. Thanks also to Kasper Hornbæk for some interesting feedback on the PeerView interface. Finally, thanks to my girlfriend Jeanette Vedelsjørne Gersager who helped me regain what I was at times lacking in moral fibre.

Errors, omissions and typos are hopefully few and far between, but whatever may persist are of course entirely of my doing.
Chapter 2

Purpose, methodology and overview

2.1 Introduction

This Chapter provides an overview of the prototype software that was developed for this thesis and the problems that it addresses. This Chapter also includes a description of the work methodology used for the present thesis, i.e. what was the intended progression of events from inception to conclusion. The concluding Chapter (Chapter 7) assesses how this methodology was implemented, i.e. how was the correlation between what was intended and what actually happened.

2.2 The purpose of PeerView

PeerView is an exploratory prototype. Its purpose is to provide groups of users involved in intellectual work with a common information space and facilities for communicating about the contents of that information space. The notion of a common information space (CIS) is not unequivocally defined in existing literature, but in [4], Bannon and Bødker discuss the concept and give a description that can be summed up for the purposes of this work by the following quote from [48]:

“…a central archive of organizational information with some level of ‘shared’ agreement as to the meaning of this information (locally constructed), despite the marked differences concerning the origins and context of these information items.”

They go on to note that:

“Cooperative work is not facilitated simply by the provision of a shared database, but requires the active construction by the participants of a common information space where the meanings of the shared objects are debated and resolved, at least locally and temporarily.”

PeerView is primarily intended as a CIS system for software developers since the artifacts of their shared work are often voluminous and it can be difficult to formulate the mental
equivalent of the physical representation that manual workers have of their work simply by virtue of their artifacts being physical. In other words, if one's subject matter are strings of symbols, the very act of making sense of what one is working on takes effort, whereas with physical matter, the act of making sense is an unconscious "making of sense" through processing of sensory data that does not require the sometimes painstaking intellectual effort needed to understand complex symbolic representation. Common information spaces thus seem to derive their justification from the way in which they can help automate the work of constructing and maintaining representations of external activities, namely shared work. The potential benefits are not only the savings in work time and effort that one would suspect to see when individual collaborators can more quickly and systematically access and absorb the information needed to carry out their work, but also the organizational rewards that can potentially be reaped from an integrative model of shared work. By the latter I mean that when workers and management can access a comprehensive and coherent model of the work that goes on within their organization or group, they acquire an object of both conversation and contention. They can discuss and monitor progress, either informally or by means provided by the workspace system itself, and can review the shared artifacts themselves in a more immediate way than if they were not accessible at a central locus. It seems plausible that this can ultimately affect not only the work process itself, but also the organization within which that process is undertaken since such a model can give all stakeholders in the organization a common focal point for discussion. The above is largely speculative and the intention behind PeerView is not only to give users a support tool for their collaborative work, but also to explore how these ideas can best be translated into reality.

In [27], Gutwin and Greenberg refer to a particular type of common information space, namely the shared workspace as “a bounded space where people can see and manipulate artifacts related to their activities.” [27, Section 2.2]. They go on to note that “In these spaces, the focus of the activity is on the task artifacts: the visible and manipulable objects through the task is carried out.” [27, Section 2.3]. This form of common information space is limited in that it usually seeks to reproduce or simulate a physical workspace such as a desktop or a conference room. Other forms of shared information, such as organizational structure or awareness of where coworkers are physically located and what they are doing, need not be present in the shared workspace or even deducible from it. If it is, the relationship between workspace artifacts and such awareness information might be entirely coincidental. PeerView's dominant feature is such a shared workspace where users can share and manipulate artifacts. The purpose of providing this workspace is both to give users a window onto the work of whatever group they are currently associated with and to experiment with the design and implementation of this type of common information space. The latter has yielded useful information, both through the process of development itself and through experimental evaluation, as discussed in Chapters 4 and 5, respectively. This information can be fed into the development of other systems that use common information spaces to support collaboration, one of which is outlined in Chapter 6.

In addition, PeerView demonstrates novel methods of navigation, at least in a groupware context, by giving users a fully zoomable desktop on which the shared artifacts can be placed and manipulated. This is an experimental way of addressing the “detail/overview” problem of how to represent both local detail and global overview in a usable manner which is discussed in [22], where the authors describe a different solution to this problem, namely the fisheye lens. PeerView also addresses the practical problem of catering to groups operating in a
heterogeneous computing environment by being implemented in pure Java, i.e. without any
native code, and therefore able to run on a variety of machine architectures and platforms.

Being a prototype, PeerView also helps answer the development question: how can a fully
fledged CIS support system best be developed? PeerView may be limited in its functionality
but it incorporates all the components that should be present in more ambitious systems
requiring many man-years of development time, and the experience gained during all phases
of PeerView development can facilitate implementation of such systems. Chapters 3 and 5
describe the development process and discuss what lessons can be learned from it.

PeerView imposes little structure on its users and their work process. This means that it
affords users a great degree of freedom which is in keeping with an observation by Grintner
[23]: “one of the emergent trends in workflow research . . . is to make the systems more flexible
to accommodate the contingent aspects of work.”. However, it also means that PeerView does
little to systematize and automate work for its users. Later systems, such as that described
in Chapter 6, will have to address the specific needs of a more narrowly defined user audience
to go beyond the simple functionality found in PeerView. In a sense, this implies a range of
possible applications from the pure, undressed support of shared information in a CIS to a
highly structured, user customizable workflow system centered around a CIS. Viewed in that
light, PeerView is clearly towards the former end of the spectrum. For that same reason,
PeerView may appeal to a broad audience but does not purport to appeal to an equally broad
range of work situations. That is, although PeerView may be used by anyone from the office
clerk with basic computer literacy to the veteran software developer, it does not necessarily
support all of their work situations and nor does it have to in order to fulfill its purpose.

2.3 Methodology

The methodology used in this project is a hybrid resulting from the need to reconcile scientific
exploration with the practicalities of software development. The idea for a distributed ap-
lication supporting intellectual work through rendering and review sprang from reading [52]
among other works, and from personal experience with software development projects where
lack of communication and coordination can complicate or derail efforts. Now, this conception
of an idea could have been followed by lengthy elaboration process where the initial ideas were
translated into a detailed paper model, i.e. a verbal and diagrammatic description, and not
necessarily implemented as an executable application. This approach would obviously have
allowed for a much more ambitious initial design since the lengthy implementation cycle of
“code-execute-debug” could have been eliminated. However, I believed such an approach to
be of questionable scientific value at best and worthless at worst since a pure “armchair de-
sign” that was not put to an experimental test through implementation would be much like
a physics theory that was never empirically tested, i.e. insubstantial though possibly intellec-
tually stimulating. Also, most software development methodologies advocate incremental or
iterative development [6, 42, 32] which corresponds to my personal experience which is that
most system development benefit from many short iterations rather than monolithic, phased
design as in the traditional “waterfall” model [51, p. 9]. I therefore decided to advance quickly
from idea to design and implementation, followed by a posteriori analysis of the development
process and evaluation of the finished prototype in the hope that this would yield valuable
feedback based on empirical data that could help improve the design and development of

10
future systems.
Putting a methodology on a summary form as done below is slightly misguided as methodology is more a mindset and structured approach than a formulaic procedure. However, it is illustrative to see the overall steps taken put into a list form so that is done here:

1. Inception of PeerView
2. Elaboration of PeerView
3. Construction of PeerView
4. Transition
5. Evaluation
6. A posteriori analysis
7. Elaboration of successor system (next iteration)

To anyone familiar with the unified software development process [32], the above sequence will be recognizable, at least in part, because the activities “Inception”, “Elaboration”, “Construction” and “Transition” are identical to the phases in a “workflow cycle” to use unified process parlance. However, this is not in itself of scientific merit since these developmental phases do not include formal evaluation of the end product and the process by which it was brought about, and therefore does little to verify their validity. The evaluation and analysis phases of the process used for this work are intended to address this shortcoming and help give this thesis the scientific potency needed to have an impact beyond the software product resulting from it. In Chapter 7, I discuss how the actual progression of events corresponds with the process intended and what room there might be for improvement.

2.4 Supporting Intellectual Work Through Artifact Rendering and Group Review

The following is a slightly abbreviated version of an article that appeared as [62] and has been submitted to an international journal. Only the introduction has been cut to avoid overlap with Section 1. The article appears here to provide an overview of the PeerView software.
Supporting Intellectual Work through Artifact Rendering and Group Review

Lars Yde
Department of Computing, University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen East, Denmark
larsyde@diku.dk

Jyrki Katajainen
Department of Computing, University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen East, Denmark
jyrki@diku.dk

Abstract. Intellectual teamwork, such as that done by software development teams, is characterized by the intangibility of its subject matter. This can make it difficult for team members and outside stakeholders to gain an overview of the product being built. Many software tools of various designs have addressed this problem, often by using graphs and charts to model ongoing projects. In this paper, we suggest a design based on artifact rendering and group review and argue that it can promote overview and communication. We also present software built from that design and describe its potential uses and audience as well as its implementation.

Keywords: Computer-supported collaborative work, component-based development

Category: H.4.3, H.5.3, K.4.3

2.5 Overview

In [Section 2.6] we describe a prototype of a tool intended to support intellectual teamwork in general. That is, we aim at supporting any process in which a symbolic product, an artifact, is being produced by individuals in group collaboration, so the use is not limited to software developers. Our main design goal was to promote overview by artifact rendering and communication by group review. By artifact rendering we mean the representation of intellectual artifacts by some medium to facilitate processing by humans, and by group review the activity through which team members exchange information and commentary on the artifact being built. Our prototype uses visualization to render an artifact that consists of textual and graphical documents, but other forms can be envisaged, i.e. auditory representation. Also, discussion fora can be associated with parts of the artifact rendered.

Our prototype was developed using the Java programming language and a set of compatible component technologies. In [Section 2.7] we discuss both and argue that implementation was possible only through such component-based development, given our limited resources. Next, in [Section 2.8], we propose some potential uses of the prototype, both in its present form and in subsequent versions. We then go on to discuss earlier related work in [Section 2.9] and finally, in [Section 2.10], we outline our plans for a more advanced system.
2.6 Description of the Prototype

The prototype aimed at fulfilling the basic needs of artifact rendering and group review was named PeerView and work on it was begun in the middle of May 2000 as part of Lars Yde's M.Sc. thesis. At the time of writing (mid October), a beta release has been completed and a stable version is planned for release before the end of the year. The system is released under a freeware license both in its binary and source form. In this section, we outline PeerView's design.

2.6.1 System Overview

PeerView provides users with a dynamic representation of on-going work by rendering the artifact of that work in a form which allows easy overview as well as inspection and discussion of detail. PeerView accomplishes this by allowing groups of users to share documents and have them rendered in a scalable visual panorama. The set of documents is constantly kept updated so as to provide a real-time window onto the artifact (e.g., a collection of code files) being developed. With each document is associated a discussion forum that enables a group to have discussions on the document in a manner similar to a USENET [57] newsgroup having discussions of a set topic.

2.6.2 Architecture

PeerView is a client/server application with “fat” clients and a “thin” server, meaning that most system functionality is placed in the client application and relatively little in the server. Each PeerView user is assumed to be running a single instance of the client program on his or her machine, but multiple instances are possible, provided they are running from separate locations (different directories). The server may run on one of the users’ machines or it may run on a separate host. Currently, PeerView supports two types of connection: TCP/IP sockets and HTTP. The former is intended for local or closed networks, the latter for use over the Internet, but both may of course be used differently.

2.6.3 The PeerView Client

The client application provides a window onto a set of documents via a zoomable panorama [Fig. 2.1, centre] where documents can be arranged in an arbitrary pattern and zoomed to arbitrary scale as well as moved and sized individually. By centreing a document, the user can access its discussion forum [Fig. 2.1, bottom] and read as well as add contributions to the on-going discussion on that document. The top of the client window is occupied by a selection panel consisting of standard drop-down menus and toolbar buttons. The drop-down box in the right-hand side of the toolbar panel lists the documents currently displayed in the panorama so that each can be selected by name as well as by navigating the panorama using a mouse.

From the selection panel, the user can choose to add or remove documents, customize the client using a preferences dialog or access and modify the group directory, i.e., the list of groups currently available. The directory appears on top of the main window as shown in [Fig. 2.2] where the panorama is zoomed to overview, displaying in this case 32 documents arranged
in a grid pattern which can be customized by the user from the preferences dialog available from the drop-down menus. Each group is listed with its current number of participants, the number of documents it comprises and their total volume in bytes. Using the button panel at the bottom of the group directory box, the user can join, create, edit and delete groups. After joining a group, the other group members, if any, are instructed by the client application to submit their documents so that they can be added to the panorama of the new group member.
The new member's own documents are then broadcast to the other group members so that their panoramas can be similarly updated.

As indicated earlier, centring a document (by double-clicking it) causes the PeerView client to request the discussion forum for that document from the server and, upon receipt, display it in the bottommost section of the main window. From there, the user can select a position in the topic tree [Fig. 2.3, left] and using the small vertical toolbar panel choose to compose a new message, delete an existing message that has not yet been responded to, or view the property sheet of the selected message.

### 2.6.4 The PeerView Server

The central purpose of the server is to maintain a database of clients, groups and discussion fora which is reflected in its user interface [see Fig. 2.4]. It consists of little else than a listing of the groups supported, a toolbar and a menu panel from which a few, basic functions (setup and help) can be selected.

### 2.6.5 Design Rationale

The choice of a client/server architecture was made based on the responsibilities of the server. A peer-to-peer architecture would have been equally feasible but would merely have placed the server functionality in an arbitrary client, resulting in little discernible difference to the user. More significant is the use of a zoomable panorama. As explained earlier, the underlying motivation is to provide overview without loss of content. The user can achieve this by zooming the arrangement of documents to his or her preferred scale, of course, but by combining zooming with a linear setup, an effect similar to that found in work by Eick et al. on the visualization

Figure 2.3: Discussion forum.

Figure 2.4: The PeerView server window.
tool SeeSoft [52, p. 315] can be produced. SeeSoft uses a linear arrangement of multi-coloured columns to render an overview of a collection of code files and their associated statistics (such as time of modification), thus giving users access to large amounts of information at a glance. On a more theoretical level, the idea of a customizable panorama that can be viewed at arbitrary levels of detail is in keeping with the notion of micro/macro readings as described by Tufte in [55, p. 38 ff.]:

"...the power of micro/macro designs holds for every type of data display as well as for topographic views and landscape panoramas. Such designs can report immense detail, organizing complexity through multiple and (often) hierarchical layers of contextual reading."

Tufte’s focus is on the presentation of cartographical and scientific data but his observations seem to apply equally well here.

2.7 Implementation Overview

The most important considerations in choosing implementation technology for PeerView were: broad user appeal, ease-of-use, high degree of portability, extensible structure and reliability. The Java programming language seemed the obvious choice given these requirements, but implementing all of the planned functionality bottom-up within the set time-frame of four months was almost surely infeasible since implementation was in the hands of a single developer (Lars Yde). The solution lay in using component libraries to implement some of the low-level features such as data distribution and visualization that would otherwise have demanded weeks of development, testing, and fine-tuning to function satisfactorily. The specific choice of components and the reason for choosing them are detailed below.

2.7.1 The User Interface

The PeerView interface is implemented using the Swing classes found in the Java Development Kit (version 1.2.2) [50]. These are light-weight graphical components meaning that they make no (or only very limited) use of operating-system-specific code, thus ensuring maximum portability and a minimum of visual variability across platforms. The zoomable panorama which occupies the centre portion of the client window is built using the Jazz toolkit (version 1.0) [34], which provides primitives for building scene graphs and scaling text and/or graphics. (A scene graph is a tree structure for organizing graphical objects.) Jazz interacts smoothly with other Java technologies and its structure facilitates future extensions and modifications. In that sense, it allows development time to be shortened without imposing arbitrary restrictions on future development efforts.

2.7.2 Communication and Data Distribution

The PeerView communication infrastructure is implemented using the Java Shared Data Toolkit (JSDT, version 2.0) [36]. It is a relatively low-level toolkit compared to some of the alternatives surveyed, such as NCSA Habanero [28], TANGO Interactive [53], and DreamTeam
[18], all available online and free of charge. However, it has the distinct advantage of being 100% pure Java, meaning that the risk of incompatibility problems — a frequent stumbling block to developers — seemed minimized.

The JSDT requires a server to maintain a directory of clients and groups, and allows all members of a group to communicate and exchange data using an arbitrary number of channels. Currently, PeerView uses only a single channel per group, but extending the design later on would be easy as would adding a number of advanced features such as data encryption, access restriction and communication statistics gathering, all of which are supported in part or whole by the JSDT.

As mentioned earlier, PeerView currently supports two communication protocols, namely TCP/IP sockets and HTTP, but can easily be extended with additional protocols since the JSDT design is completely independent of its underlying implementation. Such an extension is, however, a low priority as other, more interesting features (suggested in the previous paragraph) are higher on the agenda.

2.8 Potential Uses for PeerView

PeerView was designed for a specific purpose, namely support of intellectual teamwork, but was deliberately made flexible through simple design and portable technology. It is therefore suited for other uses than that of facilitating intellectual work. Some possible applications of PeerView are listed below. Many of these are inspired by examples found at the Jazz website [34].

Repository inspection: PeerView could be used for the inspection of document repositories such as those maintained by the version management systems that are often used in software development. It is planned to try this in practice in the near future by allowing users to download the PeerView client software from a website and then access a CVS repository using that software (CVS is short for Concurrent Versions System — a popular open source version management system, see [14]).

Web browsing: PeerView can display HTML documents in its present form and can be extended to support hyperlinks in future versions, and could, after such an extension has been implemented, function as an inter-/intranet browser.

Presentation: Documents can be displayed and arranged in PeerView’s panorama and subsequently used in a presentation much like conventional slides but in a more manageable way, for example, by projecting the panorama onto a canvas and then using a mouse to navigate while giving the presentation.

Authoring: PeerView could be used as an authoring tool by researchers who could have separate copies of a shared manuscript placed on the panorama and then edit and annotate their version while conferring with others about their copies. This could also be used for commentary and proof-reading and would scale well since the zoomable PeerView panorama need not consume more user screen space as the number of authors increases.
Education: Due to its simple design and graphical interface, PeerView is accessible to most users, including children and some groups of disabled people, and so could be used both for distance learning in geographically dispersed communities and as a support tool in conventional education. One obvious example of the latter is letting visually impaired students study teaching materials at their preferred magnification using the zoomable PeerView panorama.

The above points to some general areas of application, but the number of specific uses is potentially quite large because of PeerView’s extensible, open-ended architecture and design.

2.9 Earlier Work

The extensive directory of computer-supported collaborative work and groupware resources at [56] lists a large number of groupware applications for shared editing, conferencing, emailing, and other forms of group collaboration. In theory, such functionality can serve the same purpose as PeerView, but in practice, many of these products are designed for different purposes and are thus ill suited for artifact rendering and group review. Two examples are Cybozu Office [15] and Lotus Notes [38], both of which are large suites of tools for collaborative work that facilitate resource sharing and communication. However, their size and complexity makes it impractical to use them for the narrowly defined and specific purpose that PeerView has.

Several academic initiatives have explored areas related to artifact rendering and group review and have produced software that addresses the same underlying problems as PeerView. Examples are the TeamRooms software [46] created for supporting team collaboration and group awareness, and the WORLDS software [60] used for distributed access to one or more repositories of information. Both are similar in architecture to PeerView and are aimed at facilitating collaborative work through object sharing and group communication, but their designs differ from PeerView’s in key areas. Both TeamRooms and WORLDS are based on spatial metaphors, i.e., the notion of shared electronic spaces. TeamRooms realizes this idea by letting users define rooms which are persistent fora where objects can be shared and communication can take place between the visitors of the room. Each room is equipped with facilities for collaborative work in the form of applets for exchanging information and carrying out other tasks. The TeamRooms user interface is dominated by a panorama onto which the contents of the room (i.e., the shared objects) are projected for inspection and manipulation by all users “present” in the room. An implication of such a design is of course that the set of shared objects can easily occupy more than the visible screen space and TeamRooms addresses this problem by offering a room overview radar that gives an outline, non-scalable bird’s eye view of the room (the work on radar overview has been continued at the University of Calgary [24] where TeamRooms was developed).

On the conceptual level, PeerView differs from TeamRooms by being a tool for artifact rendering and group review whereas Teamrooms provides shared electronic spaces for collaborative work. This is the central difference since the conceptual nature of a system usually dictates future design and development efforts. In terms of design, PeerView has fewer features than TeamRooms and consequently a simpler interface. It also addresses the issue of artifact overview differently, namely through its scalable, configurable panorama, but it is similar in its
use of object sharing by projection onto a panorama surface. Technologically, the most important difference is that PeerView is written in Java while TeamRooms is a Tcl/Tk application which obviously impacts portability and potential audience since Java is platform-independent and Tcl/Tk is not, although it is widely supported. The WORLDS collaboration environment, which is called Orbit, differs from PeerView in much the same way as TeamRooms, but has a more complex architecture and, consequently, more stringent system requirements than both TeamRooms and PeerView.

2.10 Future Plans

Given the open design and implementation of PeerView, we expect little difficulty in integrating new features suggested by future users and ourselves. A handful of useful extensions have already been brought to our attention and are listed below:

- New display managers to allow advanced document formats such as RCS (the format used by CVS [14] for storing the revision history) and streaming video in addition to the existing formats which include plain text, HTML, RTF, GIF and JPG.
- A selection of layout managers, i.e., user configurable program components for automated control of the documents in the client panorama, in addition to the default manager (the grid layout manager) now available.
- An editable and extensible property sheet associated with each document.
- Discussion fora for groups of documents organized by owner, topic, or some other shared characteristic.

The above features are for integration into future releases along with any additional suggestions deemed worthwhile.

The long term plan is to gather feedback about the use of PeerView and from that assess the viability of its philosophy and design. Provided this assessment is favourable, we plan to develop a more ambitious system for real-time artifact rendering and process rendering. That is, in addition to an artifact, the organization and history of a project is rendered. The system is to assemble global statistics about the state of a project by collating information and documents gathered from individual clients distributed in a network. Relevant statistics would be the number of documents worked on by each user, the number and position of changes made to those documents and, for code files, profiling and debugging information. That information should then be organized by the server and transmitted around the network to an arbitrary number of consumers who would render it, either visually or by some other medium determined by the preferences of the user. Moreover, the system could offer the review facilities of PeerView along with a set of more advanced features. Ideally, the latter would include “scribbling pads” to communicate in pictures/diagrams in addition to words, audio and video conferencing to allow real-time communication among group members, and playback facilities for observing the evolution of a project over time. Thus, by combining artifact rendering with process rendering, a both spatially and temporally accurate representation of on-going work is given.
2.11 Conclusion

The problems that PeerView seeks to address are not new, but its approach and design are relatively untried. Only user feedback and continued research will tell if such an approach is justified and has enough substance to bear larger initiatives. If so, the potential benefits from a more ambitious project as outlined seem significant. On a more philosophical level, the potential merit of the PeerView approach comes from the fact that it is a response to a real and pervasive problem in software engineering, namely how to get a cognitive grip of an intrinsically intangible and usually highly complex subject matter. In that sense, PeerView can be seen as a small step in the evolution towards CASE tools that can help software engineering attain the maturity of other engineering disciplines and thereby, hopefully, stem the pandemic of failed projects.

Software Availability

The latest PeerView binaries are available online at:

http://www.diku.dk/research-groups/performance-engineering/PeerView/

The source code and accompanying documentation will be made available at the above address before the end of this year (2000). Future developments and release plans will also appear there as will related resources. Please feel free to download PeerView and test its usefulness for yourself.

Acknowledgements

This work was supported by the Danish Natural Science Research Council under contract 9801749 (project Performance Engineering).
Chapter 3

PeerView - background and details

The article format places some natural restrictions on style and content, so the level of detail in the above has by necessity been limited. The following paragraphs seek to remedy that by providing the background details on the rationale for the design of PeerView, the reasons for the choice of implementation technologies and architecture, and a discussion of the development process. In the below, PeerView is referred to with its current version number, 1.0, when necessary for the specific point being made or item being discussed. In all other cases, it is simply referred to as PeerView, i.e. without a version number.

3.1 Analysis

The analysis phase of the PeerView 1.0 design consisted mostly in refinement of the ideas that underpinned it, namely support for intellectual work through artifact rendering and group review. For a full-scale system, this would have included empirical research among a target audience to ascertain their preferences and proficiency. Even for a prototype system like PeerView 1.0, field research would have been useful, but I deemed it infeasible within my set timeframe. Naturally, that decision influenced my perspective on the design: PeerView was to be a vehicle for certain ideas, and at the same time be potentially useful to a wide audience to demonstrate the generality, in addition to the feasibility, of those ideas. The implication to me was that PeerView should be simple enough to maintain focus on the core ideas and facilitate wide use, yet comprehensive enough to provide both real functionality for its users and material for future research.

The central issues that had to be resolved first are discussed below in the order they were considered.

The semantics of representation

A central question was of course: what was meant by artifact rendering in practice? What needed clarification, in other words, were the semantics of representation, i.e. what constituted an artifact and which of its constituents should a representation render to provide a useful model for its users.

The answer lay in the fact that any artifact of intellectual work can be represented by a finite
string of symbols. The plausibility of this statement follows from its wording: if intellectual work has resulted in the creation of an artifact, then it has been converted from its abstract state ("thought stuff") to material form (a physical artifact). If we assume that all material objects can be exhaustively described using some symbolic formalism, then any artifact can be encoded in a computer readable form since any such formalism can be converted to binary numbers using a binary enumeration of its symbols. This is an abstract formulation, to be sure, but it allows us to focus on the essential aspect of computerized artifacts, namely content. Whether the artifact be text, graphics, audio, or something else entirely, it will always have a string representation. This is one defining property of an artifact, but it is so abstract a definition that it is almost vacuous. Any rendering mechanism would require additional information, i.e. "meta-data", about the artifact such as its medium (text, graphics, audio), its author, possibly its revision history and so on. The exact composition of the meta-data should depend on the type of rendering used: 3D graphics might require different information than 2D graphics, for example.

The form of representation

With a workable, albeit abstract, definition of the artifact concept in place, the next step was to decide on an appropriate form of representation, i.e. the appearance of the artifact rendition. I chose visual rendering as the basis for the interface design, but although this may seem the obvious choice it is not predetermined, at least not if one defines visualization as done in [52, p. 32]:

"Visualization may be defined as ‘the power or process of forming a mental picture or vision of something not actually present to the sight’…Notice that this definition allows for the use of sensory modalities other than vision, e.g. hearing…to assist in the formation of mental picture or images."

The reason for my preferring a visual rendering mechanism was the conceptual and technical difficulty I foresaw in implementing a system that supported multiple sensory modalities or merely relied on a single modality other than sight. Also, it seemed overly experimental to base a prototype design on something as relatively untried as auditory, tactile or, in the extreme, olfactory perception!

The visual interface should of course be tailored to needs and expectations of its intended audience, but there were other, more specific concerns. The design chosen would for example have to address two recurring problems for designers of graphical interfaces, namely visual clutter and overcrowdedness, meaning that interfaces can fill up with objects such as windows stacked upon each other until they make up an unmanageable jumble. For systems based on the desktop metaphors there are various solutions such as virtual desktops and multiple desktops on the same machine, but for systems with a potentially large number of objects, such as groupware applications relying on object sharing, this may not suffice. Such has been the experience of Greenberg et al. who in [47] describe the problem as follows:

"A central concern in information visualisation is how a system can present both global structure (that provides overview and context) and local detail (that reveals information in the user's area of interest)."
They then propose two complementary software solutions to the problem, namely a fisheye view mechanism for use with shared editors so that multiple authors can follow each other's work within the same focus, and a magnification lens for use with a graph editor, allowing customized magnification of portions of a graph being edited.

I did not become aware of the above research until after I had begun implementation, but I seem to have followed a similar line of reasoning, namely that the central problem was that of combining local display with global view. This initially led me to 3D graphics as that intuitively seemed the best way to represent a large volume of visual information within the confines of a desktop display. However, the advantages of 3D display, namely increased information volume, flexibility of design and navigation and appeal to human spatial cognition should be weighed against its disadvantages, summarized in [1]:

- Initial confusion and disorientation combined with the additional complexities of the interface and greater freedom of movement. Navigating with six degrees of freedom within an information space as opposed to navigating a set of 2D windows on an information space through panning and zoom controls.
- Large processing overheads are evident when a typically fast computer is reduced to crawl by a complex visualisation. Additional equipment such as a large high resolution monitor and specialized input devices are also a benefit, though not necessarily a necessity.
- Navigating any 3D environment requires a level of spatial awareness from the user. Navigating within an abstract information space demands more from this skill. Such spaces do not typically conform to our preconceptions of a 3D environment and various expectations such as a notion of “up” and “down” can easily be violated.

Also, it was my experience that 3D graphics displays are more taxing to develop than 2D graphics displays since the former is inherently more complex, thus requiring more resources to implement, debug and maintain. This gap has been closing for some time as freely available component libraries mature (such as OpenGL, see [44]) since these can help lift the burden of implementing a 3D renderer. The point remains, though, that 3D graphics add complexity to a user interface and have the added disadvantages listed above. I therefore decided to use 3D only if I could not formulate a satisfactory 2D alternative.

The challenge was therefore to create a 2D display that could be used for detailed inspection of each element in a set of artifacts yet easily transformed to provide an overview of arbitrary subsets. A scalable desktop seemed to have all the desired characteristics. If one had a 2D surface based on the standard desktop metaphor that could be scaled to arbitrary levels of magnification, one could place a set of artifacts on that desktop and navigate it along the x, y and z axes, thus solving the problem of visual clutter while adhering to GUI conventions known by most users and well supported technologically. In terms of implementation, this solution seemed manageable since scaling a desktop is equivalent to scaling a bitmap which is a well researched graphics transformation [2, p. 78 ff.] This still left the exact appearance of each artifact on the desktop to be determined, but I considered that a design question and thus left it to be determined at that stage.

23
Modes of communication

To be a prototype for both the ideas of artifact rendering and group review, PeerView should provide facilities for communicating about its rendering in addition to facilities for displaying it. There are a variety of existing models for supporting group communication, such as mailing lists, USENET news, online chat, email services, and so forth. These provide suitable metaphors for communication facilities in some existing groupware systems such as FirstClass [12] and Lotus Notes [38], and were therefore both tried and conventionalized which made them seem safe, albeit conservative, choices of communication medium. I considered opting for a more experimental solution, but decided to err on the side of caution, bearing in mind the intended audience for PeerView and the time constraints I faced.

I reasoned that PeerView 1.0 should at least have facilities for threaded discussions in the manner of a USENET discussion [57] and, preferably, support for online peer-to-peer communication (chat) since this would allow review material to be accumulated and organized in a tried and tested form, and still allow sundry communications to take place more privately, i.e. without the perhaps dissuasive logging that discussion fora entail. Adding more features might contribute to usability, but only at the expense of interface simplicity and development time.

Architecture

Since PeerView was to be used for group review about a set of artifacts which, by implication, would have been shared, it was clear that the system architecture had to be distributed. There are a number of ways to implement such an architecture; some commonly used models are peer-to-peer, client/server and multicast, so a choice still had to be made between the options available. Under ideal circumstances, this choice would be made without consideration for the technologies available; one would simply assume that the required means of implementation were obtainable and design accordingly, i.e. without thought of technological limitations other than the state-of-the-art. However, this is obviously an unrealistic stance since lack of technology support can thwart even the best of designs. It was infeasible for me to build a distributed architecture bottom up within my set timeframe. I therefore decided to only survey component libraries for building communications layers in distributed applications, preferably tailored specifically for collaborative applications. This meant that the implementation language had to be decided on since any communications library was apt to be specifically targeted at a single programming language.

I considered C++, Java and SmallTalk as possible implementation languages because they seemed to offer the best mix of extensive library bases, cross-platform support, good development facilities and reasonable run-time performance. Java was chosen over C++ and SmallTalk because of its inherent cross-platform support and the existence of suitable component libraries for the implementation of PeerView. In fact, the existence of suitable component libraries came to dictate the choice of programming language rather than vice versa. In particular, it quickly became apparent that numerous groupware toolkits [25] existed that were almost all implemented in Java and that more general purpose libraries for collaborative applications, notably the Java Shared Data Toolkit (JSDT) also were available. I chose the latter for PeerView

---

1Section 6 has a more detailed discussion on alternative implementations.
since the toolkits I surveyed all had drawbacks: either they were discontinued, as in the case of Habanero [28], not suited [18] or otherwise unsatisfactory [53]. Furthermore, JSDT had the added advantage of being pure Java and licensed by Sun which minimized the potential for development difficulty such as interfacing different technologies.

JSDT can be used to implement client/server and multicast architectures, but the server need only be assigned minimal responsibilities such as maintaining a registry of clients so it could be used to construct an architecture that effectively would function in a peer-to-peer manner. I was therefore not concerned about limiting myself to a particular architecture prematurely since JSDT seemed flexible enough to accommodate potential revisions later on in development.

3.2 Design

A commonly used distinction is that between the interface of an application, meaning the graphical components that the user interacts with, and the functionality, meaning the actions that the user can carry out by interacting with the interface. This is not the only possible division, of course, but it reflects the actual design process, so the below discussion reflects that view.

3.2.1 Interface design

Java was chosen as the implementation language for PeerView during the analysis phase, and this in part dictated the design of the GUI since the look-and-feel (as the graphical appearance and response pattern of an interface are sometimes called) of the interface widgets (i.e. windows, buttons, scrollbars etc.) should be that provided by the Java AWT or Swing interface classes. Choosing a different look-and-feel did not seem sensible as the Java classes implement all common interface primitives and can be extended arbitrarily. I chose the Swing class set which is Java’s lightweight, i.e. platform independent, GUI library rather than the AWT which shipped with early releases of Java, but is now being phased out.

The specific layout of the PeerView interface should both cater to its intended audience by being accessible and intuitive or at least adhering to convention, yet at the same time fulfill its other purpose, namely that of clearly demonstrating the ideas it is built on. The most important design goal could therefore be stated as providing the user with access to the functionality required for carrying out a particular task without distracting from that task, preferably in an aesthetically pleasing manner. In practical terms, this meant sparse displays, i.e. windows and boxes which are not densely packed with widgets, and sensible layout, e.g. associating selection controls with the display elements they controlled. This is supported by Schneiderman [49, p. 318] who reports:

“Crowded displays are more difficult to scan, especially for novice users. In a NASA study of space-shuttle displays, sparsely filled screens with approximately 70-percent blanks were searched in an average of 3.4 seconds, but more densely packed screens with approximately 30-percent blanks took an average of 5.0 seconds . . . This study also demonstrated that functionally grouped displays yielded shorter search times.”

25
It is also in keeping with the heuristics formulated by Nielsen in [43] who advocates:

“The ideal is to present exactly the information the user needs — and no more — at exactly the time and place where it is needed. Information that will be used together should be displayed close together . . .” [43, p. 116]

By aesthetically pleasing I did not mean merely according to my personal preferences, but in the sense of consistent use of simple and tasteful visual presentation to further usability. This issue is discussed at length by Nielsen who, among other things, offers the following aphoristic advice on colour, information density, and graphic design, respectively:

“Don’t overdo it. An interface should not look like an angry fruit salad of wildly contrasting, highly saturated colors. It is better to limit the design to a small number of consistently applied colors.” [43, p. 119]

“The ‘less is more’ rule does not just apply to the information contents of the screen but also to the choice of features and interaction mechanisms for a program. A common design pitfall is to believe that ‘by providing lots of options and several ways of doing things, we can satisfy everybody’. Unfortunately, you do have to make the hard choices yourself.” [43, p. 121]

“Screen layouts should use the gestalt rules for human perception to increase the users’ understanding of relationships between the dialogue. These rules say that things are seen as belonging together, as a group, or as a unit, if they are close together, are enclosed by lines or boxes, move or change together, or look alike with respect to shape color, size or topography.” [43, p. 117]

Having settled on the guidelines for design, I immediately began translating it into reality using a rapid prototyping tool. Some software engineering pundits recommend designing displays on paper before committing them to electronic form, but with the graphical design tools available today, I find this approach unwieldy. I started by designing the PeerView client main window since this was the “control center” of the application, so to speak. The result is shown in one of the figures used in chapter 2.4, namely Figure 2.1. The window is divided into three parts: the control section at the top, the middle portion containing the scalable desktop and a bottom portion containing the discussion forum, which in turn is divided into a navigation frame and a content frame. The sizes of these elements, except the control Section, can be adjusted using the divider bars2 that separate them. This layout was chosen because it correlated with the conventions of GUI design by having control at the top, work area below and text input areas at the bottom. I would have preferred to have the quality of this design verified through suitability testing in a realistic setting, but had neither the time nor the means to do so, which was another reason for making conservative design choices. The drop-down menus and toolbar in the control Section were populated by items reflecting the functionality I could foresee from the results of my analysis, and were therefore likely to change as design, and perhaps even implementation, proceeded. The same applied to the pop-up menu that could be activated

---

2 Vertical and horizontal bars with a ridged surface and arrow points at one end.
by right clicking on an artifact, i.e. I put in the menu items I believed necessary to have a functioning product in place and left room for extensions if need be. The list box in the right hand side of the toolbar contains a listing of artifacts, i.e. documents, shown in the panorama organized by author. By selecting a title from the box, the user can center the panorama on the corresponding artifact. This form of iterative development may seem meandering, but it is, in my opinion, more realistic than a rigid, linear design process that leaves little room for exploration.

The PeerView server window, shown in Figure 3.1, reflects its purpose, namely to serve as registry and nexus for communication between clients. Consequently, it is kept simple with only a control Section at its top and a listing of groups below that.

### 3.2.2 Functionality design

As I had found suitable technologies for implementation during the analysis phase, some of the design issues had already been settled implicitly. Specifically, the Java Shared Data Toolkit implements the concept of clients partaking in sessions using channels as lines of communication. Using that technology therefore meant designing functionality accordingly, i.e. supporting similar concepts. The PeerView client application was therefore designed to operate in groups, which were to be implemented as JSDT sessions (“group” was chosen because it connoted a more permanent affiliation than “session”), and communicate using JSDT channels. This was the bare minimum of communications infrastructure, but I decided to limit the initial design to that to keep implementation manageable although the JSDT contained other facilities that could have benefited the design.

The functionality of the scalable desktop was more of an open question as it depended largely on what could be implemented in the time available. The Jazz library [34] proved helpful in answering that question. Jazz is a library of classes specifically targeted at creating scalable (or zoomable, if you will) interfaces, appropriately named ZUIs, and by using that library, I could manage considerably more sophisticated designs than if I had had to implement everything
bottom up. Jazz provides primitives for constructing and managing a scenegraph [8] which is a data structure for organizing objects in a 2D or 3D environment and it implements a fast scaling transform for graphical objects. Furthermore, it can be used with the standard Swing document and bitmap classes and all in all, this seemed to promise a shortened development phase by allowing me to use component technology for both the scalable desktop itself and the artifacts to be rendered on it. In keeping with the conservative design choices made earlier, I decided to limit desktop functionality to the essential operations, namely translation\(^3\) of individual artifacts, scaling of the entire desktop and automatic layout of all artifacts. Later, after having implemented and tested these features, I realized the need for additional ones, but given the pop-up menu interface to the desktop functionality, it was straightforward to add more features by simply implementing the required functions and adding items to the menu. Specifically, it proved useful to be able to “pin” artifacts to the desktop, i.e. fix their positions so that they were not repositioned once a rearrange operation was carried out, so this and a handful of other features were added to the desktop menu before releasing the beta version.

To put these ideas of functionality into more concrete form, I wrote a series of use cases, i.e. simple narratives describing typical scenarios of use, as suggested by [33]. Writing use cases after having outlined functionality and interface was somewhat unorthodox as standard procedure is to begin by writing use cases, preferably in collaboration with the intended users of the system. However, this was not a standard development project in that the problem domain and audience were abstract rather than concrete, so it did not make sense to me to follow standard procedure. That is, in the absence of a concrete problem domain, one has to flesh out one’s ideas before use case narratives can be written about them. This could be either in the abstract, i.e. in the mind, or in more concrete form through design which is what I opted for in this case. Appendix A contains a use case diagram and brief textual descriptions of each case.

3.2.3 Class design

With both interface and functionality largely decided upon, I could outline the class structure of PeerView. The resultant class diagram is shown in Appendix A, and the following describes the rationale behind the design.

The design was kept as simple as possible to avoid complications and allow easy modifications at later stages. The classes that make up the design are divided into interface, entity, and control classes. The interface classes are those implementing the GUI, the entity classes are those that contain and manage the application data structures without performing any but minor operations on them, and control classes are those that mediate between model and interface by modifying the former and updating the latter in response to user interaction. This division is a very simple object pattern, but is nevertheless not adhered to with absolute rigour as flexibility of design had priority at all times. For example, the Constants hierarchy in Diagram A.2 is an appendage to the model classes in that it does not represent a data structure, but a repository of constants for use by one or more other classes. This allows constants to stored centrally, accessed through get/set method pairs, and appear where used in the source code as meaningfully phrased function calls rather than string literals. For more

\(^3\) Movement along the \(x\) or \(y\) axis.
on this, see the source code in Appendix C or Section 3.3. Also, most of the control code is placed in central control classes, namely ClientManager and ServerManager, and almost all the entity data in central entity classes, namely ClientData and ServerData. This was done because the interface design clearly indicated that most data structures would be either read from or written to by more than one interface class via the appropriate control class. There was therefore little need to complicate the design by having multiple entity/control pairs (one for each interface class in the extreme case).

The many classes that implement listener interfaces are event handlers. These were added as implementation progressed rather than at design time, for although event handling is the standard notification mechanism in Java and the need for handlers therefore was obvious, the exact number and interfaces were difficult to determine at this stage. An attempt to incorporate handlers in the early design was therefore apt to be inadequate. The decision to add event handlers only during implementation seemed unproblematic since handlers are highly standardized, and can be nested in other classes, so I deemed the risk of this causing unplanned structural design change to be minimal.

### 3.3 Implementation

Implementation was done over a 6 month period from the middle of May 2000 to November 2000 using the VisualAge for Java 3.0 IDE (Integrated Development Environment) [58]. As for implementation methodology, I decided to proceed iteratively and by small increments, i.e. frequent builds and application tests. This may seem an obvious approach, given that it is probably how most small scale projects are carried out, but there are alternatives. In [42], Mills et al. discusses cleanroom development where developers are forbidden to test and execute their code while writing it, and tests are based on statistical analysis. The method is directed towards projects with no or low fault tolerance, i.e. projects where the code produced must have a low error to KLOC (Kilo-Line-Of-Code) ratio, and is intended for use by teams of developers as it relies heavily on peer review. A different approach that seems more attuned to development of applications like PeerView is Extreme Programming [6] which supports the notion of small increments. It also advocates programming in pairs as a method of ensuring continuous peer review, so I could obviously not adopt extreme programming in toto. Instead, I considered its recommendations as validation of my personal experience from earlier projects which was that small increments and constant feedback stimulates motivation and helps reduce the error rate.

The order in which design features were added to the framework produced during the design phase was determined by how essential they were to the end user, or, to put it differently, how indispensable they were to the application. For example, it would be possible for an end user to make sensible use of PeerView if it provided nothing more than a client application with a scalable desktop and functionality for adding artifacts to it. It would not be possible if that user had only a working server application with a fully functional communications system but nothing else. This approach was chosen because it ensured that a usable, if amputated, system would always be in place for demonstration and preliminary releases if the need arose, and because it allowed realistic interface and functionality testing to take

---

4 A build is the compiling and linking of all components making up an application.

5 The interface classes created during GUI design and the skeleton code for entity and control classes.
place during development. The major features are discussed below in the order they were implemented. A major feature in this context is one whose implementation is non-trivial. An example of the opposite, a trivial feature, could be transfer of string literals from one list box to another, or reading the contents of a set of files.

### 3.3.1 The scalable desktop

The scalable desktop is implemented by the class `ClientPanorama`. The term “panorama” is used throughout the PeerView implementation and documentation rather than the phrase “scalable desktop” used during design, because I reasoned that the idea of a panorama window providing overview of a vista had more intuitive appeal to the average user than the more technical notion of a scalable desktop. `ClientPanorama` is derived from the Jazz library class `ZCanvas` which implements a simple scenegraph structure, a drawing surface and a viewport camera (shown in Figure 3.2).

![SceneGraph](image.png)

**Figure 3.2:** The `ZCanvas` scenegraph.

A scenegraph is a collection of hierarchically organized nodes that specify the structure of the scene that is rendered on a user’s display. The nodes can be either *leaf nodes* which have no descendants or inner nodes, so called *group nodes*, from which other group nodes or leaf nodes can descend. Operations applied to a group node affect not only that, but also its descendants which allows operations to be applied in a flexible manner to arbitrarily sized subsets of the scenegraph, including the entire scenegraph if operations are applied to the unique root node. Group nodes can be categorized by the types of operations they support and Jazz includes such group types as `ZTransformGroup`, `ZStickyGroup` and `ZInvisibleGroup` which can be used to translate, scale and rotate their descendants, to make them appear at a fixed magnification and to be rendered invisible in the scene, respectively. A separate type of node is the *camera node* which can be attached to the scenegraph at an arbitrary position and used to render an image of the corresponding scene at that position. This means that one can place multiple cameras in a scenegraph and have different scene renderings produced without making changes to the graph itself.

The `ZCanvas` scenegraph contains the minimum of objects required to render a panorama, namely a `ZRoot` object with one child, a `ZLayerGroup` object which groups the set of nodes descended from it. Jazz has several types of such group nodes, most of which allows one to apply an operation to all nodes descended from them by applying that operation to the group node. The `ZCamera` node represents a viewport onto the nodes in the scenegraph and the `ZDrawingSurface` the surface onto which the camera view is projected. Artifacts are added to a `ClientPanorama` object by inserting a `ZVisualLeaf` node under the `ZLayerGroup` of the `ClientPanorama` (or rather, its `ZCanvas` super class). A `ZVisualLeaf` is a leaf node in the scenegraph which has a visible component that can be rendered in the scenegraph’s `ZDrawingSurface`. An artifact is translated by adding a `ZTransformGroup` node above the
ZVisualLeaf corresponding to that artifact and then applying a translation operation to that ZTransformGroup. Before adding more features to the panorama class, I implemented the basic functionality of the AddFiles and RemoveFiles classes so that artifacts could be placed on and removed from the panorama. This was in keeping with the overall aim of adding functionality in the order of usefulness, but mostly a practical concern since it would be difficult to test operations on the panorama if there was no way of inserting objects into it.

Scaling

The entire panorama can be scaled, but not individual artifacts. The reason for not allowing scaling of a single artifact is not only the overarching decision to maintain simplicity of design in the initial version of PeerView, but also that such a feature could easily be counterproductive. If individual artifacts could be scaled, the resulting panorama would be a mosaic of objects at different positions along the z-axis. Translating an object would then become a potentially user-unfriendly operation since a translation by Δ of an object o at z-coordinate α along either the x or y axis is transformed to a translation by Δ in the ZDrawingSurface rendering that is visible to the user. The user would then experience differing responses to identical inputs as an indication to translate an object o at z-coordinate α by Δ would appear to move o further than if another object o’ at z-coordinate n * α, where n > 1, was translated by the same amount.

The scaling from magnification Z to the target magnification Z’ is done by interpolating between the two values over a period of length l, where l is user adjustable using the preferences Section of the PeerView client application. The interpolation follows a slow-in-slow-out trajectory and the display is updated after each step, so to the user a scaling operation appears as a smooth animation from Z to Z’ that begins and ends with a gradual slowdown rather than an abrupt halt.

Performance

When implementing the translation and scaling operations, it became clear that poor runtime performance would be very detrimental to the usability of PeerView. That is, if those core operations could not be executed smoothly and with little delay on the average system, then users were likely to be exasperated which is obviously not conducive to end user satisfaction. The default behaviour of the Jazz renderer is to render objects on the drawing surface at one of three levels of detail: low, medium or high. These settings of course reflect a trade-off between quality and speed of rendering. I was aware of this at the outset, i.e. when beginning implementation and so had expected that quality could simply be reduced during critical operations to increase the speed at which they were carried out. However, this was not sufficient, for although performance was satisfactory on my system which must today be considered an average configuration PC, it was too sluggish on other systems (such as thin/dumb terminals) that for example did not have dedicated terminal hardware for graphics display, but instead received images from a server through a local network. I therefore modified the renderer so that objects were rendered as filled grey rectangles when translation and scaling operations were carried out. This improved runtime performance significantly, but not enough for all systems so there are still minimum system requirements when using PeerView. However, given the hardware of the average user today, I do not expect this to be a serious problem.
Layout

The panorama should support automatic layout. This was decided during design since it is a feature known from other desktop GUIs that can help the user restore order in a cluttered display, cf. for example the “Arrange” item on the desktop pop-up menu of the MS Windows platform. Layout is performed by a layout manager which is a class that accepts a set of nodes and then arranges them using a layout algorithm that may be arbitrarily sophisticated. Currently, PeerView 1.0 provides only the GridPanoramaLayoutManager class which is derived from the abstract class ClientPanoramaLayoutManager that forms the root of the so far small layout manager class hierarchy. A GridPanoramaLayoutManager arranges a set of artifacts in a square grid pattern where spacing is determined by the largest of those artifacts. The user may specify the type of layout manager to be used from the preferences dialog of the client application, but as mentioned, only grid layout is currently available. The layout manager is called when the user inserts artifacts into the panorama and when she selects the “Arrange” item from the panorama pop-up menu. I chose the grid layout algorithm because it was simple to implement and the default layout in most other desktop GUIs. There is no inherent limitation to the level of complexity of the layout manager, though, and one could envision managers that laid out according to semantic criteria such as content, type (text, graphics, etc.) and size in addition of course to the wide variety of geometric layouts that can be conceived. So far, however, I have not found the time to implement any of these, but since a layout manager is simply a class derived from ClientPanoramaLayoutManager that adheres to a particular interface and accepts a set of nodes, it should be possible for other developers to add such managers to PeerView 1.0, if desired.

Additional features

After experimenting with the panorama, I found out that it would be useful to be able to resize artifacts and to fix their position, i.e. “pin” them to the panorama surface, so to speak. This would give the user an acceptable degree of freedom in customizing the display without increasing development time significantly. I therefore introduced a resize feature to the ClientPanorama class and a corresponding item to the artifact pop-up menu. The operation is carried out by dragging a corner of the artifact in question to the desired position and then left-clicking the mouse. The artifact retains its size until removed or resized again. The ability to fix an artifact at a given position was implemented by maintaining a lock for each artifact (implemented as a boolean data member) and then excluding locked artifacts from any layout operations. The lock can be toggled from the artifact pop-up menu which also allows the user to toggle the default lock setting for any new documents added to the panorama. Both the resize and the lock feature necessitated changes to some of the method definitions in the GridPanoramaLayoutManager class.

3.3.2 Message and progress bar

At the very bottom of the client application main window there is a small rectangular area which is used for displaying error and status messages and for indicating progress for operations of some duration. This area, called the message area or message bar, can be double-clicked to

---

6This method of spacing was introduced after implementing the resize feature, see Section 3.3.1
expand into a window containing a log of previously displayed messages. Implementation of the message area was mostly straightforward, but one aspect was non-trivial, namely how to make the progress bar update and display and at the same time perform the operation whose progress the bar was indicating. I tried first to have the code implementing the operation force the progress bar to update at each step of execution, but I learned from a newsgroup conversation that this approach reportedly was grossly inefficient. I decided to rely on this information and implemented the message area as recommended, namely by executing the code for a task whose progress was to be indicated in a separate thread. That code will then modify the progress bar which runs in the event-handling thread\(^7\) and is therefore automatically updated by the system when that thread is executed.

Because of Java’s platform independence, the issue of multi-threading is made somewhat more complicated than it appears from the above. Different operating systems implement different multi-tasking policies which affects the way in which threads are allotted processor resources. Notably, the UNIX and Solaris operating systems use preemptive scheduling where executing threads are only taken out of their running state when they yield control or are forced out because a thread with higher priority requests execution. The Windows NT operating system, however, uses time slicing as its multi-tasking control mechanism where processes such as threads are given a fixed size time-slice in which to execute. When that time-slice expires, the thread is put out of its running state and the next contender put into its running state instead, regardless of whether the first thread yielded control or not. In practice, this means that on systems that use preemptive scheduling, the threads started by PeerView may be preempted, i.e. put out of their running state, by unrelated processes that happen to have higher priority and be executing on the system in question at the same time. This will sometimes produce noticeable delays in updating the PeerView interface which the user will most likely experience as erratic changes in the length of the progress bar. The code for launching tasks in separate threads is in the code listing for the ClientApplication class in section C.2 (e.g. the inner class RemoveDistributedDocumentsFromPanorama).

3.3.3 Communications layer

Before the group management and discussion forum functionality could be implemented, the basic functionality for message and data exchange had to be in place. It is perhaps debatable whether this was in accordance with the decision to add features in the order of usefulness to the end user, but without means of transmitting group and message data it would be difficult to test the components responsible for those types of data. To create a communications layer, I had to add some functionality to the server application skeleton code produced during design. Specifically, the server is responsible for maintaining what the JSDT documentation refers to as a registry, i.e. a small database listing the clients and sessions registered with that server. The server also maintains a concurrently updated database of group information, messages and discussion fora, but most of that functionality was added later when both the group directory and the discussion forum components which generate that data had been implemented. Initially, only the methods for creating and destroying a registry and for opening channels of communication was implemented on the server side. On the client side, code was written for methods in the ClientManager class to enable it to connect to the server, to join

\(^7\)Java Swing events are always handled by the thread which started the application generating those events.
a session (which, cf. Section 3.2.2, corresponds to a group in PeerView group), and to send and receive data through communications channels.

3.3.4 Group directory

The group directory is shown in Figure 2.2. It provides a listing of available groups and functionality for joining a group, adding or editing one or deleting one from the listing. The interface had been constructed during the design phase so what remained was implementing the appropriate functionality and associating it with the push buttons at the bottom of the group directory dialog box. The add, edit and delete operations were “database primitives” in the sense of being the standard operations for simple interactive data structures which typically provide the ability to insert and delete elements as well as replace existing elements with new versions. Implementing these as operations on the local data structure, i.e. the one whose contents are displayed in the listing above the panel of buttons was therefore straightforward. However, after each operation, that local data structure is submitted to the server which maintains the master copy of the group listing. The server then updates the master to reflect the changes in the local structure and subsequently resubmits the master to all client currently connected to the server. The data structure is typically quite small so transmitting it from client to server in its entirety does not add significant overhead compared to just transmitting a notification of the operation performed and is simpler to implement and extend (new operations can be added without having to introduce a new type of notification, for example).

The “Join group” operation is more composite than the data management operations discussed above. To join a group, the following must be executed, assuming that a group was actually selected when the user opted to join:

1. If the user is currently member of another group, then:
   a. remove (the copies of) all other group members’ artifacts from the user’s panorama;
   b. notify the server that the user has left so that the master copy of the group directory can be updated to reflect the change in number of group members and total data volume;
   c. notify the other group members that the user has left so that they can remove (their copies of) the user’s artifacts from their panoramas.

2. Join the selected group and then
   a. broadcast copies of all the user’s artifacts to all other group members;
   b. request copies of all other group members’ artifacts and add them to the user’s panorama when received.

Minor details have been omitted from this definition; they appear from the source code in Appendix C. The operation was implemented so that the client leaving a group need only remove its copies of other group members’ artifacts and then actually leave the JSDT session corresponding to the group. The server will then be automatically notified since it monitors all channels of all sessions for leave and join operations. Upon notification, the server will
notify all other group members that the client has left the group and they will then update their panoramas accordingly. The server also changes the master copy of the group directory listing and broadcasts the updated copy to all active clients.

The group listing master copy maintained by the server is written to disk when the server exits and read again when it is next started.

### 3.3.5 Discussion forum

The discussion forum has two components: the topic tree which is a collapsible tree structure that organizes hierarchically a set of nested topics, and a message pane (also referred to as the message area when addressing the user since “pane” might not make intuitive sense), where the messages selected from the topic tree are displayed. Each time a user selects an item from the tree that corresponds to a message added by herself or another user, the client application requests that message from the server which maintains a database of all discussions and messages. The server responds to the request by sending a *message* object which is a simple data structure containing the message text and relevant meta-data such as author name, date and time of composition, etc.

The server database of messages and discussions is written to disk when the server exits and read again when it is next started. The storage format is plain text, which implies that compression might reduce the size significantly. I found it unlikely, however, that the size would become unmanageable given in realistic use scenarios if the data was stored in its uncompressed form and that therefore became the storage format for all such plain text files maintained by the server.

### 3.3.6 Artifact updating

To provide an accurate representation of a set of artifacts, the PeerView client must ensure that the objects placed in the panorama are updated at regular intervals. The decision to make the exact interval user adjustable was made during design as this affected the interface, but the specific method of updating was left undecided since I reasoned that choosing the best method would require some experimentation which required a code scaffolding, i.e. there had to be a panorama and artifacts for experiments on them to take place.

A naïve and failsafe method of updating artifacts can be formulated using pseudo-code:

1. for each artifact α
   1.1. if ((changed( α ) == true) AND (group( owner( α ) ) != null))
   1.1.1. broadcast( α, group( owner( α ) ) )
2. goto 1

The functions called in the above are assumed to be defined elsewhere. Function `changed( α )` returns `true` if the artifact `α` has been changed since the function was last called with `α` as argument. Function `owner( α )` returns a unique identification of the owner of the artifact `α`, and function `group( id )` returns a reference to the group to which the user corresponding to `id` belongs, or `null` if he or she does not belong to a group. The function `broadcast( α, g )` broadcasts a copy of the artifact `α` to every member of the group except `owner( α )`. `g`.

Since PeerView 1.0 supports both text and bitmap artifacts, this is a potentially very inefficient and costly method of updating as the size of a typical bitmap image at resolution...
1024x768x24 may be in excess of 3Mb, depending on the storage format. An alternative is to transmit only the difference between an artifact before and after a modification, as outlined below:

1. for each artifact \( \alpha \)
   1.1. if \(( \text{changed}( \alpha ) \text{ == true } \text{ AND } \text{group( owner( } \alpha ) \text{ ) } \neq \text{ null } )\)
       1.1.1. broadcast( diff( \alpha ), group( owner( \alpha ) ) )
   2. goto 1

The function \texttt{diff} computes and returns the difference between the artifact \( \alpha \) before and after the most recent change was made. Each client that received the difference data \( d \) would then execute the following:

1. \( \alpha = \texttt{combine}( \alpha, \texttt{diff}( \alpha ) ) \)

The function \texttt{combine} computes and returns the artifact \( \alpha' \) that results from merging the artifact \( \alpha \) with the difference returned by \texttt{diff}.

The runtime performance of this method can be enhanced by compressing the difference packages sent between clients, and it can be further improved by adjusting the quality (and thus time complexity) of compression to match the processing power to network bandwidth ratio of the system it is executed on. Such adjustment can be either automatic or user definable, depending on the implementation effort one is willing to invest.

Simple back-of-the-envelope calculations show that an implementation of the difference algorithm using for example the DEFLATE algorithm [16] for compression will outperform the naïve algorithm by an order of magnitude or more:

Notation:

- \( d \) the average size of a document in bytes
- \( s \) the average size in bytes of the difference between a document before and after a modification.
- \( u \) the number of update operations pr. sec.
- \( c \) the compression factor.
- \( N \) the naïve algorithm bandwidth consumption pr. sec.
- \( C \) the compressed algorithm bandwidth consumption pr. sec.

Assume:

\[
\begin{align*}
  d &= n \\
  s &= \frac{n}{10}
\end{align*}
\]

\[
\begin{align*}
  c &= 4 \\
  u &= 3
\end{align*}
\]

Then:
\[
N = d \cdot u = n \cdot 3 = 3n \\
C = \frac{s \cdot u}{c} = \frac{n}{10} \cdot 3 = 0.075n
\]

and if for example \(n = 10\)KB = 10 \(\times\) 1024 bytes = 10240 bytes then \(N = 30.720\) bytes and \(C = 768\) bytes.

Although these were loosely based calculations, they convinced me that experiments were unnecessary: the difference algorithm was the best choice, since its time/space tradeoff was best by a wide margin.

However, my first attempt at implementing a difference algorithm was hurried and not well enough thought through as a discussion with Jyrki Katajainen made clear. I had gone ahead with my initial idea for a difference algorithm which consisted in computing the difference between two byte arrays representing a document before and after a change by performing a logical XOR operation between the overlapping portions and appending whatever remained (which I surmised was likely to be negligible) to the difference which was also stored in an array. This difference array would then consist of long strings of 0’s and 1’s indicating areas of change and areas that were unaffected by the modifications made since the previous update. I then used the Java \texttt{Deflater} class \cite{JavaDeflater} to compress the difference array before it was broadcast to other clients who in turn used the Java \texttt{Inflator} class \cite{JavaInflator} to decompress the difference array. They then combined the difference array with their old version of the document it was associated with by performing the same XOR operation as was used to derive the difference. Because of the typical contents of the difference array (long sequences of 0’s and 1’s), the compression ratios were quite impressive and the entire arrangement seemed to simple and symmetric that I conveniently forgot to think actual usage scenarios through. The truth of the matter is that this XOR difference approach is only effective if the size of the array storing the document before the change is exactly the same as the size of the array storing the document after the change. Only then will the difference array contain only the difference between the two input arrays, punctuated by long sequences of identical symbols and thus yielding the favourable compression ratios compared to transmitting the entire document \footnote{Typical compression schemes will reduce a long sequence of identical symbols to a representation consisting of the symbol being repeated throughout the sequence and the number of times it is repeated, i.e. the sequence length.} was after. If, on the other hand, the change to a document resulted in a change to its size then the difference array would contain a sequence of identical symbols of length equal to the distance from the beginning of the document to the beginning of the changed portion. The remainder of the difference array, i.e. the portion corresponding to the beginning of the change in the document to the end of the document, would be an XOR combination of the document before and after it was changed, and because of its change in size, those two versions would be shifted relative to each other. If we assume that, on the average, a document will be changed at position \(\frac{n}{2}\) where \(n\) is the length of the document (and we ignore the size of the change itself), the difference array can only be compressed to about half the size that the original document could since it is, on average, half as “random”, to put it informally. An example might make this clearer:

\[\]
As the above suggests, the XOR difference array algorithm was far from optimal in that it only works satisfactorily if a change to a document does not result in a change to its size which is, of course, an overly restrictive limitation. It is, however, the algorithm used in the current implementation of PeerView as I have not had time to implement a better alternative. Some such alternatives are described in [31], where Hunt et al. survey different delta algorithms as algorithms for computing the difference between successive versions of a document are typically called in available research literature. They find that the traditional diff delta compressor found on the UNIX operating system is outperformed by more recent algorithms based on Ziv-Lempel techniques (such as the DEFLATE algorithm used in PeerView) and they present an alternative that they call vdelta which performs better, both in terms of compression ratios and in terms of compression/decompression times, than diff, diff combined with gzip compression and bdiff, a successor to diff.

The vdelta algorithm therefore looks like a promising candidate for use in PeerView. Indeed, the authors suggest that the algorithm is well suited for use in such applications [31, p. 10]. I have, however, not given implementation of an improved algorithm top priority since the current version functions satisfactorily: typical compression ratios are better than 1:15 (averaged over text and graphics) so on all but very large documents that are changed with short intervals, it performs quite well. If the system was to be developed further, however, the algorithm would have to be replaced since it consumes an unnecessary amount of bandwidth as the number of users increases.
Chapter 4

Experimental evaluation of PeerView

4.1 Introduction

Evaluating groupware is difficult because it combines the traditional problems involved in evaluating single user system with the complications introduced by scaling to multiple users [3, 45]. Among those complications are both practical concerns, such as recruiting and compensating users, planning and conducting experiments and scheduling them as well as methodological concerns such as how to best gather and use experimental data. This makes it difficult to formulate a suitable procedure for evaluating groupware. In fact, Baker et al. [3] claim that no cost-effective techniques to that end exist and that one is therefore consigned to adapting existing, single-user evaluation techniques to a groupware context.

I opted for a controlled experiment with a small group of participants who were asked to carry out a set of tasks using PeerView and subsequently to fill out a questionnaire and comment on their experience using the software. The reason for making this choice is discussed in further details in Section 4.3. The experiment itself was recorded on audio tape\textsuperscript{1} and combined with the written feedback from the three participants, it yielded substantial information on the usability of PeerView.

The below discussion is based on the materials used in the course of the experiment, all of which can be found in Appendix B.

4.2 The experiment

The experimental evaluation of PeerView was carried out on March 14th, 2001. The setting was a room in a Copenhagen computer café equipped with four medium-range PC systems running Microsoft Windows 98. The machines were fitted with 17" displays and Pentium II processors and were connected by a local area network. On one of the machines, the PeerView server software had been installed in advance since I did not consider the installation and configuration relevant to this end user experiment whose primary focus was to be on the

\textsuperscript{1}Circumstances did not allow for video recording, but as appears from the transcript (a written reproduction of the dialogue on the audio tape) in Appendix B.2, the participants' comments and questions were quite interesting in themselves and gave enough information to deduce many of their actions.
client side software. The other terminals contained only the software installed by the café proprietors, i.e. mostly communications software and computer games.

The three participants were a 27 year old woman (referred to as “Rikke” in the transcript) and two men aged 26 (“Thomas”) and 28 (“Paul”), respectively. They all had good working knowledge of how to use computers on a day-to-day basis, but had no super-user skills or prior experience with either groupware or zoomable interfaces. They were each assigned a PC and handed a task list (reproduced in Appendix B.2) which they were asked to read and proceed with. The tasks where linearly structured with each being dependent on its predecessor on the list, but required use of almost all the functionality in PeerView and therefore constituted a thorough walkthrough of the system. This setup was chosen because it allowed me to focus on the participants’ behaviour and performance rather than on explaining the intent behind a set of open tasks with no specification of how to perform them.

The first task on the list was to download and install the latest version of PeerView from its website [54]. At the time of the experiment, the latest version was 0.95, i.e. still a beta but close enough to a stable release (1.0) that I found it acceptable to conduct an experiment since I did not intend to add any major functionality nor expect to find any critical bugs\(^2\). The download process took approximately 10 minutes which was considerably longer than expected. The delay was caused by severe network lag and thus probably an accurate reflection of the fluctuations that users would experience under less controlled circumstances. Upon successful download and installation which took place without much difficulty, the participants began carrying out their tasks. One of the participants, the 28 year old man, had been given a task list which was slightly different from that given to the two others. It instructed him to create a group for the other two to join and to edit a shared document to determine whether changes were distributed once they had joined. Both he and the other two participants were later asked (by the task list) to contribute to the topic tree corresponding to the shared document he had added. The difference in task lists was necessary to strengthen the collaborative aspect of the experiment. By setting up a group and have all users join it, and then collaborate, albeit in a rudimentary fashion, on a shared document, the experiment came to encompass not only the single user dimensions of PeerView, but also its collaborative functionality.

The first task for all three, however, was to adjust the preferences so that their name and the correct server information was entered. They did this by modifying the settings in the preferences dialog box and then restarted the entire program to make the settings take effect. The last step was necessary because instant application\(^3\) had not yet been implemented. Afterwards, the participant assigned the task of creating a group carried that out while the other two added a number of documents to their respective panoramas. They then moved on to carry out the other tasks on their task lists as appears from the material in Appendix B.

4.3 Discussion

The PeerView usability experiment was limited in scope by having only three participants and a duration of only two hours including the time taken to fill out the questionnaires. In spite of

\(^2\) Program errors capable of triggering runtime failures such as system crashes.

\(^3\) Often signaled by an “Apply” button in such dialog boxes.
this, it yielded a fair amount of useful information and gave me reason to believe that increasing the duration and number of participants would probably have given a quantitatively, not a qualitatively different amount of information. I therefore trust the results to be substantial enough to safely draw conclusions about the usability of PeerView in its current form. The use of several techniques — controlled experiment, user observation, questionnaire and informal discussion after the experiment — was chosen to lend the findings more credibility. That particular set of techniques was decided upon because I deemed it well suited for gathering a diverse set of data and because it comprises the most widely used forms of evaluation among groupware researchers. In [45], Pinelle analyses the evaluation performed in 32 of 45 articles on groupware (the remainder did not contain formal evaluation), and finds that laboratory experiment is the most commonly used evaluation type, and that user observation, interview and questionnaire techniques, in that order of prevalence, are the most popular ways of carrying out the evaluation.

The experiment produced information on both the single user and collaborative usability of PeerView, as appears from the transcript (Appendix B.2). After the user⁴ assigned with the task of creating a group had done so, the other users joined the group and subsequently received copies of the artifacts that other users had placed in their panoramas. They were then each given the task of contributing to the topic tree associated with a specific document (referred to as “test__gruppe.txt” in the transcript) added by the user who created the group (the only document he was asked to put in his panorama). Towards the end of the experiment, the same user modified “test__gruppe.txt” and the other users checked their copies to see whether the change had been distributed correctly, i.e. whether their copies of that particular document had been updated to reflect the change. I had, of course, tested the update functionality separately, so the purpose was not so much to see whether it worked according to specifications, but rather to round off the collaborative component of the experiment.

### 4.3.1 Experimental findings

The experiment participants indicated or remarked repeatedly (e.g. transcript item 28,40,62 and questionnaires) that they had problems understanding the system dialogue or the consequences of the actions they performed. This was mostly due to the dialogue being in English, using unknown or counterintuitive terminology (such as referring to the nested topic structure in the bottom portion of the window as a “topic tree”) or to the interface being less informative than could be desired (as in the case where one participant asked how he could tell whether he had joined a group [transcript, item 46]). The participants did not use the help system although they were encouraged to do so which was probably due to me being present while the experiment took place and answering questions on how to interpret system messages and other trivial issues such as which keys to press to produce a tilde character).

There were a few concrete interface problems. One was that highlighted text appeared at a different scale from the highlighting itself [transcript, item 67], which made it seem as if a smaller portion of text was highlighted than was actually the case. This confusing phenomenon was referred to as an optical illusion by the participant who noticed it, and his subsequent attempts to align highlighting and text by scaling the document to 1:1 magnification further revealed that panorama scaling was perhaps overly sensitive to user input since he

---

⁴He is referred to as “Paul” in the transcript.
had difficulty controlling it. An unrelated problem was discovered by the woman participant who experienced difficulty with the visible documents box in the right hand portion of the toolbar panel. Her monitor operated at a lower resolution than the other monitors and as a consequence the visible documents drop-down box could not expand downwards. In effect, it “dropped up” rather than down by expanding in the upwards direction. This rendered its contents difficult to access as indicated in [transcript, item 42]. None of the participants seemed to notice or have use for the draggable resize dividers that separate the topic tree from the message area in the bottom of the display and the panorama from the discussion forum. At one point [transcript, item 54], I tried to draw the participants’ attention to the dividers to see whether they ignored them because they were unaware of the resize functionality or because they had no need for it. None of them used the dividers after I had made them aware of their purpose, but I doubt that can be taken as indication that they are unnecessary since different participants seemed to grapple a bit with fitting a document in the visible viewport or seeing the title of a topic in the topic tree. They might therefore have benefited from the resize functionality if they had been more comfortable with it. It would probably be desirable to somehow communicate more clearly that the dividers bars can be used for resizing, perhaps by annotating them with a label containing descriptive text.

A more abstract, general problem reported mainly in the comments section of the questionnaires was that the participants had difficulty understanding the context of their actions because it was unclear what audience PeerView was aimed at. This became clearer during the informal discussion afterwards when I explained the idea of making a prototype design before making a fully fledged program, but the participants nevertheless indicated that they felt somewhat bewildered during their work with PeerView. I suspect that this reaction is typical when users meet a program with a set of expectations shaped by work with commercial applications aimed at solving specific work problems. If development had gone through several iterations, each concluded by experimental evaluation, the problem could have been countered by using the same group of test participants each time since they would then be accustomed to the work process and its premises. The drawback of that approach is of course that such a group of seasoned test participants would quickly become unrepresentative of the user audience at large, thus probably degrading the value of any experimental findings derived from their activities.

The collaborative aspect of the experiment (as described in Section 4.2) was too limited for me to draw any firm conclusions, but it did seem to indicate that the participants had little awareness of each other. This applies to both group awareness (in the sense of being cognizant of the other participants) or workspace awareness (being aware of the others’ actions in the workspace) as indicated by [transcript, item 48, 74] where users display signs of not feeling part of a collaborative unit, seemingly in part because the PeerView interface does not make it sufficiently clear that they are. When addressing me, the participants occasionally made references to what they had overheard me saying to other participants which could be taken as suggesting that they would benefit from more direct communication facilities such as the “chat” that was planned for, but unfortunately not included in the PeerView 1.0 release due to time restrictions.
4.4 User evaluation

After the experiment was concluded, the participants and myself sat down to fill out the questionnaires handed to them and to discuss the experiment. The questionnaires were worded in a way so that they could be understood without specialized knowledge to minimize the potential for misunderstandings. They were divided into three Sections:

- A set of Likert scale propositions regarding specific usability issues. A Likert scale can be used to have informants indicate on a scale ranging from complete agreement to complete disagreement the degree to which they agree with a proposition.

- A semantic differential schema where the participants where asked to indicate on a bipolar scale the degree to which they felt a set of descriptive phrases applied to PeerView.

- A free form comments Section where participants where asked to write any comments or remarks they might have about PeerView.

The overall assessment of PeerView was favourable, but the comments indicated some of the usability problems discussed earlier, namely that the terminology used by the PeerView interface in its messages and widgets was at times difficult to understand, not only because of the language being foreign to the participants who were all Danish, but also because they used terms that were unknown to them. Another problem was that participants lacked a clear statement of whom PeerView catered to. One participant wrote that: “I have the impression that you would quickly grow accustomed to the programme and grow to like it in the proper setting” [questionnaires]. But the subsequent discussion revealed that, like the others, he had difficulty pinpointing exactly what that setting should be. I suggested that additional functionality, such as that to be included in InSiter, might make it more useful to them and their work colleagues as well as other groups, and they seemed to agree that enhanced awareness in the form of video mosaics of coworkers (for example in the style of PortHoles [17]), chat facilities and different workspace contents at different magnifications (semantic zooming) were worth introducing.
Chapter 5

Implications of PeerView 1.0 for future systems

5.1 Introduction

The PeerView 1.0 prototype is exactly that: a prototype for new ideas whose purpose is to demonstrate their feasibility and viability. It is intended as both a test vehicle, a useful tool and a learning experience on which to base design of future systems. In this Section, I will analyse \textit{a posteriori} the design, implementation and development of PeerView 1.0 to examine how a future rendering and review system (outlined in Chapter 6) can benefit from the findings of the PeerView project.

Of course, the below applies not only to successor systems, but also to PeerView 1.0 itself. The findings and conclusions might therefore conceivably be incorporated into the current version of PeerView to yield a significantly upgraded product which could aptly be called PeerView 2.0! However, I did not intend PeerView as an “open-ended” project and the experience from building version 1.0 is so comprehensive that I have found it most appropriate to use a different name, namely \textit{InSiter}, for the successor project. PeerView development may continue, though, since the project has been made open source. Time will tell.

5.2 Design analysis

5.2.1 Introduction

The discussion in the following Sections is mostly general, concerned with the guidelines for design that can be derived from PeerView. Details are only brought in where appropriate to this discussion as the purpose is to facilitate future development, not to formulate specific improvements to the current PeerView design.
5.2.2 Interface

In retrospect, it is clear that the design of the PeerView GUI rested on implicit assumptions about the suitability of metaphors as design devices. That is, by readily adopting the desktop metaphor (for the scalable desktop) and a debating society metaphor (for the discussion forum)\(^1\), I accepted the commonly held belief among GUI designers that design based on metaphors equals user friendly design [9, p. 5]. This assumption is not necessarily justified, as Blackwell documents in his Ph.D thesis. Based on comprehensive experiments where novice and expert users were asked to solve various exercises using metaphorical and non-metaphorical diagrams and notations, he concludes that [9, p. 164]:

“Most of the experiments described in this thesis ... have tested the assumption that diagrams are universally beneficial in problem solving and design. This is obviously not the case — even within the scope of these studies, a simple distinction between ‘experts’ and ‘novices’ has revealed not universal benefits, but a large difference between individuals with different amounts of experience.”

How, then, can one account for the inarguable success of the desktop metaphor as popularized by the Apple Macintosh and Microsoft Windows operating systems? Blackwell does not attribute this success to the metaphor as such, but argues instead that [9, p. 160]:

“The demonstrable advantages of graphical user interfaces, as with many types of diagram, can be explained in other terms. Direct manipulation — representing abstract computational entities as graphical objects that have a constant location on the screen until the user chooses to move them — facilitates reasoning about the interface by reducing the number of possible consequences of an action ...”

This seems to validate the choice of a desktop GUI design, although not by virtue of its metaphor design philosophy. I believe a similar argument applies to the discussion forum design as it relies on a form of direct manipulation that is both familiar to many users (for example from the graphical directory browsers found in many operating systems) and fits the description given by Blackwell. This seems to be corroborated by the predominantly positive assessment made in the questionnaires that the evaluation experiments completed (see Appendix B).

A useful lesson to be derived from the above is that one cannot ensure a sound design by relying exclusively on textbook recommendations and established dogma in the design community. Instead, the interface decided upon during the initial design phase could be translated into a working prototype or a mock-up, i.e. a “scaffolding” with no real functionality, which could then be subjected to usability testing on a group of end users. The empirical data thus gathered could then be used to modify, if necessary, the interface design, and the test cycle should then continue until the GUI design is deemed satisfactory. Of course, the initial design would rely on a set of implicit assumptions exactly as the PeerView design did. This seems inevitable since end users cannot, in general, be expected to describe a priori and in detail their ideal

\(^1\)I have no direct evidence to support that the original USENET design of a hierarchically organized newsgroup was inspired by debating societies or similar discursive fora, but my choice of that design was influenced by such considerations.
interface although most can, presumably, give constructive feedback on a concrete proposal presented to them. The designer should therefore ensure that his assumptions are grounded in empirical fact and well reasoned rationale, not only in commonly accepted practice and beliefs.

5.2.3 Functionality

It is clear that the actions that users can perform using PeerView constitute a minimum of functionality for a usable application. This is hardly surprising, considering that it is a prototype, but it does indicate that any successor system should not restrict itself to the ideas now explored if the scope of the research is to be increased. It should provide a fully fledged system that has the width and depth necessary to be useful in realistic work settings with large groups and large masses of data. The objective should not necessarily be to compete with similar, commercial systems, but to elaborate on the ideas embodied in PeerView, and demonstrate that they can "survive in the wild".

The bulk of user interaction in PeerView 1.0 is with the scalable desktop, i.e. the panorama on which artifacts are placed. The issues raised by adopting a desktop metaphor design were discussed in Section 5.2.2, and the rationale behind using a zoomable interface was presented in Section 3.1. The actions that users can perform on the panorama, i.e. its functionality, was limited to the essential operations as discussed in Section 3.1. Hence, there is ample room for improvement, both by extending the set of operations, although this would be merely a quantitative improvement, and by using a more sophisticated zoom mechanism which could make the entire interface qualitatively different. One such enhancement is semantic zooming, where the level of elevation (i.e. distance from the origin of the z axis) determines the representation used in the zoomable display. The reverse — that the choice of representation determines the elevation — has also been tried and is called goal-directed zoom in [59], where the authors describe a system that allows users to select an object and choose from a set of representations how it should be rendered. The system itself determines the appropriate form of representation at a given elevation by applying what the authors refer to as the Principle of Constant Information Density, which states that "the amount of visual information per display unit should remain constant as the user pans and zooms" [59, p. 306].

The Jazz library can be extended to support semantic zooming [8], so implementing such functionality is quite feasible using known technologies. I believe it could be of particular use in systems where the rendering was extended to arbitrarily large sets of artifacts and the process by which they were brought about. The reason is that it would allow a series of mutually supporting representations of group work to be rendered within the same display space and navigated in a convenient and familiar fashion. One example might be to have a project represented by a diagram at high elevations, by clusters of developers represented by photographs and brief biographical synopses at medium elevations and by grid arrangements of artifacts at low elevations, as done in PeerView. Chapter 6 elaborates further on this point.

Future rendering systems could of course abandon the idea of zoomable interfaces altogether and use entirely different modes of presentation. I believe, however, that ZUIs that support semantic zooming are a near ideal "2 1/2D" alternative to the conventional 2D and 3D displays, both of which have inherent limitations, as discussed in Section 3.1. Consequently, future systems would benefit from extending the PeerView 1.0 interface both qualitatively,
by introducing elevation dependent representation through semantic zooming, and, possibly, quantitatively by giving users a wider selection of options than that listed in the PeerView 1.0 artifact pop-up menu.

In terms of support for collaboration, PeerView 1.0 gives users simple workspace awareness, i.e. who the other members are and what they are doing (to artifacts) in the shared workspace, and has very rudimentary or no support for other forms of awareness, such as group-structural awareness (awareness of people’s roles in a group, their status and positions on issues) and social awareness (awareness of people’s emotional and intellectual states) [47, p. 2]. In [26], Gutwin and Greenberg find a strong positive correlation between the quality of support for workspace awareness in groupware applications and the performance and satisfaction of participants in experiments based on those applications. Their subjects used a radar view window which is referred as a “miniature” 2 to get a global perspective on the workspace and to follow the activities of other users. PeerView 1.0 naturally supports the former of these activities, but not the latter and that issue should be addressed by any successor systems. In fact, Gutwin and Greenberg write that [26, p. 517]:

“…the main finding of the study is that adding workspace awareness information to the miniature — visual indications of viewport location, cursor movement, and object movement — can significantly improve speed, efficiency, and satisfaction. These awareness components should be included in shared-workspace applications.”

This point seems to further underscore the qualities of a zoomable interface as compared to 2D and 3D alternatives which typically would require some dedicated miniature to be grafted onto the main interface to obtain the effect recommended by Gutwin and Greenberg. Zoomable interfaces allow an overview effect to be completely integrated in the ordinary operation of the interface, as PeerView demonstrates. Future systems could elaborate on that effect by integrating the features discussed in the above quote without subtracting from the interface’s ease of use, i.e. without inconveniencing the user.

5.3 Implementation analysis

5.3.1 Introduction

In this Section, I discuss those aspects of the PeerView implementation that carry useful lessons for implementation of InSiter and of system implementation in general. As with design, those lessons are mostly general and not specific to implementation details, but the discussion does bring in concrete examples where necessary.

5.3.2 Implementation language and technologies

As discussed in Section 3.1, the implementation of PeerView was facilitated by using readily available component technologies to shorten development time. Jazz [34] and JSDT [?] were the most visible such component packages, but also the Swing interface classes that are usually

2A scaled down representation of the workspace proper.
thought of and used as an integral part of the Java Development Kit were a component technology that helped make implementation by a single developer possible. I experienced little difficulty using these component technologies and ascribe this in large part to the fact that all were written in pure Java, i.e. without use of code specific to a particular platform or architecture, so called native code. Also, Jazz, JSDT and Swing all have well designed APIs that make interfacing them with one’s own code quite easy.

Judging from this experience, component-based development seems to be good practice but the following should be borne in mind:

- The interface of components should be well defined and well behaved, meaning that the API documentation should provide descriptions of both the syntax (how to use) and the semantics (what happens when used) of each component method along with a description of its side effects and complexity. If these are not available, the developer may realize that the savings in development time was too dearly bought with deficiencies in the end product. As Meyer notes in [41]: “...the quality of a component-based application is defined by the quality of its worst component.”

- The components libraries used should be compatible, preferably by being written in the implementation language of the application proper and by adhering to the same design principles. The problems that can be incurred by using code from separate languages are well known: one poignant example is the need to distinguish between different calling conventions for functions in C++ code where for example one must specify that a function should be called with the C language convention by prefixing the function definition with `extern "C"`. The problems that may arise from differences in design may be more subtle, but no less real. One example could be to make use of two component libraries, one of which was highly optimized for some time critical operation such as real-time rendering, the other designed with security concerns as the primary concern and performance as a distant second or third. Such grinding incongruity between design goals could result in components cancelling out each other’s respective benefits, making for a whole that is actually less than the sum of its parts.

- Component based development is not merely an question of picking off-the-shelf components, connecting them in some appropriate way and then executing the resultant mélange. The developer must negotiate between the ideal implementation of her design and the trade-offs involved in using component technology. In PeerView’s case, for example, the Jazz library was used to implement the zoomable interface, i.e. the scalable desktop, but there were alternatives that had a differently balanced design. OpenGL [44] was one such alternative, as indicated in Section 3.1, which has a much broader range of graphics operations than Jazz, but no specialized support for the 2D zooming operations needed to scale text and bitmap graphics at interactive rates on even low-end systems [7]. One could envision a more sophisticated system that used a combination of zooming and 3D navigation (for example by means of a scale space representation [20, p. 110]), and might therefore have to rely on a library such as OpenGL, which in turn would mean that the minimum system requirements for end users would have to be raised significantly.

Component-based development therefore affects the entire development cycle, not just class
design and implementation. Trade-offs have to be made throughout to find the best balance between the envisioned end product and the degrees of freedom afforded by available component libraries.

5.3.3 Representation of properties and messages

An application like PeerView typically contains a set of mutable and constant values that define its behaviour and interface. An example of the former are error messages most of which are typically immutable and triggered by specific (exceptional) conditions in the application. An example of the latter are preferences which determine the general appearance and functionality of an application and can usually be defined by the user.

It is a basic convention in good programming style not to use so called “magic numbers” [40, find side], i.e. constants that appear by their literal value in the code and not a symbolic name. This applies not only to numerical constants, of course, but also to string literals such as user messages. Consequently, all such literals are represented by symbolic names in PeerView. The literals themselves are stored in designated classes called ServerConstants and ClientConstants and are accessed through access methods, not directly, so a literal whose symbolic name was window size would be accessed through a function call to a method named getWindowSize. This makes it possible to concatenate the constant with one or more arguments to the get method before the resultant value is returned to the user. One example is:

```java
public final static String getREMOVING_DISTRIBUTED_DOCUMENT(String documentName, String senderName)
{
    return senderName + " has requested that " + documentName + " be removed from the panorama.";
}
```

Since some constants are shared by the client and server application, the ClientConstants and ServerConstants are derived from the class SharedConstants which contains those shared constants and the associated access methods. Both the server and client constants class contain a set of default preferences which are constants used when the user has not specified any value for a specific property.

This design is not new; using classes of static constants accessed through methods is well-known practice in Java programming, but by applying this technique consistently as in PeerView, the textual component of the user interface is separated from the implementation. This means that if a different version must be installed, either to correct an error (a “bug fix”) or to change the language, it can be done without requiring a new full install by the end user, or indeed a compilation by the developer. Users can instead download a small patch to the finished product and install that as instructed. This is a simpler method than the internationalization scheme described in [35] and could therefore be useful when the implementation should adhere to good style conventions, but not be bogged down in the details of internationalization.
5.3.4 Implementation of the distributed architecture

The distribution layer or data sharing mechanism of PeerView is implemented using the Java Shared Data Toolkit (JSDT) as discussed earlier. The JSDT provides a high level interface for use by developers of collaborative applications (among others) which is independent of the underlying communication protocol. That is, calls to the JSDT API do not specify or presume a particular protocol\(^3\) and therefore do not need to be changed if one protocol is replaced by another. As a result, one can use the JSDT with any protocol that allows data to be exchanged among a number of collaborating entities. In its current distribution (version 2.0), the JSDT supports a TCP or UDP sockets protocol which in theory could be used for both LAN, WAN and Internet traffic. However, in practice, many firewalls will block attempts to make a direct TCP connection between two machines and so the JSDT also supports an HTTP protocol which can “tunnel” through such firewalls provided the server side is properly configured. This is why PeerView allows the user to specify either protocol. The user must also specify the port to use as target when packets are routed through the network, and she must make sure that the firewall on both the server and client side allow traffic to pass through this port.

Sockets based traffic is a simple form of data exchange and for collaborative applications, IP multicast is a potentially better solution. Whereas unicasting data as done in TCP communication requires the sender to send separate packages to each receiver, a multicast protocol allows the sender to send just one package and then have it replicated by multicast routers on the network and forwarded to the intended recipients, possibly through several more replications. PeerView does not support multicasting in its present form because it is primarily intended for work environments where the data density, i.e. the amount of traffic generated, is moderate and not sufficient to “swamp” the network. However, an application that must scale to more demanding environments will probably need to use multicasting protocols as data traffic increases to achieve acceptable performance on all but dedicated networks.

As mentioned, JSDT can be made to work with any suitable protocol implementation, according to its documentation, but for multicast there is fortunately a way to minimize time consuming experimentation. It is, once again according to JSDT documentation, possible to couple the JSDT with the LRMP (Light weight Reliable Multicast Protocol) package from Inria [39] with no great difficulty. The LRMP implements a true push service meaning that clients need not be aware of the server’s name nor have a port number to connect to. Instead, they subscribe to an information channel and receive data when it is broadcast, so the distribution scheme resembles a radio channel rather than an telephone line as is the case with the sockets channels. In addition, the LRMP architecture is based on IP multicast with its associated advantages of supporting both one-to-one and one-to-many communication as well as making effective use of available bandwidth.

The primary reason PeerView was not implemented using the LRMP protocol was my reticence towards using a technology that I had no implementation (“hands-on”) experience with. Sockets based communication is the default protocol in JSDT and consequently the best documented. Also, I was familiar with the basics of sockets programming and so felt more comfortably using that technology, knowing there were deadlines to be met for the PeerView project. In retrospect, I am not sure this was the right decision because the advantages of the

---

\(^3\) The protocol may be configured via the JSDT API, but that does not detract from the orthogonality of the API as a whole.
LRMP (greater scalability, simpler user interface, more functionality) may have outweighed the cost in increased development time. It is, of course, difficult to assess since I did not experiment with the LRMP protocol and therefore have no experience to base an evaluation on. However, for any future project, InSiter in particular, a multicast implementation would be preferable to a sockets based implementation because the advantages of the former become so compelling as the scope of the application increases that it would be indefensible to use a simple unicast solution.

PeerView uses only a subset of the functionality in the JSDT library, namely the data distribution mechanism. The library also has support for data sharing which allows a group of clients to share a data reservoir (implemented as a byte array) and for implementing a security layer which enables controlled access to shared data and data distribution. This limited use of available functionality is not merely due to the limited scope of the PeerView application. I suspect that most collaborative applications, InSiter included, will have a greater need for a scalable, efficient, reliable data distribution layer than for auxiliary features such as data sharing, security and the like. The reason is that with a sound data distribution layer such as LRMP multicasting in place it is relatively straightforward to implement features such as security (for example by maintaining a registry of clients and their access privileges) and data sharing (by using a byte array and a simple token based access scheme) on top of that layer. If this is the case, then much of the JSDT functionality is unnecessary for the implementation of InSiter, although the JSDT is far simpler than the alternatives surveyed and mentioned in Section 3.1. Instead, the best solution may be a library aimed purely at data distribution and therefore optimized for that purpose. The Web Canal [39] tool from the French research institute Inria seems a good choice because it uses the LRMP protocol also from Inria and therefore has the desirable properties of that protocol as discussed earlier. Furthermore, Web Canal is freely available and pure Java so neither licensing nor compatibility concerns need hinder or limit integration into one’s own applications.

The experience gained from implementing the PeerView distribution layer is perhaps the most valuable implementation lesson to be derived from the entire project because the choice of data communication library has such a significant impact on both the internals of the implementation and the end user experience (by influencing network load, responsiveness and ease-of-use).
Chapter 6

InSiter

6.1 Introduction

In this Chapter, I present a tentative design of a system for rendering and review that builds on the experience garnered from the development and evaluation of PeerView. This system is considerably more ambitious than PeerView and is intended for practical, day-to-day use in a realistic setting, be it in academia or industry. The working title of this system is InSiter, a name chosen to signal the organization-wide scope of the rendering it presents to its users. In the following Sections, I describe the intentions and design philosophy of InSiter and present the interface design of its main components, and discuss in general terms how this could be implemented.

6.2 User audience

The intended users of InSiter are the same as for PeerView, namely first and foremost software developers, but as with PeerView, InSiter should be designed and implemented so that is accessible to most information workers (from office clerks to civil engineers) without extensive training. I believe this is not only feasible, but also desirable. Not just because it widens the potential user audience, but also because it forces design work to focus on the essentials of the product and how they are best communicated to all users. In Section 3.2.1, I described the principles of simplicity in connection with interface design and how they are propounded by industry and academia pundits. This credo of the simple and rejection of the overly elaborate\(^1\) recognizes the inherent limitations of the human cognitive system and, arguably, has some intrinsic aesthetic appeal (see also [55]). The InSiter design should therefore cater to its intended audience, i.e. software developers, but at the same time be as inclusive as possible.

6.3 Design objectives

PeerView was intended as a prototype to test the viability of the ideas of rendering and review, i.e. both the usefulness of a design philosophy based on those concepts and the feasibility of

\(^{1}\) facetiously but incisively referred to as the "keep it simple, stupid" or KISS principle by some.
translating them into reality. It did yield useful information on the latter aspect, but little on the first since it was, by necessity, limited in scope and therefore did not lend itself to the type of prolonged experimentation in realistic settings needed to make conclusive statements on usefulness.

InSiter should not be an experimental system — it is intended for “production use”, but it might still produce valuable experimental results by allowing the actual usefulness of rendering and review to come before a day, so to speak, by being used by real groups working on real projects in real organizations. The purpose is therefore to produce a workable (in the sense of being accessible and useful) and working (in the sense of being well designed and implemented) rendering and review system that builds on the experience from PeerView to provide users with a common information space and facilities for reviewing and discussing its contents.

The InSiter interface will be a ZUI (zoomable user interface) much in the same way as the PeerView interface, but it will employ a wider range of ZUI techniques because it must allow users to insert more types of artifact into its interface and allow them to apply a greater and more diverse set of operations to those artifacts than PeerView did. This is necessary to give users the freedom to customize the information space to suit their needs.

The PeerView interface was document-centric in that the only type of artifact that could be placed on its scalable desktop was documents, albeit their contents ranged from bitmaps to HTML code. The InSiter interface retains this emphasis on documents since such finite strings of symbols are after all the only artifact that information workers can produce, but it widens the scope to include the relationships between those artifacts, the people who create them and the organizational relationships between those people. By doing so, InSiter aims to render not only the outcome of a project but also the context it which it is brought about.

InSiter will allow users to form collaborative units (which will be referred to as groups for the time being) and will give them more extensive communication and review facilities than PeerView did. It will support a live video feed of each group member to be placed in the panorama so as to provide a video mosaic similar to that discussed in [17]. Further, InSiter will allow members to communicate directly on a member-to-member (unicasting) or member-to-group (broadcasting) basis by exchanging text messages which can contain hyperlinks to artifacts in the common information space and thus used as a simple means of initiating collaboration on a specific artifact. InSiter will give users the opportunity to discuss individual document artifacts by providing review facilities similar to what is found in PeerView. There seems to be little need to introduce new review facilities since the discussion forum concept in PeerView captures most of what transpires in a simple code review session which is what inspired it. Usability testings of early version of InSiter can probably reveal whether more sophisticated review facilities are desirable and so I leave those considerations to a later, more advanced stage of design.

InSiter will generate and store statistical information on document artifacts to facilitate analysis of them.
6.4 Interface design

6.4.1 Introduction

One of the most difficult design decisions is where to begin. After all, there is nothing to hinder design from beginning with how to ship the product to end users before describing how architecture and interface should be designed. However, the interface is a good place to start because it constitutes a concrete, visible commitment that can be understood by both end users and developers, and therefore sets a lower threshold for what subsequent layers of functionality and architecture must support.

Figures 6.1 to 6.6 are mock-up screen designs showing how the InSiter panorama will appear to the user. This is a simplified representation that captures the overall intent, meaning that details may differ in the final design. Based on the experience from PeerView and the experimental evaluation described in Chapter 4, I believe the exact nature of those details could favourably be decided on through repeated, i.e. iterative, evaluation of progressively refined design proposals. That way, the final design would likely be an accurate reflection of user needs, although the rules of thumb applied in the design of the PeerView interface (see section 3.2.1) can also help guide interface design.

However, for any evaluation to take place, there has to be some interface in place to begin with, and the below screen designs serve that purpose. Section 6.4.3 then describes the rationale behind these designs.
6.4.2 Screen designs

Figure 6.1: A mock-up of an InSiter panorama. Four author nodes are connected to a number of document nodes, all of which can be translated and scaled individually by the user. A variety of layout managers can be used to automatically lay out nodes so that they are arranged by author, by modification, by some geometric arrangement (as with the grid layout manager in PeerView), or by some other criterion entirely. The user can zoom manually or by double-clicking any node or edge.
Figure 6.2: Centered view of InSiter document. The user can double-click each of the three text items in the header bar to zoom quickly to a centered view of either one.
Figure 6.3: Close-up view of InSiter document. The user has manually zoomed to a magnification that allows her to see all three portions of the document header bar. From here, she can double-click any of them to have it centered automatically or continue the manual navigation.
Author information

Name: Jane Doe
Occupation: Junior programmer
Office: 115, East wing
Phone: 555-6789, extension 303
Email: jane.doe@codecorp.com
Home page: www.codecorp.com/~janedoe/

Calendar - March 2001

<table>
<thead>
<tr>
<th>M</th>
<th>T</th>
<th>W</th>
<th>T</th>
<th>F</th>
<th>S</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Work log

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Project and group affiliations

<table>
<thead>
<tr>
<th>From date</th>
<th>To date</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.4: Author information page. This portion of the InSiter document header bar contains information on the author of the document and a live video feed of her (if available).
Figure 6.5: Document information page. This portion of the InSiter document header bar contains summary information on the document and a revision diagram (lower left) that shows an outline of the document along with an indication of when each line was last modified (colour coded with red indicating recently modified and blue the opposite).
Figure 6.6: Document context page. This portion of the InSiter document header bar contains a user customizable representation of how the document relates to other artifacts. In this case, the document in question is a code file and its context is therefore appropriately represented by a UML class diagram.

6.4.3 Design rationale

InSiter aims to provide users with a comprehensive common information space as is reflected by the above screen designs. Section 6.5 describes the overall architecture that must support
the proposed designs, but the rationale behind the inclusion of such features as video feeds and code statistics merits separate attention.

Video broadcast

In [17], Lee et al. describe their experience with a system called Portholes “that allows distributed work groups to access information related to general and peripheral awareness” [17, p. 385] and report a mixed user reaction. However, they also find that user reticence was attributable to such factors as camera shyness, threat of surveillance, loss of privacy and lack of control, all of which they address with varying degrees of success by allowing users to edit, blur and remove frames from the video feed streaming from their camera. They also found that users request a "lookback" facility for determining who is accessing their video feed or has done so recently. In other words, users want to be aware of who is looking at them so that they do not feel passively monitored, but rather on par with whomever is observing them. User control and support for customization is therefore vital if the InSiter video broadcast system is to be successful. Portholes provides this through a preferences section for each user where he or she can regulate who has access to which images and how those images appear. I plan to incorporate a similar facility in InSiter, but the specifics of the design will depend on the type and quality of video used. However, users should in any case be able to edit out specific frames, i.e. block the image stream at will and at any point, and should be able to see at a glance who is currently looking at them (this reciprocal relationship between observer and the observed is sometimes referred to as “reciprocity”).

As for the technical feasibility of implementing a video broadcast system, consider the following back-of-the-envelope calculations:

Notation:

\[ f \] the size in bytes of each broadcast frame.

\[ r \] the frame rate measured in number of frames broadcast per second.

\[ c = \frac{1}{\text{compression factor}} \]

\[ b \] the bandwidth in bytes per second necessary to broadcast \( f \times r \times c \) bytes.

Assume:

- A resolution of 320 by 200 pixels and a pixel depth of 8 bits (sufficient for 256 colours or grey scale). Then \( f = 320 \times 200 \times 1 = 64.000 \).

- A reasonable frame rate of \( \frac{1}{3} \) the real-time frame rate. Then \( r = 8 \).

- A 1:10 compression ratio which is a conservative estimate, judging from [29].

Then \( c = \frac{1}{10} \).
Then:
\[ b = 64.000 \times 8 \times \frac{1}{10} = 51.200 \text{ bytes} = 50\text{KB} \]

Notice that this estimate is independent of the number of users in a group if IP multicast technology is used, as discussed in Section 6.5.2, since multicast allows the sender to broadcast to multiple recipients without multiple transmissions by instead making a single transmission to a group address which is then automatically forwarded to the members of the group in question. 50KB is a fraction of the bandwidth available in standard local area networks and within the range of many private subscribers with the increasing availability of ISDN and ADSL services. InSiter will need to distribute other forms of data, but video will probably account for a significant if not dominant portion of the total bandwidth consumption and so the above is also an estimate of the magnitude of the total InSiter bandwidth consumption.

**Code statistics**

In [52, p. 315 ff.], Eick describes the SeeSoft system which translates large amounts of code into a visual representation consisting of long vertical columns corresponding to the files containing the code in question. Each column can show the contours of a portion of code, i.e. how it would appear if viewed at a distance or, equivalently and more pertinent to InSiter, from a high elevation (that is, zoomed out so there is significant difference between the \( z \) coordinate of the viewport and that of the code). Each pixel line in a column then corresponded to a line of code and was colour coded to indicate its most recent date of revision. A similar arrangement is used in Figure 6.5 where an outline of the entire document is shown in the lower left corner. Each line is coloured so as to indicate when it was last changed and the user can double-click any portion of the outline which will result in InSiter zooming out to an appropriate viewing distance and scrolling to the portion of text corresponding to the clicked area. This feature will most likely only be enabled for documents containing computer source code since its usefulness lies in the fact that formatted source code has a distinctive signature outline that enables experienced programmers to quickly identify relevant segments of code by inspecting that outline [52, p. 318]. Also, to gather the necessary revision statistics, InSiter must either interface with the development tools used to create the documents in question, or make a line-by-line comparison of successive file versions each time a document is updated. The latter is a better solution since it does not depend on any interfacing with the constantly changing and largely proprietary development tools in use today. It does, however, carry with it a processing cost and should therefore only be done where it yields a useful outcome. The information thus gathered can also be used to implement a special inspection mode that temporarily modifies the apperance of document artifacts to give a SeeSoft-like overview representation. This inspection would expand each document so that all of its contents was visible without scrolling and then arrange the documents beside each other so that they made up a linear range of columns. By scaling to an appropriate level and perhaps activating an optional colour coding to indicate revision history, the user could then use the resulting arrangement of artifacts in the same manner as SeeSoft.

Again, computer source code files are the type of document for which such an overview representation would make most sense because of the relationship between code formatting and semantics (if-then statements and switch-case clauses in the \( C \) language are usually indented
in an easily identifiable fashion, for example). However, if the line-by-line comparison scheme suggested above was used, it would be possible to maintain revision statistics for other text document types as well. Future design refinements will have to address this question in detail.

The other type of statistical data envisaged in this design is the size evolution graph in the lower right portion of Figure 6.5. This is included because it gives a useful, if incomplete sense of how the document has evolved over time while at the same time indicating when it was created and what its current size without taking up significantly more space than those vital data alone would have taken. Also, it is trivial to implement since the size of each document will be monitored anyway in the process of maintaining updated copies of the artifacts placed in the InSiter panorama.

**Document context information**

In Figure 6.6, the example document used is shown in context using a UML diagram. This is an illustration of the general principle that when users add documents to the InSiter panorama, they can specify the relationships of those documents to other documents. They should be able to do this either by attaching a separate document containing the necessary information (such as a UML diagram in some portable format) or by entering the relationship textually or using a mouse. InSiter can then use this specification to render a document context pane as shown in Figure 6.6. This need not be a UML diagram, but can be any representation that can be rendered in the InSiter panorama such as a dependency graph connected to a document showing which other documents must be updated when that document is. InSiter could then be configured to automatically notify affected parties when a document is changed that other documents need to be maintained in response to the change. All of the above seem to pose no insurmountable technical problems: the specification of relationships can be entered using standard interface widgets, diagrams can be constructed using simple graphical primitives and responding to changes in documents is can be reduced to a question of traversing a dependency graph, if one exists and submitting messages to any users affected. I therefore leave the details of this design to a later stage.

### 6.5 Architecture

In [10], Booch, Rumbaugh and Jacobson (the so-called “three amigos”) define software architecture as:

> “An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization—these elements and their interfaces, their collaborations, and their composition.”

This abstract formulation captures what is probably a common intuition about software architecture, namely that it has to do with the relationships between the components that make
up a software system, and it does so in a way that stresses the engineering aspect of software architecture. But architecture can be interpreted differently, namely as the “style and method of design and construction of a complex system” [5], which is a definition closer to how conventional architecture can be perceived, namely as the consistent and reasoned application of design principles to achieve the most desirable balance between what is aesthetically pleasing, functionally required and economically feasible. I see the job of the software architect as much the same as that of the traditional architect, i.e. a person who design structures to be implemented by engineers and used by whomever contracted them or otherwise has use for them. The InSiter architecture is therefore designed to accomodate the needs of both parties by being:

- As simple as possible, but no simpler.
- Open and extensible.
- Flexible to allow for user customization and easy redesign at later stages.
- Robust, reliable, efficient and secure, in that order of significance.

### 6.5.1 Information architecture

I use the term “information architecture” to signify the choice and design of data structures to represent the information and interface components that the user can access through the InSiter interface. It is assumed that all data structures in the finished program correspond to interface components, although some may do so indirectly, perhaps by serving some auxiliary purpose in relation to another structure. The below only addresses the information architecture issues in InSiter that are either new, i.e. not addressed by PeerView, or of particular interest because they affect the entire design or pose some form of difficulty.

**The information space multigraph**

The most complex portion of the InSiter interface is the panorama where artifacts are placed and manipulated. The collection of artifacts visible in the panorama constitute a *multigraph*, i.e. a graph structure where pairs of vertices may be connected by multiple edges rather than a unique edge 2. The vertices correspond to either *author nodes* or *document nodes* in the InSiter panorama and the edges correspond to relationships between authors, between documents or between author and document(s) 3. Relationships may be either unilateral or bilateral and the direction of an edge between a pair of vertices indicates which apply in a given case. An example is given in Figure 6.7 where a set of author nodes are indicated by vertices marked *An* and a set of document nodes are indicated by vertices marked *dn* (with *n* being an integer).

---

2 A multigraph can be defined as a pair \( MG = (V, E) \) where \( V \neq \emptyset \) is a finite set of vertices and \( E \) is a set of two-element subsets of \( V \), where any \( v \in V \) may be in more than one \( e \in E \)

3 The terms vertex and node will be used interchangeable throughout this Chapter.
One interpretation of the graph in Figure 6.7 could be that author \(A1\)\(^4\) is the current maintainer of documents \(\{d1, d2, d3\}\) and that authors \(\{A2, A4\}\) are her co-workers. Similarly, author \(A3\) is responsible for documents \(\{d6, d7\}\), but she does not have a bilateral relationship to any of the other authors, and the unilateral relationship to author \(A2\) is most likely an indication that \(A2\) is her superior in the organization to which they all belong. The relationship between author and document is always unilateral and therefore represented by an edge directed from the author node to the document node. Documents, on the other hand, may be bilaterally or unilaterally related, as appears from Figure 6.7. A unilateral relationship may indicate a one-way dependency such as that between a C++ code file and a standard library, or it may indicate some other, user-specified relationship such as that between a legal text and its annotations or a template and the documents based on it. A bilateral relationship may indicate a two-way dependency such as what exists between a C++ definition file (typically suffixed by \'.cpp\') and the corresponding declaration file (suffixed by \'.h\'), or it may indicate an entirely different bilateral relation between documents.

The multigraph structure described above is an abstract representation of what can be found in the InSiter panorama. The user will not need to be aware of this structure when placing artifacts in the panorama, although she might be, but it will be the underlying structure behind the artifacts visible in the InSiter interface. The concrete meaning of the multigraph will depend on how it is annotated, i.e. what properties the individual vertices and edges are "decorated" with as users collaborate and add artifacts to the panorama. For instance, the graph in Figure 6.7 is an abstract representation of the panorama that results when 4 authors collaborate in a group on a collection of 10 documents in all, but it says nothing about either the rendering of nodes or the rendering of relations between them. Let us assume that author \(A1\) has added one text document containing HTML code and two bitmap graphics documents to the panorama, represented by nodes \(d1\) and \(d2, d3\), respectively. Let us further assume that

\(^4\)Rather, the author designated by the node labelled \(A1\), but I will dispense with such verbiage and simply refer to nodes as authors and edges as relationships where appropriate.
she has indicated that node $d_1$ is related to nodes $d_2, d_3$ by a dependency relation since the latter two are bitmaps that are referenced by the HTML code in the former. Let us also assume that when $A_1$ joined the group to which she currently belongs, she not only submitted her own profile to the group, but also stated her relationship with the other members, i.e. authors $A_2, A_3, A_4$ in a similar manner to how she indicated the relationship between documents. We then have two sets of information that can be used to determine the rendering of both documents and author nodes as well the relationships between them. This gives rise to an annotated multigraph which can be defined as a set of five sets $AMG = \{V, E, A, Av, Ae\}$, where $V \neq \emptyset$ is a set of vertices, $E$ a set of two-element subsets of $V$, $A$ is a set of annotations, $Av = \{(v, a) \in Ae : v \in V \wedge a \in A\}$ and $Ae = \{(e, a) : e \in E \wedge a \in A\}$. An annotated multigraph is a both structural and databased representation of the information space that an InSiter panorama constitutes and therefore seems an appropriate data structure on which to base its design. This argument is made stronger by the added benefits it brings with it, namely a large body of research on graph related issues and algorithms and much existing code for implementing such structures.

Of course, as with any graph structure, the abstract, set theoretic definition above says nothing about how it should be embedded in the plane, or, in the case of a zoomable display, in three-dimensional space. This means that there is no inherent correlation between the abstract multigraph representation and the placement of the corresponding artifacts in the panorama. An illustration of this is in the relationship between Figures 6.1, 6.8 and 6.7. Figure 6.1 shows how the InSiter panorama might appear (these are tentative design plans, so final versions may differ) after four users have joined a group and added artifacts to the panorama. The number of nodes and relationships between them (indicated by lines) correspond to the number of vertices and edges in Figure 6.7, but there is no deliberate correlation between the (arbitrary) layout in Figure 6.7 and the deliberate layout in Figure 6.1. Figure 6.8 shows the relation between the two other Figures more clearly. It is a diagrammatic representation of the panorama where attributes have been removed to indicate only the graph structure underlying the panorama contents. Neither Figure 6.1 nor Figure 6.8 contain edges between documents as this would obscure these particular illustrations, but the intention is that users should be able to regulate freely which types of nodes and relationships should be rendered in the panorama. The absence of edges indicating relationships between documents could therefore be interpreted as the user having requested that such edges not be drawn. Section 6.4 has more on this topic and also shows how document relationships are shown elsewhere in the panorama, namely in the “Document context” area of individual document nodes.
Figure 6.8: A diagrammatic representation of the panorama in Figure 6.1 to show its relationship to the multigraph in Figure 6.7. Note that the physical layout, which may have been applied manually by the user or automatically by a layout manager, has no deliberate relation to the layout of the graph in Figure 6.7 since a graph, by definition, has no inherent geometric embedding.

The multigraph structure will be represented by a modified version of the scenegraph structure that is used in the Jazz library as discussed in Section 3.3. Nodes in a Jazz scenegraph can be either leaves or inner nodes, both of which can be decorated with arbitrarily complex visual components which is the mechanism used for storing documents in PeerView. This corresponds to the annotations of the multigraph discussed above, and the relationships between nodes can be implemented by wrapping leaf nodes in a simple structure containing a collection of pointers to other leaves (or inner nodes) and tailoring algorithms that are either built from the ground up or taken from a third party library to work with this modified structure. Since the Jazz scenegraph supports many different types of operation, from translation, scaling and rotation to invisibility, hyperlinking and fisheye deformation, such a modified structure will also be endowed with a host of sophisticated graphics facilities at little or no extra cost in development time. The details of this design are not in place at the time of writing, but as the above suggests, it seems a both feasible and promising implementation model.
Data storage

InSiter will generate and need to store persistently (on disk) or transiently (in memory) the following types of data:

- **Review contributions** — these are plain text contributions by individual authors as well as the meta-data needed to identify and classify them (author name, date of creation and so forth).

- **Review discussion structure** — these are structural descriptions of the relationships between review contributions. They can be converted into a textual representation since contributions are identified by their meta-data.

- **Panorama structure** — the multigraph defining the contents of the InSiter panorama can be converted to a pure text form (“serialized”, in Java parlance) and hence easily stored in a text file.

- **Document artifact statistics** — as appears from Figure 6.5, InSiter will maintain statistical information on each document and since such statistics span the lifetime of the document, it will have to be committed to persistent storage between invocations of PeerView.

- **Author information** — this does not include the live video feeds of authors, but only encompasses their personal and other information as shown in Figure 6.4 which can all be converted to a pure text representation and thus easily saved to disk.

- **Group data** — the collaborative units that authors can form (referred to as groups at this early stage of design) are represented by textual data specifying name, description, participants and so forth, all of which can be stored as pure text.

As is suggested by the above description, I plan to store as much data as possible in a pure text form by using the Java serialization mechanism which allows data structures that implement a specific interface (the `Serializable` interface) to be converted to a textual representation and to subsequently be restored from one. By doing so, InSiter can commit all of the above data to disk using the standard stream facilities found in the JDK [50] when the application is terminated, and restore them from disk when it is subsequently restarted. This approach is used in PeerView which saves discussion fora, discussion contributions, group directories and preferences to disk as serialized representations which is a simple and compact representations that can be manually inspected using standard text processing tools. This might be useful if scouring the materials generated for a specific snippet of information such as a remark in a discussion contribution. InSiter will reap those benefits by using the text representation discussed above, and it can improve on PeerView’s storage method by using (optional) compression of files above a certain size which will of course make the text files inaccessible to simple manual inspection, but will minimize the InSiter disk “footprint”5 if so desired.

Further elaboration of the design might make it necessary to introduce new data structures, but I intend to make them serializable as well so as to not introduce exceptions to the above scheme. I expect little difficulty in doing so since any data structure can be described as a

---

5The amount of storage allocated.
collection of elements and relations between them\footnote{This may sound like a gross generalization, but consider for a moment that a “data structure” by definition is a collection of discrete symbols sets and a collection of relations (which can be defined as pairs of elements) between them, and the statement may seem more plausible.} which ought to be amenable to textual representation.

### 6.5.2 Distributed architecture

As described in Section 5.3.4, the Light-weight Reliable Multicast Protocol (LRMP) package from the French research institute INRIA seems a promising candidate for implementation of a multicast distribution architecture which in turn seems a good choice of such an architecture for collaborative systems. Indeed, in [37], Liao reports that the LRMP has been used in such applications previously and further describes the design goals of the LRMP library which have been to deliver a reliable protocol for bulk transfer of data, precisely what is needed in InSiter’s case. The LRMP is a transport protocol meaning that it implements the transport layer of a network protocol and therefore does not aim to support what is typically left to layers above the transport layer, namely authentication and session management services. This need not be a problem because the LRMP can be used in conjunction with the WebCanal [39] package, also from INRIA, which provides high-level primitives for creating and maintaining channels that can be used as communication conduits by the collaborative units. An alternative solution to using the LRMP protocol could be to build on top of the MulticastSocket class that is part of the standard JDK [50]. This class offers the bare minimum of services needed to send and receive multicast data, but can be extended using the standard Java extension mechanisms (composition and inheritance) so one could conceivably extend this class to support both managed groups and multiple channels in a manner similar to the JSDT. The advantage of that approach would be that one could customize freely the multicast protocol to suit the particular needs of InSiter which would most likely yield better performance than using a third-party solution aimed at generalized support for applications relying on multicast. Also, it would reduce the potential for complication incurred by interfacing technologies to an absolute minimum since extending a JDK class to fit into the InSiter framework would not constitute a technology interface, but simply a program component like any other.

It therefore seems entirely feasible to implement the InSiter architecture based on a multicast protocol as an implementation can be obtained over the Internet or built from the ground up using tested and well-known technology. Issues of security and reliability will have to be dealt with separately as neither the LRMP or the MulticastSocket class have built-in functionality to support those aspects. However, the Java JDK contains several classes for implementing security layers and encryption technology for Java is freely available over the Internet (see for example [13]), so fitting a distributed architecture with customized security and encryption layers seem no more complicated than implementing the multicast transport layer.

### 6.6 System requirements

InSiter is to be implemented in Java. This is a given if the Jazz library is to be used since interfacing with a different language brings with it its own set of difficulties. I do not expect performance difficulties to hamper usability or development since PeerView has demonstrated
that both the implementation of a zoomable interface and the distribution of data, even using a non-optimal delta algorithm (see Section 3.3.6), do not offer unsurmountable performance challenges. Nor do I expect the added layer of data incurred by the introduction of video transmission to degrade performance significantly as suggested by the calculation on page 61, since the frame rate can be adjusted according to available bandwidth and compression can be applied to further reduce transmission volumes. The feasibility of the former measure is documented in for example [17], and typical compression ratios of video data can be very good indeed as is demonstrated by the popular video formats AVI and MPEG. I therefore expect the minimum requirements for InSiter to be about the same as those for PeerView, i.e. an average PC system as found in many private homes and in many offices. Each machine should have a network connection in order to participate in groups, and if users wish to use the video facilities, their machines should be equipped with a camera which need not cost more than a few hundred dollars today, depending on resolution and colour range. These modest system requirements, coupled with the choice of Java as implementation language, should make InSiter available to a wide audience and help minimize the deployment difficulties that sometimes arise when new systems are shipped to users.
Chapter 7

Conclusion

7.1 Results and contributions

The work conducted in connection with this thesis has produced the PeerView software which has been the subject of two expositional articles, one quite short which appeared in the February 2001 issue of IEEE DSOnline [61] and another, somewhat longer article coauthored with Jyrki Katajainen which is included in this thesis in Chapter 2.4. The latter article has also appeared as a DIKU technical report [62] and is currently under review by an international journal. PeerView is listed in various directories of applications demonstrating groupware concepts and zoomable interface technology, and the visitor statistics I have gathered for the PeerView home page indicate that it has been frequently accessed by visitors from around the globe, many of whom seem to have downloaded the application.

In the objective terms of public exposure and amounts of code written, PeerView can be said to have fulfilled its purpose of being a prototype vehicle for artifact rendering and group review. Also, the experimental evaluation results and analysis contained in this thesis can likely help successor system(s) be developed more efficiently and to a higher standard than if they had had to be developed without any such empirical base. I believe much of this experience can be of interest also to developers of otherwise unrelated forms of groupware since Chapters 3 and 4 contain information on such matters as distribution technology, design criteria and evaluation techniques that may help developers make appropriate design and implementation choices. An outcome from PeerView that is of potential interest both to groupware developers and to developers building single-user applications is the experience using the Jazz and JSDT component technologies to shorten development time and the deliberations made in choosing them, as discussed in Chapter 3. The PeerView source code is in a sense the definitive reference to anyone who wants to study the details of how these third-party technologies have been used and it has therefore been made public for anyone to peruse.

PeerView has contributed to existing research literature on common information spaces by addressing recurring problems with largely untried techniques, such as using a zoomable display in response to the “detail/overview” problem as discussed in Section 2.2, and a threaded discussion forum for generating and organizing dialogue among users.
7.2 Lessons learned

On a personal level, PeerView has helped me develop my skills as a developer by being a non-trivial development project in territory that was largely untried for me when I started. It has also helped improve my research skills since it spans such a broad range of topics, ranging from theoretical CSCW research on common information spaces [4] to delta algorithms [31] and APIs [50, 34]. The development, research and writing processes have taught me new things about project logistics and confirmed what I already knew or suspected, namely that I tend to be overly optimistic when planning such projects and that as schedules slip, difficult and unpleasant choices have to be made by removing or scaling down planned features and lowering ambitions.

The methodology of building a prototype from an initial idea, experimentally evaluating the end product and performing an analysis of the development efforts to help guide future development proved fruitful in that it did yield useful information and did produce functional software that tested ideas and technologies in practice. Further, the PeerView software provides a “prototyping base” for similar systems since it allows developers to incorporate ideas into an existing framework, i.e. the PeerView source code, and subject them to experimental testing before making end-user versions of them.

This does not mean that the methodology described in Section 2.3 was followed with unswerving loyalty nor that it cannot be improved. For example, as appears from for example the discussion in Section 3.3.6 and 3.3.1, not all decisions were planned for or considered as thoroughly as could be desired. Further, not all phases and steps of the development process took place in the prescribed order. I believe the initial design could have been more stable if I had spent more time studying reports from similar initiatives such as [20] and [7] and drawn the appropriate lessons from the experience reported therein. An example of how this affected the process in practice is reported in Section 3.1 where I only became aware of research on the “detail/overview” display problem after I had carried through my own reasoning on the issue. The experimental evaluation described in Chapter 4 was not planned in detail until after implementation was concluded and the software made public, i.e. quite late in the work process, which resulted in it being limited to the number of participants I could recruit with short notice. In retrospect, I believe that experimental evaluation would ideally be an ongoing series of experiments that punctuated different stages of development and were planned as such, i.e. in detail from the beginning of the project. The first experiment could then be an assessment of the interface design in isolation based on the “mock-up” made during the early stages of design. The second experiment could be conducted when core functionality\(^1\) was implemented and subsequent experiments as additional functionality was added and official releases made. This would of course incur greater costs in terms of time taken to plan experiments, recruit participants and possibly compensating them economically, but the feedback into the development process would make this well worthwhile, I suspect. The techniques used for experimental evaluation worked well in this case and are widely used as documented in [45], but if developers lack the resources to regularly schedule such experiments, they could probably benefit from using a form of heuristic evaluation tailored for groupware [3] for evaluation at the early stages of development.

\(^1\)In PeerView’s case, this might adding and removing documents and manipulating the group directory.
7.3 Future work

If possible, I intend to direct future work at completing the design of InSiter which was begun in Chapter 6 through an iterative development process where successive design increments are implemented and the resulting intermediate systems subjected to evaluation which feeds back into the ongoing development process. This would, I believe, be a fruitful methodology for realising a system that could go beyond the PeerView basics to provide users with a useful, scalable and generically designed support system for collaborative work centered around a common information space. However, for this to move beyond wishful thinking there must be sufficient funding in place as InSiter is considerably beyond what can be comfortably be implemented by one or two developers in their spare time. Depending on investor sentiment, InSiter could then become a freely available system for a general user audience or a commercial, licensed system for specific customer groups. Either way, the development process in itself would probably be worth the effort since it can, as demonstrated in this thesis, be a useful asset *per se.*
Appendix A

Design documentation

A.1 Use cases

The below are detailed descriptions of all non-trivial use cases considered as the PeerView interface was being designed. Non-trivial use cases are those that involve multiple branches, iterative user interactions or are central to PeerView by being essential, i.e. indispensable, or likely to be used the most. An example of a non-trivial use case is “Add documents”. Trivial use cases are those that involve only simple editing of a set of controls or inspection of a set of values. An example of a trivial use case is “Set preferences”. Such use cases are, by and large, entirely predictable narratives and are therefore not included here. Another reason that this distinction is made is to avoid unnecessary elaboration in keeping with the XP (extreme programming) principle of simplicity [6, p. 37]. On a more subjective note, it is common sense to me that work should be done if it is called for by the task at hand, not to enforce ideological adherence to a particular methodology or notational device.

A.1.1 Add documents

Purpose

To add a set of local artifacts to the client application panorama.

Description

1. The client user activates the “Add Documents” dialog box using the relevant toolbar button or menu item.
2. The client application opens the “Add Documents” dialog box.
3. REPEAT until the user has selected all the files he or she wants to add
   3.1. The user navigates to a directory using standard controls in the dialog box.
   3.2. The user selects one or more files in the active directory.
   3.3. The user transfers files to the “Selected documents” box by clicking the appropriate button.
4. The client user clicks the OK button
5. The client application closes the dialog box.
6. FOR EACH selected document
6.1. IF the document is not already in the panorama THEN
6.1.1. The client application adds the document to the panorama
6.2. ELSE
6.2.1. The client application does not add the document to the panorama
7. IF any documents were already in the panorama THEN
7.1. The client application displays an error message.
7.2. The user acknowledges the error message by pressing its OK button.

Variation #1: Removing selected documents

3. REPEAT until the user has removed all the files he or she does not want to add
3.1. The user selects a set of documents from the “Selected documents” box
3.2. The removes the set by clicking the appropriate button.

Comments

The user may close the “Add documents” dialog box at any point by clicking the “Cancel” button.

A.1.2 Remove documents

Purpose

To remove a set of local artifacts from the panorama.

Description

1. The client user activates the “Remove documents” dialog box using the relevant toolbar button or menu item.
2. The client application opens the “Remove documents” dialog box.
3. REPEAT until the user has selected the documents he or she wants to remove
3.1. The user selects a set of documents from the list of documents in the panorama.
3.2. The user transfers the set to the list of documents to be removed.
4. The user clicks the OK button.
5. The client application removes the selected documents from the panorama.

Variation #1: Pruning the list of documents to be removed

3. REPEAT until the user has deleted from the list of documents to be removed those documents that should not be removed after all
3.1. The user selects a set of documents from the list of documents to be removed.
3.2. The user transfers the set to the list of documents in the panorama by clicking the appropriate button.
Comments

The user may close the “Remove documents” dialog box at any point by clicking the “Cancel” button.
Exit application
Set preferences
Customize layout

A.1.3 Open group directory dialog box

Purpose

To open the group directory dialog box

Description

1. The user activates the group directory dialog box by clicking the appropriate toolbar button or selecting the corresponding menu item.
2. The client application opens the group directory dialog box.

A.1.4 Update group directory

Purpose

To update the client side group directory so that its contents are synchronized with the master copy maintained by the server.

Description

1. The client application requests the group directory from the server.
2. The server receives the request and submits a copy of the group directory to the client.
3. The client receives the group directory copy and updates the group directory data structure it maintains.

Variation #1: Request failure

2. The server does not receive the request.
3. The client side connection times out after a fixed period of time.
4. The client application displays an error message in the message area/progress bar.

Variation #2: Submission failure

3. The client does not receive the group directory.
4. The server side connection times out after a fixed period of time.
5. The server application dumps an error message to the console.
Comment

This use case could be refined to handle error states better, but is kept at its current simple form to minimize future work on it. The necessary changes (better error handling) can be introduced at a later stage with little difficulty as they do not affect the structure of the use case.

A.1.5 Submit group directory

Purpose

To update the server side copy of the group directory and have the updated version broadcast to all affected clients.

Description

1. The client application submits a copy of the group directory to the server.
2. The server broadcasts a copy of the new group directory to all clients.
3. The client receives a copy of the new group directory and updates the group directory dialog box.

Variation #1: communication failure

2. The server does not receive a copy of the new group directory.
3. The client times out and notifies the user.

Variation #2: broadcast failure

3. The client does not receive a copy of the new group directory.
4. The server times out.

Comments

The handling of communication failures can be improved, for instance by having the client repeatedly attempting to recommit the group directory copy and have the server do the same for the broadcast of the group directory. However, this would be part of a more elaborate general communication exception handling scheme which seems a questionable investment of resources for a prototype application.

This use case is meant to performed as part of other use cases, but can of course be performed separately.
A.1.6 Create group

Purpose

To create a new group and have it inserted into the group directory.

Description

1. IF the group directory dialog box is not open THEN
1.1. PERFORM open group directory dialog box
2. The user clicks the “Create group” button.
3. The client application opens the “Create group” box.
4. REPEAT until the group has been given a name and description which do not already appear in combination in the group directory
   OR the user presses the “Cancel” button
4.1. The user edits the group name and description fields.
4.2. The user presses the “OK” button.
5. The client application closes the “Create group” dialog box.
6. The client application creates a new group and adds it to the group directory.
7. PERFORM submit group directory

A.1.7 Delete group

Purpose

To delete a group from the group directory listing.

Description

1. IF the group directory dialog box is not open THEN
1.1. PERFORM open group directory dialog box
2. The user clicks the “Delete group” button.
2.1. IF no group is selected in the group directory listing THEN
   2.1.1. The client application displays an error message.
   2.1.2. The user dismisses the error message by pressing its “OK” button.
2.1.3. GOTO end of use case
2.2. ELSE
   2.2.1. The client application displays a confirmation box requestion acknowledgement of the delete operation.
   2.2.2. IF the user acknowledges THEN
      2.2.2.1. The client application removes the selected group from the group directory.
      2.2.2.2. PERFORM submit group directory
   2.2.2. ELSE
      2.2.2.1. GOTO end of use case
A.1.8  Edit group

Purpose

To change the metadata of a group, i.e. its name and associated data.

Description

1. IF the group directory dialog box is not open THEN
1.1. PERFORM open group directory dialog box
2. The user clicks the “Edit group” button.
2.1. IF no group is selected in the group directory listing THEN
2.1.1. The client application displays an error message.
2.1.2. The user dismisses the error message by pressing its “OK” button.
2.1.3. GOTO end of use case
2.2. ELSE
2.2.1. The client application displays an edit box with write access to the group’s metadata
2.3. REPEAT until the group has been given a name and description which do not already appear in combination in the group directory OR the user presses the “Cancel” button
2.3.1. The user edits the box’s contents.
2.3.2. The user clicks the box’s OK button.
3. The client application replaces the old group metadata with the new.
4. PERFORM submit group directory.

A.1.9  Join group

Purpose

Description

A.1.10  Select document from visible documents box

Purpose

To select a document from the visible documents box and have it centered in the panorama.

Description

1. IF the visible documents box is not open THEN
1.1. PERFORM open visible documents box
2. The user selects a document name from the listing in the visible documents box
3. The client application makes the name the active selection in the box
4. PERFORM centre document
A.1.11  Zoom to overview

Purpose

To scale the panorama to a magnification where all documents in the panorama are visible and the viewport onto them is at the center of the panorama.

Description

1. The user double-clicks the panorama canvas, i.e. the area outside the documents.
2. The client application computes the magnification $M'$ that allows all artifacts (documents) to be visible in the panorama.
3. REPEAT until the panorama magnification equals $M'$
   3.1. change the panorama magnification by $\frac{M'-M}{n}$ where $n$ is user defined.
   3.2. display the panorama at its new magnification

A.1.12  Centre document

Purpose

To bring a document from an initial, uncentred position to a position where it occupies as much of the viewport as possible without distorting its form in any way.

Description

1. The user double-clicks one of the documents in the panorama.
2. The client application computes a position $P'$ and magnification $M'$ that the document must arrive at to be centered.
3. REPEAT until the position of the document equals $P'$ and its magnification equals $M'$
   3.1. change the document magnification by $\frac{M'-M}{n}$ where $n$ is user defined.
   3.2. change the document position by $\frac{P'-P}{n}$ where $n$ is user defined.
   3.3. display the document at its new magnification and position

A.2  Class diagrams

These are class diagrams of the classes that make up the PeerView 1.0 implementation. They have been divided into three separate diagrams to reflect their division into three Java packages. Those packages are: clientapp which contains the PeerView client application classes, serverapp which contains the server application classes, and peerviewmisc which contains classes actually shared by components in the other two as well as classes that currently are not, but are likely to be so at some later stage if development continues. Note that these diagrams reflect the current design, not the initial design which, as discussed in Section 3.2.3 did not contain event handlers and other minor classes that were unlikely to upset the design structure. Such classes were almost exclusively implemented as inner classes, and are not included in the below diagrams because I deemed that they would obfuscate rather than
clarify. Classes containing such inner classes are marked by a small square in their lower right corner and the source code corresponding to such classes can be consulted for further details. A similar argument applies to the choice of connection lines between classes: only inheritance relations (drawn as solid lines) and implements relations (drawn as dashed lines between classes and interfaces) are used.

A.3 Class diagram for package clientapp

![Class Diagram]

Figure A.1: Class diagram for the PeerView client application classes.
A.4 Class diagram for package serverapp

![Diagram of class relationships for the serverapp package]

Figure A.2: Class diagram for the PeerView server application classes.

A.5 Class diagram for package peerviewmisc

![Diagram of class relationships for the peerviewmisc package]

Figure A.3: Class diagram for shared and miscellaneous classes.
Appendix B

Evaluation experiment materials

The below Sections contain the following:

- Reproductions of the task list handed to each participant at the beginning of the experiment.
- A transcript of the audio recording of the evaluation experiment conducted on 14 March 2001.
- Reproductions of the questionnaires filled out afterwards by the participants.

All of the above are provided as copies of the original materials, which were written in Danish, as well as in an English translation for the reader’s convenience.
PeerView usability eksperiment – gennemgangsskema 1

PeerView er et prototype system beregnet til understøttelse af samarbejde mellem grupper. Systemet giver mulighed for at placere et antal dokumenter på en delt arbejdsflade og deltagte i grupper med andre som har samme mulighed. Derved præsenteres hver deltagar for et samlet billede af gruppens arbejdsområde.

Herunder er opstillet en arbejdsgang bestående af en række opgaver udført v.h.a. PeerView hvor meningen er at opgaverne skal gennemføres i den rækkefølge de optræder.

Når du udfører de enkelte opgaver vil jeg bede dig "tænke højt", altså sige (højt og tydeligt) hvis du får et indfald, har en kommentar eller en kritik på et tidspunkt i forløbet. Hvis du f.eks. synes at et vindue er uformadt uheldigt eller at systemet reagerer for langsomt, så sig det. Når alle opgaver er fuldført vil jeg udlevere et kort spørgeskema for at få din vurdering af det system du har arbejdet med.

Du kan til enhver tid åbne PeerViews hjælpsystem ved at vælge det fra "Help" menuen. Der kan du få hjælp til de forskellige bokse der vises undervejs i udførslen af opgaverne.

Arbejdsgang (gruppeopretter)

1. Start en netbrowser og hent PeerView client software fra denne web-adresse:
   http://www.worldonline.dk/~lars_vdc/peerview/
2. Installer PeerView softwaren på den maskine du arbejder ved.
3. Find PeerView i systemets start-menu og start programmet.
4. Åbn boksen "Preferences" ved at vælge det dertil hørende menupunkt eller knap.
5. Aktiver fanefladet "Server" og indtast "Ryger 4" i feltet "Server name". Feltet "Server port" skal der stå 4461 og i feltet "Connection type" skal der stå "Socket".
6. Aktiver fanefladet "Personal" og indtast dit navn samt en kort signatur som du ønsker skal tilføjes de indlæg i "debatten" du kommer til at have nogle. Signaturl feltet kan lades blankt hvis du ikke har lyst til at indtaste noget.
7. Tryk OK når du sikre dig at fanefladene har ovenstående indhold.
8. Luk PeerView ved at vælge "Exit" fra menuen eller trykke den tilsvarende knap.
10. Åbn gruppekataloget ved at vælge "Groups" fra "Setup" menuen.
11. Tilføj en gruppe ved navn "Test gruppe" ved at klikke "Create group" knappen nederst i gruppekatalogvinduet.
12. Slut dig til gruppen "Test gruppe" ved at markere den og trykke "Join group" knappen. Luk derefter boksen ved at trykke "Close".
13. Åbn "Add documents" boksen ved at vælge den fra PeerViews menu eller ved at klikke den tilsvarende knap på panelet øverst på skærmen.
15. Find det menupunkt eller knap der åbner oversigten over grupper og vælg det/den.
16. Åbn boksen "Documents in panorama" øverst i hovedvinduet og vælg dokumentet "test_gruppe.txt". Vent til dokumentet er centreret på skærmen og der er dukket et træ op i nederste del af hovedvinduet.
19. Skriv en kommentar til dokumentet og indsæt gerne evnt. tekst valgt i trin 17 ved at højre-klikke i det område hvor du skriver teksten og vælg “Paste text”.
20. Afslut boksen ved at vælge “Submit”.
21. Dobbelklik området udenfor dokumenterne i hovedvinduets midte således at der zoomes ud til et oversigtsniveau hvor du kan overskue alle dokumenterne.
24. Højre-klik på et tilfældigt dokument og vælg “Arrange all”.
25. Vælg igen dokumentet “test_gruppe.txt” fra boksen “Documents in panorama”.
26. Lav en ændring i filen “test_gruppe.txt” i det notepad vindue der er åbent på din maskine og gem filen. Se om ændringerne kommer frem i det dokument der er vist i PeerView.
27. Åbn boksen “Remove documents” og fjern alle dine dokumenter fra systemet.
   Brug hjælpesystemet hvis du har brug for det.
PeerView usability eksperiment – gennemgangsskema 1

PeerView er et prototype system beregnet til understøttelse af samarbejde mellem grupper. Systemet giver mulighed for at placere et antal dokumenter på en delt arbejdsflade og deltage i grupper med andre som har samme mulighed. Derved præsenteres hver deltager for et samlet billede af gruppens arbejdsmåde.

Herunder er opstillet en arbejdsgang bestående af en række opgaver udført v.h.a. PeerView hvor meningen er at opgaverne skal gennemføres i den rækkefølge de optræder.

Når du udfører de enkelte opgaver vil jeg bede dig "tænke højt", altså sige (højt og tydeligt) hvis du får et indfald, har en kommentar eller en kritik på et tidspunkt i forløbet. Hvis du f.eks. synes at et vindue er udformat uheldigt eller at systemet reagerer for langsomt, så sig det. Når alle opgaver er fuldført vil jeg udlevere et kort spørgeskema for at få din vurdering af det system du har arbejdet med.

Du kan til enhver tid åbne PeerViews hjælpsystem ved at vælge det fra "Help" menuen. Der kan du få hjælp til de forskellige bokse der vises undervejs i udførslen af opgaverne.

**Arbejdsgang (menigt gruppemedlem)**

1. Start en netbrowser og hent PeerView client softvaren fra denne web-adresse:  
   http://home.worldonline.dk/-lars_yde/peerview/
2. Installér PeerView softvaren på den maskine du arbejder ved.
3. Find PeerView i systemets start-menu og start programmet.
4. Åbn boksen "Preferences" ved at vælge det dertil hørende menupunkt eller knap.
5. Aktivér fanebladet “Server” og indstil "Ryger 4" i feltet "Server name". I feltet "Server port" skal der stå 4461 og i feltet "Connection type" skal der stå “Socket”.
6. Aktivér fanebladet “Personal” og indstil dit navn samt en kort signatur som du ønsker skal tilføjes de indlæg i "debatten" du kommer til at lave senere. Signatur feltet kan lades blankt hvis du ikke har lyst til at indtaste noget.
7. Tryk OK når du sikret dig at fanebladene har ovenstående indhold.
8. Luk PeerView ved at vælge “Exit” fra menuen eller tryk på den tilsvarende knap.
10. Åbn “Add documents” boksen ved at vælge den fra PeerViews menu eller ved at klikke den tilsvarende knap på panelet øverst på skærmen.
11. Tilføj et antal filer (mellem 10 og 100) ved at navigere gennem katalogerne på systemets diske, markér filer og flytte dem v.h.a. pileknapperne i boksen. Tryk OK når du har flyttet et tilstrækkeligt antal filer over i "Selected documents" boksen.
12. Find det menupunkt eller knap der åbner oversigten over grupper og vælg det/den.
14. Åbn boksen “Documents in panorama” øverst i hovedvinduet og vælg dokumentet “test_gruppe.txt”. Vent til dokumentet er centreret på skærmen og der er dukket et træ op i nederste del af hovedvinduet.

Figure B.3: Task list for group joiner, page 1


17. Skriv en kommentar til dokumentet og indsæt gerne evt. tekst valgt i trin 15 ved at højre-klikke i det område hvor du skriver teksten og vælg “Paste text”.

18. Afslut boksen ved at vælge “Submit”.

19. Dobbeltklik området udenfor dokumenterne i hovedvinduets midte således at der zoomes ud til et oversigtsniveau hvor du kan overskue alle dokumenterne.


22. Højre-klik på et tilfældigt dokument og vælg “Arrange all”.


Figure B.4: Task list for group joiner, page 2
B.1.2  English version of task list for rank-and-file members

PeerView usability experiment task list 1

PeerView is a prototype system for supporting group collaboration. The system allows its
users to place a number of documents in a shared workspace and participate in groups of
other users with the same capability. This presents each group participant with a global view
of the work undertaken by the group.

The below list contains a number of tasks which must be carried out in the order they appear
using the PeerView software. When carrying out the individual tasks, I would like you to
“think aloud”, that is, say out (loud and clear) what thoughts, comments or critical remarks
you might have during the course of events. If, for instance, you think a window has been
poorly designed or that the system reacts too slowly, then say so. When all tasks have been
completed, I will hand out a brief questionnaire to get your assessment of the system you
will have been working with.

At any time, you can open the PeerView online help system by selecting it from the “Help”
menu. This contains online help for all of the different boxes shown as the tasks are carried
out.

Task list (rank-and-file member)

1. Launch a web browser and download the PeerView client software from the below web
   address: http://home.worldonline.dk/lars_yde/peerview/.

2. Install the PeerView software on the machine you’re working with.

3. Find PeerView in the system “Start” menu and launch the programme.

4. Open the “Preferences” dialog by choosing the corresponding menu item or toolbar
   button.

5. Activate the “Server” tab page and enter “Ryger 4” in the “Server name” field. In the
   “Server port” field, you should enter 4461 and you should set the “Connection type” field
   to “Socket”.

6. Activate the “Personal” tab page and enter your name along with a brief signature that
   you would like to have added to the contributions to ongoing debates that you’ll be
   making later. The signature field may be left empty.

7. Click “OK” when the tab pages are set to the above values.

8. Close PeerView by choosing “Exit” from the drop-down menu or by clicking the corre-
   sponding toolbar button.

9. Start PeerView again using the procedure described in step 3.

10. Open the group directory by selecting the “Groups” item from the “Setup” menu.
11. Add a new group by the name “Test gruppe” by clicking the “Create group” button. Then close the box by clicking the “Close” button.

12. Open the “Add Documents” box by choosing it from the “PeerView” drop-down menu or by clicking the corresponding toolbar button.

13. Add the “test_gruppe.txt” from the “c:\temp” directory by moving it to the “Selected documents” box. Click “OK”

14. Open the “Documents in panorama” box at the top of the main window and choose the “test_gruppe.txt” document. Wait till the document has been centred on screen and a tree [structure] has appeared in the bottom portion of the main window.

15. This step is optional — you don’t have to complete it. Select a portion of the text in the [test_gruppe.txt] document that you would like to comment on. Right-click the mouse and choose “Copy text” from the pop-up menu that then appears.

16. Select an element from the tree in the lower left corner of the main window. Click the button that allows you to write a contribution (“Compose new message”). Use the online help system you have any difficulties.

17. Write a comment on the document and insert the text selected earlier, if any, by right clicking in the editable area and choosing “Paste text”.

18. Close the box by choosing “Submit”.

19. Double-click the area outside the documents in the center portion of the main window so that it zooms to overview magnification at which you’re able to survey all of the documents.

20. Right-click an arbitrary document and select “Move”. Move the document to a new position. Use the online help system if you need to. Right-click the document again and enable the “Lock” item if it isn’t already enabled.


22. Right-click an arbitrary document and choose “Arrange all”.

23. Select the “test_gruppe.txt” document from the “Documents in panorama” box.

24. Change any part of the “text_gruppe.txt” file in the Notepad window that has been opened on your Windows desktop and save the file. Check to see if the change has appeared in the document displayed by PeerView.

25. Open the “Remove documents” box and remove all of your documents from the system. Use the online help system if you need to.

26. Terminate PeerView by choosing “Exit” from the menu.
B.1.3 English version of task list for group creator

PeerView usability experiment task list 1

PeerView is a prototype system for supporting group collaboration. The system allows its users to place a number of documents in a shared workspace and participate in groups of other users with the same capability. This presents each group participant with a global view of the work undertaken by the group.

The below list contains a number of tasks which must be carried out in the order they appear using the PeerView software. When carrying out the individual tasks, I would like you to “think aloud”, that is, say out (loud and clear) what thoughts, comments or critical remarks you might have during the course of events. If, for instance, you think a window has been poorly designed or that the system reacts to slowly, then say so. When all tasks have been completed, I will hand out a brief questionnaire to get your assessment of the system you will have been working with.

At any time, you can open the PeerView online help system by selecting it from the “Help” menu. This contains online help for all of the different boxes shown as the tasks are carried out.

Task list (group creator)

1. Launch a webbrowser and download the PeerView client software from the below web address: http://home.worldonline.dk/ lars_yde/peerview/.

2. Install the PeerView software on the machine you’re working with.

3. Find PeerView in the system “Start” menu and launch the programme.

4. Open the “Preferences” dialog by choosing the corresponding menu item or toolbar button.

5. Activate the “Server” tab page and enter “Ryger 4” in the “Server name” field. In the “Server port” field, you should enter 4461 and you should set the “Connection type” field to “Socket”.

6. Activate the “Personal” tab page and enter your name along with a brief signature that you would like to have added to the contributions to ongoing debates that you’ll be making later. The signature field may be left empty.

7. Click “OK” when the tab pages are set to the above values.

8. Close PeerView by choosing “Exit” from the drop-down menu or by clicking the corresponding toolbar button.

9. Start PeerView again using the procedure described in step 3.

10. Open the “Add Documents” box by choosing it from the “PeerView” drop-down menu or by clicking the corresponding toolbar button.
11. Add a number of files (between 10 and 100) by navigating through the directories on the system disk drives, selecting files and moving them using the arrow buttons in the [Add documents] box. Click “OK” when you have moved a sufficient number of files to the “Selected documents” box.

12. Find the menu item or button that opens the group directory and select it.

13. Find the “Test gruppe” group, select it and join it by clicking the “Join group” button. If the group does not exist, then wait a while till it has been created (shouldn’t take more than a couple of minutes). Once you’ve joined the group, you can close the group directory by clicking “Close”.

14. Open the “Documents in panorama” box at the top of the main window and choose the “test_gruppe.txt” document. Wait till the document has been centred on screen and a tree [structure] has appeared in the bottom portion of the main window.

15. This step is optional — you don’t have to complete it. Select a portion of the text in the [test_gruppe.txt] document that you would like to comment on. Right-click the mouse and choose “Copy text” from the pop-up menu that then appears.

16. Select an element from the tree in the lower left corner of the main window. Click the button that allows you to write a contribution (“Compose new message”). Use the online help system you have any difficulties.

17. Write a comment on the document and insert the text selected earlier, if any, by right clicking in the editable area and choosing “Paste text”.

18. Close the box by choosing “Submit”.

19. Double-click the area outside the documents in the center portion of the main window so that it zooms to overview magnification at which you’re able to survey all of the documents.

20. Right-click an arbitrary document and select “Move”. Move the document to a new position. Use the online help system if you need to. Right-click the document again and enable the “Lock” item if it isn’t already enabled.


22. Right-click an arbitrary document and choose “Arrange all”.

23. Select the “test_gruppe.txt” document from the “Documents in panorama” box.

24. Change any part of the “text_gruppe.txt” file in the Notepad window that has been opened on your Windows desktop and save the file. Check to see if the change has appeared in the document displayed by PeerView.

25. Open the “Remove documents” box and remove all of your documents from the system. Use the online help system if you need to.

26. Terminate PeerView by choosing “Exit” from the menu.
B.2 Transscript

B.2.1 Danish version

1. Thomas: Når jeg nu skal downloade
2. Lars: Ja
3. Thomas: Hvad er det så præcist jeg skal trykke på?
4. Lars: Det er klienten du skal downloade
5. Thomas: Klienten?
6. Lars: Ja
8. Lars: Ja, men det afhænger så af systemet - i det her tilfælde skal i hente den med, så er vi sikker på det kan afvikles.
9. Lars: I skal bare downloade med
10. Rikke: Jeg skal bare trykke her [peger på skærmen]
11. Lars: Ja, tryk bare …
12. Thomas: Programmet kører direkte [InstallAnywhere installationsprogrammet]
13. Lars: Ja, det kører direkte [over netværksforbindelsen], ja
15. Lars: øhm, ja du kan bare køre det der …det tager så lidt tid.
   [ca. 10 minutters pause]
16. Lars: Proceduren her var så ikke så svær at gennemføre. Det der var det største problem var sidt jeg kunne se at indtaste adressen [url'en].
17. Rikke: Det er fordi tastaturet har en anden [opsætning (end den hun kender fra sit arbejde)] ...
18. Lars: [forklarer hvorfor InstallAnywhere er brugt til at lave installationsprogrammet]
20. Lars: Ja, i virkeligheden skulle folk så helst finde ud af det af sig selv. Jeg står selvfølgelig ikke og kan fortælle dem det når de skal downloade, så det skal helst gøres klarere. Client, det er ikke særlig …det kræver en teknisk indsigt at se at det er den klient der skal køres. Så det er faktisk en dårlig betegnelse.
21. Lars: Ja, det her Java …Jeg kunne have valgt at downloade det hele på forhånd, men det lod jeg være med for jeg syntes også i skulle igennem den her del af det.

22. Thomas: Skal jeg taste det der R-y-g-e-r-4 ind …det er jo tastet ind.

23. Lars: Jamen hvis det er det så er det fordi den husker indstillingerne fra dengang jeg testede [prøveinstallerede] det.

24. Thomas: da jeg trykkede OK, så gik den ud af programmet …Jeg flyttede nogle filer over [d.v.s mellem listboksene i “Add Documents” boksen], ikke

25. Lars: Ja,


27. Lars: BEMÆRKNING. Man må ikke tilfæje katalo ger i “Add Files” bokse.

28. Rikke: Hvad skal man …

29. Lars: Du skal først vælge dokumenter heroppe fra …ja det har du gjort.


32. Lars: Ja, den skulle gerne gå igang med at tilfæje ting igen nu.


35. Thomas: Jeg overførte nogle filer fra [available documents] boksen [til selected documents boksen].

36. Lars: Der var ingen katalo ger imellem ?

37. Thomas: Nej


40. Rikke: Kan jeg godt markere flere [filer] med CTRL eller skal jeg …

41. Lars: Nej, det kan du godt, ja

42. Rikke: [she is pointing to the visible documents box and complaining that it fall upwards. The exact words cannot be made out.]

44. Rikke: og så gør jeg ...

45. Lars: Ja, så vælger du og så må du så flytte vinduet op

46. Thomas: Nu har jeg valgt en gruppe - hvordan kan jeg se at jeg har joinet?


49. Lars: Hvis du går ud af det her [group directory boksen] vil du kunne se at i [visible documents boksen] der optræder du.


51. Rikke: Marker et, øh, det er bare en af de her mapper [i topic tree].

52. Lars: BEMÆRKNING: Når man resizer et dokument burde man måske have mulighed for at scrollle viewporten til siderne. Man kan kun resize et dokument i de dimensioner viewporten har når man begynder på det.

53. Lars: I kan resize de forskellige portioner af vinduet ved at hive i de her bjælker de er delt op, altså de her bjælker med den noprede overflade, det er nogen der kan hives i så man kan gøre dem større og mindre hvis man har lyst.

54. Paul: Det her tekstdokument, det er her ikke mere.


56. Lars: [Til Rikke] Nu gør den klar til at bevæge den [indholdet af panoramaet]

57. Rikke: Nå , men jeg tænkte på om jeg så skulle gå ind og ...altså når du har været inde og resize eller er det kun ndu har flyttet [et dokument].

58. Rikke: ...slutte af, ikke, når du har flyttet det.

59. Lars: Jo, det ...

60. Rikke: så skal jeg til at resize, ikke. Hvad gør jeg så [venstrelklikker på panoramaets baggrund]

62. Lars: Så prøv at flytte derhen hvor du vil have den. Så vent på at den kommer derud. Så klik venstre [museknap].


64. Rikke: På plads?

65. Lars: Ja

66. Thomas: Prøv at se ...[peger på markering af tekst i tekstdokument]

67. Lars: Prøv at dobbeltklikke ...det er fordi at, øh, det er fordi at den her markering her. Den har een ...den er på et niveau inde i skærmen og teksten er på et andet. Det er fordi den centrerer ikke med.

68. Thomas: Nej, okay.


70. Thomas: Nå, okay, så det er bare sådan et synsbedrag.

71. Lars: Synsbedrag, ja. Det er faktisk en meget god ting [at du opdagede det].


73. Rikke: Jeg kan slet ikke huske hvad det var. Jeg kan bare huske den så ikke ud som dengang jeg startede.

74. Lars: Okay, men, øh, det gør heller ikke så meget. Bare der var sket en ændring.

75. Rikke: Det var der.

76. Lars: Godt, det var det vigtige.


79. Lars: [Til Thomas] Det jeg skulle se var bare om de ændringer der var blevet lavet de også var blevet sendt rundt [Paul havde forladt gruppen inden Thomas kunne inspicere det delte dokument “test_gruppe.txt”].
B.2.2 English version

1. Thomas: When I’m about to download …
2. Lars: Yes?
3. Thomas: What is it then, specifically, that I should click?
4. Lars: You’re supposed to download the client.
5. Thomas: The client?
7. Thomas: That’s not described in here [in the task list] … should we download it with or without Java [The Java Virtual Machine].
8. Lars: Well, that then depends on the system - in this case you should get it with [the Java VM]. That way we’re sure it can execute.
9. Lars: You should just download including [the Java VM].
10. Rikke: I just click here [points to the display].
11. Lars: Yes, just click …
12. Thomas: The programme executes directly [The InstallAnywhere installation programme].
13. Lars: Yes, it executes directly [over the network connection], yes.
14. Thomas: It asks me where to put it [the installation programme] … run it or save it to …?
15. Lars: Ehm, yes, you can just run it … that’ll then take some time.
   [approximately 10 minutes intermission]
16. Lars: This [installation] procedure wasn’t that difficult to complete. What appeared to me to be the biggest problem was entering the URL.
17. Rikke: That’s because the keyboard is configured differently [to her workplace configuration].
18. Lars: [explains the reason for using InstallAnywhere for the installation programme].
19. Paul: I didn’t quite know which one of those [download options] to select … it was only because Thomas said [that the download should include the Java VM].
20. Lars: Yes, in real life people should then find that out by themselves. I can’t be there to tell them what to download, of course, so that should be made clearer. Client, that’s not very … it takes technical insight to see that that is the executable client programme. So that is actually a bad name for it.
21. Lars: Well, this entire Java thing … I could have downloaded it all in advance, but I didn’t because I thought you should go through that part as well.
22. Thomas: Should I enter the R-y-g-e-r-4 bit [the server name to enter in the preferences dialog box] - that’s already been entered.

23. Lars: Well, that must be because it remembers the settings from when I [test installed] the programme.

24. Thomas: When I pressed OK, it exited the programme ...I transferred some files [from one list box in the “Add Documents” dialog to another], right?

25. Lars: Yes.

26. Thomas: Then when I pressed OK it exited the programme.

27. Lars: COMMENT: Users should not be able to add directories to the “Selected documents” list box in the "Add Documents" dialog box.

28. Rikke: What are you supposed to ...

29. Lars: You must first select documents from up here [points to the “Visible documents” box] ...Yes, you already have.

30. Lars: But there’s no connection to the server yet. We’ll just close it [the application] and restart it. Since it hasn’t been able to ...it’s lost the connection. That sometimes happens.


32. Lars: Yes, it [the client application] should be adding [documents] to the panorama now.

33. Lars: It’s because [the server name] must be written with a capital [first letter] ...I think that’s the reason why ...yes.

34. Lars: Try going through the same procedure as you did the last time at this point. It was simply because it [the server name] had a lower case first letter, I think.

35. Thomas: I transferred some files from [the available documents] list box [to the selected documents list box].

36. Lars: There were no directories among [the selected items].

37. Thomas: No.

38. Lars: That was the reason [why the application had exited earlier] ...just carry on, then.


40. Rikke: Can I select multiple files using CTRL or must I ...

41. Lars: No, you can do that ...yes.

42. Rikke: [she is pointing to the visible documents box and complaining that it falls upwards. The exact words can’t be made out.]
43. Lars:COMMENT: At some resolutions the visible documents box becomes invisible because its orientation is upwards, so that should be fixed. It should be forced to drop down or its orientation should be user customizable. Something has to be done.

44. Rikke: And then I...

45. Lars: Yes, then you select [an item from the visible documents box] and then you have to move the window up [she has moved the window down so that she can select an item the visible documents box].

46. Thomas: I've selected a group now - how can I tell that I've joined?

47. Lars: That [the development of PeerView] hasn't come so far that you can see the individual members [of a group]. You can see the number of participants.

48. Thomas: [suggests it would be useful to be able to tell who are in a group].

49. Lars: Yes, but it [PeerView] is supposed to. At some point it is supposed to be able to produce a list of those who have participated, but it doesn't do that at the moment.

50. Lars: If you end this [the group directory dialog box] you can see that you appear in [the visible documents box].

51. Lars: COMMENT: There ought to be a list of participants rather than just their number. This could be done with a drop-down list of sorts, for instance in the group directory box.

52. Rikke: Select a, ehm, its just one of these folders [in the topic tree].

53. Lars: COMMENT: [After observing Rikke having problems with a resize operation]. When resizing a document, you should perhaps be able to scroll the viewport to either side. You can only resize a documents to fit the dimensions of the viewport at the time when you begin [the resize operation].

54. Lars: You can resize the different parts of the [PeerView client main] window by pulling the [divider] bars that separate them, that is, the bars with the granulated surface, they can be pulled so you can make them bigger and smaller if you like.

55. Paul: This text document [test_gruppe.txt], its not here anymore.

56. Lars: That's because you have to centre it. Now you're at overview [position]. You have to centre it.

57. Lars: [Speaking to Rikke]. Its [PeerView] preparing to move them [the panorama contents].

58. Rikke: Well, but I was wondering whether I should go in and ... I mean, when you've resized something or is it only when you've moved [a document]. [She is in doubt about how to carry out the resize and move operations].

59. Rikke: ... end [the operation], right ... when you've moved it?

60. Lars: Yes, that ...
61. Rikke: now I'm about to resize, right? What do I do next [left clicking on the panorama background].

62. Lars: That's because you're left clicking. You accidentally ended the resize [operation]. Try again.

63. Lars: Then try and move to where you want it. Then wait for it to move there [the resize to take effect]. Then click left [mouse button].

64. Lars: [Lock] simply means that it [the document] is locked. It will then stay there [at a fixed position].

65. Rikke: In place?

66. Lars: Yes.

67. Thomas: Take a look ...[points to highlighted text in a text document].

68. Lars: Try double-clicking ...that's because, ehm, that's because the highlighting here ...it has a ...its at one level of the display and the text is at another. That's because it [the highlighting] doesn't scale [with the text].

69. Thomas: Oh, okay.

70. Lars: Try highlighting it again. Move it further out. Stop. Back up. Move forward again. Stop. Release. Try moving a bit forward again. That's a bit difficult [Thomas has difficulty controlling the scaling operation - the panorama contents scale too fast]. What I wanted to show was just that it [the highlighting] only covers [the text] completely when it is at a very specific [level of magnification]. You'll get all the text, try scrolling down again, you'll get all of the [highlighted] text.

71. Thomas: Oh, like that, so it's just a kind of optical illusion.

72. Lars: Optical illusion, yes. Its actually quite a good thing [that you discovered that].

73. Lars: [Speaking to Rikke]. That gruppe.txt [the file test _ gruppe.txt] - do you remember that? Did you check [if it had been changed when you exited PeerView]?

74. Rikke: I can't at all remember what it [looked like]. I just remember that it didn't look the same as when I started.

75. Lars: Okay, but ehm, that's not really important. As long as there was a change.

76. Rikke: There was a change.

77. Lars: Good, that's what's important.

78. Lars: I should be able to see the same thing over here by Thomas. [Speaking to Thomas]. When you reach item 23 [in the task list], then let me see what has happened.

79. Lars: COMMENT: When you select some text — highlight it before copying it — the highlighting is not at the same level of magnification as the text. That produces a peculiar optical illusion. You copy the intended amount of text, all right, but it doesn't appear that way to the user. That's not very sensible.
80. Lars: [Speaking to Thomas]. What I wanted to see was just whether the changes that had been made had been distributed [Paul had left the group before Thomas could inspect the shared document “test_gruppe.txt” that Paul had changed, but Rikke had already confirmed that the change had been distributed and applied. Also, I had tested this feature independently on other occasions and saw no reason to repeat the latter steps of the experiment.]

B.3 Questionnaires

B.3.1 Danish version

The below are scanned reproductions of the original questionnaires as they were returned by the experiment participants. An English translation can be found in the next Section.
PeerView spørgeskema

Anvendelighedsspørgsmål

Her bedes du angive i hvor høj grad du er enig i nedenstående udsagn.

<table>
<thead>
<tr>
<th>Jeg fandt det let at installere PeerView.</th>
<th>Helt enig</th>
<th>Enig</th>
<th>Neutral</th>
<th>Uenig</th>
<th>Helt uenig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeg fandt grænsefladen let at arbejde med.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes systemet reagerede for langsomt på mine kommandoer.</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes udførmningen af grænsefladen (farver/form/givning) hjalp mig i mit arbejde.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Jeg synes PeerView var behageligt at arbejde med.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg havde problemer med at forstå PeerViews meddelelser til mig.</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg havde problemer med at bruge PeerViews funktioner.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Jeg synes hjælpe systemet var tilfredsstillende.</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes mulighederne for at flytte, forme og ordne dokumenter var tilstrekkelige.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Jeg synes træstrukturen nederst i hovedvinduet var svær at bruge.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes det var let at tilføj beskeder/emner til træet.</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Vurderingsspørgsmål

Her bedes du angive i hvor høj grad du synes nedenstående udtryk betegner PeerView.

<table>
<thead>
<tr>
<th>Nyttigt</th>
<th>I høj grad</th>
<th>I nogen grad</th>
<th>Ingen af delene</th>
<th>I nogen grad</th>
<th>I høj grad</th>
<th>Unnyttigt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let tilgengeligt</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Svært at bruge</td>
</tr>
<tr>
<td>Pænt</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Gråt</td>
</tr>
<tr>
<td>Sjævt at bruge</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Kedeligt at bruge</td>
</tr>
<tr>
<td>Hurtigt</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Langsomt</td>
</tr>
</tbody>
</table>
Kommentarer

PeerView er en prototype og tanken er at det på et senere tidspunkt skal videreudvikles til et mere ambitionet system. Hvis du har forslag, kommentarer eller kritik du mener vil kunne bidrage til dette produkt bedes du skrive dem herunder:

Jeg synes at der skal oplyses inden man påbegynder at bruge systemet, hvordan det kan bruges dit og hvilke situationer det vil gøre sig sædvanligt egnet til.

På den anden side vil der have stapt en nøgle for nye bruger, som måtte ikke have den store indgå i hvordan det hele gør ud på.

Dog skal det sige, at jeg godt er klar, at programmet henvender sig til en bestemt slags bruger, hvilke specifikit ved hvilket de skal bruge programmeret dit.
## PeerView spørgeskema

### Anvendelighedsspørgsmål

Her bedes du angive i hvor høj grad du er enig i nedenstående udsagn.

<table>
<thead>
<tr>
<th></th>
<th>Helt enig</th>
<th>Enig</th>
<th>Neutral</th>
<th>Uenig</th>
<th>Helt uenig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeg fandt det let at installere PeerView.</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg fandt grænsefladen let at arbejde med.</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes systemet reagerede for langsomt på mine kommandoer.</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes udformningen af grænsefladen (farver/formgivning) hjalp mig i mit arbejde.</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes PeerView var behageligt at arbejde med.</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg havde problemer med at forstå PeerViews meddelelser til mig.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg havde problemer med at bruge PeerViews funktioner.</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes hjælpesystemet var tilfredsstillende.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes mulighederne for at flytte, forme og ordne dokumenter var tilstrækkelige.</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes træstrukturen nederst i hovedvinduet var svær at bruge.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes det var let at tilføje beskeder/emner til træet.</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Vurderingsspørgsmål

Her bedes du angive i hvor høj grad du synes nedenstående udtryk betegner PeerView.

<table>
<thead>
<tr>
<th></th>
<th>1 høj grad</th>
<th>1 nogen grad</th>
<th>Ingen af delene</th>
<th>1 nogen grad</th>
<th>1 høj grad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nyttigt</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>Unyttigt</td>
</tr>
<tr>
<td>Let tilgængeligt</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>Svært at bruge</td>
</tr>
<tr>
<td>Pænt</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>Grimt</td>
</tr>
<tr>
<td>Sjovt at bruge</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Kedeligt at bruge</td>
</tr>
<tr>
<td>Hurtigt</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>Langsomt</td>
</tr>
</tbody>
</table>

**Ved ikke hvor jeg som person skulle bruge programmet til??**
Kommentarer

PeerView er en prototype og tanken er at det på et senere tidspunkt skal videreudvikles til et mere ambitiøst system. Hvis du har forslag, kommentarer eller kritik du mener vil kunne bidrage til dette produkt bedes du skrive dem herunder.

* De problemer jeg havde undervejs skyldtes til dels, at "tjære" stod på engelsk (betoningen "forsvandt" for hign).
* Jeg skulle bruge husen mede - gernejstasjerne "forsvandt"
PeerView spørgeskema

Anvendelighedsspørgsmål

Her bedes du angive i hvor høj grad du er enig i nedenstående udsagn.

<table>
<thead>
<tr>
<th>Jeg fandt det let at installere PeerView.</th>
<th>Helt enig</th>
<th>Enig</th>
<th>Neutral</th>
<th>Uenig</th>
<th>Helt uenig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeg fandt grænsefladen let at arbejde med.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes systemet reagerede for langsomt på mine kommandoer.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes udformningen af grænsefladen (farver/omgivning) hjalp mig i mit arbejde.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes PeerView var behageligt at arbejde med.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg havde problemer med at forstå PeerViews meddelelser til mig.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg havde problemer med at bruge PeerViews funktioner.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes hjælpsystemet var tilfredsstillende.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes mulighederne for at flytte, forme og ordne dokumenter var tilstrækkelige.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes træstrukturen nederst i hovedvinduet var svær at bruge.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeg synes det var let at tilføje beskeder/emner til træet.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Vurderingsspørgsmål

Her bedes du angive i hvor høj grad du synes nedenstående udtryk betegner PeerView.

<table>
<thead>
<tr>
<th></th>
<th>1 høj grad</th>
<th>I nogen grad</th>
<th>Ingen af delene</th>
<th>I nogen grad</th>
<th>1 høj grad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nyttigt</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Let tilgængeligt</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pant</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sjovt at bruge</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hurtigt</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unyttigt      Svært at bruge  Grint  Kedelig at bruge  Langsomt

Brugte den ikke

Figure B.9: Questionnaire 3, page 1
Kommentarer

PeerView er en prototype og tanken er at det på et senere tidspunkt skal videreudvikles til et mere ambitiøst system. Hvis du har forslag, kommentarer eller kritik du mener vil kunne bidrage til dette produkt bedes du skrive dem herunder:

Måske kan man installere prog. i windows og brug principippet til at udvikle problem løsninger etc.

Jeg har indtrykket af at man hurtig vil blive "DU" med programmet, og vil glad for det i det vrette miljø.

Jeg syntes også at de engelske ord er bet tekniske for en "normal" bruger.

[Signature]

Johan
### B.3.2 English version of PeerView questionnaire for first participant

#### Usability questions

Please indicate in the below table the degree to which you agree with the statements in the leftmost column.

<table>
<thead>
<tr>
<th></th>
<th>Agree entirely</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree entirely</th>
</tr>
</thead>
<tbody>
<tr>
<td>I found it easy to install PeerView</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found the interface easy to work with</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I thought the system reacted to slowly to my commands</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I thought the interface design (colours/layout) made my work easier</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found PeerView pleasant to work with</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I had difficulty understanding PeerView messages</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found the online help system satisfactory</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I thought the facilities for moving, resizing and arranging document satisfactory</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found the tree structure at the bottom of the main window easy to use</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found it easy to add messages/topics to the tree</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Evaluation questions

Please indicate in the below table the degree to which you think the phrases in the left and rightmost columns apply to PeerView.

<table>
<thead>
<tr>
<th></th>
<th>Very</th>
<th>Somewhat</th>
<th>Neither</th>
<th>Somewhat</th>
<th>Very</th>
</tr>
</thead>
<tbody>
<tr>
<td>Useful</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessible</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attractive</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fun to use</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quick to respond</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

107
Comments

PeerView is a prototype and the plan is that at some later stage it should be developed into a more ambitious system. If you have suggestions, comments or critical remarks you believe could contribute to this system, then please write them below:

I think the user should be made aware of what the programme is supposed to be used for and the types of situation it could used in before it is put to use. That way a novice user who may not have great insight into the purpose of it all will have a guideline. I do realize, though, that the programme is aimed at a particular type of user who will be well aware of what the programme is for.

B.3.3 English version of PeerView questionnaire for second participant

Usability questions

Please indicate in the below table the degree to which you agree with the statements in the leftmost column.

<table>
<thead>
<tr>
<th>Agree entirely</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
<th>Disagree entirely</th>
</tr>
</thead>
<tbody>
<tr>
<td>I found it easy to install PeerView</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found the interface easy to work with</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I thought the system reacted to slowly to my commands</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I thought the interface design (colours/layout) made my work easier</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found PeerView pleasant to work with</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I had difficulty understanding PeerView messages</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found the online help system satisfactory</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I thought the facilities for moving, resizing and arranging document satisfactory</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found the tree structure at the bottom of the main window easy to use</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>I found it easy to add messages/topics to the tree</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Very</td>
<td>Somewhat</td>
<td>Neither</td>
<td>Somewhat</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>----------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>Useful</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessible</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attractive</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fun to use</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quick to respond</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Comments**

PeerView is a prototype and the plan is that at some later stage it should be developed into a more ambitious system. If you have suggestions, comments or critical remarks you believe could contribute to this system, then please write them below:

- The problems I had were in part due to “things” being in English (the meaning “slipped” from me).

- I had to use the mouse a lot — the shortcut keys “disappeared”.

**B.3.4 English version of PeerView questionnaire for third participant**

**Usability questions**

Please indicate in the below table the degree to which you agree with the statements in the leftmost column.
<table>
<thead>
<tr>
<th>Agree entirely</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
<th>Disagree entirely</th>
</tr>
</thead>
<tbody>
<tr>
<td>I found it easy</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>to install PeerView</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>I found the interface easy to work with</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>I thought the system reacted to slowly</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>to my commands</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I thought the interface design (colours/layout) made my work easier</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>I found PeerView pleasant to work with</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>I had difficulty understanding PeerView messages</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>I found the online help system satisfactory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I thought the facilities for moving, resizing and arranging document satisfactory</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found the tree structure at the bottom of the main window easy to use</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>I found it easy to add messages/topics to the tree</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Evaluation questions**

Please indicate in the below table the degree to which you think the phrases in the left and rightmost columns apply to PeerView.

<table>
<thead>
<tr>
<th>Very</th>
<th>Somewhat</th>
<th>Neither</th>
<th>Somewhat</th>
<th>Very</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Useful</strong></td>
<td>X</td>
<td></td>
<td>Inaccessible</td>
<td>Unattractive</td>
</tr>
<tr>
<td><strong>Accessible</strong></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Attractive</strong></td>
<td>X</td>
<td></td>
<td>Dull to use</td>
<td></td>
</tr>
<tr>
<td><strong>Fun to use</strong></td>
<td>X</td>
<td></td>
<td>Slow to respond</td>
<td></td>
</tr>
<tr>
<td><strong>Quick to respond</strong></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Comments**

PeerView is a prototype and the plan is that at some later stage it should be developed into a more ambitious system. If you have suggestions, comments or critical remarks you believe could contribute to this system, then please write them below:

- Maybe you could install the programme under Windows and use the PeerView functionality to exchange problem solutions etc.
• I have the impression that you would quickly “become acquainted” with the programme and would grow to like it in the proper setting.

• I also think that the English words will seem fairly technical to the “ordinary” user.
Appendix C

Source code

C.1 Introduction

This Appendix contains the source code for PeerView. It is divided into three major Sections corresponding to the major packages of the PeerView application and is prefaced by a table of contents to help the reader find his or her way. The source code references packages that are not listed here, notably the Jazz and JSDT libraries, but all listings can be understood without knowledge of the source code for those packages. If, however, you would like to inspect said code or learn more of these libraries, the source code for Jazz is available at [34] and that for the JSDT at [7]. The PeerView source code is available at [54] as is the associated Javadoc documentation and the online help system listed in Appendix D.

C.2 List of listings
Listings

C.1  AddDocumentsAccessory.java .................................................. 114
C.2  AddFiles.java ................................................................. 119
C.3  ClientApplication.java ......................................................... 122
C.4  ClientInfo.java ............................................................... 166
C.5  ClientManager.java ........................................................... 171
C.6  ClientConstants.java .......................................................... 200
C.7  ClientPanorama.java .......................................................... 226
C.8  ClientPanoramaLayoutManager.java ................................. 252
C.9  ComposeNewMessage.java ...................................................... 255
C.10 DiscussionFrame.java ......................................................... 266
C.11 GridLayout.java ................................................................. 288
C.12 GridPanoramaLayoutManager.java .................................... 290
C.13 GroupDirectory.java ........................................................... 302
C.14 MessagePropertiesBox.java .............................................. 318
C.15 MessageWindow.java .......................................................... 323
C.16 PanoramaLayout.java .......................................................... 329
C.17 PeerViewClient.java ............................................................. 331
C.18 PreferencesDialog.java ......................................................... 332
C.19 PZCamera.java ................................................................. 351
C.20 PZCameraListener.java ......................................................... 352
C.21 PZoomHandler.java ............................................................. 352
C.22 PZPanEventHandler.java .................................................... 353
C.23 PZSwing.java ................................................................. 354
C.24 PZText.java ................................................................. 356
C.25 RemoveFiles.java .............................................................. 357
C.26 Validator.java ................................................................. 367
C.27 GroupManagementServer.java ....................................... 368
C.28 PeerViewServer.java ........................................................... 369
package clientapp;

/**
 * This class defines the right hand side interface components of the
 * "Add documents" dialog box, i.e. the "Selected documents" list box
 * and the two arrow buttons used for transferring files into and out
 * from it.
 * @author
 * @Creation date: (15-06-00 15:29:40)
 */

import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.awt.event.*;

public class AddDocumentsAccessory extends JPanel {

private JButton ivjAddButton = null;
private JPanel ivjButtonPanel = null;
private JList ivjDocumentList = null;
private JScrollPane ivjScrollPane = null;
private JButton ivjRemoveButton = null;
private javax.swing.DefaultListModel selectedDocuments;
private JLabel ivjJLabel = null;

Listing C.1: AddDocumentsAccessory.java
/**
 * AddDocumentsAccessory constructor comment.
 */
public AddDocumentsAccessory() {
    super();
    initialize();
}
/**
 * AddDocumentsAccessory constructor comment.
 * @param layout java.awt.LayoutManager
 */
public AddDocumentsAccessory(java.awt.LayoutManager layout) {
    super(layout);
}
/**
 * AddDocumentsAccessory constructor comment.
 * @param layout java.awt.LayoutManager
 * @param isDoubleBuffered boolean
 */
public AddDocumentsAccessory(java.awt.LayoutManager layout, boolean isDoubleBuffered) {
    super(layout, isDoubleBuffered);
}
/**
 * AddDocumentsAccessory constructor comment.
 * @param isDoubleBuffered boolean
 */
public AddDocumentsAccessory(boolean isDoubleBuffered) {
    super(isDoubleBuffered);
}
/**
 * Return the AddButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public javax.swing.JButton getAddButton() {
    if (ivjAddButton == null) {
        try {
            ivjAddButton = new javax.swing.JButton();
            ivjAddButton.setName("AddButton");
            ivjAddButton.setToolTipText("Add to List");
            ivjAddButton.setMnemonic('a');
            ivjAddButton.setText("");
            ivjAddButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/navigation/Forward24.gif")));
            ivjAddButton.setBorderPainted(false);
            ivjAddButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
        // user code begin (1)
        // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjAddButton;
}
/**
 * Return the ButtonPanel property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getButtonPanel() {
    if (ivjButtonPanel == null) {
        try {
            ivjButtonPanel = new javax.swing.JPanel();
            ivjButtonPanel.setName("ButtonPanel");
        } catch (Exception e) {
            // user code begin (1)
            // user code end
            throw new RuntimeException(e);
        }
    }
    return ivjButtonPanel;
}
ivjButtonPanel.setPreferredSize(new java.awt.Dimension(31, 66));
ivjButtonPanel.setLayout(new java.awt.GridBagLayout());
ivjButtonPanel.setMinimumSize(new java.awt.Dimension(31, 66));
java.awt.GridBagConstraints constraintsAddButton = new java.awt.GridBagConstraints();
constraintsAddButton.gridx = 0; constraintsAddButton.gridy = 0;
getButtonPanel().add(getAddButton(), constraintsAddButton);
java.awt.GridBagConstraints constraintsRemoveButton = new java.awt.GridBagConstraints();
constraintsRemoveButton.gridx = 0; constraintsRemoveButton.gridy = 1;
getButtonPanel().add(getRemoveButton(), constraintsRemoveButton);
// user code begin {1}
// user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}
return ivjButtonPanel;
}/**
 * Return the DocumentList property value.
 */
public java.awt.List getDocumentList() {
    if (ivjDocumentList == null) {
        try {
            ivjDocumentList = new java.awt.List();
            ivjDocumentList.setName("DocumentList");
            ivjDocumentList.setToolTipText("Documents displayed in panorama");
            ivjDocumentList.setBounds(0, 0, 112, 253);
            // user code begin {1}
            selectedDocuments = new DefaultListModel();
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjDocumentList;
}
/**
 * Return the JLabel property value.
 */
private java.awt.JLabel getJLabel() {
    if (ivjJLabel == null) {
        try {
            ivjJLabel = new java.awt.Label();
            ivjJLabel.setName("JLabel");
            ivjJLabel.setText("Selected documents:");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJLabel;
}
/**
 * Return the JScrollPane property value.
 * @return javax.swing.JScrollPane
 */
// WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JScrollPane getJScrollPane() {
if (ivjJScrollPane == null) {
try {
  ivjJScrollPane = new javax.swing.JScrollPane();
  ivjJScrollPane.setName("JScrollPane");
  getJScrollPane().setViewportView(getDocumentList());
  // user code begin {1}
  // user code end
} catch (java.lang.Throwable ivjExc) {
  // user code begin {2}
  // user code end
  handleException(ivjExc);
}
return ivjJScrollPane;
}
/**
 * Return the RemoveButton property value.
 * @return javax.swing.JButton
 */
// WARNING: THIS METHOD WILL BE REGENERATED. */
public javax.swing.JButton getRemoveButton() {
if (ivjRemoveButton == null) {
try {
  ivjRemoveButton = new javax.swing.JButton();
  ivjRemoveButton.setName("RemoveButton");
  ivjRemoveButton.setToolTipText("Remove,"+from,list");
  ivjRemoveButton.setMnemonic('r');
  ivjRemoveButton.setText("");
  ivjRemoveButton.setMaximumSize(new java.awt.Dimension(29, 35));
  ivjRemoveButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics /navigation/Back24.gif")));
  ivjRemoveButton.setBorderPainted(false);
  ivjRemoveButton.setPreferredSize(new java.awt.Dimension(29, 35));
  ivjRemoveButton.setMinimumSize(new java.awt.Dimension(29, 35));
  ivjRemoveButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
  // user code begin {1}
  // user code end
} catch (java.lang.Throwable ivjExc) {
  // user code begin {2}
  // user code end
  handleException(ivjExc);
}
return ivjRemoveButton;
}
/**
 * Insert the method’s description here.
 * Creation date: (15-06-00 16:07:02)
 * @return javax.swing.DefaultListModel
 */
public javax.swing.DefaultListModel getSelectedDocuments() {
return selectedDocuments;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {
/* Uncomment the following lines to print uncaught exceptions to stdout */

117
// System.out.println("-------- UNCAUGHT EXCEPTION --------");
// exception.printStackTrace(System.out);

/**
 * Initialize the class.
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
    try {
        // user code begin {1}
        // user code end
        setName("AddDocumentsAccessory");
        setLayout(new java.awt.GridBagLayout());
        setSize(300, 319);
        java.awt.GridBagConstraints constraintsButtonPanel = new java.awt.GridBagConstraints();
        constraintsButtonPanel.gridx = 0; constraintsButtonPanel.gridy = 1;
        constraintsButtonPanel.fill = java.awt.GridBagConstraints.VERTICAL;
        constraintsButtonPanel.anchor = java.awt.GridBagConstraints.WEST;
        constraintsButtonPanel.weightx = 1.0;
        constraintsButtonPanel.weighty = 1.0;
        add(getButtonPanel(), constraintsButtonPanel);
        java.awt.GridBagConstraints constraintsScrollPane = new java.awt.GridBagConstraints();
        constraintsScrollPane.gridx = 0; constraintsScrollPane.gridy = 1;
        constraintsScrollPane.fill = java.awt.GridBagConstraints.BOTH;
        constraintsScrollPane.anchor = java.awt.GridBagConstraints.EAST;
        constraintsScrollPane.weightx = 1.0;
        constraintsScrollPane.weighty = 1.0;
        constraintsScrollPane.insets = new java.awt.Insets(4, 33, 4, 4);
        add(getScrollPane(), constraintsScrollPane);
        java.awt.GridBagConstraints constraintsLabel = new java.awt.GridBagConstraints();
        constraintsLabel.gridx = 0; constraintsLabel.gridy = 0;
        constraintsLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
        constraintsLabel.anchor = java.awt.GridBagConstraints.NORTH;
        constraintsLabel.insets = new java.awt.Insets(6, 20, 6, 6);
        add(getLabel(), constraintsLabel);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    // user code begin {2}
    // user code end
    }

    /**
     * main entrypoint - starts the part when it is run as an application
     * @param args java.lang.String[]
     */
    public static void main(java.lang.String[] args) {
        try {
            javax.swing.JFrame frame = new javax.swing.JFrame();
            AddDocumentsAccessory addDocumentsAccessory;
            addDocumentsAccessory = new AddDocumentsAccessory();
            frame.getContentPane(addDocumentsAccessory);
            frame.setSize(addDocumentsAccessory.getSize());
            frame.addWindowListener(new java.awt.event.WindowAdapter() {
                public void windowClosing(java.awt.event.WindowEvent e) {
                    System.exit(0);
                }
            });
            frame.setVisible(true);
        } catch (Throwable exception) {
            System.err.println("Exception occurred in main() of javax.swing.JFrame");
            exception.printStackTrace(System.out);
        }
    } // user code end
package clientapp;

/**
 * The document used for adding files/documents to the panorama.
 * Creation date: (23-06-00 08:55:55)
 * @author:
 */
import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.awt.event.*;
import javax.swing.filechooser.FileFilter;
import java.io.File.*;

public class AddFiles extends JDialog {
    private JPanel ivjDialogContentPane = null;
    private PeerViewFileChooser ivjFileChooser = null;
    private ClientApplication clientApplication;

    /**
     * Constructor
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    public AddFiles() {
        super();
        initialize();
    }

    /**
     * AddFiles constructor comment.
     */
    public AddFiles(java.awt.Dialog owner) {
        super(owner);
        initialize();
    }

    /**
     * AddFiles constructor comment.
     */
    public AddFiles(java.awt.Dialog owner, String title) {
        super(owner, title);
    }

    /**
     * AddFiles constructor comment.
     */
    public AddFiles(java.awt.Dialog owner, String title, boolean modal) {
        super(owner, title, modal);
    }

Listing C.2: AddFiles.java

119
public AddFiles(java.awt.Dialog owner, boolean modal) {
    super(owner, modal);
}

public AddFiles(java.awt.Frame owner) {
    super(owner);
    initialize();
}

public AddFiles(java.awt.Frame owner, String title) {
    super(owner, title);
}

public AddFiles(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}

public void dispose() {
    getFileChooser().setPreviousFileFilterSelection( getFileChooser().getFilter() );
    super.dispose();
}

public ClientApplication getClientApplication() {
    return clientApplication;
}

public ClientApplication getClientApplication() {
    return clientApplication;
}

public boolean modal boolean
*/

public AddFiles(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}

public void dispose() {
    getFileChooser().setPreviousFileFilterSelection( getFileChooser().getFilter() );
    super.dispose();
}

public ClientApplication getClientApplication() {
    return clientApplication;
}

public ClientApplication getClientApplication() {
    return clientApplication;
}

/*
* Override the JDialog dispose method to allow the current file filter selection to be saved before
* disposing of the box.
* Creation date: (25-02-01 20:23:39)
* /

public void dispose() {
    getFileChooser().setPreviousFileFilterSelection( getFileChooser().getFilter() );
    super.dispose();
}

/*
* Insert the method's description here.
* Creation date: (16-06-00 08:34:47)
* 
* return clientapp.ClientApplication
*/

public ClientApplication getClientApplication() {
    return clientApplication;
}

/*
* Return the FileChooser property value.
* 
* return clientapp.PeerViewFileChooser
*/

/* WARNING: THIS METHOD WILL BE REGENERATED. */

private PeerViewFileChooser getFileChooser() {
    if (ivjFileChooser == null) {
        try {
            ivjFileChooser = new clientapp.PeerViewFileChooser();
            ivjFileChooser.setName("FileChooser");
        }
    }

120
ivjFileChooser.setMaximumSize(new java.awt.Dimension(400, 150));
jav.io.File ivjLocal41selectedFiles [] = {};
ivjFileChooser.setSelectedFiles(ivjLocal41selectedFiles);
ivjFileChooser.setPreferredSize(new java.awt.Dimension(400, 150));
ivjFileChooser.setMultiSelectionEnabled(true);
ivjFileChooser.setMinimumSize(new java.awt.Dimension(300, 150));
// user code begin {1}

// user code end
}
} catch (java.lang.Throwable ivjExc) {
// user code begin {2}
// user code end
handleException(ivjExc);
}
}
return ivjFileChooser;
}

/**
 * Return the JDialogContentPane property value.
 * @return javax.swing.JPanel
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJDialogContentPane() {
if (ivjJDialogContentPane == null) {
try {
    ivjJDialogContentPane = new javax.swing.JPanel();
    ivjJDialogContentPane.setName("JDialogContentPane");
    ivjJDialogContentPane.setPreferredSize(new java.awt.Dimension(600, 300));
    ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
    ivjJDialogContentPane.setMinimumSize(new java.awt.Dimension(400, 250));
    getJDialogContentPane().add(getFileChooser(), "Center");
    // user code begin {1}
    // user code end
}
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin {2}
// user code end
handleException(ivjExc);
}
}
return ivjJDialogContentPane;
}

/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {
  /* Uncomment the following lines to print uncaught exceptions to stdout */
  // System.out.println("-------- UNCAUGHT EXCEPTION --------");
  // exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
public void initialize() {
try {
    // user code begin {1}
    // user code end
    setDefaultCloseOperation(java.awt.WindowConstants.DISPOSE_ON_CLOSE);
    setResizable(true);
    setSize(600, 318);
    setModal(true);
    setTitle("Add/Documents");
    getContentPane().add(getFileChooser());
}

712 */
}
```java
} catch (java.lang.Throwable ivjExc) {
    handleException(ivjExc);
}

// user code begin {2}
ivjFileChooser.setAddFileDialog(this);
// default positioning puts the dialog at center of screen. The method used is a slight hack, but
// works fine.
setLocationRelativeTo( getContentPane() );
// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param java.lang.String[] args
 */
public static void main(java.lang.String[] args) {
    try {
        AddFiles aAddFiles;
        aAddFiles = new AddFiles((JFrame)null);
        aAddFiles.setModal(true);
        aAddFiles.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
                System.exit(0);
            }
        });
        aAddFiles.setVisible(true);
    } catch (Throwable exception) {
        System.err.println("Exception occurred in main()");
        exception.printStackTrace(System.out);
    }
}

/**
 * Insert the method's description here.
 * @param newClientApplication newClientApplication
 */
public void setClientApplication(ClientApplication newClientApplication) {
    clientApplication = newClientApplication;
}

/**
 * Overrides the JDialog show method to ensure that init is called before the box is displayed.
 * @param show()
 */

public void show()
{
    getFileChooser().init();
    super.show();
}

/**
 * Add the files selected by the user to the panorama and dispose of the dialog box.
 * @param submitAndClose()
 */

public void submitAndClose()
{
    Object selectedObjects[] = getFileChooser().getFileChooserAccessory().getSelectedDocuments().toArray();
    File selectedFiles[] = new File[selectedObjects.length];
    for (int i = 0; i < selectedFiles.length; i++)
    { selectedFiles[i] = (File)selectedObjects[i]; }
    clientApplication.getSceneManager().addDocuments(selectedFiles);
    clientApplication.repaint();
    dispose();
}

Listing C.3: ClientApplication.java

122
package clientapp;

/**
 * The main class of the PeerView client application.
 * Creation date: (14-06-00 15:36:49)
 * @author:
 */
import java.applet. *
import java.awt. *
import java.awt.event. *
import java.io. *
import java.util. *
import java.util.Properties;
import javax.swing. *
import javax.swing.border. *
import javax.swing.table. *
import javax.swing.event. *
import javax.swing.JProgressBar. *
import javax.swing.Timer. *
import edu.umd.cs.jazz. *
import edu.umd.cs.jazz.util. *
import edu.umd.cs.jazz.event. *
import edu.umd.cs.jazz.component. *
import javax.swing.tree. *
import com.sun.media.jsound. *
import peerviewmisc. *
import javax.help. *

/** The client application main class */
public class ClientApplication extends JFrame {

    /** Listener class for activating JavaHelp system */
    public class HelpActionListener implements ActionListener {
        HelpBroker helpBroker;

        public HelpActionListener( HelpBroker hb ) {
            helpBroker = hb;
        }
    } //** Event handler. Makes the online help window visible and centers it on screen */
    public void actionPerformed( ActionEvent ae ) {
        helpBroker.setDisplayed( true );
        // calculate placement of help window
        Point winLocation = getLocationOnScreen();
        Dimension winSize = getSize();
        int helpX = (int) (winLocation.getX() + (winSize.getWidth() - helpBroker.getSize().getWidth() ) / 2);
        int helpY = (int) (winLocation.getY() + (winSize.getHeight() - helpBroker.getSize().getHeight() ) / 2);
        helpBroker.setLocation( new Point( helpX, helpY ) );
    }

    /** Objects of this class can be used as caption item in the visible documents box in the client application toolbar */
    public static class CaptionItem {
        String caption;
        public String getCaption() {
        }
    }
```java
64     return caption;
66  
67      public CaptionItem( String captionArg )
68      {
69          caption = captionArg;
70      }
71  
72      public String toString()
73      {
74          return caption;
75      }
76  
77      }
78  
79      /** Objects of this class can be used as author items in the visible documents box in the client
80       application toolbar */
81      public static class AuthorItem
82      {
83          String authorName;
84          public String getAuthorName()
85          {
86              return authorName;
87          }
88          public AuthorItem( String authorNameArg )
89          {
90              authorName = authorNameArg;
91          }
92          public String toString()
93          {
94              return getAuthorName() + ":";
95          }
96      }
97  
98      /** Objects of this class can be used as document entries in the visible documents box in the client
99       application toolbar */
100     public static class DocumentItem
101     {
102         String documentID;
103         String documentName;
104  
105         public String getDocumentID()
106         {
107             return documentID;
108         }
109         public String getDocumentName()
110         {
111             return documentName;
112         }
113         public DocumentItem( String documentIDArg, String documentNameArg )
114         {
115             documentID = documentIDArg;
116             documentName = documentNameArg;
117         }
118         public String toString()
119         {
120             return "\t" + getDocumentName();
121         }
122     }
123  
124     /** Event handler for the visible documents box which is the dropdown box on the client application
125     toolbar where the * names of the documents displayed in the panorama appear
126     */
127     public class VisibleDocumentsBoxItemClickListener implements ItemListener
128     {
129         int previousSelectedIndex = 0;
130         
131         public void itemStateChanged( ItemEvent ie )
132         {
133             if ( ( isMouseListenerActive() == true ) &&

( ie.getStateChange() == ItemEvent.SELECTED || ( getDocumentBox().getSelectedIndex() == 0 ) )
{
    getDocumentBox().hidePopup();
    if ( ie.getItem() instanceof DocumentItem )
    {
        getPanoramaPanel().centerDocument( ( (DocumentItem) ie.getItem() ).getDocumentID() );
    }
    else if ( ie.getItem() instanceof AuthorItem )
    {
        int currentSelectedIndex = getDocumentBox().getSelectedIndex();
        // if stepping through the list from above, skip to the succeeding item
        if ( (currentSelectedIndex == previousSelectedIndex + 1) )
        {
            getDocumentBox().setSelectedIndex( currentSelectedIndex + 1 );
        }
        // if stepping through the list from below, skip to the preceding item
        else if ( currentSelectedIndex == previousSelectedIndex - 1 )
        {
            getDocumentBox().setSelectedIndex( currentSelectedIndex - 1 );
        }
        // if moving from non-adjacent item, negate selection
        else
        {
            getDocumentBox().setSelectedIndex( previousSelectedIndex );
        }
    }
}
}
/** Custom renderer for the visible documents box. The box is filled with items of classes
 * {Mlink #CaptionItem CaptionItem}, {Mlink #AuthorItem AuthorItem}, {Mlink #DocumentItem DocumentItem}
 * which require custom rendering to display with different levels of indentation and different icons.
 */
public class VisibleDocumentsBoxCellRenderer implements ListCellRenderer
{
/** Render method for the visible documents list. Renders caption, author names and document names with proper indentation and icons */
public Component getListCellRendererComponent( JList list, Object value, int index, boolean isSelected, boolean cellHasFocus )
{
    JLabel renderLabel = new JLabel();
    renderLabel.setIconTextGap( ClientConstants.getVISIBLE_DOCUMENTS_ICON_TEXT_GAP() );
    renderLabel.setBorder( BorderFactory.createMatteBorder( 3, 3, 3, 3, renderLabel.getBackground() ) );
    renderLabel.setMaximumSize( ClientConstants.getVISIBLE_DOCUMENTS_ITEM_MAX_SIZE() );
    renderLabel.setPreferredSize( renderLabel.getMaximumSize() );
    if ( value instanceof AuthorItem )
    {
        renderLabel.setText( ((AuthorItem) value).getAuthorName() + ":");
        renderLabel.setIcon( new ImageIcon( getClass().getResource( "toolbarButtonGraphics/development/host16.gif" ) ) );
    }
    else if ( value instanceof DocumentItem )
    {
        DocumentItem docItem = (DocumentItem) value;
        String docName = docItem.getDocumentName();
        if ( is_Selected )
        {

125
renderLabel.setIcon( new java.awt.Image( getClass().getResource("/toolbarButtonGraphics/navigation/Forward16.gif")));
renderLabel.setForeground( ClientConstants.getVISIBLDOCUMENTS_SELECTION_COLOUR() );
renderLabel.setText( ClientConstants.getVISIBLDOCUMENTS_TAB_SIZE() + docName );
// caption item last since it is the least likely (being unique)
else if ( value instanceof CaptionItem )
    renderLabel.setText( ((CaptionItem)value).getCaption() );
// renderLabel.setIcon( new java.awt.Image( getClass().getResource("/toolbarButtonGraphics/navigation/Home16.gif")));
renderLabel.setHorizontalAlignment( SwingConstants.CENTER );
// renderLabel.setBorder( BorderFactory.createMatteBorder( 6, 6, 6, 6, renderLabel.getBackground() ));
return renderLabel;
}

/** Event handler for the message area at the bottom of the client application. */
public class MessageAreaViewMouseListener extends MouseAdapter
{
    public void mouseClicked( MouseEvent me )
    {
        if ( me.getClickCount() == 2 )
        {
            showMessageWindow();
        }
    }
}

/** Event handler for the client application main window */
public class ClientWindowAdapter extends WindowAdapter
{
    public void windowActivated( WindowEvent we )
    {
        bringChildWindowsToFront();
    }
    public void windowClosing( WindowEvent we )
    {
        //System.out.println( "windowClosing ");
        try
        {
            terminateAndClose();
            dispose();
            catch ( Throwable t )
        }
    }
}

/** Event handler for the topic tree area of the client application main windows */
public class TopicTreeSelectionListener implements TreeSelectionListener
{
    public void valueChanged( TreeSelectionEvent tse )
    {
        handleTopicButtonClick( tse.getPath() );
    }
}
/** This is a small multi-purpose action listener which was implemented as an experiment to test
  whether running
  * small tasks in the event-handling thread was feasible. It worked but interfered with the GUI.
  */
public class Task implements ActionListener
{
    private int currentValue;

    public Task(int startValue)
    {
        currentValue = startValue;
    }

    public void actionPerformed(ActionEvent actionEvent)
    {
        updateMessageArea(currentValue);
        // System.out.println("ACTION!");
    }

    public void updateValue(int newValue)
    {
        currentValue = newValue;
    }

    public void advance()
    {
        currentValue++;}

} /*
/** This class is used to represent clients that participate in JSDT sessions. It therefore
  implements the JSDT Client interface.
  */
public class DefaultClient implements Client
{
    String name;

    public DefaultClient(String nameArg )
    {
        name = nameArg;
    }

    public Object authenticate (AuthenticationInfo ai )
    {
        return null;
    }

    public String getName ()
    {
        return name;
    }

} /*
public class TopicTreeMouseAdapter extends MouseAdapter
{
    public void mouseClicked( MouseEvent me )
    {
        // handleTopicTreeClick( me );
    }

} /** The renderer class for the discussion tree in the client application main window. Leaf nodes in
  * the tree are
  * rendered with the string representation of the user object embedded in the node
  */
public class DiscussionTreeCellRenderer extends DefaultTreeCellRenderer
{
public Component getTreeCellRendererComponent ( JTree jTree, Object value, boolean selected,
        boolean expanded, boolean leaf, int row, boolean hasFocus )
    {
        Component returnComponent = super.getTreeCellRendererComponent( jTree, value, selected, expanded,
                                leaf, row, hasFocus );
        if ( leaf )
            {
                if ( value instanceof Message )
                    setText( value.toString() );
                return this;
            }
    }

    /** The event handler associated with the compose new message button in the client application
     * message area */
    public class ComposeNewMessageButtonHandler implements ActionListener
    {
        public void actionPerformed ( ActionEvent ae )
        {
            composeNewMessage();
        }
    }

    /** The event handler associated with the delete message button in the client application message
     * area */
    public class DeleteMessageHandler implements ActionListener
    {
        public void actionPerformed ( ActionEvent ae )
        {
            deleteMessage();
        }
    }

    /** The event handler associated with the message properties button in the client application message
     * area */
    public class MessagePropertiesButtonHandler implements ActionListener
    {
        public void actionPerformed ( ActionEvent ae )
        {
            showMessageProperties();
        }
    }

    private JMenuItem ivjAbout = null;
    private AboutDialog ifjAboutDialog = null;
    private JMenuItem ivjAddDocuments = null;
    private AddFiles ivjAddFiles = null;
    private JMenu ivjFileMenu = null;
    private JMenu ivjClientApplicationJMenuBar = null;
    private JMenu ivjMessagePropertiesJMenuBar = null;
    private JMenuItem ivjNewFile = new JMenuItem( "new ");
    private JButton ivjExitButton = null;
    private JMenu ivjHelpMenu = null;
    private JMenuItem ivjHelpTopics = null;
    private JPanel ivjAppletContentPane = null;
    private JPanel ivjPanel2 = null;
    private JSeparator ivjSeparator1 = null;
    private JSeparator ivjSeparator2 = null;
    private JSeparator ivjSeparator3 = null;
    private JSeparator ivjSeparator4 = null;
    private JSeparator ivjSeparator5 = null;
    private JSeparator ivjSeparator6 = null;
    private JSeparator ivjSeparator7 = null;
    private JSeparator ivjSeparator8 = null;
    private JToolBar ivjToolBar1 = null;
    private JToolBar ivjToolBar2 = null;
    private JButton ivjToolBarButton1 = null;

    128
private JButton ivjToolBarButton2 = null;
private JButton ivjToolBarButton3 = null;
private JPanel ivjMessagePanel = null;
private BoxLayout ivjMessagePanelBoxLayout = null;
private ClientPanorama ivjPanoramaPanel = null;
private JMenuItem ivjRemoveDocuments = null;
private RemoveFiles ivjRemoveFiles1 = null;
private HelpDialog ivjHelpDialog1 = null;
private ClientManager clientManager;
private JProgressBar ivjMessageArea = null;
private JButton ivjToolBarButton5 = null;
private JMenuItem ivjSetup = null;
private JButton ivjPreferencesButton = null;
private JMenuItem ivjPreferencesItem = null;
private PreferencesDialog ivjPreferencesDialog1 = null;
private JTree ivjDiscussionTree = null;
private JPanel ivjPanel111 = null;
private JPanel ivjPanel121 = null;
private JTextPane ivjMessageDisplayArea = null;
private JScrollPane ivjInnerSplitPane = null;
private JScrollPane ivjOuterSplitPane = null;
private JToolBar ivjToolBar2 = null;
private JMenuItem ivjFileMenu = null;
private GroupDirectory ivjGroupDirectory1 = null;
private JMenuItem ivjGroups = null;
private JButton ivjComposeNewMessageButton = null;
private JButton ivjMessagePropertiesButton = null;
private JButton ivjDeleteButton = null;
private JScrollPane ivj JScrollPane1 = null;
private boolean terminated = false;
private MessageWindow messageWindow = null;
private JComboBox ivjDocumentBox = null;

class IvjEventHandler implements java.awt.event.ActionListener {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        if (e.getSource() == ClientApplication.this.getAbout())
            connEtoM1(e);
        if (e.getSource() == ClientApplication.this.getAddDocuments())
            connEtoM4(e);
        if (e.getSource() == ClientApplication.this.getRemoveDocuments())
            connEtoM5(e);
        if (e.getSource() == ClientApplication.this.getExit())
            connEtoM7(e);
        if (e.getSource() == ClientApplication.this.getJToolBarButton3())
            connEtoM8(e);
        if (e.getSource() == ClientApplication.this.getJToolBarButton1())
            connEtoM9(e);
        if (e.getSource() == ClientApplication.this.getPreferencesItem())
            PreferencesMenuItemsToDialog(e);
        if (e.getSource() == ClientApplication.this.getPreferencesButton())
            PreferencesButtonToDialog(e);
        if (e.getSource() == ClientApplication.this.getGroups())
            connEtoM2(e);
        if (e.getSource() == ClientApplication.this.getJToolBarButton5())
            connEtoM3(e);
        if (e.getSource() == ClientApplication.this.getExitButton())
            connEtoM6(e);
        if (e.getSource() == ClientApplication.this.getExit())
            connEtoM10(e);
    }
    
    private java.util.Hashtable visibleDocumentsList = new Hashtable();
    private javax.swing.Timer messageAreaTimer;
    private boolean itemListeractive = true;
    /**
     * ClientApplication constructor comment.
     */
public ClientApplication() {
    super();
    initialize();
}

public ClientApplication(String title) {
    super(title);
}

public void addDocumentToVisibleList(String senderName, String authorName, String documentNameAndPath, String documentName) {
    String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
    if ( getVisibleDocumentsList().containsKey( senderName )) {
        ArrayList docList = (ArrayList) getVisibleDocumentsList().get( senderName );
        docList.add( new DocumentItem( fqPath, documentName ));
    } else {
        ArrayList docList = new ArrayList();
        docList.add( new AuthorItem( authorName ));
        docList.add( new DocumentItem( fqPath, documentName ));
        getVisibleDocumentsList().put( senderName, docList );
    }
}

double advanceTask() {
    if ( ivjMessageArea.getValue() >= ivjMessageArea.getMaximum() ) {
        initMessageAreaTimer();
        updateMessageArea( ivjMessageArea.getValue() );
    } else {
        updateMessageArea( ivjMessageArea.getValue()+1 );
    }
}

public void bringChildWindowsOnTop() {
    for ( int i = 0; i < getOwnedWindows().length; i++ ) {
        getOwnedWindows()[i].toFront();
    }
}
public void center()
{

Dimension size;
Dimension preferredSize = new Dimension(
    Integer.parseInt((String) ClientInfo.getPreferences().getProperty(ClientConstants.
        getFRAME_HORIZONTAL_SIZE())),
    Integer.parseInt((String) ClientInfo.getPreferences().getProperty(ClientConstants.
        getFRAME_VERTICAL_SIZE())));

Point position;
Point preferredPosition = new Point(
    Integer.parseInt((String) ClientInfo.getPreferences().getProperty(ClientConstants.
        getFRAME_X_COORDINATE())),
    Integer.parseInt((String) ClientInfo.getPreferences().getProperty(ClientConstants.
        getFRAME_Y_COORDINATE())));

Point defaultPosition = new Point(
    Integer.parseInt(ClientConstants.getDefault_FRAME_X_COORDINATE()),
    Integer.parseInt(ClientConstants.getDefault_FRAME_Y_COORDINATE()));

// if no value has overridden the default setting then query the OS for the dimensions of the display
if (preferredSize.equals(defaultSize))
{
    size = new Dimension(((int) Toolkit().getScreenSize().getWidth() * ClientConstants.
        getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()),
    (int) Toolkit().getScreenSize().getHeight() * ClientConstants.
        getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE());
}
else
{
    size = preferredSize;
}
if (preferredPosition.equals(defaultPosition))
{
    position = new Point(((int) Toolkit().getScreenSize().getWidth() * (1.0 - ClientConstants.
        getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()))/2),
    (int) Toolkit().getScreenSize().getHeight() * (1.0 - ClientConstants.
        getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE())/2));
}
else
{
    position = preferredPosition;
}
} // System.out.println(size.toString());
setSize(size);
setLocation(position);
setState(java.awt.Frame.NORMAL);
repaint();

/**
 * check preconditions for opening a dialog box and, if fulfilled, open it, await completion and
 * validation and then
 * insert the new message, if any, into the topic tree where indicated by the user.
 * Creation date: 10-07-00 22:00:41
 */
public void composeNewMessage()
{
    if (getClientManager().verifyGroupConnectionManagement() == false)
    {
        return;
    }
    if (getDiscussionTree().getSelectionCount() == 0)
{  
    ErrorBox noSelectionErrorBox = new ErrorBox();
    noSelectionErrorBox.setText( ClientConstants.NO_TOPIC_TREE_SELECTION_ERROR_MESSAGE() );
    noSelectionErrorBox.show();
    return;
}
else
{
    ComposeNewMessage composeNewMessage = new ComposeNewMessage();
    composeNewMessage.setClientApplication( this );
    composeNewMessage.setAuthor( ClientInfo.getPreferences().getProperty( ClientConstants.AUTHOR_NAME() ) );
    composeNewMessage.show();
    if ( composeNewMessage.getMessage() != null )
    {
        // construct new message based on the user input in the dialog box and insert it into the topic tree.
        Message message = composeNewMessage.getMessage();
        message.setCreator( ClientInfo.getPreferences().getProperty( ClientConstants.NAME() ) );
        message.setDocument( getClientManager().getDocument().getCurrentDocumentName() );

        TreePath selectionPath = getSelectionPath();
        DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) selectionPath.getLastPathComponent();
        DefaultMutableTreeNode newTreeNode = new DefaultMutableTreeNode( message.getMessageID(), false ) ;
        DefaultMutableTreeNode parentNode = null;
        DefaultMutableTreeNode newNode = null;
        int insertionIndex = 0;
        if ( selectedNode.isLeaf() && ( selectedNode.getParent() != null ) )
        {
            // create leaf node associated with new message
            parentNode = ( DefaultMutableTreeNode ) selectedNode.getParent();
            newNode = newTreeNode;
            // insert node immediately below the selected leaf node
            insertionIndex = parentNode.getIndex( selectedNode ) + 1;
        }
        else
        {
            newNode = new DefaultMutableTreeNode( message.getTitle(), true );
            newNode.add( newNode );
            parentNode = selectedNode;
            // insert node below bottommost child of the selected node’s parent
            insertionIndex = parentNode.getChildCount();

            ( DefaultTreeNode ) getModel().insertNodeInto( newNode, parentNode, parentNode.getChildCount() );
            //getDiscussionTree().scrollPathToVisible( new TreePath( newNode.getPath() ));
            //getDiscussionTree().treeDidChange();
            getDiscussionTree().repaint();
            clientManager.sendMessage( message );
            clientManager.sendTopicTreeUpdate( ( DefaultTreeModel ) getModel() );
        }
    }
}
/**
 * @param arg1 java.awt.event.ActionEvent
 * @return void
 */
private void comnEtoMl(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin {1}
        // user code end
    }
}
getAboutDialog1().show();
    // user code begin {2}
    // user code end
    } catch (java.lang.Throwable ivExc) {
    // user code begin {3}
    // user code end
    handleException(ivExc);
    }
}
/**
 * connEtoM0: (Exit.action.actionPerformed(java.awt.event.ActionEvent) --> ClientApplication.
 * terminateAndClose())
 * @param arg1 java.awt.event.ActionEvent
 */
/*
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM0(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin {1}
        // user code end
        this.terminateAndClose();
        // user code begin {2}
        // user code end
        } catch (java.lang.Throwable ivExc) {
            // user code begin {3}
            // user code end
            handleException(ivExc);
        }
}
/**
 * connEtoM2: (Groups.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory1.show())
 * @param arg1 java.awt.event.ActionEvent
 */
/*
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM2(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin {1}
        getGroupDirectory1().updateTable();
        // user code end
        getGroupDirectory1().show();
        // user code begin {2}
        // user code end
        } catch (java.lang.Throwable ivExc) {
            // user code begin {3}
            // user code end
            handleException(ivExc);
        }
    } catch (java.awt.AWTException ivExc) {
        // user code begin {4}
        // user code end
        handleException(ivExc);
    }
    }
}  
686 /**
688 * show())
689 */
690 private void connEtoM3(java.awt.event.ActionEvent arg1) {
691     try {
692         // user code begin {1}
693         // user code end
694         getGroupDirectory1().show();
695         // user code begin {2}
696         // user code end
697         } catch (java.lang.Throwable ivjExc) {
698             // user code begin {3}
699             // user code end
700             handleException(ivjExc);
701         }
702     }
703     */
704     */
705     * connEtoM4: (AddDocuments.action.actionPerformed(java.awt.event.ActionEvent) --> AddFile1.show())
706     */
707     private void connEtoM4(java.awt.event.ActionEvent arg1) {
708     try {
709         // user code begin {1}
710         // user code end
711         getAddFile1().show();
712         // user code begin {2}
713         // user code end
714         } catch (java.lang.Throwable ivjExc) {
715             // user code begin {3}
716             // user code end
717             handleException(ivjExc);
718         }
719     }
720     */
721     */
722     * connEtoM5: (RemoveDocuments.action.actionPerformed(java.awt.event.ActionEvent) --> RemoveFile1.show
723     */
724     * (java.awt.event.ActionEvent))
725     */
726     private void connEtoM5(java.awt.event.ActionEvent arg1) {
727     try {
728         // user code begin {1}
729         // user code end
730         getRemoveFile1().show();
731         // user code begin {2}
732         // user code end
733         } catch (java.lang.Throwable ivjExc) {
734             // user code begin {3}
735             // user code end
736             handleException(ivjExc);
737         }
738     }
739     */
740     */
741     * connEtoM6: (ExitButton.action.actionPerformed(java.awt.event.ActionEvent) --> ClientApplication.
742     * terminateAndClose())
743     */
744     private void connEtoM6(java.awt.event.ActionEvent arg1) {
745         try {
// user code begin {1}
// user code end
this.terminateAndClose();
// user code begin {2}
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin {3}
// user code end
handleException(ivjExc);
}

/**
 * connEtoM7: (Exit.action.actionPerformed(java.awt.event.ActionEvent) --> ClientApplication.dispose())
 * @param arg1 java.awt.event.ActionEvent
 */
private void connEtoM7(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin {1}
        prepareTermination();
        // user code end
        this.dispose();
        // user code begin {2}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
        // user code end
        handleException(ivjExc);
    }
}

/**
 * connEtoM8: (JToolBarButton3.action.actionPerformed(java.awt.event.ActionEvent) --> AddFile1.show())
 * @param arg1 java.awt.event.ActionEvent
 */
private void connEtoM8(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin {1}
        // user code end
        getAddFile1().show();
        // user code begin {2}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
        // user code end
        handleException(ivjExc);
    }
}

/**
 * connEtoM9: (JToolBarButton1.action.actionPerformed(java.awt.event.ActionEvent) --> RemoveFile1.show())
 * @param arg1 java.awt.event.ActionEvent
 */
private void connEtoM9(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin {1}
        // user code end
        getRemoveFile1().show();
        // user code begin {2}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
        // user code end
        handleException(ivjExc);
    }
}
/**
 * Deletes a message from the topic tree if one is selected and has not been replied to.
 *
 * Creation date: (12-07-00 10:23:54)
 */

public void deleteMessage()
{
    if ( getClientManager().verifyGroupConnectionManagement() == false )
    {
        return;
    }
    if ( getDiscussionTree().getSelectionCount() == 0 )
    {
        ErrorBox noSelectionErrorBox = new ErrorBox();
        noSelectionErrorBox.setText( ClientConstants.getNO_TOPIC_TREE_SELECTION_ERROR_MESSAGE() );
        noSelectionErrorBox.show();
    }
    else
    {
        TreePath selectedPath = getDiscussionTree().getSelectionPath();
        DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) selectedPath.getLastPathComponent();
        // if the selected node is not a leaf, i.e. if it is an inner node, then exit immediately, without
        // displaying an error message
        if ( selectedNode.isLeaf() == false )
        {
            return;
        }
        DefaultMutableTreeNode parentNode = (DefaultMutableTreeNode) selectedNode.getParent();
        DefaultMutableTreeNode successorNode = (DefaultMutableTreeNode) parentNode.getChildAfter( selectedNode );
        // if the selected node has not been responded to AND this client composed the message
        if ( ((successorNode == null) || (successorNode.isLeaf() == false ))&&
            (getClientManager().getClientInfo().getCurrentMessage().getCreator().equalsIgnoreCase( (String ) ClientInfo.getPreferences().get( ClientConstants.getName() ) ))
        )
        {
            ConfirmationBox confirmationBox = new ConfirmationBox();
            confirmationBox.setText( ClientConstants.getCONFIRMATION_MESSAGE_DELETION( (String) getClientManager().getClientInfo().getCurrentMessage().getTitle() ) );
            confirmationBox.show();
            if ( confirmationBox.isConfirmed() )
            {
                // user has confirmed deletion, so proceed accordingly
                clientManager.sendMessageDeletion( (String) selectedNode.getUserObject() );
                DefaultMutableTreeNode predecessorNode = (DefaultMutableTreeNode) parentNode.getChildBefore( selectedNode );
                // if selectedNode is the first and only child of an inner node, then remove that also
                if ( predecessorNode == null )
                {
                    parentNode.removeFromParent();
                    selectedNode.removeFromParent();
                    (DefaultTreeModel) getDiscussionTree().getModel() .reload();
                    clientManager.sendTopicTreeUpdate( (DefaultTreeModel) getDiscussionTree().getModel() );
                }
            }
            else
            {
                ErrorBox invalidSelection = new ErrorBox();
                invalidSelection.setText( ClientConstants.getINVALID_DELETION_ATTEMPTED_MESSAGE() );
                invalidSelection.show();
            }
        }
    }
}
/**
 * Insert the method's description here.
 * Creation date: (28-07-00 00:27:12)
 */
protected void finalize() throws Throwable
{
    // System.out.println( "finalized ");
    // ensure proper termination
    /* if ( isTerminated() == false )
    {
        terminateAndClose();
    }*/
    * Return the About property value.
    * @return javax.swing.JMenuItem
    */
private javax.swing.JMenuItem getAbout() {
    if (ivjAbout == null) {
        try {
            ivjAbout = new javax.swing.JMenuItem();
            ivjAbout.setName("About");
            ivjAbout.setText("About");
        // user code begin (1)
        // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjAbout;
}
* Return the AboutDialog1 property value.
* @return clientapp.AboutDialog
*/
private AboutDialog getAboutDialog1() {
    if (ivjAboutDialog1 == null) {
        try {
            ivjAboutDialog1 = new peerviewmisc.AboutDialog();
            ivjAboutDialog1.setName("AboutDialog1");
            ivjAboutDialog1.setDefaultCloseOperation( javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjAboutDialog1;
}/**
 * Return the AddDocuments property value.
 * @return javax.swing.JMenuItem
 */
public javax.swing.JMenuItem getAddDocuments() {
    if (ivjAddDocuments == null) {
        try {
            ivjAddDocuments = new javax.swing.JMenuItem();
            ivjAddDocuments.setName("AddDocuments");
        }
    }
}
private AddFiles getAddFiles1() {
    if (ivjAddFiles1 == null) {
        try {
            ivjAddFiles1 = new clientapp.AddFiles();
            ivjAddFiles1.setName("AddFiles1");
            ivjAddFiles1.setDefaultCloseOperation(java.awt.WindowConstants.DISPOSE_ON_CLOSE);
            // user code begin (1)
            ivjAddFiles1.setClientApplication(this);
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
        }
        return ivjAddFiles1;
    }
    /**
     * Return the AddFiles1 property value.
     * @return clientapp.AddFiles
     */
    /** WARNING: THIS METHOD WILL BE REGENERATED. */
    private java.awt.JMenuBar getApplicationJMenuBar() {
        if (ivjClientApplicationJMenuBar == null) {
            try {
                ivjClientApplicationJMenuBar = new java.awt.JMenuBar();
                ivjClientApplicationJMenuBar.setName("ClientApplicationJMenuBar");
                ivjClientApplicationJMenuBar.add(getFileMenu());
                ivjClientApplicationJMenuBar.add(setup());
                ivjClientApplicationJMenuBar.add(getHelpMenu());
                // user code begin (1)
                // user code end
                } catch (java.lang.Throwable ivjExc) {
                    // user code begin (2)
                    // user code end
                    handleException(ivjExc);
                }
            }
            return ivjClientApplicationJMenuBar;
        }
    /**
     * Return the client manager object associated with this client and create it if it doesn't already exist.
     * @return clientapp.ClientManager
     */
    public ClientManager getClientManager() {
        if (clientManager == null ) {
            
            138
clientManager = new ClientManager(this);
}
return clientManager;
}

/**
 * Return the DefaultToolBarButton property value.
 */
private javax.swing.JButton getComposeNewMessageButton() {
  if (ivjComposeNewMessageButton == null) {
    try {
      ivjComposeNewMessageButton = new javax.swing.JButton();
      ivjComposeNewMessageButton.setName("ComposeNewMessageButton");
      ivjComposeNewMessageButton.setToolTipText("Compose new message");
      ivjComposeNewMessageButton.setMnemonic('C');
      ivjComposeNewMessageButton.setAlignment(SwingConstants.CENTER);
      ivjComposeNewMessageButton.addActionListener(new ComposeNewMessageButtonHandler());
      // user code begin {1}
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin {2}
      // user code end
      handleException(ivjExc);
    }
  } else {
    return ivjComposeNewMessageButton;
  }

  /**
   * Return the DeleteButton property value.
   */
  private javax.swing.JButton getDeleteButton() {
    if (ivjDeleteButton == null) {
      try {
        ivjDeleteButton = new javax.swing.JButton();
        ivjDeleteButton.setName("DeleteButton");
        ivjDeleteButton.setToolTipText("Delete message");
        ivjDeleteButton.setMnemonic('D');
        ivjDeleteButton.setAlignment(SwingConstants.CENTER);
        ivjDeleteButton.addActionListener(new DeleteButtonHandler());
        // user code begin {1}
        // user code end
      } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
      }
    } else {
      return ivjDeleteButton;
    }

    /**
     * Return the DiscussionTree property value. This method ought to be called
     * getTopicTree() - TO DO
     */
    return javax.swing.JTree�
private javax.swing.JTree getDiscussionTree() {
    try {
        ivjDiscussionTree = new javax.swing.JTree();
        ivjDiscussionTree.setName("DiscussionTree");
        ivjDiscussionTree.setToolTipText("Discussion.tree");
        ivjDiscussionTree.setMaximumSize(new java.awt.Dimension(100000, 100000));
        ivjDiscussionTree.setShowsRootHandles(true);
        ivjDiscussionTree.setPreferredSize(new java.awt.Dimension(78, 1));
        ivjDiscussionTree.setMinimumSize(new java.awt.Dimension(1, 1));
        // user code begin {1}
        DefaultMutableTreeNode rootNode = new DefaultMutableTreeNode();
        ivjDiscussionTree.setModel( new DefaultTreeModel( rootNode ));
        ivjDiscussionTree.setCellVisible( true );
        // ivjDiscussionTree.setCellRenderer( new DiscussionTreeCellRenderer() );
        ivjDiscussionTree.addTreeSelectionListener( new TreeSelectionListener() );
        ivjDiscussionTree.setSelectionModel( new Single_TREE_SELECTION );
        ivjDiscussionTree.setShowsRootHandles( true );
        // make sure that the root is expanded
        ivjDiscussionTree.expandRow(0);
        // user code end
    } catch (javax.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
    }
    return ivjDiscussionTree;
}

/**
 * Return the DocumentBox property value.
 * @return javax.swing.JComboBox
 */
private javax.swing.JComboBox getDocumentBox() {
    try {
        ivjDocumentBox = new JComboBox();
        ivjDocumentBox.setName("DocumentBox");
        ivjDocumentBox.setToolTipText("Visible.documents");
        // user code begin {1}
        ivjDocumentBox.setEditable( false );
        ivjDocumentBox.addItemListener( new VisibleDocumentsBoxItemListener() );
        ivjDocumentBox.setRenderer( new VisibleDocumentsBoxCellRenderer() );
        ivjDocumentBox.insertItemAt( new CaptionItem( ClientConstants.VISIBLE_DOCUMENTS_BOX_CAPTION() ), 0 );
        ivjDocumentBox.setSelectedIndex( 0 );
        ivjDocumentBox.setBorder( BorderFactory.createEtchedBorder() );
        // user code end
    } catch (javax.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
    }
    return ivjDocumentBox;
}

/**
 * Return the Exit property value.
 * @return javax.swing.JMenuItem
 */
private javax.swing.JMenuItem getExit() {
    if (ivjExit == null) {
        try {
            ivjExit = new javax.swing.JMenuItem();
            ivjExit.setName("Exit");
            ivjExit.setText("Exit");
            // user code begin (1)
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
        }
        return ivjExit;
    }

    /**
     * Return the ExitButton property value.
     * @return javax.swing.JButton
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getExitButton() {
        if (ivjExitButton == null) {
            try {
                ivjExitButton = new javax.swing.JButton();
                ivjExitButton.setName("ExitButton");
                ivjExitButton.setToolTipText("Exit");
                ivjExitButton.setText("");
                ivjExitButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
                ivjExitButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
                ivjExitButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/general/Stop24.gif")));
                ivjExitButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (1)
                // user code end
                } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
        }
        return ivjExitButton;
    }

    /**
     * Return the DocumentMenu property value.
     * @return javax.swing.JMenu
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JMenu getFileMenu() {
        if (ivjFileMenu == null) {
            try {
                ivjFileMenu = new javax.swing.JMenu();
                ivjFileMenu.setName("FileMenu");
                ivjFileMenu.setText("File");
                ivjFileMenu.add(getAddDocuments());
                ivjFileMenu.add(getRemoveDocuments());
                ivjFileMenu.add(getSeparator14());
                ivjFileMenu.add(getExit());
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (1)
                // user code end
                } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
        }
        return ivjFileMenu;
    }
```java
return ivjFileMenu;

/**
 * Return the GroupDirectory1 property value.
 * @return clientapp.GroupDirectory
 */
public GroupDirectory getGroupDirectory1() {
    if (ivjGroupDirectory1 == null) {
        try {
            ivjGroupDirectory1 = new clientapp.GroupDirectory();
            ivjGroupDirectory1.setInitialDirectory("GroupDirectory1");
            ivjGroupDirectory1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            // user code begin (1)
            ivjGroupDirectory1.setClientApplication( this );
            ivjGroupDirectory1.initTable();
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjGroupDirectory1;
}

/**
 * Return the Communication property value.
 * @return javax.swing.JMenuItem
 */
public javax.swing.JMenuItem getGroups() {
    if (ivjGroups == null) {
        try {
            ivjGroups = new javax.swing.JMenuItem();
            ivjGroups.setName("Groups");
            ivjGroups.setMnemonic( 'g' );
            ivjGroups.setText("Groups");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjGroups;
}

/**
 * Return the HelpDialog1 property value.
 * @return peerviewmisc.HelpDialog
 */
private peerviewmisc.HelpDialog getHelpDialog1() {
    if (ivjHelpDialog1 == null) {
        try {
            ivjHelpDialog1 = new peerviewmisc.HelpDialog();
            ivjHelpDialog1.setName("HelpDialog1");
            ivjHelpDialog1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
```
```java
return ivjHelpDialog1;
}
/**
 * Return the HelpMenu property value.
 */
private javax.swing.JMenu getHelpMenu() {
    try {
        if (ivjHelpMenu == null) {
            ivjHelpMenu = new javax.swing.JMenu();
            ivjHelpMenu.setName("HelpMenu");
            ivjHelpMenu.setText("Help");
            ivjHelpMenu.add(getHelpTopics());
            ivjHelpMenu.add(getAbout());
            // user code begin {1}
            // help system setup
            HelpSet hs;
            try {
                String helpPathAndName = ClientConstants.getClientDocumentationHelpsetName();
                java.net.URL haURL = HelpSet.findHelpSet(null, helpPathAndName);
                hs = new HelpSet(null, haURL);
                HelpBroker hb = hs.createHelpBroker();
                getHelpTopics().addActionListener(new HelpActionListener(hb));
            }
            catch (Exception E) {
                System.err.println(ClientConstants.getCouldNotInitializeHelp());
            }
            // user code end
            } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
            }
        }
        return ivjHelpMenu;
    }
/**
 * Return the HelpTopics property value.
 */
private javax.swing.JMenuItem getHelpTopics() {
    try {
        if (ivjHelpTopics == null) {
            ivjHelpTopics = new javax.swing.JMenuItem();
            ivjHelpTopics.setName("HelpTopics");
            ivjHelpTopics.setText("HelpTopics");
            // user code begin {1}
            // user code end
            } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
            }
        }
        return ivjHelpTopics;
    }
/**
 * Return the JSplitPane1 property value.
 */
private javax.swing.JSplitPane getSplitPane()
```
private javax.swing.JSplitPane getInnerSplitPane() {
    if (ijvInnerSplitPane == null) {
        try {
            ijvInnerSplitPane = new javax.swing.JSplitPane();
            ijvInnerSplitPane.setName("InnerSplitPane");
            ijvInnerSplitPane.setDividerSize(6);
            ijvInnerSplitPane.setMaximumSize(new java.awt.Dimension(100000, 100000));
            ijvInnerSplitPane.setDividerLocation(300);
            ijvInnerSplitPane.setPreferredSize(new java.awt.Dimension(1, 1));
            ijvInnerSplitPane.setOneTouchExpandable(true);
            ijvInnerSplitPane.setBounds(224, 808, 798, 503);
            getInnerSplitPane().add(getPanel11(), "left");
            getInnerSplitPane().add(getPanel121(), "right");
            // user code begin {1}
            ijvInnerSplitPane.resetToPreferredSizes();
            // user code end
            // catch (java.lang.Throwable ivjExc) {
            //   // user code begin {2}
            //   // user code end
            //   handleException(ivjExc);
            // }
            // return ivjInnerSplitPane;
            /*
             * WARNING: THIS METHOD WILL BE REGENERATED. */
            private javax.swing.JPanel getJAppletContentPane() {
                // return javax.swing.JPanel
                try {
                    iAppletContentPane = new javax.swing.JPanel();
                    iAppletContentPane.setName("JAppletContentPane");
                    iAppletContentPane.setLayout(new java.awt.BorderLayout());
                    getJAppletContentPane().add(getContentPane11(), "South");
                    getJAppletContentPane().add(getJToolBar1(), "North");
                    getJAppletContentPane().add(getJUserSplitPane1(), "Center");
                    // user code begin {1}
                    // user code end
                    // catch (java.lang.Throwable ivjExc) {
                    //   // user code begin {2}
                    //   // user code end
                    //   handleException(ivjExc);
                    // }
                }
                // return iAppletContentPane;
                /*
                 * WARNING: THIS METHOD WILL BE REGENERATED. */
                private javax.swing.JPanel getPanel11() {
                    if (ijvPanel11 == null) {
                        try {
                            iPanel11 = new javax.swing.JPanel();
                            iPanel11.setName("Panel11");
                            iPanel11.setPreferredSize(new java.awt.Dimension(150, 1));
                            iPanel11.setLayout(new java.awt.GridBagLayout());
                            iPanel11.setBackground(java.awt.SystemColor.window);
                            iPanel11.setMinimumSize(new java.awt.Dimension(1, 1));
                            // java.awt.GridBagConstraints constraintsDiscussionTree = new java.awt.GridBagConstraints();
                            constraintsDiscussionTree.gridx = 0; constraintsDiscussionTree.gridy = 0;
                        }
                        // return iPanel11;
                    }
```java
constraintsDiscussionTree.fill = java.awt.GridBagConstraints.BOTH;
constraintsDiscussionTree.anchor = java.awt.GridBagConstraints.WEST;
constraintsDiscussionTree.weightx = 1.0;
constraintsDiscussionTree.weighty = 1.0;
getJPanel11().add(getDiscussionTree(), constraintsDiscussionTree);

java.awt GridBagConstraints constraintsJToolBar2 = new java.awt GridBagConstraints();
constraintsJToolBar2.gridx = 1; constraintsJToolBar2.gridy = 0;
constraintsJToolBar2.fill = java.awt GridBagConstraints.VERTICAL;
constraintsJToolBar2.anchor = java.awt GridBagConstraints.NORTH;
getJPanel11().add(getJToolBar2(), constraintsJToolBar2);

// user code begin {1}

// user code end

} catch (java.lang Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}

return ivjJPanel11;

/**
 * Return the JPanel21 property value.
 */
private javax.swing.JPanel getJPanel21() {
    if (ivjJPanel21 == null) {
        try {
            ivjJPanel21 = new javax.swing.JPanel();
            ivjJPanel21.setName("JPanel21");
            ivjJPanel21.setPreferredSize(new java.awt.Dimension(460, 1));
            ivjJPanel21.setLayout(new java.awt.BorderLayout());
            ivjJPanel21.setMinimumSize(new java.awt.Dimension(1, 1));
            getJPanel21().add(getScrollPane(), "center");
        }
        catch (java.lang Throwable ivjExc) {
            // user code begin {1}
            // user code end
        }
        return ivjJPanel21;
    }

    /**
     * Return the JPanel3 property value.
     */
    private javax.swing.JPanel getJPanel3() {
        if (ivjJPanel3 == null) {
            try {
                ivjJPanel3 = new javax.swing.JPanel();
                ivjJPanel3.setName("JPanel3");
                ivjJPanel3.setLayout(new java.awt.GridBagLayout());

                java.awt GridBagConstraints constraintsJSeparator11 = new java.awt GridBagConstraints();
                constraintsJSeparator11.gridx = 2; constraintsJSeparator11.gridy = 1;
                constraintsJSeparator11.gridwidth = 0;
                constraintsJSeparator11.ipadx = -1;
                constraintsJSeparator11.ipady = 1;
                constraintsJSeparator11 insets = new java.awt.Insets(19, 0, 20, 0);
                getJPanel3().add(getJSeparator11(), constraintsJSeparator11);

                java.awt GridBagConstraints constraintsJSeparator10 = new java.awt GridBagConstraints();
```
constraintsSeparator10.gridx = 3; constraintsSeparator10.gridy = 1;
constraintsSeparator10.gridwidth = 0;
constraintsSeparator10.ipadx = -1;
constraintsSeparator10.ipady = 1;
constraintsSeparator10.insets = new java.awt.Insets(19, 0, 20, 0);
getPanel13().add(getSeparator10(), constraints=Separator10);
java.awt.GridBagConstraints constrainsToolBarButton3 = new java.awt.GridBagConstraints();
constraintsToolBarButton3.gridx = 3; constrainsToolBarButton3.gridy = 1;
constraintsToolBarButton3.insets = new java.awt.Insets(5, 5, 5, 2);
getPanel13().add(getToolBarButton3(), constrains=ToolBarButton3);
java.awt.GridBagConstraints constrainsToolBarButton1 = new java.awt.GridBagConstraints();
constraintsToolBarButton1.gridx = 4; constrainsToolBarButton1.gridy = 1;
constraintsToolBarButton1.insets = new java.awt.Insets(5, 3, 5, 0);
getPanel13().add(getToolBarButton1(), constrains=ToolBarButton1);
java.awt.GridBagConstraints constrainsSeparator2 = new java.awt.GridBagConstraints();
constraintsSeparator2.gridx = 5; constrainsSeparator2.gridy = 1;
constraintsSeparator2.gridwidth = 0;
constraintsSeparator2.ipadx = -1;
constraintsSeparator2.ipady = 1;
constraintsSeparator2.insets = new java.awt.Insets(19, 0, 20, 0);
getPanel13().add(getSeparator2(), constrains=Separator2);
java.awt.GridBagConstraints constrainsSeparator5 = new java.awt.GridBagConstraints();
constraintsSeparator5.gridx = 6; constrainsSeparator5.gridy = 1;
constraintsSeparator5.gridwidth = 0;
constraintsSeparator5.ipadx = -1;
constraintsSeparator5.ipady = 1;
constraintsSeparator5.insets = new java.awt.Insets(19, 0, 20, 0);
getPanel13().add(getSeparator5(), constrains=Separator5);
java.awt.GridBagConstraints constrainsSeparator6 = new java.awt.GridBagConstraints();
constraintsSeparator6.gridx = 0; constrainsSeparator6.gridy = 1;
constraintsSeparator6.gridwidth = 0;
constraintsSeparator6.ipadx = -1;
constraintsSeparator6.ipady = 1;
constraintsSeparator6.insets = new java.awt.Insets(5, 3, 5, 0);
getPanel13().add(getSeparator6(), constrains=Separator6);
java.awt.GridBagConstraints constrainsSeparator13 = new java.awt.GridBagConstraints();
constraintsSeparator13.gridx = 6; constrainsSeparator13.gridy = 1;
constraintsSeparator13.gridwidth = 0;
constraintsSeparator13.ipadx = -1;
constraintsSeparator13.ipady = 1;
constraintsSeparator13.insets = new java.awt.Insets(19, 0, 20, 0);
getPanel13().add(getSeparator13(), constrains=Separator13);
java.awt.GridBagConstraints constrainsSeparator7 = new java.awt.GridBagConstraints();
constraintsSeparator7.gridx = 9; constrainsSeparator7.gridy = 1;
constraintsSeparator7.gridwidth = 0;
constraintsSeparator7.ipadx = -1;
constraintsSeparator7.ipady = 1;
constraintsSeparator7.insets = new java.awt.Insets(19, 0, 20, 0);
getPanel13().add(getSeparator7(), constrains=Separator7);
java.awt.GridBagConstraints constrainsToolBarButton2 = new java.awt.GridBagConstraints();
constraintsToolBarButton2.gridx = 9; constrainsToolBarButton2.gridy = 1;
constraintsToolBarButton2.insets = new java.awt.Insets(5, 5, 5, 2);
getPanel13().add(getToolBarButton2(), constrains=ToolBarButton2);
java.awt.GridBagConstraints constrainsExitButton = new java.awt.GridBagConstraints();
constraintsExitButton.gridx = 10; constraintsExitButton.gridy = 1;
constraintsExitButton.insets = new java.awt.Insets(5, 3, 5, 5);
gPanel3().add(getExitButton(), constraintsExitButton);

java.awt.GridBagConstraints constraintsJSeparator9 = new java.awt.GridBagConstraints();
constraintsJSeparator9.gridx = 11; constraintsJSeparator9.gridy = 1;
constraintsJSeparator9.gridwidth = 0;
constraintsJSeparator9.ipadx = -1;
constraintsJSeparator9.ipady = 1;
constraintsJSeparator9.insets = new java.awt.Insets(19, 0, 20, 0);
gPanel3().add(getJSeparator9(), constraintsJSeparator9);

java.awt.GridBagConstraints constraintsJSeparator6 = new java.awt.GridBagConstraints();
constraintsJSeparator6.gridx = 12; constraintsJSeparator6.gridy = 1;
constraintsJSeparator6.weightx = 1.0;
constraintsJSeparator6.gridwidth = 0;
constraintsJSeparator6.ipadx = -1;
constraintsJSeparator6.ipady = 1;
constraintsJSeparator6.insets = new java.awt.Insets(19, 0, 20, 0);
gPanel3().add(getJSeparator6(), constraintsJSeparator6);

java.awt.GridBagConstraints constraintsDocumentBox = new java.awt.GridBagConstraints();
constraintsDocumentBox.gridx = 13; constraintsDocumentBox.gridy = 1;
constraintsDocumentBox.anchor = java.awt.GridBagConstraints.EAST;
constraintsDocumentBox.weightx = 0.0;
constraintsDocumentBox.weighty = 0.0;
constraintsDocumentBox.ipadx = 250;
constraintsDocumentBox.ipady = 10;
constraintsDocumentBox.insets = new java.awt.Insets(7, 5, 7, 7);
gPanel3().add(getDocumentBox(), constraintsDocumentBox);

// user code begin (1)
// user code end
}
}

} catch (java.lang.Throwable ivJExc) {
  // user code begin (2)
  // user code end
  handleException(ivJExc);
}

} return ivjPanel3;

/**
 * Return the JScrollPanel1 property value.
 * @return javax.swing.JScrollPanel
 */
private javax.swing.JScrollPanel getJScrollPanel1() {
  if (ivjJScrollPanel1 == null) {
    try {
      ivjJScrollPanel1 = new javax.swing.JScrollPanel();
      ivjJScrollPanel1.setName("JScrollPanel1");
      getJScrollPanel1().setViewportView(getMessageDisplayArea());
      // user code begin (1)
      // user code end
    }
    }
    catch (java.lang.Throwable ivJExc) {
      // user code begin (2)
      // user code end
      handleException(ivJExc);
    }

  } return ivjJScrollPanel1;

/**
 * Return the JSeparator10 property value.
 * @return javax.swing.JSeparator
 */
private javax.swing.JSeparator getJSeparator10() {
if (ivjJSeparator10 == null) {
    try {
        ivjJSeparator10 = new javax.swing.JSeparator();
        ivjJSeparator10.setName("JSeparator10");
        // user code begin (1)
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
    return ivjJSeparator10;
}/**
 * Return the JSeparator11 property value.
 * @return javax.swing.JSeparator
 */
private javax.swing.JSeparator getJSeparator11() {
    if (ivjJSeparator11 == null) {
        try {
            ivjJSeparator11 = new javax.swing.JSeparator();
            ivjJSeparator11.setName("JSeparator11");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
        return ivjJSeparator11;
}/**
 * Return the JSeparator13 property value.
 * @return javax.swing.JSeparator
 */
private javax.swing.JSeparator getJSeparator13() {
    if (ivjJSeparator13 == null) {
        try {
            ivjJSeparator13 = new javax.swing.JSeparator();
            ivjJSeparator13.setName("JSeparator13");
            ivjJSeparator13.setFont(new java.awt.Font("dialog", 0, 24));
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
        return ivjJSeparator13;
}/**
 * Return the JSeparator14 property value.
 * @return javax.swing.JSeparator
 */
private javax.swing.JSeparator getJSeparator14() {
    if (ivjJSeparator14 == null) {
        try {
            ivjJSeparator14 = new javax.swing.JSeparator();
            ivjJSeparator14.setName("JSeparator14");
            // user code begin (1)
        }
// user code end

catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
}

} // user code end

/**
 * Return the JSeparator2 property value.
 */
private javax.swing.JSeparator getJSeparator2() {
    if (ivjJSeparator2 == null) {
        try {
            ivjJSeparator2 = new javax.swing.JSeparator();
            ivjJSeparator2.setName("JSeparator2");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJSeparator2;
}

/**
 * Return the JSeparator5 property value.
 */
private javax.swing.JSeparator getJSeparator5() {
    if (ivjJSeparator5 == null) {
        try {
            ivjJSeparator5 = new javax.swing.JSeparator();
            ivjJSeparator5.setName("JSeparator5");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJSeparator5;
}

/**
 * Return the JSeparator6 property value.
 */
private javax.swing.JSeparator getJSeparator6() {
    if (ivjJSeparator6 == null) {
        try {
            ivjJSeparator6 = new javax.swing.JSeparator();
            ivjJSeparator6.setName("JSeparator6");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJSeparator6;
}
return ivjJSeparator6;
}

/**
 * Return the JSeparator7 property value.
 * @return javax.swing.JSeparator
 */
private javax.swing.JSeparator getJSeparator7() {
    if (ivjJSeparator7 == null) {
        try {
            ivjJSeparator7 = new javax.swing.JSeparator();
            ivjJSeparator7.setName("JSeparator7");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJSeparator7;
}

/**
 * Return the JSeparator9 property value.
 * @return javax.swing.JSeparator
 */
private javax.swing.JSeparator getJSeparator9() {
    if (ivjJSeparator9 == null) {
        try {
            ivjJSeparator9 = new javax.swing.JSeparator();
            ivjJSeparator9.setName("JSeparator9");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJSeparator9;
}

/**
 * Return the JToolBar property value.
 * @return javax.swing.JToolBar
 */
private javax.swing.JToolBar getJToolBar() {
    try {
        ivjJToolBar1 = new javax.swing.JToolBar();
        ivjJToolBar1.setName("JToolBar1");
        ivjJToolBar1.setFloatable(false);
        getJToolBar().add(getJPanel3(), getJPanel3().getName());
        // user code begin (1)
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
    return ivjJToolBar1;
}
* Return the JToolBar2 property value.
1792 * @return javax.swing.JToolBar
1793 */
1794 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JToolBar getJToolBar2() {
1796 if (ivjJToolBar2 == null) {
1797 try {
1798 ivjJToolBar2 = new javax.swing.JToolBar();
1799 ivjJToolBar2.setName("JToolBar2");
1800 ivjJToolBar2.setFloatable(false);
1801 ivjJToolBar2.setBackground(java.awt.Color.white);
1802 ivjJToolBar2.setOrientation(java.awt.swing.SwingConstants.VERTICAL);
1803 ivjJToolBar2.add(getComposeNewMessageButton());
1804 ivjJToolBar2().add(getDeleteButton(), getDeleteButton().getName());
1805 ivjJToolBar2().add(getMessagePropertiesButton(), getMessagePropertiesButton().getName());
1806 // user code begin {1}
1807 // user code end
1808 } catch (javax.lang.Throwable ivjExc) {
1809 // user code begin {2}
1810 // user code end
1811 handleException(ivjExc);
1812 }
1813 return ivjJToolBar2;
1814 }
1816 **
1817 * Return the JToolBarButton1 property value.
1818 * @return javax.swing.JButton
1819 */
1820 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getJToolBarButton1() {
1822 if (ivjJToolBarButton1 == null) {
1823 try {
1824 ivjJToolBarButton1 = new javax.swing.JButton();
1825 ivjJToolBarButton1.setSetName("JToolBarButton1");
1826 ivjJToolBarButton1.setTool1TipText("Remove Documents");
1827 ivjJToolBarButton1.setTool1Text("");
1828 // ivjJToolBarButton1.setHover1TextPosition(java.awt.swing.SwingConstants.CENTER);
1829 // ivjJToolBarButton1.setHover1TextPosition(java.awt.swing.SwingConstants.BOTTOM);
1830 ivjJToolBarButton1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/general/Export24.gif")));
1831 ivjJToolBarButton1.setMargin(new java.awt.Insets(0, 0, 0, 0));
1832 // user code begin {1}
1833 // user code end
1834 } catch (javax.lang.Throwable ivjExc) {
1835 // user code begin {2}
1836 // user code end
1837 handleException(ivjExc);
1838 }
1840 return ivjJToolBarButton1;
1841 }
1842 **
1843 * Return the JToolBarButton2 property value.
1844 * @return javax.swing.JButton
1845 */
1846 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getJToolBarButton2() {
1848 if (ivjJToolBarButton2 == null) {
1849 try {
1850 ivjJToolBarButton2 = new javax.swing.JButton();
1851 ivjJToolBarButton2.setSetName("JToolBarButton2");
1852 ivjJToolBarButton2.setTool1TipText("");
1853 ivjJToolBarButton2.setTool1Text("");
1854 // ivjJToolBarButton2.setHover1TextPosition(java.awt.swing.SwingConstants.CENTER);
1855 // ivjJToolBarButton2.setHover1TextPosition(java.awt.swing.SwingConstants.BOTTOM);
ivjToolBarButton2.setIcon(new java.swing.ImageIcon(getClass().getResource("/
 toolBarButtonGraphics\general\Help24.gif")));
ivjToolBarButton2.setMargin(new java.awt.Insets(0, 0, 0, 0));
// user code begin {1}
// user code end
}
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}
}
return ivjToolBarButton2;
}
/**
 * Return the JToolBarButton3 property value.
 */
private java.awt.Button getJToolBarButton3() {
    if (ivjJToolBarButton3 == null) {
        try {
            ivjJToolBarButton3 = new java.awt.Button();
            ivjJToolBarButton3.setName("JToolBarButton3");
            ivjJToolBarButton3.setToolTipText("Add documents");
            ivjJToolBarButton3.setText("");
            ivjJToolBarButton3.setHorizontalTextPosition(java.awt.SwingConstants.CENTER);
            ivjJToolBarButton3.setVerticalTextPosition(java.awt.SwingConstants.BOTTOM);
            ivjJToolBarButton3.setIcon(new java.awt.ImageIcon(getClass().getResource("/
 toolBarButtonGraphics\general\Import24.gif")));
            ivjJToolBarButton3.setMargin(new java.awt.Insets(0, 0, 0, 0));
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
}
    return ivjJToolBarButton3;
}
/**
 * Return the JToolBarButton5 property value.
 */
private java.awt.Button getJToolBarButton5() {
    if (ivjJToolBarButton5 == null) {
        try {
            ivjJToolBarButton5 = new java.awt.Button();
            ivjJToolBarButton5.setName("JToolBarButton5");
            ivjJToolBarButton5.setToolTipText("Group directory");
            ivjJToolBarButton5.setText("");
            ivjJToolBarButton5.setHorizontalTextPosition(java.awt.SwingConstants.CENTER);
            ivjJToolBarButton5.setVerticalTextPosition(java.awt.SwingConstants.BOTTOM);
            ivjJToolBarButton5.setIcon(new java.awt.ImageIcon(getClass().getResource("/
 toolBarButtonGraphics\general\Preferences24.gif")));
            ivjJToolBarButton5.setMargin(new java.awt.Insets(0, 0, 0, 0));
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
}
    return ivjJToolBarButton5;
}
/**
   * Return the JProgressbar1 property value.
   * @return javax.swing.JProgressBar
   */
   public javax.swing.JProgressBar getMessageArea() {
      try {
         ivjMessageArea = new javax.swing.JProgressBar();
         ivjMessageArea.setName("MessageArea");
         ivjMessageArea.setString("MessageArea
         "double-click here to open in separate window...");
         ivjMessageArea.setStringPainted(true);
         // user code begin (1)
         ivjMessageArea.setPreferredSize(new Dimension(Integer.MAX_VALUE, (int) ivjMessageArea.
         getPreferredSize().getHeight()));
         ivjMessageArea.setMaximum(ivjMessageArea.getPreferredSize());
         ivjMessageArea.addMouseListener(new MessageAreaMouseListener());
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin (2)
         handleException(ivjExc);
      }
      return ivjMessageArea;
   }
   */
   // Return the ButtonPanelFlowLayout property value.
   // @return java.awt.FlowLayout
   */
   private java.awt.FlowLayout getMessageAreaFlowLayout() {
      java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
      try {
         /* Create part */
         ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
         ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      }
      return ivjButtonPanelFlowLayout;
   }
   */
   // Insert the method's description here.
   // Creation date: (31-08-00 02:10:56)
   // @return javax.swing.Timer
   */
   public javax.swing.Timer getMessageAreaTimer() {
      return MessageAreaTimer;
   }
   */
   // Return the MessageDisplayArea property value.
   // @return javax.swing.JTextPane
   */
   private javax.swing.JTextPane getMessageDisplayArea() {
      if (ivjMessageDisplayArea == null) {
         try {
            ivjMessageDisplayArea = new javax.swing.JTextPane();
            ivjMessageDisplayArea.setName("MessageDisplayArea");
            ivjMessageDisplayArea.setToolTipText("null");
            ivjMessageDisplayArea.setPreferredSize(new java.awt.Dimension(7, 1));
            ivjMessageDisplayArea.setBounds(0, 0, 490, 490);
            ivjMessageDisplayArea.setEnabled(false);
            ivjMessageDisplayArea.setMinimumSize(new java.awt.Dimension(6, 1));
         } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
         }
      }
   }
// user code begin {1}
ivjMessageDisplayArea.setToolTipText(null);
// user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}
return ivjMessageDisplayArea;

/**
* Return the MessagePanel property value.
* @return javax.swing.JPanel
*/

private javax.swing.JPanel getMessagePanel() {
    try {
        ivjMessagePanel = new javax.swing.JPanel();
        ivjMessagePanel.setName("MessagePanel");
        ivjMessagePanel.setLayout(getMessagePanelBoxLayout());
        getMessagePanel().add(getMessageArea(), getMessageArea().getName());
        // user code begin {1}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
    }
    return ivjMessagePanel;
}

/**
* Return the MessagePanelBoxLayout property value.
* @return javax.swingBoxLayout
*/

private javax.swing.BoxLayout getMessagePanelBoxLayout() {
    try {
        ivjMessagePanelBoxLayout = null;
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    return ivjMessagePanelBoxLayout;
}

/**
* Return the JToolBarButton4 property value.
* @return javax.swing.JButton
*/
ivjMessagePropertiesButton.setMargin(new java.awt.Insets(0, 0, 0, 0));  
// user code begin (1)  
ivjMessagePropertiesButton.addActionListener(new MessagePropertiesButtonHandler());  
// user code end  
}  
catch (java.lang.Throwable ivjExc) {  
  // user code begin (2)  
  // user code end
  handleException(ivjExc);
}

return ivjMessagePropertiesButton;

/**  
* Insert the method’s description here.  
* Creation date: (16-08-00 17:06:09)  
* @return clientapp.MessageWindow */  
public MessageWindow getMessageBoxWindow()  
if (messageWindow == null)  
{  
  messageWindow = new MessageBoxWindow();  
}

return messageWindow;

/**  
* Return the JSplitPanel property value.  
* @return javax.swing.JSplitPane */  
private javax.swing.JSplitPane getOuterSplitPane()  
if (ivjOuterSplitPane == null) {  
  try {  
    ivjOuterSplitPane = new javax.swing.JSplitPane(javax.swing.JSplitPane.VERTICAL_SPLIT);  
    ivjOuterSplitPane.setName("OuterSplitPane");  
    ivjOuterSplitPane.setDividerSize(6);  
    ivjOuterSplitPane.setLastDividerLocation(600);  
    ivjOuterSplitPane.setAlignmentY(javax.awt.Component.BOTTOM_ALIGNMENT);  
    ivjOuterSplitPane.setDividerLocation(600);  
    ivjOuterSplitPane.setPreferredSize(new java.awt.Dimension(589, 1000));  
    ivjOuterSplitPane.setAlignmentX(java.awt.Component.RIGHT_ALIGNMENT);  
    ivjOuterSplitPane.setOneTouchExpandable(true);  
    ivjOuterSplitPane.setMinimumSize(new java.awt.Dimension(503, 1000));  
    getOuterSplitPane().add(getPanoramaPanel(), "top");  
    // user code begin (1)  
    ivjOuterSplitPane.setDividerLocation( 0.8 );  
    // user code end
  }  
catch (java.lang.Throwable ivjExc) {  
  // user code begin (2)  
  // user code end
  handleException(ivjExc);
}

return ivjOuterSplitPane;

/**  
* Insert the method’s description here.  
* Creation date: (04-08-00 14:39:32)  
* @return clientapp.ClientPanorama */  
public ClientPanorama getPanoramaPanel() {  
if (ivjPanoramaPanel == null) {  
  try {  
    ivjPanoramaPanel = new clientapp.ClientPanorama();
  }

155
ivjPanoramaPanel.setName("PanoramaPanel");
ivjPanoramaPanel.setToolTipText("null");
ivjPanoramaPanel.setPreferredSize(new java.awt.Dimension(300, 300));
ivjPanoramaPanel.setAlignment(VBox.CENTER);
ivjPanoramaPanel.setMinimumSize(new java.awt.Dimension(1, 1));
// user code begin (1)
// user code end
}
} catch (java.lang.Throwable ivjExc) {
  // user code begin (2)
  // user code end
  handleException(ivjExc);
}
return ivjPanoramaPanel;

/**
 * Return the JToolBar.Button4 property value.
 */
private java.awt.Button getPreferencesButton() {
if (ivjPreferencesButton == null) {
  try {
    ivjPreferencesButton = new java.awt.Button();
    ivjPreferencesButton.setName("PreferencesButton");
    ivjPreferencesButton.setToolTipText("Preferences");
    ivjPreferencesButton.setHorizontalTextPosition(java.awt.SwingConstants.CENTER);
    ivjPreferencesButton.setVerticalTextPosition(java.awt.SwingConstants.BOTTOM);
    ivjPreferencesButton.setIcon(new java.awt.ImageIcon(getClass().getResource("/toolBarButtonGraphics\development\Host24.gif")));
    ivjPreferencesButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
  // user code begin (1)
  // user code end
  } catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
  }
}
return ivjPreferencesButton;

/**
 * Return the PreferencesDialog1 property value.
 */
private clientapp.PreferencesDialog getPreferencesDialog1() {
if (ivjPreferencesDialog1 == null) {
  try {
    ivjPreferencesDialog1 = new clientapp.PreferencesDialog();
    ivjPreferencesDialog1.setName("PreferencesDialog1");
    ivjPreferencesDialog1.setDefaultCloseOperation(java.awt.WindowConstants.DISPOSE_ON_CLOSE);
    // user code begin (1)
    ivjPreferencesDialog1.setClientApplication( this );
    ivjPreferencesDialog1.init();
    // user code end
  } catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
  }
}
return ivjPreferencesDialog1;

/**
* Return the OptionsItem property value.
* @return javax.swing.JMenuItem
*/
private javax.swing.JMenuItem getPreferencesItem() {
    if (ivjPreferencesItem == null) {
        try {
            ivjPreferencesItem = new javax.swing.JMenuItem();
            ivjPreferencesItem.setName("PreferencesItem");
            ivjPreferencesItem.setText("Preferences");
            // user code begin (1)
            // user code end
            } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
        return ivjPreferencesItem;
    }
    /**
     * Return the RemoveDocuments property value.
     * @return javax.swing.JMenuItem
     */
    public javax.swing.JMenuItem getRemoveDocuments() {
        if (ivjRemoveDocuments == null) {
            try {
                ivjRemoveDocuments = new javax.swing.JMenuItem();
                ivjRemoveDocuments.setName("RemoveDocuments");
                ivjRemoveDocuments.setText("Remove,Documents...");
                // user code begin (1)
                // user code end
                } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
            return ivjRemoveDocuments;
        }
        /**
         * Return the RemoveFiles1 property value.
         * @return clientapp.RemoveFiles
         */
        private RemoveFiles getRemoveFiles1() {
            if (ivjRemoveFiles1 == null) {
                try {
                    ivjRemoveFiles1 = new clientapp.RemoveFiles();
                    ivjRemoveFiles1.setName("RemoveFiles1");
                    // user code begin (1)
                    ivjRemoveFiles1.setClientApplication( this );
                    ivjRemoveFiles1.initialize();
                    // user code end
                    } catch (java.lang.Throwable ivjExc) {
                    // user code begin (2)
                    // user code end
                    handleException(ivjExc);
                }
                return ivjRemoveFiles1;
            }
            /**
             * Return the Setup property value.
             * @return javax.swing.JMenuItem
private javax.swing.JMenuItem getSetup() {
    if (ivjSetup == null) {
        try {
            ivjSetup = new javax.swing.JMenuItem();
            ivjSetup.setName("Setup");
            ivjSetup.setMnemonic(’S’);
            ivjSetup.setToolTipText("Setup");
            ivjSetup.add(getGroups());
            ivjSetup.add(getPreferencesItem());
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
        return ivjSetup;
    }
    //**
    * Return the visibleDocumentsList property.
    * Creation date: (30-08-00 16:14:33)
    * @return java.util.Hashtable
    */
    public java.util.Hashtable getVisibleDocumentsList() {
        return visibleDocumentsList;
    }
    //**
    * Called whenever the part throws an exception.
    * @param exception java.lang.Throwable
    */
    private void handleException(java.lang.Throwable exception) {
        /* Uncomment the following lines to print uncaught exceptions to stdout */
        System.out.println("----------UNCAUGHT EXCEPTION----------");
        exception.printStackTrace(System.out);
    }
    //**
    * Insert the method’s description here.
    * Creation date: (11-08-00 17:57:56)
    * @param mouseEvent java.awt.event.MouseEvent
    */
    public void handleTopicTreeClick(TreePath selPath) {
        if (selPath != null) {
            DefaultMutableTreeNode treeNode = (DefaultMutableTreeNode) selPath.getLastPathComponent();
            if (treeNode.isLeaf() == true & treeNode.getUserObject() != null) {
                // retrieve ID for the message object associated with the selected node and request it from the server
                String messageID = (String) treeNode.getUserObject();
                getClientManager().requestMessageFromServer(messageID);
                // System.out.println( "Send it back after " + messageID );
            } else {
                setMessage(null);
                getClientManager().getClientInfo().setCurrentMessage(null);
            }
        }
    }
}
* Initializes connections
* @Exception java.lang.Exception The exception description.
*/
private void initConnections() throws java.lang.Exception {
    // user code begin {1}
    // user code end
    getAbout().addActionListener(ivjEventHandler);
    getDocuments().addActionListener(ivjEventHandler);
    getRemoveDocuments().addActionListener(ivjEventHandler);
    getExit().addActionListener(ivjEventHandler);
    getJToolBarButton3().addActionListener(ivjEventHandler);
    getJToolBarButton1().addActionListener(ivjEventHandler);
    getHelpTopics().addActionListener(ivjEventHandler);
    getJToolBarButton2().addActionListener(ivjEventHandler);
    getPreferencesItem().addActionListener(ivjEventHandler);
    getPreferencesButton().addActionListener(ivjEventHandler);
    getGroups().addActionListener(ivjEventHandler);
    getJToolBarButton6().addActionListener(ivjEventHandler);
    getExitButton().addActionListener(ivjEventHandler);
}
// user code begin {2}
try {
    // user code begin {1}
    initPanoramaPanel();
    // user code end
    setName("ClientApplication");
    setTitle("PeerView_client");
    setSize(800, 600);
    setJMenuBar(getClientApplication.JMenuBar());
    getContentPane().add(getJAppletContentPane());
    initConnections();
} catch (java.lang.Throwable ivjExc) {
    handleException(ivjExc);
}
// user code begin {2}
setDefaultCloseOperation( WindowConstants.DO NOTHING_ON_CLOSE );
addWindowListener( new ClientWindowListener());
generateWindow().setVisible( false );
genPanoramaPanel().setOwner(this);
resetMessageArea();
// the below code is inserted here due to difficulty using the visual composer to properly display
// bottom components of the outer split pane.
generateOuterSplitPane().add( generateInnerSplitPane(), "bottom");
center();
genClientManager().configure();
// user code end
// user code begin {2}
/**
 * Insert the method's description here.
 * @param minimum int
 * @param maximum int
 * @param value int
 * @param string java.lang.String
 */
public void initMessageArea(int minimum, int maximum, int value, String string)
```java
2374  {
2376      ivjMessageArea.setMinum( minimum );
2378      ivjMessageArea.setMaxum( maximum );
2379      ivjMessageArea.setValue( value );
2380      getMessageWindow().addString( string );
2381      // make sure the message disappears after a set interval if the progress bar is fixed
2382      if ( minimum == maximum )
2383          { 
2384              initMessageAreaTimer();
2385          }
2386      else 
2387          {
2388              if ( getMessageAreaTimer() != null )
2389                  {
2390                      getMessageAreaTimer().stop();
2391                  }
2392          }
2393  }/*
2394  /**
2395  * * Initialize the message area timer.
2396  */
2397 private void initMessageAreaTimer()
2398  {
2399      // start a timer to allow the current message a brief display interval before being deleted.
2400      if ( ! ( getMessageAreaTimer() == null ) )
2401          ( getMessageAreaTimer().isRunning() == false )
2402              {
2403                  setMessageAreaTimer( new Timer
2404                      ( ClientConstants.getRESET_MESSAGE_AREA_DELAY() ,
2405                      new ActionListener()
2406                          {
2407                              public void actionPerformed( ActionEvent ae )
2408                                  {
2409                                      resetMessageArea();
2410                                  }
2411                          });
2412          });
2413      getMessageAreaTimer().setRepeats( false );
2414      getMessageAreaTimer().start();
2415  }
2416  */
2417  /**
2418  * Return the PanoramaPanel property value.
2419  */
2420  public ClientPanorama initPanoramaPanel()
2421  {
2422      if ( ivjPanoramaPanel == null ) {
2423          try {
2424              ivjPanoramaPanel = new clientapp.ClientPanorama();
2425              ivjPanoramaPanel.setName("PanoramaPanel");
2426              ivjPanoramaPanel.setPreferredSize( new java.awt.Dimension(300, 300) );
2427              ivjPanoramaPanel.setAlignmentY( java.awt.Component.BOTTOM_ALIGNMENT );
2428              ivjPanoramaPanel.setMinimumSize( new java.awt.Dimension(1, 1) );
2429              // user code begin {1}
2430              // user code end
2431          } catch (java.lang.Throwable ivjExc) {
2432              // user code begin {2}
2433              // user code end
2434              handleException(ivjExc);
2435          } }
2436      } catch (java.lang.Throwable ivjExc) {
2437          // user code begin {2}
2438          // user code end
2439          handleException(ivjExc);
2440      }
2441  }
```
return isVisible();
}

/**
 * Insert the method's description here.
 * Creation date: (06-09-00 22:18:03)
 * @return boolean
 */
public boolean isItemListenerActive() {
  return itemListListenerActive;
}

/**
 * Insert the method's description here.
 * Creation date: (15-06-00 08:48:46)
 * @return boolean
 */
public boolean isTerminated() {
  return terminated;
}

/**
 * Lock interface by disabling those menu items that give access to activities that run in separate
targets.
 * The purpose of this is to minimize the potential for thread conflicts when operations such as
adddocuments and
removedocuments are set to execute in their own threads.
 * Creation date: (09-10-00 21:49:32)
 * @param value boolean
 */
public void lockInterface(boolean value)
{
  getAddDocuments().setEnabled( !value );
  removeDocuments().setEnabled( !value );
  getGroups().setEnabled( !value );
  getPanoramaPanel().setPanoramaLock( value );
}

/**
 * main entry point - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args)
{
  try {
    ClientApplication aClientApplication;
    aClientApplication = new ClientApplication();
    aClientApplication.addWindowListener(new java.awt.event.WindowAdapter()
    {
      public void windowClosing(java.awt.event.WindowEvent e)
      {
        System.exit(0);
      }
    });
    aClientApplication.setVisible(true);
  }
  } catch (Throwable exception) {
    System.err.println("Exception occurred in main() of
"+javax.swing.JFrame");
    exception.printStackTrace(System.out);
  }
}

/**
 * PreferencesButtonToDialog: (PreferencesButton.action.actionPerformed(java.awt.event.ActionEvent) -->
PreferencesDialog1.show(JV)
 * @param arg1 java.awt.event.ActionEvent
 */
private void PreferencesButtonToDialog(java.awt.event.ActionEvent arg1) {
  try {
    // user code begin (1)
    // user code end
    getPreferencesDialog1().show();
  }
}
// user code begin {2}
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin {3}
// user code end
handleException(ivjExc);
}
/**
 * @param arg1 java.awt.event.ActionEvent
 */
// WARNING: THIS METHOD WILL BE REGENERATED. */
private void PreferencesMenuToDialog(java.awt.event.ActionEvent arg1) {
  try {
    // user code begin {1}
    // user code end
    getPreferencesDialog1().show();
    // user code begin {2}
    // user code end
  } catch (java.lang.Throwable ivjExc) {
    // user code begin {3}
    // user code end
  }
  handleException(ivjExc);
}
/**
 * Prepare termination of the object.
 * @param arg1 java.awt.event.ActionEvent
 */
public void prepareTermination() {
  if ( getClientManager().getClientInfo().isLocalMode() == false )
  {
    getClientManager().disconnectFromGroup( getClientManager().getClientInfo().getActiveGroup() );
    getClientManager().informServerOfExit();
    getClientManager().getClientInfo().writePreferences();
    setTerminated( true );
  }
}
/**
 * Remove a document from the visible document list box.
 * @param argName java.lang.String
 */
public void removeDocumentFromVisibleList(String argName, String documentName) {
  String fqPath = ClientConstants.createFullyQualifiedDocumentName( argName, documentName );
  if ( getVisibleDocumentsList().containsKey( argName ))
  {
    ArrayList docList = (ArrayList) getVisibleDocumentsList().get( argName );
    for ( int i = 0; i < docList.size(); i++ )
    {
      if ( docList.get( i ) instanceof DocumentItem )
      {
        if ( ((DocumentItem) docList.get( i )).getDocumentID().equals( fqPath ))
        {
          docList.remove( i );
        }
      }
    }
    // if group member no longer contributes to panorama, i.e. if the author's name is the only entry
    // then remove the list

    if ( docList.size() == 1 )
  }}

162
```java
2568  }
2569  }
2570  }
2571  */
2572  /**
2573   * Clear the message area of the client application main window, i.e. empty it so that it appears blank
2574   * to the user.
2575   * creation date: (18-06-00 17:17:37)
2576   */
2577   public void resetMessageArea()
2578   {
2579       getMessageArea().setSize( getMessagePanel().getSize() );
2580       getMessageArea().setText( ClientConstants.getMessageAreaDefaultString() );
2581       getMessageArea().setValue( 0 );
2582       validate();
2583       repaint();
2584   }
2585   */
2586   /**
2587   * Set method for the client manager object.
2588   * creation date: (16-06-00 00:04:31)
2589   */
2590   public void setClientManager(ClientManager newClientManager)
2591   {
2592       clientManager = newClientManager;
2593   }
2594   */
2595   /**
2596   * Set method for the itemListenerActive field.
2597   * creation date: (05-09-00 22:15:03)
2598   */
2599   public void setItemListenerActive(boolean newItemListenerActive)
2600   {
2601       itemListenerActive = newItemListenerActive;
2602   }
2603   */
2604   /**
2605   * Set the message display area parameters to reflect the contents of the argument or reset to default
2606   * values if argument is null.
2607   * creation date: (10-07-00 14:44:52)
2608   */
2609   public void setMessage( Message message )
2610   {
2611       if ( message != null )
2612           {
2613               getMessageDisplayArea().setText( message.getContents() );
2614               getMessageDisplayArea().setToolTipText( message.toString() );
2615           }
2616       else
2617           {
2618               getMessageDisplayArea().setText( "" );
2619               getMessageDisplayArea().setToolTipText( null );
2620           }
2621       getMessageDisplayArea().repaint();
2622   }
2623   */
2624   /**
2625   * Set method for the messageAreaTimer field.
2626   * creation date: (31-08-00 02:18:56)
2627   */
2628   public void setMessageAreaTimer( javax.swing.Timer newMessageAreaTimer )
2629   {
2630       messageAreaTimer = newMessageAreaTimer;
2631   }
2632   */
2633   */
2634   /**
2635   * Set method for the messageWindow field.
2636   * creation date: (16-08-00 17:06:09)
2637   */
2638   ```
* @param newMessageWindow clientapp.MessageWindow

```java
public void setMessageWindow(MessageWindow newMessageWindow) {
    messageWindow = newMessageWindow;
}
```

* @param newTerminated boolean

```java
public void setTerminated(boolean newTerminated) {
    terminated = newTerminated;
}
```

```java
public void setTopicTree(javax.swing.tree.DefaultTreeModel newTree) {
    // reset selection to top row to avoid triggering the change listener and have a spurious message
    // inserted into the message area
    if (getDiscussionTree().isSelectionEmpty() == false) {
        getDiscussionTree().setSelectionRow(0);
    }
    getDiscussionTree().setModel(newTree);
    getDiscussionTree().revalidate();
    getDiscussionTree().repaint();
    // System.out.println("Indice de setTopicTree");
}
```

* @param newVisibleDocumentsList java.util.Hashtable

```java
public void setVisibleDocumentsList(java.util.Hashtable newVisibleDocumentsList) {
    visibleDocumentsList = newVisibleDocumentsList;
}
```

* Initialize the message properties box with the contents of the current message, if any.

```java
public void showMessageProperties() {
    if (getDiscussionTree().getSelectionCount() == 0) {
        ErrorBox noSelectionErrorBox = new ErrorBox();
        noSelectionErrorBox.setText(ClientConstants.NO_TOPIC_TREE_SELECTION_ERROR_MESSAGE);
        noSelectionErrorBox.show();
        return;
    }
    else if (getClientManager().verifyGroupConnectionManagement() == false) {
        return;
    }
    else if (getClientManager().getClientInfo().getCurrentMessage() != null) {
        MessagePropertiesBox messagePropertiesBox = new MessagePropertiesBox();
        messagePropertiesBox.initTable(getClientManager().getClientInfo().getCurrentMessage());
        messagePropertiesBox.show();
    }
}
```

```java
```
The message window is the log window that can be activate by double clicking the message bar at the bottom of the client application window. This method makes that message window visible.

* Creation date: (16-08-00 17:43:02)

```java
/**
 * public void showMessageWindow()
 * {
 * getMageWindow().setVisible( true );
 * }
 */

/**
 * Prepare termination of the client application and then dispose of it.
 * @ Creation date: (12-09-00 21:19:40)
 */
public void terminateAndClose()
{
 prepareTermination();
 getClientManager().prepareForTermination();
 this.dispose();

/**
 * Test method. Purely for development purposes. Will be removed in stable releases.
 * @ Creation date: (29-06-00 18:52:19)
 */
public void test()
{
 PeerViewClient client = new PeerViewClient("PeerViewClient");
 // Default client = new DefaultClient("/TEST");

 try {
 URLString url = URLString.createSessionURL("localhost", 4461 ,"socket","test");
 if (SessionFactory.sessionExists( url ))
 {
 // System.out.println("Session findes");
 }
 Session session = SessionFactory.createSession(client, url, true );
 if ( session.channelExists("test" ))
 {
 // System.out.println("channelExists");
 Channel channel = session.createChannel(client, "test", true, true, true);
 channel.addConsumer(client,client);
 }
 catch (JSDException JSDE)
 {
 getChannelManager().handleJSDException( JSDE );
 }

 /**
 * Change value field of the progress/message bar at the bottom of the client application main window.
 * @param newValue int
 */
public void updateMessageArea(int newValue)
{
ująMessageArea.getModel().setValue(newValue);
_ipvMessageArea.repaint();
 // ipvMessageArea.paintImmediately( ipvMessageArea.getBounds() );
 }

/**
 * Convenience method for updating the visible documents box.
 * @ Creation date: (30-08-00 22:30:36)
 */
public void updateVisibleDocumentsBox()

165
```java
{  
    updateVisibleDocumentsBox( getVisibleDocumentsList() );
}
/**
 * Update contents of visible documents box.
 * Creation date: (30-08-00 01:03:02)
 * @param documents java.util.Hashtable
 */
public void updateVisibleDocumentsBox(Hashtable documents)
{
    // temporarily disable itemlistener to avoid repeated triggering
    // the below ought to be substitutable with a call to removeAllItem but an attempt to do so seemed to
    // fail.
    setItemListenerActive( false );
    /* while ( getDocumentBox().getItemCount() > 0 )
    {
        getDocumentBox().removeItemAt( 0 );
    }
*/
    getDocumentBox().setModel( new DefaultComboBoxModel() );
    getDocumentBox().insertItem( new CaptionItem( ClientConstants.getVISIBLE_DOCUMENTS_BOX_CAPTION() )
        , 0 );
    Collection values = documents.values();
    Iterator iterator = values.iterator();
    while ( iterator.hasNext() )
    {
        ArrayList itemList = (ArrayList) iterator.next();
        Iterator itemListIterator = itemList.iterator();
        while ( itemListIterator.hasNext() )
        {
            Object item = itemListIterator.next();
            getDocumentBox().addItem( item );
            // System.out.println( item.toString() );
        }
    }  
    setItemListenerActive( true );
    getDocumentBox().setSelectedIndex( 0 );
}
}

Listing C.4: ClientInfo.java
package clientapp;
/**
 * Creation date: (02-07-00 18:41:23)
 * @author:
 */
import java.util.Hashtable;
import java.util.Properties;
import java.io.File;
import java.util.Hashtable;
import java.io.*;
import java.awt.datatransfer.*;
/**
   * The main entity class for the client application. This class holds data structures and fields that
   * must be shared by
   * several client application components.
   */
public class ClientInfo implements ClipboardOwner
{
    private com.sun.media.jndt.Session groupManagementSession = null;
    private static java.util.Properties preferences = null;
    private java.util.Hashtable groups = null;
    private com.sun.media.jndt.Channel groupManagementChannel = null;
    private peerviewmisc.workGroup activeGroup = null;
}
```
private java.lang.String currentDocumentName = null;
priate peerviewmisc.Message currentMessage = null;
private java.util.HasSet localFiles = new HashSet();
private java.awt.datatransfer.Clipboard clipboard = new Clipboard( ClientConstants.getCLIPBOARD_NAME () );
private boolean localMode = true;
/**
 * @ClientInfo constructor comment.
 */
public ClientInfo () {
    super();
    initialize();
}
/**
 * Create unique client id as concatenation of default name + system time in milliseconds + local host
 * IP address or
 * default name + system time if the IP address is unavailable.
 * @Creation date: (16-07-00 13:50:53)
 * @Exception java.lang.String
 */
public static String createUniqueClientID () {
    String ID = ClientInfo.getPreferences().getProperty( ClientConstants.getName() )+ (new java.util.Date () .toString() );
    try {
        ID = ID + java.net.InetAddress.getLocalHost().toString();
    } catch ( java.net.UnknownHostException UHE ) {
        System.err.println( UHE.toString() );
        System.exit( 0 );
    } return ID;
}
/**
 * Insert the method's description here.
 * @Creation date: (16-07-00 13:38:00)
 * @Exception java.lang.Throwable The exception description.
 */
protected void finalize () throws java.lang.Throwable {
    // write preferences
}
/**
 * Insert the method's description here.
 * @Creation date: (06-07-00 13:15:33)
 * @Return peerviewmisc.WorkGroup
 */
public peerviewmisc.WorkGroup getActiveGroup () {
    return activeGroup;
}
/**
 * Insert the method's description here.
 * @Creation date: (07-08-00 15:57:42)
 * @Return java.awt.datatransfer.Clipboard
 */
public java.awt.datatransfer.Clipboard getClipBoard () {
    return clipboard;
}
/**
 * Insert the method's description here.
 * @Creation date: (11-07-00 15:10:56)
 * @Return java.lang.String
 */
public java.lang.String getCurrentDocumentName () {
return currentDocumentName;
}
/**
 * Insert the method's description here.
 * Creation date: (19-07-00 17:30:43)
 * @return peerviewmisc.Message
 */
public peerviewmisc.Message getCurrentMessage() {
    return currentMessage;
}
/**
 * Insert the method's description here.
 * Creation date: (04-07-00 17:42:59)
 * @return com.sun.media.jsdt.Channel
 */
public com.sun.media.jsdt.Channel getGroupManagementChannel() {
    return groupManagementChannel;
}
/**
 * Insert the method's description here.
 * Creation date: (04-07-00 18:57:40)
 * @return com.sun.media.jsdt.Session
 */
public com.sun.media.jsdt.Session getGroupManagementSession() {
    return groupManagementSession;
}
/**
 * Insert the method's description here.
 * Creation date: (04-07-00 16:21:40)
 * @return java.util.Hashtable
 */
public java.util.Hashtable getGroups() {
    return groups;
}
/**
 * Insert the method's description here.
 * Creation date: (03-08-00 01:49:44)
 * @return java.util.HashSet
 */
public java.util.HashSet getLocalFiles() {
    return localFiles;
}
/**
 * Insert the method's description here.
 * Creation date: (02-07-00 22:08:51)
 * @return java.util.Properties
 */
public static java.util.Properties getPreferences() {
    if (preferences == null) {
        preferences = new Properties( ClientConstants.getDefaultPreferences() );
        // if the client has not yet been assigned a unique identifier, do so now. The new ID will be
        // saved to disk upon exit
        if ( ((String)ClientInfo.getPreferences().getProperty(ClientConstants.getClientName())).
            equalsIgnoreCase( ClientConstants.getDefaultClientName() )) {
            ClientInfo.getPreferences().put( ClientConstants.getClientName(), createUniqueClientID() );
            // System.out.println( ClientInfo.getPreferences().get( ClientConstants.getClientName() ));
        }
    }
    return preferences;
}
/**
 * Insert the method's description here.
* creation date: (16-07-00 13:36:57)
  */
public void initialize()
{
  
  /**
   * Insert the method's description here.
   * Creation date: (10-08-00 16:37:24)
   * @return boolean
   */
public boolean isLocalMode()
{
  return localMode;
}

/**
 * Obligatory method for meeting the ClipboardOwner interface specification
 * Creation date: (07-08-00 17:28:28)
 * @param clipboard java.awt.datatransfer.Clipboard
 * @param contents java.awt.datatransfer.Transferable
 */
public void lostOwnership(Clipboard clipboard, Transferable contents)
{
  
  /**
   * Read preferences from Properties object on disk. Each call to this function
   * results in a read from disk rather than an internal lookup, i.e. the values
   * read from disk are not cached as a class member.
   * This eliminates data redundancy and ensures consistency. The overhead is
   * negligible as the file in question is essentially plain text.
   * Algorithm:
   * 1. Construct new Properties object P
   * 2. Initialize P with default preferences
   * 3. Read user preferences from disk file
   * 4. IF read is successful THEN
   * 4.1. Store user preferences in P
   * 5. Return P
   * This approach will return a valid object even if the read fails or in case of
   * the application being used for the first time.
   * Creation date: (26-06-00 13:54:56)
   */
public static void readPreferences()
{
try
{
   FileInputStream f = new FileInputStream(ClientConstants.getPREFRENCES_FILE_NAME());
   preferences.load(f);
   f.close();
}
catch (FileNotFoundException ffe)
{
   // the exception can be handled by ignoring it since the default values
   // in p will substitute for what should have been read from file.
}
catch (IOException ie)
{
   // any IOException != FileNotFoundException should be dealt with in the
   // conventional manner
   System.err.println( ie.toString() );
   System.exit(0);
}
}

/**
 * Insert the method's description here.
 * Creation date: (06-07-00 13:15:33)
 * @param newActiveGroup peerviewmisc.WorkGroup
public void setActiveGroup(peerviewmisc.WorkGroup newActiveGroup) {
  activeGroup = newActiveGroup;
}

/**
 * Insert the method’s description here.
 * Creation date: (07-08-00 15:57:43)
 * @param newClipBoard java.awt.datatransfer.Clipboard
 */
public void setClipBoard(java.awt.datatransfer.Clipboard newClipBoard) {
  clipBoard = newClipBoard;
}

/**
 * Insert the method’s description here.
 * Creation date: (11-07-00 15:18:56)
 * @param newCurrentDocumentName java.lang.String
 */
public void setCurrentDocumentName(java.lang.String newCurrentDocumentName) {
  currentDocumentName = newCurrentDocumentName;
}

/**
 * Insert the method’s description here.
 * Creation date: (19-07-00 17:30:43)
 * @param newCurrentMessage peerviewmisc.Message
 */
public void setCurrentMessage(peerviewmisc.Message newCurrentMessage) {
  currentMessage = newCurrentMessage;
}

/**
 * Insert the method’s description here.
 * Creation date: (04-07-00 17:42:59)
 * @param newGroupManagementChannel com.sun.media.jsdt.Channel
 */
public void setGroupManagementChannel(com.sun.media.jsdt.Channel newGroupManagementChannel) {
  groupManagementChannel = newGroupManagementChannel;
}

/**
 * Insert the method’s description here.
 * Creation date: (02-07-00 18:57:40)
 * @param newGroupManagementSession com.sun.media.jsdt.Session
 */
public void setGroupManagementSession(com.sun.media.jsdt.Session newGroupManagementSession) {
  groupManagementSession = newGroupManagementSession;
}

/**
 * Insert the method’s description here.
 * Creation date: (04-07-00 16:21:40)
 * @param newGroups java.util.HashSet
 */
public void setGroups(java.util.HashSet newGroups) {
  groups = newGroups;
}

/**
 * Insert the method’s description here.
 * Creation date: (03-08-00 01:49:44)
 * @param newLocalFiles java.util.HashSet
 */
public void setLocalFiles(java.util.HashSet newLocalFiles) {
  localFiles = newLocalFiles;
}

/**
 * Insert the method’s description here.
 * Creation date: (10-08-00 16:37:24)
 * @param newLocalMode boolean
 */
public void setLocalMode(boolean newLocalMode) {
}
localMode = newLocalMode;
}

/**
 * Insert the method's description here.
 * Creation date: (02-07-00 22:08:51)
 * @param newPreferences java.util.Properties
 */
public static void setPreferences(java.util.Properties newPreferences) {
    preferences = newPreferences;
}

/**
 * Write preferences to disk file.
 * Creation date: (17-07-00 12:13:32)
 * @param preferences java.util.Properties
 */
public static void writePreferences()
{
    // write Properties object to disk
    try
    {
        FileOutputStream f = new FileOutputStream( ClientConstants.getPREFERENCES_FILE_NAME() );
        preferences.store( f, ClientConstants.getProperties_HEADER() );
        f.close();
    }
    catch (IOException ie)
    {
        System.err.println( ie.toString() );
        System.exit(0);
    }
}

package clientapp;

import java.io.*;
import javax.swing.JComponent;
import javax.swing.*;
import javax.swing.ImageIcon;
import javax.swing.JEditorPane;
import javax.swing.JTextArea;
import com.sun.media.jaffa.*;
import com.sun.media.jaffa.event.*;
import java.util.Properties;
import java.util.Hashtable;
import peerview.misc.*;
import java.util.zip.*;
import java.util.Enumeration;
import java.util.Iterator;
import java.util.Random;
import java.swing.tree.*;
import java.util.GregorianCalendar;
import java.awt.datatransfer.*;

/**
 * The main control class for the client application. This class holds methods and control structures
 * shared by several
 * client application components.
 * Creation date: (14-06-00 21:27:43)
 * @author:
 */
public class ClientManager
{
/* public class DocPackage implements Serializable

    String docAndPathName;
    java.awt.Rectangle bounds;

    public DocPackage( String docAndPathNameArg, java.awt.Rectangle boundsArg )
    {
        docAndPathName = docAndPathNameArg;
        bounds = boundsArg;
    }

    public String getDocAndPathName()
    {
        return docAndPathName;
    }

    public java.awt.Rectangle getBounds()
    {
        return bounds;
    }

    public void setDocAndPathName( String docAndPathNameArg )
    {
        docAndPathName = docAndPathNameArg;
    }

    public void setBounds( java.awt.Rectangle boundsArg )
    {
        bounds = boundsArg;
    }
*/

    /** Event handler for JSPI connections. Ensures automatic notification in case of connection failure
        */
    public class GroupManagementListener implements ConnectionListener
    {
        public void connectionFailed( ConnectionEvent ce )
        {
            groupManagementConnectionFailed( ce );
        }
    }

    // the below inner classes derived from Thread are used instead of member methods to allow execution
    // in separate threads.
    // This is necessary to allow the event-dispatching thread to process events and thus enable proper
    // updating of the message area progress bar
    // in the client main window. Initially, I used paintImmediately to achieve the desired effect but
    // that seemed to produce flicker on some platforms and
    // so had to be abandoned.
    /** An object of this class should be instantiated with a set of documents and launched in a separate
        thread so that
        * it executes without blocking the event dispatching thread and thus the GUI.
        */
    public class RemoveDistributedDocumentsFromPanorama extends Thread
    {
        public void run()
        {
            owner.lockInterface( true );
            java.util.Set keys = owner.getPanoramaPanel().getDocuments().keySet();
            Iterator iterator = keys.iterator();
            owner.initMessageArea( 1, owner.getPanoramaPanel().getDocuments().size(), 1, ClientConstants.
                getREMOVING_DISTRIBUTED_DOCUMENTS_MESSAGE() );
            // copy to array first to avoid ConcurrentModificationException
            // this should be considered an interim solution till I come up with a cleaner alternative
            String keyArray[] = new String[ keys.size() ];
for ( int i = 0; iterator.hasNext(); i++ )
{
    keyArray[i] = (String) iterator.next();
}
for ( int i = 0; i < keyArray.length; i++ )
{
    String key = keyArray[i];
edu.umd.cs.jazz.ZVisualLeaf leaf =
    (edu.umd.cs.jazz.ZVisualLeaf) owner.getPanoramaPanel().
    getDocuments().get( key );
    owner.getPanoramaPanel().removeGlobalDocument( (String) leaf.getClientProperty( 
        ClientConstants.getSENDER_NAME() ),
    (String) leaf.getClientProperty( ClientConstants. 
        getDOCUMENT_NAME() ));
    owner.advanceTask();
yield();
}
owner.getPanoramaPanel().getDocuments().clear();
owner.resetMessageArea();
owner.lockInterface( false );
}
/**
 * An object of this class should be instantiated with a set S of file names and launched in a
 * separate thread. It will * add the files whose names are in S to the panorama.
 */
public class AddDocuments extends Thread
{
    File newFiles[];
    public AddDocuments( File newFilesArg[] )
    {
        newFiles = newFilesArg;
    }
    public void run()
    {
        owner.lockInterface( true );
        boolean duplicateFlag = false;
        // prepare progress indicator for the document retrieval process
        owner.initMessageArea( 0, newFiles.length, 0, ClientConstants.RETRIEVING_DOCUMENTS() );
        java.util.Hashtable fileSet = new java.util.Hashtable();
        for ( int i = 0; i < newFiles.length; i++ )
        {
            owner.advanceTask();
            if ( getFileInfo().getLocalFiles().contains( newFiles[i] ) == true )
            {
                // if file is already in the panorama, raise the flag and continue with the next file in
                duplicateFlag = true;
                continue;
            }
            else
            {
                getFileInfo().getLocalFiles().add( newFiles[i] );
            }
            try
            {
                owner.getPanoramaPanel().addLocalDocument( readDocument( newFiles[i] ),
                ClientInfo.getPreferences().getProperty( ClientConstants. 
                    getName() ),
                ClientInfo.getPreferences().getProperty( ClientConstants. 
                    getAUTHOR_NAME() ),
                newFiles[i].getPath(),
                newFiles[i].lastModified() );
            }
        }
catch ( IOException IGE )
{

}
informServerOfNewDocument( newFiles[i].getPath() );
fileSet.add( newFiles[i] );
yield();
}
broadcastDocuments( fileSet );
updatePanorama();
// System.out.println("inde i addDocuments Atrd");
owner.resetMessageArea();
if ( duplicateFlag == true )
{
    MessageBox messagebox = new MessageBox();
    messagebox.setText( ClientConstants.getPANORAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE() );
    messagebox.show();
}
owner.lockInterface( false );
}

/** This class should be instantiated with a set S of files and launched in a separate thread. It
will remove * from the panorama the documents corresponding the members of S.
*/
public class RemoveDocuments extends Thread
{
    File[] removables;

    public RemoveDocuments( File[] removablesArg )
    {
        removables = removablesArg;
    }

    public void run()
    {
        owner.lockInterface( true );
        owner.getPanoramaPanel().setPanoramaLock( true );
        owner.initializeMessageArea(i, removables.length, i, ClientConstants.getREMOVING_DOCUMENTS() );
        java.util.Vector filePaths = new java.util.Vector();
        for ( int i = 0 ; i < removables.length; i++ )
        {
            owner.getPanoramaPanel().removeLocalDocument( ClientInfo.getPreferences().getProperty( ClientConstants.getName() ), removables[i].getPath() );
            getClientInfo().getLocalFiles().remove( removables[i] );
            filePaths.add( removables[i].getPath() );
            owner.advanceTask();
yield();
        }
        if ( getClientInfo().isLocalMode() == false )
        {
            broadcastDocumentRemovals( filePaths );
        }
        updatePanorama();
        owner.resetMessageArea();
        owner.getPanoramaPanel().setPanoramaLock( false );
        owner.lockInterface( false );
    }
}

/** This class should be instantiated with a recipient ID and author name and launched in a separate
thread. It * will send to the recipient the set of local documents currently displayed in the panorama.
*/
public class UnicastDocuments extends Thread
{
    String recipientName;
    String recipientAuthorName;
public UnicastDocuments( String recipientNameArg, String recipientAuthorNameArg )
{
    recipientName = recipientNameArg;
    recipientAuthorName = recipientAuthorNameArg;
}

public void run()
{
    if ( getClientInfo().getLocalFiles().size() > 0 )
    {
        WorkGroup activeGroup = getClientInfo().getActiveGroup();
        URLString url = null;
        Session session = null;
        Channel groupChannel = null;
        java.util.Vector rawDocumentData = new java.util.Vector();
        try
        {
            url = activeGroup.getSessionURL();
            session = SessionFactory.createSession( getGroupManagementClient(), url, false );
            groupChannel = session.createChannel( getGroupManagementClient(), activeGroup,
                getWorkGroupChannelID(), true, true, false );
        }
        catch ( JSDEException JSDTE )
        {
            handleJSDEException( JSDTE );
            return;
        }
        owner.initMessageArea(1, getClientInfo().getLocalFiles().size(), 1,
            ClientConstants.getUNICASTING_DOCUMENTS_MESSAGE( getClientInfo(),
                getLocalFiles().size(),
                recipientAuthorName ));
        Iterator iterator = getClientInfo().getLocalFiles().iterator();
        while ( iterator.hasNext() )
        {
            File file = (File)iterator.next();
            // read contents of file and compress it
            try
            {
                byte fileContents[] = readDocument( file );
                CompressedDocumentPackage compressedDocumentPackage = deflateDocument( fileContents );
                compressedDocumentPackage.setPath( file.getPath() );
                // separate name from path here to avoid confusion over separator char across platforms
                compressedDocumentPackage.setName( file.getPath().substring( file.getPath().lastIndexOf( File.separator ) + 1 ) );
                rawDocumentData.add( compressedDocumentPackage );
                owner.advanceTask();
                yield();
            }
            catch ( IOException IOE )
            {
                System.err.println( IOE.toString() );
                System.exit( 0 );
            }
        }
    }
}

try
{
    DataPackage outgoingPackage = new DataPackage();
    outgoingPackage.setContents( rawDocumentData );
    outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.getNEW_DOCUMENTS() );
    outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(), getClientInfo().getPreferences().getProperty( ClientConstants.getAUTHOR_NAME() ) );
    Data data = new Data( outgoingPackage );
    groupChannel.sendToClient( getGroupManagementClient(), recipientName, data );
}
catch (JSDEException jsde) {
    handleJSDEException(jsde);
    owner.setMessageArea();
}

/** Use this class to remove non-local documents from the panorama.
 * This class should be instantiated with the name of an author (to be included in the notification message shown
 * to the user), a sender (which will be used to identify the documents) and a set of file names.
 * Objects should be
 * launched in a separate thread.
 */
public class RemoveDistributedDocuments extends Thread {
    String authorName;
    String senderName;
    java.util.Vector filePaths;

    public RemoveDistributedDocuments(String authorNameArg, String senderNameArg, java.util.Vector
            filePathsArg)
    {
        authorName = authorNameArg;
        senderName = senderNameArg;
        filePaths = filePathsArg;
    }

    public void run()
    {
        owner.lockInterface(true);
        owner.initMessageArea(1, filePaths.size(), 1, ClientConstants.
                getMOVING_DISTRIBUTED/Documents_BY_REQUEST(authorName, filePaths.size()));
        Iterator iterator = filePaths.iterator();
        while (iterator.hasNext())
        {
            owner.getPanoramaPanel().removeGlobalDocument(senderName, (String) iterator.next());
            owner.advanceTask();
            yield();
        }
        updatePanorama();
        owner.resetMessageArea();
        owner.lockInterface(false);
    }
}

/** This class should be used to add non-local documents to the panorama. It accepts a documents
 * package sent by
 * another client and the ID of that client. Objects of this class should be launched in a separate
 * thread.
 */
public class AddDistributedDocuments extends Thread {
    DataPackage incomingPackage = null;
    String senderName = null;

    public AddDistributedDocuments(DataPackage incomingPackageArg, String senderNameArg)
    {
        incomingPackage = incomingPackageArg;
        senderName = senderNameArg;
    }

    public void run()
    {
        owner.lockInterface(true);
    }
}
java.util.Vector documents = (java.util.Vector) incomingPackage.getContents();
String authorName = incomingPackage.getProperties().getProperty( DataPackage.getAUTHOR_NAME() );
owner.setMessageArea( i, documents.size(), i, ClientConstants.getADDING_DISTRIBUTED_DOCUMENTS( authorName, documents.size() ));
for ( int i = 0; i < documents.size(); i++ )
{
        get( i );
    byte[] rawDocumentData = inflateDocument( compressedDocumentPackage );
    owner.getPanoramaPanel().addGlobalDocument( rawDocumentData, senderName, authorName, 
        compressedDocumentPackage.getPath(), compressedDocumentPackage.getName() );
    owner.advanceTask();
    yield();
}
updatePanorama();
owner.resetMessageArea();
owner.lockInterface( false );
}

/** Client object to be used with the group management session. */
public class GroupManagementClient implements Client, ChannelConsumer
{
    String name;

    public GroupManagementClient()
    {
    }

    public String getName()
    {
        return name;
    }

    public void setName( String nameArg )
    {
        name = nameArg;
    }

    /** Accept and parse incoming data. The if-else structure maps message types to functions. */
    public synchronized void dataReceived( Data data )
    {
        DataPackage incomingPackage = null;
        try
        {
            incomingPackage = (DataPackage) data.getDataAsObject();
        }
        catch ( Exception E )
        {
            System.err.println( E.toString() );
            System.exit(0);
        }
        if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase( ClientConstants.getGROUP_DIRECTORY() ) )
        {
            updateGroupDirectory( (HasTable) incomingPackage.getContents() );
        }
        else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase( ClientConstants.getNEW_DOCUMENT() ) )
        {
            addDistributedDocument( incomingPackage, data.getSenderName() );
        }
        else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase( ClientConstants.getNEW_DOCUMENTS() ) )
        {
```java
{
(new AddDistributedDocuments( incomingPackage, data.getSenderName() )).start();
}
else if ( incomingPackage.getProperties().getProperty( DataPackage.DESCRIPTION ) ).equalsIgnoreCase( ClientConstants.getRequestDocuments() )
{
(new UnicastDocuments( data.getSenderName(), incomingPackage.getProperties().getProperty( DataPackage.getAUTHOR_NAME() ) )).start();
}
else if ( incomingPackage.getProperties().getProperty( DataPackage.DESCRIPTION ) ).equalsIgnoreCase( ClientConstants.getTOC_TREE() )
{
setTopicTree( (DefaultTreeModel) incomingPackage.getProperties().getProperty( DataPackage.getTOC_TREE_PATH() ) );
}
else if ( incomingPackage.getProperties().getProperty( DataPackage.DESCRIPTION ) ).equalsIgnoreCase( ClientConstants.getMessage() )
{
setDisplayedMessage( (Message) incomingPackage.getProperties().getContents() );
}
else if ( incomingPackage.getProperties().getProperty( DataPackage.DESCRIPTION ) ).equalsIgnoreCase( ClientConstants.getREMOVE_DOCUMENT() )
{
removeDistributedDocument( incomingPackage.getProperties().getProperty( DataPackage.AUTHOR_NAME() ), data.getSenderName(), (String) incomingPackage.getProperties().getContents() );
}
else if ( incomingPackage.getProperties().getProperty( DataPackage.DESCRIPTION ) ).equalsIgnoreCase( ClientConstants.getREMOVE_DOCUMENTS() )
{
(new RemovedDistributedDocuments( incomingPackage.getProperties().getProperty( DataPackage.AUTHOR_NAME() ), incomingPackage.getProperties().getProperty( DataPackage.CLIENT_NAME() ), (java.util.Vector) incomingPackage.getProperties().getContents() ) ).start();
}
else if ( incomingPackage.getProperties().getProperty( DataPackage.DESCRIPTION ) ).equalsIgnoreCase( ClientConstants.getClient_LEFT_GROUP() )
{
owner.initMessageArea( 1, 1, 1, ClientConstants.getClient_LEFT_GROUP_MESSAGE( (String) incomingPackage.getProperties().getProperty( DataPackage.getAUTHOR_NAME() ) ) );
}
else if ( incomingPackage.getProperties().getProperty( DataPackage.DESCRIPTION ) ).equalsIgnoreCase( ClientConstants.getDocument_UPDATE() )
{
updateDistributedDocument( incomingPackage.getProperties().getProperty( DataPackage.getDOPATH() ), data.getSenderName(), (CompressedDocumentPackage) incomingPackage.getProperties().getContents() );
}
else if ( incomingPackage.getProperties().getProperty( DataPackage.DESCRIPTION ) ).equalsIgnoreCase( ClientConstants.getTOC_TREE_UPDATE() )
{
updateTopicTree( data.getSenderName(), (String) incomingPackage.getProperties().getProperty( DataPackage.getDOPATH() ), (DefaultTreeModel) incomingPackage.getProperties().getContents() );
}
else if ( incomingPackage.getProperties().getProperty( DataPackage.DESCRIPTION ) ).equalsIgnoreCase( ClientConstants.getUPDATE_PANORAMA() )
{
updatePanoramaByRequest();
}
}
public Object authenticate( AuthenticationInfo ai )
{
return null;
}
```
private ClientApplication owner;

private ClientInfo clientInfo = null;
private GroupManagementClient groupManagementClient = null;

/**
 * Constructor.
 *
 * @param contents byte[]
 * @param path java.lang.String[]
 */

public ClientManager(ClientApplication newOwner) {
    super();
    owner = newOwner;
    initialize();
}

/**
 * Inflate a data package received over a network connection and add it to
 * the panorama using the data package properties as descriptive
 * parameters.
 *
 * @param contents byte[]
 * @param path java.lang.String[]
 */

public void addDistributedDocument(DataPackage incomingPackage, String sender) {
    byte[] rawFileContents = inflateDocument((CompressedDocumentPackage) incomingPackage.getContents());
    owner.getPanoramaPanel().addGlobalDocument(rawFileContents, sender, incomingPackage.getProperties().getProperty(DataPackage.getAUTHOR_NAME()), incomingPackage.getProperties().getProperty(DataPackage.getDOCPATH()), incomingPackage.getProperties().getProperty(DataPackage.getDOCSNAME()));
    owner.initMessageArea(1, 1, ClientConstants.getADDED_DISTRIBUTED_DOCUMENT_MESSAGE(incomingPackage.getProperties().getProperty(DataPackage.getAUTHOR_NAME())), incomingPackage.getProperties().getProperty(DataPackage.getDOCPATH()), incomingPackage.getPanoramaProperties(), incomingPackage.getProperties().getProperty(DataPackage.getAUTHOR_NAME()));
}

/**
 * Add documents to the panorama by launching a separate thread to handle the task.
 *
 * @param newFiles java.io.File[]
 */

public void addDocuments(File[] newFiles) {
    AddDocuments addDocuments = new AddDocuments(newFiles);
    addDocuments.start();
    /* try
    * addDocuments.join();
    */
    catch (InterruptedException ie) {
        System.err.println(ie.toString());
    }
}

/**
 * Read a listing of document names from disk and add the corresponding documents to
 * the panorama.
 *
 * @param docs java.util.Vector
 */

public void addDocumentsFromDisk(java.util.Vector docs) {
    if (docs != null) {
        File files[] = new File[docs.size()];
        for (int i = 0; i < docs.size(); i++) {
        }
// the below assumes that the local documents set now contains all the newly added documents from
// the above call to
// addDocuments. A cleaner solution would be to merge the adding of files with the repositioning
// readDocumentList feature wasn’t included in the original design, so pending a revision of the
code, this should do OK.
for (int i = 0; i < docs.size(); i++)
{
    DocPackage docPackage = (DocPackage) docs.get( i );
    File file = new File( docPackage.getDocNameAndPath() );
    files[ i ] = file;
}
AddDocuments addDocuments = new AddDocuments( files );
addDocuments.start();
try
{
    addDocuments.join();
}
catch ( InterruptedException ie )
{
    System.err.println( ie.toString() );
}

public void broadcastDocumentRemovals(java.util.Vector files)
{
    Session session = null;
    Channel groupChannel = null;
    if ( ( getClientInfo().isLocalMode() == false ) &&
        ( files.size() > 0 ) )
    {
        WorkGroup activeGroup = getClientInfo().getActiveGroup();
        owner.initMessageArea(1, 1, ClientConstants.getBroadcastString_DOCUMENT_REMOVALS() );
        DataPackage outgoingPackage = new DataPackage();
        outgoingPackage.setContents( files );
        outgoingPackage.getProperties().setProperty( DataPackage.getDescription(), ClientConstants.
            getRemoveDocuments() );
        outgoingPackage.getProperties().setProperty( DataPackage.getWorkGroupID(), activeGroup.
            getWorkGroupID() );
        outgoingPackage.getProperties().setProperty( DataPackage.getAuthor_Name(),
            ClientInfo.getPreferences().getProperty( ClientConstants.
            getAuthor_Name() ) );
        outgoingPackage.getProperties().setProperty( DataPackage.getClient_Name(),
            null );
        outgoingPackage.getProperties().setProperty( DataPackage.getAuthor_Name(),
            null );
        owner.sendMessage( activeGroup, outgoingPackage, groupChannel );
    }
}
ClientInfo.getPreferences().getProperty( ClientConstants.getName() )
);

try {
    URLString url = getClientInfo().getActiveGroup().getSessionURL();
    Session session = SessionFactory.createSession( getGroupManagementClient(), url, false );
    GroupChannel groupChannel = session.createChannel( getGroupManagementClient(), activeGroup, 
        getWorkGroupChannelID(), true, true, false );
    Data data = new Data( outgoingPackage );
    groupChannel.sendToOthers( getGroupManagementClient(), data );
}

catch ( JSUException JSUVE ) {
    handleJSUException( JSUVE );
    owner.resetMessageArea();
}

}/**
 * Transmit a set of files to all members of the group that this client belongs to, if any.
 * @param files
 */
public void broadcastDocuments( java.util.HashSet files )
{
    if (( getClientInfo().isLocalMode() == false )&& ( files.size() > 0 ))
    {
        WorkGroup activeGroup = getClientInfo().getActiveGroup();
        URLString url = null;
        Session session = null;
        GroupChannel groupChannel = null;
        String clientNames[] = null;
        int numClientNames = 0;
        java.util.Vector rawDocumentData = new java.util.Vector();
        try {
            url = getClientInfo().getActiveGroup().getSessionURL();
            session = SessionFactory.createSession( getGroupManagementClient(), url, false );
            groupChannel = session.createChannel( getGroupManagementClient(), activeGroup, 
                    getWorkGroupChannelID(), true, true, false );
        }
        catch ( JSUException JSUVE ) {
            handleJSUException( JSUVE );
        }
        try {
            numClientNames = groupChannel.listClientNames().length;
        }
        catch ( JSUException JSUVE ) {
            handleJSUException( JSUVE );
        }
        // inform user that broadcast is in progress
        if ( numClientNames > 1 )
        {
            owner.initMessageArea(1, files.size(), 1, ClientConstants.getBroadcastingDocumentsMessage( 
                files.size() ));
        }
    
    Iterator iterator = files.iterator();
    while ( iterator.hasNext() )
    {
        File file = (File) iterator.next();
        // if more members than this client
        CompressedDocumentPackage compressedDocumentPackage = null;
        try
// read contents of file and compress it
byte fileContents[] = readDocument( file );
compressedDocumentPackage = deflateDocument( fileContents );
compressedDocumentPackage.setPath( file.getPath() );
// separate name from path here to avoid confusion over separator char across platforms.
compressedDocumentPackage.setName( file.getPath().substring( file.getPath().lastIndexOf( "File.
separator ")+1 ) );
//System.out.println( "compressedPackageName" + compressedDocumentPackage.getName() );
}
catch ( IOException IOE )
{
    System.err.println( IOE.toString() );
    System.exit( 0 );
}
rawDocumentData.add( compressedDocumentPackage );
if ( numClientNames > 1 )
{
    owner.advanceTask();
}
// submit raw document data
if ( numClientNames > 1 )
{
    DataPackage outgoingPackage = new DataPackage();
    outgoingPackage.setContents( rawDocumentData );
    outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
        getNEWDOCUMENTS() );
    outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(),
        ClientInfo.getPreferences().getProperty( ClientConstants.
            getAUTHOR_NAME() ) );
    outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), getClientInfo().
        getActiveGroup().getWorkGroupId() );
    try
    {
        Data data = new Data( outgoingPackage );
        // System.out.println( "Broadcasted" );
        groupChannel.sendToOthers( getGroupManagementClient(), data );
    }
    catch ( JSDEException JSDEE )
    {
        handleJSDEException( JSDEE );
    }
    if ( numClientNames > 1 )
    {
        owner.resetMessageArea();
    }
}
/*
 * Send a difference array to the other members of the group that the client belongs to, if any, so that
 * they can
 * update their copy of the relevant document. This method is called when a document has been modified
 * to keep the
 * remote copies in sync with the master at the author client.
 * @param compressedDifferenceBitmapPackage peerviewmisc.CompressedDocumentPackage
 */
public void broadcastDocumentUpdate( String documentName, CompressedDocumentPackage
    compressedDifferenceBitmapPackage )
{
    DataPackage outgoingPackage = new DataPackage();
    outgoingPackage.setContents( compressedDifferenceBitmapPackage );
outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
getDOCUMENT_UPDATE() );
outgoingPackage.getProperties().setProperty( DataPackage.getDOC_PATH(), documentName );
try
{
if ( getClientInfo().getActiveGroup() != null )
{
    URLString url = getClientInfo().getActiveGroup().getSessionURL();
    Session session = SessionFactory.createSession( getGroupManagementClient(), url, false );
    Channel groupChannel = session.createChannel( getGroupManagementClient(), getClientInfo().
getActiveGroup().getWorkGroupChannelID(), true, true, false );
    groupChannel.sendToOthers( getGroupManagementClient(), new Data( outgoingPackage ) );
}
}

/* Set all properties of the topic tree to null so that it appears empty to the user.
 * Creation date: (01-08-00 15:53:53)
 * public void clearTopicTree()
{
    owner.setMessage( null );
    getClientInfo().setCurrentMessage( null );
    getClientInfo().setCurrentDocumentName( null );
    owner.setTopictree( null );
}

/**
 * Computes a bitmap combination of two input bitmaps.
 * Starts by XORing together the first n bytes of the two bitmaps, where n is the length of the shorter
 * array.
 * Then appends the remainder of newRawFileContents, if any, to the resulting bitmap and returns it.
 * This method is meant for use in the document update phase, both in the transmitting and receiving end
 * Specifically, the sender computes the difference between an old and a new version of a document and
 * broadcasts it
 * in gzipped form to one or more recipients who then apply the below method to their old version and
 * the difference
 * bitmap, ending up with a transformed document identical to the sender’s version.
 * Creation date: (21-08-00 16:18:01)
 * @param oldRawFileContents byte[]
 * @param newRawFileContents byte[]
 * public byte[] computeBitmapCombination(byte[] oldRawFileContents, byte[] newRawFileContents)
{
    int shortestArrayLength;
    int longestArrayLength;
    if ( oldRawFileContents.length < newRawFileContents.length )
    {
        shortestArrayLength = oldRawFileContents.length;
        longestArrayLength = newRawFileContents.length;
    }
    else
    {
        shortestArrayLength = newRawFileContents.length;
        longestArrayLength = oldRawFileContents.length;
    }
    byte differenceBitmap[] = new byte[ longestArrayLength ];
    // XOR the two file contents arrays and store the result in the differenceBitmap array
    int i;
    for ( i = 0 ; i < shortestArrayLength; i++ )
    {
byte oldByte = oldRawFileContents[i];
byte newByte = newRawFileContents[i];
    // (a AND b) OR ( a AND b ) => XOR( a, b )
    differenceBitmap[i] = (byte) ((byte) ( oldByte | newByte )) &
        (byte) ( (byte).MAX_VALUE - oldByte) | (byte).MAX_VALUE - newByte));
    }
    // copy remainder of new file contents, if any, to differenceBitmap array
    for ( ; i < newRawFileContents.length; i++ )
    {
        differenceBitmap[i] = newRawFileContents[i];
    }
    return differenceBitmap;
}

/**
   * Perform various initialization maneuvers at start-up
   * creation date: (21-10-00 21:38:37)
   */
public void configure()
{
    connectToGroupManagement();
    requestGroupsFromServer();
    connectToGroup();
    addDocumentsFromDisk( readDocumentList() );
}

/**
   * Connect to the most recently joined group, if any, as specified in the preferences.
   * creation date: (09-08-00 23:48:20)
   */
public void connectToGroup()
{
    if ( getClientInfo().getGroups() != null )
    {
        String activeGroupUponExitID = (String) getClientInfo().getPreferences().getProperty( ClientConstants.
            getACTIVE_GROUP_UPON_EXIT() );
        WorkGroup workGroup = (WorkGroup) getClientInfo().getGroups().get( activeGroupUponExitID );
        if ( activeGroupUponExitID.equalsIgnoreCase( ClientConstants.getDEFAULT_GROUP_ID() ) == false )
        {
            connectToGroup( workGroup );
        }
        else
        {
            getClientInfo().setActiveGroup( null );
            informUserOfLocalMode();
        }
    }

    /**
    * Connect to the workgroup specified by the parameter, if possible.
    * creation date: (10-08-00 00:38:15)
    * @param groupURL com.sun.media.jsdt.URLString
    */
public void connectToGroup( WorkGroup workGroup )
{
    Session session;
    Channel channel;
    try
    {
        URLString url = workGroup.getSessionURL();
        session = SessionFactory.createSession( getGroupManagementClient(), url, true );
        channel = session.createChannel( getGroupManagementClient(),
            workGroup.getWorkGroupChannelID(),
            true, true, true);
        channel.addConsumer( getGroupManagementClient(), getGroupManagementClient() );
        getClientInfo().setActiveGroup( workGroup );
    }
getClientInfo().setLocalMode( false );
// broadcast to other group members
broadcastDocuments( getClientInfo().getLocalFiles() );
// request documents from other members of group
requestDocuments();
// announce successful join
informUserOfGroupJoin( workGroup );
}
catch ( JSNTEException JSNTE )
{
    handleJSNTEException( JSNTE );
}

/**
 * Connect the client to the server via the group management session.
 * Creation date: (02-07-00 19:16:42)
 */
public void connectToGroupManagement()
{
    Session session = null;
    Channel groupManagementChannel = null;
    if ( getClientInfo().getGroupManagementSession() == null )
    {
        try
        {
            // disconnect first if already connected
            if ( getClientInfo().getGroupManagementSession() != null )
            {
                disconnectFromGroup( getClientInfo().getActiveGroup() );
            }
            clientInfo.getGroupManagementSession().leave( getGroupManagementClient() );
            //clientInfo.getGroupManagementChannel().leave( getGroupManagementClient() );
            clientInfo.setGroupManagementSession( null );
            clientInfo.setGroupManagementChannel( null );
        }
        URLString url = URLString.createSessionURL( (String)clientInfo.getPreferences().getProperty(
            ClientConstants.getServer_NAME() ),
            Integer.parseInt( (String) (clientInfo.getPreferences().getProperty( ClientConstants.getServer_PORT() ))),
            (String) clientInfo.getPreferences().getProperty( ClientConstants.getConnection_TYPE() ),
            clientInfo.getGroupManagementSession().getGroupManagement_SESSION_NAME() )
        if ( SessionFactory.getSessionExists( url ) )
        {
            session = SessionFactory.createSession( getGroupManagementClient(), url, true);
groupManagementChannel = session.createChannel( getGroupManagementClient(), ClientConstants.
group_CHANNEL_NAME(), true, true, true);
groupManagementChannel.addConsumer( getGroupManagementClient(), getGroupManagementClient() );
        }
        else
        {
            informUserOfLocalMode();
            owner.initMessageArea( 1, 1, 1, ClientConstants.getCONNECTED_TO_GROUP_MANAGEMENT() );
        }
    }
    catch ( NameInUseException NIE )
    {
        owner.initMessageArea( 1, 1, 1, ClientConstants.getFAILED_TO_CONNECT_TO_SERVER() );
    }
    informServerofExit();
}
catch ( JSDEException jsde )
{
    handleJSDEException( jsde );
    informUserOfLocalMode();
}

getClientInfo().setGroupManagementSession(session);
getClientInfo().setGroupManagementChannel(groupManagementChannel);
}

/**
 * Copy the text selected in the visual component associated with the selectedNode parameter to the client
 * clipboard.
 * Creation date: (07-08-00 17:01:17)
 * @param selectedNode edu.umd.cs.jazz.ZVisualLeaf
 */
public void copyText( edu.umd.cs.jazz.ZVisualLeaf selectedNode )
{
    JScrollPane selectedPane = ( JScrollPane ) ( JView ) selectedNode.getVisualComponent() .getComponent ();
    // if the selected node is capable of containing text. The check is mostly to be on the safe side; it is
    // the caller tries to ensure that the copy text menu item is invoked only on text documents.
    if ( selectedPane.getViewport() .getView() instanceof javax.swing.text.JTextComponent )
    {
        String selectedText = ( ( javax.swing.text.JTextComponent ) selectedPane.getViewport() .getView() ).
            getSelectedText();
        if ( selectedText != null )
        {
            ( ( javax.swing.text.JTextComponent ) selectedPane.getViewport() .getView() ).copy();
            owner.initMessageArea( 1, 1, 1, ClientConstants .COPYED_TEXT_MESSAGE() );
        }
        else
        {
            owner.initMessageArea( 1, 1, 1, ClientConstants .NO_TEXT_TO_COPY_MESSAGE() );
        }
    }
}

/**
 * Submit a request to the server that a group be created and added to the group directory.
 * Creation date: (17-07-00 17:28:50)
 * @param workGroup peerviewmisc.WorkGroup
 */
public void createWorkGroup( WorkGroup workGroup )
{
    DataPackage outgoingPackage = new DataPackage();
    outgoingPackage.setContent( workGroup );
    outgoingPackage.getProperties().setProperty( DataPackage .DESCRIPTION(), ClientConstants .REQUEST_ADDROUP() );
    try
    {
        if ( getClientInfo() .getGroupManagementChannel() != null )
        {
            getClientInfo() .getGroupManagementChannel().sendToClient( ( ( WorkGroupClient )
                ClientInfo .getPreferences() .getProperty( ClientConstants .
                    getSERVER_NAME() ),
                    new Data( outgoingPackage ) );
        }
    }
    catch ( JSDEException jsde )
    {
        handleJSDEException( jsde );
    }
}
/**
 * Compress and package the contents of an array representing a document.
 * Creation date: (06-07-00 20:49:16)
 * @return CompressedDocumentPackage
 * @param uncompressedContents byte
 */
public CompressedDocumentPackage deflateDocument(byte[] uncompressedContents)
{
    Deflater deflater = new Deflater( Integer.parseInt( ClientInfo.getPreferences().getProperty( ClientConstants.getCOMPRESSION_LEVEL() ) ));
    byte[] compressedContents = new byte[ uncompressedContents.length ];
    deflater.setInput( uncompressedContents, 0 , uncompressedContents.length );
    deflater.finish();
    int compressedSize = deflater.deflate( compressedContents );
    byte[] compressedAndCondensedContents = new byte[ compressedContents.length ];
    for ( int i = 0 ; i < compressedSize ; i++ )
    {
        compressedAndCondensedContents[ i ] = compressedContents[ i ];
    }
    CompressedDocumentPackage compressedDocumentPackage = new CompressedDocumentPackage();
    compressedDocumentPackage.setCompressedContents( compressedAndCondensedContents );
    compressedDocumentPackage.setCompressedLength( compressedSize );
    compressedDocumentPackage.setUncompressedLength( uncompressedContents.length );
    return compressedDocumentPackage;
}
/**
 * Leave a work group (passed as argument).
 * Creation date: (10-08-00 21:14:06)
 * @param workGroup peerviewmisc.WorkGroup
 */
public void disconnectFromGroup(WorkGroup workGroup)
{
    if ( workGroup != null )
    {
        // broadcastDocumentRemovals( getClientInfo().getLocalFiles() );
        // the next step is carried out to allow the broadcasts to conclude - they are apparently
        // disrupted if the leave is executed prematurely
        removeDistributedDocumentsFromPanorama();
        // obtain reference to the session that this client currently subscribes to and leave it. This
        // will automatically
        // expel the client from any objects subordinate to that session.
        try
        {
            URLString url = workGroup.getSessionURL();
            Session session = SessionFactory.createSession( getGroupManagementClient(), url, false );
            session.leave( getGroupManagementClient() );
        }
        catch ( JSDEException JSDTE )
        {
            System.err.println( JSDTE.toString() );
            System.exit( 0 );
        }
    }
}
/**
 * Insert the method's description here.
 * Creation date: (02-07-00 19:07:18)
 * @return clientapp.ClientInfo
 */
public ClientInfo getClientInfo()
{
    if ( clientInfo == null )
    {
```java
{
    clientInfo = new ClientInfo();
    return clientInfo;
}
/**
 * Insert the method's description here.
 * Creation date: (02-07-00 22:53:11)
 * @return clientapp.GroupManagementClient
 */
public GroupManagementClient getGroupManagementClient()
{
    if (groupManagementClient == null)
    {
        groupManagementClient = new GroupManagementClient();
        groupManagementClient.setName( ClientInfo.getPreferences().getProperty( ClientConstants.getName() ));
    }
    return groupManagementClient;
}
/**
 * Insert the method's description here.
 * Creation date: (09-10-00 12:53:32)
 * @param ce com.sun.media.jsdt.event.ConnectionEvent
 */
public void handleException(ConnectionEvent ce)
{
    owner.initializeArea( 1, 1, 1, ClientConstants.getConnectionFailure( ce.toString() ));
}
/**
 * Handle the different subclasses of JSDTException, i.e. the different types of exception that JSDT
 * actions may throw,
 * by issuing appropriate messages and taking corrective or conclusive action.
 * Creation date: (08-08-00 16:34:28)
 * @param exception com.sun.media.jsdt.JSDTException
 */
public void handleException(JSDTException exception)
{
    String message = (String) ClientConstants.getJSDT_EXCEPTION_MESSAGES_TABLE().get( exception.getClass().toString() );
    message += "\n\n" + exception.toString() + "\n"
    System.err.println( message );
    owner.initializeArea( 1, 1, message );
}
/**
 * Inflate the contents of a compressed document.
 * Creation date: (08-08-00 01:56:27)
 */
public byte[] inflateDocument( CompressedDocumentPackage compressedDocumentPackage )
{
    byte uncompressedData[] = new byte[ compressedDocumentPackage.getUncompressedLength() ];
    try
    {
        Inflater inflator = new Inflater();
        inflator.setInput( compressedDocumentPackage.getCompressedContents() );
        int inflatedSize = inflator.inflate( uncompressedData );
    }
    catch ( DataFormatException DFE )
    {
        System.err.println( DFE.toString() );
        System.exit( 0 );
        // System.out.println( "Deflated: " + compressedDocumentPackage.getCompressedLength() + " Inflated : " + compressedDocumentPackage.getUncompressedLength() );
    }
    return uncompressedData;
```
public void informGroupMembersOfDocumentRemoval(String fullyQualifiedDocumentName)
{
  DataPackage outgoingPackage = new DataPackage();
  outgoingPackage.setContents( fullyQualifiedDocumentName );
  outgoingPackage.getProperties().setProperty( DataPackage.DESCRIPTION(), ClientConstants.
      getClient_REMOVED_LOCAL/Documents() );
  try {
    if ( getClientInfo().isLocalMode() == false )
      {
        URLString url = clientInfo.getActiveGroup().getSessionURL();
        Session session = SessionFactory.createSession( getGroupManagementClient(), url, false );
        Channel channel = session.createChannel( getGroupManagementClient(), clientInfo.getActiveGroup() 
            .getWorkGroupChannelID(), true, true, false );
        channel.sendToOthers( getGroupManagementClient(), new Data( outgoingPackage ) );
      }
  }
  catch ( JSUTException JSUTE )
    { handleJSUTException( JSUTE ); }
  }
-folded-

public void informGroupMembersOfPanoramaUpdate()
{
  if ( getClientInfo().isLocalMode() == false )
  {
    URLString url;
    Session session;
    Channel groupChannel = null;
    try {
      url = getClientInfo().getActiveGroup().getSessionURL();
      session = SessionFactory.createSession( getGroupManagementClient(), url, false );
      groupChannel = session.createChannel( getGroupManagementClient(), getClientInfo().
          getActiveGroup().getWorkGroupChannelID(), true, true, false );
      DataPackage outgoingPackage = new DataPackage();
      outgoingPackage.getProperties().setProperty( DataPackage.DESCRIPTION(), ClientConstants.
          getUPDATE_PANORAMA() );
      groupChannel.sendToOthers( getGroupManagementClient(), new Data( outgoingPackage ) );
    }
    catch ( JSUTException JSUTE )
      { handleJSUTException( JSUTE ); }
  }
-folded-

public void informServerOfExit()
```java
{ DataPackage outgoingPackage = new DataPackage();

outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
    getCLIENT_LEFT() );

try {
    if ( getClientInfo().getGroupManagementChannel() != null )
    {
        getClientInfo().getGroupManagementChannel().sendToClient( getGroupManagementClient(),
            ClientInfo.getPreferences().getProperty( ClientConstants.getServer_NAME() ),
            new Data( outgoingPackage ));
    }
}

catch ( JSUTException JSUTE )
{
    handleJSUTException( JSUTE );
}

} /*
   * Inform the server that the client has exited the group to which it belonged, if any.
   * Creation date: (31-10-00 23:58:01)
   */

public void informServerOfGroupExit()
{
    if ( getClientInfo().getActiveGroup() != null )
    {
        DataPackage outgoingPackage = new DataPackage();

        outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
            getCLIENT_LEFT() );

        outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), getClientInfo().
            getActiveGroup().getWorkGROUP_ID() );

        try {
            if ( getClientInfo().getGroupManagementChannel() != null )
            {
                getClientInfo().getGroupManagementChannel().sendToClient( getGroupManagementClient(),
                    ClientInfo.getPreferences().getProperty( ClientConstants.getServer_NAME() ),
                    new Data( outgoingPackage ));
            }
        }

        catch ( JSUTException JSUTE )
        {
            handleJSUTException( JSUTE );
        }
    }
}

/*
   * Inform the server that the client has added one or more documents to the panorama.
   * Creation date: (04-10-00 11:39:50)
   * @param docPath java.lang.String
   */

public void informServerOfNewDocument(String docPath)
{
    DataPackage outgoingPackage = new DataPackage();

    outgoingPackage.setContents( docPath );
    outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
        getCLIENT_ADDEDDocuments() );

    outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(), ClientInfo.getPreferences
        ().getProperty( ClientConstants.getAUTHOR_NAME() ) );

    try {
```
if ( getClientInfo().getGroupManagementChannel() != null )
{
    getClientInfo().getGroupManagementChannel().sendToClient( 
        getGroupManagementClient(),
        ClientInfo.getPreferences().getProperty( ClientConstants.SERVER_NAME() ),
        new Data( outgoingPackage ));
}

} catch ( JSUTException jsute )
{
    handleJSUTException( jsute );
}

/**
 * Display a message in the client progress bar indicating the client has joined a specific work group.
 * Creation date: (26-07-00 12:34:45)
 * @param workGroup peerviewmisc.WorkGroup
 */
public void informUserOfGroupJoin(WorkGroup workGroup)
{
    if (this client is the only group member)
    {
        if ( workGroup.getParticipants().size() <= 1 )
        {
            owner.initMessageArea( 1, 1, ClientConstants.CLIENT_JOINED_GROUP_MESSAGE( workGroup ));
        }
        else
        {
            owner.initMessageArea( 1, 1, ClientConstants.CLIENT_JOINED_GROUP_WITH_MEMBERS_MESSAGE( workGroup ));
        }
    }

    /**
     * Display a notification message that the client is operating in local mode, i.e. disconnected from the server.
     * Creation date: (10-08-00 04:34:36)
     */
public void informUserOfLocalMode()
{
    if ( getClientInfo().isLocalMode() == false )
    {
        MessageBox messagebox = new MessageBox();
        messagebox.setText( ClientConstants.LOCAL_MODE_MESSAGE() );
        messagebox.show();
        getClientInfo().setLocalMode( true );
    }

    /**
     * Insert the method’s description here.
     * Creation date: (04-07-00 17:02:27)
     */
public void initialize()
{
    clearTopicTree();

    /**
     * Leave the group management communication channel.
     * Creation date: (20-10-00 15:43:39)
     */
public void leaveGroupManagement()
{
    try
    {
        if ( getClientInfo().getGroupManagementChannel() != null )
        {
            getClientInfo().getGroupManagementChannel().leave( getGroupManagementClient() );
        }
    }
if ( getClientInfo().getGroupManagementSession() != null )
{
    getClientInfo().getGroupManagementSession().leave( getGroupManagementClient() );
}

} catch ( JSDEException e )
{
    handleJSDEException( e );
}

/**
 * Perform termination housekeeping such as updating preferences and
 * sundry other chores.
 * Creation date: (09-08-00 23:23:16)
 */
public void prepareforTermination()
{
    java.awt.Dimension size = owner.getSize();
    java.awt.Point location = owner.getLocation();

    ClientInfo.getPreferences().setProperty( ClientConstants.FRAME_HORIZONTAL_SIZE(),
        String.valueOf( (int) size.getWidth() ) );

    ClientInfo.getPreferences().setProperty( ClientConstants.FRAME_VERTICAL_SIZE(),
        String.valueOf( (int) size.getHeight() ) );

    ClientInfo.getPreferences().setProperty( ClientConstants.FRAME_X_COORDINATE(),
        String.valueOf( (int) location.getX() ) );

    ClientInfo.getPreferences().setProperty( ClientConstants.FRAME_Y_COORDINATE(),
        String.valueOf( (int) location.getY() ) );

    if ( getClientInfo().isLocalMode() == false )
    {
        ClientInfo.getPreferences().setProperty( ClientConstants.ACTIVE_GROUP_UPON_EXIT(),
            getClientInfo().getActiveGroup().getWorkGroupID() );

        getClientInfo().writePreferences();

        writeDocumentList();

        leaveGroupManagement();

        owner.getPanoramaPanel().getUpdateTimer().stop();

        try
        {
            System.runFinalization();

            Runtime.getRuntime().exit(0);
        }

        catch ( Exception E )
        {
            System.out.println( ClientConstants.COULD_NOT_EXIT_PROPERLY() );
        }
    }

    */

    * Creation date: (14-06-00 22:34:24)
    * @param newFileNames java.lang.File[]
    */

    public byte[] readDocumentFile( File newFile ) throws IOException
    {
        ByteArrayInputStream input = new FileInputStream( newFile );
        try
        {
            return readDocumentFile( newFile );
        }
        catch (IOException e)
        {
            System.err.println( ClientConstants.failedToAccessDocument( newFile.tostring() ));
            System.exit(1); //SKALE NORS: fejlededelelse åb pe fremkomme i status area i client window
        }
    }
return rawFileContents;

/**
 * Read the list of documents from disk and insert the file names it contains into a vector.
 * Creation date: (06-09-00 21:17:16)
 */
public java.util.Vector readDocumentList()
{
    java.util.Vector docs = null;
    try {
        FileInputStream fis = new FileInputStream(ClientConstants.getDocumentListFilename());
        ObjectInputStream ois = new ObjectInputStream( fis );
        try {
            docs = (java.util.Vector) ois.readObject();
        }
        catch ( ClassNotFoundException CNFE )
        {
            System.err.println( CNFE.toString() );
        }
        ois.close();
    }
    catch ( IOException ioe )
    {
        System.err.println( ClientConstants.getCouldNotReadDocumentList() );
        owner.setMessageArea( 1, 1, 1, ClientConstants.getCouldNotReadDocumentList() );
    }
    return docs;
}

/**
 * Remove from the panorama all documents originating from other members of the group to which this
 * client currently belongs.
 * Creation date: (15-07-00 16:23:13)
 * @param senderName java.lang.String
 * @param fullyQualifiedDocumentName java.lang.String
 */
public void removeDistributedDocument(String authorName, String senderName, String documentPath)
{
    owner.setMessageArea( 1, 1, 1, ClientConstants.getRemovingDistributedDocument( documentPath,
           authorName ) );
    owner.getPanoramaPanel().removeGlobalDocument( senderName, documentPath );
    owner.getPanoramaPanel().revalidate();
    owner.getPanoramaPanel().repaint();
}

/**
 * Remove all distributed documents from the local panorama and clear the hashtable indexing them.
 * Creation date: (26-07-00 02:54:40)
 */
public void removeDistributedDocumentsFromPanorama()
{
    new RemoveDistributedDocumentsFromPanorama() .start();
    owner.getPanoramaPanel().zoomToOverview();
}

/**
 * Launch a removal process for a set of documents in a separate thread (to allow concurrent updating of
 * the GUI).
 * Creation date: (03-10-00 11:23:07)
 * @param removables java.io.File[]
 */
public void removeDocuments(File[] removables)
{
    new RemoveDocuments( removables ) .start();
}

/*
Submit a request to the server that it removes a specific group from the group directory.

Creation date: (26-07-00 18:57:18)

@returns workGroup peerviewmisc.WorkGroup

```java
public void removeWorkGroup(workGroup workGroup)
{
    DataPackage outgoingPackage = new DataPackage();
    outgoingPackage.setContents( workGroup );
    outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
            getREQUEST_REMOVE_GROUP() );
    try
    {
        if ( getClientInfo().getGroupManagementChannel() != null )
            getClientInfo().getGroupManagementChannel().sendToClient( getGroupManagementClient(),
                    ClientInfo.getPreferences().getProperty( ClientConstants.
                            getSERVER_NAME() ),
                    new Data( outgoingPackage ) );
    }
    catch ( JSUtException JSDTE )
    {
        handleJSDTEException( JSDTE );
    }
```
* Request the directory of groups from the server. Assumes that group management is up and running.
* The directory will be received by the groupManagementClient data member.
* Creation date: (04-07-00 16:27:33)

```java
public void requestGroupsFromServer()
{
    DataPackage outgoingPackage = new DataPackage();
    outgoingPackage.setContents( null );
    outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
        getREQUEST_GROUP_DIRECTORY() );
    try
    {
        if ( getClientInfo().getGroupManagementChannel() != null )
        {
            for ( int i = 0; i < getClientInfo().getGroupManagementChannel().listConsumerNames().length; i ++)
            {
                // System.out.println( getClientInfo().getGroupManagementChannel().listConsumerNames()[i] );
                getClientInfo().getGroupManagementChannel().sendToClient( groupManagementClient,
                   ClientInfo.getPreferences(),getProperty( ClientConstants.
                   getSERVER_NAME() ),
                   new Data( outgoingPackage ));
            }
        }
        catch ( JSDEException JSDE )
        {
            handleJSDEException( JSDE );
        }
    }
    /**
    * Request a specific message from the message database maintained by the server.
    * Creation date: (10-07-00 02:11:23)
    * @param messageID java.lang.String
    */
    public void requestMessageFromServer(String messageID)
    {
        DataPackage outgoingPackage = new DataPackage();
        outgoingPackage.setContents( null );
        outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
            getREQUEST_MESSAGE() );
        outgoingPackage.getProperties().setProperty( DataPackage.getMessageID(), messageID );
        try
        {
            if ( getClientInfo().getGroupManagementChannel() != null )
            {
                for ( int i = 0; i < getClientInfo().getGroupManagementChannel().listConsumerNames().length; i ++)
                {
                    // System.out.println( getClientInfo().getGroupManagementChannel().listConsumerNames()[i] );
                    clientInfo.getGroupManagementChannel().sendToClient( groupManagementClient,
                        ClientInfo.getPreferences(),getProperty( ClientConstants.
                        getSERVER_NAME() ),
                        new Data( outgoingPackage ));
                }
            }
            catch ( JSDEException JSDE )
            {
                handleJSDEException( JSDE );
            }
        }
    }
    /**
    * Request that the server submit the current topic tree for the document designated by documentName.
    * Creation date: (09-07-00 18:36:18)
    */
```
public void requestTopicTree(String documentName)
{
    System.out.println("Initiating requestTopicTree");
    DataPackage outgoingPackage = new DataPackage();
    outgoingPackage.setContents(documentName);
    outgoingPackage.getProperties().setProperty(DataPackage.DESCRIPTION, ClientConstants.
        getREQUEST_TOPIC_TREE());
    try {
        if (getClientInfo().getGroupManagementChannel() != null)
        {
            getClientInfo().getGroupManagementChannel().sendToClient(getGroupManagementClient(),
                ClientInfo.getPreferences().getProperty(ClientConstants.
                    getSERVER_NAME()),
                new Data(outgoingPackage));
        }
    }
    catch (JSDEException jsde)
    {
        handleJSDEException(jsde);
    }
    /**
     * Notify the server that the client added a message.
     * Creation date: (11-07-00 21:23:22)
     */
    public void sendMessage(Message newMessage)
    {
        DataPackage outgoingPackage = new DataPackage();
        outgoingPackage.setContents(newMessage);
        outgoingPackage.getProperties().setProperty(DataPackage.DESCRIPTION, ClientConstants.
            getCLIENT_ADDED_MESSAGE());
        try {
            if (getClientInfo().getGroupManagementChannel() != null)
            {
                clientInfo.getGroupManagementChannel().sendToClient(getGroupManagementClient(),
                    ClientInfo.getPreferences().getProperty(ClientConstants.
                        getSERVER_NAME()),
                    new Data(outgoingPackage));
            }
        }
        catch (JSDEException jsde)
        {
            handleJSDEException(jsde);
        }
    }
    /**
     * Submit a request to the server that a specific message be deleted.
     * Creation date: (12-07-00 12:06:06)
     */
    public void sendMessageDeletion(String messageId)
    {
        DataPackage outgoingPackage = new DataPackage();
        outgoingPackage.setContents(messageId);
        outgoingPackage.getProperties().setProperty(DataPackage.DESCRIPTION, ClientConstants.
            getDELETE_MESSAGE());
        try {
            if (getClientInfo().getGroupManagementChannel() != null)
            {
public void sendTopicTreeUpdate( DefaultTreeModel newTree )
{
    DataPackage outgoingPackage = new DataPackage();
    outgoingPackage.setContents( newTree );
    outgoingPackage.getProperties().setProperty( DataPackage.DESCRIPTION, ClientConstants.TOPIC_TREE_UPDATE );
    outgoingPackage.getProperties().setProperty( DataPackage.DOCPATH, ClientInfo().getCurrentDocumentName() );
    try
    {
        if ( getClientInfo().getGroupManagementChannel() != null )
        {
            clientInfo.getGroupManagementChannel().sendToClient( ClientInfo().getPreferences().getProperty( ClientConstants.SERVER_NAME ),
                new Data( outgoingPackage ));
        }
    }
    catch ( JSDUException jsduE )
    {
        handleJSDUException( jsduE );
    }
}

/**
 * Insert the method's description here.
 * Creation date: (02-07-00 19:07:18)
 * @param newClientInfo clientapp.ClientInfo
 */
public void setClientInfo(ClientInfo newClientInfo) {
    clientInfo = newClientInfo;
}

/**
 * Update the text area for message display in the discussion panel at the bottom of the client main window.
 * Creation date: (10-07-00 02:30:47)
 * @param newMessage peerviewmisc.Message
 */
public void setDisplayedMessage(Message newMessage)
{
    owner.setMessage( newMessage );
    getClientInfo().setCurrentMessage( newMessage );
}

/**
 * Insert the method's description here.
 * Creation date: (02-07-00 22:53:11)
 * @param newGroupManagementClient clientapp.GroupManagementClient
 */
public void setGroupManagementClient( GroupManagementClient newGroupManagementClient )
{
    groupManagementClient = newGroupManagementClient;
}
/*
 * Insert the method's description here.
 */
public void setTopicTree(DefaultTreeModel newTree, String documentName)
{
    setDisplayedMessage( null );
    clientInfo().setCurrentDocumentName( documentName );
    owner.setTopicTree( newTree );
}

/**
 * Update one of the distributed documents shown in the panorama.
 */
public void updateDistributedDocument(String documentName, String senderName, CompressedDocumentPackage
    compressedDifferenceBitmapPackage)
{
    owner.initMessageArea( 1, 1, 1, ClientConstants.UPDATING_DISTRIBUTED_DOCUMENT( documentName ));
    byte rawFileContents[] = inflateDocument( compressedDifferenceBitmapPackage );
    owner.getPanoramaPanel().updateDistributedDocument( senderName, documentName, rawFileContents );
}

/**
 * Extract difference bitmap by XORing old and new file contents. Then submit the result as a deflated,
 * i.e. gziped,
 * data package to other members, if any, of active group.
 */
public void updateDocument(String path, String clientId, byte[] oldRawFileContents, byte[]
    newRawFileContents)
{
    byte differenceBitmap[] = computeBitmapCombination( oldRawFileContents, newRawFileContents );
    CompressedDocumentPackage compressedDifferenceBitmapPackage = deflateDocument( differenceBitmap );
    broadcastDocumentUpdate( path, compressedDifferenceBitmapPackage );
}

/**
 * Insert the method's description here.
 */
public void updateGroupDirectory(Hashtable newDirectory)
{
    clientInfo().setGroups( newDirectory );
    owner.getGroupDirectory1().updateTable();
    if ( getclientInfo().getActiveGroup() != null )
    {
        getclientInfo().setActiveGroup( (workGroup) newDirectory.get( getclientInfo().getActiveGroup().
            getWorkGroupID() ) );
    }
}

/**
 * Insert the method's description here.
 */
public void updatePanorama()
{
    owner.updateVisibleDocumentsBox();
}
// System.out.println("updatePanorama");
owner.getPanoramaPanel().validateLayout();
*/
/**
 * Insert the method's description here.
 * Creation date: (31-08-00 02:14:49)
 */
public void updatePanoramaByRequest()
{
    owner.initMessageArea( 1, 1, 1, ClientConstants.getUPDATE_PANORAMA_BY_REQUEST_MESSAGE() );
    updatePanorama();
}
/**
 * Insert the method's description here.
 * Creation date: (27-08-00 21:21:06)
 * @param fullyQualifiedDocumentName java.lang.String
 * @param treeUpdate java.swing.tree.DefaultTreeModel
 */
public void updateTopicTree(String senderName, String documentName, DefaultTreeModel treeUpdate)
{
    String fqName = ClientConstants.createFullyQualifiedName( senderName, documentName );
    if ( getClientInfo().getCurrentDocumentName().equals( fqName )
    
    setTopicTree( treeUpdate, fqName );
    }
    */
/**
 * Test connection to group management and notify user if it isn't active.
 * Creation date: (09-10-00 18:57:46)
 */
public boolean verifyGroupConnectionManagement()
{
    if ( getClientInfo().getGroupManagementSession() == null )
    
    initMessageArea( 1, 1, 1, ClientConstants.getNOT_CONNECTED_TO_SERVER() );
    return false;
    else
    
    return true;
    }
    */
/**
 * Extract a list of documents displayed in the panorama and write it to disk. This can then be restored
 * automatically
 * the next time the client is started.
 * Creation date: (04-09-00 21:48:43)
 */
public void writeDocumentList()
{
    try
    
    FileOutputstream fos = new FileOutputStream( ClientConstants.getDocumentList_FILENAME() );
    ObjectOutputStream oos = new ObjectOutputStream( fos );
    java.util.Vector docs = new java.util.Vector();
    Iterator iterator = owner.getPanoramaPanel().getLocalDocuments().values().iterator();
    while ( iterator.hasNext() )
    
    edu.uml.cs.jazz.ZVisualLeaf leaf = (edu.uml.cs.jazz.ZVisualLeaf) iterator.next();
    DocPackage docPackage = new DocPackage( (String) leaf.getClientProperty( ClientConstants.
    getDOCUMENT_NAME() ),
    new java.awt.Dimension( (int) leaf.getVisualComponent().getBounds().
    getMinX() );
    }
Listing C.6: ClientConstants.java

```java
package clientapp;

import java.awt.Color;
import java.util.Hashtable;
import edu/umd.cs.jazz.*;
import edu/umd.cs.jazz.component.*;
import edu/umd.cs.jazz.event.*;
import java.util.Properties;
import peerview.misc.SharedConstants;
import java.util.Random;

/**
 * This class contains constants specific to the client application as well as those derived from the {
 * @link peerview.misc.SharedConstants SharedConstants} class.
 * @creation date: (15-06-00 22:57:59)
 * @author:
 */
public class ClientConstants extends SharedConstants {

    private final static Color BACKGROUND_COLOUR = new Color(200, 200, 200);
    public final static java.lang.String PROCESSING(documents, to, the, panorama);
    public final static java.lang.String MESSAGE_AREA_DEFAULT_STRING = "Message,area:1,dialog,here,
    to_open, in, separate, window."
    private final static int UPDATE_INTERVAL_IN_MILLISECS = 1;
    private final static java.lang.String RETRIEVING(documents, to, panorama);

    private final static java.lang.String PLAIN_TEXT = "text/plain";
    private final static java.lang.String HTML = "text/html";
    private final static java.lang.String RTF = "text/rtf";
    private final static java.lang.String BITMAP = "bitmap";
    private final static java.lang.String ALL_FILES = "All_files";
    private final static java.lang.String[] FileTypes = { "*.txt", "Plain_text(.txt)", PLAIN_TEXT,
    {"*.html", "HTML",".html",".htm"}, HTML,
    {"*.rtf", "RTF",".rtf"}, RTF,
    {"*.cpp", "C++_source(.cpp),.cc", PLAIN_TEXT},
    {"*.h", "C++_header(.h)", PLAIN_TEXT,
    {"*.java", "Java_source(.java)", PLAIN_TEXT,
    {"*.gif", "GIF,(.gif)", BITMAP,
    {"*.jpg" }, "JPG,(.jpg)", BITMAP } );
```

200
private final static java.lang.String[] FILE_TYPE_SYNONYM_TABLE = {
    "\".txt\"",
    "\".html\", "\".htm\",
    "\".ruf\",
    "\".cpp\", "\".cc\",
    "\".h\",
    "\".java\",
    "\".gif\",
    "\".jpg\"
};

private final static java.lang.String OUT_OF_BOUNDS_EXCEPTION_TEXT = "Could not add all documents to panorama due to layout restrictions";
private final static int FILE_EXTENSION_INDEX = 0;
private final static int FILE_TYPE_DESCRIPTION_INDEX = 1;
private final static int TYPE_INDEX = 2;

private final static java.lang.String REMOVING_DOCUMENTS = "Removing documents from panorama";

private final static java.lang.String[] DISPLAY_QUALITIES = {
    "Low", String.valueOf(ZDrawingSurface.RENDER_QUALITY_LOW),
    "Medium", String.valueOf(ZDrawingSurface.RENDER_QUALITY_MEDIUM),
    "High", String.valueOf(ZDrawingSurface.RENDER_QUALITY_HIGH)
};

public final static java.lang.String GRID_LAYOUT = "GridLayout";

public final static java.lang.String UPDATE_INTERVAL = "UpdateInterval";

private final static java.lang.String NAME = "unique_client_ID";
private final static java.lang.String SIGNATURE = "Signature";

private final static java.lang.String DISPLAY_QUALITY = "DisplayQuality";

private final static java.lang.String LAYOUT_SCHEME = "LayoutScheme";

public final static java.lang.String LOCAL_HOST = "localhost";

public final static java.lang.String ANIMATION_DURATION_IN_MILLISECONDS = "AnimationDuration";

private final static java.lang.String AUTHOR_NAME = "AuthorName";

private final static java.lang.String DEFAULT_DOCUMENT_WIDTH = "DefaultDocumentWidth";

private final static java.lang.String DEFAULT_DOCUMENT_HEIGHT = "DefaultDocumentHeight";

private final static java.lang.String DEFAULT_HORIZONTAL_DOCUMENT_SPACING = "DefaultHorizontalDocumentSpacing";

private final static java.lang.String DEFAULT_VERTICAL_DOCUMENT_SPACING = "DefaultVerticalDocumentSpacing";

private final static java.lang.String GRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS = "GridLayoutDefaultNumberOfRows";

private final static java.lang.String GRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS = "GridLayoutDefaultNumberOfColumns";

private final static java.lang.String FRAME_HORIZONTAL_SIZE = "FrameHorizontalSize";

private final static java.lang.String FRAME_VERTICAL_SIZE = "FrameVerticalSize";

private final static java.lang.String DEFAULT_FRAME_HORIZONTAL_SIZE = "0";

private final static java.lang.String DEFAULT_FRAME_VERTICAL_SIZE = "0";

private final static java.lang.String FRAME_X_COORDINATE = "FrameXCoordinate";

private final static java.lang.String FRAME_Y_COORDINATE = "FrameYCoordinate";

private final static java.lang.String DEFAULT_FRAME_X_COORDINATE = "0";

private final static java.lang.String DEFAULT_FRAME_Y_COORDINATE = "0";

private final static java.lang.String ACTIVE_GROUP_UPON_EXIT = "ActiveGroupUponExit";

private final static java.lang.String ZOOM_TO_OVERVIEW_ON_UPDATE = "ZoomToOverviewOnUpdate";

private final static java.lang.String ZOOM_TO_OVERVIEW = "zoomToOverviewOnUpdate";

private final static java.lang.String DOCUMENTS_BOX = "ShowFullPathNamesInVisibleDocumentsBox";

private final static java.lang.String TRUE = "true";

private final static java.lang.String FALSE = "false";

private final static java.lang.String DEFAULT_GRID_LAYOUT_DOCUMENT_WIDTH = "DefaultGridLayoutDocumentWidth";
private final static java.lang.String DEFAULT_GRID_LAYOUT_DOCUMENT_HEIGHT = "Default_grid_layout документ,Height";
private final static java.lang.String DEFAULT_GRID_LAYOUT_HORIZONTALSpacing = "Default_grid_layout, horizontal_spacing";
private final static java.lang.String DEFAULT_GRID_LAYOUT_VERTICAL_SPACING = "Default grid layout, vertical spacing";
private final static java.lang.String LOAD_DOCUMENT_LIST_ON_STARTUP = "Load document list on startup";
private final static java.lang.String[] DEFAULT_PREFERENCES = {
    "UPDATE_INTERVAL", "5",
    "NAME", "DEFAULT_CLIENT_NAME",
    "AUTHOR_NAME", "?",
    "SIGNATURE", """,
    "DISPLAY_QUALITY", "High",
    "LAYOUT_SCHEME", "GRID_LAYOUT",
    "SERVER_NAME", "DEFAULT_SERVER_NAME",
    "SERVER_PORT", String.valueOf(DEFAULT_SERVER_PORT),
    "CONNECTION_TYPE", "DEFAULT_CONNECTION_TYPE",
    "ANIMATION_DURATION_IN_MILLISECONDS", "1000",
    "DEFAULT документов ширина", "400",
    "DEFAULT DOCUMENT_HEIGHT", "500",
    "DEFAULT_HORIZONTAL DOCUMENT SPACING", "500",
    "DEFAULT Vertical DOCUMENT SPACING", "500",
    "GRID LAYOUT DEFAULT_NUMBER OF ROWS", "10",
    "GRID LAYOUT DEFAULT_NUMBER OF COLUMNS", "10",
    "FRAME_HORIZONTAL_SIZE", "DEFAULT_FRAME_HORIZONTAL_SIZE",
    "FRAME_VERTICAL_SIZE", "DEFAULT_FRAME_VERTICAL_SIZE",
    "FRAME X COORDINATE", "DEFAULT_FRAME X COORDINATE",
    "FRAME Y COORDINATE", "DEFAULT_FRAME Y COORDINATE",
    "ACTIVE GROUP UPON EXIT", "DEFAULT_GROUP ID",
    "Zoom to Overview on Update", "Zoom to Overview",
    "getCOMPRESSION_LEVEL()", String.valueOf(
        getDEFAULT_COMPRESSION_LEVEL() ),
    "SHOW FULL PATH NAMES IN VISIBLE DOCUMENTS BOX", "FALSE",
    "DEFAULT_GRID LAYOUT DOCUMENT_WIDTH", "500",
    "DEFAULT_GRID LAYOUT DOCUMENT_HEIGHT", "600",
    "DEFAULT GRID LAYOUT HORIZONTAL SPACING", "100",
    "DEFAULT GRID LAYOUT vertical SPACING", "100",
    "LOAD DOCUMENT_LIST ON_STARTUP", "true",
    "getREGISTRY_PORT()", "getDEFAULT_REGISTRY_PORT()",
    "getAUTOLOCK_NODES()", "false",
    "getMESSAGE LOG_FILE_NAME()", "message.log"
};

private final static java.lang.String PREFERENCES_FILE_NAME = "prefs.ini";
private final static java.lang.String[] LAYOUT_SCHEMES = {"Grid\layout"};
private final static java.lang.String PROPERTIES_HEADER = "PeerView\client\settings";
private final static double DEFAULT_PANORAMA_PERCENTUAL_SIZE = 100;

private final static int LOCAL DOCUMENT = 0;
public final static java.lang.String DISTRIBUTED_DOCUMENT = "Distributed\document";
private final static java.lang.String REQUEST DOCUMENTS = "Request\documents";

private final static java.lang.String DOCUMENT_PATH = "Document\path";
private final static java.lang.String NO_TOPIC_TREE_SELECTION_ERROR_MESSAGE = "You must select an item from the topic tree.
private final static double SCALE_FACTOR_SCALING = 100.0;
public final static java.lang.String INVALID DELETION ATTEMPTED MESSAGE = "You can only delete messages that are composed by you and which have not been responded to.
public final static java.lang.String MESSAGE_PROPERTIES NAME COLUMN HEADING = "Name"
public final static java.lang.String MESSAGE_PROPERTIES VALUE COLUMN HEADING = "Value"

private final static int GROUP_DIRECTORY_COLUMN_WIDTH = 200;
public final static java.lang.String NO GROUP SELECTED TO JOIN MESSAGE = "You must select a group from the list before joining.

private final static java.lang.String GROUP AT MAXIMUM MEMBERS MESSAGE = "You cannot join the selected group because it has reached its maximum number of participants",
private final static java.lang.String ALREADY JOINED TO GROUP MESSAGE = "You have already joined the selected group.";

202
public final static java.lang.String BROADCASTING_DOCUMENT_REMOVALS = "Instructing the other group members to remove your documents from their panormas.";
private final static java.lang.String NO_GROUP_SELECTED_TO_EDIT_MESSAGE = "You must select a group from the list before you can edit.";
private final static java.lang.String GROUP_HAS_ACTIVE_PARTICIPANTS_ERROR_MESSAGE = "The group cannot be deleted because it currently has one or more active participants.";
private final static java.util.Hashtable MEDIA_TYPES = new Hashtable();
private final static java.awt.Color RESIZE_RECTANGLE_PEN_COLOUR = Color.green;
private final static java.awt.Color COLOR_MATE_BORDER_COLOUR = Color.gray;
private final static int DOCUMENT_BORDER_WIDTH = 6;
private final static java.awt.Color DOCUMENT_LINE_BORDER_COLOUR = Color.lightGray;
private final static int ZOOM_VECTOR_WIDTH = 2;
private final static float RENDER_CUTOFF = (float)0.1;
private final static java.lang.String PANORAMA_ALREADY_CONTAINED DOCUMENTS_MESSAGE = "One or more of the files you attempted to add were already part of the panorama.";
private final static double FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE = 0.9;
private final static java.lang.String CLIPBOARD_NAME = "Peerview clipboard";
private final static java.lang.String COPIED_TEXT_MESSAGE = "Copied text to clipboard, no text will be pasted into a message.";
private final static java.lang.String NO_TEXT_TO_COPY_MESSAGE = "No text was copied, since none was selected.";
private final static int RESET_MESSAGE_AREA_DELAY = 7000;
private final static java.lang.String CONTENTS_AND_SIGNATURE_DIVIDER = "\\n\\n\\n";
private final static java.lang.String LOCAL_NODE_MESSAGE = "Unable to connect to server or the most recently joined group. Please try joining another group or connect to a server."
private final static java.lang.String ADDITIONAL_CONNECTION_CONFIRMATION_MESSAGE = "There is currently no connection to the server. Should a connection attempt be made?";
private final static int GROUP_DIRECTORY_DELAY = 5000;
private final static java.lang.String REQUESTING_GROUPS_DIRECTORY = "Requesting group directory from server..."
private final static java.lang.String FAILED_TO_RECEIVE_GROUP_DIRECTORY = "Failed to receive group directory, directory information may still arrive shortly.";
private final static java.lang.String CONNECTED_TO_GROUP_MANAGEMENT = "Successfully connected to server.";
private final static java.lang.String FAILED_TO_CONNECT_TO_SERVER = "Could not connect to server.";
private final static java.lang.String CONFIRM_CONNECT_TO_SERVER = "Connecting to a new server will disconnect you from the group, and server you are currently connected to. If you do not want to continue, select No.";
private final static java.lang.String TITLE_REQUIRED_MESSAGE = "You must provide a title for your message before it can be submitted.";
private final static java.lang.String NOTHING_TO_REMOVE_MESSAGE = "Nothing to remove.";
private final static java.lang.String GROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS = "The group cannot be deleted; it has one or more active participants.";
private final static java.lang.String GROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS = "This group cannot be edited; it has one or more active participants.";
private final static java.lang.String REMOVING/Documents MESSAGE = "Removing documents from the panorama...";
private final static java.lang.String LAST_MODIFIED = "Last modified.";
private final static java.lang.String DOCUMENT_NAME = "Document name";
private final static java.lang.String RAW_BITMAP = "Raw bitmap";
private final static java.lang.String DOCUMENT_UPDATE = "Document update";
private final static java.lang.String UPDATING/Documents = "Now analyzing your documents and broadcasting updates as necessary to the other group members.";
private final static int ANIMATION_MAX_SPEED = 7000;
private final static int ANIMATION_MIN_SPEED = 300;
private final static int UPDATE_NOTIFICATION_THRESHOLD = 15;
private final static java.lang.String VISIBLE_DOCUMENTS_BOX_CAPTION = "Documents in panorama";
private final static int VISIBLE_DOCUMENTS_ICON_TEXT_GAP = 8;
private final static java.lang.String VISIBLE_DOCUMENTS_TAB_SIZE = "
private final static java.lang.String SENDER_NAME = "Sender name";
private final static java.lang.String FINISHED_ADDING_DOCUMENTS = "Finished adding documents";
private final static java.awt.Color VISIBLE_DOCUMENTS_SELECTION_COLOUR = new Color(0, 0, 255);
private final static java.lang.String UPDATE_PANORAMA = "Update panorama";
private final static java.lang.String UPDATE_PANORAMA_BY_REQUEST_MESSAGE = "Updating the panorama by request from other group member."
private final static java.awt.Dimension VISIBLE_DOCUMENTS_ICON_MAX_SIZE = new java.awt.Dimension(350, 30);
private final static java.lang.String ABOUT_BOX_MESSAGE = "Release 0.900 (beta).";
private final static int MAX_GRID_LAYOUT_DOCUMENT_HEIGHT = 2000;
private final static int MIN_GRID_LAYOUT_DOCUMENT_HEIGHT = 300;
private final static int MAX_GRID_LAYOUT_DOCUMENT_WIDTH = 1650;
private final static int MIN_GRID_LAYOUT_DOCUMENT_WIDTH = 400;
private final static int MAX_GRID_LAYOUT_HORIZONTAL_SPACING = 600;
private final static int MIN_GRID_LAYOUT_HORIZONTAL_SPACING = 0;
private final static int MAX_GRID_LAYOUT_VERTICAL_SPACING = 0;
private final static int MIN_GRID_LAYOUT_VERTICAL_SPACING = 0;
private final static java.lang.String DOCUMENT_LIST_FILE_NAME = "doList.dat";
private final static java.lang.String COULD_NOT_READ_DOCUMENT_LIST = "Unable to read and process the list of documents from disk."
private final static java.lang.String COULD_NOT_WRITE_DOCUMENT_LIST = "Could not write the list of documents to disk."
private final static int TEXT_DOCUMENT_THRESHOLD = 1000000;
private final static java.lang.String NOT_CONNECTED_TO_SERVER = "This action cannot be completed because there is currently no connection to the server."
private final static int MAX_DOCUMENT_LENGTH = 76;
private final static java.lang.String NODE_LOCKED = "Node locked."
private final static java.lang.String AUTOLOCK_NODES = "Autolock nodes."
private final static java.lang.String MESSAGE_LOG_FILE_NAME = "Message_log_filename"
private final static java.lang.String MESSAGE_LOG_WRITE_ERROR = "Could not write the message log to disk."
private final static java.lang.String MESSAGE_LOG_WRITE_MESSAGE = "Wrote message log to disk."
private final static java.lang.String NOTIFYING_SERVER_OF_NAME_IN_USE = "Your access to the server is blocked. Requesting a clean-up..."
private final static java.lang.String CLIENT_DOCUMENTATION_HELPER_NAME = "preview.hs";
private final static java.lang.String COULD_NOT_INITIALIZE_HELP = "Could not initialize the help system. " Online help will not be available.";
private final static java.lang.String CLIENT_HELP_PATH = "help";
private final static java.util.Hashtable FILE_TYPE_SYNONYM = null;

/**
232  * ClientMessages constructor comment.
233  */
234  public ClientConstants()
235  {
236    super();
237    initialize();
238  }
239  /**
240   * Insert the method's description here.
241   * Creation date: (15-06-00 23:12:13)
242   */
243  public String failedToAccessDocument(String fileName)
244  {
245    return "An error occurred while trying to access the document with filename \"" + fileName + \\n.3 .3 )-
246  }
247  /**
248  * Insert the method's description here.
249  * Creation date: (02-09-00 12:29:51)
250  */
251  public final static java.lang.String ABOUT_BOX_MESSAGE() {
return ABOUT_BOX_MESSAGE;

/**
 * Insert the method's description here.
 * Creation date: (09-08-00 23:05:17)
 * @return java.lang.String
 */
public final static java.lang.String getACTIVE_GROUP_UPON_EXIT() {
    return ACTIVE_GROUP_UPON_EXIT;
}

/**
 * Insert the method's description here.
 * Creation date: (19-09-00 20:59:44)
 * @param documentName java.lang.String
 * @param senderName java.lang.String
 */
public final static String getADDED_DISTRIBUTED_DOCUMENT_MESSAGE(String documentName, String senderName) {
    return "Received," + documentName + "," + senderName;
}

/**
 * Insert the method's description here.
 * Creation date: (06-09-00 14:50:29)
 * @return java.lang.String
 * @param authorName java.lang.String
 * @param numDocs int
 */
public final static String getADDING_DISTRIBUTED_DOCUMENTS(String authorName, int numDocs) {
    return "Adding," + numDocs + ",", + senderName;
}

/**
 * Insert the method's description here.
 * Creation date: (06-10-00 23:06:26)
 * @return java.lang.String
 */
public final static java.lang.String getALL_FILES() {
    return ALL_FILES;
}

/**
 * Insert the method's description here.
 * Creation date: (25-07-00 15:52:24)
 * @return java.lang.String
 */
public final static java.lang.String getALREADY_JOINED_TO_GROUP_MESSAGE() {
    return ALREADY_JOINED_TO_GROUP_MESSAGE;
}

/**
 * Insert the method's description here.
 * Creation date: (13-07-00 13:08:57)
 * @return java.lang.String
 */
public final static java.lang.String getANIMATION_DURATION_IN_MILLISECONDS() {
    return ANIMATION_DURATION_IN_MILLISECONDS;
}

/**
 * Insert the method's description here.
 * Creation date: (22-08-00 17:53:01)
 * @return double
 */
public final static int getANIMATION_MAX_SPEED() {
    return ANIMATION_MAX_SPEED;
}

/**
 * Insert the method's description here.
 * Creation date: (22-08-00 17:56:11)
 */
```java
* @return double
*/
public final static int getANIMATION_MIN_SPEED() {
  return ANIMATION_MIN_SPEED;
}
/**
 * Insert the method's description here.
 * Creation date: (10-08-00 15:38:52)
 * @return java.lang.String
*/
public final static java.lang.String getATTEMPT_CONNECTION_CONFIRMATION_MESSAGE() {
  return ATTEMPT_CONNECTION_CONFIRMATION_MESSAGE;
}
/**
 * Insert the method's description here.
 * Creation date: (17-07-00 13:52:34)
 * @return java.lang.String
*/
public final static java.lang.String getAUTHOR_NAME() {
  return AUTHOR_NAME;
}
/**
 * Insert the method's description here.
 * Creation date: (14-10-00 11:31:51)
 * @return java.lang.String
*/
public final static java.lang.String getAUTOLOCK_NODES() {
  return AUTOLOCK_NODES;
}
/**
 * Insert the method's description here.
 * Creation date: (16-06-00 17:55:28)
 * @return int
*/
public final static Color getBACKGROUND_COLOUR() {
  return BACKGROUND_COLOUR;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 20:22:01)
 * @return java.lang.String
*/
public final static java.lang.String getBITMAP() {
  return BITMAP;
}
/**
 * Insert the method's description here.
 * Creation date: (26-07-00 14:29:56)
 * @return java.lang.String
*/
public final static java.lang.String getBROADCASTING_DOCUMENT_REMOVALS() {
  return BROADCASTING_DOCUMENT_REMOVALS;
}
/**
 * Insert the method's description here.
 * Creation date: (08-07-00 19:00:37)
 * @return java.lang.String
 * @param number0fDocuments int
 * @param number0fRecipients int
*/
public static String getBROADCASTINGDOCUMENTS_MESSAGE(int number0fDocuments) {
  return "Broadcasting,\n" + String.valueOf( number0fDocuments )+ "document(s)";
}
/**
 * Insert the method's description here.
 * Creation date: (08-07-00 19:00:37)
*/
```
* @return java.lang.String
* @param numberofDocuments int
* @param numberofRecipients int
*
* public static String getBROADCDSTRING_DOCUMENTS_MESSAGE(int numberofDocuments, int numberofRecipients) {
    return "Broadcasting to other group members: " + String.valueOf(numberofDocuments) + ", document(s) to you" + String.valueOf(numberofRecipients) + ", recipient(s).";
}

/**
 * Insert the method’s description here.
 */
* @return java.lang.String
*
* public final static java.lang.String getClient_DOCUMENTATION_HELPSET_NAME() {
* return CLIENT_DOCUMENTATION_HELPSET_NAME;
*}

/**
 * Insert the method’s description here.
 */
* @return java.lang.String
*
* public final static java.lang.String getClient_HELP_PATH() {
* return CLIENT_HELP_PATH;
*}

/**
 * Insert the method’s description here.
 */
* @param workGroup peerviewmisc.WorkGroup
* return getClient_JOINED_GROUP_MESSAGE(peerviewmisc.WorkGroup workGroup) {
* return "You have now joined, " + workGroup.getGroupName() + ", ";
*}

/**
 * Insert the method’s description here.
 */
* @param workGroup peerviewmisc.WorkGroup
* return getClient_JOINED_GROUP_WITH_MEMBERS_MESSAGE(peerviewmisc.WorkGroup workGroup) {
* return "You have now joined, " + workGroup.getGroupName() + ", " + "The other members have been notified, and should now be submitting their documents if any, to you."
*}

/**
 * Insert the method’s description here.
 */
* @param clientsAuthorName java.lang.String
* return getClient_LEFT_GROUP_MESSAGE(String clientsAuthorName) {
* return clientsAuthorName + " has left the group, and his/her documents have been removed from your panorama."
*}

/**
 * Insert the method’s description here.
 */
* @return java.lang.String
*
* public final static java.lang.String getCLIPBOARD_NAME() {
* return CLIPBOARD_NAME;
*}

/**
 * Insert the method’s description here.
 */
* @return java.lang.String
* return java.lang.String
* @param numberofDocuments int
* @param numberofRecipients int
*
* public static String getBROADCDSTRING_DOCUMENTS_MESSAGE(int numberofDocuments, int numberofRecipients) {
    return "Broadcasting to other group members: " + String.valueOf(numberofDocuments) + ", document(s) to you" + String.valueOf(numberofRecipients) + ", recipient(s).";
}

/**
 * Insert the method’s description here.
 */
* @return java.lang.String
*
* public final static java.lang.String getClient_DOCUMENTATION_HELPSET_NAME() {
* return CLIENT_DOCUMENTATION_HELPSET_NAME;
*}

/**
 * Insert the method’s description here.
 */
* @return java.lang.String
*
* public final static java.lang.String getClient_HELP_PATH() {
* return CLIENT_HELP_PATH;
*}

/**
 * Insert the method’s description here.
 */
* @param workGroup peerviewmisc.WorkGroup
* return getClient_JOINED_GROUP_MESSAGE(peerviewmisc.WorkGroup workGroup) {
* return "You have now joined, " + workGroup.getGroupName() + ", ";
*}

/**
 * Insert the method’s description here.
 */
* @param workGroup peerviewmisc.WorkGroup
* return getClient_JOINED_GROUP_WITH_MEMBERS_MESSAGE(peerviewmisc.WorkGroup workGroup) {
* return "You have now joined, " + workGroup.getGroupName() + ", " + "The other members have been notified, and should now be submitting their documents if any, to you."
*}

/**
 * Insert the method’s description here.
 */
* @param clientsAuthorName java.lang.String
* return getClient_LEFT_GROUP_MESSAGE(String clientsAuthorName) {
* return clientsAuthorName + " has left the group, and his/her documents have been removed from your panorama."
*}

/**
 * Insert the method’s description here.
 */
* @return java.lang.String
* return java.lang.String
* @param numberofDocuments int
* @param numberofRecipients int
* */

207
```java
/*
 * public final static java.lang.String getCONFIRM_CONNECT_TO_SERVER() {
 * return CONFIRM_CONNECT_TO_SERVER;
 * }
 */
  
  /**
   * Insert the method's description here.
   * Creation date: (12-07-00 11:09:46)
   * @return java.lang.String
   * @param messageID java.lang.String
   */
  public static String getCONFIRM_MESSAGE_DELETION(String messageID) {
    return "Delete:\u201d + messageID + ",\u201d";
  }

  /**
   * Insert the method's description here.
   * Creation date: (20-07-00 20:41:02)
   * @param groupName java.lang.String
   */
  public static String getCONFIRM_WORKGROUP_DELETION(String groupName) {
    return "Delete,\u201d the grup:\u201d + ",\u201d + groupName + ",\u201d";
  }

  /*
   * Insert the method's description here.
   * Creation date: (10-08-00 20:39:43)
   * @return java.lang.String
   */
  public final static java.lang.String getCONNECTED_TO_GROUP_MANAGEMENT() {
    return CONNECTED_TO_GROUP_MANAGEMENT;
  }

  /*
   * Insert the method's description here.
   * Creation date: (09-10-00 12:57:14)
   * @return java.lang.String
   * @param connectionEventString java.lang.String
   */
  public static final String getConnectionFailure(String connectionEventString) {
    return "Connection failure,\u201d communication with the server has been disrupted,\u201d", +
      connectionEventString + ",\u201d";
  }

  /*
   * Insert the method's description here.
   * Creation date: (08-08-00 10:28:46)
   * @return java.lang.String
   */
  public final static java.lang.String getCONTENTS_AND_SIGNATURE_DIVIDER() {
    return CONTENTS_AND_SIGNATURE_DIVIDER;
  }

  /*
   * Insert the method's description here.
   * Creation date: (07-08-00 17:36:30)
   * @return java.lang.String
   */
  public final static java.lang.String getCOPIED_TEXT_MESSAGE() {
    return COPIED_TEXT_MESSAGE;
  }

  /*
   * Insert the method's description here.
   * Creation date: (10-08-00 02:27:09)
   * @param groupName java.lang.String
   */
  public static String getCOULD_NOT_CONNECT_TO_GROUP_MESSAGE(String groupName) {
    return COULD_NOT_CONNECT_TO_GROUP_MESSAGE;
  }
```
{  return "Could not establish a connection to the group:\n" + groupName + "\n";
}
/**
 * Insert the method's description here.
 * @return java.lang.String
 */
public final static java.lang.String get_COULD_NOT_INITIALIZE_HELP() {
    return CULD_NOT_INITIALIZE_HELP;
}
/**
 * Insert the method's description here.
 * @return java.lang.String
 */
public final static java.lang.String get_COULD_NOT_READ_DOCUMENT_LIST() {
    return CULD_NOT_READ_DOCUMENT_LIST;
}
/**
 * Insert the method's description here.
 * @return java.lang.String
 */
public final static java.lang.String get_COULD_NOT_WRITE_DOCUMENT_LIST() {
    return CULD_NOT_WRITE_DOCUMENT_LIST;
}
/**
 * Insert the method's description here.
 * @return java.lang.String
 */
public final static java.lang.String get_DEFAULT_DOCUMENT_HEIGHT() {
    return DEFAULT_DOCUMENT_HEIGHT;
}
/**
 * Insert the method's description here.
 * @return java.lang.String
 */
public final static java.lang.String get_DEFAULT_DOCUMENT_WIDTH() {
    return DEFAULT_DOCUMENT_WIDTH;
}
/**
 * Insert the method's description here.
 * @return java.lang.String
 */
public final static java.lang.String get_DEFAULT_FRAME_HORIZONTAL_SIZE() {
    return DEFAULT_FRAME_HORIZONTAL_SIZE;
}
/**
 * Insert the method's description here.
 * @return java.lang.String
 */
public final static java.lang.String get_DEFAULT_FRAME_VERTICAL_SIZE() {
    return DEFAULT_FRAME_VERTICAL_SIZE;
}
/**
 * Insert the method's description here.
 * @return java.lang.String
 */
public final static java.lang.String get_DEFAULT_FRAME_X_COORDINATE() {
    return DEFAULT_FRAME_X_COORDINATE;
}
```java
/**
 * Insert the method's description here.
 * Creation date: (05-08-00 10:11:11)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_FRAME_Y_COORDINATE() {
    return DEFAULT_FRAME_Y_COORDINATE;
}
/**
 * Insert the method's description here.
 * Creation date: (06-09-00 00:48:38)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_GRID_LAYOUT_DOCUMENT_HEIGHT() {
    return DEFAULT_GRID_LAYOUT_DOCUMENT_HEIGHT;
}
/**
 * Insert the method's description here.
 * Creation date: (06-09-00 00:47:52)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_GRID_LAYOUT_DOCUMENT_WIDTH() {
    return DEFAULT_GRID_LAYOUT_DOCUMENT_WIDTH;
}
/**
 * Insert the method's description here.
 * Creation date: (06-09-00 00:50:09)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_GRID_LAYOUT_HORIZONTAL_SPACING() {
    return DEFAULT_GRID_LAYOUT_HORIZONTAL_SPACING;
}
/**
 * Insert the method's description here.
 * Creation date: (06-09-00 00:50:42)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_GRID_LAYOUT_VERTICAL_SPACING() {
    return DEFAULT_GRID_LAYOUT_VERTICAL_SPACING;
}
/**
 * Insert the method's description here.
 * Creation date: (11-08-00 17:12:57)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_GROUP_ID() {
    return DEFAULT_GROUP_ID;
}
/**
 * Insert the method's description here.
 * Creation date: (19-07-00 10:52:34)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_HORIZONTAL_DOCUMENT_SPACING() {
    return DEFAULT_HORIZONTAL_DOCUMENT_SPACING;
}
/**
 * Insert the method's description here.
 * Creation date: (28-06-00 13:37:46)
 * @return double
 */
public final static double getDEFAULT_PANORAMA_PERCENTUAL_SIZE() {
    return DEFAULT_PANORAMA_PERCENTUAL_SIZE;
}
```
/**
 * Insert the method’s description here.
 * Creation date: (25-06-00 12:17:06)
 * @return java.lang.String[]
 */
 public final static Properties getDEFAULT_PREFERENCES() {
  Properties p = new Properties();
  for (int i = 0; i < DEFAULT_PREFERENCES.length; i++)
  { p.setProperty( DEFAULT_PREFERENCES[i][0], DEFAULT_PREFERENCES[i][1] );
  } return p;
} /*
* Insert the method’s description here.
* Creation date: (19-07-00 10:53:26)
* @return java.lang.String
*/
 public final static java.lang.String getDEFAULT_VERTICAL_DOCUMENT_SPACING() { return DEFAULT_VERTICAL_DOCUMENT_SPACING;
} /*
* Insert the method’s description here.
* Creation date: (25-06-00 20:56:23)
* @return java.lang.String[]
*/
 public final static Hashtable getDISPLAY_QUALITIES() {
  Hashtable displayQualities = new Hashtable();
  for (int i = 0; i < DISPLAY_QUALITIES.length; i++)
  { displayQualities.put( DISPLAY_QUALITIES[i][0], DISPLAY_QUALITIES[i][1] );
  } return displayQualities;
} /*
* Insert the method’s description here.
* Creation date: (25-06-00 16:36:00)
* @return java.lang.String
*/
 public final static java.lang.String getDISPLAY_QUALITY() { return DISPLAY_QUALITY;
} /*
* Insert the method’s description here.
* Creation date: (08-07-00 13:59:10)
* @return java.lang.String
*/
 public final static java.lang.String getDISTRIBUTED_DOCUMENT() { return DISTRIBUTED_DOCUMENT;
} /*
* Insert the method’s description here.
* Creation date: (15-08-00 17:50:16)
* @return java.lang.String
*/
 public final static java.lang.String getDO_NOT_ZOOM_TO_OVERVIEW() { return DO_NOT_ZOOM_TO_OVERVIEW;
} /*
* Insert the method’s description here.
* Creation date: (30-07-00 20:40:10)
* @return int
*/
public final static int getDOCUMENT_BORDER_WIDTH() {
    return DOCUMENT_BORDER_WIDTH;
}

/**
 * Insert the method's description here.
 * Creation date: (22-08-00 01:26:41)
 */
public final static String getDOCUMENT_LABEL(String documentName, String authorName, long size, java.
    util.Date lastUpdate) {
    java.util.GregorianCalendar calendar = new java.util.GregorianCalendar();
    calendar.setTime(lastUpdate);
    return "Name:" + documentName + "\n" +
    "Author:" + authorName + "\n" +
    "Size:" + String.valueOf(size) + "\nbytes" + "\n" +
    "Last updated:" + (new java.text.SimpleDateFormat()).format(lastUpdate) + ":" + String.valueOf(  
    (calendar.get(java.util.Calendar.SECOND));
}

/**
 * Insert the method's description here.
 * Creation date: (30-07-00 20:49:44)
 */
public final static java.awt.Color getDOCUMENT_LINE_BORDER_COLOUR() {
    return DOCUMENT_LINE_BORDER_COLOUR;
}

/**
 * Insert the method's description here.
 * Creation date: (06-09-00 21:36:21)
 */
public final static java.lang.String getDOCUMENT_LIST_FILENAME() {
    return DOCUMENT_LIST_FILENAME;
}

/**
 * Insert the method's description here.
 * Creation date: (30-07-00 20:44:17)
 */
public final static java.awt.Color getDOCUMENT_MATTE_BORDER_COLOUR() {
    return DOCUMENT_MATTE_BORDER_COLOUR;
}

/**
 * Insert the method's description here.
 * Creation date: (21-08-00 12:34:32)
 */
public final static java.lang.String getDOCUMENT_NAME() {
    return DOCUMENT_NAME;
}

/**
 * Insert the method's description here.
 * Creation date: (09-07-00 17:44:32)
 */
public final static java.lang.String getDOCUMENT_PATH() {
    return DOCUMENT_PATH;
}

/**
 * Insert the method's description here.
 * Creation date: (06-09-00 14:52:54)
 */
```java
public static final String getDOCUMENT_SIZE_OUT_OF_BOUNDS()
{
    return "Documents must be between, " + String.valueOf(getMIN_GRID_LAYOUT_DOCUMENT_WIDTH()) + ", and, "
    + String.valueOf(getMAX_GRID_LAYOUT_DOCUMENT_WIDTH()) + ", in, width, and, between, "
    + String.valueOf(getMIN_GRID_LAYOUT_DOCUMENT_HEIGHT()) + ", and, "
    + String.valueOf(getMAX_GRID_LAYOUT_DOCUMENT_HEIGHT()) + ", in, height. ";
}

/**
 * Insert the method's description here.
 * Creation date: (21-08-00 16:54:07)
 */
public final static java.lang.String getDOCUMENT_UPDATE()
{
    return DOCUMENT_UPDATE;
}

/**
 * Insert the method's description here.
 * Creation date: (10-08-00 20:46:02)
 */
public final static java.lang.String getFAILED_TO_CONNECT_TO_SERVER()
{
    return FAILED_TO_CONNECT_TO_SERVER;
}

/**
 * Insert the method's description here.
 * Creation date: (10-08-00 16:02:46)
 */
public final static java.lang.String getFAILED_TO_RECEIVE_GROUP_DIRECTORY()
{
    return FAILED_TO_RECEIVE_GROUP_DIRECTORY;
}

/**
 * Insert the method's description here.
 * Creation date: (21-06-00 19:28:09)
 */
public final static int getFILE_EXTENSION_INDEX()
{
    return FILE_EXTENSION_INDEX;
}

/**
 * Insert the method's description here.
 * Creation date: (22-06-00 19:28:32)
 */
public final static int getFILE_TYPE_DESCRIPTION_INDEX()
{
    return FILE_TYPE_DESCRIPTION_INDEX;
}

/**
 * Insert the method's description here.
 * Creation date: (23-02-01 08:49:06)
 */
public final static java.lang.String[] getFILE_TYPE_SYNONYM_TABLE()
{
    return FILE_TYPE_SYNONYM_TABLE;
}
```
public static java.util.Hashtable getTypeSynonyms()
{
    if (TYPE_SYNONYMS == null)
    {
        TYPE_SYNONYMS = new Hashtable();
        for (int i = 0; i < getTypeSynonymTable().length; i++)
        {
            java.util.ArrayList synonymList = new java.util.ArrayList();
            for (int j = 0; j < getTypeSynonymTable()[i].length; j++)
            {
                synonymList.add(getTypeSynonymTable()[i][j]);
            }
            TYPE_SYNONYMS.put(getTypeSynonymTable()[i][0], synonymList);
        }
    }
    return TYPE_SYNONYMS;
}

/**
 * Insert the method’s description here.
 * Creation date: (29-06-00 08:23:56)
 * @return java.lang.String[]
 */
public final static java.lang.String[] getTypes()
{
    return FileTypes;
}

/**
 * Insert the method’s description here.
 * Creation date: (30-08-00 22:01:04)
 * @return java.lang.String
 */
public final static java.lang.String getFINISHED_ADDING_DOCUMENTS()
{
    return FINISHED_ADDING_DOCUMENTS;
}

/**
 * Insert the method’s description here.
 * Creation date: (03-08-00 17:20:36)
 * @return java.lang.String
 */
public final static java.lang.String getFRAME_HORIZONTAL_SIZE()
{
    return FRAME_HORIZONTAL_SIZE;
}

/**
 * Insert the method’s description here.
 * Creation date: (06-08-00 20:11:49)
 * @return double
 */
public final static double getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()
{
    return FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE;
}

/**
 * Insert the method’s description here.
 * Creation date: (03-08-00 17:21:19)
 * @return java.lang.String
 */
public final static java.lang.String getFRAME_VERTICAL_SIZE()
{
    return FRAME_VERTICAL_SIZE;
}

/**
 * Insert the method’s description here.
 * Creation date: (03-08-00 18:08:16)
 * @return java.lang.String
 */
public final static java.lang.String getFRAME_X_COORDINATE() {
    return FRAME_X_COORDINATE;
}

/*
 * Insert the method's description here.
 * Creation date: (03-08-00 16:09:56)
 * @return java.lang.String
 */
public final static java.lang.String getFRAME_Y_COORDINATE() {
    return FRAME_Y_COORDINATE;
}

/*
 * Insert the method's description here.
 * Creation date: (25-06-00 13:32:26)
 * @return java.lang.String
 */
public final static java.lang.String getGRID_LAYOUT() {
    return GRID_LAYOUT;
}

/*
 * Insert the method's description here.
 * Creation date: (19-07-00 13:00:42)
 * @return java.lang.String
 */
public final static java.lang.String getGRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS() {
    return GRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS;
}

/*
 * Insert the method's description here.
 * Creation date: (19-07-00 13:00:07)
 * @return java.lang.String
 */
public final static java.lang.String getGRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS() {
    return GRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS;
}

/*
 * Insert the method's description here.
 * Creation date: (06-09-00 15:05:26)
 * @return java.lang.String
 */
public final static String getGRID_LAYOUT_SPACING_OUT_OF_BOUNDS() {
    return "Horizontal, document spacing must be between 0, " +
            getMIN_GRID_LAYOUT_HORIZONTAL_SPACING() + ", and, " +
            getMAX_GRID_LAYOUT_HORIZONTAL_SPACING() + ",. " +
            "Vertical, document spacing must be between 0, " +
            getMIN_GRID_LAYOUT_VERTICAL_SPACING() + ", and, " +
            getMAX_GRID_LAYOUT_VERTICAL_SPACING() + ",."
        ;
}

/*
 * Insert the method's description here.
 * Creation date: (25-07-00 14:56:17)
 * @return java.lang.String
 */
public final static java.lang.String getGROUP_AT_MAXIMUM_MEMBERS_MESSAGE() {
    return GROUP_AT_MAXIMUM_MEMBERS_MESSAGE;
}

/*
 * Insert the method's description here.
 * Creation date: (18-08-00 02:02:33)
 * @return int
 */
public final static String getGROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS() {
    return GROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS;
}
/**
 * Insert the method's description here.
 */
public final static java.lang.String getGROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS() {
    return GROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS;
}

/**
 * Insert the method's description here.
 */
public final static int getGROUP_DIRECTORY_COLUMN_WIDTH() {
    return GROUP_DIRECTORY_COLUMN_WIDTH;
}

/**
 * Insert the method's description here.
 */
public final static int getGROUP_DIRECTORY_REQUEST_DELAY() {
    return GROUP_DIRECTORY_REQUEST_DELAY;
}

/**
 * Insert the method's description here.
 */
public final static java.lang.String getGROUP_HAS_ACTIVE_PARTICIPANTS_ERROR_MESSAGE() {
    return GROUP_HAS_ACTIVE_PARTICIPANTS_ERROR_MESSAGE;
}

/**
 * Insert the method's description here.
 */
public final static java.lang.String getHTML() {
    return HTML;
}

/**
 * Insert the method's description here.
 */
public final static java.lang.String getINVALID_DELETION_ATTEMPTED_MESSAGE() {
    return INVALID_DELETION_ATTEMPTED_MESSAGE;
}

/**
 * Insert the method's description here.
 */
public final static java.lang.String getLastModified() {
    return LAST_MODIFIED;
}

/**
 * Insert the method's description here.
 */
public final static java.lang.String getLAYOUT_SCHEMA() {
    return LAYOUT_SCHEMA;
}

/**
 * Insert the method's description here.
 */
public final static java.lang.String[] getLAYOUT_SCHEMES() {
    return LAYOUT_SCHEMES;
}

/**
 * Insert the method's description here.
 */
public final static java.lang.String getLOAD_DOCUMENT_LIST_ON_STARTUP() {
    return LOAD_DOCUMENT_LIST_ON_STARTUP;
}

/**
 * Insert the method's description here.
 */
public final static int getLOCAL_DOCUMENT() {
    return LOCAL_DOCUMENT;
}

/**
 * Insert the method's description here.
 */
public final static java.lang.String getLOCAL_HOST() {
    return LOCAL_HOST;
}

/**
 * Insert the method's description here.
 */
public final static java.lang.String getLOCAL_NODE_MESSAGE() {
    return LOCAL_NODE_MESSAGE;
}

/**
 * Insert the method's description here.
 */
public final static int getMAX_DOCNAME_LENGTH() {
    return MAX_DOCNAME_LENGTH;
}

/**
 * Insert the method's description here.
 */
public final static int getMAX_GRID_LAYOUT_DOCUMENT_HEIGHT() {
    return MAX_GRID_LAYOUT_DOCUMENT_HEIGHT;
}

/**
 * Insert the method's description here.
 */
public final static int getMAX_GRID_LAYOUT_DOCUMENT_WIDTH() {
    return MAX_GRID_LAYOUT_DOCUMENT_WIDTH;
}

/**
 * Insert the method's description here.
 */
public final static int getMAX_HEIGHT() {
    return MAX_HEIGHT;
}

/**
 * Insert the method's description here.
 */
public final static int getMAX_WIDTH() {
    return MAX_WIDTH;
}
public final static int getMAX_GRID_LAYOUT_HORIZONTAL_SPACING() {
    return MAX_GRID_LAYOUT_HORIZONTAL_SPACING;
}

/**
 * Insert the method's description here.
 * Creation date: (06-09-00 15:02:56)
 */
public final static int getMAX_GRID_LAYOUT_VERTICAL_SPACING() {
    return MAX_GRID_LAYOUT_VERTICAL_SPACING;
}

/**
 * Insert the method's description here.
 * Creation date: (27-07-00 12:36:59)
 * @return java.util.Hashtable
 */
public final static java.util.Hashtable getMEDIA_TYPES() {
    if ( MEDIA_TYPES.size() == 0 )
        for ( int i = 0; i < getFileTypes().length; i++ )
            MEDIA_TYPES.put( getFileTypes()[i][0], getFileTypes()[i][1] );
    return MEDIA_TYPES;
}

/**
 * Insert the method's description here.
 * Creation date: (18-06-00 17:33:32)
 * @return java.lang.String
 */
public final static java.lang.String getMESSAGE_AREA_DEFAULT_STRING() {
    return MESSAGE_AREA_DEFAULT_STRING;
}

/**
 * Insert the method's description here.
 * Creation date: (29-10-00 16:42:19)
 * @return java.lang.String
 */
public final static java.lang.String getMESSAGE_LOG_FILENAME() {
    return MESSAGE_LOG_FILENAME;
}

/**
 * Insert the method's description here.
 * Creation date: (29-10-00 16:44:55)
 * @return java.lang.String
 */
public final static java.lang.String getMESSAGE_LOG_MESSAGE() {
    return MESSAGE_LOG_MESSAGE;
}

/**
 * Insert the method's description here.
 * Creation date: (29-10-00 16:43:20)
 * @return java.lang.String
 */
public final static java.lang.String getMESSAGE_LOG_WRITE_ERROR() {
    return MESSAGE_LOG_WRITE_ERROR;
}

/**
 * Insert the method's description here.
 * Creation date: (19-07-00 20:06:46)
 * @return java.lang.String
 */
public final static java.lang.String getMESSAGE_PROPERTIES_NAME_COLUMN_HEADING() {
return MESSAGE_PROPERTIES_NAME_COLUMN_HEADER;
}

/**
 * Insert the method's description here.
 * Creation date: (19-07-00 20:07:10)
 * @return java.lang.String
 */
public final static java.lang.String getMESSAGE_PROPERTIES_VALUE_COLUMN_HEADER() {
    return MESSAGE_PROPERTIES_VALUE_COLUMN_HEADER;
}

/**
 * Insert the method's description here.
 * Creation date: (05-09-00 14:48:17)
 * @return int
 */
public final static int getMIN_GRID_LAYOUT_DOCUMENT_HEIGHT() {
    return MIN_GRID_LAYOUT_DOCUMENT_HEIGHT;
}

/**
 * Insert the method's description here.
 * Creation date: (05-09-00 14:51:49)
 * @return int
 */
public final static int getMIN_GRID_LAYOUT_DOCUMENT_WIDTH() {
    return MIN_GRID_LAYOUT_DOCUMENT_WIDTH;
}

/**
 * Insert the method's description here.
 * Creation date: (06-09-00 15:00:54)
 * @return int
 */
public final static int getMIN_GRID_LAYOUT_HORIZONTAL_SPACING() {
    return MIN_GRID_LAYOUT_HORIZONTAL_SPACING;
}

/**
 * Insert the method's description here.
 * Creation date: (06-09-00 15:02:09)
 * @return int
 */
public final static int getMIN_GRID_LAYOUT_VERTICAL_SPACING() {
    return MIN_GRID_LAYOUT_VERTICAL_SPACING;
}

/**
 * Insert the method's description here.
 * Creation date: (25-06-00 16:33:43)
 * @return java.lang.String
 */
public final static java.lang.String getName() {
    return NAME;
}

/**
 * Insert the method's description here.
 * Creation date: (13-06-00 19:55:27)
 * @return java.lang.String
 */
public final static java.lang.String getNO_GROUP_SELECTED_TO_DELETE() {
    return NO_GROUP_SELECTED_TO_DELETE;
}

/**
 * Insert the method's description here.
 * Creation date: (26-07-00 18:23:40)
 * @return java.lang.String
 */
public final static java.lang.String getNO_GROUP_SELECTED_TO_EDIT_MESSAGE() {
    return NO_GROUP_SELECTED_TO_EDIT_MESSAGE;
}
public final static java.lang.String getNO_GROUP_SELECTED_TO_JOIN_MESSAGE() {
    return NO_GROUP_SELECTED_TO_JOIN_MESSAGE;
}

public final static java.lang.String getNO_TEXT_TO_COPY_MESSAGE() {
    return NO_TEXT_TO_COPY_MESSAGE;
}

public final static java.lang.String getNO_TOPIC_TREE_SELECTION_ERROR_MESSAGE() {
    return NO_TOPIC_TREE_SELECTION_ERROR_MESSAGE;
}

public final static java.lang.String getNODE_LOCKED() {
    return NODE_LOCKED;
}

public final static java.lang.String getNOT_CONNECTED_TO_SERVER() {
    return NOT_CONNECTED_TO_SERVER;
}

public final static java.lang.String getNOTHING_TO_REMOVE_MESSAGE() {
    return NOTHING_TO_REMOVE_MESSAGE;
}

public final static java.lang.String getNOTIFYING_SERVER_OF_NAME_IN_USE() {
    return NOTIFYING_SERVER_OF_NAME_IN_USE;
}

public final static java.lang.String getOUT_OF_BOUNDS_EXCEPTION_TEXT() {
    return OUT_OF_BOUNDS_EXCEPTION_TEXT;
}
* @return java.lang.String

public final static java.lang.String getPANOGRAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE() {
    return PANORAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE;
}

/**
 * Insert the method’s description here.
 * @return java.lang.String
 **/
public final static java.lang.String getPLAIN_TEXT() {
    return PLAIN_TEXT;
}

/**
 * Insert the method’s description here.
 * @return java.lang.String
 **/
public final static java.lang.String getPREFERENCES_FILE_NAME() {
    return PREFERENCES_FILE_NAME;
}

/**
 * Insert the method’s description here.
 * @return java.lang.String
 **/
public final static java.lang.String getPROCESSING_DOCUMENTS() {
    return PROCESSING_DOCUMENTS;
}

/**
 * Insert the method’s description here.
 * @return java.lang.String
 **/
public final static java.lang.String getPROPERTIES_HEADER() {
    return PROPERTIES_HEADER;
}

/**
 * Insert the method’s description here.
 * @return java.lang.String
 **/
public final static java.lang.String getRAW_BITMAP() {
    return RAW_BITMAP;
}

/**
 * Insert the method’s description here.
 * @param documentName java.lang.String
 * @param senderName java.lang.String
 */
public final static String getREMOVING_DISTRIBUTED_DOCUMENT(String documentName, String senderName) {
    return senderName + "\has\_\requested\_\that\_\be\_\removed\_\from\_\the\_\panorama\.";
}

/**
 * Insert the method’s description here.
 * @param documentName java.lang.String
 * @param numDoks int
 */
public final static String getREMOVING_DISTRIBUTED_DOCUMENTS_BY_REQUEST(String authorName, int numDoks) {
    return "Removing\_\documents\_\from\_\the\_\panorama\_\by\_\request\_\from\_\author\_\name:\n\" + authorName;
}
/* Insert the method's description here. */
public final static java.lang.String getREMOVING_DISTRIBUTED_DOCUMENTS_MESSAGE() {
    return REMOVING_DISTRIBUTED_DOCUMENTS_MESSAGE;
}
/* Insert the method's description here. */
public final static java.lang.String getREMOVING_DOCUMENTS() {
    return REMOVING_DOCUMENTS;
}
/* Insert the method's description here. */
public final static float getRENDER_CUTOFF() {
    return RENDER_CUTOFF;
}
/* Insert the method's description here. */
public final static java.lang.String getREQUEST_DOCUMENTS() {
    return REQUEST_DOCUMENTS;
}
/* Insert the method's description here. */
public static String getREQUESTING_DOCUMENTS_MESSAGE(int numberOfDocuments) {
    return "Receiving and adding a total of " + String.valueOf(numberOfDocuments) + " documents from the other members of the group."
}
/* Insert the method's description here. */
public final static java.lang.String getREQUESTING_GROUPS_DIRECTORY() {
    return REQUESTING_GROUPS_DIRECTORY;
}
/* Insert the method's description here. */
public final static int getRESET_MESSAGE_AREA_DELAY() {
    return RESET_MESSAGE_AREA_DELAY;
}
/* Insert the method's description here. */
public final static java.awt.Color getRESIZE_RECTANGLE_PEN_COLOUR() {
    return RESIZE_RECTANGLE_PEN_COLOUR;
}
/**
 * Insert the method’s description here.
 * Creation date: (19-06-00 22:01:04)
 * @return java.lang.String
 */
public final static java.lang.String getRETRIEVING_DOCUMENTS() {
    return RETRIEVING_DOCUMENTS;
}
/**
 * Insert the method’s description here.
 * Creation date: (22-06-00 20:21:32)
 * @return java.lang.String
 */
public final static java.lang.String getRTF() {
    return RTF;
}
/**
 * Insert the method’s description here.
 * Creation date: (14-07-00 14:38:06)
 * @return double
 */
public final static double getSCALE_FACTOR_SCALING() {
    return SCALE_FACTOR_SCALING;
}
/**
 * Insert the method’s description here.
 * Creation date: (30-08-00 16:46:24)
 * @return java.lang.String
 */
public final static java.lang.String getSENDER_NAME() {
    return SENDER_NAME;
}
/**
 * Insert the method’s description here.
 * Creation date: (30-08-00 17:24:57)
 * @return java.lang.String
 */
public final static java.lang.String getSHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX() {
    return SHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX;
}
/**
 * Insert the method’s description here.
 * Creation date: (05-10-00 23:11:42)
 * @return int
 */
public final static int getTEXT_DOCUMENT_THRESHOLD() {
    return TEXT_DOCUMENT_THRESHOLD;
}
/**
 * Insert the method’s description here.
 * Creation date: (11-08-00 00:22:49)
 * @return java.lang.String
 */
public final static java.lang.String getTITLE_REQUIRED_MESSAGE() {
    return TITLE_REQUIRED_MESSAGE;
}
* Creation date: (30-08-00 17:25:50)
* @return java.lang.String
*/
1502 public final static java.lang.String getTRUE() {
    return TRUE;
}
1504 /**
 * Insert the method’s description here.
 * Creation date: (22-06-00 19:30:10)
 * @return int
 */
1510 public final static int getTYPE_INDEX() {
    return TYPE_INDEX;
}
1512 /**
 * Insert the method’s description here.
 * Creation date: (08-07-00 20:59:09)
 * @param number0fDocuments int
 * @param recipientName java.lang.String
 */
1518 public static String getUNICAST документов MESSAGE(int number0fDocuments, String recipientName) {
    return "New group member: unicast"," + String.valueOf( number0fDocuments) + ",documents, to,, + recipientName + ",by, request.";
1522 */
1524 /**
 * Insert the method’s description here.
 * Creation date: (25-06-00 16:33:06)
 * @return java.lang.String
 */
1530 public final static java.lang.String getUPDATE_INTERVAL() {
    return UPDATE_INTERVAL;
}
1532 /**
 * Insert the method’s description here.
 * Creation date: (19-06-00 08:37:29)
 * @return int
 */
1534 public final static int getUPDATE_INTERVAL_IN_MILLISECS() {
    return UPDATE_INTERVAL_IN_MILLISECS;
}
1538 */
1540 /**
 * Insert the method’s description here.
 * Creation date: (22-08-00 20:16:10)
 * @return int
 */
1542 public final static int getUPDATE_NOTIFICATION_THRESHOLD() {
    return UPDATE_NOTIFICATION_THRESHOLD;
}
1546 */
1548 /**
 * Insert the method’s description here.
 * Creation date: (30-08-00 22:55:10)
 * @return java.lang.String
 */
1550 public final static java.lang.String getUPDATE_PANORAMA() {
    return UPDATE_PANORAMA;
}
1554 */
1556 /**
 * Insert the method’s description here.
 * Creation date: (31-08-00 02:16:31)
 * @return java.lang.String
 */
1558 public final static java.lang.String getUPDATE_PANORAMA_BY_REQUEST_MESSAGE() {
    return UPDATE_PANORAMA_BY_REQUEST_MESSAGE;
}
1560 */
224
* Insert the method’s description here.
* Creation date: (21-08-00 23:11:12)
* @return java.lang.String
* @param documentName java.lang.String
* @param senderName java.lang.String
*/
1570 public final static String getUPDATING_DISTRIBUTED_DOCUMENT(String documentName) {
    return "Received update of", documentName + ", Now updating...";
1572 }
1574 /**
1576 * Insert the method’s description here.
* Creation date: (21-08-00 23:11:12)
* @return java.lang.String
* @param documentName java.lang.String
* @param senderName java.lang.String
*/
1578 public final static String getUPDATING_DISTRIBUTED_DOCUMENT(String documentName, String senderName) {
    return "Received update of", documentName + " (from)" + senderName + ", Now updating...";
1582 }
1584 /**
1586 * Insert the method’s description here.
* Creation date: (21-08-00 20:42:50)
1588 public final static java.lang.String getUPDATING_documents() { return UPDATING_documents; }
1590 */
1592 /**
1594 * Insert the method’s description here.
* Creation date: (30-08-00 10:43:09)
1596 public final static java.lang.String getVISIBLE_DOCUMENTS_BOX_CAPTION() { return VISIBLE_DOCUMENTS_BOX_CAPTION; }
1598 */
1600 /**
1602 * Insert the method’s description here.
* Creation date: (30-08-00 14:17:20)
1604 */
1606 public final static int getVISIBLE_DOCUMENTS_ICON_TEXT_GAP() {
    return VISIBLE_DOCUMENTS_ICON_TEXT_GAP; }
1608 */
1610 * Insert the method’s description here.
* Creation date: (01-09-00 07:50:17)
1612 public final static java.awt.Dimension getVISIBLE_DOCUMENTS_ITEM_MAX_SIZE() { return VISIBLE_DOCUMENTS_ITEM_MAX_SIZE; }
1614 */
1616 /**
1618 * Insert the method’s description here.
* Creation date: (31-08-00 00:18:59)
1620 public final static java.awt.Color getVISIBLE_DOCUMENTS_SELECTION_COLOUR() {
    return VISIBLE_DOCUMENTS_SELECTION_COLOUR; }
1622 */
1624 /**
1626 * Insert the method’s description here.
* Creation date: (30-08-00 14:26:51)
1628 public final static java.lang.String getVISIBLE_DOCUMENTS_TAB_SIZE() {
    return VISIBLE_DOCUMENTS_TAB_SIZE; }

225
Listing C.7: ClientPanorama.java

```java
package clientapp;

/**
  * This class represents the client panorama
  * Creation date: (15-06-00 00:26:36)
  * @author:
  */
import java.awt.Color;
import java.awt.Dimension;
import java.awt.event.*;
import java.lang.reflect.InvocationTargetException;
import java.util.Hashtable;
import java.io.*;
```
import java.util.ArrayList;
import javax.swing.border.*;
import javax.swing.*;
import javax.swing.JProgressBar.*;
import javax.swing.Timer.*;
import javax.swing.SwingUtilities.*;
import edu.umd.cs.jazz.*;
import edu.umd.cs.jazz.util.*;
import edu.umd.cs.jazz.event.*;
import edu.umd.cs.jazz.component.*;
import java.awt.geom.*;
import java.awt.Point;
import java.awt.event.MouseEvent;
import java.awt.event.ActionEvent;
import java.util.Collection;
import java.util.Iterator;

/** This class implements the panorama that occupies the center portion of the client application window */
public class ClientPanorama extends JCanvas
{
    /** Event handler to ensure panorama updates when documents are added or removed */
    public class UpdateHandler extends JGroupAdapter
    {
        public void nodeAdded( JGroupEvent zge )
        {
            if ( zge.getChild() instanceof ZVisualLeaf )
            {
                validateLayout();
            }
        }
        public void nodeRemoved( JGroupEvent zge )
        {
            if ( zge.getChild() instanceof ZVisualLeaf )
            {
                validateLayout();
            }
        }
    }

    /** Event handler triggered by a timer to ensure continuous updating of the local documents at user determined intervals */
    public class UpdateActionListener implements ActionListener
    {
        public void actionPerformed( ActionEvent ae )
        {
            if ( isPanoramaLock() == false )
            {
                getUpdateTimer().stop();
                updateDocuments();
                getUpdateTimer().restart();
            }
        }
    }

    public class DocumentMouseListener extends MouseAdapter
    {
        public void mouseClicked( MouseEvent zme )
        {
            zme.consume();
        }
    }
}
public class CameraListener implements ZCameraListener {
    public void viewChanged( ZCameraEvent zce )
    {
    }
}

/** This listener handles user clicks on documents in the panorama. */
public class CameraMouseAdapter extends ZMouseAdapter {
    public void mouseClicked( ZMouseEvent zme )
    {
        if ( double click
            if ( zme.getClickCount() == 2 )
        {
            owner.getClientManager().clearTopicTree();
            zoomToOverview();
        }
        if ( isResizingNode() )
        {
            doResize( zme );
        }
        if ( isMovingNode() )
        {
            // previousMousePosition = null;
            setMovingNode( false );
            zme.consume();
            popDownMouseMove();
        }
    }
}

/** This listener handles clicks on the panorama canvas, i.e. outside any documents. */
public class PanoramaMouseListener extends ZMouseAdapter {
    public void mouseClicked( ZMouseEvent zme )
    {
        // if double click
        if ( zme.getClickCount() == 2 )
        {
            centerDocument( (ZVisualLeaf) zme.getNode() );
            zme.consume();
        }
    }

    public void mousePressed( ZMouseEvent zme )
    {
        if ( zme.getSource() instanceof ZVisualLeaf )
        {
            // System.out.println(" Pyr ");
            (ZSwing) ( (ZVisualLeaf) zme.getSource() ).getVisualComponent().getComponent().
            dispatchEvent( zme );
        }
        if ( isResizingNode() )
        {
            doResize( zme );
        }
        if ( isMovingNode() )
        {
            // lock node to current position if auto-lock is enabled
    }
getCurrentlyClickedNode().putClientProperty( ClientConstants.NODE_LOCKED(), ClientInfo.
getPreferences().getProperty( ClientConstants.AUTOLOCK_NODES() ));
146 // previousMousePosition = null;
148 // if right click, bring up the document pop-up menu
150 if ( zme.isMetaDown() )
152 { 154 // point mousePosition = new Point( zme.getX(), zme.getY() );
156 } else
158 { 160 // popDownPopUpMenu();
162 }
164 }
166 public void mouseReleased( ZMouseEvent zme )
168 {
170 // a cleaner solution would be to use the mouseExited event in a mouselistener on the popup menu
172 if ( isResizingNode() == true )
174 { 176 // convert to global coordinates
178 // getCamera().cameraToLocal( mousePosition, currentlyClickedNode );
180 // getCamera().cameraToLocal( upperLeftCorner, null );
182 Dimension newSize = new Dimension();
184 newSize.setSize((int) mousePosition.getX() - (int) upperLeftCorner.getX(),
186 (int) mousePosition.getY() - (int) upperLeftCorner.getY());
188 } else
190 { 192 // I tried a variety of approaches to implementing the below translation code for panorama
194 // nodes, i.e. documents.
196 // These included calculating the difference between the position between the most current
198 // mouse click and the preceding one,
199 // and then using that as the translation measure for the node in question. That scheme
201 // failed, however, and so did my next
202 // attempt which was to convert the mouse positions into the local coordinate system for the
204 // clicked node and proceed as before.
206 // This failed irrespective of whether I used the globalToLocal methods of the node or the
208 // cameraToLocal methods of the camera, which
209 // is hardly surprising given that the latter uses calls to the former to fulfill its purpose
210 // . Finally, I tried using the inverse transform
212 // of the camera directly, i.e. without intermediary function calls and that did the trick!
214 Point globalPoint = new Point();
216 getCamera().getInverseViewTransform().transform( zme.getPoint(), globalPoint );
218 float x = (float)globalPoint.getX();
220 float y = (float)globalPoint.getY();
222 }
public void actionPerformed(ActionEvent ae) {
    JMenuItem menuItem = null;
    // the below test makes explicit what is implicitly assumed: that the originator of the event
    // was a popup menu item
    if ( ae.getSource() instanceof JMenuItem )
    {
        menuItem = (JMenuItem) ae.getSource();
    } else
    {
        return;
    }
    getDocumentMenu().setVisible( false );
    if ( menuItem == getResizeItem() )
    {
        // System.out.println( "resize" );
        resizeDocument();
    } else if ( menuItem == getMoveItem() )
    {
        // System.out.println( "move" );
        moveDocument();
    } /* if ( menuItem == getZoom() )
    {
        // System.out.println( "zoom" );
        zoomDocument();
    } */
    else if ( menuItem == getCopyTextItem() )
    {
        // System.out.println( "copy text" );
        owner.getClientManager().copyText( currentlyClickedNode );
    } else if ( menuItem == getVijLockItem() )
    {
        // System.out.println( "lock item" );
        lockNode( currentlyClickedNode );
    } else if ( menuItem == getVijUnlockAllItem() )
    {
        // System.out.println( "unlockall item" );
        unlockAll();
    } else if ( menuItem == getVijAutoLockAllItem() )
    {
        // System.out.println( "autolockall item" );
        autoLock();
    } else if ( menuItem == getVijArrangeAllItem() )
    {
        // System.out.println( "arrangeall item" );
        validateLayout();
        zoomToOverview();
    }
}
public class PopUpMenuMouseHandler extends MouseAdapter
{
    public void mouseExited( MouseEvent me )
    {
        // The below was commented out in favour of a MouseMotionListener.mouseMoved event handler (below)
        // The reason was that mouseExited events didn’t seem to be generated when the mouse cursor was moved quickly off
        // the source component
        getDocumentMenu().setVisible( false );
    }

    PanoramaMouseListener panoramaMouseListener = new PanoramaMouseListener();
    PanoramaMouseMotionListener panoramaMouseMotionListener = new PanoramaMouseMotionListener();
    float initialScale = 0;
    ZVisualLeaf newLeaf;
    ZPolyline zoomVector;
    static Point previousMousePosition = null;
    private ClientApplication owner;
    private java.util.HashMap documents = new HashMap();
    private java.util.HashMap localDocuments = new HashMap();
    // value modified in popUpPopUpMenu and popDownPopUpMenu
    private edu.umd.cs.jazz.ZVisualLeaf currentlyClickedNode = null;
    private boolean movingNode = false;
    private boolean zoomingNode = false;
    private edu.umd.cs.jazz.ZBounds boundsBeforeResizing = null;
    private boolean resizingNode = false;
    private JSeparator ivjSeparator2 = null;
    private JMenuItem ivjMoveItem = null;
    private JMenuItem ivjResizeItem = null;
    private JPopupMenu ivjDocumentMenu = null;
    private JMenuItem ivjCopyListItemText = null;

    private javax.swing.Timer updateTimer = null;
    private ClientPanoramaLayoutManager layoutManager = null;
    private boolean panoramaLock = false;
    private javax.swing.JMenuItem ivjUnlockAllItem = null;
    private javax.swing.JCheckBoxMenu ivjAutoLockAllItem = null;
    private javax.swing.JSeparator ivjSeparator3 = null;
    private javax.swing.JSeparator ivjSeparator4 = null;
    private javax.swing.JMenuItem ivjArrangeAllItem = null;
    private javax.swing.JMenuItem ivjLockItem = null;

    /**
     * ClientPanorama constructor comment.
     */
    public ClientPanorama() {
        super();
        initialize();
    }

    /**
     * ClientPanorama constructor comment.
     */
    @param aRoot edu.umd.cs.jazz.ZRoot
    @param layer edu.umd.cs.jazz.ZLayerGroup
    public ClientPanorama(edu.umd.cs.jazz.ZRoot aRoot, edu.umd.cs.jazz.ZLayerGroup layer) {
        super(aRoot, layer);
    }

    /**
     * Add a new document to the panorama by inserting a leaf node and decorating it with the properties of the document, including its contents (passed as a byte array).
     */
    @Creation date: (13-08-00 01:01:23)
    @return edu.umd.cs.jazz.ZVisualLeaf
    @param rawPileContents byte[]
    @param senderName java.lang.String
* @param authorName java.lang.String
* @param documentName java.lang.String
**/
    private ZVisualLeaf addDocument(byte[] rawFileContents, String senderName, String authorName, String
documentNameAndPath, String documentName)
{
    ZVisualLeaf leaf;
    ZTransformGroup transform;
    JScrollPane pane;

    //System.out.println( "Document name " + documentName + "\n");
    //System.out.println( "Document name and path " + documentNameAndPath + "\n");
    String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
    String extension = fqPath.substring( fqPath.lastIndexOf( '.' ) ).toLowerCase();
    String mediaType = (String)ClientConstants.getMEDIA_TYPES().get( extension );
    if ( mediaType == null )
    {
        mediaType = ClientConstants.getALLFILES();
    }
    leaf = new ZVisualLeaf();
    leaf.putClientProperty( (ClientConstants.getAUTHOR_NAME(), authorName ) ;
    leaf.putClientProperty( (ClientConstants.getDocument_NAME(), documentNameAndPath ) ;
    leaf.putClientProperty( (ClientConstants.getSENDER_NAME(), senderName ) ;
    if ( mediaType.equalsIgnoreCase( ClientConstants.getPLAIN_TEXT() )
    { JTextArea textArea = new JTextArea( new String( rawFileContents ) );
        textArea.setEditable( false );
        pane = new JScrollPane( textArea );
    } else if ( mediaType.equalsIgnoreCase( ClientConstants.getHTML() ) )
    { JEditorPane editorPane = new JEditorPane( ClientConstants.getHTML(), new String( rawFileContents ) );
        editorPane.setEditable( false );
        pane = new JScrollPane( editorPane );
    } else if ( mediaType.equalsIgnoreCase( ClientConstants.getRTF() ) )
    { JEditorPane editorPane = new JEditorPane( ClientConstants.getRTF(), new String( rawFileContents ) );
        editorPane.setEditable( false );
        pane = new JScrollPane( editorPane );
    } else if ( mediaType.equalsIgnoreCase( ClientConstants.getImageMAP() ) )
    { /* the unformatted byte array containing the image data is stored separately since extracting it
        from the
        // ImageIcon after construction appears to be impossible - may be changed if I figure out a way
        that doesn't
        // incur a hefty performance penalty
        pane = new JScrollPane( new JLabel( new ImageIcon( rawFileContents ) ) );
        pane.putClientProperty( ClientConstants.getRAW_BITMAP(), rawFileContents );
    } // default case
    else
    { if ( rawFileContents.length < ClientConstants.getTEXT_DOCUMENT_THRESHOLD() )
    { JTextArea textArea = new JTextArea( new String( rawFileContents ) );
        textArea.setEditable( false );
        pane = new JScrollPane( textArea );
    } else
    { pane = new JScrollPane( new JLabel( new ImageIcon( rawFileContents ) ) );
    }
    }
    // add informative label to top of scroll pane, complete with bevelled border
```java
// Add document originating from other group members than the one using the client making the call to this function.
* Creation date: (13-08-00 00:58:00)
* param rawFileContents byte[]
* param senderName java.lang.String
* param authorName java.lang.String
* param documentName java.lang.String
*/
public ZVisualLeaf addGlobalDocument(byte[] rawFileContents, String senderName, String authorName, String documentNameAndPath, String documentName)
{
    ZVisualLeaf newLeaf = addDocument( rawFileContents, senderName, authorName, documentNameAndPath, documentName );
    String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
    getDocuments().put( fqPath, newLeaf );
    return newLeaf;
}

// Add document added by the user of the client calling this method.
* Creation date: (13-08-00 01:06:09)
* param rawFileContents byte[]
* param senderName java.lang.String
* param authorName java.lang.String
* param documentName java.lang.String
*/
public ZVisualLeaf addLocalDocument(byte[] rawFileContents, String senderName, String authorName, String documentNameAndPath, long lastModified)
{
    String documentName = documentNameAndPath.substring( documentNameAndPath.lastIndexOf( File.separator ) + 1 );
    ZVisualLeaf newLeaf = addDocument( rawFileContents, senderName, authorName, documentNameAndPath, documentName );
    newLeaf.putClientProperty( ClientConstants.getLast_MODIFIED(), new Long( lastModified ) );
    String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
    getLocalDocuments().put( fqPath, newLeaf );
    return newLeaf;
}

// Insert the method's description here.
* Creation date: (14-10-00 10:39:16)
*/
public void autoLock()
{
    if ( getInAutoLockAllItem().getState() == true )
    {
```
    ClientInfo.getPreferences().setProperty( ClientConstants.getAUTOLOCK_NODES(), ClientConstants.getTRUE() );
}
else
{
    ClientInfo.getPreferences().setProperty( ClientConstants.getAUTOLOCK_NODES(), ClientConstants.getFALSE() );
}
/**
 * Begin a zoom and pan sequence to have the node passed as argument centered and request the topic tree
 * corresponding to that node from the server.
 * Creation date: (07-08-00 09:19:29)
 * @param zme.edu.umd.cs.jazz.event.ZMouseEvent
 */
public void centerDocument( ZVisualLeaf clickedNode )
{
    String path = (String) clickedNode.getClientProperty( ClientConstants.getDocument_PATH() );
    if ( ( path.equals( owner.getClientManager().getClientInfo().getCurrentDocumentName() ) == false )
    {
        owner.getClientManager().requestTopicTree( path );
        //System.out.println( "Clicked path: " + path );
    }
    zoomAndPanToNode( clickedNode );
}
/**
 * Convenience method to allow call by documentID rather than node
 * Creation date: (30-08-00 23:51:07)
 * @param documentID java.lang.String
 */
public void centerDocument(String documentID)
{
    if ( getLocalDocuments().containsKey( documentID ) )
    {
        centerDocument( (ZVisualLeaf) getLocalDocuments().get( documentID ) );
    }
    else
    {
        centerDocument( (ZVisualLeaf) getDocuments().get( documentID ) );
    }
}
/**
 * Compute the bounding box for the set of nodes passed as argument and return it as a rectangle object.
 * I first tried using the getGlobalBounds for each node but that seemed to produce a wrongly
 * translated, although correctly sized, bounding box. The below use of getBounds followed by a
 * localToGlobal transform seems to produce a correct result. Maybe the effect is due to value caching or spurious translations by function
 * getGlobalBounds()...
 * Creation date: (12-10-00 23:59:36)
 * @return java.awt.Rectangle
 * @param nodes edu.umd.cs.jazz.ZNode[]
 */
public java.awt.Rectangle computeOverviewRectangle(ZNode[] nodes)
{
    /* float minX = Float.MAX_VALUE, minY = Float.MAX_VALUE, maxX = Float.MIN_VALUE, maxY = Float.MIN_VALUE;
    float xcandidate, ycandidate;
    for ( int i = 0; i < nodes.length; i++ )
    {
        ZBounds bounds = nodes[i].getBounds();
        nodes[i].localToGlobal( bounds );
        xcandidate = (int)bounds.getMinX();
        ycandidate = (int)bounds.getMinY();
        if ( Math.abs( xcandidate ) < Math.abs( minX ) ) xcandidate : minX;
        minY = Math.abs( xcandidate ) < Math.abs( minY ) ? minY : ycandidate ;
    */
    return null;
xCandidate = (int)bounds.getMaxX();
ycandidate = (int)bounds.getMaxY();
maxX = Math.abs(xcandidate) > Math.abs(maxX) ? xcandidate : maxX;
maxY = Math.abs(ycandidate) > Math.abs(maxY) ? ycandidate : maxY;
java.awt.Rectangle overviewRectangle = new java.awt.Rectangle((int)minX, (int)minY, (int)maxX, (int)maxY);
return overviewRectangle;
*/
return (java.awt.Rectangle) getLayer().getGlobalBounds().getBounds();
*/
/**
 * Remove the panorama pop-up menu.
 * Creation date: (20-07-00 17:59:56)
 */
public void dismissPopupMenu()
{
	documentMenu().setVisible( false );
}/**
 * Perform a resize operation.
 * Creation date: (31-07-00 20:02:40)
 */
public void doResize(ZMouseEvent zm)
{

// if right click, return to original size
if ( zm.isMetaDown() == true )
{
    // use bounds of resize rectangle to reshape active scene graph node and notify relevant objects
    // that this has been done
    Dimension newSize = new Dimension();
    //getCamera().cameraToLocal( bounds, currentlyClickedNode );
    newSize.setSize(
        (int)boundsBeforeResizing.getMaxX() - (int)boundsBeforeResizing.getMinX(),
        (int)boundsBeforeResizing.getMaxY() - (int)boundsBeforeResizing.getMinY() );
    ((PZSwing)currentlyClickedNode.getVisualComponent()).setPreferredSize( newSize );
    currentlyClickedNode.reshape();
    revalidate();
    getDrawingSurface().repaint();
    setResizingNode( false );
}
/**
 * Carry out a zooming operation
 * Creation date: (31-07-00 21:06:01)
 */
public void doZooming(ZMouseEvent zm)
{
    if ( zm.isMetaDown() == true )
    {
        currentlyClickedNode.editor().getTransformGroup().setScale( initialScale );
    }
    previousMousePosition = null;
    setZoomingNode( false );
    ((PZSwing)currentlyClickedNode.getVisualComponent()).setZoomingNode( null );
    // getLayer().removeChild( newLeaf );
    revalidate();
    getDrawingSurface().repaint();
}
/**
 * Insert all panorama documents, both local and global into an array and return it.
 * Creation date: (14-10-00 16:42:03)
 */
return edu.umd.cs.jazz.ZNode[];
*/
public ZNode[] getDocumentsAsArray()
{  
    ZNode nodes[] = new ZNode[ getLocalDocuments().size() + getDocuments().size() ];
    int i = 0;
    for ( int j = 0 ; j < getLocalDocuments().size(); j++, i++ )
    {
        nodes[ i ] = (ZNode) getLocalDocuments().values().toArray()[ j ];
    }
    for ( int j = 0 ; j < getDocuments().size(); j++, i++ )
    {
        nodes[ i ] = (ZNode) getDocuments().values().toArray()[ j ];
    }
    return nodes;
}

/**
 * Insert the method's description here.
 * Creation date: (31-07-00 19:43:21)
 * @return edu.umd.cs.jazz.util.ZBounds
 */
public edu.umd.cs.jazz.util.ZBounds getBoundsBeforeResizing() {
    return boundsBeforeResizing;
}

/**
 * Return the CopyText property value.
 * @return javax.swing.JMenuItem
 */
private javax.swing.JMenuItem getCopyTextItem() {
    if (ivjCopyTextItem == null) {
        try {
            ivjCopyTextItem = new javax.swing.JMenuItem();
            ivjCopyTextItem.setName("CopyTextItem");
            ivjCopyTextItem.setMnemonic('C');
            ivjCopyTextItem.setText("CopyText");
            // user code begin (1)
            ivjCopyTextItem.addActionListener ( new PopupActionListener() );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
        return ivjCopyTextItem;
    }
}

/**
 * Insert the method's description here.
 * Creation date: (28-07-00 00:50:07)
 * @return edu.umd.cs.jazz.ZVisualLeaf
 */
public edu.umd.cs.jazz.ZVisualLeaf getCurrentlyClickedNode() {
    return currentlyClickedNode;
}

/**
 * Helper method to return a document node, irrespective of whether it is global or local
 * Creation date: (14-10-00 11:47:11)
 * @return edu.umd.cs.jazz.ZVisualLeaf
 * @param className java.lang.String
 * @param documentName java.lang.String
 */
public ZVisualLeaf getDocument(String className, String documentName)
{
    String fqName = ClientConstants.createFullyQualifiedDocumentName( className, documentName );
    ZVisualLeaf returnLeaf = null;
    if ( localDocuments.get( fqName ) != null )
    {
        returnLeaf = (ZVisualLeaf) getLocalDocuments().get( fqName );
    }
```java
private JPopupMenu getDocumentMenu() {
    try {
        ivjDocumentMenu = new JPopupMenu();
        ivjDocumentMenu.setName("DocumentMenu");
        ivjDocumentMenu.addItem(getResizeItem());
        ivjDocumentMenu.addItem(getMoveItem());
        ivjDocumentMenu.addItem(getItem());
        ivjDocumentMenu.addItem(getSeparator());
        ivjDocumentMenu.addItem(getMenu());
        ivjDocumentMenu.addItem(getViewItem());
        ivjDocumentMenu.addItem(getLockAllItem());
        ivjDocumentMenu.addItem(getUnlockAllItem());
        // user code begin (1)
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
    return ivjDocumentMenu;
}
```

```
/**
 * Insert the method's description here.
 * Creation date: (08-07-00 01:30:02)
 * @return java.util.Hashtable
 */
public java.util.Hashtable getDocuments() {
    return documents;
}
```

```
/**
 * Insert the method's description here.
 * Creation date: (14-10-00 13:35:35)
 * @return java.swing.JMenuItem
 */
public java.swing.JMenuItem getIvjArrangeAllItem() {
    try {
        ivjArrangeAllItem = new java.swing.JMenuItem();
        ivjArrangeAllItem.setName("ArrangeAllItem");
        ivjArrangeAllItem.setText("Arrange, all");
        // user code begin (1)
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
}
```
return ivjArrangeAllItem;

}  

/**
 * Insert the method's description here.
 * Creation date: (14-10-00 09:53:08)
 * @return javax.swing.JMenuItem
 */

public javax.swing.JMenuItem getIvjAutoLockAllItem() {

if (ivjAutoLockAllItem == null) {
    try {
        ivjAutoLockAllItem = new javax.swing.JMenuItem();
        ivjAutoLockAllItem.setName("AutoLockAllItem");
        ivjAutoLockAllItem.setMnemonic("A");
        ivjAutoLockAllItem.setText("Auto-lock all");
        // user code begin {1}
        ivjAutoLockAllItem.addActionListener( new PopupActionListener() );
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
    }

    return ivjAutoLockAllItem;

}  

/**
 * Insert the method's description here.
 * Creation date: (14-10-00 13:42:49)
 * @return javax.swing.JMenuItem
 */

public javax.swing.JMenuItem getIvjLockItem() {

if (ivjLockItem == null) {
    try {
        ivjLockItem = new javax.swing.JMenuItem();
        ivjLockItem.setName("LockItem");
        ivjLockItem.setMnemonic("L");
        ivjLockItem.setText("Lock");
        // user code begin {1}
        ivjLockItem.addActionListener( new PopupActionListener() );
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
    }

    return ivjLockItem;

}  

/**
 * Insert the method's description here.
 * Creation date: (14-10-00 10:14:36)
 * @return javax.swing.JSeparator
 */

public javax.swing.JSeparator getIvjSeparator3() {

if (ivjSeparator3 == null) {
    try {
        ivjSeparator3 = new javax.swing.JSeparator();
        ivjSeparator3.setName("JSeparator3");
        // user code begin {1}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
    }

    return ivjSeparator3;

}
```java
handleException(ivjExc);
}
}
return ivjSeparator3;
}
/**
 * Insert the method’s description here.
 * Creation date: (14-10-00 10:14:56)
 * @return java.swing.JSeparator
 */
public java.awt.ContentItem getivjSeparator4() {
    if (ivjSeparator4 == null) {
        try {
            ivjSeparator4 = new java.awt.ContentItem();
            ivjSeparator4.setName("JSeparator4");
            // user code begin {1}
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
                // user code end
                handleException(ivjExc);
            }
        }
    return ivjSeparator4;
}
/**
 * Insert the method’s description here.
 * Creation date: (14-10-00 09:51:30)
 * @return java.awt.ContentItem
 */
public java.awt.ContentItem getivjUnlockAllItem() {
    if (ivjUnlockAllItem == null) {
        try {
            ivjUnlockAllItem = new java.awt.ContentItem();
            ivjUnlockAllItem.setName("UnlockAllItem");
            ivjUnlockAllItem.setMnemonic(‘N’);
            ivjUnlockAllItem.setText("Unlock all");
            // user code begin {1}
            ivjUnlockAllItem.addActionListener( new PopupActionListener() );
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
                // user code end
                handleException(ivjExc);
            }
        }
    return ivjUnlockAllItem;
}
/**
 * Return the JSeparator2 property value.
 * @return java.awt.ContentItem
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.ContentItem getJSeparator2() {
    if (ivjJSeparator2 == null) {
        try {
            ivjJSeparator2 = new java.awt.ContentItem();
            ivjJSeparator2.setName("JSeparator2");
            // user code begin {1}
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
                // user code end
                handleException(ivjExc);
            }
        }
```


```java
public ClientPanoramaLayoutManager getLayoutManager() {
    if (layoutManager == null) {
        setLayoutManager( new GridPanoramaLayoutManager() );
    }
    return layoutManager;
}
```

```java
public java.util.HashMap getLocalDocuments() {
    return localDocuments;
}
```

```java
if (ivjMoveItem == null) {
    try {
        ivjMoveItem = new javax.swing.JMenuItem();
        ivjMoveItem.setName("MoveItem");
        ivjMoveItem.setMnemonic('M');
        ivjMoveItem.setText("Move");
        // user code begin {1}
        ivjMoveItem.addActionListener( new PopupActionListener() );
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        HandleException(ivjExc);
    }
    return ivjMoveItem;
}
```

```java
public ClientApplication getOwner() {
    return owner;
}
```

```java
if (ivjResizeItem == null) {
    try {
        ivjResizeItem = new javax.swing.JMenuItem();
        ivjResizeItem.setName("ResizeItem");
    } catch (java.lang.Throwable ivjExc) {
```

240
ivjResizeItem.setMnemonic("R");
ivjResizeItem.setText("Resize");
// user code begin (1)
ivjResizeItem.addActionListener( new PopUpActionListener() );
// user code end
}
} catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
}

return ivjResizeItem;
/**
 * Insert the method’s description here.
 */
/**
 * Creation date: (22-08-00 15:44:15)
 * @return javax.swing.Timer
 */
public javax.swing.Timer getUpdateTimer()
{
  if ( updateTimer == null)
  {
    // start update process in separate thread - multiply by 1000 to get the preferred or default
    // setting in milliseconds.
    updateTimer = new Timer( Integer.parseInt( ClientInfo.getPreferences().getProperty(     
        ClientConstants.getUPDATE_INTERVAL() ))* 1000,
      new UpdateActionListener( ) );
  }
  return updateTimer;
}
/**
 * Called whenever the part throws an exception.
 */
/**
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception)
{
  /* Uncomment the following lines to print uncaught exceptions to stdout */
  // System.out.println("-------- UNCAUGHT EXCEPTION --------");
  // exception.printStackTrace(System.out);
}
/**
 * Insert the method’s description here.
 */
/**
 * Creation date: (16-06-00 17:44:24)
 */
public void initialize()
{
  try {
    // user code begin (1)
    // user code end
    setName("ClientPanorama");
    setSize(1, 1);
} catch (java.lang.Throwable ivjExc) {
   handleException(ivjExc);
  }
  // user code begin (2)
getCameraNode().addMouseListener( new CameraMouseListener() );
getCameraNode().addMouseMotionListener( new PanoramaMouseMotionListener() );
getCamera().addCameraListener( new CameraListener() );
zoomEventHandler = new PZoomHandler( getCameraNode() );
zoomEventHandler.setActive( true );
panEventHandler = new PFanHandler( getCameraNode() );
panEventHandler.setActive( true );
swingEventHandler = new PSwingHandler( this, getCameraNode() );
swingEventHandler.setActive( true );
getDrawingSurface().setRenderQuality( Integer.parseInt( (String) ClientConstants.getDISPLAY_QUALITIES         
());

241
get(ClientInfo.getPreferences().getProperty(ClientConstants.
    getDISPLAY_QUALITY())))
966      getUpdateTimer().start();
967       getLayoutManager().setSettings(ClientInfo.getPreferences());
968      //getLayer().addGroupListener(new UpdateHandler());
969      // user code end
970    }
971    /**
972     * Insert the method’s description here.
973     * Creation date: (29-07-00 18:22:25)
974     * @return boolean
975     */
976    public boolean isMovingNode()
977    {
978        return movingNode;
979    }
980    /**
981     * Insert the method’s description here.
982     * Creation date: (07-09-00 21:16:49)
983     * @return boolean
984     */
985    public boolean isPanoramaLock()
986    {
987        return panoramaLock;
988    }
989    /**
990     * Insert the method’s description here.
991     * Creation date: (31-07-00 20:04:03)
992     * @return boolean
993     */
994    public boolean isResizingNode()
995    {
996        return resizingNode;
997    }
998    /**
999     * Insert the method’s description here.
1000    * Creation date: (29-07-00 19:19:43)
1001    * @return boolean
1002    */
1003    public boolean isZoomingNode()
1004    {
1005        return zoomingNode;
1006    }
1007    /**
1008     * Insert the method’s description here.
1009     * Creation date: (14-10-00 10:37:47)
1010    * @param node edu.umd.cs.jazz.Z VISualLeaf
1011    */
1012    public void lock(ZNode node)
1013    {
1014        node.putClientProperty(ClientConstants.NODE_LOCKED(), ClientConstants.TRUE());
1015    }
1016    /**
1017     * Insert the method’s description here.
1018     * Creation date: (14-10-00 13:47:46)
1019     * @param node edu.umd.cs.jazz.ZNode
1020    */
1021    public void unlockNode(ZNode node)
1022    {
1023        if (!getIVjLockItem().getState())
1024        {
1025            lock(node);
1026        }
1027        else
1028        {
1029            unlock(node);
1030        }
1031    }
1032    /**
1033     * main entrypoint - starts the part when it is run as an application
*param args java.lang.String[]
*/

class TextEditor {
  public static void main(java.lang.String[] args) {
    try {
      JFrame frame = new javax.swing.JFrame();
      ClientPanorama aClientPanorama;
      aClientPanorama = new ClientPanorama();
      frame.getContentPane().add(aClientPanorama);
      frame.setSize(aClientPanorama.getSize());
      frame.addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent e) {
          System.exit(0);
        }
      });
      frame.setVisible(true);
    } catch (Throwable exception) {
      System.err.println("Exception occurred in main()");
      System.out.println("Exception.printStackTrace(System.out);
    }

    /*
    * Translate the document associated with the currently active node in the direction and by the amount
    * indicated by the user's mouse movements.
    * Creation date: (28-07-00 01:13:44)
    */
    public void moveDocument() {
      setMovingNode(true);
      previousMousePosition = null;
    }

    /*
    * Insert the method's description here.
    * Creation date: (22-08-00 11:18:54)
    */
    public JTextArea newDocumentLabel(String text) {
      JTextArea newLabel = new JTextArea(text);
      newLabel.setEditable(false);
      newLabel.setBackground(Color.orange); // newLabel.setSize(new Dimension(Integer.parseInt(ClientConstants.DEFAULT_DOCUMENT_WIDTH), (int)newLabel.getSize().getHeight()));
      Border bevelBorder = BorderFactory.createRaisedBevelBorder();
      Border matteBorder = BorderFactory.createMatteBorder(ClientConstants.getDocument_BORDER_WIDTH(), ClientConstants.getDocument_BORDER_WIDTH(), newLabel.getBackground());
      Border compoundBorder = BorderFactory.createCompoundBorder(bevelBorder, matteBorder);
      newLabel.setBorder(compoundBorder);
      newLabel.setLineWrap(true);
      return newLabel;
    }

    /*
    * Insert the method's description here.
    * Creation date: (28-07-00 01:02:00)
    */
    public void popDownPopUpMenu() {
      getDocumentMenu().setVisible(false);
      // setCurrentlyClickedNode(null);
    }

    /**
     * @return javax.swing.JTextField
     */
    public JTextField newTextField(String text) {
      JTextField textField = new JTextField(text);
      textField.setEditable(false);
      textField.setBackground(Color.orange); // textField.setSize(new Dimension(Integer.parseInt(ClientConstants.DEFAULT DOCUMENT_WIDTH), (int)textField.getSize().getHeight()));
      Border bevelBorder = BorderFactory.createRaisedBevelBorder();
      Border matteBorder = BorderFactory.createMatteBorder(ClientConstants.getDocument_BORDER_WIDTH(), ClientConstants.getDocument_BORDER_WIDTH(), newLabel.getBackground());
      Border compoundBorder = BorderFactory.createCompoundBorder(bevelBorder, matteBorder);
      textField.setBorder(compoundBorder);
      textField.setLineWrap(true);
      return textField;
    }
  }
}

243
public void popupPopupMenu( ZVisualLeaf node, Point position )
{
  if ( node != null )
  {
    setVisibility( node );
    JScrollPane clickedPane = ( JScrollPane)( node.getVisualComponent() ).getComponent();
    // enable text copying only if the clicked node displays text, either formatted or plain
    JViewport viewport = ( JViewport )node.getVisualComponent().getView();
    TextComponent text = ( TextComponent )viewport

    if ( clickedPane.getComponent(0) instanceof javax.swing.text.JTextComponent )
    {
      getCopyTextItem().setEnabled( true );
      if ( text != null )
      {
        getCopyTextItem().setEnabled( false );
        boolean state = clientInfo.getPreferences().getProperty( ClientConstants.AUTOLOCK_NODES ).equals( ClientConstants.AUTOLOCK_NODES ? true : false;
        if ( state )
        {
          itemViewLockItem().setState( state );
          getDocumentMenu().show( this, ( int )position.getX(), ( int )position.getY() );
          getDocumentMenu().setVisible( true );
        }
        else
        {
          state = false;
        }
      }
    }
    else
    {
      state = false;
    }
  }
}

/*
 * Remove document originating from other group member.
 * Creation date: (03-06-00 13:13:00)
 * Param path java.lang.String
 */
public void removeGlobalDocument( String senderName, String documentName )
{
  String fqPath = ClientConstants.createFullyQualifiedName( senderName, documentName );
  ZNode topNode = ( ZNode )documents().get( fqPath ).editor().getTop();
  if ( topNode != null )
  {
    topNode.getParent().removeChild( topNode );
    owner.removeDocumentFromVisibleList( senderName, documentName );
    documents().remove( fqPath );
  }
}

/*
 * Remove document inserted by the user the client making the call to this method.
 * Creation date: (23-06-00 08:39:46)
 * Param path java.lang.String
 */
public void removeLocalDocument( String senderName, String documentName )
{
  String fqPath = ClientConstants.createFullyQualifiedName( senderName, documentName );
  ZNode topNode = ( ZNode )documents().get( fqPath ).editor().getTop();
  if ( topNode != null )
  {
    topNode.getParent().removeChild( topNode );
    getLocalDocuments().remove( fqPath );
    // System.out.println( "Removed: " + fqPath );
    owner.removeDocumentFromVisibleList( senderName, documentName );
  }
}
public void resizeDocument()
{
  boundsBeforeResizing = currentlyClickedNode.getVisualComponentBounds();
  setResizingNode( true );
}

/**
 * Insert the method's description here.
 * Creation date: (28-07-00 01:13:33)
 */
public void resizeDocument()
{
  boundsBeforeResizing = currentlyClickedNode.getVisualComponentBounds();
  setResizingNode( true );
}

/**
 * Insert the method's description here.
 * Creation date: (14-07-00 10:56:28)
 * @param scaleFactor float
 * @param visualLeaf edu.umd.cs.jazz.ZVisualLeaf
 */
public void scaleDocument(double scaleFactor, ZVisualLeaf visualLeaf)
{
  ZTransformGroup zTransformGroup = visualLeaf.editor().getTransformGroup();
  AffineTransform affineTransform = zTransformGroup.getTransform();
  System.out.println( String.valueOf( scaleFactor ));
  System.out.println( String.valueOf( getCamera().getScale() ));
  // scaleFactor = zTransformGroup.getScale() + scaleFactor;
  affineTransform.scale( scaleFactor, scaleFactor );
  zTransformGroup.setTransform( affineTransform );
  // ZTransformGroup.animate( zTransformGroup, affineTransform, 1000, getDrawingSurface() );
}

/**
 * Insert the method's description here.
 * Creation date: (31-07-00 19:43:21)
 * @param newBoundsBeforeResizing edu.umd.cs.jazz.util.ZBounds
 */
public void setBoundsBeforeResizing( edu.umd.cs.jazz.util.ZBounds newBoundsBeforeResizing )
{
  boundsBeforeResizing = newBoundsBeforeResizing;
}

/**
 * Insert the method's description here.
 * Creation date: (28-07-00 00:50:07)
 * @param newCurrentlyClickedNode edu.umd.cs.jazz.ZVisualLeaf
 */
public void setCurrentlyClickedNode( edu.umd.cs.jazz.ZVisualLeaf newCurrentlyClickedNode )
{
  currentlyClickedNode = newCurrentlyClickedNode;
}

/**
 * Insert the method's description here.
 * Creation date: (06-07-00 01:30:08)
 * @param newDocuments java.util.Hashtable
 */
public void setDocuments( java.util.Hashtable newDocuments )
{
  documents = newDocuments;
}

/**
 * Insert the method's description here.
 * Creation date: (14-10-00 13:35:50)
 * @param newJvjArrangeAllItem java.swing.JMenuItem
 */
void setJvjArrangeAllItem( java.swing.JMenuItem newJvjArrangeAllItem )
{
  ivjArrangeAllItem = newJvjArrangeAllItem;
}

/**
 * Insert the method's description here.
 * Creation date: (14-10-00 09:53:08)
 * @param newJvjAutoLockAllItem java.swing.JCheckBoxMenuItem
 */
void setJvjAutoLockAllItem( java.swing.JCheckBoxMenuItem newJvjAutoLockAllItem )
{
ivjAutoLockAllItem = new ivjAutoLockAllItem;
}

/*
 * Insert the method's description here.
 */
void setIvjLockItem(java.awt.PopupMenu newIvjLockItem) {
  ivjLockItem = newIvjLockItem;
}

/*
 * Insert the method's description here.
 */
void setIvjSeparator3(java.awt.PopupMenu newIvjSeparator3) {
  ivjSeparator3 = newIvjSeparator3;
}

/*
 * Insert the method's description here.
 */
void setIvjSeparator4(java.awt.PopupMenu newIvjSeparator4) {
  ivjSeparator4 = newIvjSeparator4;
}

/*
 * Insert the method's description here.
 */
void setIvjUnlockAllItem(java.awt.PopupMenu newIvjUnlockAllItem) {
  ivjUnlockAllItem = newIvjUnlockAllItem;
}

/*
 * Insert the method's description here.
 */
public void setLayoutManager(ClientPanoramaLayoutManager newLayoutManager) {
  layoutManager = newLayoutManager;
}

/*
 * Insert the method's description here.
 */
public void setLocalDocuments(java.util.Hashtable newLocalDocuments) {
  localDocuments = newLocalDocuments;
}

/*
 * Insert the method's description here.
 */
public void setMovingNode(boolean newMovingNode) {
  movingNode = newMovingNode;
}

/*
 * Insert the method's description here.
 */
public void setOwner(ClientApplication newOwner) {
  owner = newOwner;
}
/**
 * Insert the method’s description here.
 * Creation date: (07-09-00 21:16:49)
 * @param newPanoramaLock boolean
 */
 public void setPanoramaLock(boolean newPanoramaLock) {
     panoramaLock = newPanoramaLock;
     PZswing.setUpdating( newPanoramaLock );
 }

 /**
 * Insert the method’s description here.
 * Creation date: (31-07-00 20:04:03)
 * @param newResizingNode boolean
 */
 public void setResizingNode(boolean newResizingNode) {
     resizingNode = newResizingNode;
 }

 /**
 * Insert the method’s description here.
 * Creation date: (22-08-00 15:44:15)
 * @param newUpdateTimer javax.swing.Timer
 */
 public void setUpdateTimer(javax.swing.Timer newUpdateTimer) {
     updateTimer = newUpdateTimer;
 }

 /**
 * Insert the method’s description here.
 * Creation date: (14-07-00 15:35:44)
 * @param zoomNode edu.umd.cs.jazz.ZNode
 */
 public void setZoomHandler(ZNode zoomNode) {
     zoomEventHandler = new PZoomHandler( zoomNode );
 }

 /**
 * Insert the method’s description here.
 * Creation date: (29-07-00 19:19:43)
 * @param newZoomingNode boolean
 */
 public void setZoomingNode(boolean newZoomingNode) {
     zoomingNode = newZoomingNode;
 }

 /**
 * Insert the method’s description here.
 * Creation date: (14-10-00 10:38:24)
 * @param node edu.umd.cs.jazz.ZVisualLeaf
 */
 public void unlock(ZNode node) {
     node.putClientProperty( ClientConstants.getNodeLocked(), ClientConstants.getClientFalse() );
 }

 /**
 * Insert the method’s description here.
 * Creation date: (14-10-00 10:38:46)
 */
 public void unlockAll() {
     Iterator iterator = getDocuments().values().iterator();
     while ( iterator.hasNext() ) {
         unlock((ZVisualLeaf) iterator.next());
     }
     iterator = getLocalDocuments().values().iterator();
     while ( iterator.hasNext() ) {
         unlock((ZVisualLeaf) iterator.next());
     }
}
public synchronized void updateDistributedDocument(String fqPath, byte[] differenceBitmap )
{
    byte[] oldRawFileContents = null;
    byte[] newRawFileContents = null;
    ZVisualLeaf affectedNode = (ZVisualLeaf) getDocuments().get( fqPath );
    JTextComponent affectedScrollPane = (JScrollPane) affectedNode.getVisualComponent();
    java.awt.Rectangle visibleRectangle = affectedScrollPane.getViewport().getViewRect();
    if ( affectedScrollPane.getViewport().getView() instanceof JLabel )
    {
        oldRawFileContents = owner.getGManged().computeBitmapCombination( oldRawFileContents, differenceBitmap );
        affectedScrollPane.getViewport().setView( new JLabel( new ImageIcon( newRawFileContents )));
        affectedScrollPane.putClientProperty( ClientConstants.getRAW_BITMAP(), newRawFileContents );
    }
    else if ( affectedScrollPane.getViewport().getView() instanceof JTextComponent )
    {
        oldRawFileContents = ( (JTextComponent) affectedScrollPane.getViewport().getView() ).getText().getBytes();
        newRawFileContents = owner.getGManged().computeBitmapCombination( oldRawFileContents, differenceBitmap );
        ( (JTextComponent) affectedScrollPane.getViewport().getView() ).setText( new String( newRawFileContents ));
    }
    affectedScrollPane.setColumnHeaderView( newDocumentLabel( ClientConstants.getDocument_NAME( (String) affectedNode.getClientProperty( ClientConstants.getAUTHOR_NAME() ) ),
        newRawFileContents.length, new java.util.Date() ));
    // ensure that the visible rectangle is unchanged
    affectedScrollPane.scrollRectToVisible( visibleRectangle );
    revalidate();
    getDrawingSurface().repaint();
}

public synchronized void updateDistributedDocument(String senderName, String documentName, byte[] differenceBitmap )
{
    byte[] oldRawFileContents = null;
    byte[] newRawFileContents = null;
    String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
    ZVisualLeaf affectedNode = (ZVisualLeaf) getDocuments().get( fqPath );
    // the update was of a document that is not present in the panorama. It should therefore be requested
    // again from
    // the update. This is now likely to succeed as the successful arrival of the update indicates that
    // the connection
    if ( affectedNode == null )
    {
        //
return; // To Do

PZSwing affectedVisualComponent = (PZSwing) affectedNode.getVisualComponent();
JScrollPane affectedScrollPane = ( JScrollPane) affectedVisualComponent.getComponent();
java.awt.Rectangle visibleRectangle = affectedScrollPane.getViewport().getViewRect();
if ( affectedScrollPane.getViewport().getView() instanceof JLabel )
{
  oldRawFileContents = (byte[]) affectedNode.getClientProperty( ClientConstants.getClientProperty( ClientConstants.getRAW_BITMAP() ) );
newRawFileContents = owner.getClientManager().computeBitmapCombination( oldRawFileContents, differenceBitmap );
affectedScrollPane.setViewport().ViewSet( new JLabel( new ImageIcon( newRawFileContents ) ));
affectedScrollPane.putClientProperty( ClientConstants.getClientProperty( ClientConstants.getRAW_BITMAP() ), newRawFileContents );
}
else if ( affectedScrollPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
{
  oldRawFileContents = ((javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView() ).getText().getBytes();
newRawFileContents = owner.getClientManager().computeBitmapCombination( oldRawFileContents, differenceBitmap );
( (javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView() ).setText( new String( newRawFileContents ) );
}
affectedScrollPane.setColumnHeaderView( newDocumentLabel( ClientConstants.getDocument_LABEL( (String) affectedNode.getClientProperty( ClientConstants.getDOCUMENT_NAME() ) ),
  (String) affectedNode.getClientProperty( ClientConstants.getClientProperty( ClientConstants.getAUTHOR_NAME() ) ),
  newRawFileContents.length,
  new java.util.Date() ));
// ensure that the visible rectangle is unchanged
affectedScrollPane.scrollRectToVisible( visibleRectangle );
revalidate();
getDrawingSurface().repaint();
/**
* Update the local documents displayed in the panorama as necessary.
* Creation date: (21-08-00 12:11:47)
*/
public synchronized void updateDocuments()
{
  // safe because of short-circuit evaluation (latter part is evaluated only if the former is true)
  if ( ( getLocalDocuments() != null ) && ( getLocalDocuments().size() > 0 ) )
  {
    Collection values = getLocalDocuments().values();
    Iterator iterator = values.iterator();
    // only notify the user that updating is taking place if the frequency is not too great to avoid visual clutter
    if ( getUpdateTimer().getDelay() > ( ClientConstants.getClientProperty( ClientConstants.clientProperty( ClientConstants.UPDATE_NOTIFICATION_THRESHOLD() ) ) * 1000 ) )
    {
      owner.initMessageArea( 1, values.size(), 1, ClientConstants.getClientProperty( ClientConstants.UPDATEING_DOCUMENTS() ) );
    }
    while ( iterator.hasNext() )
    {
      JVisualLeaf documentNode = ( JVisualLeaf) iterator.next();
      File file = new File( (String) documentNode.getClientProperty( ClientConstants.getClientProperty( ClientConstants.getDOCUMENT_NAME() ) ));
      long lastModified = ( (Long) documentNode.getClientProperty( ClientConstants.getClientProperty( ClientConstants.LAST_MODIFIED() ) ) ).longValue();
      if ( file.lastModified() != lastModified )
      {
        documentNode.putClientProperty( ClientConstants.getClientProperty( ClientConstants.LAST_MODIFIED() ), new Long( file.lastModified() ));
        JScrollPane documentPane = ( JScrollPane) ( PZSwing) documentNode.getClientProperty( );
        getContentPane();
      }
byte rawFileContents[] = null;

if ( documentPane.getViewport().getView() instanceof JLabel )
{
    rawFileContents = (byte[]) documentPane.getClientProperty( ClientConstants.getRaw_Bitmap() );
}
else if ( documentPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
{
    rawFileContents = ((javax.swing.text.JTextComponent) documentPane.getViewport().getView()).getText().getBytes();
}
try {
    byte newRawFileContents[] = owner.getClientManager().readDocument( file );
    owner.getClientManager().updateDocument( file.getPath(),
        ClientInfo.getPreferences().getProperty( ClientConstants.getNAME() ),
        rawFileContents, 
        newRawFileContents );
    updateLocalDocument( file.getPath(), newRawFileContents );
} catch ( IOException ioe )
{ System.err.println( ioe.toString() );
}

if ( getUpdateTimer().getDelay() > ( ClientConstants.getUPDATE_NOTIFICATION_THRESHOLD() * 1000 ) )
{
    owner.advanceTask();
}
if ( getUpdateTimer().getDelay() > ( ClientConstants.getUPDATE_NOTIFICATION_THRESHOLD() * 1000 ) )
{
    owner.resetMessageArea();
}

/**
 * Apply difference array to a document in the panorama put there by the user of the client calling this
 * method.
 * @param documentName java.lang.String
 * @param newRawFileContents byte[]
 */
public synchronized void updateLocalDocument(String documentName, byte[] newRawFileContents)
{
    String fQPath = ClientConstants.createFullyQualifiedDocumentName( ClientInfo.getPreferences().getProperty( ClientConstants.getNAME() ), documentName );
    ZVisualLeaf affectedNode = (ZVisualLeaf) getLocalDocuments().get( fQPath );
    FZSwing affectedVisualComponent = (FZSwing) affectedNode.getVisualComponent();
    JScrollPane affectedScrollPane = (JScrollPane) affectedVisualComponent.getComponent();
    if ( affectedScrollPane.getViewport().getView() instanceof JLabel )
    {
        affectedScrollPane.getViewport().setView( new JLabel( new ImageIcon( newRawFileContents ) ) );
        affectedScrollPane.putClientProperty( ClientConstants.getRaw_Bitmap(), newRawFileContents );
    }
    else if ( affectedScrollPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
    {
        ((javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView()).setText( new String( newRawFileContents ) );
    }
    affectedScrollPane.setColumnHeaderView( newDocumentLabel( ClientConstants.getDocument_LABEL( (String) affectedNode.getClientProperty( ClientConstants.getDocument_NAME() ) ),
...
(String) affectedNode.getClientProperty(ClientConstants.getAUTHOR_NAME())
newRawFileContents.length,
new java.util.Date());
revalidate();
getDrawingSurface().repaint();
}/**
* Creation date: (23-06-00 21:44:36)
*/
public void validateLayout()
{
    layoutManager().doLayout( getAllDocumentsAsArray() );
    // System.out.println("validateLayout");
    revalidate();
    getDrawingSurface().repaint();
}/**
* Zooms and pans the panorama camera to the argument node so that it is centered in and scaled to the
* viewport.
* Creation date: (12-07-00 17:32:31)
* @param node edu.umd.cs.jazz.ZNode
*/
public void zoomAndPanToNode(ZVisualLeaf node)
{
    java.awt.Rectangle targetRectangle = new java.awt.Rectangle( node.getVisualComponentGlobalBounds().
getBounds() );
    PZoomHandler.setTargetRectangle( targetRectangle );
    getCamera().center( targetRectangle,
        Integer.parseInt( ClientInfo.getPreferences().getProperty( ClientConstants.
        getANIMATION_DURATION_IN_MILLIS()) ),
      getDrawingSurface());
}
/**
* Convenience method to allow calls using the documentID rather than the node itself
* Creation date: (30-08-00 14:33:32)
* @param documentID java.lang.String
*/
public void zoomAndPanToNode(String documentID)
{
    if ( getLocalDocuments().containsKey( documentID ) )
        {  
        zoomAndPanToNode( (ZVisualLeaf) getLocalDocuments().get( documentID ) );
        }
        else
        {
            zoomAndPanToNode( (ZVisualLeaf) getDocuments().get( documentID ) );
        }
}
/**
* Insert the method’s description here.
* Creation date: (28-07-00 01:18:23)
*/
public void zoomDocument()
{
    setZoomingNode( true );
    (PZString) currentlyClickedNode.getVisualComponent().setZoomingNode( currentlyClickedNode );
    boundsBeforeResizing = currentlyClickedNode.getVisualComponentBounds();
    previousMousePosition = null;
}
/**
* Zoom and pan the camera so that the viewport contains an overview of all documents currently in the
* panorama
* Creation date: (01-08-00 15:23:15)
*/
public void zoomToOverview()
Listing C.8: ClientPanoramaLayoutManager.java

```java
package clientapp;

import edu.umd.cs.jazz.*;
import edu.umd.cs.jazz.event.*;
import java.util.*;

/**<p>
 * The abstract root of the layout manager class hierarchy.
 *<p>
 * @author
 */
 public abstract class ClientPanoramaLayoutManager implements ZLayoutManager {
 private final static String DOCUMENT_WIDTH = "Document.width";
 private final static String DOCUMENT_HEIGHT = "Document.height";
 private final static int DOCUMENT_HEIGHT_DEFAULT_VALUE = 600;
 private final static int DOCUMENT_WIDTH_DEFAULT_VALUE = 600;

 /* ClientPanoramaLayoutManager constructor comment.
 */
 public ClientPanoramaLayoutManager() {
 super();
 }

 /**<p>
 * Insert the method's description here.
 *<p>
 * @return java.lang.Object
 */
 public Object clone() {
 return null;
 }

 /**<p>
 * Insert the method's description here.
 *<p>
 * @param nodes edu.umd.cs.jazz.ZVisualLeaf[]
 */
 public void doLayout(ZNode[] nodes)
 {
 }

 /**<p>
 * doLayout method comment.
 */
 public void doLayout(ZGroup node) {}

 /**<p>
 * Insert the method's description here.
 *<p>
 * @return java.lang.String
 */
 public String doLayout() {}
}
```
public final static java.lang.String getDOCUMENT_HEIGHT() {
    return DOCUMENT_HEIGHT;
}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 14:24:13)
 * @return int
 */
public final static int getDOCUMENT_HEIGHT_DEFAULT_VALUE() {
    return DOCUMENT_HEIGHT_DEFAULT_VALUE;
}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 14:01:04)
 * @return java.lang.String
 */
public final static java.lang.String getDOCUMENT_WIDTH() {
    return DOCUMENT_WIDTH;
}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 14:29:44)
 * @return int
 */
public final static int getDOCUMENT_WIDTH_DEFAULT_VALUE() {
    return DOCUMENT_WIDTH_DEFAULT_VALUE;
}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 16:13:41)
 * @return java.awt.Dimension
 */
public abstract java.awt.Dimension getDocumentSize();
/**
 * Insert the method's description here.
 * Creation date: (18-10-00 19:42:00)
 * @return boolean
 */
@owner edu.umd.cs.jazz.ZNode
/**
 * @param node edu.umd.cs.jazz.ZNode
 */
public boolean nodeLocked(ZNode node) {
    if ( node.getClientProperty( ClientConstants.getNodeLocked() ) != null )
    {
        return node.getClientProperty( ClientConstants.getNodeLocked() ).equals( ClientConstants.getTrue() ) ? true : false;
    }
    return false;
}
/**
 * postLayout method comment.
 */
public void postLayout(ZGroup node) {
/**
 * preLayout method comment.
 */
public void preLayout(ZGroup node) {
/**
 * Non-overridable method to position individual node. Any subclasses should use this method rather than roll their own
 * in order to hide non-essential panorama implementation detail.
 * Creation date: (14-10-00 13:19:06)
 * @param node edu.umd.cs.jazz.ZNode
 * @param xCoordinate float
 * @param yCoordinate float
 */
public final boolean setPosition(ZNode node, float xCoordinate, float yCoordinate)
{ // System.out.println( node.getClientProperty( ClientConstants.getNODE_LOCKED() ));
    if ( nodeLocked( node ) == false )
    {
        node.editor().getTransformGroup().setTranslation( xCoordinate, yCoordinate );
        return true;
    }
    else
    {
        return false;
    }
}/**
 * Insert the method’s description here.
 * @param node edu.umd.cs.jazz.ZNode
 * @param position java.awt.Dimension
 * @return boolean
 */
public final boolean setPosition(ZNode node, java.awt.Point position)
{
    setPosition( node, (float) position.getX(), (float) position.getY() );
}/**
 * The newSettings argument is a set of manager specific <key,value> pairs that define the values to be
 * displayed in the
 * customization box.
 * @param newSettings java.util.Properties
 * @return void
 */
public void setSettings(java.util.Properties newSettings) {
}
}/**
 * setState method comment.
 */
public void setState(String fieldType, String fieldName, Object fieldValue) {}
}/**
 * Insert the method’s description here.
 * @param owner java.awt.JFrame
 * @return void
 */
public void show() {
    show( null );
}/**
 * Insert the method’s description here.
 * @return java.awt.Properties
 */
public abstract Properties updateSettings(Properties currentSettings);
}/**
 * writeObject method comment.
 */
public void writeObject( edu.umd.cs.jazz.io.ZObjectOutputStream out ) throws java.io.IOException {}
}/**
 * writeObjectRec all method comment.
 */
public void writeObjectRec( edu.umd.cs.jazz.io.ZObjectOutputStream out ) throws java.io.IOException {}
package clientapp;

import javax.swing.event.*;
import java.awt.event.*;
import peerviewmisc.*;
import javax.swing.JMenuItem;
import java.awt.datatransfer.*;

/**
 * This class implements the dialog box for composing new messages
 * @author
 */
public class ComposeNewMessage extends javax.swing.JDialog
{
    public class PopupActionListener implements ActionListener
    {
        public void actionPerformed(ActionEvent ae)
        {
            if (ae.getSource() instanceof JMenuItem)
            {
                JMenuItem menuItem = (JMenuItem) ae.getSource();
                if (menuItem == getPasteTextItem())
                {
                    pasteText();
                }
            }
        }
    }

    public class MouseHandler extends MouseAdapter
    {
        public void mouseClicked(MouseEvent me)
        {
            // if right-click
            if (me.isMetaDown())
            {
                popupPopupMenu(me.getPoint());
            }
        }

        public void mouseEntered(MouseEvent me)
        {
            popupDownPopupMenu();
        }
    }

    public class PopupMouseHandler extends MouseAdapter
    {
        public void mouseExited(MouseEvent me)
        {
            popupDownPopupMenu();
        }

        public void mouseEntered(MouseEvent me)
        {
            popupDownPopupMenu();
        }
    }

    public class BoxMouseHandler extends MouseMotionAdapter
    {
```java
public void mouseMoved(MouseEvent me)
{
    popDownPopupMenu();
}
}
public class SubmitButtonHandler implements ActionListener
{
    public void actionPerformed(ActionEvent ae)
    {
        submitAndClose();
    }
}
public class DiscardButtonHandler implements ActionListener
{
    public void actionPerformed(ActionEvent ae)
    {
        discardAndClose();
    }
}
private javax.swing.JLabel ivjAuthorLabel = null;
private javax.swing.JLabel ivjDateLabel = null;
private javax.swing.JButton ivjDiscardButton = null;
private javax.swing.JPanel ivjDiscardContentPane = null;
private javax.swing.JButton ivjSubmitButton = null;
private javax.swing.JTextArea ivjAuthorTextField = null;
private javax.swing.JEditorPane ivjContentsField = null;
private javax.swing.JTextField ivjDateTextField = null;
private javax.swing.JPanel ivjButtonPanelFlowLayout = null;
private javax.swing.JScrollPane ivjContentsPanel = null;
private javax.swing.JPanel ivjHeaderPanel = null;
private javax.swing.JTextField ivjTitleField = null;
private javax.swing.JLabel ivjTitleLabel = null;
private java.util.Date date = null;
private peerviewsmisc.Message message;
private javax.swing.JMenuItem ivjPasteMenuItem = null;
private javax.swing.JPopupMenu ivjPopupMenu = null;
private ClientApplication clientApplication = null;
/**
 * ComposeNewMessage constructor comment.
 */
public ComposeNewMessage()
{
    super();
    initialize();
}
/**
 * ComposeNewMessage constructor comment.
 * @param owner java.awt.Dialog
 */
public ComposeNewMessage(java.awt.Dialog owner)
{
    super(owner);
}
/**
 * ComposeNewMessage constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public ComposeNewMessage(java.awt.Dialog owner, String title)
{
    super(owner, title);
}
/**
 * ComposeNewMessage constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
```
130  * @param modal boolean
131  */
132 public ComposeNewMessage(java.awt.Dialog owner, String title, boolean modal) {
133     super(owner, title, modal);
134 }
135 /**
136  * ComposeNewMessage constructor comment.
137  * @param owner java.awt.Dialog
138  * @param modal boolean
139  */
140 public ComposeNewMessage(java.awt.Dialog owner, boolean modal) {
141     super(owner, modal);
142 }
143 /**
144  * ComposeNewMessage constructor comment.
145  * @param owner java.awt.Frame
146  */
147 public ComposeNewMessage(java.awt.Frame owner) {
148     super(owner);
149 }
150 /**
151  * ComposeNewMessage constructor comment.
152  * @param owner java.awt.Frame
153  * @param title java.lang.String
154  */
155 public ComposeNewMessage(java.awt.Frame owner, String title) {
156     super(owner, title);
157 }
158 /**
159  * ComposeNewMessage constructor comment.
160  * @param owner java.awt.Frame
161  * @param title java.lang.String
162  * @param modal boolean
163  */
164 public ComposeNewMessage(java.awt.Frame owner, String title, boolean modal) {
165     super(owner, title, modal);
166 }
167 /**
168  * ComposeNewMessage constructor comment.
169  * @param owner java.awt.Frame
170  * @param modal boolean
171  */
172 public ComposeNewMessage(java.awt.Frame owner, boolean modal) {
173     super(owner, modal);
174 }
175 /**
176  * Insert the method's description here.
177  * Creation date: (11-07-00 01:12:03)
178  */
179 public void discardAndClose() {
180     message = null;
181     dispose();
182 }
183 /**
184  * Insert the method's description here.
185  * Creation date: (11-07-00 01:19:46)
186  * @return java.lang.String
187  */
188 public String getAuthor() {
189     return getAuthorTextField().getText();
190 }
191 /**
192  * Return the AuthorLabel property value.
193  * @return javax.swing.JLabel
194  */
private javax.swing.JLabel getAuthorLabel() {
    if (ivjAuthorLabel == null) {
        try {
            ivjAuthorLabel = new javax.swing.JLabel();
            ivjAuthorLabel.setName("AuthorLabel");
            ivjAuthorLabel.setText("Author");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjAuthorLabel;
}
/**
 * Return the JTextField1 property value.
 * @return javax.swing.JTextField
 */
private javax.swing.JTextField getAuthorTextField() {
    if (ivjAuthorTextField == null) {
        try {
            ivjAuthorTextField = new javax.swing.JTextField();
            ivjAuthorTextField.setName("AuthorTextField");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjAuthorTextField;
}
/**
 * Return the JPanel2 property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getButtonPanel() {
    if (ivjButtonPanel == null) {
        try {
            ivjButtonPanel = new javax.swing.JPanel();
            ivjButtonPanel.setName("ButtonPanel");
            ivjButtonPanel.setLayout(new ButtonPanelFlowLayout());
            ivjButtonPanel.add(getSubmitButton(), getSubmitButton().getName());
            ivjButtonPanel.add(getDiscardButton(), getDiscardButton().getName());
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjButtonPanel;
}
private java.awt.FlowLayout getButtonPanelFlowLayout() {
    java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
    try {
        // Create part */
        ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
        ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    return ivjButtonPanelFlowLayout;
}

/**
 * Insert the method's description here.
 * Creation date: (07-08-00 23:36:48)
 * @return clientapp.ClientApplication */
public ClientApplication getClientApplication() {
    return clientApplication;
}

/**
 * Insert the method's description here.
 * Creation date: (11-07-00 01:19:56)
 * @return java.lang.String */
public String getContents() {
    return getContentsField().getText();
}

/**
 * Return the JEditorPane property value.
 * @return javax.swing.JEditorPane */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JEditorPane getContentsField() {
    if (ivjContentsField == null) {
        try {
            ivjContentsField = new javax.swing.JEditorPane();
            ivjContentsField.setName("ContentsField");
            ivjContentsField.setBounds(0, 0, 6, 23);
            // user code begin (1)
            ivjContentsField.addMouseListener( new MouseHandler() );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjContentsField;
}

/**
 * Return the JScrollPane property value.
 * @return javax.swing.JScrollPane */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JScrollPane getContentsPanel() {
    if (ivjContentsPanel == null) {
        try {
            ivjContentsPanel = new javax.swing.JScrollPane();
            ivjContentsPanel.setName("ContentsPanel");
            getContentsPanel().setViewportView(getContentsField());
            // user code begin (1)
            getContentsPanel().setBorder( javax.swing.BorderFactory.createMatteBorder( ClientConstants.get_BORDER_WIDTH(),
                                                                                       ClientConstants.get_BORDER_WIDTH(),
                                                                                       ClientConstants.get_BORDER_WIDTH(),
                                                                                       ClientConstants.get_BORDER_WIDTH(),
                                                                                       ClientConstants.get_BORDER_WIDTH()),
                                                                                       ClientConstants.get_BORDER_WIDTH());
                                                        
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjContentsPanel;
// user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}

return ivjContentsPanel;

/**
 * Insert the method's description here.
 * Creation date: (16-07-00 10:56:47)
 *
 * @return java.util.GregorianCalendar
 */
public java.util.Date getDate() {
    return date;
}

/**
 * Return the JTextField2 property value.
 * @return javax.swing.JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextField getDateField() {
    if (ivjDateField == null) {
        try {
            ivjDateField = new javax.swing.JTextField();
            ivjDateField.setName("DateField");
            // user code begin {1}
            // set date field text string to current time as obtained by instantiating Date and converting it to string
            date = new java.util.Date();
            ivjDateField.setText( date.toString() );
            ivjDateField.setEditable( false );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjDateField;
}

/**
 * Return the DateLabel property value.
 * @return javax.swing.JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getDateLabel() {
    if (ivjDateLabel == null) {
        try {
            ivjDateLabel = new javax.swing.JLabel();
            ivjDateLabel.setName("DateLabel");
            ivjDateLabel.setText("Date");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjDateLabel;
}

/* Return the DiscardButton property value.*/
return ivjDiscardButton;
}

/**
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getHeaderPanel() {
    if (ivjHeaderPanel == null) {
        try {
            ivjHeaderPanel = new javax.swing.JPanel();
            ivjHeaderPanel.setName("HeaderPanel");
            ivjHeaderPanel.setLayout(new java.awt.GridBagLayout());

            java.awt.GridBagConstraints constraintsAuthorLabel = new java.awt.GridBagConstraints();
            constraintsAuthorLabel.gridx = 0; constraintsAuthorLabel.gridy = 0;
            constraintsAuthorLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsAuthorLabel.insets = new java.awt.Insets(4, 4, 4);
            getHeaderPanel().add(getAuthorLabel(), constraintsAuthorLabel);

            java.awt.GridBagConstraints constraintsDateLabel = new java.awt.GridBagConstraints();
            constraintsDateLabel.gridx = 0; constraintsDateLabel.gridy = 1;
            constraintsDateLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsDateLabel.insets = new java.awt.Insets(4, 4, 4);
            getHeaderPanel().add(getDateLabel(), constraintsDateLabel);

            java.awt.GridBagConstraints constraintsTitleLabel = new java.awt.GridBagConstraints();
            constraintsTitleLabel.gridx = 0; constraintsTitleLabel.gridy = 2;
            constraintsTitleLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsTitleLabel.insets = new java.awt.Insets(4, 4, 4);
            getHeaderPanel().add(getTitleLabel(), constraintsTitleLabel);

            java.awt.GridBagConstraints constraintsAuthorTextField = new java.awt.GridBagConstraints();
            constraintsAuthorTextField.gridx = 1; constraintsAuthorTextField.gridy = 0;
            constraintsAuthorTextField.fill = java.awt.GridBagConstraints.BOTH;
            constraintsAuthorTextField.anchor = java.awt.GridBagConstraints.EAST;
            constraintsAuthorTextField.insets = new java.awt.Insets(4, 4, 4);
            getHeaderPanel().add(getAuthorTextField(), constraintsAuthorTextField);

            java.awt.GridBagConstraints constraintsDateField = new java.awt.GridBagConstraints();
            constraintsDateField.gridx = 1; constraintsDateField.gridy = 1;
            constraintsDateField.fill = java.awt.GridBagConstraints.HORIZONTAL;
            constraintsDateField.anchor = java.awt.GridBagConstraints.EAST;
            constraintsDateField.insets = new java.awt.Insets(4, 4, 4);
            getHeaderPanel().add(getDateField(), constraintsDateField);
        }
    }
    return ivjHeaderPanel;
}

/**
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextField getAuthorTextField() {
    if (ivjAuthorTextField == null) {
        try {
            ivjAuthorTextField = new javax.swing.JTextField();
            ivjAuthorTextField.setName("AuthorTextField");
            ivjAuthorTextField.setMnemonic('A');
            ivjAuthorTextField.addActionListener(new ActionListener()
                        {
                public void actionPerformed(ActionEvent e)
                {
                    ivjAuthorTextField.setBackground(Color.LIGHT_GRAY);
                }
            });
            // user code begin (1)
            ivjAuthorTextField.addActionListener(new ActionListener()
                        {
                public void actionPerformed(ActionEvent e)
                {
                    ivjAuthorTextField.setBackground(Color.LIGHT_GRAY);
                }
            });
            // user code end
            return ivjAuthorTextField;
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjAuthorTextField;
}

/**
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextField getDateField() {
    if (ivjDateField == null) {
        try {
            ivjDateField = new javax.swing.JTextField();
            ivjDateField.setName("DateField");
            ivjDateField.setMnemonic('D');
            ivjDateField.addActionListener(new ActionListener()
                        {
                public void actionPerformed(ActionEvent e)
                {
                    ivjDateField.setBackground(Color.LIGHT_GRAY);
                }
            });
            // user code begin (1)
            ivjDateField.addActionListener(new ActionListener()
                        {
                public void actionPerformed(ActionEvent e)
                {
                    ivjDateField.setBackground(Color.LIGHT_GRAY);
                }
            });
            // user code end
            return ivjDateField;
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjDateField;
}
getHeaderPanel().add(getDateField(), constraintsDateField);

java.awt.GridBagConstraints constraintsTitleField = new java.awt.GridBagConstraints();
constraintsTitleField.gridx = 1; constraintsTitleField.gridy = 2;
constraintsTitleField.fill = java.awt.GridBagConstraints.HORIZONTAL;
constraintsTitleField.weightx = 1.0;
constraintsTitleField.insets = new java.awt.Insets(4, 4, 4, 4);
getHeaderPanel().add(getTitleField(), constraintsTitleField);
// user code begin (1)
getHeaderPanel().setBorder(javax.swing.BoxLayout.createMattedBorder(ClientConstants.
  ClientConstants.BORDER_WIDTH(),
  ClientConstants.BORDER_WIDTH(),
  ClientConstants.BORDER_WIDTH(),
  ClientConstants.BORDER_WIDTH(),
  getHeaderPanel().getBackground()));
// user code end
} catch (java.lang.Throwable ivjExc) {
  // user code begin (2)
  // user code end
  handleException(ivjExc);
}

return ivjHeaderPanel;
/**
 * Return the JDialogContentPane property value.
 * @return javax.swing.JPanel
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
java.awt.swing.JPanel getJDialogContentPane() {
  if (ivjJDialogContentPane == null) {
    try {
      ivjJDialogContentPane = new javax.swing.JPanel();
      ivjJDialogContentPane.setName("JDialogContentPane");
      ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
      getJDialogContentPane().add(getHeaderPanel(), "North");
      getJDialogContentPane().add(getContentsPanel(), "Center");
      getJDialogContentPane().add(getButtonPanel(), "South");
    // user code begin (1)
    // user code end
  } catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
  }

  return ivjJDialogContentPane;
}/**
 * Insert the method's description here.
 * Creation date: (18-07-00 08:37:59)
 * @return javax.swing.JMenuMessage
 */
public javax.swing.JMenuMessage getMessage() {
  return message;
}/**
 * Return the PasteTextItem property value.
 * @return javax.swing.JMenuItem
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
javax.swing.JMenuItem getPasteTextItem() {
  if (ivjPasteTextItem == null) {
    try {
      ivjPasteTextItem = new javax.swing.JMenuItem();
      ivjPasteTextItem.setName("PasteTextItem");
    } catch (java.lang.Throwable ivjExc) {
      // user code begin (2)
      // user code end
      handleException(ivjExc);
    }

    return ivjPasteTextItem;
  } catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
  }

  return ivjPasteTextItem;
}
ivjPasteTextItem.setMnemonic('P');
ivjPasteTextItem.setText("Paste.text");
// user code begin {1}
ivjPasteTextItem.addActionListener( new PopUpActionListener() );
// user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}
}
return ivjPasteTextItem;
/**
 * Return the PopUpMenu property value.
 * @return java.awt.PopupMenu
 */
private java.awt.PopupMenu getPopUpMenu() {
    if (ivjPopUpMenu == null) {
        try {
            ivjPopUpMenu = new java.awt.PopupMenu();
            ivjPopUpMenu.setName("PopUpMenu");
            ivjPopUpMenu.add(ivjPasteTextItem);
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjPopUpMenu;
}
/**
 * Return the SubmitButton property value.
 * @return java.awt.JButton
 */
private java.awt.JButton getSubmitButton() {
    if (ivjSubmitButton == null) {
        try {
            ivjSubmitButton = new java.awt.JButton();
            ivjSubmitButton.setName("SubmitButton");
            ivjSubmitButton.setMnemonic('s');
            ivjSubmitButton.setText("Submit");
            // user code begin {1}
            ivjSubmitButton.addActionListener( new SubmitButtonHandler() );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjSubmitButton;
}
/**
 * Insert the method's description here.
 * @return java.lang.String
 */
public String getTitle() {
    return getTitleField().getText();
}
/**

* Return the JTextField property value.
* @return javax.swing.JTextField
*/
private javax.swing.JTextField getTitleField() {
  if (ivjTitleField == null) {
    try {
      ivjTitleField = new javax.swing.JTextField();
      ivjTitleField.setName("TitleField");
      // user code begin (1)
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin (2)
      // user code end
    }
    handleException(ivjExc);
  }
  return ivjTitleField;
}
/*
* Return the Subject property value.
* @return javax.swing.JLabel
*/
private javax.swing.JLabel getTitleLabel() {
  if (ivjTitleLabel == null) {
    try {
      ivjTitleLabel = new javax.swing.JLabel();
      ivjTitleLabel.setName("TitleLabel");
      ivjTitleLabel.setText("Title");
      // user code begin (1)
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin (2)
      // user code end
    }
    handleException(ivjExc);
  }
  return ivjTitleLabel;
} /*
* Called whenever the part throws an exception.
* @param exception java.lang.Throwable
*/
private void handleException(java.lang.Throwable exception) {
  /*
  // Uncomment the following lines to print uncaught exceptions to stdout
  // System.out.println("----------- UNCAUGHT EXCEPTION -----------");
  // exception.printStackTrace(System.out);
  */
  /*
  * Initialize the class.
  */
  /*
  * WARNING: THIS METHOD WILL BE REGENERATED. */
  private void initialize() {
    try {
      // user code begin (1)
      // user code end
      setName("ComposeNewMessage");
      setDefaultCloseOperation(Disposal.DISPOSE_ON_CLOSE);
      setSize(633, 490);
      setModal(true);
      setTitle("Compose new message");
      setContentPane(getDialogContentPanel());
    } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
    }
```java
656 // user code begin {2}
657 // default positioning puts the dialog at center of screen. The method used is a slight hack, but
658 // works fine.
659 setLocationRelativeTo( getOwner() );
660 // user code end
661 */
662 /*
663 * main entrypoint - starts the part when it is run as an application
664 * @param args java.lang.String[]
665 */
666 public static void main(java.lang.String[] args) {
667 try {
668 aComposeNewMessage = new ComposeNewMessage();
669 aComposeNewMessage.setModal(true);
670 aComposeNewMessage.addActionListener(new java.awt.event.ActionListener() {
671 public void windowClosing(java.awt.event.WindowEvent e) {
672 System.exit(0);
673 }
674 });
675 aComposeNewMessage.setVisible(true);
676 }
677 catch (Throwable exception) {
678 System.err.println("Exception occurred in main()", e, this.getClass().getName());
679 exception.printStackTrace(System.out);
680 }
681 /*
682 * Paste the contents of the clipboard, if any, into the message at the current caret position.
683 * Creation date: (07-08-00 23:31:17)
684 */
685 public void pasteText()
686 {
687 getContentsField().paste();
688 }
689 */
690 /*
691 * Insert the method's description here.
692 * Creation date: (08-08-00 10:03:52)
693 */
694 public void popDownPopupMenu()
695 {
696 if ( getPopupMenu().isVisible() )
697 {
698 getPopupMenu().setVisible( false );
699 }
700 */
701 /*
702 * Display the pop-up menu.
703 * Creation date: (07-08-00 23:20:22)
704 */
705 public void popUpPopupMenu(java.awt.Point position)
706 {
707 getPopupMenu().show( getContentsPanel() , (int) position.getX() , (int) position.getY() );
708 */
709 /*
710 * Insert the method's description here.
711 * Creation date: (11-07-00 00:38:26)
712 * @param newAuthorName java.lang.String
713 */
714 public void setAuthor(String newAuthorName)
715 {
716 getAuthorTextField().setText( newAuthorName );
717 */
718 /*
719 * Insert the method's description here.
720 * Creation date: (07-08-00 23:38:48)
721 */
722 ```
public void setApplication(ClientApplication newClientApplication) {
    clientApplication = newClientApplication;
}
/**
 * Insert the method’s description here.
 * Creation date: (16-07-00 10:56:47)
 */
public void setDate(java.util.Date newDate) {
    date = newDate;
}
/**
 * Insert the method’s description here.
 * Creation date: (18-07-00 08:37:59)
 */
public void setMessage(peerviewmisc.Message newMessage) {
    message = newMessage;
}
/**
 * Package the contents of the message box fields into a message data structure and then dispose of the box itself.
 * The message structure is part of the dialog box object and can be retrieved by the caller after the box is disposed of.
 * Creation date: (11-07-00 01:11:45)
 */
public void submitAndClose()
{
    if (getTitleField().getText().length() == 0 )
    {
    MessageBox messageBox = new MessageBox();
    messageBox.setText( ClientConstants.getTitle_REQUIRED_MESSAGE() );
    messageBox.show();
    return;
    }
    message = new peerviewmisc.Message();
    message.setAuthor( getAuthor() );
    message.setContents( getContents() + ClientConstants.getCONTENTS_AND_SIGNATURE_DIVIDER() + ClientInfo .getPreferences().getProperty( ClientConstants.getSIGNATURE() ));
    message.setCreationDate( getDate() );
    message.setTitle( getTitle() );
    dispose();
}

Listing C.10: DiscussionFrame.java

import java.awt.*;
import java.awt.dnd.*;
import java.awt.event.*;
import java.awt.flow.*;
import javax.swing.*;
import javax.swing.border.*;

/**
 * This class implements the discussion panel in the bottom portion of the client application window.
 * Creation date: (24-05-00 13:06:10)
 */
public class DiscussionFrame extends JFrame {
private JMenuItem ivjAboutItem = null;
private JMenuItem ivjCloseWindowItem = null;
private JMenuItem ivjContentsItem = null;
private JButton ivjDefaultToolBarBarButton = null;

private JMenuBar ivjDiscussionFrame;JMenuBar = null;
private JMenu ivjDiscussionMenu = null;
private JTree ivjDiscussionTree = null;
private JMenu ivjHelpMenu = null;
private JPanel ivjJFrameContentPane = null;
private JPanel ivjJPanel1 = null;
private JPanel ivjJPanel2 = null;
private JSeparator ivjJSeparator1 = null;
private JSplitPane ivjJSplitPanel = null;
private JToolBar ivjJToolBar1 = null;
private JButton ivjJToolBarButton1 = null;
private JButton ivjJToolBarButton2 = null;
private JButton ivjJToolBarButton3 = null;
private JTextPane ivjJMessageDisplayArea = null;
private JMenuItem ivjJMessageItem = null;
private JMenuItem ivjJNewThreadItem = null;
private JLabel ivjJAuthor = null;
private JLabel ivjJAuthorLabel = null;
private JLabel ivjJDateLabel = null;
private JLabel ivjJLabel1 = null;
private JLabel ivjJLabel2 = null;
private BoxLayout ivjJPanel2BoxLayout = null;
private JPanel ivjJPanel1HoldingHeader = null;
private JPanel ivjJPanel1HoldingHeaderFields = null;
private JPanel ivjJPanel1HoldingHeaderLabels = null;
private JLabel ivjJSubjectLabel = null;
private BoxLayout ivjJPanel1HoldingHeaderFieldsBoxLayout = null;
private BoxLayout ivjJPanel1HoldingHeaderLabelsBoxLayout = null;
private JLabel ivjJAuthorLabel1 = null;
private JDialog ivjJComposeContribution = null;
private JLabel ivjJDateLabel1 = null;
private JPanel ivjJDialogContentPanel = null;
private JPanel ivjJMessagePanel = null;
private BoxLayout ivjJMessagePanelBoxLayout = null;
private JLabel ivjJSubjectLabel1 = null;
private JScrollPane ivjJEventHandler = new JScrollPane();
private JScrollPane ivjJMessageDisplayAreaOnComposeBox = null;
private JPanel ivjJPanel1HoldingHeaderFieldsOnComposeBox = null;
private JPanel ivjJPanel1HoldingHeaderUnComposeBox = null;
private JPanel ivjJPanel1HoldingHeaderUnComposeBoxBoxLayout = null;
private JPanel ivjJPanel1HoldingHeadingLabelsOnComposeBox = null;
private BoxLayout ivjJPanel1HoldingHeadingLabelsOnComposeBoxBoxLayout = null;
private JPanel ivjJPanel3 = null;
private FlowLayout ivjJPanel3FlowLayout = null;
private BoxLayout ivjJDialogContentPanelBoxLayout = null;
private JTextField ivjJTextfield = null;
private JTextField ivjJTextfield1 = null;
private JTextField ivjJTextfield2 = null;
private JPanel ivjJPanel4 = null;
private JButton ivjJDiscardButton = null;
private JButton ivjJSubmitButton = null;

class IvjEventHandler implements java.awt.event.ActionListener {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        if (e.getSource() == DiscussionFrame.this.getMessageItem())
            connEtoM1(e);
        if (e.getSource() == DiscussionFrame.this.getJToolBarButton1())
            connEtoM2(e);
    }
    */

    DiscussionFrame constructor comment.

    */
    public DiscussionFrame() {
        super();
        initialize();
    }
}
```java
84 */
86 * DiscussionFrame constructor comment.
87 * @param title java.lang.String
88 */
89 public DiscussionFrame(String title) {
90     super(title);
91 }
92 /**
93 * @param arg1 java.awt.event.ActionEvent
94 */
95 /* WARNING: THIS METHOD WILL BE REGENERATED. */
96 private void connEtoM1(java.awt.event.ActionEvent arg1) {
97     try {
98         // user code begin {1}
99         // user code end
100         getComposeContribution().show();
101         // user code begin {2}
102         // user code end
103         } catch (java.lang.Throwable ivjExc) {
104             // user code begin {3}
105             // user code end
106             handleException(ivjExc);
107         }
108     }
109 */
110 /**
111 * @param arg1 java.awt.event.ActionEvent
112 */
113 /* WARNING: THIS METHOD WILL BE REGENERATED. */
114 private void connEtoM2(java.awt.event.ActionEvent arg1) {
115     try {
116         // user code begin {1}
117         // user code end
118         getComposeContribution().show();
119         // user code begin {2}
120         // user code end
121         } catch (java.lang.Throwable ivjExc) {
122             // user code begin {3}
123             // user code end
124             handleException(ivjExc);
125         }
126     }
127 */
128 /*
129 * @return javax.swing.JMenuItem
130 */
131 /* WARNING: THIS METHOD WILL BE REGENERATED. */
132 private javax.swing.JMenuItem getAboutItem() {
133     if (ivjAboutItem == null) {
134         try {
135             ivjAboutItem = new javax.swing.JMenuItem();
136             ivjAboutItem.setName("AboutItem");
137             ivjAboutItem.setText("About");
138             // user code begin {1}
139             // user code end
140             } catch (java.lang.Throwable ivjExc) {
141                 // user code begin {2}
142                 // user code end
143                 handleException(ivjExc);
144             }
145         }
146         return ivjAboutItem;
268
```
private javax.swing.JLabel getAuthor() {
    if (ivjAuthor == null) {
        try {
            ivjAuthor = new javax.swing.JLabel();
            ivjAuthor.setHorizontalAlignment("Author");
            ivjAuthor.setForeground(new java.awt.Color(0, 0, 0));
            ivjAuthor.setOpaque(true);
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjAuthor;
}

private javax.swing.JLabel getAuthorLabel() {
    if (ivjAuthorLabel == null) {
        try {
            ivjAuthorLabel = new javax.swing.JLabel();
            ivjAuthorLabel.setHorizontalAlignment("AuthorLabel");
            ivjAuthorLabel.setOpaque(true);
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjAuthorLabel;
}

private javax.swing.JLabel getAuthorLabel1() {
    if (ivjAuthorLabel1 == null) {
        try {
            ivjAuthorLabel1 = new javax.swing.JLabel();
            ivjAuthorLabel1.setHorizontalAlignment("AuthorLabel1");
            ivjAuthorLabel1.setOpaque(true);
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjAuthorLabel1;
}
214     // user code end
215     handleException(ivjExc);
216
217     }
218     return ivjAuthorLabel1;
219     }
220     /**
221     * Return the CloseWindowItem property value.
222     * @return javax.swing.JMenuItem
223     */
224     /** WARNING: THIS METHOD WILL BE REGENERATED. */
225     private javax.swing.JMenuItem getCloseWindowItem() {
226     if (ivjCloseWindowItem == null) {
227     try {
228     ivjCloseWindowItem = new javax.swing.JMenuItem();
229     ivjCloseWindowItem.setName("CloseWindowItem");
230     ivjCloseWindowItem.setText("Close,window");
231     // user code begin (1)
232     // user code end
233     } catch (java.lang.Throwable ivjExc) {
234     // user code begin (2)
235     // user code end
236     handleException(ivjExc);
237     }
238     return ivjCloseWindowItem;
239     }
240     /**
241     * Return the ComposeContribution property value.
242     * @return javax.swing.JDialog
243     */
244     /** WARNING: THIS METHOD WILL BE REGENERATED. */
245     private javax.swing.JDialog getComposeContribution() {
246     if (ivjComposeContribution == null) {
247     try {
248     ivjComposeContribution = new javax.swing.JDialog();
249     ivjComposeContribution.setName("ComposeContribution");
250     ivjComposeContribution.setDefaultCloseOperation(javax.swing.windowConstants.DISPOSE_ON_CLOSE);
251     ivjComposeContribution.setBounds(40, 672, 564, 500);
252     ivjComposeContribution.setTitle("Compose,new_message");
253     getComposeContribution().setContentPane(getJDialogContentPane());
254     // user code begin (1)
255     // user code end
256     } catch (java.lang.Throwable ivjExc) {
257     // user code begin (2)
258     // user code end
259     handleException(ivjExc);
260     }
261     return ivjComposeContribution;
262     }
263     /**
264     * Return the ContentsItem property value.
265     * @return javax.swing.JMenuItem
266     */
267     /** WARNING: THIS METHOD WILL BE REGENERATED. */
268     private javax.swing.JMenuItem getContentsItem() {
269     if (ivjContentsItem == null) {
270     try {
271     ivjContentsItem = new javax.swing.JMenuItem();
272     ivjContentsItem.setName("ContentsItem");
273     ivjContentsItem.setText("Contents");
274     // user code begin (1)
275     // user code end
276     } catch (java.lang.Throwable ivjExc) {
277     // user code begin (2)
// user code end
handleException(ivjExc);
}
}
return ivjContentsItem;
}
/**
 * Return the DateLabel property value.
 * @param javax.swing.JLabel
 */
/**< WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getDateLabel() {
    if (ivjDateLabel == null) {
        try {
            ivjDateLabel = new javax.swing.JLabel();
            ivjDateLabel.setName("DateLabel");
            ivjDateLabel.setText("Date:");
            ivjDateLabel.setPreferredSize(new java.awt.Dimension(29, 23));
            ivjDateLabel.setMaximumSize(new java.awt.Dimension(29, 23));
            // user code begin (1)
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
            }
            return ivjDateLabel;
        }
    }
    /**
     * Return the DateLabel1 property value.
     * @param javax.swing.JLabel
     */
    /**< WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getDateLabel1() {
    if (ivjDateLabel1 == null) {
        try {
            ivjDateLabel1 = new javax.swing.JLabel();
            ivjDateLabel1.setName("DateLabel1");
            ivjDateLabel1.setPreferredSize(new java.awt.Dimension(29, 23));
            ivjDateLabel1.setMaximumSize(new java.awt.Dimension(29, 23));
            // user code begin (1)
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
            }
            return ivjDateLabel1;
        }
    /**<
     * Return the DefaultToolBarButton property value.
     * @param javax.swing.JButton
     */
    /**< WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getDefaultToolBarButton() {
    if (ivjDefaultToolBarButton == null) {
        try {
            ivjDefaultToolBarButton = new javax.swing.JButton();
            ivjDefaultToolBarButton.setName("DefaultToolBarButton");
            ivjDefaultToolBarButton.setText(" ");
            ivjDefaultToolBarButton.setHorizontalTextPosition(java.awt.SwingConstants.CENTER);
            ivjDefaultToolBarButton.setVerticalTextPosition(java.awt.SwingConstants.BOTTOM);
        
    271
ivjDefaultToolBarButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarbuttongraphics/general/Add16.gif")));
ivjDefaultToolBarButton.setMargin(new java.awt.Insets(0, 0, 0, 0));

// user code begin (1)
// user code end
}
} catch (java.lang.Throwable ivjExc) {
  // user code begin (2)
  // user code end
  handleException(ivjExc);
}

return ivjDefaultToolBarButton;

/**
 * Return the JButton2 property value.
 * @return javax.swing.JButton
 */
private javax.swing.JButton getDiscardButton() {
  if (ivjDiscardButton == null) {
    try {
      ivjDiscardButton = new javax.swing.JButton();
      ivjDiscardButton.setName("DiscardButton");
      ivjDiscardButton.setText("Discard");
      // user code begin (1)
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin (2)
      // user code end
      handleException(ivjExc);
    }

    return ivjDiscardButton;
  }

  try {
    ivjDiscussionFrameJMenuBar = new javax.swing.JMenuBar();
    ivjDiscussionFrameJMenuBar.setName("DiscussionFrameJMenuBar");
    ivjDiscussionFrameJMenuBar.add(getDiscussionMenu());
    ivjDiscussionFrameJMenuBar.add(getHelpMenu());
    // user code begin (1)
    // user code end
  } catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
  }

  return ivjDiscussionFrameJMenuBar;
}

/**
 * Return the DiscussionMenu property value.
 * @return javax.swing.JMenu
 */
private javax.swing.JMenu getDiscussionMenu() {
  if (ivjDiscussionMenu == null) {
    try {
      ivjDiscussionMenu = new javax.swing.JMenu();
      ivjDiscussionMenu.setName("DiscussionMenu");
    }

    return ivjDiscussionMenu;
  }

  try {
    ivjDiscussionMenu = new javax.swing.JMenu();
    ivjDiscussionMenu.setName("DiscussionMenu");
  } catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
  }

  return ivjDiscussionMenu;
}
ivjDiscussionMenu.setText("Discuss");
ivjDiscussionMenu.add(getNewThreadItem());
ivjDiscussionMenu.add(getNewMessageItem());
ivjDiscussionMenu.add(getSeparator1());
ivjDiscussionMenu.add(getCloseWindowItem());
// user code begin (1)
// user code end
)
catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
}
return ivjDiscussionMenu;

/**
 * Return the DiscussionTree property value.
 * @return javax.swing.JTree
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTree getDiscussionTree() {
if (ivjDiscussionTree == null) {
try {
ivjDiscussionTree = new javax.swing.JTree();
ivjDiscussionTree.setName("DiscussionTree");
ivjDiscussionTree.setToolTipText("Discussion tree");
ivjDiscussionTree.setPreferredSize(new java.awt.Dimension(78, 200));
ivjDiscussionTree.setMaximumSize(new java.awt.Dimension(100000, 100000));
// user code begin (1)
// user code end
)
catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}

// user code begin (3)
// user code end
return ivjDiscussionTree;

/**
 * Return the HelpMenu property value.
 * @return javax.swing.JMenu
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenu getHelpMenu() {
if (ivjHelpMenu == null) {
try {
ivjHelpMenu = new javax.swing.JMenu();
ivjHelpMenu.setName("HelpMenu");
ivjHelpMenu.setText("Help");
ivjHelpMenu.add(getContentsItem());
ivjHelpMenu.add(getAboutItem());
// user code begin (1)
// user code end
)
catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}

// user code begin (3)
// user code end
return ivjHelpMenu;

/**
 * Return the .DialogContentPane property value.
 * @return javax.swing.JPanel
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getDialogContentPane()
{
    if (ivjDialogContentPane == null) {
        try {
            ivjDialogContentPane = new javax.swing.JPanel();
            ivjDialogContentPane.setName("DialogContentPane");
            ivjDialogContentPane.setLayout(new javax.swing.GroupLayout);
            ivjDialogContentPane.add(getMessagePanel(), getName());
            ivjDialogContentPane.add(getPanel3(), getName());
            ivjDialogContentPane.add(getPanel14(), getName());
            // user code begin (1)
            // user code end
            catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
        }
        return ivjDialogContentPane;
    }
    /*
     * Return the JDialogContentPaneBoxLayout property value.
     * @return javax.swing.GroupLayout
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.GroupLayout getDialogContentPaneBoxLayout()
    {
        javax.swing.GroupLayout ivjDialogContentPaneBoxLayout = null;
        try {
            // Create part/
            ivjDialogContentPaneBoxLayout = new javax.swing.GroupLayout(getDialogContentPane(), javax.swing.GroupLayout);
            // user code begin (1)
            // user code end
            catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        }
        return ivjDialogContentPaneBoxLayout;
    }
    /**
     * Return the JFrameContentPane property value.
     * @return javax.swing.JPanel
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JPanel getDialogContentPane() {
        if (ivjDialogContentPane == null) {
            try {
                ivjDialogContentPane = new javax.swing.JPanel();
                ivjDialogContentPane.setName("DialogContentPane");
                ivjDialogContentPane.setLayout(new javax.swing.GroupLayout());
                ivjDialogContentPane.add(getMessagePanel(), "North");
                ivjDialogContentPane.add(getPanel1(), "Center");
                // user code begin (1)
                // user code end
                catch (java.lang.Throwable ivjExc) {
                    // user code begin (2)
                    // user code end
                    handleException(ivjExc);
                }
            }
            return ivjDialogContentPane;
        }
    }
    /**
     * Return the JLabel1 property value.
     * @return javax.swing.JLabel
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JLabel getDialogContentPane() {
        if (ivjJLabel1 == null) {
            try {

ivjLabel1 = new javax.swing.JLabel();
ivjLabel1.setName("JLabel1");
ivjLabel1.setText("JLabel1");
ivjLabel1.setForeground(java.awt.SystemColor.controlText);
ivjLabel1.setMaximumSize(new java.awt.Dimension(45, 23));
ivjLabel1.setMinimumSize(new java.awt.Dimension(45, 23));
// user code begin {1}
// user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}
}
return ivjLabel1;

/**
 * Return the JLabel2 property value.
 * @return javax.swing.JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getJLabel2() {
    if (ivjLabel2 == null) {
        try {
            ivjLabel2 = new javax.swing.JLabel();
            ivjLabel2.setName("JLabel2");
            ivjLabel2.setText("JLabel2");
            ivjLabel2.setMaximumSize(new java.awt.Dimension(45, 23));
            ivjLabel2.setMinimumSize(new java.awt.Dimension(45, 23));
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjLabel2;
}

/**
 * Return the JPanel1 property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJPanel1() {
    if (ivjPanel1 == null) {
        try {
            ivjPanel1 = new javax.swing.JPanel();
            ivjPanel1.setName("JPanel1");
            ivjPanel1.setLayout(new java.awt.BorderLayout());
            getJPanel1().add(getDiscussionTree(), "Center");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjPanel1;
}

/**
 * Return the JPanel2 property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJPanel12()
{
    if (ivj JPanel12 == null) {
        try {
            ivj JPanel12 = new javax.swing.JPanel();
            ivj JPanel12.setName("JPanel12");
            ivj JPanel12.setLayout(new JPanel2BoxLayout());
            getJPanel2().add(getPanelHolderHeader(), getPanelHolderHeader().getName());
            getJPanel2().add(getMessageDisplayArea(), getMessageDisplayArea().getName());
            // user code begin (1)
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
        }
    return ivj JPanel12;
    } /*
    * Return the JPanel2BoxLayout property value.
    */
    java.swing.BoxLayout ivj JPanel2BoxLayout = null;
    try {
        /* Create part */
        ivj JPanel2BoxLayout = new javax.swing.BoxLayout(getJPanel2(), java.swing.BoxLayout.Y_AXIS);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    return ivj JPanel2BoxLayout;
} /*
* Return the JPanel3 property value.
* @return java.swing.JPanel
*/
private javax.swing.JPanel getJPanel3()
{
    if (ivj JPanel3 == null) {
        try {
            ivj JPanel3 = new javax.swing.JPanel();
            ivj JPanel3.setName("JPanel3");
            ivj JPanel3.setLayout(new JPanel3FlowLayout());
            getJPanel3().add(getSubmitButton(), getSubmitButton().getName());
            getJPanel3().add(getDiscardButton(), getDiscardButton().getName());
            // user code begin (1)
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
        }
    return ivj JPanel3;
    } /*
    * Return the JPanel3FlowLayout property value.
    */
    java.awt.FlowLayout ivj JPanel3FlowLayout = null;
    try {
```java
/* create part */
ivjPanel3setLayout = new java.awt.FlowLayout();
ivjPanel3setLayout.setOrientation(java.awt.FlowLayout.RIGHT);
} catch (java.lang.Throwable ivjExc) {
    handleException(ivjExc);
}
return ivjPanel3FlowLayout;
}
/**
 * Return the JPanel4 property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJPanel4() {
    if (ivjPanel4 == null) {
        try {
            ivjPanel4 = new javax.swing.JPanel();
            ivjPanel4.setName("JPanel4");
            ivjPanel4.setLayout(null);
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjPanel4;
}
/**
 * Return the JSeparator1 property value.
 * @return javax.swing.JSeparator
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JSeparator getJSeparator1() {
    if (ivjJSeparator1 == null) {
        try {
            ivjJSeparator1 = new javax.swing.JSeparator();
            ivjJSeparator1.setName("JSeparator1");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJSeparator1;
}
/**
 * Return the JSplitPane property value.
 * @return javax.swing.JSplitPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JSplitPane getJSplitPanel() {
    if (ivjJSplitPanel == null) {
        try {
            ivjJSplitPanel = new javax.swing.JSplitPane(javax.swing.JSplitPane.VERTICAL_SPLIT);
            ivjJSplitPanel.setName("JSplitPanel");
            ivjJSplitPanel.setDividerLocation(200);
            getJSplitPanel().add(getJPanel1(), "top");
            getJSplitPanel().add(getJPanel2(), "bottom");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
        }
    }
    return ivjJSplitPanel;
}
```
// user code end
handleException(ivjExc);
}
}
return ivjJSplitPanel1;
}
/**
 * Return the JTextField1 property value.
 */
private javax.swing.JTextField getJTextField1() {
    try {
        ivjJTextField1 = new javax.swing.JTextField();
        ivjJTextField1.setName("JTextField1");
        ivjJTextField1.setPreferredSize(new java.awt.Dimension(4, 23));
        ivjJTextField1.setBorder(new javax.swing.plaf.BorderUIResource.CompoundBorderUIResource(null, null));
        ivjJTextField1.setMaximumSize(new java.awt.Dimension(2147483647, 23));
        ivjJTextField1.setMinimumSize(new java.awt.Dimension(4, 23));
        ivjJTextField1.setEditable(false);
        // user code begin (1)
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
    return ivjJTextField1;
}
/**
 * Return the JTextField11 property value.
 */
private javax.swing.JTextField getJTextField11() {
    try {
        ivjJTextField11 = new javax.swing.JTextField();
        ivjJTextField11.setName("JTextField11");
        ivjJTextField11.setBorder(new javax.swing.plaf.BorderUIResource.CompoundBorderUIResource(null, null));
        ivjJTextField11.setMaximumSize(new java.awt.Dimension(2147483647, 23));
        ivjJTextField11.setPreferredSize(new java.awt.Dimension(4, 23));
        ivjJTextField11.setMinimumSize(new java.awt.Dimension(4, 23));
        ivjJTextField11.setEditable(false);
        // user code begin (1)
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
    return ivjJTextField11;
}
/**
 * Return the JTextField12 property value.
 */
private javax.swing.JTextField getJTextField12() {
    try {

ivjTextField12 = new javax.swing.JTextField();
ivjTextField12.setName("JTextField12");
ivjTextField12.setPreferredSize(new java.awt.Dimension(4, 23));
ivjTextField12.setMaximumSize(new java.awt.Dimension(2169483647, 23));
// user code begin (1)
// user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
}
// user code end
return ivjTextField12;
}
/**
 * Return the JToolBar property value.
 * @return javax.swing.JToolBar
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JToolBar getJToolBar1() {
    if (ivjJToolBar1 == null) {
        try {
            ivjJToolBar1 = new javax.swing.JToolBar();
            ivjJToolBar1.setName("JToolBar1");
            getJToolBar1().add(getJToolBarButton1(), getJToolBarButton1().getName());
            ivjJToolBar1.addDefaultToolBarButton();
            ivjJToolBar1.addSeparator();
            getJToolBar1().add(getJToolBarButton2(), getJToolBarButton2().getName());
            getJToolBar1().add(getJToolBarButton3(), getJToolBarButton3().getName());
            // user code begin (1)
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
        // user code end
        return ivjJToolBar1;
    }
    /**
     * Return the JToolBarButton1 property value.
     * @return javax.swing.JButton
     */
    /** WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getJToolBarButton1() {
        if (ivjJToolBarButton1 == null) {
            try {
                ivjJToolBarButton1 = new javax.swing.JButton();
                ivjJToolBarButton1.setName("JToolBarButton1");
                ivjJToolBarButton1.setText("");
                ivjJToolBarButton1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
                ivjJToolBarButton1.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
                ivjJToolBarButton1.setMinimumSize(new java.awt.Insets(0, 0, 0));
                ivjJToolBarButton1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
                // user code begin (1)
                // user code end
                } catch (java.lang.Throwable ivjExc) {
                    // user code begin (2)
                    // user code end
                    handleException(ivjExc);
                }
            // user code end
            return ivjJToolBarButton1;
        }
/**
 * Return the JToolBarButton2 property value.
 * @return javax.swing.JButton
 */

private javax.swing.JButton getJToolBarButton2() {
  if (ivjJToolBarButton2 == null) {
    try {
      ivjJToolBarButton2 = new javax.swing.JToolBar();
      ivjJToolBarButton2.setName("JToolBarButton2");
      ivjJToolBarButton2.setText(" ");
      ivjJToolBarButton2.setVerticalTextPosition(javax.swing.SwingConstants.CENTER);
      ivjJToolBarButton2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/general/Help16.gif")));
      ivjJToolBarButton2.setMargin(new java.awt.Insets(0, 0, 0, 0));
      // user code begin (1)
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin (2)
      // user code end
      handleException(ivjExc);
    }
  }
  return ivjJToolBarButton2;
}

/**
 * Return the JToolBarButton3 property value.
 * @return javax.swing.JButton
 */

private javax.swing.JButton getJToolBarButton3() {
  if (ivjJToolBarButton3 == null) {
    try {
      ivjJToolBarButton3 = new javax.swing.JToolBar();
      ivjJToolBarButton3.setName("JToolBarButton3");
      ivjJToolBarButton3.setText(" ");
      ivjJToolBarButton3.setVerticalTextPosition(javax.swing.SwingConstants.CENTER);
      ivjJToolBarButton3.setMargin(new java.awt.Insets(0, 0, 0, 0));
      // user code begin (1)
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin (2)
      // user code end
      handleException(ivjExc);
    }
  }
  return ivjJToolBarButton3;
}

/**
 * Return the MessageDisplayArea property value.
 * @return javax.swing.JTextPane
 */

private javax.swing.JTextPane getMessageDisplayArea() {
  if (ivjMessageDisplayArea == null) {
    try {
      ivjMessageDisplayArea = new javax.swing.JTextPane();
      ivjMessageDisplayArea.setName("MessageDisplayArea");
      ivjMessageDisplayArea.setToolTipText("MessageDisplayArea");
      ivjMessageDisplayArea.setEnabled(false);
      // user code begin (1)
    }
ivjMessageDisplayArea.setBorder(BorderFactory.createEtchedBorder());
// user code end
}
} catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
return ivjMessageDisplayArea;
/**
 * Return the MessageDisplayArea property value.
 * @return javax.swing.JTextPane
 */
/**< WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing JTextPane getMessageDisplayAreaOnComposeBox() {
if ((ivjMessageDisplayAreaOnComposeBox == null)) {
try {
  ivjMessageDisplayAreaOnComposeBox = new javax.swing JTextPane();
  ivjMessageDisplayAreaOnComposeBox.setName("MessageDisplayAreaOnComposeBox");
  ivjMessageDisplayAreaOnComposeBox.setToolTipText("Message.composition.area");
  ivjMessageDisplayAreaOnComposeBox.setEnabled(true);
  // user code begin (1)
  ivjMessageDisplayAreaOnComposeBox.setBorder(BorderFactory.createEtchedBorder());
  // user code end
}
} catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
return ivjMessageDisplayAreaOnComposeBox;
/**
 * Return the MessagePanel property value.
 * @return javax.swing.JPanel
 */
/**< WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing JPanel getMessagePanel() {
if ((ivjMessagePanel == null)) {
try {
  ivjMessagePanel = new javax.swing JPanel();
  ivjMessagePanel.setName("MessagePanel");
  ivjMessagePanel.setLayout(getMessagePanelBoxLayout());
  ivjMessagePanel.add(getPanelHoldingHeaderOnComposeBox());
  getMessagePanel().add(getMessageDisplayAreaOnComposeBox(), getMessageDisplayAreaOnComposeBox());
  // user code begin (1)
  // user code end
}
} catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
return ivjMessagePanel;
/**
 * Return the MessagePanelBoxLayout property value.
 * @return javax.swingBoxLayout
 */
/**< WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swingBoxLayout getMessagePanelBoxLayout() {
javax.swingBoxLayout ivjMessagePanelBoxLayout = null;
try {
/* Create part */
}
ivjMessagePanelBoxLayout = new javax.swing.BoxLayout(getMessagePanel(), javax.swingBoxLayout.Y_AXIS);
}
catch (java.lang.Throwable ivjExc) {
    handleException(ivjExc);
}
return ivjMessagePanelBoxLayout;

/**
 * Return the NewMessageItem property value.
 * @return javax.swing.JMenu
 *
 */
private javax.swing.JMenuItem getNewMessageItem() {
    if (ivjNewMessageItem == null) {
        try {
            ivjNewMessageItem = new javax.swing.JMenuItem();
            ivjNewMessageItem.setName("NewMessageItem");
            ivjNewMessageItem.setText("New, message...");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjNewMessageItem;
}

/**
 * Return the NewThreadItem property value.
 * @return javax.swing.JMenuItem
 *
 */
private javax.swing.JMenuItem getNewThreadItem() {
    if (ivjNewThreadItem == null) {
        try {
            ivjNewThreadItem = new javax.swing.JMenuItem();
            ivjNewThreadItem.setName("NewThreadItem");
            ivjNewThreadItem.setText("New, thread");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjNewThreadItem;
}

/**
 * Return the PanelHoldingHeader property value.
 * @return javax.swing.JPanel
 *
 */
private javax.swing.JPanel getPanelHoldingHeader() {
    if (ivjPanelHoldingHeader == null) {
        try {
            ivjPanelHoldingHeader = new javax.swing.JPanel();
            ivjPanelHoldingHeader.setName("PanelHoldingHeader");
            ivjPanelHoldingHeader.setLayout(new javax.swing.BoxLayout.getPanelHoldingHeader(), javax.swing.BoxLayout.X_AXIS);
            ivjPanelHoldingHeader.setMaximumSize(new java.awt.Dimension(496, 120));
            ivjPanelHoldingHeader.setPreferredSize(new java.awt.Dimension(496, 70));
            ivjPanelHoldingHeader.setAlignmentX(java.awt.Component.LEFT_ALIGNMENT);
            ivjPanelHoldingHeader.setMinimumSize(new java.awt.Dimension(496, 120));
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
getPanelHoldingHeader().add(getPanelHoldingHeadingLabels(), getPanelHoldingHeadingLabels().
getName());
getPanelHoldingHeader().add(getPanelHoldingHeaderFields(), getPanelHoldingHeaderFields().getName()
());
// user code begin (1)
// user code end
} catch (javax.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
return ivjPanelHoldingHeader;
}/**
 * Return the PanelHoldingHeaderFields property value.
 * @return javax.swing.JPanel
*/
/**< WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getPanelHoldingHeaderFields() {
if (ivjPanelHoldingHeaderFields == null) {
try {
ivjPanelHoldingHeaderFields = new javax.swing.JPanel();
ivjPanelHoldingHeaderFields.setName("PanelHoldingHeaderFields");
ivjPanelHoldingHeaderFields.setLayout(getPanelHoldingHeaderFieldsBoxLayout());
ivjPanelHoldingHeaderFields.setMaximumSize(new java.awt.Dimension(460, 69));
ivjPanelHoldingHeaderFields.setMinimumSize(new java.awt.Dimension(460, 69));
getPanelHoldingHeaderFields().add(getAuthor(), getAuthor().getName());
getPanelHoldingHeaderFields().add(getLabel12(), getLabel12().getName());
getPanelHoldingHeaderFields().add(getLabel11(), getLabel11().getName());
// user code begin (1)
ivjPanelHoldingHeaderFields.setBorder(BorderFactory.createMatteBorder(6, 6, 6, 6,
ivjPanelHoldingHeaderFields.getBackground()));
// user code end
} catch (javax.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
return ivjPanelHoldingHeaderFields;
}
/**< WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.BoxLayout getPanelHoldingHeaderFieldsBoxLayout() {
javax.swing<BoxLayout> ivjPanelHoldingHeaderFieldsBoxLayout = null;
try {
/**< create part */
ivjPanelHoldingHeaderFieldsBoxLayout = new javax.swing.BoxLayout(getPanelHoldingHeaderFields(),
javax.swing.BoxLayout.Y_AXIS);
} catch (javax.lang.Throwable ivjExc) {
handleException(ivjExc);
}
return ivjPanelHoldingHeaderFieldsBoxLayout;
/**< WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getPanelHoldingHeaderFieldsUnComposeBox() {
if (ivjPanelHoldingHeaderFieldsUnComposeBox == null) {
try {

283
ivJPanelHoldingHeaderFieldsOnComposeBox = new javax.swing.JPanel();
ivJPanelHoldingHeaderFieldsOnComposeBox.setName("PanelHoldingHeaderFieldsOnComposeBox");
ivJPanelHoldingHeaderFieldsOnComposeBox.setPreferredSize(new java.awt.Dimension(470000, 76));
ivJPanelHoldingHeaderFieldsOnComposeBox.setLayout();
ivJPanelHoldingHeaderFieldsOnComposeBox.setMaximumSize(new java.awt.Dimension(450000, 76));
ivJPanelHoldingHeaderFieldsOnComposeBox.add(getJTextField1());
getPanelHoldingHeaderFieldsOnComposeBox().add(getJTextField11(), getJTextField11().getName());
getPanelHoldingHeaderFieldsOnComposeBox().add(getJTextField12(), getJTextField12().getName());
// user code begin (1)
ivJPanelHoldingHeaderFieldsOnComposeBox.setBorder(BorderFactory.createMatteBorder(6, 6, 6, 6,
ivJPanelHoldingHeaderFieldsOnComposeBox.getBackground()));

// user code end
}
)
} catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
}
} handleException(ivjExc);
}

/**
 * Return the PanelHoldingHeaderFieldsOnComposeBoxBoxLayout property value.
 */
private javax.swing.BoxLayout getPanelHoldingHeaderFieldsOnComposeBoxBoxLayout() {
    javax.swing.BoxLayout ivjPanelHoldingHeaderFieldsOnComposeBoxBoxLayout = null;
    try {
        /* Create part */
        ivjPanelHoldingHeaderFieldsOnComposeBoxBoxLayout = new javax.swing.BoxLayout(
            getPanelHoldingHeaderFieldsOnComposeBox(), javax.swingBoxLayout.Y_AXIS);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    return ivjPanelHoldingHeaderFieldsOnComposeBoxBoxLayout;
}

/**
 * Return the PanelHoldingHeader1 property value.
 */
private javax.swing.JPanel getPanelHoldingHeader1() {
    try {
        ivJPanelHoldingHeader1 = new javax.swing.JPanel();
        ivJPanelHoldingHeader1.setName("PanelHoldingHeader1");
        ivJPanelHoldingHeader1.setPreferredSize(new java.awt.Dimension(496000, 120));
        ivJPanelHoldingHeader1.setAlignmentX();
        ivJPanelHoldingHeader1.setMinimumSize(new java.awt.Dimension(496, 120));
        getPanelHoldingHeader1().add(getPanelHoldingHeadingLabelOnComposeBox(),
            getPanelHoldingHeadingLabelOnComposeBox().getName());
        getPanelHoldingHeader1().add(getPanelHoldingHeaderFieldsOnComposeBox(),
            getPanelHoldingHeaderFieldsOnComposeBox().getName());
    }
    catch (java.lang.Throwable ivjExc) {
        // user code begin (1)
    }
    // user code end
}
}
} catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
}
} handleException(ivjExc);
return ivjPanel.HoldingHeaderOnComposeBox;

/**
 * Return the PanelHoldingHeadingLabels property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getPanelHoldingHeadingLabels() {
    if (ivjPanelHoldingHeadingLabels == null) {
        try {
            ivjPanelHoldingHeadingLabels = new javax.swing.JPanel();
            ivjPanelHoldingHeadingLabels.setName("PanelHoldingHeadingLabels");
            ivjPanelHoldingHeadingLabels.setLayout(getPanelHoldingHeadingLabelsBoxLayout());
            getPanelHoldingHeadingLabels().add(getAuthorLabel(), getAuthorLabel().getName());
            getPanelHoldingHeadingLabels().add(getDateLabel(), getDateLabel().getName());
            getPanelHoldingHeadingLabels().add(getSubjectLabel(), getSubjectLabel().getName());
            // user code begin (1)
            ivjPanelHoldingHeadingLabels.setBorders(BorderFactory.createMatteBorder(6, 6, 6, 6, 6, ivjPanelHoldingHeadingLabels.getBackground()));
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjPanelHoldingHeadingLabels;
}

/**
 * Return the PanelHoldingHeadingLabelsBoxLayout property value.
 * @return javax.swing.BoxLayout
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.BoxLayout getPanelHoldingHeadingLabelsBoxLayout() {
    javax.swing.BoxLayout ivjPanelHoldingHeadingLabelsBoxLayout = null;
    try {
        //</use>
        ivjPanelHoldingHeadingLabelsBoxLayout = new javax.swing.BoxLayout(getPanelHoldingHeadingLabels(), javax.swing.BoxLayout.Y_AXIS);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    return ivjPanelHoldingHeadingLabelsBoxLayout;
}

/**
 * Return the PanelHoldingHeadingLabels1 property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getPanelHoldingHeadingLabels1OnComposeBox() {
    if (ivjPanelHoldingHeadingLabels1OnComposeBox == null) {
        try {
            ivjPanelHoldingHeadingLabels1OnComposeBox = new javax.swing.JPanel();
            ivjPanelHoldingHeadingLabels1OnComposeBox.setName("PanelHoldingHeadingLabels1OnComposeBox");
            ivjPanelHoldingHeadingLabels1OnComposeBox.setPreferredSize(new java.awt.Dimension(65, 70));
            ivjPanelHoldingHeadingLabels1OnComposeBox.setLayout(getPanelHoldingHeadingLabels1OnComposeBoxBoxLayout());
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (1)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjPanelHoldingHeadingLabels1OnComposeBox;
}
ivjPanelHoldingHeadingLabelsOnComposeBox.setBorder(BorderFactory.createMatteBorder(6, 6, 6, 6, ivjPanelHoldingHeadingLabelsOnComposeBox.getBackground()));

} catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
}

return ivjPanelHoldingHeadingLabelsOnComposeBox;

/**
 * Return the JPanelHoldingHeadingLabelsOnComposeBoxBoxLayout property value.
 * @return javax.swing.BoxLayout
 */
private javax.swing.BoxLayout getPanelHoldingHeadingLabelsOnComposeBoxBoxLayout() {
    try {
        /* Create part */
        ivjPanelHoldingHeadingLabelsOnComposeBoxBoxLayout = new javax.swing.BoxLayout(
            getPanelHoldingHeadingLabelsOnComposeBox(), javax.swingBoxLayout.Y_AXIS);

    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }

    return ivjPanelHoldingHeadingLabelsOnComposeBoxBoxLayout;

} /**
 * Return the JLabel property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getSubjectLabel() {
    if (ivjSubjectLabel == null) {
        try {
            ivjSubjectLabel = new javax.swing.JLabel();
            ivjSubjectLabel.setName("SubjectLabel");
            ivjSubjectLabel.setText("Subject");
            ivjSubjectLabel.setMinimumSize(new java.awt.Dimension(46, 23));
            ivjSubjectLabel.setPreferredSize(new java.awt.Dimension(46, 23));
            // user code begin (1)
            // user code end

        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }

    }

return ivjSubjectLabel;

} /**
 * Return the JLabel property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getSubjectLabel1() {
    if (ivjSubjectLabel1 == null) {
        try {
            ivjSubjectLabel1 = new javax.swing.JLabel();
            ivjSubjectLabel1.setName("SubjectLabel1");
            ivjSubjectLabel1.setPreferredSize(new java.awt.Dimension(46, 23));
            ivjSubjectLabel1.setText("Subject:");
            ivjSubjectLabel1.setMaximumSize(new java.awt.Dimension(46, 23));
            ivjSubjectLabel1.setMinimumSize(new java.awt.Dimension(46, 23));
            // user code begin (1)
            // user code end

        } catch (java.}
```java
1308     } catch (java.lang.Throwable ivjExc) {
1309         // user code begin {2}
1310         // user code end
1311         handleException(ivjExc);
1312     }
1313     }
1314     return ivjSubjectLabel1;
1315 }
1316 /**
1317  * Return the JButton1 property value.
1318  */
1319  private javax.swing.JButton getSubmitButton() {
1320     if (ivjSubmitButton == null) {
1321         try {
1322             ivjSubmitButton = new javax.swing.JButton();
1323             ivjSubmitButton.setName("SubmitButton");
1324             ivjSubmitButton.setText("Submit");
1325             // user code begin {1}
1326             // user code end
1327         } catch (java.lang.Throwable ivjExc) {
1328             // user code begin {2}
1329             // user code end
1330         }
1331     }
1332     return ivjSubmitButton;
1333 }
1334 /**
1335  * Called whenever the part throws an exception.
1336  * @param exception java.lang.Throwable
1337  */
1338  private void handleException(java.lang.Throwable exception) {
1339      /* Uncomment the following lines to print uncaught exceptions to stdout */
1340      // System.out.println("---------- UNCAUGHT EXCEPTION ----------");
1341      // exception.printStackTrace(System.out);
1342  }
1343 /**
1344  * Initializes connections
1345  * @Exception java.lang.Exception The exception description.
1346  */
1347  private void initConnections() throws java.lang.Exception {
1348       // user code begin {1}
1349       // user code end
1350       JMenuItem().addActionListener(ivjEventHandler);
1351       JToolBarButton().addActionListener(ivjEventHandler);
1352  }
1353  */
1354  private void initialize() {
1355         try {
1356             // user code begin {1}
1357             // user code end
1358             setName("DiscussionFrame");
1359             setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
1360             setTitle("Discussion of xxx.yyy");
1361             setSize(600, 600);
1362             setJMenuBar(getDiscussionFrameJMenuBar());
1363             getContentPane().add(getFrameContentPane());
1364             initConnections();
1365         } catch (java.lang.Throwable ivjExc) {
287
```
```java
package clientapp;

import java.awt.Dimension;
import java.lang.Exception;

/**
 * Default layout class for the client panorama. Manages a rectangular grid ordering that ought to be
 * easily customizable.
 * Creation date: (22-06-00 11:42:53)
 * Author:
 */
public class GridLayout extends PanoramaLayout {
    private int width;
    private int height;

    /**
     * GridLayout constructor comment.
     */
    public GridLayout() {
        super();
        setHeight( Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
                getGRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS() ) ));
        setWidth( Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
                getGRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS() ) ));
    }

    /**
     * Convert the vertical and horizontal indices into a x,y coordinate pair designating a position in the
     * panorama.
     * Creation date: (22-06-00 12:02:00)
     */
    public java.awt.Dimension computePlacement() throws Exception {
        Dimension newPlacement = new Dimension( (int) (horizontalIndex * (int)(preferredDocumentSize.
                getWeight() + getXSpacing())),
            (int) (verticalIndex * (int)(preferredDocumentSize.getHeight() +
                getYSpacing())));
        if (horizontalIndex >= width)
```

Listing C.11: GridLayout.java
```java
{ if (verticalIndex >= height) 
{ throw new Exception( ClientConstants.OUT_OF_BOUNDS_EXCEPTION_TEXT() ); 
} else
40 { verticalIndex++; 
42 } horizontalIndex = 0; 
} } else
46 { horizontalIndex++; 
48 } return newPlacement; 
} */
/** 
* Insert the method's description here. 
* Creation date: (22-06-00 11:43:44) 
* @return int */
56 public int getHeight() { return height; 
58 } */
/** 
* Compute and return the dimensions of a rectangle containing all the document nodes being laid out. 
* Creation date: (01-06-00 19:43:12) 
* @return java.awt.Rectangle */
64 public java.awt.Rectangle getOverviewRectangle() 
{ java.awt.Rectangle newRectangle = new java.awt.Rectangle(); 
if ( getVerticalIndex() == 0 ) 
68 { 
newRectangle.setRect( -(getXSpacing()/2), 0, 
getHorizontalIndex() * ( getPreferredDocumentSize().getWidth() + getXSpacing() ), 
getVerticalIndex() * ( getPreferredDocumentSize().getHeight() + getYSpacing() )); 
72 } else
74 { 
newRectangle.setRect( -(getXSpacing()/2), 0, 
getWidth() * ( getPreferredDocumentSize().getWidth() + getXSpacing() ), 
getHeight() * ( getPreferredDocumentSize().getHeight() + getYSpacing() )); 
78 } return newRectangle; 
} */
/** 
* Insert the method’s description here. 
* Creation date: (22-06-00 11:43:29) 
* @return int */
86 public int getWidth() { return width; 
88 } */
```
Listing C.12: GridPanoramaLayoutManager.java

class GridPanoramaLayoutManager extends ClientPanoramaLayoutManager {
    private javax.swing.JPanel ivjButtonPanel = null;
    private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
    private javax.swing.JButton ivjCancelButton = null;
    private javax.swing.JTextField ivjDocumentHeightField = null;
    private javax.swing.JLabel ivjDocumentHeightLabel = null;
    private javax.swing.JTextField ivjDocumentWidthField = null;
    private javax.swing.JLabel ivjDocumentWidthLabel = null;
    private javax.swing.JTextField ivjHorizontalSpacingField = null;
    private javax.swing.JLabel ivjHorizontalSpacingLabel = null;
    private javax.swing.JPanel ivjDialogContentPane = null;
    private javax.swing.JPanel ivjMainPanel = null;
    private javax.swing.JButton ivjOKButton = null;
    private javax.swing.JTextField ivjVerticalSpacingField = null;
    private javax.swing.JLabel ivjVerticalSpacingLabel = null;
    private final static java.lang.String GRID_LAYOUT_DOCUMENT_HORIZIONTAL_SPACING = "GridLayout_document_horizontal_spacing";
    private final static java.lang.String GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING = "GridLayout_document_vertical_spacing";
    private final static int GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE = 200;
    private final static int GRID_LAYOUT_DOCUMENT_HORIZIONTAL_SPACING_DEFAULT_VALUE = 200;

    public class CancelHandler implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
            setCommitted(false);
            getContentPane().dispose();
        }
    }

    public class OKHandler implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
        }
    }
}
```java
setCommitted( true );
getGridLayoutCustomizationBox().dispose();
}
private boolean committed = false;
private final static java.lang.String GRID_LAYOUTDOCUMENT_HEIGHT = "GridLayout.document.height";
private final static java.lang.String GRID_LAYOUTDOCUMENT_WIDTH = "GridLayout.document.width";
private final static int GRID_LAYOUTDOCUMENT_WIDTH_DEFAULT_VALUE = 600;
private final static int GRID_LAYOUTDOCUMENT_HEIGHT_DEFAULT_VALUE = 600;
/**
 * GridPanoramaLayoutManager constructor comment.
 */
public GridPanoramaLayoutManager() {
    super();
    initialize();
}
/**
 * Insert the method’s description here.
 * Creation date: (12-10-00 11:50:40)
 */
public Object clone() {
    return null;
}
/**
 * Computes a list of Point objects representing a grid structure on which the nodes can subsequently be
 * distributed.
 */
public java.util.ArrayList computeCoordinates(edu.umd.cs.jazz.ZNode[] nodes) {
    int xCoordinate = 0;
    int yCoordinate = 0;
    int xIncrement = 0;
    int yIncrement = 0;
    // let document with the largest bounding box determine the spacing between all the documents
    for ( int i = 0; i < nodes.length; i++ )
    {
        xIncrement = (int)Math.round( nodes[i].getBounds().getWidth() ) > xIncrement ? (int)Math.round( nodes[i].getBounds().getWidth() ) : xIncrement;
        yIncrement = (int)Math.round( nodes[i].getBounds().getHeight() ) > yIncrement ? (int)Math.round( nodes[i].getBounds().getHeight() ) : yIncrement;
    }
    xIncrement += Integer.parseInt( getHorizontalSpacingField().getText() );
    yIncrement += Integer.parseInt( getVerticalSpacingField().getText() );
    int sideLength = Math.round( (float) Math.ceil( Math.sqrt( (double) nodes.length ) ) );
    ArrayList list = new ArrayList();
    for ( int i = 0; i < nodes.length; i++ )
    {
        if ( (i % sideLength) == 0 )
        {
            xCoordinate = 0;
        }
        else
        {
            xCoordinate += xIncrement;
        }
        yCoordinate = (i / sideLength ) * yIncrement;
        list.add( new java.awt.Point( xCoordinate, yCoordinate ) );
        // System.out.println( String.valueOf( xCoordinate ) + "." + String.valueOf( yCoordinate ) );
    }
    return list;
}
/**
 * Distribute a set of nodes at positions given in the second argument list.
 * Creation date: (29-08-00 08:28:01)
 */
```
public void distributeCoordinates(eduumd.cs.jazz.ZNode[] nodes, ArrayList coordinates) {
    // translate as many nodes as there are coordinate pairs and leave the remainder unchanged
    int i, j;
    for (i = 0; j = 0; i < coordinates.size(); i++) {
        if (nodeLocked(nodes[i]) == false) {
            java.awt.Point coordinatePair = (java.awt.Point) coordinates.get(j);
            setPosition(nodes[i], coordinatePair);
            j++;
        }
    }
}

/**
 * Insert the method's description here.
 * @param nodes edu.umd.cs.jazz.ZNode[]
 */
public void doLayout(edu.umd.cs.jazz.ZNode[] nodes) {
    // System.out.println("doLayout");
    distributeCoordinates(nodes, computeCoordinates(nodes));
}

/**
 * Insert the method's description here.
 * @param node edu.umd.cs.jazz.ZGroup
 */
public void doLayout(edu.umd.cs.jazz.ZGroup node) {
}

/**
 * Return the ButtonPanel property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getButtonPanel() {
    if (ivjButtonPanel == null) {
        try {
            ivjButtonPanel = new javax.swing.JPanel();
            ivjButtonPanel.setName("ButtonPanel");
            ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
            getButtonPanel().add(getOKButton(), getOKButton().getName());
            getButtonPanel().add(getCancelButton(), getCancelButton().getName());
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjButtonPanel;
}

/**
 * Return the ButtonPanelFlowLayout property value.
 * @return java.awt.FlowLayout
 */
private java.awt.FlowLayout getButtonPanelFlowLayout() {
    java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
    try {
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    return ivjButtonPanel;
}
/* Create part */
ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
} catch (java.lang.Throwable ivjExc) {
  handleException(ivjExc);
};
return ivjButtonPanelFlowLayout;

/**
 * Return the CancelButton property value.
 */
private javax.swing.JButton getCancelButton() {
  if (ivjCancelButton == null) {
    try {
      ivjCancelButton = new javax.swing.JButton();
      ivjCancelButton.setName("CancelButton");
      ivjCancelButton.setMnemonic('C');
      ivjCancelButton.setText("Cancel");
      // user code begin [1]
      ivjCancelButton.addActionListener(new CancelHandler());
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin [2]
      // user code end
      handleException(ivjExc);
    }
  }
  return ivjCancelButton;
}

/**
 * Return the DocumentHeightField property value.
 */
private javax.swing.JTextField getDocumentHeightField() {
  if (ivjDocumentHeightField == null) {
    try {
      ivjDocumentHeightField = new javax.swing.JTextField();
      ivjDocumentHeightField.setName("DocumentHeightField");
      // user code begin [1]
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin [2]
      // user code end
      handleException(ivjExc);
    }
  }
  return ivjDocumentHeightField;
}

/**
 * Return the DocumentHeightLabel property value.
 */
private javax.swing.JLabel getDocumentHeightLabel() {
  if (ivjDocumentHeightLabel == null) {
    try {
      ivjDocumentHeightLabel = new javax.swing.JLabel();
      ivjDocumentHeightLabel.setName("DocumentHeightLabel");
      ivjDocumentHeightLabel.setText("Document Height:");
      // user code begin [1]
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin [2]
242     // user code end
243     handleException(ivjExc);
244     }
245     return ivjDocumentHeightLabel;
246 }
247 /**<
248 * Return the dimensions of a document.
249 * Creation date: (06-09-00 00:46:03)
250 * @return java.awt.Dimension
251 */
252 public java.awt.Dimension getDocumentSize()
253 {
254     int height = getDocumentHeightField().getText().equals("") ?
255         getGRID_LAYOUT_DOCUMENT_HEIGHT_DEFAULT_VALUE() :
256         Integer.parseInt( getDocumentHeightField().getText() );
257     int width = getDocumentWidthField().getText().equals("") ?
258         getGRID_LAYOUT_DOCUMENT_WIDTH_DEFAULT_VALUE() :
259         Integer.parseInt( getDocumentWidthField().getText() );
260     return new java.awt.Dimension( height, width );
261 }
262 /**<
263 * Return the DocumentWidthField property value.
264 * @return javax.swing.JTextField
265 */
266 private javax.swing.JTextField getDocumentWidthField() {
267     try {
268         ivjDocumentWidthField = new javax.swing.JTextField();
269         ivjDocumentWidthField.setName("DocumentWidthField");
270         // user code begin (1)
271         // user code end
272     } catch (java.lang.Throwable ivjExc) {
273         // user code begin (2)
274         // user code end
275         handleException(ivjExc);
276     }
277     return ivjDocumentWidthField;
278 }
279 /**<
280 * Return the DocumentWidthLabel property value.
281 * @return javax.swing.JLabel
282 */
283 private javax.swing.JLabel getDocumentWidthLabel() {
284     try {
285         ivjDocumentWidthLabel = new javax.swing.JLabel();
286         ivjDocumentWidthLabel.setName("DocumentWidthLabel");
287         ivjDocumentWidthLabel.setText("Document.width:");
288         // user code begin (1)
289         // user code end
290     } catch (java.lang.Throwable ivjExc) {
291         // user code begin (2)
292         // user code end
293         handleException(ivjExc);
294     }
295     return ivjDocumentWidthLabel;
296 }
297 /**<
298 * Insert the method's description here.
299 * Creation date: (12-10-00 16:01:26)
300 * @return java.lang.String
public final static java.lang.String getGRID_LAYOUT_DOCUMENT_HEIGHT() {
    return GRID_LAYOUT_DOCUMENT_HEIGHT;
}

/*
 * Insert the method’s description here.
 * Creation date: (12-10-00 16:18:11)
 * @return int
 */
public final static int getGRID_LAYOUT_DOCUMENT_HEIGHT_DEFAULT_VALUE() {
    return GRID_LAYOUT_DOCUMENT_HEIGHT_DEFAULT_VALUE;
}

/*
 * Insert the method’s description here.
 * Creation date: (12-10-00 13:49:28)
 * @return java.lang.String
 */
public final static java.lang.String getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING() {
    return GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING;
}

/*
 * Insert the method’s description here.
 * Creation date: (12-10-00 14:35:06)
 * @return int
 */
public final static int getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING_DEFAULT_VALUE() {
    return GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING_DEFAULT_VALUE;
}

/*
 * Insert the method’s description here.
 * Creation date: (12-10-00 13:49:50)
 * @return java.lang.String
 */
public final static java.lang.String getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING() {
    return GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING;
}

/*
 * Insert the method’s description here.
 * Creation date: (12-10-00 14:34:26)
 * @return int
 */
public final static int getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE() {
    return GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE;
}

/*
 * Insert the method’s description here.
 * Creation date: (12-10-00 16:01:47)
 * @return java.lang.String
 */
public final static java.lang.String getGRID_LAYOUT_DOCUMENT_WIDTH() {
    return GRID_LAYOUT_DOCUMENT_WIDTH;
}

/*
 * Insert the method’s description here.
 * Creation date: (12-10-00 16:17:06)
 * @return int
 */
public final static int getGRID_LAYOUT_DOCUMENT_WIDTH_DEFAULT_VALUE() {
    return GRID_LAYOUT_DOCUMENT_WIDTH_DEFAULT_VALUE;
}

/*
 * Return the GridLayoutCustomizationBox property value.
 */
public javax.swing.JDialog getGridLayoutCustomizationBox() {

/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JDialog getGridLayoutCustomizationBox() {
if (ivjGridLayoutCustomizationBox == null) {
    try {
        ivjGridLayoutCustomizationBox = new javax.swing.JDialog();
        ivjGridLayoutCustomizationBox.setName("GridLayoutCustomizationBox");
        ivjGridLayoutCustomizationBox.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        ivjGridLayoutCustomizationBox.setBounds(24, 44, 398, 300);
        ivjGridLayoutCustomizationBox.setModal(true);
        ivjGridLayoutCustomizationBox.setTitle("GridLayoutCustomizationBox");
        getGridLayoutCustomizationBox().setContentPane(getDialogContentPane());
        // user code begin {1}
        // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjGridLayoutCustomizationBox;
}
/**
 * Return the HorizontalSpacingField property value.
 * @return javax.swing.JTextField
 */
private javax.swing.JTextField getHorizontalSpacingField() {
    if (ivjHorizontalSpacingField == null) {
        try {
            ivjHorizontalSpacingField = new javax.swing.JTextField();
            ivjHorizontalSpacingField.setName("HorizontalSpacingField");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjHorizontalSpacingField;
}
/**
 * Return the HorizontalSpacingLabel property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getHorizontalSpacingLabel() {
    if (ivjHorizontalSpacingLabel == null) {
        try {
            ivjHorizontalSpacingLabel = new javax.swing.JLabel();
            ivjHorizontalSpacingLabel.setName("HorizontalSpacingLabel");
            ivjHorizontalSpacingLabel.setText("HorizontalSpacing");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjHorizontalSpacingLabel;
}
/**
 * Return theDialogContentPane property value.
 * @return javax.swing.JPanel
 */
/*
*/
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getDialogContentPanel() {
    if (ivjDialogContentPanel == null) {
        try {
            ivjDialogContentPanel = new javax.swing.JPanel();
            ivjDialogContentPanel.setName("JDialogContentPane");
            ivjDialogContentPanel.setLayout(new java.awt.BorderLayout());
            getDialogContentPanel().add(getMainPanel(), "Center");
            getDialogContentPanel().add(getButtonPanel(), "South");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjDialogContentPanel;
}

/**
 * Return the MainPanel property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getMainPanel() {
    if (ivjMainPanel == null) {
        try {
            ivjMainPanel = new javax.swing.JPanel();
            ivjMainPanel.setName("MainPanel");
            ivjMainPanel.setLayout(new java.awt.GridBagLayout());

            constraintsDocumentWidthLabel.gridx = 0;
            constraintsDocumentWidthLabel.gridy = 0;
            constraintsDocumentWidthLabel.anchor = java.awt.GridBagConstraints.WEST;
            getMainPanel().add(getDocumentWidthLabel(), constraintsDocumentWidthLabel);

            java.awt.GridBagConstraints constraintsDocumentHeightLabel = new java.awt.GridBagConstraints();
            constraintsDocumentHeightLabel.gridx = 0;
            constraintsDocumentHeightLabel.gridy = 1;
            constraintsDocumentHeightLabel.anchor = java.awt.GridBagConstraints.WEST;
            getMainPanel().add(getDocumentHeightLabel(), constraintsDocumentHeightLabel);

            java.awt.GridBagConstraints constraintsHorizontalSpacingLabel = new java.awt.GridBagConstraints();
            constraintsHorizontalSpacingLabel.gridx = 0;
            constraintsHorizontalSpacingLabel.gridy = 2;
            constraintsHorizontalSpacingLabel.anchor = java.awt.GridBagConstraints.WEST;
            getMainPanel().add(getHorizontalSpacingLabel(), constraintsHorizontalSpacingLabel);

            java.awt.GridBagConstraints constraintsVerticalSpacingLabel = new java.awt.GridBagConstraints();
            constraintsVerticalSpacingLabel.gridx = 0;
            constraintsVerticalSpacingLabel.gridy = 3;
            constraintsVerticalSpacingLabel.anchor = java.awt.GridBagConstraints.WEST;
            getMainPanel().add(getVerticalSpacingLabel(), constraintsVerticalSpacingLabel);

            java.awt.GridBagConstraints constraintsDocumentWidthField = new java.awt.GridBagConstraints();
            constraintsDocumentWidthField.gridx = 1;
            constraintsDocumentWidthField.gridy = 0;
            constraintsDocumentWidthField.fill = java.awt.GridBagConstraints.HORIZONTAL;
            getMainPanel().add(getDocumentWidthField(), constraintsDocumentWidthField);
            getMainPanel().add(getDocumentWidthField().getVal(), constraintsDocumentWidthField);
            getMainPanel().add(getDocumentWidthField().getAnchor(), constraintsDocumentWidthField);
            getMainPanel().add(getDocumentWidthField().getWeightx(), constraintsDocumentWidthField);
            getMainPanel().add(getDocumentWidthField().getWeighty(), constraintsDocumentWidthField);
            getMainPanel().add(getDocumentWidthField().getInsets(), constraintsDocumentWidthField);
        }
    }
    return ivjMainPanel;
getMainPanel().add(getDocumentWidthField(), constraintsDocumentWidthField);

java.awt.GridBagConstraints constraintsDocumentHeightField = new java.awt.GridBagConstraints();
constraintsDocumentHeightField.gridx = 1; constraintsDocumentHeightField.gridy = 1;
constraintsDocumentHeightField.fill = java.awt GridBagConstraints.HORIZONTAL;
constraintsDocumentHeightField.anchor = java.awt GridBagConstraints.EAST;
constraintsDocumentHeightField.weightx = 1.0;
constraintsDocumentHeightField.weighty = 1.0;
getMainPanel().add(getDocumentHeightField(), constraintsDocumentHeightField);

java.awt GridBagConstraints constraintsHorizontalSpacingField = new java.awt GridBagConstraints();
constraintsHorizontalSpacingField.gridx = 1; constraintsHorizontalSpacingField.gridy = 2;
constraintsHorizontalSpacingField.fill = java.awt GridBagConstraints.HORIZONTAL;
constraintsHorizontalSpacingField.anchor = java.awt GridBagConstraints.EAST;
constraintsHorizontalSpacingField.weightx = 1.0;
constraintsHorizontalSpacingField.weighty = 1.0;
getMainPanel().add(getHorizontalSpacingField(), constraintsHorizontalSpacingField);

java.awt GridBagConstraints constraintsVerticalSpacingField = new java.awt GridBagConstraints();
constraintsVerticalSpacingField.gridx = 1; constraintsVerticalSpacingField.gridy = 3;
constraintsVerticalSpacingField.fill = java.awt GridBagConstraints.HORIZONTAL;
constraintsVerticalSpacingField.anchor = java.awt GridBagConstraints.EAST;
constraintsVerticalSpacingField.weightx = 1.0;
constraintsVerticalSpacingField.weighty = 1.0;
getMainPanel().add(getVerticalSpacingField(), constraintsVerticalSpacingField);

try {  // user code begin {1}
    ivjMainPanel.setBorder( ClientConstants.createDialogBorder( ivjMainPanel.getBackground() ));
    // user code end }
} catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
}

return ivjMainPanel;

/**
 * Return the OKButton property value.
 * @return java.swing.JButton
 */

private java.swing.JButton getOKButton() {
    if (ivjOKButton == null) {
        try {
            ivjOKButton = new java.swing.JButton();
            ivjOKButton.setName("OKButton");
            ivjOKButton.setMnemonic('O');
            ivjOKButton.setText("OK");
            // user code begin {1}
            ivjOKButton.addActionListener( new OKHandler() );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }

    return ivjOKButton;
}

/**
 * Return the VerticalSpacingField property value.
 * @return java.swing.JTextField
 */

298
private javax.swing.JTextField getVerticalSpacingField() {
    if (ivjVerticalSpacingField == null) {
        try {
            ivjVerticalSpacingField = new javax.swing.JTextField();
            ivjVerticalSpacingField.setName("VerticalSpacingField");
            // user code begin {1}
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
                // user code end
                handleException(ivjExc);
            }
            return ivjVerticalSpacingField;
        }
        */
        * WARNING: THIS METHOD WILL BE REGENERATED. */
        * Return the VerticalSpacingLabel property value. *
        * @return javax.swing.JLabel *
        */
        private javax.swing.JLabel getVerticalSpacingLabel() {
            if (ivjVerticalSpacingLabel == null) {
                try {
                    ivjVerticalSpacingLabel = new javax.swing.JLabel();
                    ivjVerticalSpacingLabel.setName("VerticalSpacingLabel");
                    ivjVerticalSpacingLabel.setText("VerticalSpacing:");
                    // user code begin {1}
                    // user code end
                    } catch (java.lang.Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                    }
                    return ivjVerticalSpacingLabel;
                }
                */
                * Called whenever the part throws an exception. *
                * @param exception java.lang.Throwable *
                */
                private void handleException(java.lang.Throwable exception) {
                    /* Uncomment the following lines to print uncaught exceptions to stdout */
                    // System.out.println("--------- UNCAUGHT EXCEPTION --------");
                    // exception.printStackTrace(System.out);
                    */
                    * Initialize the class. *
                    */
                    private void initialize() {
                        try {
                            // user code begin {1}
                            // user code end
                            } catch (java.lang.Throwable ivjExc) {
                                handleException(ivjExc);
                            }
                            // user code begin {2}
                            // user code end
                        }
                        */
                        * Insert the method's description here. *
                        * Creation date: (12-10-00 15:31:06) *
                        * @return boolean
```java
/*
 * The main entry point of the program. It sets up the new committed state when the program is finished.
 *
 * @param committed boolean
 */
public static void main(java.lang.String[] args)
{
  try {
    GridPanoramaLayoutManager aGridPanoramaLayoutManager
        = new GridPanoramaLayoutManager();
    System.err.println("Exception, occurred in main() of clientapp.ClientPanoramaLayout");
    exception.printStackTrace(System.out);
  }
  catch (Throwable exception)
  {
    System.err.println("Exception, occurred in main() of clientapp.ClientPanoramaLayout");
    exception.printStackTrace(System.out);
  }
  /* Insert the method's description here. */
  System.out.println("Creation date: (12-10-00 16:31:06)\n" + committed + newCommitted);
  /* Set document specific preferences. */
  System.out.println("Creation date: (12-10-00 13:46:17)\n" + committed + newCommitted);
  //public void setSettings(java.util.Properties newSettings)
  {
    if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HEIGHT() )!= null )
    {
      getDocumentHeightField().setText( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HEIGHT() ) );
    }
    else
    {
      getDocumentHeightField().setText( String.valueOf( getDOCUMENT_HEIGHT_DEFAULT_VALUE() ) );
    }
    if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_WIDTH() )!= null )
    {
      getDocumentWidthField().setText( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_WIDTH() ) );
    }
    else
    {
      getDocumentWidthField().setText( String.valueOf( getDOCUMENT_WIDTH_DEFAULT_VALUE() ) );
    }
    if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING() )!= null )
    {
      getHorizontalSpacingField().setText( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING() ) );
    }
    else
    {
      getHorizontalSpacingField().setText( String.valueOf( getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE() ) );
    }
    if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING() )!= null )
    {
      getVerticalSpacingField().setText( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING() ) );
    }
    else
    {
      getVerticalSpacingField().setText( String.valueOf( getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE() ) );
    }
  }
}
*/

class GridPanoramaLayoutManager
{...
}`
getVerticalSpacingField().setText( String.valueOf(
    getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE() ));

} }

/**
 * Insert the method’s description here.
 * Creation date: (12-10-00 15:37:30)
 * @param owner javax.swing.JDialog
 */
public void show(javax.swing.JDialog owner)
{
    getGridLayoutCustomizationBox().setLocationRelativeTo( owner );
    getGridLayoutCustomizationBox().show();
}

/**
 * Take a properties table and update or insert (if non-existent) the <key,value> pairs defining the
 * configuration of this layout manager
 * Creation date: (12-10-00 14:49:56)
 * @return java.util.Properties
 * @param currentSettings java.util.Properties
 */
public java.util.Properties updateSettings(java.util.Properties currentSettings)
{
    currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_HEIGHT(), getDocumentHeightField().getText() );
    currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_WIDTH(), getDocumentWidthField().getText() );
    currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING(), getHorizontalSpacingField().getText() );
    currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING(), getVerticalSpacingField().getText() );
    return currentSettings;
}

/**
 * Validate the contents of the document preferences configuration box.
 * Creation date: (06-09-00 13:55:19)
 * @return boolean
 */
public boolean validateFields()
{
    peerviewmisc.MessageBox messageBox = new peerviewmisc.MessageBox();
    boolean valid = true;
    if ( ( Integer.parseInt( getDocumentHeightField().getText() ) > ClientConstants.
        getMAX_GRID_LAYOUT_DOCUMENT_HEIGHT() )
        || ( Integer.parseInt( getDocumentHeightField().getText() ) < ClientConstants.
        getMIN_GRID_LAYOUT DOCUMENT_HEIGHT() ) )
    {
        messageBox.setText( ClientConstants.getDocument_SIZE_OUT_OF_BOUNDS() );
        valid = false;
    }
    else if ( ( Integer.parseInt( getDocumentWidthField().getText() ) > ClientConstants.
        getMAX_GRID_LAYOUTDOCUMENT_WIDTH() )
        || ( Integer.parseInt( getDocumentWidthField().getText() ) < ClientConstants.
        getMIN_GRID_LAYOUT DOCUMENT_WIDTH() ) )
    {
        messageBox.setText( ClientConstants.getDocument_SIZE_OUT_OF_BOUNDS() );
        valid = false;
    }
    else if ( ( Integer.parseInt( getHorizontalSpacingField().getText() ) > ClientConstants.
        getMAX_GRID_LAYOUT_HORIZONTAL_SPACING() )
        || ( Integer.parseInt( getHorizontalSpacingField().getText() ) < ClientConstants.
        getMIN_GRID_LAYOUT_HORIZONTAL_SPACING() ) )
    {
        messageBox.setText( ClientConstants.getGRID_LAYOUT_SPACING_OUT_OF_BOUNDS() );
        valid = false;
    }
    return valid;
}
else if( ( Integer.parseInt( getVerticalSpacingField().getText() ) > ClientConstants.
getMAX_GRID_LAYOUT_VERTICAL_SPACING()) || ( Integer.parseInt( getVerticalSpacingField().getText() ) < ClientConstants.
getMIN_GRID_LAYOUT_VERTICAL_SPACING()) )
{
messageBox.setText( ClientConstants.getGRID_LAYOUT_SPACING_OUT_OF_BOUNDS() );
valid = false;
}
if( valid == false )
{
messageBox.show();
}
return valid;
}

Listing C.13: GroupDirectory.java

package clientapp;

/**
 * The client group directory where the listing of available along with their properties such as number
 * of participants and data volume are displayed.
 * Creation date: (30-06-00 20:37:33)
 * @author
 */
import peerviewmisc.*;
import java.awt.event.*;
import com.sun.media.jedt.*;
import java.util.Collection;
import java.util.Iterator;
import java.util.Vector;
import java.util Enumeration;
public class GroupDirectory extends javax.swing.JDialog {

    public class GroupDirectoryWindowListener extends WindowAdapter {
        public void windowActivated( WindowEvent we )
        {
            initWindow();
        }
    }
    public class CreateGroupButtonHandler implements ActionListener {
        public void actionPerformed( ActionEvent ae )
        {
            createNewGroup();
        }
    }
    public class EditGroupButtonHandler implements ActionListener {
        public void actionPerformed( ActionEvent ae )
        {
            // System.out.println( "ramte editexistinggroup" );
            editExistingGroup();
        }
    }
    public class DeleteGroupButtonHandler implements ActionListener {
        public void actionPerformed( ActionEvent ae )
        {
            // System.out.println( "ramte deletegroupbuttonhandler" );
        }
    }

    public class GroupDirectory {

}
deleteGroup();
}
}

public class JoinGroupButtonHandler implements ActionListener
{
public void actionPerformed(ActionEvent ae)
{
    joinGroup();
}
}

public class CloseButtonHandler implements ActionListener
{
public void actionPerformed(ActionEvent ae)
{
    dispose();
}
}

private javax.swing.JButton ivjCloseButton = null;
private javax.swing.JButton ivjCreateGroupButton = null;
private javax.swing.JButton ivjDeleteGroupButton = null;
private javax.swing.JPanel ivjDialogContentPane = null;
private javax.swing.JButton ivjJoinGroupButton = null;
private javax.swing.JPanel ivjPanel12 = null;
private java.awt.FlowLayout ivjPanel2FlowLayout = null;
private javax.swing.JPanel ivjPanel13 = null;
private java.awt.FlowLayout ivjPanel3FlowLayout = null;
private javax.swing.JScrollPane ivjScrollPane1 = null;
private JTable ivjScrollPaneTable = null;
private DeleteGroup ivjDeleteGroup1 = null;
private NewGroup ivjNewGroup1 = null;
private peerviewmisc.WorkGroup workGroup = null;
private EditGroup ivjEditGroup1 = null;
private javax.swing.JButton ivjEditGroupButton = null;
private javax.swing.JScrollPane ivjScrollPaneTable = null;
private javax.swing.JScrollPane ivjScrollPanePanel = null;
private peerviewmisc.WorkGroup[] groupDirectoryValues = null;
IvjEventHandler ivjEventHandler = new IvjEventHandler();
private ClientApplication clientApplication = null;

class IvjEventHandler implements java.awt.event.ActionListener {
public void actionPerformed(java.awt.event.ActionEvent e) {
    if (e.getSource() == GroupDirectory.this.getJoinGroupButton())
        connToM1(e);
    if (e.getSource() == GroupDirectory.this.getCreateGroupButton())
        connToM2(e);
    if (e.getSource() == GroupDirectory.this.getEditGroupButton())
        connToM3(e);
    if (e.getSource() == GroupDirectory.this.deleteGroupButton())
        connToM4(e);
    if (e.getSource() == GroupDirectory.this.getCloseButton())
        connToM5(e);
}
/*
 * GroupDirectory constructor comment.
 */
public GroupDirectory() {
    super();
    initialize();
} /*
 * GroupDirectory constructor comment.
 */
@param owner java.awt.Dialog
```java
public GroupDirectory(java.awt.Dialog owner) {
    super(owner);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public GroupDirectory(java.awt.Dialog owner, String title) {
    super(owner, title);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public GroupDirectory(java.awt.Dialog owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Frame
 */
public GroupDirectory(java.awt.Frame owner) {
    super(owner);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public GroupDirectory(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public GroupDirectory(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Frame
 */
public GroupDirectory(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Frame
 */
public GroupDirectory(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}
```
private void connEtoM1(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin (1)
        // user code end
        this.joinGroup();
        // user code begin (2)
        // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (3)
            // user code end
            handleException(ivjExc);
        }
    }

    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    /*
     * @param arg1 java.awt.event.ActionEvent
     */
    private void connEtoM2(java.awt.event.ActionEvent arg1) {
        try {
            // user code begin (1)
            // user code end
            this.createNewGroup();
            // user code begin (2)
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (3)
                // user code end
                handleException(ivjExc);
            }
        }

        /* WARNING: THIS METHOD WILL BE REGENERATED. */
        /*
         * @param arg1 java.awt.event.ActionEvent
         */
        private void connEtoM3(java.awt.event.ActionEvent arg1) {
            try {
                // user code begin (1)
                // System.out.println("FYR");
                // user code end
                this.editExistingGroup();
                // user code begin (2)
                // user code end
                } catch (java.lang.Throwable ivjExc) {
                    // user code begin (3)
                    // user code end
                    handleException(ivjExc);
                }
            }

            /* WARNING: THIS METHOD WILL BE REGENERATED. */
            /*
             * @param arg1 java.awt.event.ActionEvent
             */
            private void connEtoM4(java.awt.event.ActionEvent arg1) {
                try {
                    // user code begin (1)
                    // user code end
                    this.deleteGroup();
                    // user code begin (2)
                    // user code end
                }
            }
} catch (java.lang.Throwable ivjExc) {
    // user code begin {3}
    // user code end
246    handleException(ivjExc);
    
248 }
/**
250 * @param arg1 java.awt.event.ActionEvent
252 */
254 private void connEtoN5(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin {1}
        // user code end
258        this.dispose();
        // user code begin {2}
260        // user code end
262    } catch (java.lang.Throwable ivjExc) {
264        // user code begin {3}
        // user code end
266        handleException(ivjExc);
268    }
270 */
272 public void createNewGroup()
274 {
276    NewGroup newGroup = getNewGroup1();
278    newGroup.setWorkGroup( new WorkGroup() );
280    newGroup.show();
282    WorkGroup workGroup = newGroup.getWorkGroup();
284    // if the workgroup box wasn't cancelled
286    if ( newGroup.isIsDiscarded() == false )
288    {
290     // complete the non-editable fields of the new workgroup
292     workGroup.setCreationDate( new java.util.Date() );
294     workGroup.setCreator( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
296     getAuthor_Name() ) );
298     getClientApplication().getClientManager().createWorkGroup( workGroup );
300 }
302 */
304 public void deleteGroup()
306 {
308    int selectedRow = getScrollPaneTable().getSelectionModel().getMinSelectionIndex();
310    // if no selection
312    if ( selectedRow == -1 )
314    {
316       MessageBox messagebox = new MessageBox();
318       messagebox.setText( ClientConstants.getNo_Group_Selected_To_Delete() );
320       messagebox.show();
322   }
324   else
326   { WorkGroup selectedGroup = (WorkGroup) groupDirectoryValues[ selectedRow ];
328   // if the selected group has active participants
330 }
if ( selectedGroup.getParticipants().size() > 0 )
{
    MessageBox messagebox = new MessageBox();
    messagebox.setText( ClientConstants.getGROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS() );
    messagebox.show();
    return;
}

ConfirmationBox confirmationBox = new ConfirmationBox();
confirmationBox.setText( ClientConstants.getCONFIRM_WORKGROUP_DELETION( selectedGroup.getGroupName() ) );
confirmationBox.show();
if ( confirmationBox.isConfirmed() == true )
{
    getClientApplication().getClientManager().removeWorkGroup( selectedGroup );
}

/**
 * Disable all buttons that trigger actions involving communication with the server. Used in case of an
 * inactive server
 * or a broken communications channel.
 * Creation date: (10-08-00 16:13:10)
 */
private void disableButtons()
{
    getJoinGroupButton().setEnabled( false );
    getCreateGroupButton().setEnabled( false );
    getEditGroupButton().setEnabled( false );
    getDeleteGroupButton().setEnabled( false );
}

/**
 * Display an edit box containing the attributes of the group currently selected in the group directory.
 * Creation date: (03-07-00 21:44:52)
 */
public void editExistingGroup()
{
    int selectedRow = getScrollPaneTable().getSelectionModel().getMinSelectionIndex();
    // if no selection
    if ( selectedRow == -1 )
    {
        MessageBox messagebox = new MessageBox();
        messagebox.setText( ClientConstants.getNO_GROUP_SELECTED_TO_EDIT_MESSAGE() );
        messagebox.show();
    }
    else
    {
        WorkGroup backup = new WorkGroup( (WorkGroup) groupDirectoryValues[ selectedRow ] );
    }
    // if the selected group has active participants
    if ( backup.getParticipants().size() > 0 )
    {
        MessageBox messagebox = new MessageBox();
        messagebox.setText( ClientConstants.getGROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS() );
        messagebox.show();
        return;
    }
    getEditGroup1().setWorkGroup( (WorkGroup) groupDirectoryValues[ selectedRow ] );
    getEditGroup1().show();
    if ( getEditGroup1().isIsDiscarded() == false )
    {
        getClientApplication().getClientManager().removeWorkGroup( backup );
        WorkGroup newWorkGroup = getEditGroup1().getWorkGroup();
        newWorkGroup.setCreationDate( new java.util.Date() );
        newWorkGroup.setCreator( ClientInfo.getPreferences().getProperty( ClientConstants.getAUTHOR_NAME() ) );
    }

    307
// the below automatically causes the server to broadcast a group directory update to all
// clients, thus
   // eliminating the need to update the locally cached copy in the clientInfo object.
   getApplication().getClientManager().createGroup( newGroup );

/**
* Insert the method's description here.
* Creation date: (10-08-00 16:13:27)
*/
private void enableButtons()
{
   // Enable the buttons.
   getJoinGroupButton().setEnabled( true );
   getCreateGroupButton().setEnabled( true );
   getEditGroupButton().setEnabled( true );
   getDeleteGroupButton().setEnabled( true );

   // Enable the jPanel1 property value.
   /*
   * @return javax.swing.JPanel
   */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getButtonPanel() {
      if (ivjButtonPanel == null) {
         try {
            ivjButtonPanel = new javax.swing.JPanel();
            ivjButtonPanel.setName("ButtonPanel");
            ivjButtonPanel.setLayout(new java.awt.BorderLayout());
            getButtonPanel().add(getJPanel3(), "East");
            getButtonPanel().add(getJPanel2(), "West");

            // user code begin {1}
            getButtonPanel().setBorder( javax.swing.BorderFactory.createMatteBorder(
               ClientConstants.getBorder_WIDTH(),
               ClientConstants.getBorder_WIDTH(),
               ClientConstants.getBorder_WIDTH(),
               ClientConstants.getBorder_WIDTH(),
               ClientConstants.getBorder_WIDTH(),
            )
            getButtonPanel().setBackground( ) );

            // user code end
         }
         catch (java.lang.Throwable ivjEx) {
            // user code begin {2}
            // user code end
            handleException(ivjEx);
        }

        return ivjButtonPanel;
    }

    /**
    * Insert the method's description here.
    * Creation date: (10-08-00 15:41:17)
    * @return clientapp.ClientApplication
    */
    public ClientApplication getClientApplication() {
        return clientApplication;
    }

    /**
     * Return the CloseButton property value.
     * @return javax.swing.JButton
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getCloseButton() {
        if (ivjCloseButton == null) {
            try {
                ivjCloseButton = new javax.swing.JButton();
                ivjCloseButton.setName("CloseButton");
            }
            catch (java.lang.Throwable ivjEx) {
                // user code begin {3}
                // user code end
                handleException(ivjEx);
            }
        }
        return ivjCloseButton;
    }

// clientapp.ClientApplication
ivjCloseButton.setMnemonic('c');
ivjCloseButton.setText("Close");
// user code begin (1)
ivjCloseButton.addActionListener( new CloseButtonHandler() );
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
return ivjCloseButton;
/**
 * Return the CreateGroupButton property value.
 * @return javax.swing.JButton
 */
/*
 * WARNING: THIS METHOD WILL BE REGENERATED.
 */
private javax.swing.JButton getCreateGroupButton() {
if (ivjCreateGroupButton == null) {
try {
ivjCreateGroupButton = new javax.swing.JButton();
ivjCreateGroupButton.setName("CreateGroupButton");
ivjCreateGroupButton.setMnemonic('g');
ivjCreateGroupButton.setText("Create Group");
// user code begin (1)
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
return ivjCreateGroupButton;
}/**
 * Return the DeleteGroup1 property value.
 * @return peerviewmisc.DeleteGroup
 */
/*
 * WARNING: THIS METHOD WILL BE REGENERATED.
 */
public peerviewmisc.DeleteGroup getDeleteGroup1() {
if (ivjDeleteGroup1 == null) {
try {
ivjDeleteGroup1 = new peerviewmisc.DeleteGroup();
ivjDeleteGroup1.setName("DeleteGroup1");
// user code begin (1)
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
return ivjDeleteGroup1;
}/**
 * Return the DeleteGroupButton property value.
 * @return javax.swing.JButton
 */
/*
 * WARNING: THIS METHOD WILL BE REGENERATED.
 */
private javax.swing.JButton getDeleteGroupButton() {
if (ivjDeleteGroupButton == null) {
try {
ivjDeleteGroupButton = new javax.swing.JButton();
ivjDeleteGroupButton.setName("DeleteGroupButton");
}
ivjDeleteGroupButton.setMnemonic('D');
ivjDeleteGroupButton.setText("Delete group");
// user code begin (1)
//ivjDeleteGroupButton.addActionListener( new DeleteGroupButtonHandler() );
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
return ivjDeleteGroupButton;
/**
 * Return the EditGroup1 property value.
 * @return peerviewmisc.EditGroup1
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public peerviewmisc.EditGroup1 getEditGroup1() {
if (ivjEditGroup1 == null) {
try {
    ivjEditGroup1 = new peerviewmisc.EditGroup1();
    ivjEditGroup1.setName("EditGroup1");
    ivjEditGroup1.setDefaultCloseOperation( javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    // user code begin (1)
    ivjEditGroup1.setGroupDirectory( this );
    // user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
    }
    }
return ivjEditGroup1;
/**
 * Return the EditGroupButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public javax.swing.JButton getEditGroupButton() {
if (ivjEditGroupButton == null) {
try {
    ivjEditGroupButton = new javax.swing.JButton();
    ivjEditGroupButton.setName("EditGroupButton");
    ivjEditGroupButton.setMnemonic('E');
    ivjEditGroupButton.setText("Edit Group");
    // user code begin (1)
    //ivjEditGroupButton.addActionListener( new EditGroupButtonHandler() );
    // user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
    }
    return ivjEditGroupButton;
}
/**
 * Insert the method's description here.
 */
/* Creation date: (26-07-00 21:09:59)
 * @return peerviewmisc.WorkGroup[]
 */
public peerviewmisc.WorkGroup[] getGroupDirectoryValues() {
return groupDirectoryValues;
}
```java
/**
 * Return the JDialogContentPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJDialogContentPane() {
    if (ivjJDialogContentPane == null) {
        try {
            ivjJDialogContentPane = new javax.swing.JPanel();
            ivjJDialogContentPane.setName("JDialogContentPane");
            ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
            getJDialogContentPane().add(getTablePanel(), "Center");
            getJDialogContentPane().add(getButtonPanel(), "South");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJDialogContentPane;
}

/**
 * Return the JoinGroupButton property value.
 * @return javax.swing.JButton
 */
private javax.swing.JButton getJoinGroupButton() {
    if (ivjJoinGroupButton == null) {
        try {
            ivjJoinGroupButton = new javax.swing.JButton();
            ivjJoinGroupButton.setName("JoinGroupButton");
            ivjJoinGroupButton.setMnemonic('J');
            ivjJoinGroupButton.setText("Join group");
            // user code begin (1)
            //ivjJoinGroupButton.addActionListener(new JoinGroupButtonHandler());
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJoinGroupButton;
}

/**
 * Return the JPanel2 property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getPanel2() {
    if (ivjPanel2 == null) {
        try {
            ivjPanel2 = new javax.swing.JPanel();
            ivjPanel2.setName("JPanel2");
            ivjPanel2.setLayout(new javax.swing.GroupLayout.SequentialLayout());
            getContentPane().add(getJoinGroupButton(), getContentPane().getName());
            getContentPane().add(getCreateGroupButton(), getContentPane().getName());
            getContentPane().add(getDeleteGroupButton());
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
        }
    }
    return ivjPanel2;
}
```

311
handleException(ivjExc);
}
}
/**
 * Return the JPanel2FlowLayout property value.
 * @return java.awt.FlowLayout
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.FlowLayout getJPanel2FlowLayout() {
    java.awt.FlowLayout ivjPanel2FlowLayout = null;
    try {
        /* Create part */
        ivjPanel2FlowLayout = new java.awt.FlowLayout();
        ivjPanel2FlowLayout.setAlignment(java.awt.FlowLayout.LEFT);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    return ivjPanel2FlowLayout;
}
/**
 * Return the JPanel3 property value.
 * @return javax.swing.JPanel
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJPanel3() {
    if (ivjPanel3 == null) {
        try {
            ivjPanel3 = new javax.swing.JPanel();
            ivjPanel3.setName("JPanel3");
            ivjPanel3.setLayout(getJPanel3FlowLayout());
            getJPanel3().add(getCloseButton(), getCloseButton().getName());
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjPanel3;
}
/**
 * Return the JPanel3FlowLayout property value.
 * @return java.awt.FlowLayout
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.FlowLayout getJPanel3FlowLayout() {
    java.awt.FlowLayout ivjPanel3FlowLayout = null;
    try {
        /* Create part */
        ivjPanel3FlowLayout = new java.awt.FlowLayout();
        ivjPanel3FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    return ivjPanel3FlowLayout;
}
/**
 * Return the JScrollPane property value.
 * @return javax.swing.JScrollPane
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JScrollPane getJScrollPane1() {
    if (ivjJScrollPane1 == null) {

try {
    ivjJScrollPane = new javax.swing.JScrollPane();
    ivjJScrollPane.setName("JScrollPane");
    ivjJScrollPane.setVerticalScrollBarPolicy(javax.swing.JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    ivjJScrollPane.setHorizontalScrollBarPolicy(javax.swing.JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
    ivjJScrollPane.setDoubleBuffered(true);
    getJScrollPane().setViewportView(getScrollPaneTable());
    // user code begin {1}
    // user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}

return ivjJScrollPane;

/**
 * Return the NewGroup1 property value.
 * @return peerviewmisc.NewGroup
 */

/**< WARNING: THIS METHOD WILL BE REGENERATED. */
public peerviewmisc.NewGroup getNewGroup1() {
    if (ivjNewGroup1 == null) {
        try {
            ivjNewGroup1 = new peerviewmisc.NewGroup();
            ivjNewGroup1.setName("NewGroup1");
            // user code begin {1}
            ivjNewGroup1.setGroupDirectory( this );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }

    return ivjNewGroup1;
}

/**< WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTable getScrollPaneTable() {
    if (ivjScrollPaneTable == null) {
        try {
            ivjScrollPaneTable = new javax.swing.JTable();
            ivjScrollPaneTable.setName("ScrollPaneTable");
            getJScrollPane().setColumnHeaderView(ivjScrollPaneTable.getTableHeader());
            getJScrollPane().getViewPort().setBackingStoreEnabled(true);
            ivjScrollPaneTable.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_OFF);
            ivjScrollPaneTable.setPreferredSize(new java.awt.Dimension(716,412));
            ivjScrollPaneTable.setBounds(0, 0, 716, 412);
            ivjScrollPaneTable.setPreferredScrollableViewportSize(new java.awt.Dimension(700, 400));
            ivjScrollPaneTable.setAutoCreateColumnsFromModel(true);
            // user code begin {1}
            ivjScrollPaneTable.setSelectionMode( javax.swing.ListSelectionModel.SINGLE_SELECTION );
            ivjScrollPaneTable.setAutoResizeMode( javax.swing.JTable.AUTO_RESIZE_OFF );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }

    return ivjScrollPaneTable;
}
return ivjScrollPaneTable;

/**
 * Insert the method's description here.
 * Creation date: (04-07-00 19:38:32)
 * @return javax.swing.table.DefaultTableModel
 */
public javax.swing.table.DefaultTableModel getTableModel()
{
    return tableModel;
}
/**
 * Return the JPanel4 property value.
 * @return javax.swing.JPanel
 */

/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getTablePanel()
{
    try {
        ivjTablePanel = new javax.swing.JPanel();
        ivjTablePanel.setName("TablePanel");
        ivjTablePanel.setLayout(new java.awt.BorderLayout());
        getTablePanel().add(getJScrollPane(), "Center");
        // user code begin if
        getTablePanel().setBorder(ClientConstants.createDialogBorder( getTablePanel().getBackground() )
        );
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
    }
    return ivjTablePanel;
}
/**
 * Insert the method's description here.
 * Creation date: (03-07-00 21:00:14)
 * @return peerviewmisc.WorkGroup
 */
public peerviewmisc.WorkGroup getWorkGroup()
{
    return workGroup;
}
/**
 * Traverses the group directory (list of active groups) to determine if any group matches the group
 * name and description parameters
 * Creation date: (25-07-00 16:09:36)
 * @return boolean
 * @param groupName java.lang.String
 * @param description java.lang.String
 */
public boolean groupExists(String groupName, String description)
{
    boolean exists = false;
    for ( int i = 0; i < groupDirectoryValues.length; i++ )
    {
        if ( ((WorkGroup) groupDirectoryValues[i]).getGroupName().equalsIgnoreCase( groupName ) &
                (WorkGroup) groupDirectoryValues[i].getDescription().equalsIgnoreCase( description ))
        {
            exists = true;
            break;
        }
    }
    return exists;
}
/**
 * Called whenever the part throws an exception.

* @param exception java.lang.Throwable
* 
* private void handleException(java.lang.Throwable exception) {
*     /* Uncomment the following lines to print uncaught exceptions to stdout */
*     // System.out.println("---------- UNCAUGHT EXCEPTION ----------");
*     // exception.printStackTrace(System.out);
* }
* } /*
* * Initializes connections
* * @exception java.lang.Exception The exception description.
* */
* private void initConnections() throws java.lang.Exception {
*     // user code begin {1}
*     // user code end
*     getContentPane().addActionListener(ivjEventEventHandler);  
*     getCreateGroupButton().addActionListener(ivjEventEventHandler); 
*     getEditGroupButton().addActionListener(ivjEventEventHandler); 
*     getDeleteGroupButton().addActionListener(ivjEventEventHandler); 
*     getCloseButton().addActionListener(ivjEventEventHandler); 
* }
* } /*
* * Initialize the class.
* */
* private void initialize() {
*     try {
*         // user code begin {1}
*         // user code end
*     } catch (java.lang.Throwable ivjExc) {
*         handleException(ivjExc);
*     } /*
* * Insert the method's description here.
* * Creation date: (04-07-00 23:11:34)
* */
* public void initTable() {
*     WorkGroup workGroup = new WorkGroup();
*     tableModel = new javax.swing.table.DefaultTableModel();
*     ivjScrollPane.setModel( tableModel );
*     String columnNames[] = WorkGroup.getFieldNames();
*     for ( int i = 0 ; i < columnNames.length ; i++ )
*     {
*         tableModel.addColumn( columnNames[i] );
*     }
* } /*
* * Insert the method's description here.
* * Creation date: (16-08-00 18:15:00)
* */
*
public void initWindow()
{
    
    /**
     * Insert the method’s description here.
     */
    
    // Creation date: (06-07-00 12:48:00)
    
    public void joinGroup()
    {
        int selectedIndex = getScrollPane().getSelectedRow();
        // if no row is selected
        if (selectedIndex == -1)
        {
            MessageBox messageBox = new MessageBox();
            messageBox.setText(ClientConstants.getNO_GROUP_SELECTED_TO_JOIN_MESSAGE());
            messageBox.show();
            return;
        }
        WorkGroup workGroup = (WorkGroup)gdirectoryValues[selectedIndex];
        URLString url = workGroup.getSessionURL();
        Session session;
        Channel channel;
        String clientNames[] = null;
        try
        {
            session = SessionFactory.createSession(getClientApplication().getClientManager().
               getGroupManagementClient(), url, false);
            channel = session.createChannel(getClientApplication().getClientManager().
               getGroupManagementClient(),
               workGroup.getChannelID(),
               true, true, false);
            clientNames = channel.listClientNames();
        }
        // the below is a remnant from an attempt to resolve the problem using a different method than a for
        // loop cycling
        // through the member names, comparing each to the unique name of this client.
        catch (NameInUseException NIVE)
        {
            MessageBox messageBox = new MessageBox();
            messageBox.setText(ClientConstants.getALREADY_JOINED_TO_GROUP_MESSAGE());
            messageBox.show();
            return;
        }
        catch (JSDException JSDTE)
        {
            getClientApplication().getClientManager().handleJSDException(JSDTE);
        }
        for (int i = 0; i < clientNames.length; i++)
        {
            if (clientNames[i].equalsIgnoreCase(ClientInfo.getPreferences().getProperty(ClientConstants.
               getNAME())))
            {
                MessageBox messageBox = new MessageBox();
                messageBox.setText(ClientConstants.getALREADY_JOINED_TO_GROUP_MESSAGE());
                messageBox.show();
                return;
            }
        }
    }
    }
*/
WorkGroup activeGroup = getClientApplication().getClientManager().getClientInfo().getActiveGroup();
getClientApplication().getClientManager().disconnectFromGroup( activeGroup );
getClientApplication().getClientManager().connectToGroup( workGroup );
*/
/**
 * main entrypoint - starts the part when it is run as an application
 */
public static void main(java.lang.String[] args) {
    try {
        GroupDirectory aGroupDirectory;
aGroupDirectory = new GroupDirectory();
aGroupDirectory.setModal(true);
aGroupDirectory.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
                System.exit(0);
            }
        });
aGroupDirectory.setVisible(true);
    } catch (Throwable exception) {
        System.err.println("Exception occurred in main()");
        exception.printStackTrace(System.out);
    }
}
/**
 * Insert the method's description here.
 */
public void setClientApplication(ClientApplication newClientApplication) {
    clientApplication = newClientApplication;
}
/**
 * Insert the method's description here.
 */
public void setGroupDirectoryValues(peerviewmisc.WorkGroup [] newGroupDirectoryValues) {
    groupDirectoryValues = newGroupDirectoryValues;
}
/**
 * Insert the method's description here.
 */
public void setTableModel(java.awt.swing.table.DefaultTableModel newTableModel) {
    tableModel = newTableModel;
}
/**
 * Insert the method's description here.
 */
public void setWorkGroup(peerviewmisc.WorkGroup newWorkGroup) {
    workGroup = newWorkGroup;
}
/**
 * Update the table model underlying the group directory.
 */
public void updateTable() {
    if ( getClientApplication().getClientManager().getClientInfo().getGroups() == null )
    {
}
Listing C.14: MessagePropertiesBox.java

class MessagePropertiesBox extends javax.swing.JDialog {
public class CloseButtonHandler implements ActionListener {
    public void actionPerformed(java.awt.event.ActionEvent ae) {
        dispose();
    }
}
private javax.swing.JPanel ivjDialogContentPane = null;
private javax.swing.JScrollPane ivjScrollPane = null;
private javax.swing.JTable ivjScrollPaneTable = null;
private javax.swing.JButton ivjCloseButton = null;
private javax.swing.table.DefaultTableModel tableModel = new DefaultTableModel();
private javax.swing.JPanel ivjButtonPanel = null;
/**
 * MessagePropertiesBox constructor comment.
 */
public MessagePropertiesBox() {
    super();
    initialize();
}
/*
 * MessagePropertiesBox constructor comment.
 */

public MessagePropertiesBox(java.awt.Dialog owner) {
  super(owner);
}

/*
 * MessagePropertiesBox constructor comment.
 * @param owner java.awt.Dialog
 */

public MessagePropertiesBox(java.awt.Dialog owner, String title) {
  super(owner, title);
}

/*
 * MessagePropertiesBox constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */

public MessagePropertiesBox(java.awt.Dialog owner, String title, boolean modal) {
  super(owner, title, modal);
}

/*
 * MessagePropertiesBox constructor comment.
 */

public MessagePropertiesBox(java.awt.Frame owner) {
  super(owner);
}

/*
 * MessagePropertiesBox constructor comment.
 * @param owner java.awt.Frame
 */

public MessagePropertiesBox(java.awt.Frame owner, String title) {
  super(owner, title);
}

/*
 * MessagePropertiesBox constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */

public MessagePropertiesBox(java.awt.Frame owner, String title, boolean modal) {
  super(owner, title, modal);
}

/*
 * MessagePropertiesBox constructor comment.
 */

public MessagePropertiesBox(java.awt.Frame owner, boolean modal) {
  super(owner, modal);
}

/*
 * MessagePropertiesBox constructor comment.
 */

public MessagePropertiesBox(java.awt.Frame owner, boolean modal) {
  super(owner, modal);
}

/*
 * Return the JPanel1 property value.
 */
private javax.swing.JButton getCloseButton() {
    try {
        ivjCloseButton = new javax.swing.JButton();
        ivjCloseButton.setName("closeButton");
        ivjCloseButton.setMnemonic('c');
        ivjCloseButton.setText("close");
        // user code begin (1)
        ivjCloseButton.addActionListener(new CloseButtonHandler());
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
}

private javax.swing.JTextField getMacCaptionTextField() {
    try {
        ivjMacCaptionTextField = new javax.swing.JTextField();
        ivjMacCaptionTextField.setName("macCaptionTextField");
        ivjMacCaptionTextField.setComponentOrientation(java.awt.ComponentOrientation.RIGHT);
        // user code begin
        // user code end
        return ivjMacCaptionTextField;
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
}

private javax.swing.JButton getOkButton() {
    try {
        ivjOkButton = new javax.swing.JButton();
        ivjOkButton.setName("okButton");
        ivjOkButton.addActionListener(new OkButtonHandler());
        // user code begin (1)
        ivjOkButton.addActionListener(new OkButtonHandler());
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
}

private javax.swing.JTextField getMacMacCaptionTextField() {
    try {
        ivjMacMacCaptionTextField = new javax.swing.JTextField();
        ivjMacMacCaptionTextField.setName("macMacCaptionTextField");
        ivjMacMacCaptionTextField.setComponentOrientation(java.awt.ComponentOrientation.RIGHT);
        // user code begin
        // user code end
        return ivjMacMacCaptionTextField;
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
}
try {
    ivjDialogContentPane = new javax.swing.JPanel();
    ivjDialogContentPane.setName("DialogContentPane");
    ivjDialogContentPane.setLayout(new java.awt.BorderLayout());
    getDialogContentPane().add(getButtonPanel(), "South");
    getDialogContentPane().add(getScrollPane(), "Center");
    // user code begin {1}
    // user code end
}
// callback
}
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}
}
/**
 * Return the JScrollPane property value.
 */
private javax.swing.JScrollPane get JScrollPane() {
if (ivj JScrollPane == null) {
    try {
        ivj JScrollPane = new javax.swing.JScrollPane();
        ivj JScrollPane.setName(" JScrollPane1");
        ivj JScrollPane.set Autoscroll (true);
        ivj JScrollPane.setVerticalScrollBarPolicy (javax.swing.JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
        ivj JScrollPane.setHorizontalScrollBarPolicy (javax.swing.JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        getScrollPane().setViewportView (getScrollPaneTable());
        // user code begin {1}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
    }
}
    // user code begin {1}
    // user code end
}
/**
 * Return the JScrollPaneTable property value.
 */
private javax.swing.JTable getScrollPaneTable() {
if (ivj JScrollPaneTable == null) {
    try {
        ivj JScrollPaneTable = new javax.swing.JTable();
        ivj JScrollPaneTable.setName(" JScrollPaneTable");
        getScrollPane().setColumnHeaderView (ivj JScrollPaneTable.getColumnModel().getTableHeader());
        getScrollPane().setViewport().setBackingStoreEnabled (true);
        ivj JScrollPaneTable.setBackground (java.awt.Color.white);
        ivj JScrollPaneTable.setForeground (java.awt.Color.black);
        ivj JScrollPaneTable.setForeground (java.awt.SystemColor.tableCellEditorColor); //default
        ivj JScrollPaneTable.setPassword (new java.awt.Font("dialog", 0, 12));
        ivj JScrollPaneTable.setIntercellSpacing (new java.awt.Dimension(5, 5));
        ivj JScrollPaneTable.set Bounds (0, 0, 200, 200);
        // user code begin {1}
        ivj JScrollPaneTable.setModel (tableModel);
        ivj JScrollPaneTable.setSelectionMode (javax.swing.ListSelectionModel.SINGLE_SELECTION);
        ivj JScrollPaneTable.setAutoResizeMode (javax.swing.JTable.AUTO_RESIZE_OFF);
        // user code end
    } catch (java.lang.Throwable ivjExc) {
// user code begin {2}
// user code end
handleException(ivjExc);
}
return ivjScrollPaneTable;
}/**
 * Insert the method’s description here.
 */
public javax.swing.table.DefaultTableModel getTableModel(){
   return tableModel;
}
/**
 * Called whenever the part throws an exception.
 */
private void handleException(java.lang.Throwable exception){
   /* Uncomment the following lines to print uncaught exceptions to stdout */
   // System.out.println("---------- UNCAUGHT EXCEPTION ----------");
   // exception.printStackTrace(System.out);
   */
   /* Initialize the class.
*/
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
   try{
      // user code begin {1}
      // user code end
      setName("MessagePropertiesBox");
      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
      setSize(415, 257);
      setTitle("Message.properties");
      setDefaultCloseOperation(getDialogContent());
   } catch (java.lang.Throwable ivjExc) { 
      handleException(ivjExc);
   }
   // user code begin {2}
   // default positioning puts the dialog at center of screen. The method used is a slight hack, but works fine.
   setLocationRelativeTo( getOwner() );
   getScrollPaneTable().setAutoResizeMode( javax.swing.JTable.AUTO_RESIZE_OFF );
   // user code end
}
}" // user code end
/**
 * Insert the method’s description here.
 */
public void message(peerview.misc.Message message) {
   // The below ought to be replaceable with calls to addColumn, but an attempt to do so didn’t work as intended - perhaps a JDK bug?
   Object fieldNames[] = message.getFieldNames();
   Vector values = message.getDataAsVector();
   tableModel.addColumn( ClientConstants.getMessagePropertiesNameColumnHeading() );
   tableModel.addColumn( ClientConstants.getMessagePropertiesValueColumnHeading() );
   for ( int i = 0; i < fieldNames.length; i++ ) {
      Vector v = new Vector();
      v.addElement( fieldNames[i] );
   }
v.addElement( values.get(i) );
    tableModel.addRow( v );
}

getScrollPane().setColumnSelectionAllowed( false );
getScrollPane().setRowSelectionAllowed( false );
getScrollPane().setEnabled( false );
getScrollPane().setCellEditor( null );
getScrollPane().removeEditor();
}

/**
 * main entrypoint - starts the part when it is run as an application
 */
public static void main(java.lang.String[] args) {
    try {
        MessagePropertiesBox aMessagePropertiesBox;
        aMessagePropertiesBox = new MessagePropertiesBox();
        aMessagePropertiesBox.setModal(true);
        aMessagePropertiesBox.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
                System.exit(0);
            }
        });
        aMessagePropertiesBox.setVisible(true);
    } catch (Throwable exception) {
        System.err.println("Exception occurred in main() of MessageWindow");
        exception.printStackTrace(System.out);
    }
}

/**
 * Insert the method's description here.
 */

import java.awt.event.*;
import javax.swing.event.*;

package clientapp;

import java.awt.event.*;
import javax.swing.event.*;

/**
 * This class implements the message window that can be activated by double clicking in the message bar
 * at the bottom of the client application window.
 * Creation date: (16-08-00 16:44:45)
 */
public class MessageWindow extends javax.swing.JDialog {
    public class CloseItemHandler implements ActionListener {
        public void actionPerformed(ActionEvent ae ) {
            closeMessageWindow();
        }
    }
    public class SaveItemHandler implements ActionListener {
        public void actionPerformed(ActionEvent ae ) {
            // Add code here
        }
    }
    
    Listing C.15: MessageWindow.java
saveMessageWindowContents();

private javax.swing.JMenuItem ivjCloseItem = null;
private javax.swing.JMenu ivjFileMenu = null;
private javax.swing.JPanel ivjJDialogContentPane = null;
private javax.swing JScrollPane ivjScrollPane1 = null;
private javax.swing.JSeparator ivjJSeparator1 = null;
private javax.swing.JMenuBar ivjMessageWindowJMenuBar = null;
private javax.swing.JMenuItem ivjSaveItem = null;
private javax.swing.JTextArea ivjTextArea = null;
private ClientApplication clientApplication = null;
/**
 * MessageWindow constructor comment.
 */
public MessageWindow() {
    super();
    initialize();
}
/**
 * MessageWindow constructor comment.
 * @param owner java.awt.Dialog
 */
public MessageWindow(java.awt.Dialog owner) {
    super(owner);
}
/**
 * MessageWindow constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public MessageWindow(java.awt.Dialog owner, String title) {
    super(owner, title);
}
/**
 * MessageWindow constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public MessageWindow(java.awt.Dialog owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * MessageWindow constructor comment.
 * @param owner java.awt.Dialog
 */
public MessageWindow(java.awt.Dialog owner) {
    super(owner);
}
/**
 * MessageWindow constructor comment.
 * @param owner java.awt.Frame
 */
public MessageWindow(java.awt.Frame owner) {
    super(owner);
}
/**
 * MessageWindow constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public MessageWindow(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
 * MessageWindow constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public MessageWindow(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}

/**
 * MessageWindow constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public MessageWindow(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}

/**
 * Insert the method's description here.
 * @param string java.lang.String
 */
public void addString(String string) {
    getTextArea().append( string + '
' );
}

/**
 * Insert the method's description here.
 * @param string java.lang.String
 */
public void closeMessageWindow() {
    setVisible( false );
}

/**
 * Insert the method's description here.
 * @param string java.lang.String
 */
public ClientApplication getClientApplication() {
    return clientApplication;
}

/**
 * Return the CloseItem property value.
 * @return javax.swing.JMenuItem
 */
private JMenuItem getCloseItem() {
    if (ivjCloseItem == null) {
        try {
            ivjCloseItem = new javax.swing.JMenuItem();
            ivjCloseItem.setName("CloseItem");
            ivjCloseItem.setMnemonic('C');
            ivjCloseItem.setText("Close");
            // user code begin {1}
            ivjCloseItem.addActionListener( new CloseItemHandler() );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjCloseItem;
}

/**
* Return the FileMenu property value.
* @return javax.swing.JMenu

/**
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenu getFileMenu() {
    if (ivjFileMenu == null) {
        try {
            ivjFileMenu = new javax.swing.JMenu();
            ivjFileMenu.setName("FileMenu");
            ivjFileMenu.setMnemonic('F');
            ivjFileMenu.setText("File");
            ivjFileMenu.addItem(getSaveItem());
            ivjFileMenu.addItem(getNewItem());
            ivjFileMenu.addItem(getCloseItem());
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjFileMenu;
}

/**
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getDialogContentPane() {
    if (ivjDialogContentPane == null) {
        try {
            ivjDialogContentPane = new javax.swing.JPanel();
            ivjDialogContentPane.setName("DialogContentPane");
            ivjDialogContentPane.setLayout(new java.awt.BorderLayout());
            ivjDialogContentPane.add(getTextArea(), "Center");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjDialogContentPane;
}

/**
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JScrollPane getScrollPane() {
    if (ivjScrollPanel == null) {
        try {
            ivjScrollPanel = new javax.swing.JScrollPane();
            ivjScrollPanel.setName("JSscrollPanel");
            ivjScrollPanel.add(getTextArea());
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjScrollPanel;
}
```java
private javax.swing.JSeparator getJSeparator1() {
  if (ivjJSeparator1 == null) {
    try {
      ivjJSeparator1 = new javax.swing.JSeparator();
      ivjJSeparator1.setName("JSeparator1");
      // user code begin {1}
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin {2}
      // user code end
      handleException(ivjExc);
    }
  }
  return ivjJSeparator1;
}

private javax.swing.JMenuBar getMessageWindowJMenuBar() {
  if (ivjMessageWindowJMenuBar == null) {
    try {
      ivjMessageWindowJMenuBar = new javax.swing.JMenuBar();
      ivjMessageWindowJMenuBar.setName("MessageWindowJMenuBar");
      ivjMessageWindowJMenuBar.add(getFileMenu());
      // user code begin {1}
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin {2}
      // user code end
      handleException(ivjExc);
    }
  }
  return ivjMessageWindowJMenuBar;
}

private javax.swing.JMenuItem getSaveItem() {
  if (ivjSaveItem == null) {
    try {
      ivjSaveItem = new javax.swing.JMenuItem();
      ivjSaveItem.setName("SaveItem");
      ivjSaveItem.setMnemonic('S');
      ivjSaveItem.setText("Save");
      // user code begin {1}
      ivjSaveItem.addActionListener( new SaveItemHandler() );
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin {2}
      // user code end
      handleException(ivjExc);
    }
  }
  return ivjSaveItem;
```

* Return the TextArea property value.
* @return javax.swing.JTextArea

/**
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextArea getTextArea() {
    if (ivjTextArea == null) {
        try {
            ivjTextArea = new javax.swing.JTextArea();
            ivjTextArea.setName("TextArea");
            ivjTextArea.setToolTipText("Messages generated by the peerview client application.");
            ivjTextArea.setBounds(0, 0, 7, 6);
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjTextArea;
}

/**
* Called whenever the part throws an exception.
* @param exception java.lang.Throwable
*/
private void handleException(java.lang.Throwable exception) {
    // Uncomment the following lines to print uncaught exceptions to stdout */
    // System.out.println("-------- UNCAUGHT EXCEPTION --------");
    // exception.printStackTrace(System.out);
    // user code begin
    // user code end
    try {
        // user code begin {1}
        // user code end
        setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
        setTitle("MessageWindow");
        setSize(612, 479);
        setResizable(false);
        setMenuBar(getMessageWindowJMenuBar());
        getContentPane().add(getJDialogDialogContentPane());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    // user code begin {2}
    getTextArea().setEditable( false );
    getTextArea().setLineWrap( false );
    setDefaultCloseOperation( javax.swing.JDialog.HIDE_ON_CLOSE );
    setLocationRelativeTo( getContentPane() );
    // user code end
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    // user code begin
    // user code end
    } catch (java.lang.RuntimeException ivjExc) {
        handleException(ivjExc);
    }
}

/**
* main entrypoint - starts the part when it is run as an application
* @param args java.lang.String[]
*/
public static void main(java.lang.String[] args) {
    try {
        MessageWindow aMessageWindow;
        aMessageWindow = new MessageWindow();
        aMessageWindow.setVisible(true);
        aMessageWindow.addWindowListener(new java.awt.event.WindowAdapter() {
        };
    } catch (java.lang.RuntimeException ivjExc) {
        handleException(ivjExc);
    }
}}
package clientapp;

/**
 * Abstract superclass for client panorama layout managers.
 * @author
 * @copyright
 */
public abstract class PanoramaLayout {
    protected java.awt.Dimension preferredDocumentSize = new Dimension();
    protected int xSpacing;
    protected int ySpacing;
    protected java.awt.Dimension previousPlacement = null;
    protected int horizontalIndex = 0;
    protected int verticalIndex = 0;
    /**
     * PanoramaLayout constructor comment.
     * @param
     */
    public PanoramaLayout ( )
    {
    }
preferredDocumentSize = new Dimension( Integer.parseInt( (String) ClientInfo.getPreferences().getProperty(
        ClientConstants.getDEFAULT_DOCUMENT_WIDTH() ) ),
    Integer.parseInt( (String) ClientInfo.getPreferences().getProperty(
        clientConstants.getDEFAULTDOCUMENT_HEIGHT() ) ) );
xSpacing = Integer.parseInt( (String) ClientInfo.getPreferences().getProperty(
        ClientConstants.getDEFAULT_HORIZONTAL_DOCUMENT_SPACING() ) );
ySpacing = Integer.parseInt( (String) ClientInfo.getPreferences().getProperty(
        ClientConstants.getDEFAULT_VERTICAL_DOCUMENT_SPACING() ) );
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 11:50:28)
 * @return java.awt.Dimension
 */
public abstract Dimension computePlacement() throws Exception;
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 15:42:44)
 * @return int
 */
public int getHorizontalIndex() {
    return horizontalIndex;
}
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 19:42:20)
 * @return java.awt.Rectangle
 */
public abstract java.awt.Rectangle getOverviewRectangle();
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 11:32:36)
 * @return java.awt.Dimension
 */
public java.awt.Dimension getPreferredDocumentSize() {
    return preferredDocumentSize;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 12:02:54)
 * @return java.awt.Dimension
 */
public java.awt.Dimension getPreviousPlacement() {
    return previousPlacement;
}
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 15:42:53)
 * @return int
 */
public int getVerticalIndex() {
    return verticalIndex;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 11:37:06)
 * @return int
 */
public int getXSpacing() {
    return xSpacing;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 11:37:26)
 * @return int
 */
public int getVSpacing() {
    return ySpacing;
}
/**
 * Insert the method’s description here.
 * Creation date: (23-06-00 22:22:44)
 */
public abstract void resetLayout();
/**
 * Insert the method’s description here.
 * Creation date: (01-08-00 15:42:44)
 * @param newHorizontalIndex int
 */
public void setHorizontalIndex(int newHorizontalIndex) {
    horizontalIndex = newHorizontalIndex;
}
/**
 * Insert the method’s description here.
 * Creation date: (22-06-00 11:32:36)
 * @param newPreferredDocumentSize java.awt.Dimension
 */
public void setPreferredDocumentSize(java.awt.Dimension newPreferredDocumentSize) {
    preferredDocumentSize = newPreferredDocumentSize;
}
/**
 * Insert the method’s description here.
 * Creation date: (22-06-00 12:02:54)
 * @param newPreviousPlacement java.awt.Dimension
 */
public void setPreviousPlacement(java.awt.Dimension newPreviousPlacement) {
    previousPlacement = newPreviousPlacement;
}
/**
 * Insert the method’s description here.
 * Creation date: (01-08-00 15:42:53)
 * @param newVerticalIndex int
 */
public void setVerticalIndex(int newVerticalIndex) {
    verticalIndex = newVerticalIndex;
}
/**
 * Insert the method’s description here.
 * Creation date: (22-06-00 11:37:06)
 * @param newXSpacing int
 */
public void setXSpacing(int newXSpacing) {
    xSpacing = newXSpacing;
}
/**
 * Insert the method’s description here.
 * Creation date: (22-06-00 11:37:26)
 * @param newYSpacing int
 */
public void setYSpacing(int newYSpacing) {
    ySpacing = newYSpacing;
}

Listing C.17: PeerViewClient.java

package clientapp;
import com.sun.media.jsdt.*;
import com.sun.media.jsdt.event.*;
/**
 * Insert the type’s description here.
 */
* Creation date: (29-06-00 20:26:21)

public class PeerViewClient implements Client, ChannelConsumer {
    private java.lang.String name;
    /**
     * PeerViewClient constructor comment.
     */
    public PeerViewClient(String nameArg)
    {
        super();
        name = nameArg;
    }
    /**
     * authenticate method comment.
     */
    public Object authenticate(AuthenticationInfo arg1) {
        return null;
    }
    /**
     * Insert the method's description here.
     * Creation date: (30-06-00 09:42:34)
     * Parameter type parameter.
     */
    public synchronized void dataReceived(Data data)
    {
        // System.out.println( data.getDataAsString() );
    }
    /**
     * Insert the method's description here.
     * Creation date: (30-06-00 09:00:06)
     */
    public java.lang.String getName()
    {
        return name;
    }
    /**
     * Insert the method's description here.
     * Creation date: (30-06-00 09:00:06)
     * Parameter type parameter.
     */
    public void setName(java.lang.String newName) {
        name = newName;
    }
}

Listing C.18: PreferencesDialog.java

import javax.swing.*;
import javax.swing.border.*;
import javax.swing.event.*;
import java.util.Properties;
import java.io.*;
import java.awt.event.*;
import java.util.Hashtable;
import java.util.HashSet;
import java.util.*;
import javax.swing/misc.*;

/** This class implements the preferences dialog and its associated panes. */
public class PreferencesDialog extends JDialog
{
    public class Changer implements ChangeListener

332
{ public void stateChanged(ChangeEvent ce) {
  // System.out.println(getAnimationSpeedSlider().getValue());
}
};

public class OKHandler implements ActionListener {
  public void actionPerformed(ActionEvent ae) {
    submitAndClose();
  }
}

public class CancelHandler implements ActionListener {
  public void actionPerformed(ActionEvent ae) {
    cancelAndClose();
  }
}

public class ConnectHandler implements ActionListener {
  public void actionPerformed(ActionEvent ae) {
    connectToServer();
  }
}

public class CustomizeHandler implements ActionListener {
  public void actionPerformed(ActionEvent ae) {
    customizeLayout();
  }
}

private JPanel ivjDialogContentPane = null;
private JTabbedPane ivjJTabbedPane1 = null;
private JPanel ivjDisplayPage = null;
private JLabel ivjNameLabel = null;
private JPanel ivjPersonalPage = null;
private JLabel ivjSignatureLabel = null;
private JButton ivjCancelButton = null;
private JPanel ivjPanel1 = null;
private java.awt.FlowLayout ivjJPanel1FlowLayout = null;
private JButton ivjOKButton = null;
private JScrollPane ivjJScrollPane1 = null;
private ClientApplication clientApplication = null;
private JButton ivjCustomizeButton = null;
private JComboBox ivjDisplayQualityBox = null;
private JLabel ivjDisplayQualityLabel = null;
private JComboBox ivjLayoutSchemeBox = null;
private JLabel ivjLayoutSchemeLabel = null;
ivjEventHandler ivjEventHandler = new ivjEventHandler();
private JTextField ivjNameTextField = null;
private JTextField ivjSignatureTextField = null;
private JLabel ivjServerName = null;
private JPanel ivjServerPage = null;
private JLabel ivjServerPortLabel = null;
private JTextField ivjServerNameField = null;
private JTextField ivjServerPortField = null;
private JLabel ivjConnectionType = null;
private JComboBox ivjConnectionTypeBox = null;

333
private JTextField ivjUpdateIntervalField = null;
private JLabel ivjUpdateIntervalLabel = null;
private JButton ivjConnectButton = null;
private JLabel ivjAnimationSpeedLabel = null;
private JLabel ivjPauseLabel = null;
private JLabel ivjAnimationSpeedSlider = null;
private JSlider ivjAnimationSpeedSlider = null;

class IvjEventHandler implements java.awt.event.ActionListener {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        if (e.getSource() == PreferencesDialog.this.getCancelButton())
            cancelDialog(e);
    }
    /*
     * PreferencesDialog constructor comment.
     */
    public PreferencesDialog() {
        super();
        initialize();
    }
    /*
     * PreferencesDialog constructor comment.
     * @param owner java.awt.Dialog
     */
    public PreferencesDialog(java.awt.Dialog owner) {
        super(owner);
    }
    /*
     * PreferencesDialog constructor comment.
     * @param owner java.awt.Dialog
     * @param title java.lang.String
     */
    public PreferencesDialog(java.awt.Dialog owner, String title) {
        super(owner, title);
    }
    /*
     * PreferencesDialog constructor comment.
     * @param owner java.awt.Dialog
     * @param title java.lang.String
     * @param modal boolean
     */
    public PreferencesDialog(java.awt.Dialog owner, String title, boolean modal) {
        super(owner, title, modal);
    }
    /*
     * PreferencesDialog constructor comment.
     * @param owner java.awt.Dialog
     * @param modal boolean
     */
    public PreferencesDialog(java.awt.Dialog owner, boolean modal) {
        super(owner, modal);
    }
    /*
     * PreferencesDialog constructor comment.
     * @param owner java.awt.Frame
     */
    public PreferencesDialog(java.awt.Frame owner) {
        super(owner);
    }
    /*
     * PreferencesDialog constructor comment.
     * @param owner java.awt.Frame
     * @param title java.lang.String
     */
    public PreferencesDialog(java.awt.Frame owner, String title) {
    }
super(owner, title);

/**
 * PreferencesDialog constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public PreferencesDialog(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}

/**
 * PreferencesDialog constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public PreferencesDialog(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}

/**
 * Insert the method's description here.
 * @Creation date: (06-08-00 20:27:21)
 */
public void cancelAndClose()
{
    dispose();
}

/**
 * Insert the method's description here.
 * @Creation date: (10-08-00 20:27:10)
 */
public void connectToServer()
{
    ConfirmationBox confirmationBox = new ConfirmationBox();
    confirmationBox.setText(AVSL.Show.CONFIRM_CONNECT_TO_SERVER());
    confirmationBox.show();
    if ( confirmationBox.isConfirmed() )
    {        getClientApplication().getClientManager().connectToGroupManagement();
    }
}

/**
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtcM1(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin {1}
        // user code end
        this.dispose();
        // user code begin {2}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
        // user code end
        handleException(ivjExc);
    }
}

/**
 * Insert the method's description here.
 * @Creation date: (06-09-00 02:13:19)
 */
public void customizeLayout()
{
getClientApplication().getPanoramaPanel().getLayoutManager().show( this );
getClientApplication().getPanoramaPanel().getLayoutManager().updateSettings( ClientInfo.
getPreferences() );

/**
 * Return the AnimationSpeedLabel property value.
 * @return javax.swing.JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getAnimationSpeedLabel() {
    try {
        ivjAnimationSpeedLabel = new javax.swing.JLabel();
        ivjAnimationSpeedLabel.setName("AnimationSpeedLabel");
        ivjAnimationSpeedLabel.setText("Animation_speed:");
        // user code begin (1)
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
    return ivjAnimationSpeedLabel;
}

/**
 * Return the JSlider1 property value.
 * @return javax.swing.JSlider
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JSlider getAnimationSpeedSlider() {
    try {
        ivjAnimationSpeedSlider = new javax.swing.JSlider();
        ivjAnimationSpeedSlider.setName("AnimationSpeedSlider");
        // user code begin (1)
        ivjAnimationSpeedSlider.setMinimum( ClientConstants.getANIMATION_MIN_SPEED() );
        ivjAnimationSpeedSlider.setMaximum( ClientConstants.getANIMATION_MAX_SPEED() );
        ivjAnimationSpeedSlider.setPaintTicks( true );
        ivjAnimationSpeedSlider.setPaintTrack( true );
        ivjAnimationSpeedSlider.addChangeListener( new Changer() );
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
    return ivjAnimationSpeedSlider;
}

/**
 * Return the CancelButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getCancelButton() {
    try {
        ivjCancelButton = new javax.swing.JButton();
        ivjCancelButton.setName("CancelButton");
        ivjCancelButton.setText("Cancel");
        // user code begin (1)
        ivjCancelButton.addActionListener( new CancelHandler() );
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
    }
    return ivjCancelButton;
}
// user code end
handleException(ivjExc);
}

{return ivjCancelButton;
}

/**
 * Insert the method’s description here.
 * Creation date: (25-06-00 20:14:31)
 */
public ClientApplication getClientApplication() {
    return clientApplication;
}

/**
 * Return the ConnectButton property value.
 */
private javax.swing.JButton getConnectButton() {
    if (ivjConnectButton == null) {
        try {
            ivjConnectButton = new javax.swing.JButton();
            ivjConnectButton.setName("ConnectButton");
            ivjConnectButton.setMnemonic('c');
            ivjConnectButton.setText("Connect, to, server");
            // user code begin (1)
            ivjConnectButton.setEnabled( false );
            ivjConnectButton.addActionListener( new ConnectHandler() );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjConnectButton;
}

/**
 * Return the ConnectionType property value.
 */
private javax.swing.JLabel getConnectionType() {
    if (ivjConnectionType == null) {
        try {
            ivjConnectionType = new javax.swing.JLabel();
            ivjConnectionType.setName("ConnectionType");
            ivjConnectionType.setText("Connection, type:");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjConnectionType;
}

/**
 * Return the ConnectionTypeBox property value.
 */
private javax.swing.JComboBox getConnectionTypeBox() {
    if (ivjConnectionTypeBox == null) {

337
try {
    ivjConnectionTypeBox = new javax.swing.JComboBox();
    ivjConnectionTypeBox.setName("ConnectionTypeBox");
    // user code begin (1)
    // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
}
return ivjConnectionTypeBox;
*/
/**
 * Return the CustomizeButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getCustomizeButton() {
    if (ivjCustomizeButton == null) {
        try {
            ivjCustomizeButton = new javax.swing.JButton();
            ivjCustomizeButton.setName("CustomizeButton");
            ivjCustomizeButton.setMnemonic('u');
            ivjCustomizeButton.setText("Customize");
            // user code begin (1)
            ivjCustomizeButton.addActionListener(new CustomizeHandler());
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjCustomizeButton;
*/
/**
 * Return the DisplayPage property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getDisplayPage() {
    if (ivjDisplayPage == null) {
        try {
            ivjDisplayPage = new javax.swing.JPanel();
            ivjDisplayPage.setName("DisplayPage");
            ivjDisplayPage.setLayout(new java.awt.GridBagLayout());
            java.awt.GridBagConstraints constraintsDisplayQualityLabel = new java.awt.GridBagConstraints();
            constraintsDisplayQualityLabel.gridx = 0;
            constraintsDisplayQualityLabel.gridy = 0;
            constraintsDisplayQualityLabel.gridwidth = 2;
            constraintsDisplayQualityLabel.gridheight = 2;
            constraintsDisplayQualityLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsDisplayQualityLabel.weighty = 1.0;
            constraintsDisplayQualityLabel.insets = new java.awt.Insets(6, 6, 6, 6);
            getDisplayPage().add(getDisplayQualityLabel(), constraintsDisplayQualityLabel);
        } catch (java.awt.GridBagConstraints constraintsLayoutSchemeLabel = new java.awt.GridBagConstraints();
            constraintsLayoutSchemeLabel.gridx = 0;
            constraintsLayoutSchemeLabel.gridy = 1;
            constraintsLayoutSchemeLabel.gridwidth = 2;
            constraintsLayoutSchemeLabel.gridheight = 2;
            constraintsLayoutSchemeLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsLayoutSchemeLabel.weighty = 1.0;
            constraintsLayoutSchemeLabel.insets = new java.awt.Insets(6, 6, 6, 6);
            getDisplayPage().add(getLayoutSchemeLabel(), constraintsLayoutSchemeLabel);
        } catch (java.awt.GridBagConstraints constraintsDisplayQualityBox = new java.awt.GridBagConstraints();
            constraintsDisplayQualityBox.gridx = 2;
            constraintsDisplayQualityBox.gridy = 0;
```java
constraintsDisplayQualityBox.width = 2;
constraintsDisplayQualityBox.fill = java.awt.GridBagConstraints.HORIZONTAL;
constraintsDisplayQualityBox.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsDisplayQualityBox.weightx = 0.3;
constraintsDisplayQualityBox.weighty = 1.0;
constraintsDisplayQualityBox.insets = new java.awt.Insets(6, 6, 6, 6);
getDisplayPage().add(getDisplayQualityBox(), constraintsDisplayQualityBox);

java.awt.GridBagConstraints constraintsLayoutSchemeBox = new java.awt.GridBagConstraints();
constraintsLayoutSchemeBox.gridx = 2; constraintsLayoutSchemeBox.gridy = 1;
constraintsLayoutSchemeBox.fill = java.awt.GridBagConstraints.HORIZONTAL;
constraintsLayoutSchemeBox.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsLayoutSchemeBox.weightx = 0.3;
constraintsLayoutSchemeBox.weighty = 1.0;
constraintsLayoutSchemeBox.insets = new java.awt.Insets(6, 6, 6, 6);
getDisplayPage().add(getLayoutSchemeBox(), constraintsLayoutSchemeBox);

java.awt.GridBagConstraints constraintsCustomizeButton = new java.awt.GridBagConstraints();
constraintsCustomizeButton.gridx = 3; constraintsCustomizeButton.gridy = 1;
constraintsCustomizeButton.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsCustomizeButton.insets = new java.awt.Insets(4, 4, 4, 4);
getDisplayPage().add(getCustomizeButton(), constraintsCustomizeButton);

java.awt.GridBagConstraints constraintsUpdateIntervalLabel = new java.awt.GridBagConstraints();
constraintsUpdateIntervalLabel.gridx = 1; constraintsUpdateIntervalLabel.gridy = 2;
constraintsUpdateIntervalLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsUpdateIntervalLabel.weightx = 1.0;
constraintsUpdateIntervalLabel.weighty = 1.0;
constraintsUpdateIntervalLabel.insets = new java.awt.Insets(4, 4, 4, 4);
getDisplayPage().add(getUpdateIntervalLabel(), constraintsUpdateIntervalLabel);

java.awt.GridBagConstraints constraintsUpdateIntervalField = new java.awt.GridBagConstraints();
constraintsUpdateIntervalField.gridx = 2; constraintsUpdateIntervalField.gridy = 2;
constraintsUpdateIntervalField.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsUpdateIntervalField.weightx = 1.0;
constraintsUpdateIntervalField.weighty = 1.0;
constraintsUpdateIntervalField.ipadx = 100;
constraintsUpdateIntervalField.insets = new java.awt.Insets(4, 4, 4, 4);
getDisplayPage().add(getUpdateIntervalField(), constraintsUpdateIntervalField);

java.awt.GridBagConstraints constraintsAnimationSpeedLabel = new java.awt.GridBagConstraints();
constraintsAnimationSpeedLabel.gridx = 1; constraintsAnimationSpeedLabel.gridy = 4;
constraintsAnimationSpeedLabel.anchor = java.awt.GridBagConstraints.WEST;
constraintsAnimationSpeedLabel.insets = new java.awt.Insets(4, 4, 4, 4);
getDisplayPage().add(getAnimationSpeedLabel(), constraintsAnimationSpeedLabel);

java.awt.GridBagConstraints constraintsAnimationSpeedSlider = new java.awt.GridBagConstraints();
constraintsAnimationSpeedSlider.gridx = 2; constraintsAnimationSpeedSlider.gridy = 4;
constraintsAnimationSpeedSlider.gridwidth = 2;
constraintsAnimationSpeedSlider.fill = java.awt.GridBagConstraints.HORIZONTAL;
constraintsAnimationSpeedSlider.anchor = java.awt.GridBagConstraints.EAST;
constraintsAnimationSpeedSlider.weightx = 1.0;
constraintsAnimationSpeedSlider.insets = new java.awt.Insets(4, 4, 4, 4);
getDisplayPage().add(getAnimationSpeedSlider(), constraintsAnimationSpeedSlider);

java.awt.GridBagConstraints constraintsSlowLabel = new java.awt.GridBagConstraints();
constraintsSlowLabel.gridx = 3; constraintsSlowLabel.gridy = 4;
constraintsSlowLabel.anchor = java.awt.GridBagConstraints.WEST;
constraintsSlowLabel.insets = new java.awt.Insets(4, 4, 4, 4);
getDisplayPage().add(getSlowLabel(), constraintsSlowLabel);

java.awt.GridBagConstraints constraintsFastLabel = new java.awt.GridBagConstraints();
constraintsFastLabel.gridx = 3; constraintsFastLabel.gridy = 4;
constraintsFastLabel.anchor = java.awt.GridBagConstraints.EAST;
constraintsFastLabel.insets = new java.awt.Insets(4, 4, 4, 4);
getDisplayPage().add(getFastLabel(), constraintsFastLabel);

// user code begin (1)
```
// user code end
}
)
catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
}
}
}
return ivjDisplayPage;

/**
 * Return the DisplayQualityBox property value.
 */
private javax.swing.JComboBox getDisplayQualityBox() {
    try {
        ivjDisplayQualityBox = new javax.swing.JComboBox();
        ivjDisplayQualityBox.setName("DisplayQualityBox");
        // user code begin (1)
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
    return ivjDisplayQualityBox;
}

/**
 * Return the DisplayTypeLabel property value.
 */
private javax.swing.JLabel getDisplayQualityLabel() {
    try {
        ivjDisplayQualityLabel = new javax.swing.JLabel();
        ivjDisplayQualityLabel.setName("DisplayQualityLabel");
        ivjDisplayQualityLabel.setText("Display.quality:");
        // user code begin (1)
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
    return ivjDisplayQualityLabel;
}

/**
 * Return the FastLabel property value.
 */
private javax.swing.JLabel getFastLabel() {
    try {
        ivjFastLabel = new javax.swing.JLabel();
        ivjFastLabel.setName("FastLabel");
        ivjFastLabel.setText("Fast");
        // user code begin (1)
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
    }
handleException(ivjExc);
}
}
return ivjFastLabel;

/**
 * Return the JDialogContentPane property value.
 */
private javax.swing.JPanel getDialogContentPane() {
if (ivjDialogContentPane == null) {
try {
    ivjDialogContentPane = new javax.swing.JPanel();
    ivjDialogContentPane.setName("DialogContentPane");
    ivjDialogContentPane.setLayout(new java.awt.BorderLayout());
    getContentPane().add(getTabbedPane(), "center");
    getContentPane().add(getJPanel1(), "South");
    // user code begin {1}
    // user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}
return ivjDialogContentPane;
}

/**
 * Return the JPanel1 property value.
 */
private javax.swing.JPanel getJPanel1() {
if (ivjJPanel1 == null) {
try {
    ivjJPanel1 = new javax.swing.JPanel();
    ivjJPanel1.setName("JPanel1");
    ivjJPanel1.setLayout(new JPanel1FlowLayout());
    getContentPane().add(getOKButton(), getContentPane().getName());
    getContentPane().add(getCancelButton(), getContentPane().getName());
    // user code begin {1}
    // user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}
return ivjJPanel1;
}

/**
 * Return the JPanel1FlowLayout property value.
 */
private java.awt.FlowLayout getJPanel1FlowLayout() {
java.awt.FlowLayout ivjJPanel1FlowLayout = null;
try {
    // Create part */
    ivjJPanel1FlowLayout = new java.awt.FlowLayout();
    getContentPane().setLayout(ivjJPanel1FlowLayout);
    getContentPane().setAlignment(java.awt.FlowLayout.RIGHT);
} catch (java.lang.Throwable ivjExc) {
    handleException(ivjExc);
}
return ivjJPanel1FlowLayout;
612  /**
613   * Return the JScrollBar1 property value.
614   * @return javax.swing.JScrollBar
615   */
616  /** WARNING: THIS METHOD WILL BE REGENERATED. */
617  private javax.swing.JScrollBar getJScrollBar1() {
618     if (ivjJScrollBar1 == null) {
619         try {
620             ivjJScrollBar1 = new javax.swing.JScrollBar();
621             ivjJScrollBar1.setName("JScrollBar1");
622             getJScrollBar1().setViewport(viewPane());
623             // user code begin {1}
624             // user code end
625             } catch (java.lang.Throwable ivjExc) {
626                 // user code begin {2}
627                 // user code end
628                 handleException(ivjExc);
629             }
630             return ivjJScrollBar1;
631         }
632     }/**
633     * Return the JTabbedPane1 property value.
634     * @return javax.swing.JTabbedPane
635     */
636  /** WARNING: THIS METHOD WILL BE REGENERATED. */
637  private javax.swing.JTabbedPane getJTabbedPane1() {
638     if (ivjJTabbedPane1 == null) {
639         try {
640             ivjJTabbedPane1 = new javax.swing.JTabbedPane();
641             ivjJTabbedPane1.setName("JTabbedPane1");
642             ivjJTabbedPane1.addTab("Display", null, displayPage(), null, 0);
643             ivjJTabbedPane1.addTab("Personal", null, personalPage(), null, 1);
644             ivjJTabbedPane1.addTab("Server", null, serverPage(), null, 2);
645             // user code begin {1}
646             // user code end
647             } catch (java.lang.Throwable ivjExc) {
648                 // user code begin {2}
649                 // user code end
650                 handleException(ivjExc);
651             }
652     }
653     return ivjJTabbedPane1;
654 }/**
655     * Return the JComboBox5 property value.
656     * @return javax.swing.JComboBox
657     */
658  /** WARNING: THIS METHOD WILL BE REGENERATED. */
659  private javax.swing.JComboBox getLayoutSchemeBox() {
660     if (ivjLayoutSchemeBox == null) {
661         try {
662             ivjLayoutSchemeBox = new javax.swing.JComboBox();
663             ivjLayoutSchemeBox.setName("LayoutSchemeBox");
664             // user code begin {1}
665             for (int i = 0; i < ClientConstants.getLAYOUT_SCHEMES().length; i++)
666             {  
667                 ivjLayoutSchemeBox.addItem( ClientConstants.getLAYOUT_SCHEMES()[i] );
668             }
669             // user code end
670             } catch (java.lang.Throwable ivjExc) {
671                 // user code begin {2}
672                 // user code end
673                 handleException(ivjExc);
674             }
675         }
676 }
public class MyClass {

  private String myString;

  public MyClass() {
    myString = "Hello, World!";
  }

  public String getMyString() {
    return myString;
  }

  public void setMyString(String value) {
    myString = value;
  }

  public static void main(String[] args) {
    MyClass myObject = new MyClass();
    System.out.println(myObject.getMyString());
    myObject.setMyString("New Value");
    System.out.println(myObject.getMyString());
  }
}

343
* Return the OKButton property value.
* @return javax.swing.JButton
 *
 private javax.swing.JButton getOKButton () {
 if (ivjOKButton == null) {
 try {
     ivjOKButton = new javax.swing.JButton();
     ivjOKButton.setName("OKButton");
     ivjOKButton.setText("OK");
     // user code begin {1}
     // user code end
 } catch (java.lang.Throwable ivjExc) {
     // user code begin {2}
     // user code end
     handleException(ivjExc);
 }
 }
 return ivjOKButton;
 }
 */

 * Return the PersonalPage property value.
 * @return javax.swing.JPanel
 *
 private javax.swing.JPanel getPersonalPage () {
 if (ivjPersonalPage == null) {
 try {
     ivjPersonalPage = new javax.swing.JPanel();
     ivjPersonalPage.setName("PersonalPage");
     ivjPersonalPage.setLayout(new java.awt.GridBagLayout());
     java.awt.GridBagConstraints constraintsNameLabel = new java.awt.GridBagConstraints();
     constraintsNameLabel.gridx = 0; constraintsNameLabel.gridy = 0;
     constraintsNameLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
     constraintsNameLabel.weightx = 0.1;
     constraintsNameLabel.insets = new java.awt.Insets(6, 6, 6, 6);
     getPersonalPage().add(getNameLabel(), constraintsNameLabel);
     java.awt.GridBagConstraints constraintsSignatureLabel = new java.awt.GridBagConstraints();
     constraintsSignatureLabel.gridx = 0; constraintsSignatureLabel.gridy = 1;
     constraintsSignatureLabel.gridwidth = 2;
     constraintsSignatureLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
     constraintsSignatureLabel.weighty = 0.5;
     constraintsSignatureLabel.insets = new java.awt.Insets(6, 6, 6, 6);
     getPersonalPage().add(getSignatureLabel(), constraintsSignatureLabel);
     java.awt.GridBagConstraints constraintsNameTextField = new java.awt.GridBagConstraints();
     constraintsNameTextField.gridx = 1; constraintsNameTextField.gridy = 0;
     constraintsNameTextField.gridwidth = 2;
     constraintsNameTextField.fill = java.awt.GridBagConstraints.HORIZONTAL;
     constraintsNameTextField.anchor = java.awt.GridBagConstraints.NORTHEAST;
     constraintsNameTextField.weightx = 1.0;
     constraintsNameTextField.weighty = 0.1;
     constraintsNameTextField.insets = new java.awt.Insets(6, 6, 6, 6);
     getPersonalPage().add(getNameTextField(), constraintsNameTextField);
     java.awt.GridBagConstraints constraintsScrollPane = new java.awt.GridBagConstraints();
     constraintsScrollPane.gridx = 2; constraintsScrollPane.gridy = 1;
     constraintsScrollPane.gridwidth = 2;
     constraintsScrollPane.fill = java.awt.GridBagConstraints.BOTH;
     constraintsScrollPane.anchor = java.awt.GridBagConstraints.NORTHEAST;
     constraintsScrollPane.weightx = 0.5;
     constraintsScrollPane.weighty = 0.5;
     constraintsScrollPane.insets = new java.awt.Insets(6, 6, 6, 6);
     getPersonalPage().add(getScrollPane(), constraintsScrollPane);
 }

 344
// user code begin {1}
// user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}
// user code end

} return ivjPersonalPage;

/**
 * Return the ServerName property value.
 * @return javax.swing.JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getServerName() {
    if (ivjServerName == null) {
        try {
            ivjServerName = new javax.swing.JLabel();
            ivjServerName.setName("ServerName");
            ivjServerName.setText("ServerName");
            // user code begin {1}
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
                // user code end
                handleException(ivjExc);
            }
        }
        // user code end
        return ivjServerName;
    }
/**
 * Return the JTextField1 property value.
 * @return javax.swing.JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextField getServerNameField() {
    if (ivjServerNameField == null) {
        try {
            ivjServerNameField = new javax.swing.JTextField();
            ivjServerNameField.setName("ServerNameField");
            // user code begin {1}
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
                // user code end
                handleException(ivjExc);
            }
        }
        // user code end
        return ivjServerNameField;
    }
/**
 * Return the ServerPage property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getServerPage() {
    if (ivjServerPage == null) {
        try {
            ivjServerPage = new javax.swing.JPanel();
            ivjServerPage.setName("ServerPage");
            ivjServerPage.setOpaque(true);
            ivjServerPage.setLayout(new java.awt.GridBagLayout());
            java.awt.GridBagConstraints constraintsServerName = new java.awt.GridBagConstraints();
            constraintsServerName.gridx = 0; constraintsServerName.gridy = 0;
constraintsServerName.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsServerName.weightx = 1.0;
constraintsServerName.insets = new java.awt.Insets(4, 4, 4);
getServerPage().add(getServerName(), constraintsServerName);

java.awt.GridBagConstraints constraintsServerNameField = new java.awt.GridBagConstraints();
constraintsServerNameField.gridx = 1; constraintsServerNameField.gridy = 0;
constraintsServerNameField.fill = java.awt.GridBagConstraints.HORIZONTAL;
constraintsServerNameField.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsServerNameField.weightx = 1.0;
constraintsServerNameField.weighty = 1.0;
constraintsServerNameField.insets = new java.awt.Insets(4, 4, 4);
getServerPage().add(getServerNameField(), constraintsServerNameField);

java.awt.GridBagConstraints constraintsServerPortLabel = new java.awt.GridBagConstraints();
constraintsServerPortLabel.gridx = 0; constraintsServerPortLabel.gridy = 1;
constraintsServerPortLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsServerPortLabel.weightx = 1.0;
constraintsServerPortLabel.insets = new java.awt.Insets(4, 4, 4);
getServerPage().add(getServerPortLabel(), constraintsServerPortLabel);

java.awt.GridBagConstraints constraintsServerPortField = new java.awt.GridBagConstraints();
constraintsServerPortField.gridx = 1; constraintsServerPortField.gridy = 1;
constraintsServerPortField.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsServerPortField.weightx = 1.0;
constraintsServerPortField.weighty = 1.0;
constraintsServerPortField.ipadx = 100;
constraintsServerPortField.insets = new java.awt.Insets(4, 4, 4);
getServerPage().add(getServerPortField(), constraintsServerPortField);

java.awt.GridBagConstraints constraintsConnectionTypeBox = new java.awt.GridBagConstraints();
constraintsConnectionTypeBox.gridx = 1; constraintsConnectionTypeBox.gridy = 2;
constraintsConnectionTypeBox.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsConnectionTypeBox.weightx = 1.0;
constraintsConnectionTypeBox.insets = new java.awt.Insets(4, 4, 4);
getServerPage().add(getConnectionTypeBox(), constraintsConnectionTypeBox);

java.awt.GridBagConstraints constraintsConnectionType = new java.awt.GridBagConstraints();
constraintsConnectionType.gridx = 0; constraintsConnectionType.gridy = 2;
constraintsConnectionType.insets = new java.awt.Insets(4, 4, 4);
getServerPage().add(getConnectionType(), constraintsConnectionType);

java.awt.GridBagConstraints constraintsConnectButton = new java.awt.GridBagConstraints();
constraintsConnectButton.gridx = 1; constraintsConnectButton.gridy = 2;
constraintsConnectButton.anchor = java.awt.GridBagConstraints.EAST;
constraintsConnectButton.insets = new java.awt.Insets(4, 4, 4);
getServerPage().add(getConnectButton(), constraintsConnectButton);

} catch (java.lang.Throwable ivjExc) {
  // user code begin (1)
  // user code end
  handleException(ivjExc);
}

return ivjServerPage;
} /* This method will be regenerated. */

private java.awt.JTextField getServerPortField() {
  try {
    ivjServerPortField = new javax.swing.JTextField();
  }
ivjServerPortField.setName("ServerPortField");

// user code begin (1)
// user code end

catch (javax.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
}
}

return ivjServerPortField;

/**
 * Return the ServerPortLabel property value.
 * @return javax.swing.JLabel
 */

/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getServerPortLabel() {

if (ivjServerPortLabel == null) {
    try {
        ivjServerPortLabel = new javax.swing.JLabel();
        ivjServerPortLabel.setName("ServerPortLabel");
        ivjServerPortLabel.setText("Server.port:");
        // user code begin (1)
        // user code end
    } catch (javax.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }

    return ivjServerPortLabel;
}

/**
 * Return the SignatureLabel property value.
 * @return javax.swing.JLabel
 */

/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getSignatureLabel() {

if (ivjSignatureLabel == null) {
    try {
        ivjSignatureLabel = new javax.swing.JLabel();
        ivjSignatureLabel.setName("SignatureLabel");
        ivjSignatureLabel.setText("Signature:");
        // user code begin (1)
        // user code end
    } catch (javax.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }

    return ivjSignatureLabel;
}

/**
 * Return the JTextPane1 property value.
 * @return javax.swing.JTextPane
 */

/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextPane getSignaturePane() {

if (ivjSignaturePane == null) {
    try {
        ivjSignaturePane = new javax.swing.JTextPane();
        ivjSignaturePane.setName("SignaturePane");
        ivjSignaturePane.setBounds(0, 0, 306, 93);
        // user code begin (1)
        // user code end
    } catch (javax.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
    }

    return ivjSignaturePane;
}
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}
}

public java.awt.Component getVizSignaturePanel() {
    return ivjSignaturePanel;
}

/**
 * Return the SlowLabel property value.
 *
 */
private javax.swing.JLabel getSlowLabel() {
    if (ivjSlowLabel == null) {
        try {
            ivjSlowLabel = new javax.swing.JLabel();
            ivjSlowLabel.setName("SlowLabel");
            ivjSlowLabel.setText("Slow");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjSlowLabel;
}

/**
 * Return the UpdateIntervalField property value.
 *
 */
private javax.swing.JTextField getUpdateIntervalField() {
    if (ivjUpdateIntervalField == null) {
        try {
            ivjUpdateIntervalField = new javax.swing.JTextField();
            ivjUpdateIntervalField.setName("UpdateIntervalField");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjUpdateIntervalField;
}

/**
 * Return the UpdateIntervalLabel property value.
 *
 */
private javax.swing.JLabel getUpdateIntervalLabel() {
    if (ivjUpdateIntervalLabel == null) {
        try {
            ivjUpdateIntervalLabel = new javax.swing.JLabel();
            ivjUpdateIntervalLabel.setName("UpdateIntervalLabel");
            ivjUpdateIntervalLabel.setText("UpdateIntervalLabel");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjUpdateIntervalLabel;
}
```java
}  
return ivjUpdateIntervalLabel;
}  
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {
  /**
   * Uncomment the following lines to print uncaught exceptions to stdout */
   // System.out.println("-------- UNCAUGHT EXCEPTION --------");
   // exception.printStackTrace(System.out);
  */
  /**
   * Insert the method's description here.
   * Creation date: (06-08-00 17:19:26)
   */
  */
  public void init()
  {  
    initialize();
  }
  /**
   * Initializes connections
   * @exception java.lang.Exception The exception description.
   */
  /**
   * WARNING: THIS METHOD WILL BE REGENERATED. */
  private void initConnections() throws java.lang.Exception {
    /** user code begin {1}
    */
    /** user code end
    */
    getCancelButton().addActionListener(ivjEventHandler);
  }
  /**
   * WARNING: THIS METHOD WILL BE REGENERATED. */
  private void initialize() {
    try {
      /** user code begin {1}
      */
      /** user code end
      */
      setName("PreferencesDialog");
      setDefaultCloseOperationjavax.swing.WindowConstants.DISPOSE_ON_CLOSE);
      setSize(609, 298);
      setModal(true);
      setTitle("Preferences");
      getContentPane().addDialogContentPane();
      initConnections();
      } // catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
      } // user code begin {2}
      getCancelButton().addActionListener(new ORhandler());
    } // user code end
    } // user code begin {2}
    } // user code end
    } // user code begin {1}
    } // user code end
    */
    */
    main entrypoint - starts the part when it is run as an application
    * @param args java.lang.String[]
    */
    public static void main(java.lang.String[] args) {
      try {
        PreferencesDialog aPreferencesDialog;
        aPreferencesDialog = new PreferencesDialog(new ClientApplication());
        aPreferencesDialog.setModal(true);
        aPreferencesDialog.addWindowListener(new java.awt.event.WindowAdapter() {
          public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
          }
        });
```
```java
});
});
PreferencesDialog.setVisible(true);
}

} catch (Throwable exception) {
    System.err.println("Exception occurred in main()");
    exception.printStackTrace(System.out);
}

/**
   * Insert the method description here.
   * Creation date: (26-06-00 20:14:31)
   * @param newClientApplication clientapp.ClientApplication
   */
public void setClientApplication(ClientApplication newClientApplication) {
    clientApplication = newClientApplication;
}

/**
   * Overridden show method of the super-class to ensure proper initialization before the box is displayed
   * Creation date: (06-08-00 20:42:56)
   */
public void show()
{
    if (getConnectionTypeBox().getItemCount() == 0)
    {
        for (int i = 0; i < ClientConstants.getConnection_TYPES().length; i++)
        {
            getConnectionTypeBox().add Item( ClientConstants.getConnection_TYPES()[i] );
        }
    }
    if (getDisplayQualityBox().getItemCount() == 0)
    {
        Iterator iterator = ClientConstants.getDISPLAY_QUALITIES().keySet().iterator();
        while ( iterator.hasNext() )
        {
            getDisplayQualityBox().add Item( iterator.next() );
        }
    }

    Properties p = ClientInfo .getPreferences();
    setServerNameField().setText((String)p.getProperty( ClientConstants.getServer_NAME() ));
    setServerPortField().setText((String)p.getProperty( ClientConstants.getServer_PORT() ));
    setNameTextField().setText((String)p.getProperty( ClientConstants.getAUTHOR_NAME() ));
    setSignaturePane().setText((String)p.getProperty( ClientConstants.getSIGNATURE() ));
    setLayoutSchemeBox().setSelectedItem( p.getProperty( ClientConstants.getLAYOUT_SCHEME() ));
    getDisplayQualityBox().setSelectedItem( p.getProperty( ClientConstants.getDISPLAY_QUALITY() ));
    getAnimationSpeedSlider().setValue( ClientConstants.getANIMATION_MAX_SPEED() -
                                            Integer.parseInt((String)p.getProperty( ClientConstants.
                                            getANIMATION_DURATION_IN_MILLISECONDS() ));

    // System.out.println( getAnimationSpeedSlider().getValue() );
    setLocationRelativeTo( getContentPane() );
    super.show();

    /**
     * Commit preferences from preferences dialog pane and dispose of the dialog box.
     * Creation date: (26-06-00 07:50:16)
     */
    public void submitAndClose()
    {
        Properties p = ClientInfo .getPreferences();
        // fetch values from dialog box and insert into Properties object
        if (getDisplayQualityBox().getSelectedItem() != null )
        {
            p.setProperty( ClientConstants.getDISPLAY_QUALITY(), (String)getDisplayQualityBox().
                            getSelectedItem() );
```
if (getLayoutSchemeBox().getSelectedItem() != null)
{
    p.setProperty(ClientConstants.getLAYOUT_SCHEME(), (String)getLayoutSchemeBox().getSelectedItem());
}

p.setProperty(ClientConstants.getAUTHOR_NAME(), (String)getNameTextField().getText());
p.setProperty(ClientConstants.getSIGNATURE(), (String)getNamePane().getText());
p.setProperty(ClientConstants.getServer_NAME(), (String)getNameTextField().getText());
p.setProperty(ClientConstants.getServer_PORT(), (String)getServerPortField().getText());
if (getConnectionTypeBox().getSelectedItem() != null)
{
    p.setProperty(ClientConstants.getConnection_TYPE(), (String)getConnectionTypeBox().getSelectedItem());
}

p.setProperty(ClientConstants.getUPDATE_INTERVAL(), (String)getTextIntervalField().getText());
// store the inverse of the slider value to have it interpreted as a temporal value
p.setProperty(ClientConstants.getANIMATION_DURATION_IN_MILLISECONDS(), String.valueOf(ClientConstants.getANIMATION_MAX_SPEED() - getAnimationSpeedSlider().getValue()));
ClientInfo.setPreferences(p);
ClientInfo.writePreferences();
}
}

Listing C.19: PZCamera.java

package clientapp;

/**<n  * This is a customized version of the JAZZ ZCamera class. I’m not satisfied with this class as it
  * stands since the code
  * in render() is bit of a kludge but it’ll have to do for now.
  * @author:
  * @ creation date: (01-08-00 16:49:47
  * @ PZCamera constructor comment.
  */
public PZCamera()
{
    super();
    </n/**
    * PZCamera constructor comment.
    * @param layer edu.umd.cs.jazz.ZLayerGroup
    * @param aSurface edu.umd.cs.jazz.ZDrawingSurface
    */
public PZCamera( edu.umd.cs.jazz.ZLayerGroup layer, edu.umd.cs.jazz.ZDrawingSurface aSurface )
{
    super(layer, aSurface);
}</n/**
    * Insert the method’s description here.
    * @ creation date: (01-08-00 16:54:15
    * @ return boolean
    */
public boolean isCentering()
{
    return centering;
}</n/**
    * Insert the method’s description here.
    * @ creation date: (01-08-00 16:56:11
    * @param renderContext edu.umd.cs.jazz.util.ZRenderContext
    */
public void render(edu.umd.cs.jazz.util.ZRenderContext renderContext)
{
    if (isCentering() == true)
    {
        return;
    }
/**
   * Insert the method’s description here.
   * Creation date: (01-08-00 16:54:15)
   */
    public void setCentering(boolean newCentering) {
        centering = newCentering;
    }
}

Listing C.20: PZCameraListener.java

package clientapp;
/**
  * Simple extension of the JAZZ camera adapter.
  * Creation date: (13-07-00 12:16:07)
  */
public class PZCameraListener extends edu.umd.cs.jazz.event.ZCameraAdapter {
    /**
   * Insert the method’s description here.
   * Creation date: (13-07-00 12:17:43)
   */
    public void viewChanged(edu.umd.cs.jazz.event.ZCameraEvent cameraEvent)
    {
    }
}

Listing C.21: PZoomHandler.java

package clientapp;
/**
  * An extension of the JAZZ zoom handler. This solution is a bit kludgy and should be refined when
  * convenient.
  * Creation date: (16-06-00 23:46:26)
  */
public class PZoomHandler extends edu.umd.cs.jazz.event.ZoomEventHandler {
    // kan være lidt suspekt at bruge en static variabel til alle swing komponenter (se pZswing.render)
    private static boolean Zooming = false;
    private static java.awt.Rectangle targetRectangle = null;
    /**
    * PZoomHandler constructor comment.
    * @param node edu.umd.cs.jazz.ZNode
    */
    public PZoomHandler(edu.umd.cs.jazz.ZNode node) {
        super(node);
    }
    /**
   * Insert the method’s description here.
   * Creation date: (01-08-00 23:28:20)
   * @return java.awt.Rectangle
   */
public static java.awt.Rectangle getTargetRectangle() {
    return targetRectangle;
}
/**
 * Insert the method's description here.
 * @return boolean
 */
public static boolean isZooming() {
    return Zooming;
}
/**
 * Insert the method's description here.
 * @param newValue boolean
 */
public static void setZooming(boolean newValue) {
    Zooming = newValue;
}
/**
 * Insert the method's description here.
 * @param newValue java.awt.Rectangle
 */
public static void setTargetRectangle(java.awt.Rectangle newValue) {
    targetRectangle = newValue;
}
/**
 * Insert the method's description here.
 * @param newValue java.awt.Rectangle
 */
public static void startZooming() {
    Zooming = true;
    super.startZooming();
}
/**
 * Insert the method's description here.
 * @param newValue java.awt.Rectangle
 */
public static void stopZooming() {
    Zooming = false;
    super.stopZooming();
}
}

package clientapp;
/**
 * Simple extension of the JAZZ pan event handler.
 * @author:
 */
public class PZPanEventHandler extends edu.umd.cs.jazz.event.ZPanEventHandler {
    private static boolean Panning = false;
    /**
     * PZPanEventHandler constructor comment.
     * @param node edu.umd.cs.jazz.ZNode
     */
    public PZPanEventHandler(edu.umd.cs.jazz.ZNode node) {

Listing C.22: PZPanEventHandler.java
super(node);

/**
 * Insert the method’s description here.
 * Creation date: (17-06-00 13:46:49)
 * @return boolean
 */
public static boolean isPanning() {
    return Panning;
}

/**
 * Insert the method’s description here.
 * Creation date: (17-06-00 13:47:43)
 * @param mouseEvent edu.umd.cs.jazz.event.ZMouseEvent
 */
public void mousePressed(edu.umd.cs.jazz.event.ZMouseEvent mouseEvent) {
    Panning = true;
    super.mousePressed(mouseEvent);
}

/**
 * Insert the method’s description here.
 * Creation date: (17-06-00 13:48:07)
 * @param mouseEvent edu.umd.cs.jazz.event.ZMouseEvent
 */
public void mouseReleased(edu.umd.cs.jazz.event.ZMouseEvent mouseEvent) {
    Panning = false;
    super.mouseReleased(mouseEvent);
}

Listing C.23: PZSwing.java

package clientapp;

/**
 * Customization of the JAZZ Swing component wrapper.
 * Creation date: (17-06-00 10:17:04)
 * @author
 */
public class PZSwing extends edu.umd.cs.jazz.component.ZSwing {
    private edu.umd.cs.jazz.ZVisualLeaf zoomingNode = null;
    private float scaleFactor = (float)1.0;
    private java.awt.geom.AffineTransform originalTransform = null;
    private static boolean updating = false;

    /**
     * PZSwing constructor comment.
     * @param zbc edu.umd.cs.jazz.util.ZCanvas
     * @param component javax.swing.JComponent
     */
    public PZSwing(edu.umd.cs.jazz.util.ZCanvas zbc, javax.swing.JComponent component) {
        super(zbc, component);
    }

    /**
     * Insert the method’s description here.
     * Creation date: (30-07-00 17:50:17)
     * @return java.awt.geom.AffineTransform
     */
    public java.awt.geom.AffineTransform getOriginalTransform() {
        return originalTransform;
    }

    /**
     * Insert the method’s description here.
     * Creation date: (30-07-00 17:43:52)
     * @return float
     */
public float getScaleFactor() {
    return scaleFactor;
}

public edu.umd.cs.jazz.ZVisualLeaf getZoomingNode() {
    return zoomingNode;
}

public static boolean isUpdating() {
    return updating;
}

public void render(edu.umd.cs.jazz.util.ZRenderContext renderContext) {
    // don't render if updating - for efficiency and safety.
    if ( isUpdating() ) {
        return;
    }
    // skal smukkeseres
    if ( PZoomHandler.isZooming() || PPanEventHandler.isPanning() )
    {
        paintAsGreek( renderContext.getGraphics2D() );
        return;
    }
    // if a centering animation is in progress, paint as grey "blobs" rather than fully rendered graphics
    if ( PZoomHandler.getTargetRectangle() != null )
    {
        // if the animation is ongoing
        //System.out.println( renderContext.getRenderingCamera().getViewBounds() );
        //System.out.println( PZoomHandler.getTargetRectangle() );
        int currentWidth = (int)renderContext.getRenderingCamera().getViewBounds().getWidth();
        int targetWidth = (int)PZoomHandler.getTargetRectangle().getWidth();
        int currentHeight = (int)renderContext.getRenderingCamera().getViewBounds().getHeight();
        int targetHeight = (int)PZoomHandler.getTargetRectangle().getHeight();
        // AT ØK: Udskif ændretende magiske tal med konstant - eksperimenter for at finde en passende
        if ( ((currentWidth < (targetWidth - 3)) || (currentWidth > (targetWidth + 3))) &&
             ((currentHeight < (targetHeight - 3)) || (currentHeight > (targetHeight + 3))) )
        {
            paintAsGreek( renderContext.getGraphics2D() );
            return;
        }
    } else {
        PZoomHandler.setTargetRectangle( null );
    }
}

if ( zoomingNode instanceof edu.umd.cs.jazz.ZVisualLeaf )
{
    zoomingNode.editor().getTransformGroup().scale( scaleFactor );
}
super.render(renderContext);
}
* Insert the method's description here.
* Creation date: (30-07-00 17:50:17)
* @param newOriginalTransform java.awt.geom.AffineTransform
* */
public void setOriginalTransform(java.awt.geom.AffineTransform newOriginalTransform) {
    originalTransform = newOriginalTransform;
}
/**
* Insert the method's description here.
* Creation date: (30-07-00 17:43:52)
* @param newScaleFactor float
* */
public voidsetScaleFactor(float newScaleFactor) {
    scaleFactor = newScaleFactor;
}
/**
* Insert the method's description here.
* Creation date: (07-09-00 23:31:48)
* @param newUpdating boolean
* */
public static void setUpdating(boolean newUpdating) {
    updating = newUpdating;
}
/**
* Insert the method's description here.
* Creation date: (30-07-00 17:26:06)
* @param newZoomingNode edu.umd.cs.jazz.ZVisualLeaf
* */
public void setZoomingNode( edu.umd.cs.jazz.ZVisualLeaf newZoomingNode) {
    zoomingNode = newZoomingNode;
}

package clientapp;
/**
* Component for representing text items in the client panorama
* Creation date: (16-06-00 23:05:35)
* @author:
*/
public class PZText extends edu.umd.cs.jazz.component.ZText {
  /**
   * PZText constructor comment.
   */
  public PZText() {
    super();
  }
  /**
   * PZText constructor comment.
   * @param str java.lang.String
   */
  public PZText(String str) {
    super(str);
  }
  /**
   * PZText constructor comment.
   * @param str java.lang.String
   * @param font java.awt.Font
   */
  public PZText(String str, java.awt.Font font) {
    super(str, font);
  }
  /**
   */

Listing C.24: PZText.java
package clientapp;

import java.awt.event.*;
import java.awt.BorderLayout;
import javax.swing.DefaultListModel;
import java.io.File;

/**
 * This class implements the remove files dialog box.
 * @author
 */

public class RemoveFiles extends javax.swing.JDialog {

public class RemoveButtonHandler implements ActionListener {
    public void actionPerformed(ActionEvent actionEvent)
    {
        submitAndClose();
    }
}

public class CancelButtonHandler implements ActionListener {
    public void actionPerformed(ActionEvent actionEvent)
    {
        dispose();
    }
}

public class MoveLeftButtonHandler implements ActionListener {
    public void actionPerformed(ActionEvent actionEvent)
    {
        moveFromDisplayedToRemove();
    }
}

public class MoveRightButtonHandler implements ActionListener {
    public void actionPerformed(ActionEvent actionEvent)
    {
        moveFromRemoveToDisplayed();
    }
}

private javax.swing.JButton ivjCancelButton = null;
private javax.swing.JPanel ivjDialogContentPane = null;
private javax.swing.JScrollPane ivjScrollPanel1 = null;
private javax.swing.JButton ivjRemoveButton = null;
private javax.swing.DefaultListModel listModel = new DefaultListModel();

Listing C.25: RemoveFiles.java
private javax.swing.JPanel ivjButtonPanel = null;
private javax.swing.JLabel ivjDisplayedLabel = null;
private javax.swing.JLabel ivjDocumentsToRemoce = null;
private javax.swing.JScrollPane ivjScrollPane2 = null;
private javax.swing.JList ivjRemovelist = null;
private javax.swing.JLabel ivjDisplayedList = null;
private javax.swing.JButton ivjMoveLeftButton = null;
private javax.swing.JButton ivjMoveRightButton = null;
private javax.swing.DefaultListModel removeListModel = new DefaultListModel();
private javax.swing.JPanel ivjBottomPanel = null;
private java.awt.FlowLayout ivjBottomPanelFlowLayout = null;
private javax.swing.JPanel ivjMainPanel = null;
/**
 * RemoveFiles constructor comment.
 */
public RemoveFiles() {
    super();
    initialize();
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Dialog
 */
public RemoveFiles(java.awt.Dialog owner) {
    super(owner);
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Dialog
 */
public RemoveFiles(java.awt.Dialog owner, String title) {
    super(owner, title);
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Dialog
 */
public RemoveFiles(java.awt.Dialog owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * RemoveFiles constructor comment.
 */
public RemoveFiles(java.awt.Frame owner) {
    super(owner);
}
/**
 * RemoveFiles constructor comment.
 */
public RemoveFiles(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public RemoveFiles(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public RemoveFiles(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}
/**
 * Insert the method's description here.
 * @param paths java.lang.String[]
 */
public void fillListbox(java.util.Hashtable localFiles)
{
    java.util.Iterator iterator = localFiles.iterator();
    while (iterator.hasNext() == true)
    {
        getlistModel().addElement((File) iterator.next());
    }
}
/**
 * Return the JPanel1 property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getbottomPanel()
{
    if (ivjBottomPanel == null) {
        try {
            ivjBottomPanel = new javax.swing.JPanel();
            ivjBottomPanel.setName("BottomPanel");
            ivjBottomPanel.setLayout(getBottomPanelflowLayout());
            ivjBottomPanel.add(getRemoveButton());
            getBottomPanel().add(getCancelButton(), getCancelButton().getName());
            // user code begin (1)
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
            return ivjBottomPanel;
        }
        /**
         * Return the BottomPanelflowLayout property value.
         * @return java.awt.FlowLayout
         */
        /* WARNING: THIS METHOD WILL BE REGENERATED. */
        private java.awt.FlowLayout getBottomPanelflowLayout()
        {
            java.awt.FlowLayout ivjBottomPanelflowLayout = null;
            try {
                /* Create part */
                ivjBottomPanelflowLayout = new java.awt.FlowLayout();
                ivjBottomPanelflowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
                } catch (java.lang.Throwable ivjExc) {
                    handleException(ivjExc);
        */
}
```java
188     return ivjButtonPanelFlowLayout;
189 
190     /**
191     * Return the ButtonPanel property value.
192     */
193     @return javax.swing.JPanel
194     */
195     /** WARNING: THIS METHOD WILL BE REGENERATED. */
196     private javax.swing.JPanel getButtonPanel() {
197         if (ivjButtonPanel == null) {
198             try {
199                 ivjButtonPanel = new javax.swing.JPanel();
200                 ivjButtonPanel.setName("ButtonPanel");
201                 ivjButtonPanel.setPreferredSize(new java.awt.Dimension(31, 66));
202                 ivjButtonPanel.setLayout(new java.awt.GridBagLayout());
203                 ivjButtonPanel.setMinimumSize(new java.awt.Dimension(31, 66));
204                 ivjButtonPanel.setMaximumSize(new java.awt.Dimension(31, 2147483647));
205                 java.awt.GridBagConstraints constraintsMoveLeftButton = new java.awt.GridBagConstraints();
206                 constraintsMoveLeftButton.gridx = 0; constraintsMoveLeftButton.gridy = 0;
207                 getButtonPanel().add(getMoveLeftButton(), constraintsMoveLeftButton);
208                 java.awt.GridBagConstraints constraintsMoveRightButton = new java.awt.GridBagConstraints();
209                 constraintsMoveRightButton.gridx = 0; constraintsMoveRightButton.gridy = 1;
210                 getButtonPanel().add(getMoveRightButton(), constraintsMoveRightButton);
211                 // user code begin (1)
212                 // user code end
213             } catch (java.lang.Throwable ivjExc) {
214                 // user code begin (2)
215                 // user code end
216                 handleException(ivjExc);
217             }
218             }
219             return ivjButtonPanel;
220         }
221         /**
222         * Return the CancelButton property value.
223         */
224         @return javax.swing.JButton
225         */
226         /** WARNING: THIS METHOD WILL BE REGENERATED. */
227         private javax.swing.JButton getCancelButton() {
228             if (ivjCancelButton == null) {
229                 try {
230                     ivjCancelButton = new javax.swing.JButton();
231                     ivjCancelButton.setText("CancelButton");
232                     // user code begin (1)
233                     ivjCancelButton.addActionListener(new CancelButtonHandler());
234                     // user code end
235                 } catch (java.lang.Throwable ivjExc) {
236                     // user code begin (2)
237                     // user code end
238                     handleException(ivjExc);
239                 }
240             }
241             return ivjCancelButton;
242         }
243         /**
244         * Insert the method's description here.
245         */
246         @return clientapp.ClientApplication
247         */
248         public ClientApplication getClientApplication() {
249             return clientApplication;
250         }
251         /**
252         */
```
* Return the DisplayedLabel property value.
* @return javax.swing.JLabel
*<
private javax.swing.JLabel getDisplayedLabel() {  
    if (ivjDisplayedLabel == null) {  
        try {  
            ivjDisplayedLabel = new javax.swing.JLabel();  
            ivjDisplayedLabel.setName("DisplayedLabel");  
            ivjDisplayedLabel.setText("Currently_displayed");  
            // user code begin {1}  
            // user code end  
        } catch (java.lang.Throwable ivjExc) {  
            // user code begin {2}  
            // user code end  
        }  
        handleException(ivjExc);  
    }  
    return ivjDisplayedLabel;  
}

/*  
* Return the JList property value.  
* @return javax.swing.JList  
*/  
private javax.swing.JList getDisplayedList() {  
    if (ivjDisplayedList == null) {  
        try {  
            ivjDisplayedList = new javax.swing.JList();  
            ivjDisplayedList.setName("DisplayedList");  
            ivjDisplayedList.setBounds(0, 0, 0, 0);  
            // user code begin {1}  
            // user code end  
        } catch (java.lang.Throwable ivjExc) {  
            // user code begin {2}  
            // user code end  
        }  
        handleException(ivjExc);  
    }  
    return ivjDisplayedList;  
}

/*  
* Return the DocumentsToRemove property value.  
* @return javax.swing.JLabel  
*/  
private javax.swing.JLabel getDocumentsToRemove() {  
    if (ivjDocumentsToRemove == null) {  
        try {  
            ivjDocumentsToRemove = new javax.swing.JLabel();  
            ivjDocumentsToRemove.setName("DocumentsToRemove");  
            ivjDocumentsToRemove.setText("To_be_removed");  
            // user code begin {1}  
            // user code end  
        } catch (java.lang.Throwable ivjExc) {  
            // user code begin {2}  
            // user code end  
        }  
        handleException(ivjExc);  
    }  
    return ivjDocumentsToRemove;  
}

/*  
* Return the JDialogContentPane property value.  
* @return javax.swing.JPanel  
*/
private javax.swing.JDialog getDialogContentPane() {
    if (ivjDialogContentPane == null) {
        try {
            ivjDialogContentPane = new javax.swing.JPanel();
            ivjDialogContentPane.setName("DialogContentPane");
            ivjDialogContentPane.setLayout(new java.awt.BorderLayout());
            getDialogContentPane().add(getBottomPanel(), "South");
            getDialogContentPane().add(getMainPanel(), "Center");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjDialogContentPane;
}

private javax.swing.JScrollPane get JScrollPane1() {
    if (ivjScrollPane1 == null) {
        try {
            ivjScrollPane1 = new javax.swing.JScrollPane();
            ivjScrollPane1.setName("JSscrollPane1");
            get JScrollPane1().setViewportView(get DisplayedList());
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjScrollPane1;
}

private javax.swing.JScrollPane get JScrollPane2() {
    if (ivjScrollPane2 == null) {
        try {
            ivjScrollPane2 = new javax.swing.JScrollPane();
            ivjScrollPane2.setName("JSscrollPane2");
            get JScrollPane2().setViewportView(get RemoveList());
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjScrollPane2;
}
public javax.swing.DefaultListModel getListModel() {
    return listModel;
}
/**
 * Return the JPanle2 property value.
 * @return java.awt.JButton
 */
private javax.swing.JPanel getMainPanel() {
    try {
        ivjMainPanel = new javax.swing.JPanel();
        ivjMainPanel.setName("MainPanel");
        ivjMainPanel.setLayout(new java.awt.GridBagLayout());
        java.awt.GridBagConstraints constraintsButtonPanel = new java.awt.GridBagConstraints();
        constraintsButtonPanel.gridx = 1; constraintsButtonPanel.gridy = 1;
        constraintsButtonPanel.fill = java.awt.GridBagConstraints.VERTICAL;
        getMainPanel().add(getButtonPanel(), constraintsButtonPanel);
        java.awt.GridBagConstraints constraintsDisplayedLabel = new java.awt.GridBagConstraints();
        constraintsDisplayedLabel.gridx = 0; constraintsDisplayedLabel.gridy = 0;
        constraintsDisplayedLabel.anchor = java.awt.GridBagConstraints.WEST;
        constraintsDisplayedLabel.insets = new java.awt.Insets(4, 4, 4, 4);
        getMainPanel().add(getDisplayedLabel(), constraintsDisplayedLabel);
        java.awt.GridBagConstraints constraintsScrollPane = new java.awt.GridBagConstraints();
        constraintsScrollPane.gridx = 0; constraintsScrollPane.gridy = 1;
        constraintsScrollPane.fill = java.awt.GridBagConstraints.HORIZONTAL;
        constraintsScrollPane.anchor = java.awt.GridBagConstraints.WEST;
        constraintsScrollPane.weightx = 1.0;
        constraintsScrollPane.weighty = 1.0;
        constraintsScrollPane.insets = new java.awt.Insets(4, 4, 4, 4);
        getMainPanel().add(getScrollPane(), constraintsScrollPane);
        java.awt.GridBagConstraints constraintsScrollPane2 = new java.awt.GridBagConstraints();
        constraintsScrollPane2.gridx = 1; constraintsScrollPane2.gridy = 1;
        constraintsScrollPane2.fill = java.awt.GridBagConstraints.HORIZONTAL;
        constraintsScrollPane2.anchor = java.awt.GridBagConstraints.EAST;
        constraintsScrollPane2.weightx = 1.0;
        constraintsScrollPane2.weighty = 1.0;
        constraintsScrollPane2.insets = new java.awt.Insets(4, 4, 4, 4);
        getMainPanel().add(getScrollPane2(), constraintsScrollPane2);
        java.awt.GridBagConstraints constraintsDocumentsToRemove = new java.awt.GridBagConstraints();
        constraintsDocumentsToRemove.gridx = 2; constraintsDocumentsToRemove.gridy = 0;
        constraintsDocumentsToRemove.gridwidth = 1;
        constraintsDocumentsToRemove.weightx = 1.0;
        constraintsDocumentsToRemove.insets = new java.awt.Insets(4, 4, 4, 4);
        getMainPanel().add(getDocumentsToRemove(), constraintsDocumentsToRemove);
    } // user code begin (1)
    catch (javax.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    } // user code end
    return ivjMainPanel;
}
private javax.swing.JButton getMoveLeftButton() {
    if (ivjMoveLeftButton == null) {
        try {
            ivjMoveLeftButton = new javax.swing.JButton();
            ivjMoveLeftButton.setName("MoveLeftButton");
            ivjMoveLeftButton.setToolTipText("Add to, list");
            ivjMoveLeftButton.setMnemonic('a');
            ivjMoveLeftButton.setText("");
            ivjMoveLeftButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbuttonGraphics/navigation/Forward24.gif")));
            ivjMoveLeftButton.setBorderPainted(false);
            ivjMoveLeftButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
            // user code begin (1)
            ivjMoveLeftButton.addActionListener(new MoveLeftButtonHandler());
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjMoveLeftButton;
}

/**
 * Return the RemoveButton1 property value.
 * @return javax.swing.JButton
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getMoveRightButton() {
    if (ijMoveRightButton == null) {
        try {
            iijMoveRightButton = new javax.swing.JButton();
            iijMoveRightButton.setName("MoveRightButton");
            iijMoveRightButton.setToolTipText("Remove, from, list");
            iijMoveRightButton.setMnemonic('r');
            iijMoveRightButton.setText("");
            iijMoveRightButton.setMaximumSize(new java.awt.Dimension(29, 36));
            iijMoveRightButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbuttonGraphics/navigation/Back24.gif")));
            iijMoveRightButton.setBorderPainted(false);
            iijMoveRightButton.setPreferredSize(new java.awt.Dimension(29, 36));
            iijMoveRightButton.setMinimumSize(new java.awt.Dimension(29, 36));
            iijMoveRightButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
            // user code begin (1)
            iijMoveRightButton.addActionListener(new MoveRightButtonHandler());
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return iijMoveRightButton;
}

/**
 * Return the RemoveButton property value.
 * @return javax.swing.JButton
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getRemoveButton() {
    if (ivjRemoveButton == null) {
        try {
            ivjRemoveButton = new javax.swing.JButton();
            ivjRemoveButton.setName("RemoveButton");
            ivjRemoveButton.setMnemonic('r');
            ivjRemoveButton.setText("Remove");
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjRemoveButton;
}
// user code begin {1}
ivjRemoveButton.addActionListener(new RemoveButtonHandler());
// user code end

} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}

return ivjRemoveButton;
}

/**
 * Return the RemoveList property value.
 * @return javax.swing.JList
 */
private javax.swing.JList getRemoveList() {
    if (ivjRemoveList == null) {
        try {
            ivjRemoveList = new javax.swing.JList();
            ivjRemoveList.setName("RemoveList");
            ivjRemoveList.setBackground(new java.awt.Color(white));
            ivjRemoveList.setBounds(0, 0, 160, 120);
            // user code begin {1}
            ivjRemoveList.setModel(removeListModel);
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }

    }

    return ivjRemoveList;
}

/**
 * Insert the method's description here.
 * Creation date: (23-06-00 19:58:23)
 * @return javax.swing.DefaultListModel
 */
public javax.swing.DefaultListModel getRemoveListModel() {
    return removeListModel;
}

/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {
    // Uncomment the following lines to print uncaught exceptions to stdout */
    System.out.println("-------- UNCAUGHT EXCEPTION --------");
    // exception.printStackTrace(System.out);
}

/**
 * Insert the method's description here.
 * Creation date: (22-06-00 23:11:52)
 */
public void initialize() {
    try {
        // user code begin {1}
        // user code end
        setName("RemoveFiles");
        setDefaultCloseOperation(java.awt.WindowConstants.DISPOSE_ON_CLOSE);
        setSize(600, 300);
        setTitle("Remove,Documents");
        contentPane = new JDialogContentPane();
        // user code end
    } catch (java.lang.Throwable ivjExc) {
handleException(ivjExc);

// user code begin {2}
getDialogContentPane().setBorder( javax.swing.BorderFactory.createMatteBorder(6,6,6,6, 
getDialogContentPane().getBackground()));
// user code end

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
  try {
    RemoveFiles aRemoveFiles;
    aRemoveFiles = new RemoveFiles();
    aRemoveFiles.setModal(true);
    aRemoveFiles.addActionListener(new java.awt.event.ActionListener() {
      public void windowClosing(java.awt.event.WindowEvent e) {
        System.exit(0);
      }
    });
    aRemoveFiles.setVisible(true);
  } catch (Throwable exception) {
    System.err.println("Exception occurred in main()");
    exception.printStackTrace(System.out);
  }
  /* Insert the method's description here.
   * Creation date: (25-08-00 20:04:34)
   */
  public void moveFromDisplayedToRemove() {
    for (int i = listModel.size()-1; i >= 0; i--)
      if ( getDisplayedList().isSelectedIndex(i) )
        removeListModel.addElement( listModel.get(i) );
        listModel.remove(i);
  }
  /* Insert the method's description here.
   * Creation date: (23-06-00 20:06:86)
   */
  public void moveFromRemoveToDisplayed() {
    for (int i = removeListModel.size()-1; i >= 0; i--)
      if ( getRemoveList().isSelectedIndex(i) )
        listModel.addElement( removeListModel.get(i) );
        removeListModel.remove(i);
  }
  /* Insert the method's description here.
   * Creation date: (23-06-00 00:18:50)
   */
  public void setClientApplication(ClientApplication newClientApplication) {
    clientApplication = newClientApplication;
  }
  /*
Listing C.26: Validator.java

package clientapp;

/**
 * Factory class with various static member functions for validating data entered into the client
 * application.
 * @author
 */
public class Validator {
    private ClientApplication clientApplication = null;
}
/**
 * Validator constructor comment.
 */
public Validator( ClientApplication clientApplication )
{
    super();
    setClientApplication( clientApplication );
}
/**
 * Insert the method's description here. 
 * Creation date: (26-07-00 16:07:18)
 * @return clientapp.ClientApplication
 */
public ClientApplication getClientApplication() {
    return ClientApplication;
}
/**
 * Insert the method's description here. 
 * Creation date: (26-07-00 16:07:18)
 * @param newClientApplication clientapp.ClientApplication
 */
public void setClientApplication(ClientApplication newClientApplication) {
    ClientApplication = newClientApplication;
}
}
package serverapp;

import com.sun.media.jsdt.*;

/**
 * This class is intended for use by PeerView servers connecting to a JSDT session.
 * Creation date: (03-07-00 12:17:46)
 * Author:
 */
public class GroupManagementServer implements Client, ChannelConsumer {
    private java.lang.String name = "";
    
    public GroupManagementServer() {
        super();
    }
    /**
     * Insert the method's description here.
     * Creation date: (03-07-00 12:30:56)
     * @param ai com.sun.media.jsdt.AuthenticationInfo
     */
    public Object authenticate(AuthenticationInfo ai) {
        return null;
    }
    /**
     * dataReceived method comment.
     */
    public synchronized void dataReceived(Data data)
    {
    }
    /**
     * Insert the method's description here.
     * Creation date: (03-07-00 12:21:42)
     * @return java.lang.String
     */
    public java.lang.String getName() {
        return null;
    }

Listing C.27: GroupManagementServer.java
Listing C.28: PeerViewServer.java

```java
import com.sun.media.jsdt.*;
/**
 * The PeerView server class.
 * Creation date: (2007-07-00 12:21:42)
 * @param newName java.lang.String
 */
public void setName(java.lang.String newName) {
    name = newName;
}
}
```

Listing C.29: ServerApplication.java

```java
package serverapp;
import com.sun.media.jsdt.*;
/**
 * The PeerView server class.
 * Creation date: (2007-06-00 10:00:51)
 * @author
 */
public class PeerViewServer extends ServerInfo implements Client, ChannelConsumer {
    private java.lang.String name;
    /**
     * PeerViewServer constructor comment.
     */
    public PeerViewServer(String nameArg) {
        super();
        name = nameArg;
    }
    /**
     * authenticate method comment.
     */
    public Object authenticate(AuthenticationInfo arg1) {
        return null;
    }
    /**
     * Insert the method’s description here.
     * Creation date: (2007-06-00 14:24:04)
     * @param data com.sun.media.jsdt.Data
     */
    public synchronized void dataReceived(Data data) {
        // System.out.println( data.getDataAsString() );
    }
    /**
     * Insert the method’s description here.
     * Creation date: (2007-06-00 10:18:32)
     * @return java.lang.String
     */
    public java.lang.String getName() {
        return name;
    }
    /**
     * Insert the method’s description here.
     * Creation date: (2007-06-00 10:18:32)
     * @param newName java.lang.String
     */
    public void setName(java.lang.String newName) {
        name = newName;
    }
    }
```
package serverapp;

import com.sun.media.jdt.*;
import peerviewmisc.*;
import java.util.*;
import java.awt.Dimension;
import java.awt.Point;

/**
 * This class implements the server application main window.
 * Creation date: (29-06-00 17:57:51)
 * Author:
 */
public class ServerApplication extends javax.swing.JFrame {
    public Channel channel;
    private peerviewmisc.AboutDialog ivjAboutDialog = null;
    private javax.swing.JMenuItem ivjAboutMenuItem = null;
    private javax.swing.JMenuItem ivjContentsMenuItem = null;
    ivjEventHandler ivjEventHandler = new ivjEventHandler();
    private javax.swing.JButton ivjExitButton = null;
    private javax.swing.JMenuItem ivjExitMenuItem = null;
    private javax.swing.JButton ivjHelpButton = null;
    private javax.swing.JMenuItem ivjHelpMenu = null;
    private javax.swing.JPopupMenu ivjJFrameContextMenu = null;
    private javax.swing.JMenuItem ivjPanel1 = null;
    private javax.swing.JMenuItem ivjScrollPane = null;
    private javax.swing.JMenuItem ivjSeparator1 = null;
    private javax.swing.JMenuItem ivjToolBar = null;
    private javax.swing.JMenuItem ivjServerApplicationJMenu = null;
    private javax.swing.JMenuItem ivjServerMenu = null;
    private ServerSetup ivjServerSetup = null;
    private javax.swing.JButton ivjSetupButton = null;
    private javax.swing.JMenuItem ivjSetupMenu = null;
    private ServerManager serverManager = null;
    private javax.swing.JMenuItem ivjTableDefaultTableModel = null;
    private HelpDialog ivjHelpDialog = null;

    class ivjEventHandler implements java.awt.event.ActionListener {
        public void actionPerformed(java.awt.event.ActionEvent e) {
            if (e.getSource() == ServerApplication.this.getSetupMenuItem())
                connToM1(e);
            if (e.getSource() == ServerApplication.this.getAboutMenuItem())
                connToM3(e);
            if (e.getSource() == ServerApplication.this.getExitButton())
                connToM4(e);
            if (e.getSource() == ServerApplication.this.getHelpButton())
                connToM5(e);
            if (e.getSource() == ServerApplication.this.getContentsMenuItem())
                connToM6(e);
            if (e.getSource() == ServerApplication.this.getSetupButton())
                connToM7(e);
            if (e.getSource() == ServerApplication.this.getExitMenuitem())
                connToM8(e);
            if (e.getSource() == ServerApplication.this.getExitButton())
                connToM9(e);
        }
    }
}

/**
 * ServerApplication constructor comment.
 */
public ServerApplication() {
    super();
    initialize();
}
/**

public ServerApplication(String title) {
    super(title);
}

/* Center and size the window at predefined dimensions and position relative to the screen resolution */
/* Creation date: (06-08-00 19:20:24) */
public void center()
{
    Dimension size;
    Dimension preferredSize = new Dimension(
        Integer.parseInt((String) ServerInfo.getPreferences().getProperty(ServerConstants.
            getFRAME_HORIZONTAL_SIZE())),
        Integer.parseInt((String) ServerInfo.getPreferences().getProperty(ServerConstants.
            getFRAME_VERTICAL_SIZE())));
    Dimension defaultSize = new Dimension(
        Integer.parseInt(ServerConstants.getDefault_FRAME_HORIZONTAL_SIZE()),
        Integer.parseInt(ServerConstants.getDefault_FRAME_VERTICAL_SIZE()));
    Point position;
    Point preferredPosition = new Point(
        Integer.parseInt((String) ServerInfo.getPreferences().getProperty(ServerConstants.
            getFRAME_X_COORDINATE())),
        Integer.parseInt((String) ServerInfo.getPreferences().getProperty(ServerConstants.
            getFRAME_Y_COORDINATE())));
    Point defaultPosition = new Point(
        Integer.parseInt(ServerConstants.getDefault_FRAME_X_COORDINATE()),
        Integer.parseInt(ServerConstants.getDefault_FRAME_Y_COORDINATE())));
    // if no value has overridden the default setting then query the OS for the dimensions of the display
    if (preferredSize.equals(defaultSize)) {
        size = new Dimension((int) Toolkit().getScreenSize().getWidth() * ServerConstants.
            getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE(),
        (int) Toolkit().getScreenSize().getHeight() * ServerConstants.
            getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE());
    }
    else {
        size = preferredSize;
    }
    if (preferredPosition.equals(defaultPosition)) {
        position = new Point((int) Toolkit().getScreenSize().getWidth() * ((1.0 - ServerConstants.
            getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()) / 2),
        (int) Toolkit().getScreenSize().getHeight() * ((1.0 - ServerConstants.
            getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()) / 2));
    }
    else {
        position = preferredPosition;
    }
    // System.out.println( size.toString() );
    setSize(size);
    setLocation(position);
    setState(java.awt.Frame.NORMAL);
    repaint();
}

/* connEtcM1: (SetupMenuItem.action.actionPerformed(java.awt.event.ActionEvent) -> ServerSetup.show()) */
/* @param argi java.awt.event.ActionEvent */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtcM1(java.awt.event.ActionEvent argi) {
    try {

    }
// user code begin {1}
// user code end
getServerSetup().show();
// user code begin {2}
// user code end
} catch (java.lang.Throwable ivjExc) {
  // user code begin {3}
  // user code end
  handleException(ivjExc);
}

/**
  * connEtoM2: ExitMenu.action.actionPerformed(java.awt.event.ActionEvent) --> ServerApplication.dispose()
  */
private void connEtoM2(java.awt.event.ActionEvent arg1) {
  try {
    // user code begin {1}
    // user code end
    this.terminateAndClose();
    // user code begin {2}
    // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
    }
  }

  /**
  * connEtoM3: AboutMenu.action.actionPerformed(java.awt.event.ActionEvent) --> AboutDialog.show()
  */
  private void connEtoM3(java.awt.event.ActionEvent arg1) {
    try {
      // user code begin {1}
      // user code end
      getAboutDialog().show();
      // user code begin {2}
      // user code end
      } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
        // user code end
        handleException(ivjExc);
      }
  }

  /**
  * connEtoM4: NewGroupItem.action.actionPerformed(java.awt.event.ActionEvent) --> NewGroup1.show()
  */
  private void connEtoM4(java.awt.event.ActionEvent arg1) {
    try {
      // user code begin {1}
      // user code end
      this.dispose();
      // user code begin {2}
      // user code end
      } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
        // user code end
        handleException(ivjExc);
      }
  }

}
190 /*
191 * connEtoM5: (EditGroupItem.action.actionPerformed(java.awt.event.ActionEvent) --> EditGroup1.show())
192 * @param arg1 java.awt.event.ActionEvent
193 */
194 /** WARNING: THIS METHOD WILL BE REGENERATED. */
195 private void connEtoM5(java.awt.event.ActionEvent arg1) {
196     try {
197         // user code begin {1}
198         // user code end
199         getHelpDialog1().show();
200         // user code begin {2}
201         // user code end
202     } catch (java.lang.Throwable ivjExc) {
203         // user code begin {3}
204         // user code end
205         handleException(ivjExc);
206     }
207 */
208 /*
209 * connEtoM6: (DeleteGroupItem.action.actionPerformed(java.awt.event.ActionEvent) --> DeleteGroup1.show())
210 * @param arg1 java.awt.event.ActionEvent
211 */
212 /** WARNING: THIS METHOD WILL BE REGENERATED. */
213 private void connEtoM6(java.awt.event.ActionEvent arg1) {
214     try {
215         // user code begin {1}
216         // user code end
217         getHelpDialog1().show();
218         // user code begin {2}
219         // user code end
220     } catch (java.lang.Throwable ivjExc) {
221         // user code begin {3}
222         // user code end
223         handleException(ivjExc);
224     }
225 */
226 /*
227 * connEtoM7: (DefaultToolBarButton.action.actionPerformed(java.awt.event.ActionEvent) --> NewGroup1.show())
228 * @param arg1 java.awt.event.ActionEvent
229 */
230 /** WARNING: THIS METHOD WILL BE REGENERATED. */
231 private void connEtoM7(java.awt.event.ActionEvent arg1) {
232     try {
233         // user code begin {1}
234         // user code end
235         getServerSetup().show();
236         // user code begin {2}
237         // user code end
238     } catch (java.lang.Throwable ivjExc) {
239         // user code begin {3}
240         // user code end
241         handleException(ivjExc);
242     }
243 */
244 /*
245 * connEtoM8: (ExitButton.action.actionPerformed(java.awt.event.ActionEvent) --> ServerApplication.terminateAndClose())
246 * @param arg1 java.awt.event.ActionEvent
247 */
248 /** WARNING: THIS METHOD WILL BE REGENERATED. */
249 private void connEtoM8(java.awt.event.ActionEvent arg1) {
250     try {
251         // user code begin {1}
252         // user code end
253 */
this.terminateAndClose();
    // user code begin {2}
    // user code end
256  } catch (java.lang.Throwable ivjExc) {
    // user code begin {3}
    // user code end
258   handleException(ivjExc);
260  }
262 */
264 * Insert the method’s description here.*
266 * Creation date: (07-07-00 01:53:28)*
268 */
270 protected void finalize() throws Throwable {
272   terminateAndClose();
274   super.finalize();
276 */
278 * Return the AboutDialog property value.*
280 * @return clientapp.AboutDialog
282 */
284 /* WARNING: THIS METHOD WILL BE REGENERATED. */
286 private clientapp.AboutDialog getAboutDialog() {
288   if (ivjAboutDialog == null) {
290     try {
292       ivjAboutDialog = new clientapp.AboutDialog();
294       ivjAboutDialog.setName("AboutDialog");
296       ivjAboutDialog.setDefaultCloseOperation(java.awt.WindowConstants.DISPOSE_ON_CLOSE);
298     } catch (java.lang.Throwable ivjExc) {
300       handleException(ivjExc);
302     }
304   }
306   return ivjAboutDialog;
308 */
310 * Return the AboutMenuItem property value.*
312 * @return java.awt JMenuItem
314 */
316 /* WARNING: THIS METHOD WILL BE REGENERATED. */
318 private java.awt.JMenuItem getAboutMenuItem() {
320   if (ivjAboutMenuItem == null) {
322     try {
324       ivjAboutMenuItem = new java.awt.JMenuItem();
326       ivjAboutMenuItem.setName("AboutMenuItem");
328       ivjAboutMenuItem.setText("About");
330     } catch (java.lang.Throwable ivjExc) {
332       handleException(ivjExc);
334     }
336   }
338   return ivjAboutMenuItem;
340 */
342 * Return the ContentsMenuItem property value.*
344 * @return java.awt JMenuItem
346 */
348 /* WARNING: THIS METHOD WILL BE REGENERATED. */
350 private java.awt.JMenuItem getContentsMenuItem() {
if (ivjContentsMenuItem == null) {
    try {
        ivjContentsMenuItem = new javax.swing.JMenuItem();
        ivjContentsMenuItem.setName("ContentsMenuItem");
        ivjContentsMenuItem.setText("Contents");
        // user code begin (1)
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
    return ivjContentsMenuItem;
}
/**
 * Return the ExitButton property value.
 */
* @return javax.swing.JButton
*/
private javax.swing.JButton getExitButton() {
    if (ivjExitButton == null) {
        try {
            ivjExitButton = new javax.swing.JButton();
            ivjExitButton.setName("ExitButton");
            ivjExitButton.setToolTipText("Exit");
            ivjExitButton.setText("");
            ivjExitButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
            ivjExitButton.setVerticalTextPosition(javax.swing.SwingConstants.CENTER);
            ivjExitButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/ general/Stop24.gif")));
            ivjExitButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
        return ivjExitButton;
    }
/**
 * Return the ExitMenuItem property value.
 */
* @return javax.swing.JMenuItem
*/
private javax.swing.JMenuItem getExitMenuItem() {
    if (ivjExitMenuItem == null) {
        try {
            ivjExitMenuItem = new javax.swing.JMenuItem();
            ivjExitMenuItem.setName("ExitMenuItem");
            ivjExitMenuItem.setText("Exit");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
        return ivjExitMenuItem;
    }
/**
 * Return the HelpButton property value.
 */
* @return javax.swing.JButton
*/

private javax.swing.JButton getHelpButton() {
    if (ivjHelpButton == null) {
        try {
            ivjHelpButton = new javax.swing.JButton();
            ivjHelpButton.setName("HelpButton");
            ivjHelpButton.setToolTipText("Help");
            ivjHelpButton.setText("");
            ivjHelpButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
            ivjHelpButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
            ivjHelpButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/help24.png")));
            ivjHelpButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
        return ivjHelpButton;
    }
    /*
     * WARNING: THIS METHOD WILL BE REGENERATED. */
    private javaw. Swing.JMenu getHelpMenu() {
        if (ivjHelpMenu == null) {
            try {
                ivjHelpMenu = new javaw. Swing. JMenu();
                ivjHelpMenu.setName("HelpMenu");
                ivjHelpMenu.setText("Help");
                ivjHelpMenu.add(getContentsMenuItem());
                ivjHelpMenu.add(getAboutMenuItem());
                // user code begin {1}
                // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
                // user code end
                handleException(ivjExc);
            }
        }
    }
    /**
     * Return the HelpDialog property value.
     * @return peerviewmisc. HelpDialog
     */
    /*
     * WARNING: THIS METHOD WILL BE REGENERATED. */
    private peerviewmisc. HelpDialog getHelpDialog() {
        try {
            ivjHelpDialog1 = new peerviewmisc. HelpDialog();
            ivjHelpDialog1.setName("HelpDialog1");
            ivjHelpDialog1.setDefaultCloseOperation(javax.swing. WindowConstants.DISPOSE_ON_CLOSE);
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
        return ivjHelpDialog1;
    }
    /**
     * Return the HelpMenu property value.
     * @return javaw. Swing.JMenu
     */
    /*
     * WARNING: THIS METHOD WILL BE REGENERATED. */
    private javaw. Swing. JMenu getHelpMenu() {
        try {
            ivjHelpMenu = new javaw. Swing. JMenu();
            ivjHelpMenu.setName("HelpMenu");
            ivjHelpMenu.setText("Help");
            ivjHelpMenu.add(getContentsMenuItem());
            ivjHelpMenu.add(getAboutMenuItem());
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
}
return ivjHelpMenu;

/**
 * Return the JFrameContentPane property value.
 * @return javax.swing.JPanel
 */
/*
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJFrameContentPane ()
{
    try {
        ivjFrameContentPane = new javax.swing.JPanel();
        ivjFrameContentPane.setName("JFrameContentPane");
        ivjFrameContentPane.setLayout (new java.awt.BorderLayout ());
        ivjFrameContentPane().add (getToolBar (), "North");
        ivjFrameContentPane().add (getPanel (), "Center");
        // user code begin {1}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException (ivjExc);
    }
    return ivjFrameContentPane;
}

/**
 * Return the JPanel1 property value.
 * @return javax.swing.JPanel
 */
/*
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getPanel1 ()
{
    try {
        ivjPanel1 = new javax.swing.JPanel();
        ivjPanel1.setName("Panel1");
        ivjPanel1.setLayout (new java.awt.BorderLayout ());
        ivjPanel1().add (getScrollPane (), "Center");
        // user code begin {1}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException (ivjExc);
    }
    return ivjPanel1;
}

/**
 * Return the JScrollPane property value.
 * @return javax.swing.JScrollPane
 */
/*
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JScrollPane getScrollPane ()
{
    try {
        ivjScrollPane = new javax.swing.JScrollPane();
        ivjScrollPane.setName("ScrollPane");
        getScrollPane().setViewportView (getScrollPaneTable ());
        // user code begin {1}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException (ivjExc);
    }
}
return ivjJScrollPane;

/**
 * Return the JSeparator1 property value.
 * @return javax.swing.JSeparator
 */
private javax.swing.JSeparator getJSeparator1() { if (ivjJSeparator1 == null) {
  try {
    ivjJSeparator1 = new javax.swing.JSeparator();
    ivjJSeparator1.setName("JSeparator1");
    // user code begin {1}
    // user code end
  } catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
  }
  return ivjJSeparator1;
}

/**
 * Return the JToolBar1 property value.
 * @return javax.swing.JToolBar
 */
private javax.swing.JToolBar getJToolBar1() { if (ivjJToolBar1 == null) {
  try {
    ivjJToolBar1 = new javax.swing.JToolBar();
    ivjJToolBar1.setName("JToolBar1");
    getJToolBar1().add(getSetupButton(), getSetupButton().getName());
    ivjJToolBar1.addSeparator();
    ivjJToolBar1.addSeparator();
    getJToolBar1().add(getHelpButton(), getHelpButton().getName());
    getJToolBar1().add(getExitButton(), getExitButton().getName());
    // user code begin {1}
    // user code end
  } catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
  }
  return ivjJToolBar1;
}

/**
 * Return the JScrollPane property value.
 * @return javax.swing.JScrollPane
 */
private javax.swing.JScrollPane getScrollPane1() { if (ivjScrollPane1 == null) {
  try {
    ivjScrollPane1 = new javax.swing.JScrollPane();
    ivjScrollPane1.setName("JScrollPane");
    get JScrollPane1().setColumnHeaderView(ivjScrollPane1.get TableHeader());
    get JScrollPane1().setViewport().setBack ingStoreEnabled(true);
    ivjScrollPane1.setAutoResizeMode(javax.swing.JScrollPane.AUTO_RESIZE_OFF);
    ivjScrollPane1.setPreferredSize(new java.awt.Dimension(716,412));
    ivjScrollPane1.setBounds(0, 0, 716, 412);
    ivjScrollPane1.setPreferredViewportSize(new java.awt.Dimension(700, 400));
    ivjScrollPane1.setAutoCreateColumnsFromModel(true);
    // user code begin {1}
    // user code end
    ivjScrollPane1.setModel( getModel());
}
// user code end

catch (java.lang.Throwable ivjExc) {
    // user code begin (2)
    // user code end
    handleException(ivjExc);
}

// user code end

/**
 * Return the ServerApplication.JMenuBar property value.
 * @return javax.swing.JMenuBar
 */

private javax.swing.JMenuBar getServerApplicationJMenuBar() {
    if (ivjServerApplicationJMenuBar == null) {
        try {
            ivjServerApplicationJMenuBar = new javax.swing.JMenuBar();
            ivjServerApplicationJMenuBar.setName("ServerApplication.JMenuBar");
            ivjServerApplicationJMenuBar.add(getSetupMenu());
            ivjServerApplicationJMenuBar.add(getHelpMenu());
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjServerApplicationJMenuBar;
}

/**
 * Insert the method's description here.
 * @return serverapp.ServerManager
 */

public ServerManager getServerManager() {
    if (serverManager == null) {
        serverManager = new ServerManager();
        serverManager.setServerApplication( this );
    }
    return serverManager;
}

/**
 * Return the ServerMenu property value.
 * @return javax.swing.JMenu
 */

private javax.swing.JMenu getServerMenu() {
    if (ivjServerMenu == null) {
        try {
            ivjServerMenu = new javax.swing.JMenu();
            ivjServerMenu.setName("ServerMenu");
            ivjServerMenu.setText("Setup");
            ivjServerMenu.add(getSetupMenuItem());
            ivjServerMenu.add(getSeparator());
            ivjServerMenu.add(getMenuExitItem());
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjServerMenu;
}
public ServerSetup getServerSetup()
{
    try
    {
        ivjServerSetup = new serverapp.ServerSetup();
        ivjServerSetup.setName("ServerSetup");
        ivjServerSetup.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        if (ivjServerSetup == null) {
            throw new RuntimeException("Uninitialized ServerSetup instance");
        }
        return ivjServerSetup;
    }
    catch (Exception e)
    {
        throw new RuntimeException(e);
    }
}

/*
 * Return the ServerSetup property value.
 */
private ServerSetup getServerSetup()
{
    if (ivjServerSetup == null) {
        try
        {
            ivjServerSetup = new serverapp.ServerSetup();
            ivjServerSetup.setName("ServerSetup");
            ivjServerSetup.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            // user code begin (1)
            // user code end
        } catch (Exception e) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjServerSetup;
}

/*
 * Return the SetupButton property value.
 */
private JButton getSetupButton()
{
    if (ivjSetupButton == null) {
        try
        {
            ivjSetupButton = new JButton();
            ivjSetupButton.setName("SetupButton");
            ivjSetupButton.setToolTipText("Server setup");
            ivjSetupButton.setText("Setup");
            ivjSetupButton.setHorizontalAlignment(JButton.CENTER);
            ivjSetupButton.setVerticalAlignment(JButton.TOP);
            ivjSetupButton.setIcon(new ImageIcon(Toolkit.getDefaultToolkit().getImage(">
        } catch (Exception e) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjSetupButton;
}

/*
 * Return the SetupMenu property value.
 */
private JMenu getSetupMenu()
{
    if (ivjSetupMenu == null) {
        try
        {
            ivjSetupMenu = new JMenu();
            ivjSetupMenu.setName("SetupMenu");
            ivjSetupMenu.setText("Setup");
            // user code begin (1)
            // user code end
        } catch (Exception e) {
            // user code begin (2)
        }
    }
    return ivjSetupMenu;
    712    // user code end    
    713    handleException(ivjExc);
    714    }
    715    }
    716    return ivjSetupMenuItem;
    717 }
    718 /**
    719 * Insert the method's description here.
    720 * Creation date: (17-07-00 21:34:13)
    721 * @return javax.swing.table.DefaultTableModel
    722 */
    723 public javax.swing.table.DefaultTableModel getTableModel() {
    724    if (tableModel == null)
    725    {
    726        tableModel = new javax.swing.table.DefaultTableModel();
    727    }
    728    return tableModel;
    729 }
    730 /**
    731 * Called whenever the part throws an exception.
    732 * @param exception java.lang.Throwable
    733 */
    734 private void handleException(java.lang.Throwable exception) {
    735    /* Uncomment the following lines to print uncaught exceptions to stdout */
    736    // System.out.println("-------- UNCAUGHT EXCEPTION ---------");
    737    // exception.printStackTrace(System.out);
    738 }
    739 /**
    740 * Initializes connections
    741 * @exception java.lang.Exception The exception description.
    742 * @see util.print
    743 */
    744 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    745 private void initConnections() throws java.lang.Exception {
    746    // user code begin {1}
    747    // user code end
    748    getSetupMenuItem().addActionListener(ivjEventHandler);
    749    getAboutMenuItem().addActionListener(ivjEventHandler);
    750    getHelpButton().addActionListener(ivjEventHandler);
    751    getContentsMenuitem().addActionListener(ivjEventHandler);
    752    getSetupButton().addActionListener(ivjEventHandler);
    753    getExitMenuitem().addActionListener(ivjEventHandler);
    754 }
    755 /*
    756 * Setup group table so that it reflects the current state of the group directory.
    757 * Creation date: (17-07-00 17:00:46)
    758 */
    759 public void initGroupTable() {
    760    WorkGroup workGroup = new WorkGroup();
    761    tableModel = new javax.swing.table.DefaultTableModel();
    762    ivjScrollPaneTable.setModel( tableModel );
    763    String columnNames[] = WorkGroup.getFieldNames();
    764    for ( int i = 0 ; i < columnNames.length; i++ )
    765    {
    766        tableModel.addColumn( columnNames[i] );
    767    }
    768    Collection values = serverManager.getServerInfo().getGroups().values();
    769    Iterator iterator = values.iterator();
    770    while ( iterator.hasNext() )
    771    {
    772        workGroup = (WorkGroup)iterator.next();
    773        Vector v = workGroup.getGroupsAsVector();
    774        tableModel.addRow( v );
    775    }
```java
    }

    } // user code begin {1}
    // user code end
    setName("ServerApplication");
    setDefaultCloseOperation(java.awt.WindowConstants.DISPOSE_ON_CLOSE);
    setTitle("PeerView_server");
    setSize(698, 528);
    setJMenuBar(getServerApplication().JMenuBar());
    getContentPane().add(JFrameContentPane());
    initConnections();
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    } // user code begin {2}
    getServerManager().setServerApplication( this );
    getServerManager().configure();
    center();
    initTable();
    updateTable();
    } // user code end

/**
 * Insert the method's description here.
 * Creation date: (04-07-00 23:11:34)
 */
public void initTable()
{
    WorkGroup workGroup = new WorkGroup();
    DefaultTableModel tableModel = new javax.swing.table.DefaultTableModel();
    ivjScrollPaneTable.getContentPane().setModel ( tableModel );
    String columnNames[] = WorkGroup.getFieldNames();
    for ( int i = 0; i < columnNames.length; i++ )
    {
        tableModel.addColumn( columnNames[i] );
    }
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args)
{
    parseCommandLineArguments( args );
    if ( ServerInfo.isVisible() == false )
    {
        ServerManager serverManager = new ServerManager();
        serverManager.configure();
        return;
    }
    try {
    ServerApplication aServerApplication;
    aServerApplication = new ServerApplication();
    aServerApplication.addEventListener(new java.awt.event.MouseAdapter()
{
        public void mouseClicked(java.awt.event.MouseEvent e) {
            System.exit(0);
        }
    });
    }
```
aServerApplication.setVisible( ServerInfo.isVisible() );
} catch ( Throwable exception ) {
    System.err.println( "Exception occurred in main() of javax.swing.JFrame" );
    exception.printStackTrace(System.out);
}

/**
 * Parse the array of command line arguments in compliance with the POSIX specifications.
 * @param args java.lang.String[]
 */
public static void parseCommandLineArguments(String[] args)
{
    for ( int i = 0 ; i < args.length; i++ )
    {
        if ( args[i].equalsIgnoreCase( ServerConstants.INVISIBLE_ARGUMENT() ) )
            ServerInfo.setVisible( false );
        else if ( args[i].equalsIgnoreCase( ServerConstants.INVISIBLE_NODE_MESSAGE() ) )
        {
            System.err.println( ServerConstants.INVISIBLE_NODE_MESSAGE() );
        }
        else if ( args[i].equalsIgnoreCase( ServerConstants.SOCKET_ARGUMENT() ) )
        {
            ServerInfo.getPreferences().put( ServerConstants.getConnection_type(), ServerConstants.getSOCKET() );
        }
        else if ( args[i].equalsIgnoreCase( ServerConstants.HTTP_ARGUMENT() ) )
        {
            ServerInfo.getPreferences().put( ServerConstants.getConnection_type(), ServerConstants.getHTTP() );
        }
    }
}

/**
 * Insert the method's description here.
 * @param newServerManager ServerManager
 */
public void setServerManager(ServerManager newServerManager) {
    serverManager = newServerManager;
}

/**
 * Insert the method's description here.
 * @param newTableModel javax.swing.table.DefaultTableModel
 */
public void setTableModel(javax.swing.table.DefaultTableModel newTableModel) {
    tableModel = newTableModel;
}

/**
 * Insert the method's description here.
 * @param newTableModel javax.swing.table.DefaultTableModel
 */
public void terminateAndClose()
{
    serverManager().terminate();
    dispose();
}

/**
 * Update the table underlying the server copy of the group directory.
 * @param newTableModel javax.swing.table.DefaultTableModel
 */
public void updateTable()
{
    // clear the table
    for ( int i = 0; i < tableModel.getRowCount(); i++ )
    {
    }
package serverapp;

import javax.swing.border.*;
import javax.swing.BorderFactory.*;
import peerviewmisc.SharedConstants;
import java.util.Properties;

/**
 * This class holds constants specific to the PeerView server as well as those inherited from PeerViewSharedConstants.
 * @author
 * @author
 */

public class ServerConstants extends SharedConstants {

    private final static java.lang.String PREferences_FILE_NAME = "srefs.ini";
    private final static java.lang.String GROUPS_FILE_NAME = "groups.dat";
    private final static java.lang.String GROUP_TABLE_CAST_FAILURE_MESSAGE = "Could not complete read of group table, from disk";
    private final static java.lang.String ROOT_NODE_MESSAGE_TITLE = "System Notice";
    private final static java.lang.String PROPERTIES_HEADER = "";
    private final static java.lang.String FRAME_HORZONTAL_SIZE = "Frame,horizontal size";
    private final static java.lang.String FRAME_VERTICAL_SIZE = "Frame,vertical size";
    private final static java.lang.String DEFAULT_FRAME_HORIZONTAL_SIZE = "0";
    private final static java.lang.String DEFAULT_FRAME_VERTICAL_SIZE = "0";
    private final static java.lang.String FRAME_X_COORDINATE = "frame,x,coordinate";
    private final static java.lang.String FRAME_Y_COORDINATE = "frame,y,coordinate";
    private final static java.lang.String DEFAULT_FRAME_X_COORDINATE = "0";
    private final static java.lang.String DEFAULT_FRAME_Y_COORDINATE = "0";
    final static String DEFAULT_PREFERENCES[] = {
            {SERVER_NAME, DEFAULT_SERVER_NAME},
            {SERVER_PORT, String.valueOf(DEFAULT_SERVER_PORT)},
            {CONNECTION_TYPE, DEFAULT_CONNECTION_TYPE},
            {FRAME_HORZONTAL_SIZE, DEFAULT_FRAME_HORIZONTAL_SIZE},
            {FRAME_VERTICAL_SIZE, DEFAULT_FRAME_VERTICAL_SIZE},
            {FRAME_X_COORDINATE, DEFAULT_FRAME_X_COORDINATE},
            {FRAME_Y_COORDINATE, DEFAULT_FRAME_Y_COORDINATE},
            {getREGISTRY_PORT(), getDEFAULT_REGISTRY_PORT()},
    }

    private final static double FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE = 0.5;
    private final static java.lang.String MESSAGES_FILE_NAME = "messages.dat";

    public ServerConstants() {
        super();
    }

    public void init() {
        super.init();
    }
}

Listing C.30: ServerConstants.java
private final static java.lang.String COLDU_CLOSE_GROUP_FILE = "Could not close group directory file. The directory may not have been properly initialized."

/**
 * ServerConstants constructor comment.
 */
	public ServerConstants() {
		super();
	}

/**
 * Insert the method's description here.
 * Creation date: (06-10-00 09:48:06)
 */
	public final static java.lang.String getCOLDU_CLOSE_GROUP_FILE() {
		return COLDU_CLOSE_GROUP_FILE;
	}

/**
 * Insert the method's description here.
 * Creation date: (15-08-00 09:41:01)
 */
	public final static java.lang.String getCOLDU_READ_GROUPS() {
		return COLDU_READ_GROUPS;
	}

/**
 * Insert the method's description here.
 * Creation date: (14-08-00 23:24:44)
 */
	public final static java.lang.String getCOLDU_READ_MESSAGES() {
		return COLDU_READ_MESSAGES;
	}

/**
 * Insert the method's description here.
 * Creation date: (15-08-00 08:42:06)
 */
	public final static java.lang.String getCOLDU_WRITE_GROUPS() {
		return COLDU_WRITE_GROUPS;
	}

/**
 * Insert the method's description here.
 * Creation date: (14-08-00 23:29:56)
 */
	public final static java.lang.String getCOLDU_WRITE_MESSAGES() {
		return COLDU_WRITE_MESSAGES;
	}

/**
 * Insert the method's description here.
 * Creation date: (06-08-00 19:49:43)
 */
	public final static java.lang.String GET_DEFAULT_FRAME_HORIZONTAL_SIZE() {
		return DEFAULT_FRAME_HORIZONTAL_SIZE;
	}
/**
 * Insert the method's description here.
 * Creation date: (05-08-00 19:50:26)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_FRAME_VERTICAL_SIZE() {
    return DEFAULT_FRAME_VERTICAL_SIZE;
}

/**
 * Insert the method's description here.
 * Creation date: (05-08-00 19:51:49)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_FRAME_X_COORDINATE() {
    return DEFAULT_FRAME_X_COORDINATE;
}

/**
 * Insert the method's description here.
 * Creation date: (05-08-00 19:52:26)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_FRAME_Y_COORDINATE() {
    return DEFAULT_FRAME_Y_COORDINATE;
}

/**
 * Insert the method's description here.
 * Creation date: (03-07-00 14:34:27)
 * @return java.util.Properties
 */
public static java.util.Properties getDEFAULT_PREFERENCES() {
    Properties p = new Properties();
    for (int i = 0; i < DEFAULT_PREFERENCES.length; i++)
    {
        p.setProperty( DEFAULT_PREFERENCES[i][0], DEFAULT_PREFERENCES[i][1] );
    }
    return p;
}

/**
 * Insert the method's description here.
 * Creation date: (05-08-00 19:49:06)
 * @return java.lang.String
 */
public final static java.lang.String getFRAME_HORIZONTAL_SIZE() {
    return FRAME_HORIZONTAL_SIZE;
}

/**
 * Insert the method's description here.
 * Creation date: (05-08-00 20:06:17)
 * @return int
 */
public final static double getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE() {
    return FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE;
}

/**
 * Insert the method's description here.
 * Creation date: (05-08-00 19:49:24)
 * @return java.lang.String
 */
public final static java.lang.String getFRAME_VERTICAL_SIZE() {
    return FRAME_VERTICAL_SIZE;
}

/**
 * Insert the method's description here.

public final static java.lang.String getFRAME_X COORDINATE() { return FRAME_X COORDINATE; }

/**
 * Insert the method's description here.
 * @return java.lang.String
 */

public final static java.lang.String getFRAME_Y COORDINATE() { return FRAME_Y COORDINATE; }

/**
 * Insert the method's description here.
 * @return java.lang.String
 */

public final static java.lang.String getGROUP_TABLE_CAST_FAILURE_MESSAGE() { return GROUP_TABLE_CAST_FAILURE_MESSAGE; }

/**
 * Insert the method's description here.
 * @return java.lang.String
 */

public final static java.lang.String getGROUPS_FILE_NAME() { return GROUPS_FILE_NAME; }

/**
 * Insert the method's description here.
 * @return java.lang.String
 */

public final static java.lang.String getINVISIBLE_ARGUMENT() { return INVISIBLE_ARGUMENT; }

/**
 * Insert the method's description here.
 * @return java.lang.String
 */

public final static java.lang.String getINVISIBLE_MODE_MESSAGE() { return INVISIBLE_MODE_MESSAGE; }

/**
 * Insert the method's description here.
 * @return java.lang.String
 */

public final static java.lang.String getMESSAGES_FILE_NAME() { return MESSAGES_FILE_NAME; }

/**
 * Insert the method's description here.
 * @return java.lang.String
 */

public final static java.lang.String getPREFERENCES_FILE_NAME() { return PREFERENCES_FILE_NAME; }

/**
 * Insert the method's description here.
 * @return java.lang.String
 */

public final static java.lang.String getPREPARE_FILE_NAME() { return PREPARE_FILE_NAME; }

/**
 * @return java.lang.String
 */
public final static java.lang.String getProperties_HEADER() {
    return PROPERTIES_HEADER;
}

/**
 * Insert the method’s description here.
 * Creation date: (10-07-00 00:50:06)
 * @param documentName java.lang.String
 * @param senderName java.lang.String
 */

public static String get_ROOT_NODE_MESSAGE_CONTENTS(String documentName, String senderName, java.util.Date creationDate)
{
    return "This is the discussion forum for the document,\" + documentName + \" which was added by,\" + senderName + \" on\" + creationDate.toString();
}

/**
 * Insert the method’s description here.
 * Creation date: (10-07-00 00:50:06)
 * @param documentName java.lang.String
 * @param senderName java.lang.String
 */

public static String get_ROOT_NODE_MESSAGE_CONTENTS(String documentName, String senderName, java.util.GregorianCalendar creationDate)
{
    return "This is the discussion forum for the document,\" + documentName + \" which was added by,\" + senderName + \" on\" + creationDate.getTime().toString();
}

/**
 * Insert the method’s description here.
 * Creation date: (10-07-00 00:48:37)
 * @return java.lang.String
 */

public final static java.lang.String get_ROOT_NODE_MESSAGE_TITLE()
{
    return ROOT_NODE_MESSAGE_TITLE;
}

Listing C.31: ServerInfo.java

package serverapp;
import java.util.Properties;
import java.util.Hashtable;
import peerviewmisc.*;
import java.io.*;

/**
 * The PeerView server main entity object. This class holds data structures and fields shared by several server components.
 * Creation date: (26-06-00 22:16:34)
 * @author:
 */

public class ServerInfo
{
    private com.sun.media.jsdt.Channel groupManagementChannel = null;
    private java.util.Hashtable groups = new Hashtable();
    private static java.util.Properties preferences = null;
    private com.sun.media.jsdt.Session groupManagementSession;
    private int mostRecentPortNumber = ServerConstants.DEFAULT_SERVER_PORT();
    private java.util.Hashtable discussions = new Hashtable();
    private java.util.Hashtable messages = new Hashtable();
}

388
private static boolean visible = true;

/**
 * ServerInfo constructor comment.
 */
public ServerInfo()
{
    super();
}

/**
 * Insert the method's description here.
 * Creation date: (14-08-00 23:05:59)
 * @exception java.lang.Throwable The exception description.
 */
protected void finalize() throws java.lang.Throwable
{
    ...
}

/**
 * Insert the method's description here.
 * Creation date: (09-07-00 15:03:03)
 * @return java.util.Hashtable
 */
public java.util.Hashtable getDiscussions()
{
    return discussions;
}

/**
 * Insert the method's description here.
 * Creation date: (01-07-00 01:20:56)
 * @return com.sun.media.jsdt.Channel
 */
public com.sun.media.jsdt.Channel getGroupManagementChannel()
{
    return groupManagementChannel;
}

/**
 * Insert the method's description here.
 * Creation date: (04-07-00 17:12:12)
 * @return com.sun.media.jsdt.Session
 */
public com.sun.media.jsdt.Session getGroupManagementSession()
{
    return groupManagementSession;
}

/**
 * Insert the method's description here.
 * Creation date: (03-07-00 12:11:40)
 * @return java.util.Properties
 */
public java.util.Hashtable getGroups()
{
    if (groups == null)
    {
        groups = new Hashtable();
    }
    return groups;
}

/**
 * Insert the method's description here.
 * Creation date: (10-07-00 01:27:49)
 * @return java.util.Hashtable
 */
public java.util.Hashtable getMessages()
{
    return messages;
}

/**
 * Insert the method's description here.
 * Creation date: (07-07-00 17:30:36)
 */

389
public int getMostRecentPortNumber() {
    return mostRecentPortNumber;
}
/**
 * Insert the method's description here.
 * Creation date: (02-07-00 22:08:51)
 * @return java.util.Properties
 */
public static java.util.Properties getPreferences()
{
    if ( preferences == null)
    {
        preferences = new Properties( ServerConstants.getDefault_PREFERENCES() );
        readPreferences();
    }
    return preferences;
}
/**
 * Insert the method's description here.
 * Creation date: (10-09-00 07:09:29)
 * @return boolean
 */
public static boolean isVisible()
{
    return visible;
}
/**
 * Read preferences from Properties object on disk. Each call to this function results in a read
 * from disk rather than an internal lookup, i.e. the values read from disk are not cached as a class
 * member.
 * This eliminates data redundancy and ensures consistency. The overhead is negligible as the file in
 * question is essentially
 * plain text.
 * Algorithm:
 * 1. Construct new Properties object P
 * 2. Initialize P with default preferences
 * 3. Read user preferences from disk file
 * 4. IF read is successful THEN
 * 4.1. Store user preferences in P
 * 5. Return P
 * This approach will return a valid object even if the read fails or in case of the application being
 * used for the first
 * time.
 * Creation date: (25-06-00 13:54:56)
 */
public static void readPreferences()
{
    try
    {
        FileInputStream f = new FileInputStream( ServerConstants.getPREFERENCES_FILE_NAME() );
        getPreferences().load(f);
        f.close();
    }
    catch (FileNotFoundException ffe)
    {
        // the exception can be handled by ignoring it since the default values in p will substitute for
        // what should
        // have been read from file.
    }
    catch (IOException ie)
    {
        // any IOException != FileNotFound should be dealt with in the conventional manner
        System.err.println( ie.toString() );
        System.exit(0);
    }
}
/* Insert the method’s description here. */
public void setDiscussions(java.util.Hashtable newDiscussions) {
    discussions = newDiscussions;
}
/**
 * Insert the method’s description here.
 * Creation date: (01-07-00 01:20:55)
 * @param newGroupManagementChannel com.sun.media.jsdt.Channel
 */
public void setGroupManagementChannel(com.sun.media.jsdt.Channel newGroupManagementChannel) {
    groupManagementChannel = newGroupManagementChannel;
}
/**
 * Insert the method’s description here.
 * Creation date: (04-07-00 17:12:12)
 * @param newGroupManagementSession com.sun.media.jsdt.Session
 */
public void setGroupManagementSession(com.sun.media.jsdt.Session newGroupManagementSession) {
    groupManagementSession = newGroupManagementSession;
}
/**
 * Insert the method’s description here.
 * Creation date: (03-07-00 12:11:40)
 * @param newGroups java.util.Properties
 */
public void setGroups(java.util.Hashtable newGroups) {
    groups = newGroups;
}
/**
 * Insert the method’s description here.
 * Creation date: (10-07-00 01:27:49)
 * @param newMessages java.util.Hashtable
 */
public void setMessages(java.util.Hashtable newMessages) {
    messages = newMessages;
}
/**
 * Insert the method’s description here.
 * Creation date: (07-07-00 17:30:36)
 * @param newLastRecentPortNumber int
 */
public void setMostRecentPortNumber(int newMostRecentPortNumber) {
    mostRecentPortNumber = newMostRecentPortNumber;
}
/**
 * Insert the method’s description here.
 * Creation date: (03-07-00 14:38:46)
 * @param newPreferences java.util.Properties
 */
public void setPreferences(java.util.Properties newPreferences) {
    preferences = newPreferences;
}
/**
 * Insert the method’s description here.
 * Creation date: (10-09-00 07:59:29)
 * @param newVisible boolean
 */
public static void setVisible(boolean newVisible) {
    visible = newVisible;
}
/**
 * Write preferences to disk file.
 * Creation date: (17-07-00 12:13:32)
 */
* @param preferences java.util.Properties
 */
public static void writePreferences()
{
  // write Properties object to disk
  try
  {
    FileOutputStream f = new FileOutputStream( ServerConstants.getPREFERENCES_FILE_NAME() );
    getPreferences().store( f, ServerConstants.getPROPERTIES_HEADER() );
    f.close();
  }
  catch (IOException ie)
  {
    System.err.println( ie.toString() );
    System.exit(0);
  }
}
}

Listing C.32: ServerManager.java

package serverapp;

import com.sun.media.jsdt.*;
import java.io.*;
import java.util.HashMap;
import peerview.misc.*;
import java.util.Collection;
import java.util.Iterator;
import javax.swing.tree.*;
import java.util.Date;
import com.sun.media.jsdt.event.*;

/**
 * This is the main control class for the PeerView server application. It contains methods and control
 * structures shared
 * by several server components.
 * Creation date: (27-06-00 20:17:36)
 * @author:
 */
public class ServerManager
{
  private ServerInfo serverInfo = null;

  public class DocNameAndSize implements Serializable
  {
    public String docName = null;
    public long size = 0;

    public DocNameAndSize(String docNameArg, long sizeArg)
    {
      docName = docNameArg;
      size = sizeArg;
    }

    public String getDocName()
    {
      return docName;
    }

    public void setDocName(String newDocName)
    {
      docName = newDocName;
    }

    public long getSize()
    {
      return size;
    }
  }
public void setSize(long newSize)
{
    size = newSize;
}
);

public class GroupListener extends SessionAdaptor
{
    public String workGroupID = null;
    public GroupListener(String workGroupIdArg)
    {
        workGroupID = workGroupIdArg;
    }

    public void sessionJoined(SessionEvent se)
    {
        clientJoinedGroup(se.getClientName(), (WorkGroup) getServerInfo().getGroups().get(workGroupID));
    }

    public void sessionLeft(SessionEvent se)
    {
        try
        {       
            for (int i = 0; i < se.getSession().listClientNames().length; i++)
            {       
                System.out.println(se.getSession().listClientNames()[i]);
            }
        } catch (Exception E) {
        
        }
        clientLeftGroup(se.getClientName(), (WorkGroup) getServerInfo().getGroups().get(workGroupID));
    }
}

public class DefaultClient implements com.sun.media.jsdt.Client
{
    String name;

    public DefaultClient(String nameArg)
    {
        name = nameArg;
    }

    public Object authenticate(com.sun.media.jsdt.AuthenticationInfo ai)
    {
        return null;
    }

    public String getName()
    {
        return name;
    }
};

public class GroupManagementServer implements Client, ChannelConsumer
{
    String name;

    public GroupManagementServer()
    {
    }

    public String getName()
    {
    }
return name;
}

public void setName(String newName)
{
    name = newName;
}

public Object authenticate(AuthenticationInfo ai)
{
    return null;
}

public synchronized void dataReceived(Data data)
{
    DataPackage incomingPackage = null;
    try
    {
        incomingPackage = (DataPackage) data.getDataAsObject();
    }
    catch (Exception E)
    {
        System.err.println(E.toString());
        System.exit(0);
    }
    System.out.println("Server_dfid:" + incomingPackage.getProperties().getProperty(DataPackage.getDESCRIPTION()));
    // Determine the type of incoming package
    if (incomingPackage.getProperties().getProperty(DataPackage.getDESCRIPTION()).
        equalsIgnoreCase(ServerConstants.getREQUEST_ADD_GROUP_DIRECTORY()))
    {
        sendGroupDirectory(data.getSenderName());
    }
    else if (incomingPackage.getProperties().getProperty(DataPackage.getDESCRIPTION()).
        equalsIgnoreCase(ServerConstants.getREQUEST_ADD_GROUP()))
    {
        addGroup((peerviewmisc.WorkGroup) incomingPackage.getContents());
    }
    else if (incomingPackage.getProperties().getProperty(DataPackage.getDESCRIPTION()).
        equalsIgnoreCase(ServerConstants.getREQUEST_ADDED_DOCUMENTS()))
    {
        createDiscussionForum((String) incomingPackage.getProperties().getProperty(DataPackage.getAUTHOR_NAME()),
                                (String) incomingPackage.getContents(), data.getSenderName());
    }
    else if (incomingPackage.getProperties().getProperty(DataPackage.getDESCRIPTION()).
        equalsIgnoreCase(ServerConstants.getREQUEST_TOPIC_TREE()))
    {
        sendTopicTree((String) incomingPackage.getContents(), data.getSenderName());
    }
    else if (incomingPackage.getProperties().getProperty(DataPackage.getDESCRIPTION()).
        equalsIgnoreCase(ServerConstants.getREQUEST_MESSAGE()))
    {
        sendMessage((String) incomingPackage.getProperties().getProperty(DataPackage.getMessageID( ) ), data.getSenderName());
    }
    else if (incomingPackage.getProperties().getProperty(DataPackage.getDESCRIPTION()).
        equalsIgnoreCase(ServerConstants.getCLIENT_ADDED_MESSAGE()))
    {
        addMessage((Message) incomingPackage.getContents());
    }
}
else if (incomingPackage.getProperties().get("DataPackage.DESCRIPTION").equalsIgnoreCase(ServerConstants.get("TREE_UPDATE")))
    {
        updateTopicTree((String) incomingPackage.getProperties().get("DataPackage.DOCUMENT_PATH"), (DefaultTreeModel) incomingPackage.getContents());
    }

else if (incomingPackage.getProperties().get("DataPackage.DESCRIPTION").equalsIgnoreCase(ServerConstants.get("DELETE_MESSAGE")))
    {
        deleteMessage((String) incomingPackage.getContents());
    }

else if (incomingPackage.getProperties().get("DataPackage.DESCRIPTION").equalsIgnoreCase(ServerConstants.get("WORKGROUP_UPDATE_FROM_CLIENT")))
    {
        updateWorkGroup((WorkGroup) incomingPackage.getContents());
    }

else if (incomingPackage.getProperties().get("DataPackage.DESCRIPTION").equalsIgnoreCase(ServerConstants.get("NEW_DOCUMENTS")))
    {
        clientAddedDocuments(data.getSenderId(), incomingPackage.getProperties().get("DataPackage.AUTOR_NAME"), incomingPackage.getProperties().get("DataPackage.WORKGROUP_ID"), (java.util.Vector) incomingPackage.getContents());
        broadcastGroupDirectory();
    }

else if (incomingPackage.getProperties().get("DataPackage.DESCRIPTION").equalsIgnoreCase(ServerConstants.get("REMOVE_DOCUMENTS")))
    {
        clientRemovedDocuments(data.getSenderId(), incomingPackage.getProperties().get("DataPackage.WORKGROUP_ID"), (java.util.Vector) incomingPackage.getContents());
        broadcastGroupDirectory();
    }

else if (incomingPackage.getProperties().get("DataPackage.DESCRIPTION").equalsIgnoreCase(ServerConstants.get("CLIENT_LEFT")))
    {
        clientLeft(data.getSenderId());
    }

private PeerViewServer defaultServer = new PeerViewServer(ServerConstants.get("DEFAULT_SERVER_NAME"));
private GroupManagementServer groupManagementServer = null;
private ServerApplication serverApplication = null;

/**
 * ServerManager constructor comment.
 */
public ServerManager()
    {
        super();
        initialize();
    }

/**
 * Creation date: (07-06-00 02:16:02)
 * &param group peeviewmisc.WorkGroup

395
public void activateGroup(WorkGroup workGroup) {
    Session session = null;
    URLString url = null;
    GroupManagementServer newServer = new GroupManagementServer();
    newServer.setName( ServerInfo.getPreferences().getProperty( ServerConstants.getServer_NAME() ) );
    try
    {
        boolean created = false;
        /*
         * Cycle through port numbers until an unoccupied one is found and use it for creating a new
         * session
         * while ( created == false )
         * try
         * url = URLString.createSessionURL( ServerInfo.getPreferences().getProperty( ServerConstants.getServer_NAME() ),
         *     Integer.parseInt( ServerInfo.getPreferences().getProperty( ServerConstants.getServer_PORT() ) ),
         *     ServerInfo.getPreferences().getProperty( ServerConstants.getConnection_TYPE() ) ),
         *     workGroup.getWorkGroupId() );
         *     System.out.println( workGroup.getWorkGroupId() );
         * session = SessionFactory.createSession( newServer, url, true );
         * created = true;
         * session.addSessionListener( new GroupListener( workGroup.getWorkGroupId() ) );
         * /
         * catch ( PortInUseException PUE )
         * {
         *     serverInfo.setMostRecentPortNumber( serverInfo.getMostRecentPortNumber() + 1 );
         * }
         */
        Channel groupChannel = session.createChannel( newServer, workGroup.getWorkGroupChannelID(), true, true, true );
        groupChannel.addConsumer( newServer, newServer );
        workGroup.setSessionURL( url );
        }
        catch ( JSDEException JSDTE )
        {
            handleJSDEException( JSDTE );
        }
    }
    } /*
    * Activate the groups read from disk file by creating and registering the appropriate sessions and
    * channels.
    * Creation date: (06-07-00 23:33:46)
    */
    public void activateGroups()
    {
        Collection values = serverInfo.getGroups().values();
        Iterator iterator = values.iterator();
        while ( iterator.hasNext() )
        {
            activateGroup( (WorkGroup) iterator.next() );
        }
    } /*
    * Insert the method's description here.
    * Creation date: (06-07-00 19:06:53)
    * @param workGroup peerviewmisc.WorkGroup
    */
public void addGroup(overview.misc.WorkGroup workGroup) {
    serverInfo.getGroups().put(workGroup.getWorkGroupID(), workGroup);
    activateGroup(workGroup);
    broadcastGroupDirectory();
    serverApplication.updateTable();
}
/**
 * Insert the method's description here.
 * @param groupName java.lang.String
 */
public void addMessage(Message newMessage) {
    serverInfo.getMessage().put(newMessage.getMessageID(), newMessage);
}
/**
 * Insert the method's description here.
 * @param newGroupName java.lang.String
 * @param newGroupDescription java.util.Vector
 */
public void broadcastDocumentRemovals(String groupName, WorkGroup workGroup, java.util.Vector documents) {
    DataPackage outgoingPackage = new DataPackage();
    outgoingPackage.setContents(documents);
    outgoingPackage.getProperties().setProperty(DataPackage.DESCRIPTION(), ServerConstants.
        getGROUP_DOC_REMOVALS());
    outgoingPackage.getProperties().setProperty(DataPackage.WORKGROUP_ID(), workGroup.getWorkGroupID());
    outgoingPackage.getProperties().setProperty(DataPackage.CLIENT_NAME(), groupName);
    outgoingPackage.getProperties().setProperty(DataPackage.AUTHOR_NAME(), ServerInfo.getServerName());
    try {
        String url = workGroup.getURL();
        Session session = SessionFactory.createSession(getGroupManagementServer(), url, false);
        Channel groupChannel = session.createChannel(getGroupManagementServer(), workGroup.
            getWorkGroupChannelID(), true, true, false);
        Data data = new Data(outgoingPackage);
        groupChannel.sendToOthers(getGroupManagementServer(), data);
    } catch (JSDEException JSDDE) {
        handleJSDEException(JSDDE);
    }
}
/**
 * Insert the method's description here.
 * @param groupName java.lang.String
 */
public void broadcastGroupDirectory() {
    DataPackage outgoingPackage = new DataPackage();
    outgoingPackage.setContents(serverInfo.getGroups());
    outgoingPackage.getProperties().setProperty(DataPackage.DESCRIPTION(), ServerConstants.
        getGROUP_DIRECTORY());
    try {
        if (serverInfo.getGroupManagementChannel() != null)
            serverInfo.getGroupManagementChannel().sendToOthers(getGroupManagementServer(),
new Data( outgoingPackage );

```java
}
}

```catch ( JSDEException JSDE )
{
	handleJSDEException( JSDE );
}

```java
if ( getServerApplication() != null )
{
	sendToOthers( GroupManagementServer(),
		new Data( outgoingPackage ));
}

```catch ( JSDEException JSDE )
{
	handleJSDEException( JSDE );
}

```java
/**
 * Insert the method’s description here.
 * Creation date: (27-08-00 20:58:01)
 */
public void broadcastTopicTree( String documentName, DefaultTreeModel treeUpdate)
{
	DataPackage outgoingPackage = new DataPackage();
	outingPackage.getProperties().setProperty( DataPackage.DESCRIPTION(), ServerConstants.

tOPIC_TREE_UPDATE());
	outingPackage.getProperties().setProperty( DataPackage.DOC_PATH(), documentName );

try
{
	sendToOthers( GroupManagementServer(),
		new Data( outgoingPackage ));
}

```java
catch ( JSDEException JSDE )
{
	handleJSDEException( JSDE );
}

```java
/**
 * Insert the method’s description here.
 * Creation date: (19-09-00 18:40:29)
 */
public void clientAddedDocuments(String senderName, String authorName, String groupID, java.util.Vector

documents)
{
	WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( groupID );

```java
for ( int i = 0 ; i < documents.size(); i++ )
{
	CompressedDocumentPackage docPackage = (CompressedDocumentPackage) documents.get( i );
	Pair docNameAndSize = new Pair( docPackage.getPath(), String.valueOf( docPackage.
		getUncompressedLength() ));
	(documents.get(groupToBeUpdated.getDocuments().get( senderName )).add(docNameAndSize);

groupToBeUpdated.getTotalDataVolume( groupToBeUpdated.getTotalDataVolume() + docPackage.

tgetUncompressedLength() );

groupToBeUpdated.setNumberOfDocuments( groupToBeUpdated.getNumberOfDocuments() + 1 );
	sendToOthers( GroupManagementServer(),
		new Data( outgoingPackage ));

tServerInfo().getGroups().put( groupID, groupToBeUpdated );
}

```java
/**
 * Update the data structure holding the groups and submit it to all clients connected to the group
 * management channel.
 * Creation date: (19-09-00 17:55:14)
 */
```
public void clientJoinedGroup(String clientName, WorkGroup workGroup)
{
    WorkGroup groupToUpdate = (WorkGroup) getServerInfo().getGroups().get( workGroup.getGroupId() );

groupToUpdate.getParticipants().add( clientName );
groupToUpdate.getDocuments().put( clientName, new java.util.Vector() );
getServerInfo().getGroups().put( groupToUpdate.getGroupId(), groupToUpdate );
broadcastGroupDirectory();
}

/**
 * Forcibly expel client from group management session if it has not already left. This to ensure
 * consistency when clients
 * terminate abruptly, i.e. do not exit properly.
 * Creation date: (01-11-00 23:46:19)
 *
 * @param clientId java.lang.String
 */
public void clientLeft(String clientId)
{
    try
    {
        for ( int i = 0 ; i < getServerInfo().getGroupManagementSession().listClientNames().length; i++ )
        {
            if ( getServerInfo().getGroupManagementSession().listClientNames()[i].equalsIgnoreCase( clientId ) )
            {
                //System.out.println( "Found client : " + clientId );
                DefaultClient tempClient = new DefaultClient( clientId );
                getServerInfo().getGroupManagementSession().leave( tempClient );
                break;
            }
        }
    }
    catch ( JSUTException JSUTE )
    {
        handleJSUTException( JSUTE );
    }

    /**
     * This method is triggered when a client leaves a group. The server updates the group's entry in the
     * group directory
     * and resubmits it to other clients. Client that partake in the group that the client left are notified
     * of which
     * documents to remove from their panoramas.
     * Creation date: (19-09-00 17:55:46)
     * @param clientId java.lang.String
     * @param workGroup peerviewmisc.WorkGroup
     */
public void clientLeftGroup(String clientId, WorkGroup workGroup)
{
    WorkGroup groupToUpdate = (WorkGroup) getServerInfo().getGroups().get( workGroup.getGroupId() );

    if ( groupToUpdate.getParticipants().contains( clientId ) )
    {
        java.util.Vector documentNames = new java.util.Vector();
        java.util.Vector documents = (java.util.Vector) groupToUpdate.getDocuments().get( clientId );
        Iterator docIterator = documents.iterator();
        while ( docIterator.hasNext() )
        {
            Pair docNameAndSize = (Pair) docIterator.next();
            groupToUpdate.setTotalDataVolume( groupToUpdate.getTotalDataVolume() - Integer.parseInt( docNameAndSize.getValue() ) );
            groupToUpdate.setNumberOfDocuments( groupToUpdate.getNumberOfDocuments() - 1 );
            documentNames.add( docNameAndSize.getKey() );
        }
    }
broadcastDocumentRemovals( clientName, groupToBeUpdated, documentNames );

groupToBeUpdated.getParticipants().remove( clientName );
groupToBeUpdated.getDocuments().remove( clientName );
getServerInfo().getGroups().put( groupToBeUpdated.getGroupId(), groupToBeUpdated );
broadcastGroupDirectory();

/**
 * This method is triggered when a client removes one or more documents from his or her panorama. The server updates the group that the client belongs to and resubmits the group directory to all clients.
 * Creation date: (19-09-00 19:08:00)
 * @param files java.util.HashSet
 */
public void clientRemovedDocuments(String senderName, String groupId, java.util.Vector filePaths)
{
    WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( groupId );
    Iterator iterator = filePaths.iterator();
    while ( iterator.hasNext() )
    {
        String path = (String) iterator.next();
        java.util.Vector documents = (java.util.Vector) groupToBeUpdated.getDocuments().get( senderName );
        Iterator docIterator = documents.iterator();
        Pair docNameAndSize = null;
        while ( docIterator.hasNext() )
        {
            docNameAndSize = (Pair) docIterator.next();
            if ( docNameAndSize.getKey().equals( path ) )
            {
                groupToBeUpdated.setTotalDataVolume( groupToBeUpdated.getTotalDataVolume() - Integer.parseInt( docNameAndSize.getValue() ) );
                groupToBeUpdated.setNumberOfDocuments( groupToBeUpdated.getNumberOfDocuments() - 1 );
                documents.remove( docNameAndSize );
                break;
            }
        }
        getServerInfo().getGroups().put( groupToBeUpdated.getGroupId(), groupToBeUpdated );
    }

    /**
     * Perform miscellaneous initialization operations.
     * Creation date: (10-10-00 23:12:57)
     */
    public void configure()
    {
        startRegistry();
        createGroupManagement();
        readMessages();
        readGroups();
        readDiscussions();
    }

    /**
     * Insert the method’s description here.
     * Creation date: (29-06-00 08:57:23)
     * @return java.lang.String
     * @param sessionName java.lang.String
     */
    public String createClientName(String sessionName)
    {
        return ServerInfo.getPreferences().getProperty( ServerConstants.getServerName() )+ sessionName;
    }
* Create a new discussion forum and initialize it to the default structure, namely a root node and system notification
* message.
* Creation date: (09-07-00 14:55:18)
* @param newDocument java.io.File
*/
public void createDiscussionForum(String authorName, String newDocument, String senderName)
{
    String fqDocumentName = ServerConstants.createFullyQualifiedDocumentName( senderName, newDocument );
    // if a discussion forum for this document does not already exist then create one
    if ( getServerInfo().getDiscussions().containsKey( fqDocumentName ) == false )
    {
        Message message = new Message();
        message.setAuthor( ServerInfo.getPreferences().getProperty( ServerConstants.getServer_NAME() ) );
        message.setCreationDate( new java.util.Date() );
        message.setCreator( ServerInfo.getPreferences().getProperty( ServerConstants.getServer_NAME() ) );
        message.setTitle( ServerConstants.getRoot_NODE_MESSAGE_TITLE() );
        message.setContents( ServerConstants.getRoot_NODE_MESSAGE_CONTENTS( newDocument, authorName, message.getCreationDate() ) );
        message.setDocument( newDocument );
        getServerInfo().getMessages().put( message.getMessageID(), message );
        // udekif med konstant
        DefaultMutableTreeNode innerNode = new DefaultMutableTreeNode( newDocument + ", authored by: " + authorName, true );
        DefaultMutableTreeNode treeNode = new DefaultMutableTreeNode( message.getMessageID(), false );
        innerNode.add( treeNode );
        getServerInfo().getDiscussions().put( fqDocumentName, new DefaultTreeNode( innerNode ) );
        // System.out.println( "New topic tree: " + fqDocumentName );
    }
}
/**
* Setup the group management session and channel and activate it.
* Creation date: (03-07-00 11:29:03)
*/
public void createGroupManagement()
{
    Session session = null;
    Channel groupRequestChannel = null;
    try {
        String url = URLString.createSessionURL( (String)ServerInfo.getPreferences().getProperty( ServerConstants.getServer_NAME() ),
            ServerInfo.getPreferences().getProperty( ServerConstants.getServer_PORT() ),
            (String)ServerInfo.getPreferences().getProperty( ServerConstants.getGROUP_MANAGEMENT_SESSION_NAME() ) );
        session = SessionFactory.createSession( getGroupManagementServer(), url, true);
        groupRequestChannel = session.createChannel( getGroupManagementServer(),
            ServerConstants.getGROUP_CHANNEL_NAME(),
            true, true, true);
        groupRequestChannel.addConsumer( getGroupManagementServer(), getGroupManagementServer() );
        DataPackage pack = new DataPackage();
        pack.setContents( null );
        groupRequestChannel.sendToAll( getGroupManagementServer(), new Data( pack ) );
        // System.out.println( url.toString() );
    }
    catch (JSDEException jsde)
    {
        handleJSDEException( JSDE );
    }
    getServerInfo().setGroupManagementSession( session );
    getServerInfo().setGroupManagementChannel( groupRequestChannel );
}
/**
 * Shut down a group.
 * Creation date: (27-07-00 00:36:12)
 * @param workGroup peerviewmisc.WorkGroup
 */
public void deactivateGroup(WorkGroup workGroup)
{
    try
    {
        URLString url = workGroup.getSessionURL();
        Session session = SessionFactory.createSession( getGroupManagementServer(), url, false );
        Channel channel = session.createChannel( getGroupManagementServer(), workGroup, getWorkGroupChannelID(), true, true, false );
        channel.destroy( getGroupManagementServer() );
        session.destroy( getGroupManagementServer() );
    }
    catch ( JSDTEException JSDTE )
    {
        handleJSDTEException( JSDTE );
    }
}
/**
 * Insert the method’s description here.
 * Creation date: (20-10-00 15:06:46)
 */
public void deactivateGroups()
{
    Collection values = serverInfo.getGroups().values();
    Iterator iterator = values.iterator();
    while ( iterator.hasNext() )
    {
        deactivateGroup( (WorkGroup) iterator.next() );
    }
}
/**
 * Insert the method’s description here.
 * Creation date: (12-07-00 12:14:10)
 * @param messageId java.lang.String
 */
public void deleteMessage(String messageId)
{
    serverInfo.getMessages().remove( messageId );
}
/**
 * Insert the method’s description here.
 * Creation date: (20-10-00 15:15:34)
 */
public void destroyGroupManagement()
{
    try
    {
        getServerInfo().getGroupManagementChannel().destroy( getGroupManagementServer() );
        getServerInfo().getGroupManagementSession().destroy( getGroupManagementServer() );
    }
    catch ( JSDTEException JSDTE )
    {
        handleJSDTEException( JSDTE );
    }
}
/**
 * Insert the method’s description here.
 * Creation date: (04-07-00 16:11:19)
 */
protected void finalize()
{  
	/*
	* Insert the method's description here.
	* Creation date: (30-06-00 11:17:21)
	* @return serverapp.PeerViewServer
	*/
654 public PeerViewServer getDefaultServer() {
655     return defaultServer;
656 }

	/*
	* Insert the method's description here.
	* Creation date: (03-07-00 12:23:42)
	* @return serverapp.GroupManagementServer
	*/
662 public GroupManagementServer getGroupManagementServer() {
664     if (groupManagementServer == null)
666         {
668             groupManagementServer = new GroupManagementServer();
670             groupManagementServer.setName(ServerInfo.getPreferences().getProperty(ServerConstants.getServer_NAME()));
672         }
674     return groupManagementServer;
676 }

	/*
	* Insert the method's description here.
	* Creation date: (02-09-00 08:46:31)
	* @return serverapp.ServerApplication
	*/
684 public ServerApplication getServerApplication() {
686     if (serverInfo == null)
688         {
690             serverInfo = new ServerInfo();
692     }
694     return serverInfo;
696 }

	/*
	* Handle the different subclasses of JSDException, i.e. the different types of exception that JSDE
	* actions may throw.
	* by issuing appropriate messages and taking corrective or conclusive action.
	* Creation date: (08-08-00 16:34:28)
	* @param exception com.sun.media.jsd.JSDEException
	*/
698 public void handleJSDEException(JSDEException exception)
699 {
700     String message = (String) ServerConstants.getJSDE_EXCEPTION_MESSAGES_TABLE().get( exception.getClassName(), exception.toString() );
702     System.err.println( message );
704 }

	/*
	* Insert the method's description here.
	* Creation date: (01-07-00 01:14:14)
	*/
708 public void initialize() {
/**
 * Read the discussion "database" (a text file, essentially) from disk and initialize the corresponding data structure.
 * Creation date: (24-08-00 08:45:43)
 */

public void readDiscussions()
{
    try
    {
        FileInputStream fisstream = new FileInputStream( ServerConstants.getDISCUSSIONS_FILE_NAME() );
        ObjectInputStream oistream = new ObjectInputStream( fisstream );
        Hashtable discussionsTable = ( Hashtable ) oistream.readObject();
        if ( discussionsTable != null )
        {
            getServerInfo().setDiscussions( discussionsTable );
        } else
        {
            getServerInfo().setDiscussions( new Hashtable() );
        }
        // System.out.println( "Discussionstablesize: " + discussionsTable.size() );
        oistream.close();
    }
    catch ( Exception e )
    {
        System.err.println( ServerConstants.getCOULD_NOT_READ_MESSAGES() );
    }
}

/**
 * Read the groups directory from persistent storage, i.e. disk file
 * Creation date: (04-07-00 15:03:54)
 */

public void readGroups()
{
    try
    {
        FileInputStream fis = null;
        ObjectInputStream ois = null;
        fis = new FileInputStream( ServerConstants.getGROUPS_FILE_NAME() );
        ois = new ObjectInputStream( fis );
        // while ( ois.available() > 0 )
        // {
        //     // the below simulates a loop by trying a readObject operation until an exception is thrown. Not kosher, but
        //     // the only way that seems to work at the moment.
        //     while (true)
        //     {
        //         try
        //         {
        //             WorkGroup workGroup = ( WorkGroup ) ois.readObject();
        //             workGroup.setNumberOfDocuments( 0 );
        //             workGroup.setSizeDataVolume( 0 );
        //             workGroup.setSizeTables( new java.util.Hashtable() );
        //             workGroup.setSizeParticipants( new java.util.HashSet() );
        //             // System.out.println( "WorkGroup: " + workGroup.getWorkGroupId() );
        //             getServerInfo().getGroups().put( workGroup.getWorkGroupId(), workGroup);
        //             activateGroup( workGroup );
        //         }
        //     }
        //     catch ( Exception E )
        //     {
        //         break;
        //     }
        // }
774     // }
776     ois.close();
778     broadcastGroupDirectory();
780     }
782     catch ( IOException e )
784     {  
786     System.err.println( ServerConstants.getCouldNotReadGroups() );
788     }
789     /**
790     * Insert the method's description here.
791     * Creation date: (14-08-00 23:03:40)
792     */
793     public void readMessages()
794     { try
795     { FileInputStream fistream = new FileInputStream( ServerConstants.getMessages_FILE_NAME() );
796     ObjectInputStream oistream = new ObjectInputStream( fistream );
797     Hasatable messageTable = ( Hasatable ) oistream.readObject();
798     if ( messageTable != null )
799     {  
800     getServerInfo().setMessages( messageTable );
801     } else
802     {  
803     getServerInfo().setMessages( new Hasatable() );
804     // System.out.println( "MessageTable: " + messageTable.size() );
805     oistream.close();
806     }
808     catch ( Exception e )
809     { System.err.println( ServerConstants.getCouldNotRead_MESSAGES() );
810     }
811     }
812     /**
813     * Insert the method's description here.
814     * Creation date: (27-07-00 00:31:10)
815     * @param workGroup peerviewmisc.WorkGroup
816     */
817     public void removeWorkGroup(WorkGroup workGroup)
818     {  
819     System.out.println( "I_removeWorkgroup" );
820     deactiveGroup( workGroup );
821     System.out.println( "after_deactiveGroup" );
822     getServerInfo().getGroups().remove( workGroup.getWorkGroupID() );
823     System.out.println( "about_so.broadcast_group_directory.from_removeWorkgroup" );
824     broadcastGroupDirectory();
825     }
826     /**
827     * Insert the method's description here.
828     * Creation date: (07-07-00 14:53:16)
829     * @param recipientName java.lang.String
830     */
831     public void sendGroupDirectory(String recipientName)
832     { try
833     DataPackage outgoingPackage = new DataPackage();
834     outgoingPackage.setContents( serverInfo.getGroups() );
835     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
836     getGROUP_DIRECTORY() );
837     try
838     { 
839     }
{ serverInfo.getChannel().sendToClient( groupManagementServer, 
    new Data( outgoingPackage ));
System.out.println( "Sent :group:directory" );
} catch ( JSDEException JSDE )
{ handleJSDEException( JSDE );
}

/**
 * Insert the method's description here.
 * Creation date: (10-07-00 02:20:49)
 * @param messageID java.lang.String
 * @param recipientName java.lang.String
 */
public void sendMessage(String messageID, String recipientName)
{
    DataPackage outgoingPackage = new DataPackage();
    // System.out.println( "Sent message: " + messageID );
    outgoingPackage.setContents( serverInfo.getMessages().get( messageID ));
    outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.getMessage 
    () );
    try
    {
        serverInfo.getChannel().sendToClient( groupManagementServer, recipientName, new 
            Data( outgoingPackage ));
    } catch ( JSDEException JSDE )
    {
        handleJSDEException( JSDE );
    }
}

/**
 * Precondition: documentName is the fully qualified name of a document as specified in method
 * SharedConstants.createFullyQualifiedDocumentName
 * Creation date: (09-07-00 22:20:13)
 * @param documentName java.lang.String
 * @param recipientName java.lang.String
 */
public void sendTopicTree(String documentName, String recipientName)
{
    DataPackage outgoingPackage = new DataPackage();
    // System.out.println( "Sending topic tree: " + documentName );
    outgoingPackage.setContents( getServerInfo().getDiscussions().get( documentName ));
    outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
        getTOPICTREE() );
    outgoingPackage.getProperties().setProperty( DataPackage.getDocument(), documentName );
    try
    {
        serverInfo.getChannel().sendToClient( groupManagementServer, recipientName, new 
            Data( outgoingPackage ));
    } catch ( JSDEException JSDE )
    {
        handleJSDEException( JSDE );
    }

    */
    * Insert the method's description here.
    * Creation date: (30-06-00 11:17:21)

406
* @param newDefaultServer serverapp.PeerViewServer
*/
public void setDefaultServer(PeerViewServer newDefaultServer) {
    defaultServer = newDefaultServer;
}
/**
 * Insert the method's description here.
 * Creation date: (03-07-00 12:23:42)
 * @param newGroupManagementServer serverapp.GroupManagementServer
*/
public void setGroupManagementServer(GroupManagementServer newGroupManagementServer) {
    groupManagementServer = newGroupManagementServer;
}
/**
 * Insert the method's description here.
 * Creation date: (02-09-00 08:46:31)
 * @param newServerApplication serverapp.ServerApplication
*/
public void setServerApplication(ServerApplication newServerApplication) {
    serverApplication = newServerApplication;
}
/**
 * Insert the method's description here.
 * Creation date: (28-06-00 22:59:39)
 * @param newServerInfo serverapp.ServerInfo
*/
public void setServerInfo(ServerInfo newServerInfo) {
    serverInfo = newServerInfo;
}
/**
 * Insert the method's description here.
 * Creation date: (27-06-00 20:18:18)
*/
public void startRegistry()
{
    try
    {
        RegistryFactory.registryPort = Integer.parseInt( ServerInfo.getPreferences().getProperty( ServerConstants.getConnection() ));
        com.sun.media.jdsdt.RegistryFactory.startRegistry( (String) ServerInfo.getPreferences().getProperty( ServerConstants.getConnection() ));
        // Integer.parseInt( ServerInfo.getPreferences().getProperty( ServerConstants.getConnection() ));
        if ( RegistryFactory.registryExists( ServerInfo.getPreferences().getProperty( ServerConstants.getConnection() ) ) )
        {
            System.out.println( "Successfully started registry" );
        }
    }
    catch (com.sun.media.jsdt.NoRegistryException nre)
    {
        // System.out.println( (String) ServerInfo.getPreferences().getProperty( ServerConstants.getConnection() ));
        // System.out.println( ServerInfo.getPreferences().getProperty( ServerConstants.getConnection() ));
        System.err.println( nre.toString() );
        System.exit(0);
    }
    catch (com.sun.media.jsdt.RegistryExistsException ree)
    {
        System.err.println( ree.toString() );
        System.exit(0);
    }
}
public void stopRegistry()
{
    try
    {
        RegistryFactory.stopRegistry( ServerInfo.getPreferences().getProperty( ServerConstants.getCONNECTION_TYPE() ),
                Integer.parseInt( ServerInfo.getPreferences().getProperty( ServerConstants.getSERVER_PORT() ) ) );
    }
    catch ( JSDException JSDTE )
    {
        handleJSDException( JSDTE );
    }
}

public void terminate()
{
    deactivateGroups();
    writeDiscussions();
    writeMessages();
    writeGroups();
    getServerInfo().writePreferences();
    destroyGroupManagement();
    try
    {
        System.runFinalization();
        Runtime.getRuntime().exit( 0 );
    }
    catch ( Exception E )
    {
        System.out.println( ServerConstants.getCOULD_NOT_EXIT_PROPORLY() );
    }
}

public void updateTopicTree(String documentName, DefaultTreeModel newTree)
{
    getServerInfo().getDiscussions().put( documentName, newTree );
    broadcastTopicTree( documentName, newTree );
}

public void updateWorkGroup(WorkGroup workGroup)
{
    WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( workGroup.getWorkGroupId() );

    groupToBeUpdated.setNumberOfDocuments( groupToBeUpdated.getNumberOfDocuments() + workGroup.getNumberOfDocuments() );
groupToBeUpdated.setTotalDataVolume( groupToBeUpdated.getTotalDataVolume() + workGroup.
  getTotalDataVolume() );

toStream = new FileOutput( ServerConstants.getDISCUSSIONS_FILE_NAME() );
  ObjectOutput oostream = new ObjectOutputStream( toStream );
  oostream.writeObject( getServerInfo().getDiscussions() );
  // System.out.println( "WroteDiscussions" );
  oostream.flush();
  oostream.close();
  }
}
catch ( Exception e )
  {
    System.err.println( ServerConstants.getCOULD_NOT_WRITE_MESSAGES() );
  }

/**
* Write registered groups to persistent storage, i.e. disk file
* Creation date: (04-07-00 15:03:41)
*/
public void writeGroups()
{
  try
  {
    FileOutput fos = new java.io.FileOutput( ServerConstants.getGROUPS_FILE_NAME() );
    ObjectOutput oos = new ObjectOutputStream( fos );
    Collection values = serverInfo.getGroups().values();
    Iterator iterator = values.iterator();

    while ( iterator.hasNext() )
    {
      workGroup = (workGroup) iterator.next();
      workGroup.setParticipants( null );
      workGroup.setDocuments( null );
      oos.writeObject( workGroup );
      /* oos.writeObject( workGroup.getGroupName() );
      oos.writeObject( workGroup.getMaximumParticipants() );
      oos.writeObject( workGroup.getDescription() );
      oos.writeObject( workGroup.getCreationDate().toString() );
      oos.writeObject( workGroup.getCreator() );
      */
      // System.out.println( "Wrote group" );
      oos.flush();
      oos.close();
    }
    catch ( IOException IOE )
    {
      System.err.println( ServerConstants.getCOULD_NOT_WRITE_GROUPS() );
    }
  }
*/

/**
* Insert the method’s description here.
* Creation date: (14-08-00 23:20:34)
*/
```java
public void writeMessages()
{
    try
    {
        FileOutputStream fosstream = new FileOutputStream( ServerConstants.getMESSAGES_FILE_NAME() );
        ObjectOutputStream oostream = new ObjectOutputStream( fosstream );
        oostream.writeObject( getServerInfo().getMessages() );
        // System.out.println( "wroteMessages" );
        oostream.flush();
        oostream.close();
    }
    catch ( Exception e )
    {
        System.err.println( ServerConstants.getGUILD_NOT_WRITE_MESSAGES() );
    }
}

Listing C.33: ServerSetup.java

package serverapp;

import java.awt.*;
import java.awt.border.*;
import java.awt.event.*;
import java.util.Properties;
import java.io.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

/**
 * The server setup dialog box.
 * Creation date: (28-05-00 20:58:57)
 * @author ...
 */
public class ServerSetup extends JDialog
{
    public class OKButtonHandler implements ActionListener
    {
        public void actionPerformed( ActionEvent ae )
        {
            submitAndClose();
        }
    };

    public class CancelButtonHandler implements ActionListener
    {
        public void actionPerformed( ActionEvent ae )
        {
            cancelAndClose();
        }
    };

    private JButton ivjCancelButton = null;
    private JPanel ivjDialogContentPane = null;
    private JPanel ivjDialog1 = null;
    private java.awt.FlowLayout ivjDialogContentFlowLayout = null;
    private JButton ivjOKButton = null;
    private JComboBox ivjProtocolComboBox = null;
    private JLabel ivjProtocolLabel = null;
    private JTextField ivjServerNameField = null;
    private JLabel ivjServerNameLabel = null;
    private JLabel ivjServerPortLabel = null;
    private JTextField ivjServerPortField = null;
    private JPanel ivjTopPanel = null;
```

410
/**
 * ServerSetup constructor comment.
 */
public ServerSetup() {
    super();
    initialize();
}
/**
 * ServerSetup constructor comment.
 */
public ServerSetup(java.awt.Dialog owner) {
    super(owner);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public ServerSetup(java.awt.Dialog owner, String title) {
    super(owner, title);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public ServerSetup(java.awt.Dialog owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Frame
 */
public ServerSetup(java.awt.Frame owner) {
    super(owner);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public ServerSetup(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public ServerSetup(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Frame
 */

* @param modal boolean

```java
public ServerSetup(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}
```

```
/**
 * Insert the method's description here.
 * Creation date: (06-08-00 20:54:49)
 */
public void cancelAndClose()
{
    dispose();
}
/**
 * Return the CancelButton property value.
 */
private javax.swing.JButton getCancelButton()
{
    if (ivjCancelButton == null) {
        try {
            ivjCancelButton = new javax.swing.JButton();
            ivjCancelButton.setName("CancelButton");
            ivjCancelButton.setText("Cancel");
            // user code begin {1}
            ivjCancelButton.addActionListener( new CancelButtonHandler() );
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
                // user code end
                handleException(ivjExc);
            }
        return ivjCancelButton;
    }
    /**
     * Return the JDialogContentPane property value.
     */
    private javax.swing.JPanel getJDialogContentPane()
    {
        if (ivjJDialogContentPane == null) {
            try {
                ivjJDialogContentPane = new javax.swing.JPanel();
                ivjDialogContentPane.setName("JDialogContentPane");
                ivjDialogContentPane.setLayout(new java.awt.BorderLayout());
                getJDialogContentPane().add(getJPanel1(), "South");
                getJDialogContentPane().add(getTopPanel(), "center");
                // user code begin {1}
                // user code end
                } catch (java.lang.Throwable ivjExc) {
                    // user code begin {2}
                    // user code end
                    handleException(ivjExc);
                }
        return ivjJDialogContentPane;
    }
    /**
     * Return the JPanel1 property value.
     */
    private javax.swing.JPanel getJPanel1()
    {
        if (ivjJPanel1 == null) {
            try {
```
ivjPanel1 = new javax.swing.JPanel();
ivjPanel1.setName("JPanel1");
ivjPanel1.setLayout(getJPanel1FlowLayout());
gejPanel().add(OKButton(), getOKButton().getName());
gejPanel1().add(cancelButton(), getCancelButton().getName());
// user code begin {1}
// user code end
}
} catch (java.lang.Throwable ivjExc) { // user code begin {2}
// user code end
handleException(ivjExc);
}
return ivjPanel1;

/**
 * Return the JPanel1FlowLayout property value.
 */
private java.awt.FlowLayout getJPanel1FlowLayout() {
java.awt.FlowLayout ivjPanel1FlowLayout = null;
try {
/* Create part */
ivjPanel1FlowLayout = new java.awt.FlowLayout();
ivjPanel1FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
} catch (java.lang.Throwable ivjExc) { // user code begin {2}
// user code end
handleException(ivjExc);
}
return ivjPanel1FlowLayout;

/**
 * Return the OKButton property value.
 */
private javax.swing.JButton getOKButton() {
if (ivjOKButton == null) {
try {
ivjOKButton = new javax.swing.JButton();
ivjOKButton.setName("OKButton");
ivjOKButton.setText("OK");
// user code begin {1}
ivjOKButton.addActionListener(new OKButtonHandler());
// user code end
} catch (java.lang.Throwable ivjExc) { // user code begin {2}
// user code end
handleException(ivjExc);
}
return ivjOKButton;
}
/**
 * Return the JComboBox property value.
 */
private javax.swing.JComboBox getProtocolComboBox() {
if (ivjProtocolComboBox == null) {
try {
ivjProtocolComboBox = new javax.swing.JComboBox();
ivjProtocolComboBox.setName("ProtocolComboBox");
// user code begin {1}
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin \{2
// user code end
handleException(ivjExc);
}
}
return ivjProtocolComboBox;
/**
* Return the ProtocolLabel property value.
* @return javax.swing.JLabel
*/
private javax.swing.JLabel getProtocolLabel() {
if (ivjProtocolLabel == null) {
try {
ivjProtocolLabel = new javax.swing.JLabel();
ivjProtocolLabel.setName("ProtocolLabel");
ivjProtocolLabel.setText("Connection:");
// user code begin \{1
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin \{2
// user code end
handleException(ivjExc);
}
return ivjProtocolLabel;
/**
* Return the ServerNameField property value.
* @return javax.swing.JTextField
*/
private javax.swing.JTextField getServerNameField() {
if (ivjServerNameField == null) {
try {
ivjServerNameField = new javax.swing.JTextField();
ivjServerNameField.setName("ServerNameField");
// user code begin \{1
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin \{2
// user code end
handleException(ivjExc);
}
return ivjServerNameField;
/**
* Return the ServerNameLabel property value.
* @return javax.swing.JLabel
*/
private javax.swing.JLabel getServerNameLabel() {
if (ivjServerNameLabel == null) {
try {
ivjServerNameLabel = new javax.swing.JLabel();
ivjServerNameLabel.setName("ServerNameLabel");
ivjServerNameLabel.setText("Server, name:");
// user code begin \{1
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin \{2
// user code end
handleException(ivjExc);
}
return ivjServerNameLabel;
} * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextField getServerPortField() {
  if (ivjServerPortField == null) {
    try {
      ivjServerPortField = new javax.swing.JTextField();
      ivjServerPortField.setName("ServerPortField");
      // user code begin (1)
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin (2)
      // user code end
      handleException(ivjExc);
    }
    return ivjServerPortField;
  }
  * WARNING: THIS METHOD WILL BE REGENERATED. */
  private javax.swing.JLabel getServerPortLabel() {
    if (ivjServerPortLabel == null) {
      try {
        ivjServerPortLabel = new javax.swing.JLabel();
        ivjServerPortLabel.setName("ServerPortLabel");
        ivjServerPortLabel.setText("Server.port:");
        // user code begin (1)
        // user code end
      } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
      }
      return ivjServerPortLabel;
    }
    * WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JPanel getTopPanel() {
      if (ivjTopPanel == null) {
        try {
          ivjTopPanel = new javax.swing.JPanel();
          ivjTopPanel.setName("TopPanel");
          ivjTopPanel.setLayout(new java.awt.GridBagLayout());
          java.awt.GridBagConstraints constraintsServerNameLabel = new java.awt.GridBagConstraints();
          constraintsServerNameLabel.gridx = 0;
          constraintsServerNameLabel.gridy = 0;
          constraintsServerNameLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
          constraintsServerNameLabel.weighty = 0.2;
          constraintsServerNameLabel.insets = new java.awt.Insets(6, 6, 6, 6);
          ivjTopPanel.add(getServerNameLabel(), constraintsServerNameLabel);
          java.awt.GridBagConstraints constraintsServerNameField = new java.awt.GridBagConstraints();
          constraintsServerNameField.gridx = 1;
          constraintsServerNameField.gridy = 0;
          constraintsServerNameField.fill = java.awt.GridBagConstraints.HORIZONTAL;
          }
constraintsServerNameField.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsServerNameField.weightx = 1.0;
constraintsServerNameField.weighty = 0.2;
constraintsServerNameField.insets = new java.awt.Insets(6, 6, 6);
getTopPanel().add(getServerNameField(), constraintsServerNameField);

java.awt.GridBagConstraints constraintsServerPortLabel = new java.awt.GridBagConstraints();
constraintsServerPortLabel.gridx = 0; constraintsServerPortLabel.gridy = 1;
constraintsServerPortLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsServerPortLabel.weighty = 0.2;
constraintsServerPortLabel.insets = new java.awt.Insets(6, 6, 6);
getTopPanel().add(getServerPortLabel(), constraintsServerPortLabel);

java.awt.GridBagConstraints constraintsServerPortField = new java.awt.GridBagConstraints();
constraintsServerPortField.gridx = 1; constraintsServerPortField.gridy = 1;
constraintsServerPortField.weightx = 1.0;
constraintsServerPortField.weighty = 0.2;
constraintsServerPortField.insets = new java.awt.Insets(6, 6, 6);
getTopPanel().add(getServerPortField(), constraintsServerPortField);

constraintsProtocolLabel.gridx = 0; constraintsProtocolLabel.gridy = 2;
constraintsProtocolLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsProtocolLabel.weighty = 0.2;
constraintsProtocolLabel.insets = new java.awt.Insets(6, 6, 6);
getTopPanel().add(getProtocolLabel(), constraintsProtocolLabel);

java.awt.GridBagConstraints constraintsProtocolComboBox = new java.awt.GridBagConstraints();
constraintsProtocolComboBox.gridx = 1; constraintsProtocolComboBox.gridy = 2;
constraintsProtocolComboBox.anchor = java.awt.GridBagConstraints.NORTHWEST;
constraintsProtocolComboBox.weightx = 1.0;
constraintsProtocolComboBox.weighty = 0.2;
constraintsProtocolComboBox.insets = new java.awt.Insets(6, 6, 6);
getTopPanel().add(getProtocolComboBox(), constraintsProtocolComboBox);

// user code begin {1}
Border matteBorder = BorderFactory.createMatteBorder(6, 6, 6, 6, ivjTopPanel.getBackground());
Border lineBorder = BorderFactory.createLineBorder(Color.gray);
Border compoundBorder = BorderFactory.createCompoundBorder(lineBorder, matteBorder);
Border secondCompoundBorder = BorderFactory.createCompoundBorder(matteBorder, compoundBorder);
ivjTopPanel.setBorder(secondCompoundBorder);
// user code end
}
} catch (java.lang.Throwable ivjExc) { // user code begin {2}
// user code end
handleException(ivjExc);
}
}

/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {

/* Uncomment the following lines to print uncaught exceptions to stdout */
// System.out.println("-------- UNCAUGHT EXCEPTION --------");
// exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {

416
try {
    // user code begin {1}
    // user code end
    setName("ServerSetup");
    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    setSize(468, 365);
    setTitle("Server_setup");
    setContentPane(getDialogContent());
    catch (java.lang.Throwable e) {
        handleException(e);
    }
    // user code begin {2}
    setLocationRelativeTo( getOwner() );
    // user code end
    */
    /**
     * main entrypoint - starts the part when it is run as an application
    */
    public static void main(java.lang.String[] args) {
        try {
            ServerSetup aServerSetup;
            aServerSetup = new ServerSetup();
            aServerSetup.setModal(true);
            aServerSetup.addWindowListener(new java.awt.event.WindowAdapter() {
                public void windowClosing(java.awt.event.WindowEvent e) {
                    System.exit(0);
                }
            });
            aServerSetup.setVisible(true);
        } catch (Throwable exception) {
            System.err.println("Exception occurred in main()")
                .printStackTrace(System.out);
        }
        /**
         * Insert the method’s description here.
         * Creation date: (06-08-00 20:33:03)
        */
    public void show() {
        getServerNameField().setText( ServerInfo.getSharedPreferences().getProperty( ServerConstants.getSERVER_NAME() ) );
        getServerPortField().setText( ServerInfo.getSharedPreferences().getProperty( ServerConstants.getSERVER_PORT() ) );
        if ( getProtocolComboBox().getItemCount() == 0 ) {
            for ( int i = 0; i < ServerConstants.getCONNECTION_TYPES().length; i++ ) {
                getProtocolComboBox().addItem( ServerConstants.getCONNECTION_TYPES()[i] );
            }
            super.show();
        }
        /**
         * Insert the method’s description here.
         * Creation date: (06-08-00 20:54:36)
        */
    public void submitAndClose() {
        ServerInfo.getSharedPreferences().setProperty( ServerConstants.getSERVER_NAME(), getServerNameField().getText() );
        ServerInfo.getSharedPreferences().setProperty( ServerConstants.getSERVER_PORT(), getServerPortField().getText() );
        ServerInfo.getSharedPreferences().setProperty( ServerConstants.getCONNECTION_TYPE(), (String) getProtocolComboBox().getSelectedItem() );
    }
}
package peerviewmisc;

import javax.swing.*;
import javax.swing.border.*;
import javax.swing.table.*;
import javax.swing.event.*;

/**<n
 * The PeerView about dialog box class
 * @author:
 */
public class AboutDialog extends JOptionPane {
    private JPanel ivjDialogContentPane = null;
    private JLabel ivjLogoLabel = null;
    private JLabel ivjRulerLabel = null;
    private JLabel ivjTextLabel = null;
    /**<n
     * AboutDialog constructor comment.
     */
    public AboutDialog() {
        super();
        initialize();
    }
    /**<n
     * AboutDialog constructor comment.
     * @param owner java.awt.Dialog
     */
    public AboutDialog(java.awt.Dialog owner) {
        super(owner);
    }
    /**<n
     * AboutDialog constructor comment.
     * @param owner java.awt.Dialog
     * @param title java.lang.String
     */
    public AboutDialog(java.awt.Dialog owner, String title) {
        super(owner, title);
    }
    /**<n
     * AboutDialog constructor comment.
     * @param owner java.awt.Dialog
     * @param title java.lang.String
     * @param modal boolean
     */
    public AboutDialog(java.awt.Dialog owner, String title, boolean modal) {
        super(owner, title, modal);
    }
    /**<n
     * AboutDialog constructor comment.
     * @param owner java.awt.Dialog
     * @param modal boolean
     */
    public AboutDialog(java.awt.Dialog owner, boolean modal) {
        super(owner, modal);
    }
    /**<n
     * AboutDialog constructor comment.
     */
};
/* @param owner java.awt.Frame */

public AboutDialog(java.awt.Frame owner) {
    super(owner);
}
/**
 * AboutDialog constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */

public AboutDialog(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
 * AboutDialog constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */

public AboutDialog(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * @param owner java.awt.Frame
 * @param modal boolean
 */

public AboutDialog(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}
/**
 * Return the JDialogContentPane property value.
 * @return javax.swing.JPanel
 */

/* WARNING: THIS METHOD WILL BE REGENERATED. */

private javax.swing.JPanel getJDialogContentPane() {
    if (ivjJDialogContentPane == null) {
        try {
            ivjJDialogContentPane = new javax.swing.JPanel();
            ivjJDialogContentPane.setName("JDialogContentPane");
            ivjJDialogContentPane.setLayout(new java.awt.GridBagLayout());
            java.awt.GridBagConstraints constraintsLogoLabel = new java.awt.GridBagConstraints();
            constraintsLogoLabel.gridx = 0; constraintsLogoLabel.gridy = 0;
            constraintsLogoLabel.gridwidth = 2;
            constraintsLogoLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
            constraintsLogoLabel.weightx = 1.0;
            constraintsLogoLabel.ipady = 5;
            constraintsLogoLabel.insets = new java.awt.Insets(5, 5, 0, 5);
            getJDialogContentPane().add(getLogoLabel(), constraintsLogoLabel);

            java.awt.GridBagConstraints constraintsRulerLabel = new java.awt.GridBagConstraints();
            constraintsRulerLabel.gridx = 0; constraintsRulerLabel.gridy = 1;
            constraintsRulerLabel.gridwidth = 3;
            constraintsRulerLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
            constraintsRulerLabel.ipady = 10;
            constraintsRulerLabel.insets = new java.awt.Insets(0, 5, 5, 10);
            getJDialogContentPane().add(getRulerLabel(), constraintsRulerLabel);

            java.awt.GridBagConstraints constraintsTextLabel = new java.awt.GridBagConstraints();
            constraintsTextLabel.gridx = 0; constraintsTextLabel.gridy = 2;
            constraintsTextLabel.gridwidth = 2;
            constraintsTextLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
            constraintsTextLabel.weightx = 1.0;
            constraintsTextLabel.ipady = 10;
        }
    }
}
constraintsTextLabel.insets = new java.awt.Insets(5, 5, 5, 5);
getDialogContentPane().add(getTextLabel(), constraintsTextLabel);
// user code begin {1}
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin {2}
// user code end
handleException(ivjExc);
}
}  
return ivjDialogContentPane;
/**
 * Return the LogoLabel property value.
 * @return javax.swing.JLabel
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getLogoLabel() {
if (ivjLogoLabel == null) {
try {
ivjLogoLabel = new javax.swing.JLabel();
ivjLogoLabel.setName("LogoLabel");
ivjLogoLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/peerviewlogo2.gif")));
ivjLogoLabel.setText("");
ivjLogoLabel.setHorizontalAlignment(java.awt.swing.SwingConstants.CENTER);
ivjLogoLabel.setHorizontaTextPosition(java.awt.swing.SwingConstants.CENTER);
// user code begin {1}
/* Border.raisedBorder = BorderFactory.createEtchedBorder(new java.awt.Color(33, 132, 132), new 
java.awt.Color(169, 169, 169)); */
Border.matteBorder = BorderFactory.createEtchedBorder(6, 6, 6, 6, ivjLogoLabel.getBackground());
Border.compoundBorder = BorderFactory.createCompoundBorder(matteBorder, raisedBorder);
ivjLogoLabel.setBorder(compoundBorder);
*/
ivjLogoLabel.setBackground(new java.awt.Color.white);
ivjLogoLabel.setForeground(new java.awt.Color.white);
ivjLogoLabel.setVerticalTextPosition(java.awt.swing.SwingConstants.BOTTOM);
// ivjLogoLabel.setVerticalAlignment(SwingConstants.BOTTOM);
ivjLogoLabel.setText("Release_0.900_2 (beta)");
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin {2}
// user code end
handleException(ivjExc);
}
}  
return ivjLogoLabel;
}
/**
 * Return the RulerLabel property value.
 * @return javax.swing.JLabel
 */
/** WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getRulerLabel() {
if (ivjRulerLabel == null) {
try {
ivjRulerLabel = new javax.swing.JLabel();
ivjRulerLabel.setName("RulerLabel");
ivjRulerLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ruler2.gif")));
ivjRulerLabel.setAlignmentX(java.awt.Component.CENTER_ALIGNMENT);
ivjRulerLabel.setText("");
ivjRulerLabel.setHorizontalAlignment(java.awt.swing.SwingConstants.CENTER);
// user code begin {1}
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin {2}
/*
190       // user code end
191       handleException(ivjExc);
192     }
193     }
194     return ivjRuleLabel;
195   }/*
196      * Return the TextLabel property value.
197      * @return javax.swing.JLabel
198      */
199     /* WARNING: THIS METHOD WILL BE REGENERATED. */
200     private javax.swing.JLabel getTextLabel() {
201       if (ivjTextLabel == null) {
202         try {
203           ivjTextLabel = new javax.swing.JLabel();
204           ivjTextLabel.setName("TextLabel");
205           ivjTextLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/text.gif")));
206           ivjTextLabel.setAlignmentX(java.awt.Component.CENTER_ALIGNMENT);
207           ivjTextLabel.setText("*");
208           ivjTextLabel.setHorizontalAlignment(java.awt.SwingConstants.CENTER);
209           // user code begin (1)
210           // user code end
211         } catch (java.lang.Throwable ivjExc) {
212             // user code begin (2)
213             // user code end
214             handleException(ivjExc);
215         }
216       }
217       return ivjTextLabel;
218     }
219     }/**
220        * Called whenever the part throws an exception.
221        * @param exception java.lang.Throwable
222        */
223     private void handleException(java.lang.Throwable exception) {
224       /* Uncomment the following lines to print uncaught exceptions to stdout */
225       // System.out.println("----------- UNCAUGHT EXCEPTION -----------");
226       // exception.printStackTrace(System.out);
227     }/**
228      * Initialize the class.
229      */
230      /* WARNING: THIS METHOD WILL BE REGENERATED. */
231     private void initialize() {
232       try {
233         // user code begin (1)
234         // user code end
235         setName("AboutDialog");
236         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
237         setTitle("AboutPeerView");
238         setBackground(java.awt.Color.window);
239         setModal(true);
240         setSize(483, 267);
241         setResizable(false);
242         getContentPane().add(jDialogContentPane());
243         } catch (java.lang.Throwable ivjExc) {
244             handleException(ivjExc);
245         } // user code begin (2)
246         setLocationRelativeTo( getOwner() );
247         // user code end
248      }/**
249      * main entrypoint - starts the part when it is run as an application
250      * @param args java.lang.String[]
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
public static void main(java.lang.String[] args) {
    try {
        AboutDialog aboutDialog = new AboutDialog();
        aboutDialog.setModal(true);
        aboutDialog.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
                System.exit(0);
            }
        });
        aboutDialog.setVisible(true);
    } catch (Throwable exception) {
        System.err.println("Exception occurred in main() of AboutDialog");
        exception.printStackTrace(System.out);
    }
}

package peerviewmisc;

import java.io.Serializable;

/**
 * This class implements a container for transmitting compressed byte arrays and their descriptors around a network using
 * the Java serialization mechanism for marshalling and unmarshalling.
 * @author
 */
public class CompressedDocumentPackage implements Serializable {
    private int uncompressedLength;
    private int compressedLength;
    private byte[] compressedContents = null;
    private java.lang.String path = null;
    private java.lang.String name = null;

    /**
     * CompressedDocumentPackage constructor comment.
     */
    public CompressedDocumentPackage() {
        super();
    }

    /**
     * Insert the method’s description here.
     * @return byte[]
     */
    public byte[] getCompressedContents() {
        return compressedContents;
    }

    /**
     * Insert the method’s description here.
     * @return int
     */
    public int getCompressedLength() {
        return compressedLength;
    }

    /**
     * Insert the method’s description here.
     * @return java.lang.String
     */
    public java.lang.String getName() {
        return name;
    }

Listing C.35: CompressedDocumentPackage.java
package peerviewmisc;

import javax.swing.event.*;
import java.awt.event.*;

Listing C.36: ConfirmationBox.java
/**
 * This class implements the confirmation box displayed when a user is asked to confirm or deny an action.
 * Creation date: (12-07-00 11:11:22)
 * @author
 */

public class ConfirmationBox extends javax.swing.JDialog {

    public class OKButtonHandler implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
            submitAndClose();
        }
    }

    public class CancelButtonHandler implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
            cancelAndClose();
        }
    }

    private javax.swing.JButton ivjCancelButton = null;
    private javax.swing.JPanel ivjDialogContentPanel = null;
    private javax.swing.JButton ivjOKButton = null;
    private boolean confirmed;
    private javax.swing.JLabel ivjIconLabel = null;
    private javax.swing.JPanel ivjMainPanel = null;
    private javax.swing.JTextField ivjMessagePanel = null;
    private javax.swing.JPanel ivjButtonPanel = null;
    private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;

    /**
     * ConfirmationBox constructor comment.
     */
    public ConfirmationBox() {
        super();
        initialize();
    }

    /**
     * ConfirmationBox constructor comment.
     * @param owner java.awt.Dialog
     */
    public ConfirmationBox(java.awt.Dialog owner) {
        super(owner);
    }

    /**
     * ConfirmationBox constructor comment.
     * @param owner java.awt.Dialog
     * @param title java.lang.String
     */
    public ConfirmationBox(java.awt.Dialog owner, String title) {
        super(owner, title);
    }

    /**
     * ConfirmationBox constructor comment.
     * @param owner java.awt.Dialog
     * @param title java.lang.String
     * @param modal boolean
     */
    public ConfirmationBox(java.awt.Dialog owner, String title, boolean modal) {
        super(owner, title, modal);
    }

    /**
     * ConfirmationBox constructor comment.
     */
* @param owner java.awt.Dialog
  * @param modal boolean
  */
public ConfirmationBox(java.awt.Dialog owner, boolean modal) {
    super(owner, modal);
}
/**
 * ConfirmationBox constructor comment.
 * @param owner java.awt.Frame
 */
public ConfirmationBox(java.awt.Frame owner) {
    super(owner);
}
/**
 * ConfirmationBox constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public ConfirmationBox(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
 * ConfirmationBox constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public ConfirmationBox(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * ConfirmationBox constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public ConfirmationBox(java.awt.Frame owner, boolean modal) {
}
/**
 * Insert the method's description here.
 * Creation date: (12-07-00 11:37:04)
 */
public void cancelAndClose()
{
    setConfirmed( false );
    dispose();
}
/**
 * Return the JPanel1 property value.
 * @return javax.swing.JPanel
 */
public javax.swing.JPanel getButtonPanel() {
    if (ivjButtonPanel == null) {
        try {
            ivjButtonPanel = new javax.swing.JPanel();
            ivjButtonPanel.setName("ButtonPanel");
            ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
            getButtonPanel().add(getOKButton(), getOKButton().getName());
            getButtonPanel().add(getCancelButton(), getCancelButton().getName());
            // user code begin (1)
            // user code end
        } catch (javax.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjButtonPanel;
}
private javax.swing.JButton getCancelButton() {
    if (ivjCancelButton == null) {
        try {
            ivjCancelButton = new javax.swing.JButton();
            ivjCancelButton.setName("CancelButton");
            ivjCancelButton.setText("Cancel");
            // user code begin (1)
            ivjCancelButton.addActionListener(new CancelButtonHandler());
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjCancelButton;
}

private javax.swing.JLabel getIconLabel() {
    if (ivjIconLabel == null) {
        try {
            ivjIconLabel = new javax.swing.JLabel();
            ivjIconLabel.setName("IconLabel");
            ivjIconLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/general/ContextualHelp24.gif")));
            ivjIconLabel.setText("";
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjIconLabel;
}
private javax.swing.JDialog getDialogContentPane() {
    if (ivjDialogContentPane == null) {
        try {
            ivjDialogContentPane = new javax.swing.JDialog();
            ivjDialogContentPane.setName("DialogContentPanel");
            ivjDialogContentPane.setLayout(new java.awt.BorderLayout());
            getContentPane().add(getContentPane(), "South");
            getContentPane().add(getContentPane(), "Center");
            // user code begin (1)
            // user code end
        } catch (java.lang.Exception ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjDialogContentPane;
}

private javax.swing.JDialog getMainPanel() {
    if (ivjMainPanel == null) {
        try {
            ivjMainPanel = new javax.swing.JDialog();
            ivjMainPanel.setName("MainPanel");
            ivjMainPanel.setLayout(new java.awt.GridBagLayout());
            java.awt.GridBagConstraints constraintsMessagePanel = new java.awt.GridBagConstraints();
            constraintsMessagePanel.gridx = 1; constraintsMessagePanel.gridy = 0;
            constraintsMessagePanel.fill = java.awt.GridBagConstraints.NONE;
            constraintsMessagePanel.weightx = 1.0;
            getContentPane().setInsets = new java.awt.Insets(4, 4, 4, 4);
            getContentPane().add(getContentPane(), constraintsMessagePanel);
            java.awt.GridBagConstraints constraintsIconLabel = new java.awt.GridBagConstraints();
            constraintsIconLabel.gridx = 0; constraintsIconLabel.gridy = 0;
            constraintsIconLabel.insets = new java.awt.Insets(4, 4, 4, 4);
            getContentPane().add(getContentPane(), constraintsIconLabel);
            // user code begin (1)
            getContentPane().setBorder( SharedConstants.createDialogBorder( getContentPane().getBackground() ));
            // user code end
        } catch (java.lang.Exception ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjMainPanel;
}

private javax.swing.JTextField getMessagePanel() {
    if (ivjMessagePanel == null) {
        try {
            ivjMessagePanel = new javax.swing.JTextField();
            ivjMessagePanel.setName("MessagePanel");
            // user code begin (1)
            // user code end
            handleException(ivjExc);
        } catch (java.lang.Exception ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjMessagePanel;
}
ivjMessagePanel.setEditable(false);
// user code begin {1}
ivjMessagePanel.setBackground( getButtonPanel().getBackground() );
// user code end
} catch (java.lang.Throwable ivjExc) {
    // user code begin {2}
    // user code end
    handleException(ivjExc);
}
return ivjMessagePanel;

/**
 * Return the OKButton property value.
 */
private javax.swing.JButton getOKButton() {
    if (ivjOKButton == null) {
        try {
            ivjOKButton = new javax.swing.JButton();
            ivjOKButton.setName("OKButton");
            ivjOKButton.setMnemonic('O');
            ivjOKButton.setText("OK");
            // user code begin {1}
            ivjOKButton.addActionListener( new OKButtonHandler() );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjOKButton;
}

/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {
    /* Uncomment the following lines to print uncaught exceptions to stdout */
    // System.out.println("-------- UNCAUGHT EXCEPTION --------");
    // exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
private void initialize() {
    try {
        // user code begin {1}
        // user code end
        setName("ConfirmationBox");
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setSize(456, 191);
        setEnabled(true);
        setModal(true);
        setTitle("Confirmation");
        getContentPane().add(dialogContentPane());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    // user code begin {2}
    // user code end
}
public boolean isConfirmed() {
    return confirmed;
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
    try {
        ConfirmationBox aConfirmationBox;
        aConfirmationBox = new ConfirmationBox();
        aConfirmationBox.setModal(true);
        aConfirmationBox.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
                System.exit(0);
            }
        });
        aConfirmationBox.setVisible(true);
    }
    catch (Throwable exception) {
        System.err.println("Exception occurred in main() of javax.swing.JDialog");
        exception.printStackTrace(System.out);
    }
}

/**
 * Insert the method’s description here.
 * Creation date: (12-07-00 11:26:11)
 * @param newConfirmed boolean
 */
public void setConfirmed(boolean newConfirmed) {
    confirmed = newConfirmed;
}

/**
 * Insert the method’s description here.
 * Creation date: (12-07-00 11:44:14)
 * @param newText java.lang.String
 */
public void setText(String newText) {
    messagePanel().setText( newText );
    messagePanel().setPreferredSize( SharedConstants.computeDisplayDimension( newText, messagePanel ()>.getFontMetrics( messagePanel ().getFont () ) ) );
    messagePanel().setSize( messagePanel ().getPreferredSize() );
    setLocationRelativeTo( getOwner() );
}

/**
 * Insert the method’s description here.
 * Creation date: (12-07-00 11:36:53)
 */
public void submitAndClose() {
    setConfirmed( true );
    dispose();
}

Listing C.37: DataPackage.java

package peerviewmisc;
import java.io.*;

429
import java.util.Hashtable;

/**
 * This class implements a container used for transmitting data between clients.
 * Creation date: (04-07-00 17:10:18)
 * Author:
 */

class DataPackage implements Serializable {

    private final static java.lang.String AUTHOR_NAME = "Author, name";
    private final static java.lang.String DOCPATH = "Document, name and path";
    private final static java.lang.String DocNAME = "Document, name (without path)";
    private final static java.lang.String DESCRIPTION = "Description";
    private java.util.Properties properties = new java.util.Properties();
    private java.lang.Object contents = null;
    private final static java.lang.String WORKGROUP_ID = "Workgroup, ID";
    private final static java.lang.String MESSAGE_ID = "Message, ID";
    private final static java.lang.String CLIENT_NAME = "Client, name";

    // DataPackage constructor comment.
    public DataPackage() {
        super();
    }

    public final static java.lang.String getAUTHOR_NAME() {
        return AUTHOR_NAME;
    }

    public final static java.lang.String getCLIENT_NAME() {
        return CLIENT_NAME;
    }

    public java.lang.Object getContents() {
        return contents;
    }

    public final static java.lang.String getDESCRIPTION() {
        return DESCRIPTION;
    }

    public final static java.lang.String getDOCNAMES() {
        return DocNAME;
    }

}
public final static java.lang.String getDCPATH() {
    return DCPATH;
}
/**
 * Insert the method’s description here.
 * Creation date: (28-10-00 00:03:49)
 */
public final static java.lang.String getMessage_ID() {
    return MESSAGE_ID;
}
/**
 * Insert the method’s description here.
 * Creation date: (27-10-00 22:55:47)
 */
public java.util.Properties getProperties() {
    return properties;
}
/**
 * Insert the method’s description here.
 * Creation date: (27-10-00 23:18:46)
 */
public final static java.lang.String getWORKGROUP_ID() {
    return WORKGROUP_ID;
}
/**
 * Insert the method’s description here.
 * Creation date: (27-10-00 22:59:23)
 */
public void setContents(java.lang.Object newContents) {
    contents = newContents;
}
/**
 * Insert the method’s description here.
 * Creation date: (27-10-00 22:55:47)
 */
public void setProperties(java.util.Properties newProperties) {
    properties = newProperties;
}

Listing C.38: DeleteGroup.java

package peerviewmisc;
/**
 * This class implements the delete group dialog box.
 * Creation date: (29-06-00 16:19:12)
 */
public class DeleteGroup extends DisplayGroup {
    /**
     * DeleteGroup constructor comment.
     */
    public DeleteGroup() {
        super();
        initialize();
    }
}
* DeleteGroup constructor comment.
public DeleteGroup(java.awt.Dialog owner) {
    super(owner);
}
/**
    * DeleteGroup constructor comment.
    * @param owner java.awt.Dialog
    * @param title java.lang.String
    */
public DeleteGroup(java.awt.Dialog owner, String title) {
    super(owner, title);
}
/**
    * DeleteGroup constructor comment.
    * @param owner java.awt.Dialog
    * @param modal boolean
    */
public DeleteGroup(java.awt.Dialog owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
    * DeleteGroup constructor comment.
    * @param owner java.awt.Frame
    */
public DeleteGroup(java.awt.Frame owner) {
    super(owner);
}
/**
    * DeleteGroup constructor comment.
    * @param owner java.awt.Frame
    * @param title java.lang.String
    */
public DeleteGroup(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
    * DeleteGroup constructor comment.
    * @param owner java.awt.Frame
    */
public DeleteGroup(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
    * DeleteGroup constructor comment.
    * @param owner java.awt.Frame
    */
public DeleteGroup(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}
/**
    * DeleteGroup constructor comment.
    * @param owner java.awt.Frame
    */
public DeleteGroup(java.awt.Frame owner, Throwable

432
private void handleException(java.lang.Throwable exception) {
  /* Uncomment the following lines to print uncaught exceptions to stdout */
  // System.out.println("-------- UNCAUGHT EXCEPTION --------");
  // exception.printStackTrace(System.out);
}

/*
 * Initialize the class.
 */

/*
 * WARNING: THIS METHOD WILL BE REGENERATED. */

private void initialize() {
  try {
    // user code begin {1}
    // user code end
    setName("DeleteGroup");
    } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
    }
  // user code begin {2}
    setTitle( "DeleteGroup" );
    editable( false );
    getOKButton().setText( "Delete" );
  // user code end
  }

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */

public static void main(java.lang.String[] args) {
  try {
    DeleteGroup aDeleteGroup;
    aDeleteGroup = new DeleteGroup();
    aDeleteGroup.setModal(true);
    aDeleteGroup.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(java.awt.event.WindowEvent e) {
        System.exit(0);
      }
    });
    aDeleteGroup.setVisible(true);
  } catch (Throwable exception) {
    System.err.println("Exception occurred.in.main() of.serverapp.DeleteGroup");
    exception.printStackTrace(System.out);
  }

/**
 * Insert the method's description here.
 * @param args java.lang.String[]
 */

public void submitAndClose() {} } }

package peerviewmisc;

import javax.swing.JComponent;
import javax.swing.JTextField;
import javax.swing.text.JTextComponent;
import java.awt.Component;
import java.awt.event.*;
import java.awt.event.*;

/**
 * This abstract class is the root of the class hierarchy of dialog boxes that accept input from the
 * user.
 * @param args java.lang.String[]
 */

Listing C.39: DisplayGroup.java
public abstract class DisplayGroup extends javax.swing.JDialog {

    public class OKButtonHandler implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
            submitAndClose();
        }
    }

    public class CancelButtonHandler implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
            cancelAndClose();
        }
    }

    protected javax.swing.JButton ivjCancelButton = null;
    protected javax.swing.JLabel ivjDescriptionLabel = null;
    protected javax.swing.JLabel ivjGroupNameLabel = null;
    protected javax.swing.JDialog ivjDialogContentPane = null;
    protected javax.swing.JButton ivjOKButton = null;
    protected javax.swing.JDialog ivjButtonPanel = null;
    protected java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
    protected javax.swing.JDialog ivjMainPanel = null;
    protected WorkGroup workGroup = null;
    protected javax.swing.JTextField ivjDescriptionField = null;
    protected javax.swing.JTextField ivjGroupNameField = null;
    protected clientapp.GroupDirectory groupDirectory;
    protected boolean isDiscarded;
    protected javax.swing.JScrollPane ivjDescriptionScrollPane = null;

    /**
     * DisplayGroup constructor comment.
     */
    public DisplayGroup() {
        super();
        initialize();
    }

    /**
     * DisplayGroup constructor comment.
     *
     * @param owner java.awt.Dialog
     */
    public DisplayGroup(java.awt.Dialog owner) {
        super(owner);
    }

    /**
     * DisplayGroup constructor comment.
     *
     * @param owner java.awt.Dialog
     * @param title java.lang.String
     */
    public DisplayGroup(java.awt.Dialog owner, String title) {
        super(owner, title);
    }

    /**
     * DisplayGroup constructor comment.
     *
     * @param owner java.awt.Dialog
     * @param title java.lang.String
     * @param modal boolean
     */
    public DisplayGroup(java.awt.Dialog owner, String title, boolean modal) {
        super(owner, title, modal);
    }

    /**
     * DisplayGroup constructor comment.
     */
}
* @param owner java.awt.Dialog
* @param modal boolean
*/
public DisplayGroup(java.awt.Dialog owner, boolean modal) {
    super(owner, modal);
}
/**
* DisplayGroup constructor comment.
* @param owner java.awt.Frame
*/
public DisplayGroup(java.awt.Frame owner) {
    super(owner);
}
/**
* DisplayGroup constructor comment.
* @param owner java.awt.Frame
* @param title java.lang.String
*/
public DisplayGroup(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
* DisplayGroup constructor comment.
* @param owner java.awt.Frame
* @param title java.lang.String
* @param modal boolean
*/
public DisplayGroup(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
* Set editability of text fields in the main panel of the display group dialog box.
* @param edit boolean
*/
public void editable(boolean edit)

Component components[] = getMainPanel().getComponents();
for (int i = 0; i < components.length; i++)
{
    if (components[i] instanceof JTextComponent)
    {
        ((JTextComponent) components[i]).setEditable(edit);
    }
}
/**
* Return the JPanel2 property value.

435
/* @return javax.swing.JPanel */

/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getButtonPanel() {
    if (ivjButtonPanel == null) {
        try {
            ivjButtonPanel = new javax.swing.JPanel();
            ivjButtonPanel.setName("ButtonPanel");
            ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
            getButtonPanel().add(getOkButton(), getOkButton().getName());
            getButtonPanel().add(getCancelButton(), getCancelButton().getName());
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
        return ivjButtonPanel;
    }
    /**
     * Return the ButtonPanelFlowLayout property value.
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private java.awt.FlowLayout getButtonPanelFlowLayout() {
        java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
        try {
            /** create part */
            ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
            ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
        return ivjButtonPanelFlowLayout;
    }
    /**
     * Return the CancelButton property value.
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    public javax.swing.JButton getCancelButton() {
        if (ivjCancelButton == null) {
            try {
                ivjCancelButton = new javax.swing.JButton();
                ivjCancelButton.setName("CancelButton");
                ivjCancelButton.setText("Cancel");
                // user code begin {1}
                ivjCancelButton.addActionListener(new CancelButtonHandler());
                // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
                // user code end
                handleException(ivjExc);
            }
        }
        return ivjCancelButton;
    }
    /**
     * Return the JPanel property value.
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JPanel getDescriptionPanel() {
        if (ivjDescriptionPanel == null) {
            try {
                ivjDescriptionPanel = new javax.swing.JPanel();
                ivjDescriptionPanel.setName("DescriptionPanel");
                // user code begin {1}
                // user code end
                handleException(ivjExc);
            } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
                // user code end
                handleException(ivjExc);
            }
        }
        return ivjDescriptionPanel;
    }
}

436
try {
    ivjDescriptionField = new javax.swing.JTextField();
    ivjDescriptionField.setName("DescriptionField");
    ivjDescriptionField.setBounds(0, 0, 312, 157);
    // user code begin (1)
    // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin (2)
        // user code end
        handleException(ivjExc);
    }
}
return ivjDescriptionField;
/**
 * Return the DescriptionLabel property value.
 * @return javax.swing.JLabel
 */
/*
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getDescriptionLabel() {
    if (ivjDescriptionLabel == null) {
        try {
            ivjDescriptionLabel = new javax.swing.JLabel();
            ivjDescriptionLabel.setName("DescriptionLabel");
            ivjDescriptionLabel.setText("Description");
            // user code begin (1)
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
        }
return ivjDescriptionLabel;
}
/**
 * Return the DescriptionScrollPane property value.
 * @return javax.swing.JScrollPane
 */
/*
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JScrollPane getDescriptionScrollPane() {
    if (ivjDescriptionScrollPane == null) {
        try {
            ivjDescriptionScrollPane = new javax.swing.JScrollPane();
            ivjDescriptionScrollPane.setName("DescriptionScrollPane");
            getDescriptionScrollPane().setViewportView(getDescriptionField());
            // user code begin (1)
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
        }
return ivjDescriptionScrollPane;
}/**
 * Insert the method's description here.
 * creation date: (26-07-00 17:26:16)
 * @return clientapp.GroupDirectory
 */
public clientapp.GroupDirectory getGroupDirectory() {
    return groupDirectory;
}/**
 * Return the JTextField1 property value.
 */
/**
 * @return javax.swing.JTextField
 */

private javax.swing.JTextField getGroupNameField() {
    if (ivjGroupNameField == null)
    
    try {
        ivjGroupNameField = new javax.swing.JTextField();
        ivjGroupNameField.setName("GroupNameField");
        // user code begin {1}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
    }
    
    return ivjGroupNameField;
}

/**
 * @return javax.swing.JLabel
 */

private javax.swing.JLabel getGroupNameLabel() {
    if (ivjGroupNameLabel == null)
    
    try {
        ivjGroupNameLabel = new javax.swing.JLabel();
        ivjGroupNameLabel.setName("GroupNameLabel");
        ivjGroupNameLabel.setText("GroupName");
        // user code begin {1}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
    }
    
    return ivjGroupNameLabel;
}

/**
 * @return javax.swing.JPanel
 */

private javax.swing.JPanel getDialogContentPane() {
    if (ivjDialogContentPane == null)
    
    try {
        ivjDialogContentPane = new javax.swing.JPanel();
        ivjDialogContentPane.setName("DialogContentPane");
        ivjDialogContentPane.setLayout(new java.awt.BorderLayout());
        getDialogContentPane().add(getMainPanel(), "Center");
        getDialogContentPane().add(getButtonPanel(), "South");
        // user code begin {1}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
    }
    
    return ivjDialogContentPane;
}

/**
 * @return JPanel1 property value.
 */
private javax.swing.JComponent getMainPanel() {
    if (ivjMainPanel == null) {
        try {
            ivjMainPanel = new javax.swing.JPanel();
            ivjMainPanel.setName("MainPanel");
            ivjMainPanel.setLayout(new java.awt.GridBagLayout());
            java.awt.GridBagConstraints constraintsGroupNameLabel = new java.awt GridBagConstraints();
            constraintsGroupNameLabel.gridx = 0; constraintsGroupNameLabel.gridy = 0;
            constraintsGroupNameLabel.anchor = java.awt GridBagConstraints.NORTHWEST;
            constraintsGroupNameLabel.insets = new java.awt.Insets(4, 4, 4);
            getMainPanel().add(getGroupNameLabel(), constraintsGroupNameLabel);
            java.awt GridBagConstraints constraintsDescriptionLabel = new java.awt GridBagConstraints();
            constraintsDescriptionLabel.gridx = 0; constraintsDescriptionLabel.gridy = 2;
            constraintsDescriptionLabel.anchor = java.awt GridBagConstraints.NORTHWEST;
            constraintsDescriptionLabel.insets = new java.awt.Insets(4, 4, 4);
            getMainPanel().add(getDescriptionLabel(), constraintsDescriptionLabel);
            java.awt GridBagConstraints constraintsGroupNameField = new java.awt GridBagConstraints();
            constraintsGroupNameField.gridx = 1; constraintsGroupNameField.gridy = 0;
            constraintsGroupNameField.fill = java.awt GridBagConstraints.HORIZONTAL;
            constraintsGroupNameField.anchor = java.awt GridBagConstraints.EAST;
            constraintsGroupNameField.weightx = 1.0;
            constraintsGroupNameField.insets = new java.awt.Insets(4, 4, 4);
            getMainPanel().add(getGroupNameField(), constraintsGroupNameField);
            java.awt GridBagConstraints constraintsDescriptionScrollPane = new java.awt GridBagConstraints();
            constraintsDescriptionScrollPane.gridx = 1; constraintsDescriptionScrollPane.gridy = 2;
            constraintsDescriptionScrollPane.fill = java.awt GridBagConstraints.BOTH;
            constraintsDescriptionScrollPane.weightx = 1.0;
            constraintsDescriptionScrollPane.weighty = 1.0;
            constraintsDescriptionScrollPane.insets = new java.awt.Insets(4, 4, 4);
            getMainPanel().add(getDescriptionScrollPane(), constraintsDescriptionScrollPane);
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (1)
            getMainPanel().setBorder( SharedConstants.createDialogBorder( getMainPanel().getBackground() ));
            // user code end
        } finally {
            return ivjMainPanel;
        }
    }
    return ivjMainPanel;
}

public java.awt.Button getOKButton() {
    if (ivjOKButton == null) {
        try {
            ivjOKButton = new javax.swing.Button();
            ivjOKButton.setName("OKButton");
            ivjOKButton.setMnemonic('O');
            ivjOKButton.setText("OK");
            // user code begin (1)
            ivjOKButton.addActionListener( new OKButtonHandler() );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjOKButton;
}
public WorkGroup getWorkGroup() {
    return workGroup;
}

/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {
    /**
     * Uncomment the following lines to print uncaught exceptions to stdout */
     // System.out.println("-------- UNCAUGHT EXCEPTION " + exception.getClass());
     // exception.printStackTrace(System.out);
    }
    /**
     * WARNING: THIS METHOD WILL BE REGENERATED. */
    private void initialize() {
        try {
            // user code begin {1}
            // user code end
            setName("DisplayGroup");
            setDefaultCloseOperation(java.awt.WindowConstants.DISPOSE_ON_CLOSE);
            setSize(426, 240);
            setContentPane(getDialogContentPane());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
        // user code begin {2}
        setModal( true );
        // set to true as default to ensure that cancelling using the upper right corner X doesn’t result in
        // a spurious entry
        setIsDiscarded( true );
        setLocationRelativeTo( getOwner() );
    }
    // user code end
    /**
     * Insert the method’s description here.
     * Creation date: (26-07-00 19:10:36)
     * @return boolean
     */
    public boolean isIsDiscarded() {
        return isIsDiscarded;
    }
    /**
     * main entrypoint - starts the part when it is run as an application
     * @param args java.lang.String[]
     */
    public static void main(java.lang.String[] args) {
    }
    /* try {*/
        DisplayGroup aDisplayGroup;
        aDisplayGroup = new DisplayGroup();
        aDisplayGroup.setModal(true);
        aDisplayGroup.addWindowListener(new java.awt.event.WindowAdapter () {
            public void windowClosing(java.awt.event.WindowEvent e) {
                System.exit(0);
            }
        });
    */
}
package peerviewmisc;

/**
 * This class implements a container which represents a node in the client
 * panorama by storing the name of its associated
 * document and the size, position and locking status of the node.
 * @author
 */
public class DocPackage implements java.io.Serializable {
    private java.lang.String docNameAndPath = null;
    private java.awt.Dimension size = null;
    private java.awt.Dimension position = null;
    private java.lang.String NodeLocked = null;
    /* DocPackage constructor comment. */
    public DocPackage() {
        super();
    }
}
* Insert the method’s description here.
* Creation date: (11-10-00 12:12:19)
* @param docNameAndPathArg java.lang.String
* @param size java.awt.Dimension
* @param position java.awt.Dimension
 */
public DocPackage(String docNameAndPathArg, java.awt.Dimension sizeArg, java.awt.Dimension positionArg )
{
    setDocNameAndPath( docNameAndPathArg );
    setSize( sizeArg );
    setPosition( positionArg );
}
/**
 * Insert the method’s description here.
 * Creation date: (11-10-00 12:03:36)
 * @return java.lang.String
 */
public java.lang.String getDocNameAndPath() {
    return docNameAndPath;
}
/**
 * Insert the method’s description here.
 * Creation date: (16-10-00 09:51:21)
 * @return java.lang.String
 */
public java.lang.String getNodeLocked() {
    return NodeLocked;
}
/**
 * Insert the method’s description here.
 * Creation date: (11-10-00 12:04:44)
 * @return java.awt.Dimension
 */
public java.awt.Dimension getPosition() {
    return position;
}
/**
 * Insert the method’s description here.
 * Creation date: (11-10-00 12:03:54)
 * @return java.awt.Dimension
 */
public java.awt.Dimension getSize() {
    return size;
}
/**
 * Insert the method’s description here.
 * Creation date: (11-10-00 12:03:36)
 * @param newDocNameAndPath java.lang.String
 */
public void setDocNameAndPath(java.lang.String newDocNameAndPath) {
    docNameAndPath = newDocNameAndPath;
}
/**
 * Insert the method’s description here.
 * Creation date: (16-10-00 09:51:21)
 * @param newNodeLocked java.lang.String
 */
public void setNodeLocked(java.lang.String newNodeLocked) {
    NodeLocked = newNodeLocked;
}
/**
 * Insert the method’s description here.
 * Creation date: (11-10-00 12:04:44)
 * @param newPosition java.awt.Dimension
 */
public void setPosition(java.awt.Dimension newPosition) {
position = newPosition;

/**
 * Insert the method's description here.
 * Creation date: (11-10-00 12:03:54)
 */
public void setSize(java.awt.Dimension newSize) {
    size = newSize;
}

package peerviewmisc;

/**
 * The edit group dialog box.
 * Creation date: (29-06-00 16:08:16)
 * @author:
 */
public class EditGroup extends DisplayGroup {

    /**
     * EditGroup constructor comment.
     */
    public EditGroup() {
        super();
        initialize();
    }

    /**
     * EditGroup constructor comment.
     * @param owner java.awt.Dialog
     */
    public EditGroup(java.awt.Dialog owner) {
        super(owner);
    }

    /**
     * EditGroup constructor comment.
     * @param owner java.awt.Dialog
     * @param title java.lang.String
     */
    public EditGroup(java.awt.Dialog owner, String title) {
        super(owner, title);
    }

    /**
     * EditGroup constructor comment.
     * @param owner java.awt.Dialog
     * @param title java.lang.String
     * @param modal boolean
     */
    public EditGroup(java.awt.Dialog owner, String title, boolean modal) {
        super(owner, title, modal);
    }

    /**
     * EditGroup constructor comment.
     * @param owner java.awt.Dialog
     * @param modal boolean
     */
    public EditGroup(java.awt.Dialog owner, boolean modal) {
    }

    /**
     * EditGroup constructor comment.
     * @param owner java.awt.Frame
     */

Listing C.41: EditGroup.java

443
public EditGroup(java.awt.Frame owner) {
    super(owner);
}
/**
 * EditGroup constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public EditGroup(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
 * EditGroup constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public EditGroup(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {
    /* Uncomment the following lines to print uncaught exceptions to stdout */
    // System.out.println("---------- UNCAUGHT EXCEPTION ----------");
    // exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/**
 * WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
    try {
        // user code begin {1}
        // user code end
        setName("EditGroup");
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    // user code begin {2}
    setTitle("EditGroup");
    editable( true );
    getOKButton().setText("Submit");
    getCancelButton().setLabelMnemonic( 's' );
    getCancelButton().setMnemonic( "Discard" );
    // user code end
    }
    */
    * main entrypoint - starts the part when it is run as an application
    * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
    try {
        EditGroup aEditGroup;
aEditGroup = new EditGroup();
aEditGroup.setModal(true);
aEditGroup.addWindowListener(new java.awt.event.WindowAdapter()
  public void windowClosing(java.awt.event.ActionEvent e) {
    System.exit(0);
  });
aEditGroup.setVisible(true);
}
} catch (Throwable exception) {
  System.err.println("Exception occurred in main()",(serverapp.EditGroup");
excepion.printStackTrace(System.out);
}
*/
/**
 * Insert the method's description here.
 * Creation date: (04-07-00 00:18:52)
 */
public void submitAndClose()
{
  String groupName = ivjGroupNameField.getText();
  String description = ivjDescriptionField.getText();
  ErrorBox errorBox = new ErrorBox();
  if (groupName.length() == 0)
  {
    errorBox.setText(SharedConstants.getGROUP_NAME_MISSING_MESSAGE());
    errorBox.show();
    return;
  }
  getWorkGroup().setDescription(ivjDescriptionField.getText());
  getWorkGroup().setGroupName(ivjGroupNameField.getText());
  setInDiscarded(false);
  dispose();
}
}

package peerviewmisc;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.table.*;
import javax.swing.event.*;

/**
 * The dialog box displayed when the user is notified of an error.
 * Creation date: (07-06-00 00:37:01)
 * @author:
 */
public class ErrorBox extends JDialog {
  public class CloseButtonHandler implements ActionListener {
    public void actionPerformed(ActionEvent ae) {
      dispose();
    }
  }
  private JPanel ivjDialogContentPane = null;
  private JPanel ivjButtonPanel = null;
  private FlowLayout ivjButtonPanelFlowLayout = null;
  private JLabel ivjIconLabel = null;
  private JPanel ivjMainPanel = null;

Listing C.42: ErrorBox.java
private JTextPane ivjMessagePanel = null;
private JButton ivjCloseButton = null;
/**
 * ErrorBox constructor comment.
 */
public ErrorBox() {
    super();
    initialize();
}
/**
 * ErrorBox constructor comment.
 * @param owner java.awt.Dialog
 */
public ErrorBox(java.awt.Dialog owner) {
    super(owner);
}
/**
 * ErrorBox constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public ErrorBox(java.awt.Dialog owner, String title) {
    super(owner, title);
}
/**
 * ErrorBox constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public ErrorBox(java.awt.Dialog owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * ErrorBox constructor comment.
 * @param owner java.awt.Frame
 */
public ErrorBox(java.awt.Frame owner) {
    super(owner);
}
/**
 * ErrorBox constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public ErrorBox(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
 * ErrorBox constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public ErrorBox(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
96  * ErrorBox constructor comment.
97  * @param owner java.awt.Frame
98  * @param modal boolean
99  */
100 public ErrorBox(java.awt.Frame owner, boolean modal) {
101     super(owner, modal);
102 }
103 /**
104  * Return the JPanel1 property value.
105  * @return javax.swing.JPanel
106  */
107  /** WARNING: THIS METHOD WILL BE REGENERATED. */
108 private javax.swing.JPanel getButtonPanel() {
109     if (ivjButtonPanel == null) {
110         try {
111             ivjButtonPanel = new javax.swing.JPanel();
112             ivjButtonPanel.setName("ButtonPanel");
113             ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
114             getButtonPanel().add(getCloseButton(), getCloseButton().getName());
115             // user code begin (1)
116             // user code end
117         } catch (java.lang.Throwable ivjExc) {
118             // user code begin (2)
119             // user code end
120             handleException(ivjExc);
121         }
122         return ivjButtonPanel;
123     }
124     /**
125     * Return the ButtonPanelFlowLayout property value.
126     * @return java.awt.FlowLayout
127     */
128     /** WARNING: THIS METHOD WILL BE REGENERATED. */
129     private java.awt.FlowLayout getButtonPanelFlowLayout() {
130         java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
131         try {
132             /* create part */
133             ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
134             ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
135         } catch (java.lang.Throwable ivjExc) {
136             handleException(ivjExc);
137         }
138         return ivjButtonPanelFlowLayout;
139     }
140     /**
141     * Return the OK property value.
142     * @return javax.swing.JButton
143     */
144     /** WARNING: THIS METHOD WILL BE REGENERATED. */
145     private javax.swing.JButton getCloseButton() {
146         if (ivjCloseButton == null) {
147             try {
148                 ivjCloseButton = new javax.swing.JButton();
149                 ivjCloseButton.setName("CloseButton");
150                 ivjCloseButton.setMnemonic("C");
151                 ivjCloseButton.setText("Close");
152                 // user code begin (1)
153                 ivjCloseButton.addActionListener(new CloseButtonHandler());
154                 // user code end
155             } catch (java.lang.Throwable ivjExc) {
156                 // user code begin (2)
157                 // user code end
158                 handleException(ivjExc);
159             }
160         }
```java
private javax.swing.JLabel getIconLabel() {
    if (ivjIconLabel == null) {
        try {
            ivjIconLabel = new javax.swing.JLabel();
            ivjIconLabel.setName("IconLabel");
            ivjIconLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonImages/Scop24.gif")));
            ivjIconLabel.setText("");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    } // user code begin {3}
    // user code end
    return ivjIconLabel;
} // user code begin {4}
// user code end

private javax.swing.JPanel getDialogContentPane() {
    if (ivjDialogContentPane == null) {
        try {
            ivjDialogContentPane = new javax.swing.JPanel();
            ivjDialogContentPane.setName("DialogContentPane");
            ivjDialogContentPane.setLayout(new java.awt.BorderLayout());
            getDialogContentPane().add(getButtonPanel(), "South");
            getDialogContentPane().add(getMainPanel(), "Center");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    } // user code begin {3}
    // user code end
    return ivjDialogContentPane;
} // user code begin {4}
// user code end

private javax.swing.JPanel getMainPanel() {
    if (ivjMainPanel == null) {
        try {
            ivjMainPanel = new javax.swing.JPanel();
            ivjMainPanel.setName("MainPanel");
            ivjMainPanel.setLayout(new java.awt.GridBagLayout());
            java.awt.GridBagConstraints constraintsIconLabel = new java.awt.GridBagConstraints();
            constraintsIconLabel.gridx = 0; constraintsIconLabel.gridy = 0;
            constraintsIconLabel.anchor = java.awt.GridBagConstraints.WEST;
            constraintsIconLabel.insets = new java.awt.Insets(4, 4, 4, 4);
            getMainPanel().add(getIconLabel(), constraintsIconLabel);
```
java.awt.GridBagConstraints constraintsMessagePanel = new java.awt.GridBagConstraints();
constraintMessagePanel.gridx = 1; constraintsMessagePanel.gridy = 0;
constraintMessagePanel.fill = java.awt.GridBagConstraints.HORIZONTAL;
constraintMessagePanel.weightx = 1.0;
constraintMessagePanel.weighty = 1.0;
constraintMessagePanel.insets = new java.awt.Insets(4, 4, 4, 4);
getMainPanel().add(getMessagePanel(), constraintsMessagePanel);
// user code begin {1}
ivjMainPanel.setBorder(SharedConstants.createDialogBorder(getMainPanel().getBackground()));
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin {2}
// user code end
handleException(ivjExc);
}
return ivjMainPanel;
/**
 * Return the JTextPane property value.
 * @return javax.swing.JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.swing.JTextField getMessagePanel() {
if (ivjMessagePanel == null) {
try {
ivjMessagePanel = new javax.swing.JTextField();
ivjMessagePanel.setName("MessagePanel");
ivjMessagePanel.setFont(new java.awt.Font("sansserif", bold, 1, 12));
ivjMessagePanel.setBackground(java.awt.Color.lightGray);
ivjMessagePanel.setEditable(false);
// user code begin {1}
ivjMessagePanel.setBackground(getButtonPanel().getBackground());
// user code end
} catch (java.lang.Throwable ivjExc) {
// user code begin {2}
// user code end
handleException(ivjExc);
}
return ivjMessagePanel;
}/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {
/* Uncomment the following lines to print uncaught exceptions to stdout */
// System.out.println("---------- UNCAUGHT EXCEPTION ---------");
// exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
try {
// user code begin {1}
// user code end
setName("ErrorBox");
setDefaultCloseOperation(java.awt.WindowConstants.DISPOSE_ON_CLOSE);
setSize(401, 191);
setModal(true);
setTitle("Error message");
setContentPane(getDialogContentPanel());
}
Listing C.43: HelpDialog.java

```java
package peerviewmisc;

import javax.swing.tree.*;

/**
 * This class has now been superseded by a JavaHelp window but was originally intended as an
 * implementation of the PeerView
 * help system.
 * Creation date: (29-05-00 12:39:46)
 * @author:
 */
public class HelpDialog extends javax.swing.JDialog {
    private javax.swing.JPanel ivjHelpDialog = null;
    private javax.swing.JPanel ivjPanel1 = null;
    private java.awt.FlowLayout ivjPanel1FlowLayout = null;
    private javax.swing.JSplitPane ivjPanel1SplitPane = null;
    private javax.swing.JTextField ivjTextPanel = null;
    private javax.swing.JButton ivjOKButton = null;
    IvjEventHandle ivjEventHandler = new IvjEventHandle();
```
```java
class IVJEventHandler implements java.awt.event.ActionListener {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        if (e.getSource() == HelpDialog.this.OKButton())
            conn.toM1(e);
    }
}
private javax.swing.JTree ivjTopicTree = null;
/**
 * @param clazz java.awt.Dialog
 */
public HelpDialog() {
    super();
    initialize();
}
/**
 * @param owner java.awt.Dialog
 */
public HelpDialog(java.awt.Dialog owner) {
    super(owner);
}
/**
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public HelpDialog(java.awt.Dialog owner, String title) {
    super(owner, title);
}
/**
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public HelpDialog(java.awt.Dialog owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public HelpDialog(java.awt.Frame owner) {
    super(owner);
}
/**
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public HelpDialog(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public HelpDialog(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}
```
public HelpDialog(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}

/**
   * HelpDialog constructor comment.
   * @param owner java.awt.Frame
   * @param modal boolean
   */
public HelpDialog(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}

/**
   * connEtoM1: (OKButton.action.actionPerformed(java.awt.event.ActionEvent) --> HelpDialog.dispose())
   * @param arg1 java.awt.event.ActionEvent
   */
private void connEtoM1(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin {1}
        // user code end
        this.dispose();
        // user code begin {2}
        // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {3}
            // user code end
            handleException(ivjExc);
        }
    }

    /**
     * Return the HelpDialog property value.
     * @return javax.swing.JDialog
     */
    private javax.swing.JDialog getHelpDialog() {
        if (ivjHelpDialog == null) {
            try {
                ivjHelpDialog = new javax.swing.JDialog();
                ivjHelpDialog.setName("HelpDialog");
                ivjHelpDialog.setLayout(new java.awt.BorderLayout());
                getHelpDialog().add(getPanel1(), "South");
                getHelpDialog().add(getSplitPanel(), "Center");
                // user code begin {1}
                ivjHelpDialog.setBorder(javax.swing.BorderFactory.createMatteBorder(6, 6, 6, 6, ivjHelpDialog.getBackground()));
                // user code end
                } catch (java.lang.Throwable ivjExc) {
                    // user code begin {2}
                    // user code end
                    handleException(ivjExc);
                }

            return ivjHelpDialog;
        }
    }

    /**
     * Return the JPanel1 property value.
     * @return javax.swing.JPanel
     */
    private javax.swing.JPanel getPanel1() {
        if (ivjPanel1 == null) {
            try {
                ivjPanel1 = new javax.swing.JPanel();
                ivjPanel1.setName("JPanel1");
                ivjPanel1.setLayout(new JPanel1FlowLayout());
            }
        }
}
```java
get.JPanel1().add(getXButton(), getXButton().getName());
// user code begin (1)
// user code end

} catch (java.lang.Throwable ivjExc) {
// user code begin (2)
// user code end
handleException(ivjExc);
}
}
return ivj JPanel1;
}
/**
 * Return the JPanel1FlowLayout property value.
 */
private java.awt.FlowLayout getJPanel1FlowLayout() {
    java.awt.FlowLayout ivj JPanel1FlowLayout = null;
    try {
        /* Create part */
        ivj JPanel1FlowLayout = new java.awt.FlowLayout();
        ivj JPanel1FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    return ivj JPanel1FlowLayout;
}
/**
 * Return the JSplitPanel property value.
 */
private javax.swing.JSplitPane getJSplitPanel() {
    if (ivj JSplitPanel == null) {
        try {
            ivj JSplitPanel = new javax.swing.JSplitPane(javax.swing.JSplitPane.VERTICAL_SPLIT);
            ivj JSplitPanel.setName("JSplitPanel");
            ivj JSplitPanel().add(getTextPanel(), "bottom");
            ivj JSplitPanel().add(getTopicTree(), "top");
            // user code begin (1)
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
        return ivj JSplitPanel;
    }
    /**
     * Return the JTextPanel property value.
     */
    private javax.swing.JTextPane getJTextPanel() {
        if (ivj JTextPanel == null) {
            try {
                ivj JTextPanel = new javax.swing.JTextPanel();
                ivj JTextPanel.setName("JTextPanel");
                // user code begin (1)
                // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin (2)
                // user code end
                handleException(ivjExc);
            }
        }
    }
```

453
private javax.swing.JButton getOButton() {
    if (ivjOButton == null) {
        try {
            ivjOButton = new javax.swing.JButton();
            ivjOButton.setName("OButton");
            ivjOButton.setText("O");
            // user code begin {1}
            // user code end
            } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
                // user code end
                handleException(ivjExc);
            }
        } return ivjOButton;
    }
    */
    */
    private javax.swing.JTree getTopicTree() {
        if (ivjTopicTree == null) {
            try {
                ivjTopicTree = new javax.swing.JTree();
                ivjTopicTree.setName("TopicTree");
                // user code begin {1}
                ivjTopicTree.setShowsRootHandles( true );
                ((DefaultTreeModel) ivjTopicTree.getModel()).setAsksAllowsChildren( true );
                // user code end
                } catch (java.lang.Throwable ivjExc) {
                    // user code begin {2}
                    // user code end
                    handleException(ivjExc);
                }
        } return ivjTopicTree;
    }
    /* */
    /**
     * Called whenever the part throws an exception.
     * @param exception java.lang.Throwable
     */
    private void handleException(java.lang.Throwable exception) {
        /* Uncomment the following lines to print uncaught exceptions to stdout */
        System.out.println("-------- UNCAUGHT EXCEPTION --------");
        exception.printStackTrace(System.out);
    }
    */
    / * WARNING: THIS METHOD WILL BE REGENERATED. */
    private void initConnections() throws java.lang.Exception {
        // user code begin {1}
        // user code end
        getOButton().addActionListener(ivjEventHandler);
    }
/**
 * Initialize the class.
 */
private void initialize() {
    try {
        // user code begin {1}
        // user code end
        setName("HelpDialog");
        setDefaultCloseOperation( javax.swing.JFrame.DISPOSE_ON_CLOSE);
        setSize(426, 240);
        setTitle("Help");
        getContentPane().addHelpDialog();
        initConnections();
        } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    setLocationRelativeTo( getOwner() );
    initTree();
    // user code end

    } /*
 * Insert the method's description here.
 * Creation date: (02-09-00 21:20:16)
 */
public void initTree()
{
    javax.swing.tree.DefaultMutableTreeNode rootNode = new javax.swing.tree.DefaultMutableTreeNode("Not yet implemented");
    ( DefaultTreeModel ) getTopicTree().getModel().setRoot( rootNode );
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
    try {
        HelpDialog aHelpDialog;
        aHelpDialog = new HelpDialog();
        aHelpDialog.setModal(true);
        aHelpDialog.addWindowListener(new java.awt.event.MouseAdapter() {
        public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
        }
    });
    aHelpDialog.setVisible(true);
    } catch ( Throwable exception ) {
        System.err.println("Exception occurred in main()");
        exception.printStackTrace(System.out);
    }
}

Listing C.44: Message.java

package peerviewmisc;

import java.io.Serializable;
import java.util.Vector;

/**
 * This class implements a representation of the messages that can be displayed in the message area.
 * Creation date: (10-07-00 00:07:52)
 */
public class Message implements Serializable

455
private java.lang.String contents = null;
private java.lang.String author = null;
private java.lang.String title = null;
private java.lang.String messageId = null;
private java.lang.String document = null;
private java.util.Date creationDate = null;
private java.lang.String creator = null;


/**
 * Message constructor comment.
 */
public Message() {
    super();
}

/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:09:03)
 * @return java.lang.String
 */
public java.lang.String getAuthor() {
    return author;
}

/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:08:46)
 * @return java.lang.String
 */
public java.lang.String getContents() {
    return contents;
}

/**
 * Insert the method's description here.
 * Creation date: (16-07-00 10:40:40)
 * @return java.util.GregorianCalendar
 */
public java.util.Date getCreationDate() {
    return creationDate;
}

/**
 * Insert the method's description here.
 * Creation date: (19-07-00 17:29:31)
 * @return java.lang.String
 */
public java.lang.String getCreator() {
    return creator;
}

/**
 * Insert the method's description here.
 * Creation date: (19-07-00 19:06:13)
 * @return java.util.Vector
 */
public java.util.Vector getDataAsVector() {
    Vector v = new Vector();
    v.addElement( getAuthor() );
    v.addElement( getTitle() );
    v.addElement( getContents() );
    v.addElement( getMessageID() );
    v.addElement( getMessageId() );
    v.addElement( getDocument() );
    v.addElement( getCreationDate().toString() );
    return v;
}
/**
 * Insert the method's description here.
 * @return java.lang.String
 */
public java.lang.String getDocument() {
    return document;
}

/**
 * Insert the method's description here.
 * @return java.lang.String
 */
public final static java.lang.String[] getFIELD_NAMES() {
    return FIELD_NAMES;
}

/**
 * Insert the method's description here.
 * @param field java.lang.String
 */
public void setAuthor(java.lang.String field) {
    field = newAuthor;
}

/**
 * Insert the method's description here.
 * @param contents java.lang.String
 */
public void setContents(java.lang.String newContents) {
    contents = newContents;
}

/**
 * Insert the method's description here.
 * @param creationDate java.util.GregorianCalendar
 */
public void setCreationDate(java.util.Date newCreationDate) {
    creationDate = newCreationDate;
}

/**
 * Insert the method's description here.
 */
public void setCreator(java.lang.String newCreator) {
    creator = newCreator;
}
/**
 * Insert the method's description here.
 */
public void setDocument(java.lang.String newDocument) {
    document = newDocument;
}
/**
 * Insert the method's description here.
 */
public void setTitle(java.lang.String newTitle) {
    title = newTitle;
}
/**
 * Insert the method's description here.
 */
public String toString()
    { return title + ", [by] " + author + " [on] " + creationDate.toString(); }

/*
 * This class implements a box which is used for displaying brief messages.
 */
public class MessageBox extends javax.swing.JDialog
{
    public class CloseButtonHandler implements ActionListener
    {
        public void actionPerformed(ActionEvent ae)
        {
            dispose();
        }
    }
    private javax.swing.JPanel ivjButtonPanel = null;
    private javax.swing.FlowLayout ivjButtonPanelFlowLayout = null;
    private javax.swing.JButton ivjCloseButton = null;
    private javax.swing.JLabel ivjIconLabel = null;
    private javax.swing.JPanel ivjDialogContentPane = null;
    private javax.swing.JPanel ivjDialogContentPanel = null;
    private javax.swing.JPanel ivjMainPanel = null;
    private javax.swing.JTextPane ivjMessagePanel = null;
    /*
     * MessageBox constructor comment.
     */
    public MessageBox()
    { super();
    }
initialize();

/**
 * MessageBox constructor comment.
 * @param owner java.awt.Dialog
 */
public MessageBox(java.awt.Dialog owner) {
    super(owner);
}

/**
 * MessageBox constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public MessageBox(java.awt.Dialog owner, String title) {
    super(owner, title);
}

/**
 * MessageBox constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public MessageBox(java.awt.Dialog owner, boolean modal) {
    super(owner, modal);
}

/**
 * MessageBox constructor comment.
 * @param owner java.awt.Frame
 */
public MessageBox(java.awt.Frame owner) {
    super(owner);
}

/**
 * MessageBox constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public MessageBox(java.awt.Frame owner, String title) {
    super(owner, title);
}

/**
 * MessageBox constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public MessageBox(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}

/**
 * MessageBox constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public MessageBox(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}
/**
 * Return the JButton property value.
 * @return javax.swing.JButton
 */
private javax.swing.JButton getButtonPanel() {
  if (ivjButtonPanel == null) {
    try {
      ivjButtonPanel = new javax.swing.JPanel();
      ivjButtonPanel.setName("ButtonPanel");
      ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
      getButtonPanel().add(getCloseButton(), getCloseButton().getName());
      // user code begin {1}
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin {2}
      // user code end
      handleException(ivjExc);
    }
    return ivjButtonPanel;
  }
} /*
 */
/**
 * Return the ButtonPanelFlowLayout property value.
 * @return java.awt.FlowLayout
 */
private java.awt.FlowLayout getButtonPanelFlowLayout() {
  java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
  try {
    /* Create part */
    ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
    ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
    } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
    }
  return ivjButtonPanelFlowLayout;
} /*
 */
/**
 * Return the CloseButton property value.
 * @return javax.swing.JButton
 */
private javax.swing.JButton getCloseButton() {
  if (ivjCloseButton == null) {
    try {
      ivjCloseButton = new javax.swing.JButton();
      ivjCloseButton.setName("CloseButton");
      ivjCloseButton.setMnemonic("C");
      ivjCloseButton.setText("Close");
      // user code begin {1}
      ivjCloseButton.addActionListener(new CloseButtonHandler());
      // user code end
    } catch (java.lang.Throwable ivjExc) {
      // user code begin {2}
      // user code end
      handleException(ivjExc);
    }
    return ivjCloseButton;
  }
} /*
 */
/**
 * Return the IconLabel property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getIconLabel() {
  return new javax.swing.JLabel();
} /*
 */
private javax.swing.JLabel getIconLabel() {
    if (ivjIconLabel == null) {
        try {
            ivjIconLabel = new javax.swing.JLabel();
            ivjIconLabel.setName("IconLabel");
            ivjIconLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/general/Information24.png")));
            ivjIconLabel.setText("");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjIconLabel;
}
/**
 * Return the JDialogContentPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJDialogContentPane() {
    if (ivjJDialogContentPane == null) {
        try {
            ivjJDialogContentPane = new javax.swing.JPanel();
            ivjJDialogContentPane.setName("JDialogContentPane");
            ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
            getJDialogContentPane().add(getJDialogContentPane(), "Center");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJDialogContentPane;
}
/**
 * Return the JDialogContentPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJDialogContentPane() {
    if (ivjJDialogContentPane == null) {
        try {
            ivjJDialogContentPane = new javax.swing.JPanel();
            ivjJDialogContentPane.setName("JDialogContentPane");
            ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
            getJDialogContentPane().add(getMainPanel(), "South");
            getJDialogContentPane().add(getMainPanel(), "Center");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJDialogContentPane;
}
/*
 * Return the MainPanel property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getMainPanel() {
    if (ivjMainPanel == null) {
        try {
            ivjMainPanel = new javax.swing.JPanel();
            ivjMainPanel.setName("MainPanel");
            ivjMainPanel.setLayout(new java.awt.GridBagLayout());

            java.awt.GridBagConstraints constraintsIconLabel = new java.awt.GridBagConstraints();
            constraintsIconLabel.gridx = 0; constraintsIconLabel.gridy = 0;
            constraintsIconLabel.anchor = java.awt.GridBagConstraints.WEST;
            constraintsIconLabel.insets = new java.awt.Insets(4, 4, 4, 4);
            getMainPanel().add(getIconLabel(), constraintsIconLabel);

            java.awt.GridBagConstraints constraintsMessagePanel = new java.awt.GridBagConstraints();
            constraintsMessagePanel.gridx = 1; constraintsMessagePanel.gridy = 0;
            constraintsMessagePanel.fill = java.awt.GridBagConstraints.HORIZONTAL;
            constraintsMessagePanel.weightx = 1.0;
            constraintsMessagePanel.weighty = 1.0;
            constraintsMessagePanel.insets = new java.awt.Insets(4, 4, 4, 4);
            getMainPanel().add(getMessagePanel(), constraintsMessagePanel);

            // user code begin (1)
            ivjMainPanel.setBorder(SharedConstants.createDialogBorder(getMainPanel().getBackground()));
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
        return ivjMainPanel;
    }
}

/**
 * Return the MessagePanel property value.
 * @return javax.swing.JTextPane
 */
private javax.swing.JTextPane getMessagePanel() {
    if (ivjMessagePanel == null) {
        try {
            ivjMessagePanel = new javax.swing.JTextPane();
            ivjMessagePanel.setName("MessagePanel");
            ivjMessagePanel.setFont(new java.awt.Font("sansserif", 1, 12));
            ivjMessagePanel.setBackground(java.awt.Color.lightGray);
            ivjMessagePanel.setEditable(false);

            // user code begin (1)
            ivjMessagePanel.setBackground(getButtonPanel().getBackground());
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin (2)
            // user code end
            handleException(ivjExc);
        }
        return ivjMessagePanel;
    }
}

/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {
    /* Uncomment the following lines to print uncaught exceptions to stdout */
    // System.out.println("---------- UNCAUGHT EXCEPTION ----------");
    // exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
    try {
        // user code begin {1}
        // user code end
    }
    // user code begin {2}
    // user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
    try {
        MessageBox aMessageBox;
        aMessageBox = new MessageBox();
        aMessageBox.setModal(true);
        aMessageBox.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
                System.exit(0);
            }
        });
        aMessageBox.setVisible(true);
    } catch (Throwable exception) {
        System.err.println("Exception occurred in main()");
        exception.printStackTrace(System.out);
    }
}

/**
 * Insert the method's description here.
 * @param newText java.lang.String
 */
public void setText(String newText) {
    getMessagePanel1().setText( newText );
    getMessagePanel2().setPreferredSize( SharedConstants.computeDisplayDimension( newText , getMessagePanel2() .getFontMetrics() .getFont() ));
    getMessagePanel2().setSize( getMessagePanel2().getPreferredSize() );
    getMessagePanel2().setAlignmentY( javax.swing.GroupLayout.CENTER_ALIGNMENT );
    pack();
    setLocationRelativeTo( getOwner() );
}

package peerviewmisc;

import java.awt.event.*;

/**
 * The dialog box for specifying and submitting a new group.
 * @param creation date: (29-06-00 15:52:46)
 */
Listing C.46: NewGroup.java
public class NewGroup extends DisplayGroup {
    
    public NewGroup() {
        super();
        initialize();
    }

    public NewGroup(java.awt.Dialog owner) {
        super(owner);
    }

    public NewGroup(java.awt.Dialog owner, String title) {
        super(owner, title);
    }

    public NewGroup(java.awt.Dialog owner, String title, boolean modal) {
        super(owner, title, modal);
    }

    public NewGroup(java.awt.Dialog owner, boolean modal) {
        super(owner, modal);
    }

    public NewGroup(java.awt.Frame owner) {
        super(owner);
    }

    public NewGroup(java.awt.Frame owner, String title) {
        super(owner, title);
    }

    public NewGroup(java.awt.Frame owner, String title, boolean modal) { }
super(owner, title, modal);
}
/**
 * NewGroup constructor comment.
 */
@param owner java.awt.Frame
@param modal boolean
*/
public NewGroup(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}
/**
 * Called whenever the part throws an exception.
 */
@param exception java.lang.Throwable
*/
private void handleException(java.lang.Throwable exception) {
    /* Uncomment the following lines to print uncaught exceptions to stdout */
    // System.out.println("-------- UNCAUGHT EXCEPTION --------");
    // exception.printStackTrace(System.out);
}*/
/**
 * Initialize the class.
 */
/*
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
    try {
        // user code begin {1}
        // user code end
        setName("NewGroup");
        setModal(true);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
    // user code begin {2}
    setTitle( "NewGroup" );
    editable( true );
    getOKButton().setText( "Submit" );
    getCancelButton().setMnemonic( 's' );
    getCancelButton().setText( "Discard" );
    getCancelButton().setMnemonic( 'd' );
    // user code end
    /*
     * main entrypoint - starts the part when it is run as an application
     */
    @param args java.lang.String[]
    /*
     */
    public static void main(java.lang.String[] args) {
        try {
            NewGroup aNewGroup;
            aNewGroup = new NewGroup();
            aNewGroup.setModal(true);
            aNewGroup.addWindowListener(new java.awt.event.WindowAdapter() {
                public void windowClosing(java.awt.event.WindowEvent e) {
                    System.exit(0);
                }
            });
            aNewGroup.setVisible(true);
        } catch (Throwable exception) {
            System.err.println("Exception occurred in main()");
            exception.printStackTrace(System.out);
        }
    }
    /*
     * Insert the method's description here.
     */
    * Creation date: (04-07-00 00:18:52)
```java
public void submitAndClose()
{
    String groupName = ivjGroupNameField.getText();
    String description = ivjDescriptionField.getText();
    ErrorBox errorBox = new ErrorBox();

    if (groupName.length() == 0)
    {
        errorBox.setText(SharedConstants.get_GROUP_NAME_MISSING_MESSAGE());
        errorBox.show();
        return;
    }
    if (getGroupDirectory().groupExists(groupName, description) == true)
    {
        errorBox.setText(SharedConstants.getGROUP_EXISTS_MESSAGE());
        errorBox.show();
    }
    getWorkGroup().setDescription(ivjDescriptionField.getText());
    getWorkGroup().setGroupName(ivjGroupNameField.getText());
    setIsDiscarded(false);
    dispose();
}
```

package peerviewmisc;

/**
 * A container primitive mapping a key to a value.
 * Creation date: (20-09-00 02:31:56)
 * @author:
 */
public class Pair implements java.io.Serializable {
    private java.lang.String key = null;
    private java.lang.String value = null;
    /**
     * Pair constructor comment.
     */
    public Pair() {
        super();
    }
    /**
     * Insert the method’s description here.
     * Creation date: (20-09-00 02:36:16)
     * @param keyArg java.lang.String
     * @param valueArg java.lang.String
     */
    public Pair(String keyArg, String valueArg)
    {
        key = keyArg;
        value = valueArg;
    }
    /**
     * Insert the method’s description here.
     * Creation date: (20-09-00 02:32:13)
     * @return java.lang.String
     */
    public java.lang.String getKey()
    {
        return key;
    }
    /**
     * Insert the method’s description here.
     * Creation date: (20-09-00 02:34:30)
     * @return java.lang.String
     */
```

Listing C.47: Pair.java

466

-.

 Q
 



#ST >3  >3 8 "  ,  )-`BN \
" - "T  )->


 ;"  C(  ' R< )" )')  " 13
<@ " )')%  1ACBWJG;FGK;FGG GJ6A J6AWD ;N
<O " $(&:  YST 13R >3 8 "  

-. <T' $%  Y`BWST 13d U3 8 "  !:  Y NC\
Y!_#:  YU


 9
 ?
 Q



?



<@
<O

-.
T

9 Q



 
9



;"  C(  ' R< )" )')   " 13
" )')%  1ACBWJG;FGK;FGG GJ6A L1A G;N
" $(&:  )-ST >3  >3 8 "  
<T' $% )-`BWST 13  U3 8 "  #: )- NC\
 )-!_<: )-U

Q."24,4./6+N
 #"T:)(4>


R(
R(
R(
R(


9
?
Q
 
 
 9
 ?

 Q

 
9
?
 Q

 
9
?

 Q

:P ¬ !) r U%(')0* L!762),-%(6J,)2DP )%0?O%

'" ST
'" ST
'"&' (
'" ST


 +
<@
<O

-.



/>3d:V  U3 . '" "P3  
/>3d:V  U3M]'" "  )'"Y 3  
3d-U3 (13dS)>3  
13 - )63     . U

 $Y )' .S)%')  ; V    ');)   
"  )')%  1ACBG;FGH;FGG J;D A K6A5JK;N
-   '"PA

,
"' ) 
"' ) 
"' ) 
"' ) 
"' ) 
"' ) 
"'
"'
"'
"$T
"$T
"$T
"$T
"$T
"$T
"$T
"$T
"$T
"$T
"$T
"$T
"$T
"$T
"$T

-.
"'
"$T

8 











"







 @ ');)   \
)   ,]
) #ST >3
) #ST >3
)   !X
) #ST >3
) #ST >3

 cX c

 >
 3
 >
 3
  +  a
 >3
 >3

)    ) #ST >3 
)    ) #ST >3 
)    ) #ST >3 
 !  ) #ST 13  P3
 !  ) #ST 13  P3
 !  ) #ST 13  P3
 !  ) #ST 13  P3
 !  ) #ST 13  P3
 !  ) #ST 13  P3
 !  ) #ST 13  P3
 !  ) #ST 13  P3
 !  ) #ST 13  P3
 !  ) #ST 13  P3
 !  ) #ST 13  P3

 !  ) #ST 13  P3
 !  ) #ST 13  P3
 !  ) #ST 13  P3
 !  ) #ST 13  P3
' -)(   * 
<  ) #ST 13  U3 8
)    ) #ST >3 
 !  ) #ST 13  P3

  " !. Y!    

 X  _,H`
8 "   c   [)e
8 "  !X   +  a 
 8 c c [  c 
8 "   @   @
8 "  !c    8 

+ + e  
8 c c +
_!LL H;D 
   
[ &_
c   [)X c


 

"T "

)');>3

8  8 8   + e I_ *  "')-$( $(  * 
e  _ * '    ' ) * 
* @ ')))')  Y * 
_ * "  - )"')- $" )'"Y * 

 @ c





>3 8 "   8 c c + e %_ *$8 "T "$( * 
 3 8 "   8 c c [  c  _ *$8 "T " '" * 
>
 3 8 "  !X   +  a  @   @    [b_ * '  * 
>
8 "  !c    8  + XX c   [&_ * c -) "')- * 
8 "   c   [)X c  @ c _ * "')- $" )'"Y * 
8 "     )X @  e   _ * :' -)(  * 
8 "   c   [ @  + a + e &_ * "')- @  * 
8 "    c c   [ @  + a + e  _ *  '" "')-    * 
8 "   @ a   + XX X )X @  e   8 _ * @   ' -)(   * 
8 "  !c    8   [ @  c &_ * c -)')V"  * 
8 "     [ @  c  _ *  ')V"  * 
8 "  !c    8  )e  88+ %_ * c -)$(  * 
8 "  !e  88+ _ * e  * 
8 "  !X 8 @  88    [X +    c  e @ a   _ * XV-;)') -    "' (    *








8 "
8 "
8 "
8 "

  @ a 
    [ @
 !X a 
  @ a 

 +
 c
)e
 )c

"   @ a   )c
>3 8 "  !
 X   + 
  c c 
8 "   

XX  X
 
 88+
e 



)e  88+  _ * @    $(  * 
[X +  &
 _ *  ')V"  -    * 
%_ * X   )(  * 
X a @ + a)X @  e   8 _ * @  " $(V'T  '  

 @  e
X )X 
a  @ a   +
 [  [X +    c
e 

¬z}

  _ * @  " $(V'T ' -)(  * 
e  _ *   -  $( * 
 e @ a   _ *  
' " "')--   "' (     * 


public SharedConstants() {
    super();
}
/**
 * Similar to computeDisplayRectangle, but with a Dimension type return value.
 * @return java.awt.Dimension
 */
public static java.awt.Dimension computeDisplayDimension(String string, java.awt.FontMetrics fontMetrics) {
    java.awt.Dimension size = new java.awt.Dimension();
    int numRows = string.length() / SharedConstants.getNotificationMessagePreferredWidthInCharacters()
    size.width = fontMetrics.stringWidth(string) / (numRows + 1);
    size.height = fontMetrics.getHeight() * (numRows * 2 + 3);
    return size;
}
/**
 * Insert the method’s description here.
 * @return javax.swing.border.Border
 */
public static javax.swing.border.Border createDialogBorder(java.awt.Color colour) {
    Border matteBorder, etchedBorder, compoundBorder, returnBorder;
    matteBorder = javax.swing.borderFactory.createMatteBorder(getBORDER_WIDTH(), getBORDER_WIDTH(),
        getBORDER_WIDTH(), getBORDER_WIDTH(), colour);
    etchedBorder = javax.swing.borderFactory.createEtchedBorder();
    compoundBorder = javax.swing.borderFactory.createCompoundBorder(matteBorder, etchedBorder);
    returnBorder = javax.swing.borderFactory.createCompoundBorder(compoundBorder, matteBorder);
    return returnBorder;
}
/**
 * Constructs a representative, unique ID for a document based on the name of the document and that of its sender, i.e. a client.
 * @return java.lang.String
 */
public static String createFullQualifiedDocumentName(String senderName, String documentName) {
    return ("Sender:" + senderName + ".Document:" + documentName);
}
/**
 * Insert the method’s description here.
 * @return int
 */
public final static int getBORDER_WIDTH() {
    return BORDER_WIDTH;
}
/**
 * Insert the method’s description here.
 * @return java.lang.String
 */
public final static java.lang.String getClientAddedDocuments() {
}
return CLIENT_ADDED_DOCUMENTS;

/**
 * Insert the method's description here.
 * Creation date: (11-07-00 21:13:56)
 */
public final static java.lang.String getCLIENT_ADDED_MESSAGE() {
    return CLIENT_ADDED_MESSAGE;
}

/**
 * Insert the method's description here.
 * Creation date: (19-09-00 16:23:30)
 */
public final static java.lang.String getCLIENT_JOINED_GROUP() {
    return CLIENT_JOINED_GROUP;
}

/**
 * Insert the method's description here.
 * Creation date: (01-11-00 01:20:01)
 */
public final static java.lang.String getCLIENT_LEFT() {
    return CLIENT_LEFT;
}

/**
 * Insert the method's description here.
 * Creation date: (26-07-00 01:41:43)
 */
public final static java.lang.String getCLIENT_LEFT_GROUP() {
    return CLIENT_LEFT_GROUP;
}

/**
 * Insert the method's description here.
 * Creation date: (15-07-00 16:26:04)
 */
public final static java.lang.String getCLIENT_REMOVED_DOCUMENT() {
    return CLIENT_REMOVED_DOCUMENT;
}

/**
 * Insert the method's description here.
 * Creation date: (15-07-00 15:33:51)
 */
public final static java.lang.String getCLIENT_REMOVED_LOCAL_DOCUMENTS() {
    return CLIENT_REMOVED_LOCAL_DOCUMENTS;
}

/**
 * Insert the method's description here.
 * Creation date: (19-08-00 12:26:31)
 */
public final static java.lang.String getCOMPRESSION_LEVEL() {
    return COMPRESSION_LEVEL;
}

/**
 * Insert the method's description here.
 * Creation date: (03-07-00 09:29:42)
 */
public final static java.lang.String getCONNECTION_TYPE() {
    return CONNECTION_TYPE;
}
/**
 * Insert the method’s description here.
 * Creation date: (06-08-00 20:51:36)
 * @return java.lang.String[]
 */
224 public final static java.lang.String[] getCONNECTION_TYPES() {
    return CONNECTION_TYPES;
226 }
/**
 * Insert the method’s description here.
 * Creation date: (11/23/00 1:56:30 PM)
228 * @return java.lang.String
 */
232 public final static java.lang.String getCOULD_NOT_EXIT_PROPERLY() {
    return COULD_NOT_EXIT_PROPERLY;
234 }
/**
 * Insert the method’s description here.
 * Creation date: (16-07-00 11:32:37)
 * @return java.lang.String
 */
240 public final static java.lang.String getDefault_CLIENT_NAME() {
    return DEFAULT_CLIENT_NAME;
242 }
/**
 * Insert the method’s description here.
 * Creation date: (19-08-00 12:20:56)
 * @return int
 */
248 public final static int getDefault_COMPRESSION_LEVEL() {
    return DEFAULT_COMPRESSION_LEVEL;
250 }
/**
 * Insert the method’s description here.
 * Creation date: (03-07-00 16:51:41)
 * @return java.lang.String
 */
256 public final static java.lang.String getDefault_CONNECTION_TYPE() {
    return DEFAULT_CONNECTION_TYPE;
258 }
/**
 * Insert the method’s description here.
 * Creation date: (12-09-00 10:17:15)
 * @return java.lang.String
 */
264 public final static java.lang.String getDefault_REGISTRY_PORT() {
    return DEFAULT_REGISTRY_PORT;
266 }
/**
 * Insert the method’s description here.
 * Creation date: (03-07-00 09:26:30)
 * @return java.lang.String
 */
272 public final static java.lang.String getDefault_SERVER_NAME() {
    return DEFAULT_SERVER_NAME;
274 }
/**
 * Insert the method’s description here.
 * Creation date: (03-07-00 09:26:44)
 * @return int
 */
278 public final static int getDefault_SERVER_PORT() {
    return DEFAULT_SERVER_PORT;
282 }
/**
 * Insert the method’s description here.
 */

* Creation date: (12-07-00 12:11:27)

```java
public final static java.lang.String getDELETE_MESSAGE() {
    return DELETE_MESSAGE;
}
```

* Insert the method's description here.
* Creation date: (11-07-00 14:32:07)

```java
public final static java.lang.String getDISCUSSION_UPDATE_FROM_CLIENT() {
    return DISCUSSION_UPDATE_FROM_CLIENT;
}
```

* Insert the method's description here.
* Creation date: (24-08-00 08:41:42)

```java
public final static java.lang.String getDISCUSSIONS_FILE_NAME() {
    return DISCUSSIONS_FILE_NAME;
}
```

* Insert the method's description here.
* Creation date: (07-07-00 13:46:31)

```java
public final static java.lang.String getGROUP_CHANNEL_NAME() {
    return GROUP_CHANNEL_NAME;
}
```

* Insert the method's description here.
* Creation date: (04-07-00 17:33:19)

```java
public final static java.lang.String getGROUP_DIRECTORY() {
    return GROUP_DIRECTORY;
}
```

* Insert the method's description here.
* Creation date: (26-07-00 16:40:54)

```java
public final static java.lang.String getGROUP_EXISTS_MESSAGE() {
    return GROUP_EXISTS_MESSAGE;
}
```

* Insert the method's description here.
* Creation date: (30-06-00 22:43:36)

```java
public final static java.lang.String getGROUP_MANAGEMENT_SESSION_NAME() {
    return GROUP_MANAGEMENT_SESSION_NAME;
}
```

* Insert the method's description here.
* Creation date: (26-07-00 16:39:50)

```java
public final static java.lang.String getGROUP_NAME_MISSING_MESSAGE() {
    return GROUP_NAME_MISSING_MESSAGE;
}
```

* Insert the method's description here.
* Creation date: (10-09-00 20:04:02)
public final static java.lang.String getHTTP() {
    return HTTP;
}
/**
   * Insert the method’s description here.
   * Creation date: (10-09-00 20:03:32)
   */
public final static java.lang.String getHTTP_ARGUMENT() {
    return HTTP_ARGUMENT;
}
/**
   * Insert the method’s description here.
   * Creation date: (08-08-00 16:57:01)
   */
public final static java.lang.String[][] JSOT_EXCEPTION_MESSAGE() {
    return JSOT_EXCEPTION_MESSAGE;
}
/**
   * Insert the method’s description here.
   * Creation date: (09-08-00 07:59:38)
   */
public final static java.util.Hashtable JSOT_EXCEPTION_MESSAGES_TABLE() {
    if ( JSOT_EXCEPTION_MESSAGES_TABLE.size() == 0 )
    {
        for ( int i = 0 ; i < JSOT_EXCEPTION_MESSAGE.length; i++ )
        {
            JSOT_EXCEPTION_MESSAGES_TABLE.put( JSOT_EXCEPTION_MESSAGE[i][0], JSOT_EXCEPTION_MESSAGE[i][1] );
        }
    }
    return JSOT_EXCEPTION_MESSAGES_TABLE;
}
/**
   * Insert the method’s description here.
   * Creation date: (26-07-00 16:57:40)
   */
public final static java.lang.String getMAXIMUM_PARTICIPANTS_FIELD_INVALID_MESSAGE() {
    return MAXIMUM_PARTICIPANTS_FIELD_INVALID_MESSAGE;
}
/**
   * Insert the method’s description here.
   * Creation date: (10-07-00 02:26:24)
   */
public final static java.lang.String getMESSAGE() {
    return MESSAGE;
}
/**
   * Insert the method’s description here.
   * Creation date: (06-07-00 23:48:33)
   */
public final static java.lang.String getNEW_DOCUMENT() {
    return NEW_DOCUMENT;
}
/**
   * Insert the method’s description here.
   * Creation date: (19-09-00 18:28:18)
   */
public final static java.lang.String getNEW_DOCUMENTS() {
return NEW.Documents;

/**
 * Insert the method's description here.
 * Creation date: (10-08-00 06:56:22)
 */
public final static int getNOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS() {
  return NOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS;
}

/**
 * Insert the method's description here.
 * Creation date: (12-09-00 10:16:20)
 */
public final static java.lang.String getREGISTRY_PORT() {
  return REGISTRY_PORT;
}

/**
 * Insert the method's description here.
 * Creation date: (26-07-00 02:41:39)
 */
public final static java.lang.String getREMOVE_DOCUMENT() {
  return REMOVE_DOCUMENT;
}

/**
 * Insert the method's description here.
 * Creation date: (19-09-00 19:18:50)
 */
public final static java.lang.String getREMOVE_DOCUMENTS() {
  return REMOVE_DOCUMENTS;
}

/**
 * Insert the method's description here.
 * Creation date: (04-07-00 17:33:01)
 */
public final static java.lang.String getREQUEST_ADD_GROUP() {
  return REQUEST_ADD_GROUP;
}

/**
 * Insert the method's description here.
 * Creation date: (03-07-00 12:39:46)
 */
public final static java.lang.String getREQUEST_GROUP_DIRECTORY() {
  return REQUEST_GROUP_DIRECTORY;
}

/**
 * Insert the method's description here.
 * Creation date: (10-07-00 02:16:46)
 */
public final static java.lang.String getREQUEST_MESSAGE() {
  return REQUEST_MESSAGE;
}

/**
 * Insert the method's description here.
 * Creation date: (27-07-00 00:25:46)
 */
public final static java.lang.String getREQUEST_REMOVE_GROUP() {
  return REQUEST_REMOVE_GROUP;
}
/**
 * Insert the method’s description here.
 * Creation date: (09-07-00 21:56:56)
 * @return java.lang.String
 */
public final static java.lang.String getREQUEST_TOPIC_TREE() {
    return REQUEST_TOPIC_TREE;
}
/**
 * Insert the method’s description here.
 * Creation date: (03-07-00 14:30:27)
 * @return java.lang.String
 */
public final static java.lang.String getSERVER_NAME() {
    return SERVER_NAME;
}
/**
 * Insert the method’s description here.
 * Creation date: (03-07-00 14:30:42)
 * @return java.lang.String
 */
public final static java.lang.String getSERVER_PORT() {
    return SERVER_PORT;
}
/**
 * Insert the method’s description here.
 * Creation date: (10-09-00 20:03:50)
 * @return java.lang.String
 */
public final static java.lang.String getSOCKET() {
    return SOCKET;
}
/**
 * Insert the method’s description here.
 * Creation date: (10-09-00 20:03:13)
 * @return java.lang.String
 */
public final static java.lang.String getSOCKET_ARGUMENT() {
    return SOCKET_ARGUMENT;
}
/**
 * Insert the method’s description here.
 * Creation date: (09-07-00 22:29:03)
 * @return java.lang.String
 */
public final static java.lang.String getTOPIC_TREE() {
    return TOPIC_TREE;
}
/**
 * Insert the method’s description here.
 * Creation date: (11-07-00 21:31:23)
 * @return java.lang.String
 */
public final static java.lang.String getTOPIC_TREE_UPDATE() {
    return TOPIC_TREE_UPDATE;
}
/**
 * Insert the method’s description here.
 * Creation date: (08-07-00 17:26:19)
 * @return java.lang.String
 */
public final static java.lang.String getWORKGROUP_CHANNEL_NAME() {
    return WORKGROUP_CHANNEL_NAME;
}
/**
 * Insert the method’s description here.
 */

@  

```java
package peerviewmisc;

import java.io.*;
import java.util.Hashtable;
import com.sun.media.jsdt.*;
import java.util.Vector;
import java.util.Date;

/**
 * This class implements a representation of a PeerView group.
 * @author <empty>
 */
public class WorkGroup implements Serializable {
    private java.lang.String groupName = "";
    private java.lang.String description = "";
    private com.sun.media.jsdt.URL sessionURL = null;
    private long totalDataVolume = 0;
    private java.util.Date creationDate = new Date();
    private java.lang.String creator = "";
    private final static java.lang.String fieldNames[] = {
        "GroupName", "Description",
        "NumberOfDocuments", "TotalDataVolume", "ActiveParticipants",
        "CreationDate", "CreatedBy"};
    private java.util.Hashtable documents = new Hashtable();
    private java.util.HashSet participants = new java.util.HashSet();
    private int numberOfDocuments = 0;
    /**
     * WorkGroup constructor comment.
     */
    public WorkGroup() {
        super();
    }
    /**
     * Insert the method's description here.
     * @param workGroup the Group object
     */
    public WorkGroup(WorkGroup workGroup) {
        setCreationDate(workGroup.getCreationDate());
    }
}
```

Listing C.49: WorkGroup.java
```java
42    setCreator( workGroup.getCreator() );
43    setDescription( workGroup.getDescription() );
44    setDocuments( (HashTable) workGroup.getDocuments().clone() );
45    setGroupName( workGroup.getGroupName() );
46    setParticipants( (java.util.HashSet) workGroup.getParticipants().clone() );
47    setSessionURL( workGroup.getSessionURL() );
48    setTotalDataVolume( workGroup.getTotalDataVolume() );
49  }
50  /**
51   * Insert the method's description here.
52   * Creation date: (16-07-00 11:01:58)
53   */
54  public java.util.Date getCreationDate() {
55      return creationDate;
56  }
57  /**
58   * Insert the method's description here.
59   * Creation date: (17-07-00 17:12:13)
60   */
61  public java.lang.String getCreator() {
62      return creator;
63  }
64  /**
65   * Insert the method's description here.
66   * Creation date: (06-07-00 21:32:10)
67   */
68  public java.lang.String getDescription() {
69      return description;
70  }
71  /**
72   * Insert the method's description here.
73   * Creation date: (19-09-00 17:40:00)
74   */
75  public java.util.Hashtable getDocuments() {
76      return documents;
77  }
78  /**
79   * Insert the method's description here.
80   * Creation date: (17-07-00 19:32:07)
81   */
82  public final static java.lang.String[] getFieldNames() {
83      return fieldNames;
84  }
85  /**
86   * Insert the method's description here.
87   */
88  public java.util.Vector getGroupAsVector() {
89      Vector v = new Vector();
90      v.addElement( getGroupName() );
91      v.addElement( getDescription() );
92      v.addElement( String.valueOf( getNumberOfDocuments() ) );
93      v.addElement( String.valueOf( getTotalDataVolume() ) );
94      v.addElement( String.valueOf( participants.size() ) );
95      v.addElement( getCreationDate().toString() );
96      v.addElement( getCreator() );
97  }
```
public java.lang.String getGroupName() {
    return groupName;
}
/**
 * Insert the method's description here.
 * Creation date: (05-07-00 21:25:46)
 */

public int getNumberOfDocuments() {
    return numberOfDocuments;
}
/**
 * Insert the method's description here.
 * Creation date: (19-09-00 18:48:20)
 */

public java.util.HashSet getParticipants() {
    return participants;
}
/**
 * Insert the method's description here.
 * Creation date: (19-09-00 17:53:36)
 */

public com.sun.media.jsdt.URLString getSessionURL() {
    return sessionURL;
}
/**
 * Insert the method's description here.
 * Creation date: (07-07-00 20:11:38)
 */

public long getTotalDataVolume() {
    return totalDataVolume;
}
/**
 * Create and return unique identifier for the JSIT channel created to convey information pertaining to
 * workgroup.
 * Creation date: (13-08-00 01:42:06)
 */

public String getWorkGroupChannelID() {
    return getWorkGroupID() + SharedConstants.getWORKGROUP_CHANNEL_NAME();
}
/**
 * Same as getWorkGroupChannelID(), only with a specifiable suffix.
 * Creation date: (13-08-00 01:43:06)
 */

public String getWorkGroupChannelID(String suffix) {
    return getWorkGroupID() + suffix;
}
/**
 * Create and return a unique identifier for the workgroup
 * @return java.lang.String
 */
public String getWorkGroupID()
{
    return getGroupName() + getDescription() + getCreator();
}
/**
 * Insert the method's description here.
 * @param newCreationDate java.util.Date
 */
public void setCreationDate(java.util.Date newCreationDate) {
    creationDate = newCreationDate;
}
/**
 * Insert the method's description here.
 * @param newCreator java.lang.String
 */
public void setCreator(java.lang.String newCreator) {
    creator = newCreator;
}
/**
 * Insert the method's description here.
 * @param newDescription java.lang.String
 */
public void setDescription(java.lang.String newDescription) {
    description = newDescription;
}
/**
 * Insert the method's description here.
 * @param newDocuments java.util.Hashtable
 */
public void setDocuments(java.util.Hashtable newDocuments) {
    documents = newDocuments;
}
/**
 * Insert the method's description here.
 * @param newGroupName java.lang.String
 */
public void setGroupName(java.lang.String newGroupName) {
    groupName = newGroupName;
}
/**
 * Insert the method's description here.
 * @param newNumberOfDocuments int
 */
public void setNumberOfDocuments(int newNumberOfDocuments) {
    numberOfDocuments = newNumberOfDocuments;
}
/**
 * Insert the method's description here.
 * @param newParticipants java.util.HashSet
 */
public void setParticipants(java.util.HashSet newParticipants) {
    participants = newParticipants;
}
* Insert the method’s description here.
* Creation date: 07-07-00 20:11:38
* @param newSessionURL com.sun.media.jsdt.URLString
 */

public void setSessionURL(com.sun.media.jsdt.URLString newSessionURL) {
    sessionURL = newSessionURL;
}
/**
* Insert the method’s description here.
* Creation date: 08-07-00 20:06:01
* @param nwtotalDataVolume long
*/

public void setTotalDataVolume(long nwtotalDataVolume) {
    totalDataVolume = nwtotalDataVolume;
}
Appendix D

PeerView help system

The following is a paper transcript of the online help system that ships with the PeerView binaries and is available as a separate World Wide Web document from [54]. Note that this material is intended for electronic publication and therefore may occasionally lend itself poorly to paper reproduction. It is included here for the sake of completeness and to serve as a user’s manual for PeerView.
D.1 Introduction

Introduction

Welcome to the PeerView online help system.

PeerView is a system for artifact rendering and group review. It accomplishes its purpose through the use of a scalable, navigable panorama to which documents can be added and removed. Users of PeerView can join groups in order to share the documents in their panoramas. The documents can then be discussed by the group members using the discussion forum associated with each document. In addition, PeerView is customizable and can be configured to suit many different usage scenarios.
D.2 Add documents

Add documents

The "Add documents" box is used for adding documents to the panorama. You activate this box by selecting "Add documents" from the "File" menu or by pressing the button on the toolbar.

The features of the "Add documents" box are described below:

Look in

A listing of the drives that are visible from the location where your PeerView client was invoked. The active directory will be indicated by an expanded path from the root, i.e. the drive name, down to the directory itself.

The contents of the active directory, i.e. the files and subdirectories it contains, are displayed in the list box below the "Look in" label.

File name

The name of the file selected in the list box above the "File name" label. If multiple files are selected, this field will contain the name of the last file selected.

Files of type

A file filter indicating which type of file will be displayed in the list box above the "File name" label. Select this to exclude all other files than those relevant to your browsing.

Selected documents

A list of the documents selected from the leftmost list box. You transfer files to and from the right list box by first selecting one or more files in either box and then pressing one of the arrows between the list boxes to indicate the direction of transfer.

Buttons

Here's a brief legend to the buttons found in the "Add files" box:
Up one level. Makes the parent directory of the current directory active.

Home. Makes your home directory the active directory.

Create new folder. Creates a new folder in the active directory.

Click OK to add the files in the "Selected documents" list box to the panorama.

Click Cancel to close the "Add files" dialog without adding any documents to the panorama. Your selections, if any, will be discarded.
D.3 Remove documents

Remove documents

The "Remove documents" box is used for removing documents from the panorama. It can be activated by choosing "Remove documents" from the "File" menu or by pressing the button on the toolbar.

The features of the "Remove documents" box are described below:

Currently displayed

This list box shows which of your files are currently displayed in the panorama. Note that this does not include other group members' documents as these cannot be removed by you, only by the authors themselves. You move items from the "Currently displayed" box to the "To be removed" box by selecting one or more file names and pressing the button.

To be removed

This list box shows which of your files currently displayed in the panorama are to be removed. You move files from this box to the "Currently displayed" box selecting one or more and pressing the button.

Buttons

Remove

Close the remove box and remove all documents listed in the "To be removed" box from the panorama.

Cancel

Close the remove box without removing any documents from the panorama. Your selection, if any, will be discarded.
D.4 Panorama

Panorama

The panorama occupies the center portion of the PeerView client main window. It is where documents are placed for inspection, navigation and manipulation. The panorama is therefore usually the area where most interaction between you and the application takes place. For that same reason, the panorama can be controlled in a variety of ways as described below:

Resizing the panorama

You can adjust the size of the panorama relative to the other elements of the main window by dragging the divider bar.

Scaling the panorama

You can adjust the magnification of the panorama by scaling it up or down:

1) Position the mouse cursor on the panorama background, i.e. outside any document.
2) Press and hold the right mouse button.
3) Move the cursor left to increase magnification or right to decrease it.
4) Release the mouse button when you have reached the desired magnification.

Moving the panorama

You can move the panorama, or rather, its contents, using the mouse:

1) Position the mouse cursor on the panorama background, i.e. outside any document.
2) Press and hold the left mouse button.
3) Move the cursor to the panorama's new location.
4) Release the left mouse button.

Zooming to overview

You can get an overview of the contents of the panorama by zooming to overview. This will scale the panorama to a magnification that allows all of the contents to fit in the visible portion of the panorama:

1) Position the mouse cursor on the panorama background, i.e. outside any document
2) Double click the left mouse button

Activating a document

To activate a document, do the following:
1) Position the mouse cursor on the document in question.

2) Double click the left mouse button

This will cause the PeerView application to center the document in the panorama while zooming to a magnification that allows the document to fill as much of the visible portion of the panorama as possible.

At the same time, the client will request the topic tree for that document if one is available, i.e. if there is a connection to an active PeerView server. When received by the client, the topic tree will be displayed in the discussion forum area at the bottom of the main window.

**Document operations**

You can carry out various operations that affect one or all of the documents in the panorama. This is achieved using the panorama pop-up menu:

1) Position the mouse cursor on any document.

2) Click the right mouse button.

3) Move the mouse cursor to the item you wish to select.

4) Click the left mouse button

Here is a brief explanation of each operation that can be selected from the menu:

**Resize**

After selecting this item, the lower right corner of the document's frame will "attach" itself to the mouse cursor and follow its movements around the panorama. To release the frame, press the left mouse button. The document will retain its new size until you resize it again or remove it from the panorama.

**Move**

After selecting this item, the document will "attach" itself to the mouse cursor by its upper left corner and follow its movements around the panorama. To release the document, press the left mouse button. The
document will remain at its new size until you move it again or, if its lock is disabled, until you arrange the documents.

**Lock**
If a document's lock is enabled, the document will remain at its position even if you select "Arrange all" from the pop-up menu. If the lock is disabled, the document will be repositioned at an appropriate location in the arrangement when you select "Arrange all".

**Copy text**
Copy any selected text to the clipboard. This item is disabled if the document does not contain text.

**Unlock all**
Disable all document locks.

**Arrange all**
Arrange any non-locked documents using the active layout manager. The panorama will be zoomed to overview after selecting this item.

**Auto-lock all**
If this item is checked, any documents moved will have their lock enabled by default. Note that this does not affect documents moved prior to checking this item.
D.5 Discussion forum

Discussion forum

The discussion forum is the split pane display at the bottom of the PeerView main window. It is where discussion on documents take place and you can browse, add, delete and view messages using the controls in the forum.

The below describes the features of the discussion forum.

**Topic tree**

The structure to the left of the discussion forum which resembles the file system browsers available on many systems is called a "topic tree". It lists the "threads" of discussion, i.e. the topics that are being and have been discussed throughout the document's history as well as "messages", i.e. the individual contributions to the topics discussed.

The three small buttons in the top right corner of the discussion forum are used as follows:

- **Compose new message.** To use this command, first select an item from the topic tree. If you select a topic, i.e. an item marked by the graphic, your message will be the first in a new subtopic to the selected topic. That subtopic will have the same title as your message so you should choose something that appropriately designates the entire thread of discussion you are about to initiate.

  If the item you select is a message (an item marked by ), your new message will be inserted immediately below the selected message.

- **Delete a message.** You must select a message before clicking this button. If the message is authored by you and has not yet been responded to, you will be asked to confirm its deletion. If either of those conditions is not met, the message will not be deleted and you will therefore not be asked for confirmation. The reason for imposing this policy is to ensure that the topic tree reflects the actual flow of discussion and not an arbitrary sequence of additions and deletions.

- **View the properties of a selected message.** First select an item, then click this button to view details about it such as author's name, contents, date of creation and so forth.
Message display

The message display is the rightmost portion of the discussion forum where messages are displayed. Each time you select a message from the topic tree, the client request the contents of that message from the server and displays them in the message display.

Adjusting the size of the discussion forum

To adjust the size of the discussion forum itself, use the divider bar which separates it from the panorama.

To adjust the size of the topic tree and message display area, use the divider bar that separates the two.
D.6 Group directory

Group directory

The group directory box is used for joining, creating, deleting and editing groups. You activate the group directory by selecting "Groups" from the setup menu or by pressing the button on the toolbar.

The features of the group directory box are described below:

- **Group listing**

  The top portion of the group directory box is occupied by a listing of the groups available to you. This listing is requested from the server each time the group directory box is opened and is therefore always up-to-date. If for some reason the server is unavailable when the box is opened, the listing will be empty.

  Each row in the listing gives the following information on a single group:

  - **Group name**
    The name of the group.

  - **Group description**
    A description of the group, entered when the group was created.

  - **# of documents**
    The total number of documents contributed to the group by its members.

  - **Total data volume**
    The combined size of all documents in the group. This will give you an indication of how much network traffic you can expect when joining a given group.
**Active participants**

The number of group members currently taking part in the group.

**Creation date**

The date and time when the group was created.

**Created by**

The name of the person who created the group.

**Resizing columns**

You can adjust the width of the columns in the group listing. Place the mouse cursor over the indentation that separates two columns ( ). This should cause the cursor to change appearance and you can now drag the indentation to either side to resize the columns it separates.

**Buttons**

The following is a brief legend to the buttons available in the group directory box. Note that all of these buttons, except "Close" are disabled if there is no connection to the server since that makes it impossible to execute operations on the group listing.

- **Join group**

Join the group selected in the group listing. A message box will appear if none is selected. Note that the group join operation may take a while to complete as a fair amount of network traffic may be generated when the other group members are requested to send their documents to you and vice versa.

- **Create group**

Create a new group. You will be asked to give the name and description of the group after pressing this button. The new group will appear in the group listing.

- **Edit group**

If a group has been selected from the group listing, you will be asked to edit its name and description and confirm/discard your changes. If no group is selected, a message box to that effect will appear.

- **Delete group**

If a group has been selected from the group listing, you will be asked to confirm or cancel its deletion. If no group has been selected, a message box to that effect will appear.

- **Close**

Close the group directory box.
Preferences

The preferences box is used for controlling the behaviour of the PeerView application. You activate the preferences box by selecting "Preferences" from the "Setup" menu or by pressing the button on the toolbar.

Display setup

![Preferences dialog box]

The display setup pane is used for configuring the behaviour of the PeerView panorama. The following can be controlled using the items on this pane:

**Display quality**

The quality in which documents are rendered. The higher the quality, the better the resolution and appearance of the documents in the panorama but also, the lower the speed at which the panorama can be navigated and scaled.

**Layout scheme**

The algorithm, i.e. method, used to layout the documents in the panorama. Press the "Customize" button to adjust the settings of the layout scheme selected from the drop-down box.

**Update interval**

The interval at which PeerView will check the files you have added to the panorama for changes. Any changes will be broadcast across the network to the other group members. The lower this setting, the more up-to-date documents you can expect other group members to have. However, low settings can also cause a noticeable load on your machine and network.

**Animation speed**

The speed at which zoom operations on the panorama are performed.

**Personal setup**
The "Personal" pane is used for setting properties that identify you to other users. The following items are available:

**Name**

Your name. This needn't be your real name, any pseudonym will do, but do not change it too often since any messages composed by you will be identifiable to others by your name only.

**Signature**

A signature text that will be automatically appended to all messages you compose.

**Server setup**

The "Server" pane is used for setting the parameters used when connecting to the server at startup. The following items are available:
Server name

The name of the server you wish to connect to at startup. Enter "localhost" as in the image above to connect to a server running locally, i.e. on the machine you are running the PeerView client from. If you are connecting to a remote server, enter its DNS address, not its IP, as in:

www.myserver.myinstitution.edu

Server port

The server port that the PeerView client should connect to on startup.

IMPORTANT NOTE: You must ensure that this port is open on the server machine, i.e. exempt from any firewall protection or PeerView's connection attempt will fail.

Also, port number 4561 must always be open on the server machine, regardless of the port number you enter in this field. PeerView thus requires (at least) two ports to be open and accessible at the server end.

Connection type

The type of connection protocol to use. For connections over the internet, you should use a HTTP protocol. For LAN and localhost uses, you should probably select socket, although HTTP may also apply on some networks.

Connect to server

This feature is not available in the beta release (#0.900) of PeerView.

Buttons

Press "OK" to close the preferences box and save your changes.

Press "Cancel" to close preferences box without saving your changes.
D.8 Using a divider bar

Using a divider bar

A horizontal divider bar looks like this:

```
.................................................................................................................. (appearance may be slightly different on your platform)
```

and is used for adjusting the size of the areas adjacent to it. To use a divider bar, place the mouse cursor over it, press the left mouse button and drag the bar up or down while holding the button. Alternatively, you can press either of the two arrow symbols at the left end of the divider bar. Pressing the upwards pointing arrow will minimize the area above the divider while pressing the downwards pointing arrow will maximize it.

The above instructions also apply to vertical divider bars although the bar should of course be dragged horizontally rather than vertically.
Bibliography


[28] National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, NCSA Habanero®, Website accessible at http://havefun.ncsa.uiuc.edu/habanero/.


[34] Human-Computer Interaction Lab, University of Maryland, Welcome to Jazz!, Website accessible at http://www.cs.umd.edu/hcil/jazz/.


[54] Lars Yde, The PeerView Website, Website accessible at http://www.diku.dk/research-groups/performance-engineering/PeerView/.


