Diploma Thesis

GOMES

An Object-Oriented **G**UI for the **O**bject **M**odel Multi-User **E**xtended File**s**ystem

Beat Signer, IIIC
beat.signer@switzerland.org

July 28th 1999

Institute for Information Systems
Swiss Federal Institute of Technology (ETHZ)

Diploma Professor:
Prof. Moira C. Norrie

Supervisor:
Gabrio Rivera

Abstract

Today's file systems often lack flexibility as a consequence of being modeled too close to the underlying physical storage structure of their files. The object model multi-user extended file system (OMX-FS) is a new vision of a file system, providing more functionality and flexibility than most file systems currently do. It integrates database functionality into the operating system and strictly separates the physical storage structure of files from their logical use. The goal of this diploma thesis was the design and implementation of GOMES, an object-oriented graphical user interface for the OMX-FS, offering easy access to the full power of the OMX-FS file system. Being a Java application GOMES is highly portable. It is based on XML-RPC, a networking protocol encoding remote method invocations with the help of the extensible markup language. Using XML-RPC a remote object mechanism is designed and employed in order to communicate in a machine and programming language independent manner.

Contents

Chapter 1

Introduction

The object model multi-user extended file system OMX-FS described in [15], is an approach to a new file system architecture directly integrating database functionality into the operating system. OMX-FS is based on the object model OM [13, 14] and fully embedded into the Oberon System 3 operating system [2, 18]. The strength of the OMX-FS lies in an additional level of abstraction in order to clearly differentiate the physical storage structure of files from their logical usage. Files are moreover treated as objects and provide the full functionality of the underlying OM model as for example the distinction between typing and classification and the concepts of collections and associations. As a result, users have much more power to logically order their files. By the strict distinction between the physical storage structure of the files, which is totally hidden to the user, and their logical use, there exist no more pathnames representing the storage location of files in the user interface of the OMX-FS. Therefore the sharing of parts of the file system is a lot easier than in a common file system.

The goal of this diploma thesis was the design and implementation of GOMES, an object-oriented graphical user interface for the OMX-FS, offering easy access to the full strength of the OMX-FS file system. In a first step, we had to investigate how computer users generally work with graphical user interfaces and elaborate the most important GUI design principles to be considered when implementing a graphical user interface for a file system. The result of this case study is presented in Chapter 3. After inspecting existing visualization techniques of today's file systems and analyzing their advantages and disadvantages, an optimal way to visualize the characteristic operations and functionalities of the OM model (such as collections and associations) the OMX-FS is based on had to be found. In Chapter 4 the process of designing adequate graphical representations for the new OMX-FS concepts is summarized and the current implementation of GOMES is motivated.

The overall architecture of GOMES and the tasks of the different layers the system is based on is discussed in Chapter 2.

Since GOMES is a Java [3, 4, 8, 9, 10] client application, whereas the OMX-FS server is implemented using the Oberon programming language and normally runs on a remote machine, a solution allowing the Java virtual machine to communicate with the remote OMX-FS system had to be found. Our approach, consisting of mapping the OMX-FS objects on the Java client side by a remote object mechanism based on the remote procedure call protocol using the extensible markup language (XML-RPC), is presented in Chapter 5.

In the Appendix B interfaces of all classes used by GOMES and a short description of each class can be found, allowing to easily extend the current system.

Chapter 2

Architecture of GOMES

GOMES' overall architecture can be grouped into four main layers (see Figure 2.1). The bottom layer, the XML-RPC Layer, is responsible for the communication of GOMES and the OMX-FS server. It implements the XML-RPC protocol, a remote procedure call protocol allowing to invoke remote methods on objects of the OMX-FS (the XML-RPC protocol is explained in Chapter 5).

Based on the facility of remote procedure calls introduced by the XML-RPC layer, the second layer (Core Layer) implements an object serialization mechanism to model remote objects for all objects of the OMX-FS. This allows to use the objects of the OMX-FS like local Java objects. The whole mechanism of remote objects is completely transparent to the programmer. After method invocation on a wrapping remote object of the Core Layer, it will redirect the corresponding method call to the OMX-FS by making use of the underlying XML-RPC layer. The OMX-FS server will invoke the method on its Oberon object and return the result of the remote procedure call to the corresponding object in the Core Layer of GOMES. The Core Layer wrapper object will finally return the result similar to a local method call, i.e. the programmer will not notice anything about the network communication caused by the method call. If the result of a remote procedure call is an OMX-FS object and not a scalar value, the Core Layer will generate a new wrapper object on the fly (for more information on modeling remote objects see Chapter 5).

The two top layers, the Model Layer and the View Layer are responsible for the graphical representations of GOMES. The components of the Model Layer are directly based on the remote objects provided by the underlying Core Layer. Since the OMX-FS is a multi-user system, an update mechanism had to be implemented guaranteeing that the objects of the Model Layer are always consistent to the objects of the OMX-FS system. All components of the View Layer have their corresponding models in the Model Layer and rely on the Model-View-Controller design principle (MVC) to achieve consistency (for more

information on the usage of design patterns see [7]). The concepts of the View Layer are outlined in Chapter 4.

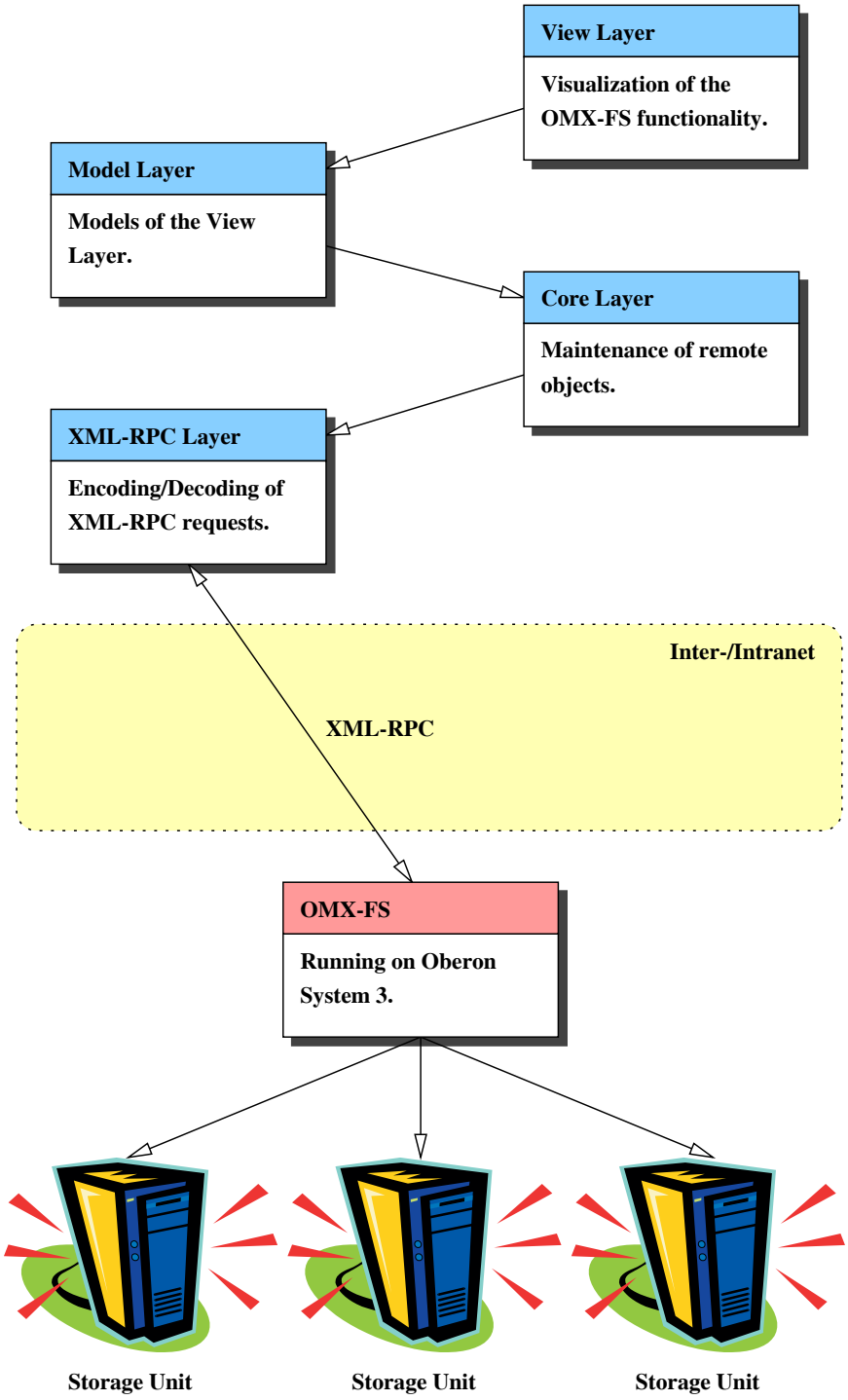

Figure 2.1: Architecture of GOMES

The clear distinction between the different layers allows to replace a layer without affecting the remaining layers. It is hence possible to use an alternative networking and object serialization protocol by just exchanging the XML-RPC Layer.

For each of these four main layers the corresponding Java package can be found in the API reference (see Appendix B). The functionality of the XML-RPC Layer is implemented in the `xmlrpc` package, the `gomes.core` package is responsible to provide the functionality of the Core Layer, while the packages `gomes.model` and `gomes.view` are the implementations of the two top layers.

Chapter 3

General Design Principles

In this chapter some design principles are explained which influenced the development of the graphical user interface for the OMX-FS. It is not a complete guidance for the design of a graphical user interfaces, but rather some aspects of good user interface design are presented. For further information about the design of user interfaces see [5, 6, 11, 17].

When designing a user interface, developers always have to ask themselves how potential users will use their applications. Are they going to specify actions and then select the objects to be processed as often used in command-line interfaces and menu-driven interfaces (action-object paradigm), or will they first select an object and then choose the corresponding operation to be executed on the specified object (object-action paradigm)? While the first approach was often used in the past, today's applications tend to use the object-oriented user interface paradigm (OOUI), hence the latter approach. An advantage of the object-action approach is that users do not have to remember what action is valid for which object. The object has just to be selected and only those actions that can be performed on it will be available. This allows the user to explore the user interface in a simple manner by just selecting objects and look at the actions available for them. It further enables a user to work directly with the objects which reflects a user's way of doing work in the real world. Considering these aspects, we tried to strictly use the object-action paradigm designing the GOMES system.

The object-action paradigm can be further supported by the introduction of a drag and drop mechanism. This provides a convenient and intuitive way to perform many tasks using direct object manipulation, by first selecting an object (drag) and then choosing the corresponding action to be performed by releasing the object on a drop target. Users should always be informed when a drag and drop operation is currently in process, e.g. by modifying the shape of the mouse pointer when a drag is initiated. The change of the mouse pointer

when moving over a potential drop target gives the user visual feedback if the target object is accepting a drop and the operation can be completed.

At an early stage of graphical user interface design the developer should decide if the application is going to be used as a stand alone application showing only one main window on the screen at any time or if the application should cover the whole screen implementing its own desktop manager allowing the placement of overlapping windows on top of other windows like papers on a real world desk. The advantage of an application modeled as a desktop is the fact that it has got a three-dimensional look, resembling a desk being familiar to users. The increased control allows them to organize the whole screen to meet their own needs and there is no longer an ultimate need to close or delete unused windows since they just can be iconified. When using a desktop manager, it should be possible to place a symbolic reference (icon) to each object shown in a window onto the main desktop. This greatly improves working performance since regular user have the chance to place frequently used objects on the desktop, guaranteeing permanent access to them. The icons should clearly identify the objects or concepts they represent. When using different versions of the same icon (e.g. small and large versions), they should have similar shape, color and detail. Icons, or generally symbols, have been found to be recognized faster and more accurately than simple textual information. The graphical attributes of icons, such as shape and color, are therefore very useful for a quick classification of objects. An example of a good classification scheme that speeds up recognition are the icons indicating the kind of dialog box shown in Figure 3.1 and Figure 3.2, respectively.

Figure 3.1: Icons provided by the Java Metal Look and Feel for fast classification of message boxes

Figure 3.2: Informational dialog box using the default Java Metal Look and Feel classification icon

If these default icons are used in a consistent way in front of each message produced by an application, users will be able to faster classify the content of a message box.

Modal dialog boxes, i.e. dialog boxes blocking the whole system and restricting the order in which a certain task can be executed are only to be used when interaction with the application cannot process while the dialog box is displayed. Every time a new dialog box is opened, the initial keyboard focus should be set to the control users are most likely to choose first except the command might cause data loss. This initial focus is especially important for users navigating through applications using only the keyboard. The default command button is the button which is selected if a user presses the *Return* key. It represents the action that a user is most likely to perform and is identified with a heavier border than the other command buttons. A command that might cause data loss should never be made a default button, even if it is the action most frequently performed (see Figure 3.3).

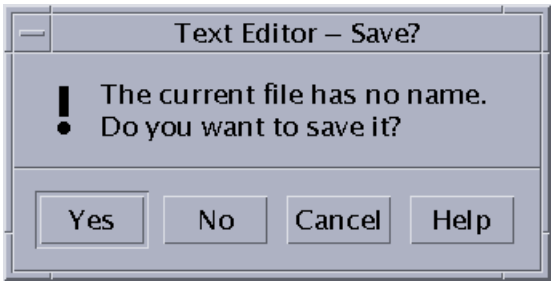

Figure 3.3: Default button *Yes* protecting the user from data loss

Consistency is a key aspect of useful graphical user interfaces. A major benefit of consistency is, that users can transfer their knowledge of parts of the application to new consistent parts they are learning. Since people like to explore and learn a new application by trial and error, the testing of new functionality should be encouraged by avoiding that users can easily lose data or even damage the whole application by just one simple operation. A system reacting oversensitive to erroneous input will discourage users from trying out new things. The exploring of new functions will be inhibited and users will work slowly and overcautious to avoid mistakes. Inconsistence in a graphical user interface is moreover very contra-productive, because it forces users to memorize all special cases and therefore unnecessarily increases the application's overall complexity. To achieve interface consistency, most operating system providers publish style guidelines for application developers. These guidelines specify the appearance and behavior of the user interface describing the windows, menus and various control mechanisms available and provide some guidance on when to use the different components. For examples of industry guidelines see [1, 12].

A goal of a good user interface is to hide the complexity of a sophisticated application (in our case the OMX-FS) by keeping the interface simple and straightforward while still providing the full power of the underlying system. Basic functions which are widely used should be immediately apparent, while advanced functions may be less obvious to new users. Often it is a good idea to introduce a user gradually to an application such that the full complexity of the system will not be visible at first. Nevertheless, the user should be able to control this process. As an example, for an experienced user it is very annoying and time consuming to always confirm a message box of the form *Do You really want to delete file XYZ?* if he wants to delete a file. While such a support is advisable for novices to protect them from an irretrievable data loss, a regular user should have the possibility to disable these mechanisms. An example of customizable support for experienced users is shown in Figure 3.4. It illustrates the possibility to disable tooltips in the GOMES system. While

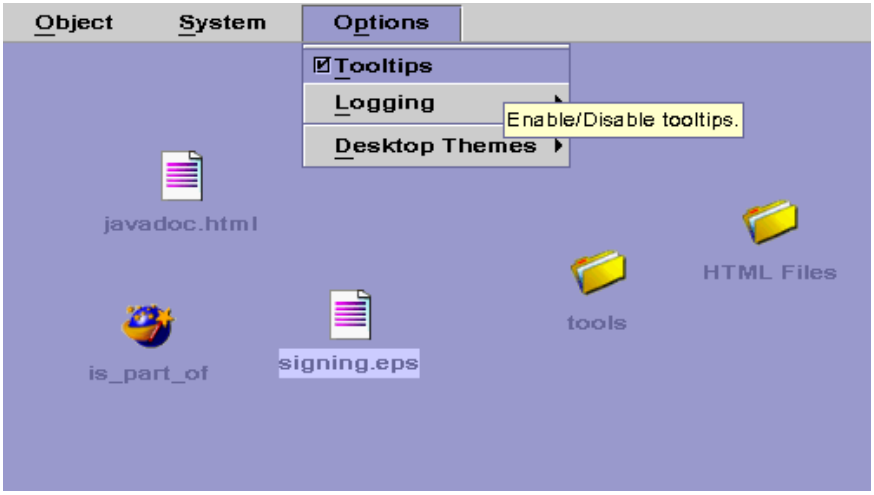

Figure 3.4: Control of tooltips in GOMES

tooltips are a good memory help for unexperienced users, it may be annoying for an expert user always being distracted by tooltips providing absolutely no new information. At any time the user should be in control of the application's graphical user interface and not being limited by the notion of correct actions to accomplish a certain task imposed by the developer designing the interface.

The user interface should immediately reflect the result of any operation by visual feedback, allowing the user to check whether the result is as expected or the action has to be undone. During modal operations which will take a longer time, the mouse pointer should be changed (e.g. to an hourglass). Additionally, some sort of progress bar may indicate the remaining time to complete the current task (Figure 3.5). Without any of this provisions, users tend to get impatient already after a short period without any visual feedback and will become frustrated, because they do not know if the system is still working correctly or if it is blocked by a faulty task.

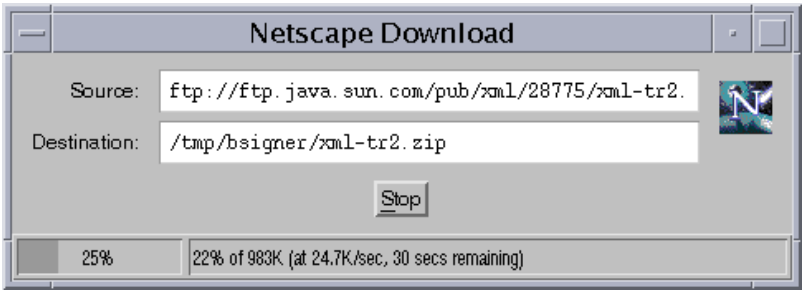

Figure 3.5: Estimated download time indicated by a progress bar

Another example of direct visual feedback supporting the user with additional context-sensitive information, is the concept of tooltips. Every time the mouse pointer is left over a graphic component for a certain time, a message box will appear, showing additional information about the corresponding component. A strict usage of tooltips greatly improves the process of learning to work with a new application.

Modern graphical user interfaces must support different interaction devices (e.g. mouse, keyboard, microphone etc.). At least they should support the use of either keyboard or mouse. Users should be allowed to switch between different input devices but it must also be possible to accomplish an entire task using the same input device. For example, it might be very tedious always having to use the mouse to scroll a window. If a user is currently editing a textual passage by keyboard, it should also be possible to scroll the window without having to change to the mouse input device, i.e. there should be a mechanism to scroll a window using the keyboard only. Furthermore providing different interaction mechanisms takes into account that users have different abilities and working environments.

Keyboard mnemonics are single underlined alphanumeric characters in a menu title, a menu item or other interface elements, that move the cursor to the corresponding choice and select it, if pressed in combination with the *Alt* key (see Figure 3.6). They support the idea of input device independency by en-

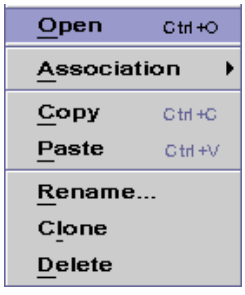

Figure 3.6: Pop-up menu using mnemonics and keyboard accelerators

suring that all application functionalities are also accessible from the keyboard, i.e. without using the mouse. Each component should have an associated mnemonic, except default and cancel buttons in dialog boxes. The *Return* key should be used for default buttons and the *Escape* key for cancel buttons instead. When a window is opened, the initial keyboard focus should be assigned to the most logical component (typically the component that most likely will be used first or otherwise the component in the upper left corner of the window). Keyboard Accelerators – single keys or combination of keys assigned to frequently performed actions – are still another keyboard alternative to the mouse. They are useful for operations frequently used by regular users, providing fast access to menu items without having to display the menu itself. Examples of keyboard accelerators are the copy and paste operators for clipboard control, which normally can be invoked by pressing *Ctrl+C* or *Ctrl+V*, respectively (as shown in Figure 3.6). The visual equivalent to keyboard accelerators is a tool bar. It contains several buttons providing the users direct access to the most frequently used commands.

Menus are very important user interface components. They enable users to choose an action which should be performed on the selected object from a list of potential operations. Like in the real world (e.g. on a restaurant's menu card) the items displayed on menus should be logically grouped by separators to help learning and to speed up the visual search process. Menus always have to be static, i.e. they should not be reordered based on the frequency they were chosen. Options found on more than one menu must be positioned consistently on all menus by placing them on the same relative position. The menu titles should be short and clearly designed (only single words) immediately orienting the user about the menu's content and its purpose. Menu entries have to provide an indication what is going to happen when the menu item is selected. A right-pointed arrow is used for multi-level menus, whereas ellipsis (the three dots: "...") after a menu item indicate that the command is not fully specified and the user will have to make additional selections in an appearing dialog box to accomplish the specification. If a menu item is not currently available, it should be disabled by dimming its text. If there is no possibility to make a menu item available it has to be omitted entirely rather than just disabling it, since a disabled menu item implies that the user has the possibility to make it available. If a menu item appears in several menus (for instance, if a copy command appears in a contextual menu as well as in a drop-down menu), the same shortcut should be usable.

A special form of menus are pop-up menus sometimes also called contextual menus as shown in Figure 3.7. They are displayed when a user presses the *right* mouse button while the mouse pointer is over an object or area associated with the corresponding menu, i.e. they show a menu in context of the selected object. A problem of contextual menus is, that they do not provide any visible indication of their existence and therefore users might not find them, especially if the application does not make extensive use of this kind of menu. For that reason any features presented in contextual menus should also be available in

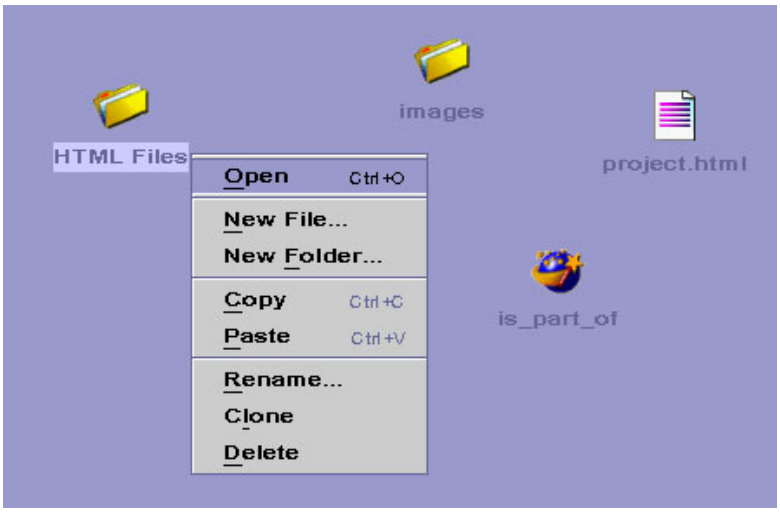

Figure 3.7: Pop-up menu invoked by pressing the *right* mouse button on the *HTML Files* folder

better visible and accessible components like drop-down menus. The keyboard accelerators and mnemonics shown in pop-up menus additionally have to be consistent with their use in corresponding drop-down menus.

Chapter 4

GUI Components of GOMES

The aim of GOMES was to find an adequate graphical representation of the rich functionality provided by the OMX-FS. Since the OMX-FS is an extension of existing file systems and therefore also includes their base functionality, we tried to visualize this common base functionality in a similar way to existing file systems, which has the great advantage that users already experienced with other file management systems will be able to use the base functionality of GOMES without much effort (an overview of the whole GOMES desktop is shown in Figure 4.7).

In the OMX-FS, files are not referenced by their pathnames and filenames like in other file systems. They are treated as objects and modeled totally independently of the underlying storage architecture. This implies that in the visualization of GOMES, a file does not have a visible unique identifier (it is even possible to have two different files with the same filename) and has always to be viewed in the current context. A great advantage of the OMX-FS is its concept of collections. They are similar to folders in other file systems and provide much the same base functionality. Whereas in most file systems a file can only be part of one folder to guarantee its identification (a file residing in several folders would not have the unique pathname necessary for identification) this restriction is no longer existent in the OMX-FS! You may now argue that the concept of *links* used by many other file systems gives you the same power as the OMX-FS' collection concept. The technique of introducing files acting as links does indeed allow to put a file virtually in several folders but at the same time evokes the severe problem of dangling links after removing the original file. Differently to other file systems, a collection can have several supercollections. This flexibility of collections is great to manage files, but does nevertheless introduce some problems when trying to visualize the collections, analyzed in the next section.

A first idea for the visualization of collections was to show them in a tree structure as used by many other file systems. This approach's problem is that, as

explained in the previous section, a collection may have several supercollections, i.e. the structure of the OMX-FS collections is not a common tree but a directed acyclic graph (DAG). A possible solution would have been to choose a more complex representation showing the whole collection DAG on the screen as shown in Figure 4.1. While this variant may be a suitable solution for a

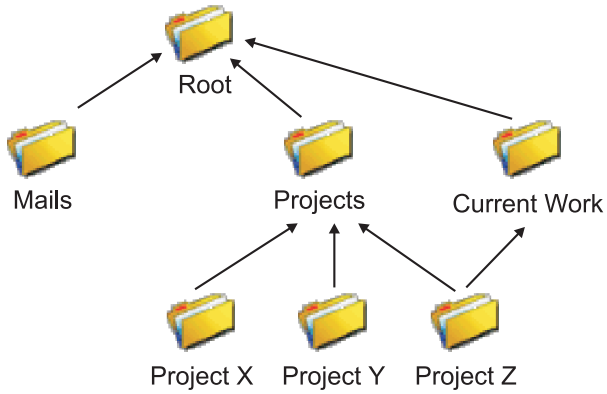

Figure 4.1: Representation of collections as a DAG

small amount of collections, it does not scale very well. How can a file system with thousands of collections be visualized without confusing the user and how can the collections be arranged?

Reminding that it should be possible to use the base functionality of GOMES similar to existing file systems, we decided to visualize the collection the same way folders are represented in many other file management systems. On the one hand the view of a collection should show the files it contains and at the same time it should also visualize its subcollections allowing a user to browse the file system. In the following we will speak of a *folder* if we mean a collection showing the files and subcollections it contains and consistently use the terms super- and subfolders for super- and subcollections.

The *GOMES File Manager*, the collection view implemented in GOMES, is shown in Figure 4.2. It contains two *explorers*, one on the left and one on the right hand side, separated by a button bar. Each explorer is a composition of three parts. The center part shows the current folder with all its files and subfolders. The user can choose three different view styles for this central folder view. He can either select a *detailed view* as shown on the left hand side providing additional information like the file size, creation time etc., a *large icon view* as shown on the right hand side showing only a large icon and the corresponding file name or the *small icon view* which is similar to the large icon view but uses smaller icons. The files and folders shown in the main folder view can either be separated (folders on top), as shown on the left hand side of Figure 4.2, or mixed as in the right main folder view. Furthermore the content of a folder view can be ordered by name, size, creation date or the date

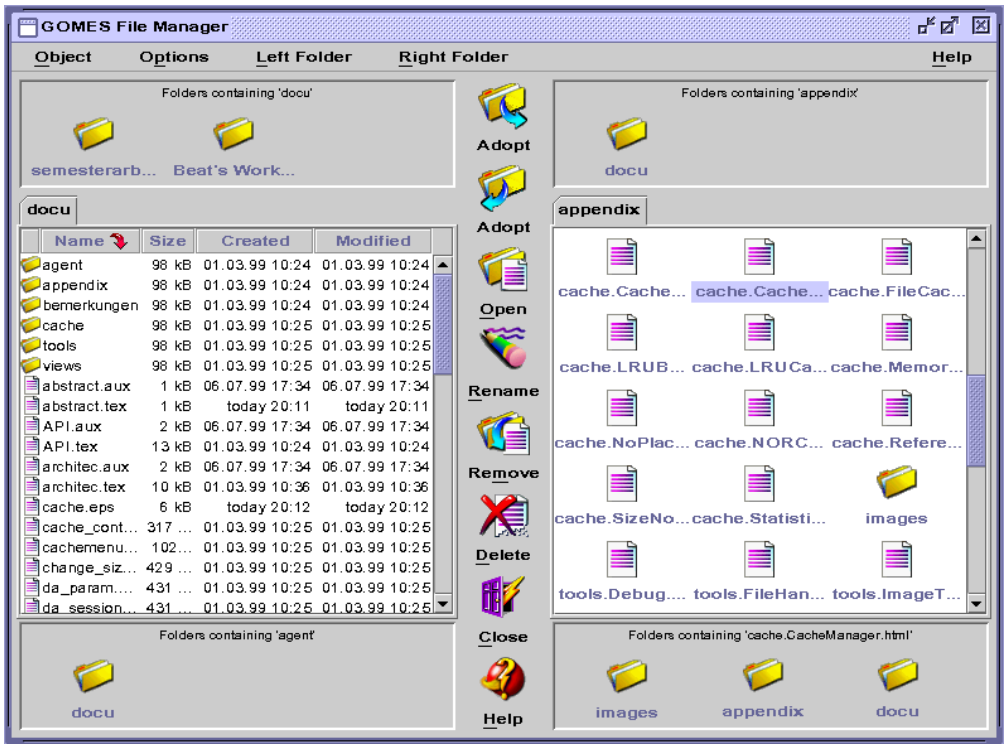

Figure 4.2: The *GOMES File Manager*

the object was last modified, either in an ascending or descending order. The sorting criteria can be specified by selecting the corresponding menu entry from the folder menu or just by clicking on the table header of the column containing the values to be sorted. A small red arrow in the header of the main folder view always provides visual feedback about the sorting criteria and sorting order (ascending or descending).

The top section of the explorer maintains all superfolders of the central main folder view. By double clicking on a folder in the top section, a user can make this folder the main folder, i.e. he can go one step upwards in the DAG structure. This mechanism is identical to the one used by many other file managers with the difference that in the GOMES system it is possible to choose from several superfolders.

As explained earlier, a file can be part of several folders but until now a user has no opportunity to easily change from one folder containing a file to another folder containing the same file. It should be possible to show all folders a specific file is part of, which is exactly the task of the bottom part of the explorer view. If the current selection in the main folder view is a file, the bottom view will show all folders the selected file is part of. Choosing a folder in the main folder results in presenting all of its superfolders in the bottom view. The explained three explorer components give users a great flexibility browsing their file system.

Although GOMES provides a drag and drop mechanism, it should additionally be possible to execute all operations without drag and drop, allowing the user to choose his favorite technique. To support binary operations, i.e. operations which need two objects as parameter, the *GOMES File Manager* contains two explorers. The selected object is always one parameter of such a binary operation while the main folder of the unselected explorer is the second parameter. The offering of two explorers allows one of them to be used as a short time memory, supporting a fast return to a certain folder. By choosing one of the adopt-buttons positioned at the top of the button bar, the views of the two main folders can be synchronized by setting the left main folder equal to the right main folder or the other way around, allowing a user to go back to the position stored in the neighbor folder. The use of one of the two folders as memory help may be sufficient for a short session but generally a user needs frequently several folders. By dragging a folder or file from the *GOMES File Manager* view and dropping it somewhere onto the main desktop, a user can gain easy access to these files and folders in the future (Figure 4.7 shows some folders and files on the main desktop).

The GOMES system totally supports five different mechanisms to choose an operation, whereas each operation can be executed at least by two of them. A first way to initiate a command is the drag and drop mechanism already discussed in the previous section. This is a very intuitive way to choose operations, especially in an object-oriented user interface as GOMES is. How do we put a document in a folder in the real world? We just pick up the document and put it into the folder. And this is exactly the same way as you can add a file or folder to a new folder in GOMES! Just drag the file and drop it on the desired folder. There are many other operations which can also be executed by drag and drop, e.g. an object can be deleted by dropping it on the trash.

A drawback of the drag and drop mechanism is that the user must always have access to the source and the target objects of the operation. A technique similar to the drag and drop mechanism but without the need of having access to the source and the target at the same time is GOMES' implementation of a clipboard. By choosing the *copy* command, a user can copy the selected object to the clipboard. This is equal to the initiation of a drag but without having to drop the object immediately. At any time the user has now access to the element in the clipboard shown in the lower right corner of the main desktop (see Figure 4.6). By selecting the *paste* command, he can initiate an binary operation with the currently selected object and the clipboard object as parameters (similar to the drop command explained earlier).

A problem of the described drag and drop but also the copy and paste mechanism is, that the user has to know which operation is implicitly bound to a drop or paste action within the current context. Users preferring to explicitly choose a command can also select the operation from the *menu bar* on the top of each window. The leftmost menu entry of the menu bar is always the *Object* menu showing all operations to be performed on the currently selected object.

Similar to the selection of a command in the menu bar is the use of pop-up menus. Every object has an associated menu which will pop up if a the user clicks on it with the right mouse button and will then show a context sensitive menu of all available operations. A common problem of pop-up menus is the user not knowing which objects are supporting a pop-up menu and which do not. To avoid this confusion, in GOMES *each* object has an associated pop-up menu.

Last but not least certain commands can also be chosen from the button bar of the explorer view mentioned earlier. The command chosen by clicking a button will always be applied to the current selection of the *GOMES File Manager* view the button bar resides in.

The second main component of the OMX-FS system to be visualized by GOMES is the *association* allowing to model dependencies between files. As an example, there could be some pictures in the OMX-FS all being part of a certain HTML page. A user will be able to model these dependencies by adding the pictures and the corresponding HTML page to a certain association, e.g. the *part_of* association. The concept of association gives a user additional power to logically order the files of his file system at a low level, enabling all applications to profit from the association mechanism. An association always posses a domain collection and a range collection restricting the possible files to be part of the corresponding association. If two files have to be connected by a certain association, one file has to be in the association's domain collection and the other in the associations range collection, i.e. there exist certain constraints.

In a first attempt we considered the integration of the visualization of associations in the existing explorer view. We immediately had to realize that collections and associations are two quite different concepts and that their representation in a single view would be very confusing. As we already stated at the beginning of this chapter, the use of the base functionality of the OMX-FS should be similar to existing file systems. While the *GOMES File Manager* gives a user access to this base functionality, the concept of associations is additional to most existing file systems. We therefore decided to design a separate view for the visualization of associations. This gives a novice-user the opportunity to use the *GOMES File Manager* for file management similar to existing file systems without being confused by parts of the association-view shown in the same window but never used, while an experienced user can make use of OMX-FS' full strength by also working with the separate association view.

Searching an adequate graphical representation for associations, we first had to answer the question how users of GOMES will make use of them. A first attempt was to allow a user to open a certain association, showing the association with all its associated files. In a second step the user would have to search the desired file he wishes the associations for either in the domain collection or in the range collection. The problem of this attempt was, that it is not very intuitive and does not rely on the object-action paradigm described in chapter 3.

Generally a user has a certain file selected and wants to get information about the associations of this specific file. In GOMES he can get access to this information by choosing the *Association* menu item either from the pop-up menu of the corresponding file (Figure 4.3) or from the *Object* menu in the title bar of the window containing the selected file. The association pop-up menu will

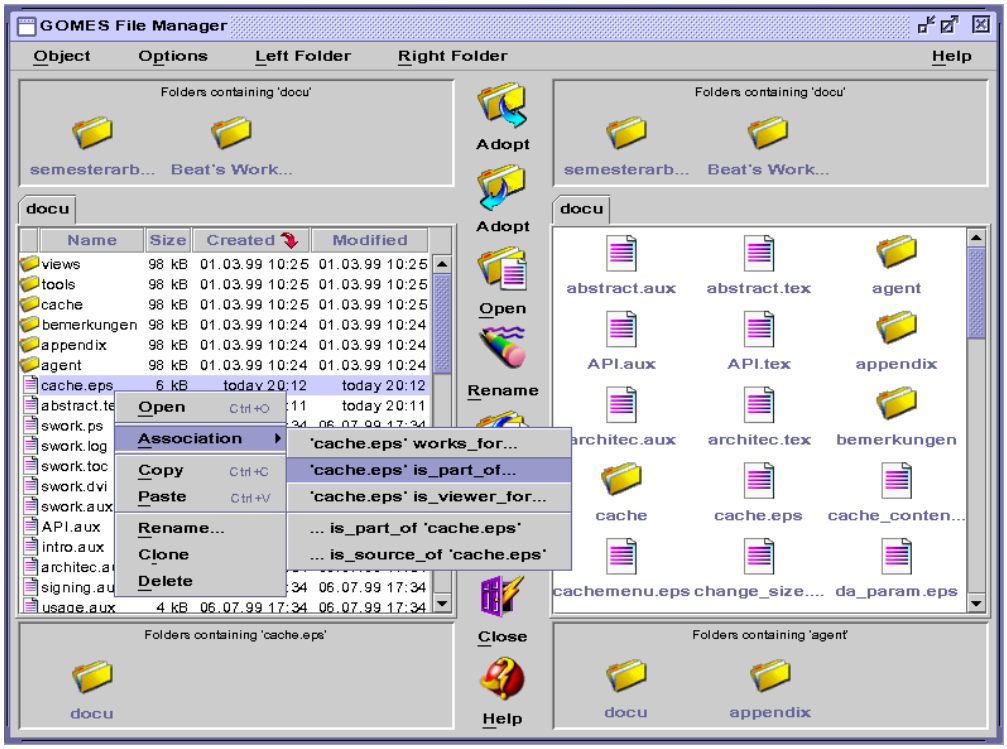

Figure 4.3: Pop-up menu showing associations of a file

show *all* potential associations the corresponding file may be part of, i.e. all associations containing one of the folders the selected file is part of either as domain or range folder. We have chosen this approach (presenting all potential associations the selected file may be part of) to enhance the procedure of generating a new association between two files as explained later.

The association pop-up menu is separated into two main sections. The upper section shows all associations the selected file may be domain file while the bottom part shows the associations it may be range file of. By selecting a particular association from the potential associations in the pop-up menu, a new *association view* as shown in Figure 4.4 will be opened.

The top section of the association view contains an iconified version of the association itself and the visualization of the domain and range folders (remember that folder is here an equivalent term for the OM model collections). The icon view of the association can be used like all other objects in the GOMES system, i.e. it is also possible to drag and drop on the association. As an example, the

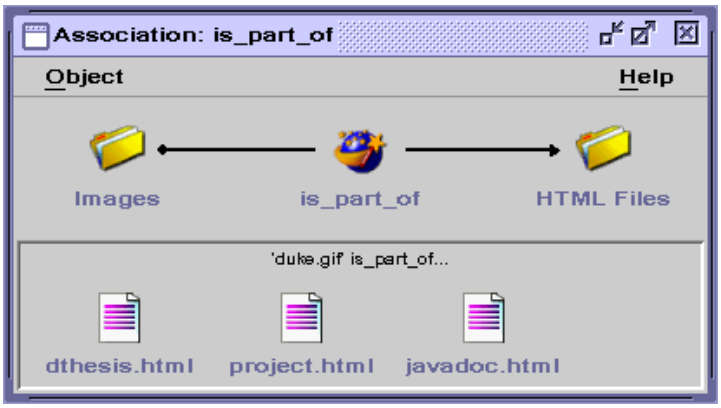

Figure 4.4: *Association view* showing a specific association

drop of an association on another association will make the drop target to the superassociation of the dropped association, consistently to the use of the drag and drop mechanism on folders. The lower part of the association view shows all files associated with the current one. Since either the domain or the range file is already defined by the file the association was opened on, to create a new association a user only has to choose the file defining the part of the association not yet specified. This method of constructing a new association evolved from the consistent use of the object-action paradigm. An alternative way to build up a new association would have been to first choose the type of the association to be built from a menu and then define the new association's domain and range file (action-object paradigm).

To concretely build a new association, the user has to drag and drop a file to the lower part of the association view. The dropped file together with the file the association view was opened on will build the domain and range file of the new association. If a user is not sure which files can be used for a certain association, a new *GOMES File Manager* view showing all potential members of the association can be opened by double clicking either the domain folder or the range folder in the upper part of the association view.

After having outlined the two main concepts to be visualized by GOMES (collection and association), in continuations the remaining parts building up the whole GOMES system are discussed.

The common operations available on all main objects of GOMES (files, folders and associations) can be invoked from the pop-up menu of the corresponding object, from the main menu or by just clicking on one of the buttons within the button bar. When renaming a file, we always have to remember that the filename is not a unique identifier anymore but rather a description of the file, i.e. it is possible to have two files with the same file name in one folder.

By *cloning* a GOMES object, a copy of the object will be generated and inserted into the OMX-FS. This process has not to be confounded with the *copying* of an object, which will copy the object into the clipboard but not generate a copy of the file within the OMX-FS system.

The two other commands not to be confounded are the *Remove* and the *Delete* of an object. The context sensitive *Remove* operation will not remove the object from the OMX-FS system at all, moreover it will remove the corresponding file from its context currently visualized. For example a remove operation applied to a file shown in a *folder view* will remove the file from the current folder whereas a remove of the same file shown in an *association view* will remove the corresponding association the file is part of. By deleting an object, the user definitively removes it from the underlying OMX-FS. Due to security aspects, the OMX-FS system will not physically delete the file but insert it in a special collection. A user has still access to the deleted files by the concept of this special *trash* collection shown in the lower left corner of the main desktop allowing him to recycle any deleted file.

Tooltips are used for different jobs within the GOMES system. On the one hand, they present additional information about objects, e.g. object size or creation date (Figure 4.5), on the other hand they are used to give a user some

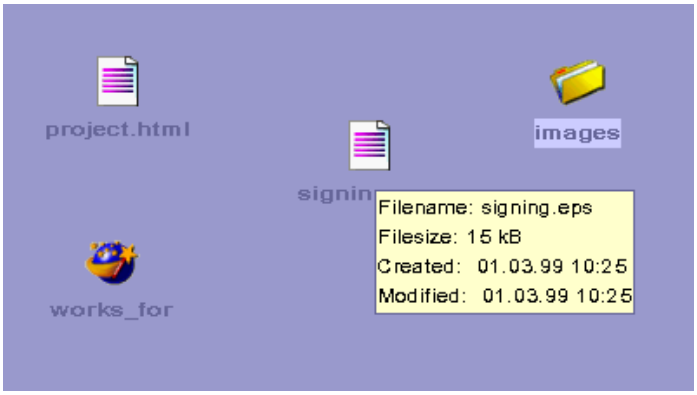

Figure 4.5: Tooltip showing additional information about a file.

kind of help about the different GOMES commands by always appearing if the mouse resides above a menu item. As shown earlier in Figure 3.4, it is possible to disable tooltips for the whole GOMES system giving expert users the opportunity not always being distracted by tooltips providing no new information. The GOMES desktop can be further customized by choosing a desktop theme (color schema) and arranging the objects on the desktop.

Finally the status bar positioned at the bottom of the desktop (see Figure 4.6) permanently informs the user about the current state of the GOMES system. It is partitioned into a message part and a clipboard section. The message part

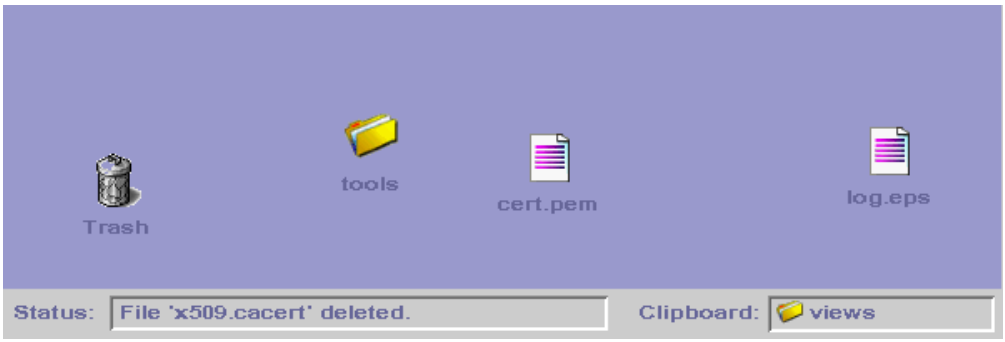

Figure 4.6: Status bar of the GOMES desktop

immediately reflects the result of any operation by showing textual feedback allowing the user to check whether the result is as expected. It additionally supports the user with information if a chosen operation is not possible in the current context. The second part of the status bar is the clipboard view which shows always the clipboard's content. The object shown in the clipboard view will be the source for the next paste action the user performs. Furthermore, it allows the user to check if a copy action was successful immediately updating the clipboard view with the new clipboard object.

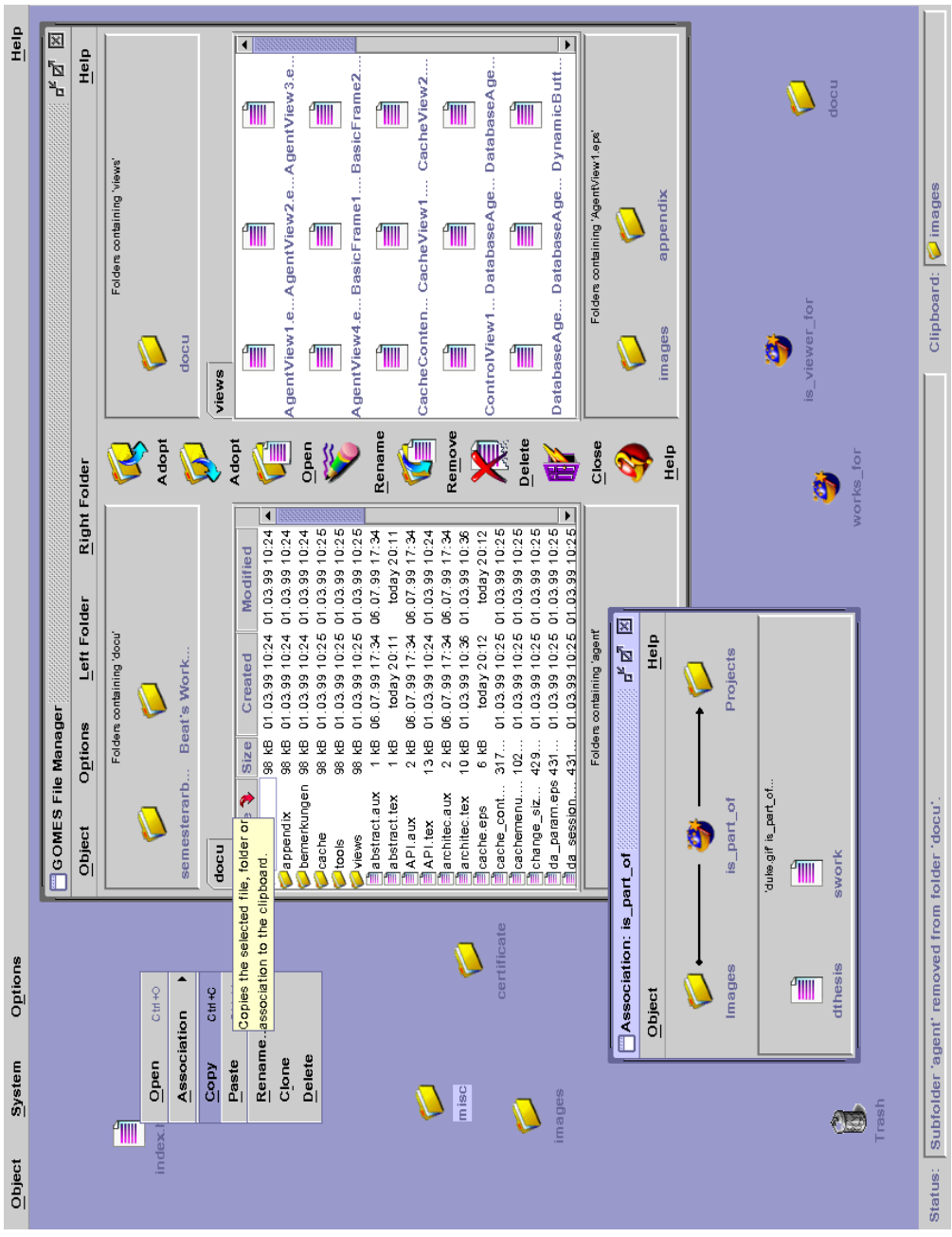

Figure 4.7: The whole GOMES desktop

Chapter 5

XML-RPC

As stated earlier, the OMX-FS is implemented in Oberon-2, an object-oriented version of the Oberon programming language. Generally, the object-oriented graphical user interface GOMES will not be executed on the same machine the OMX-FS server is running. Since GOMES nevertheless has to be able to make use of the whole file management functionality provided by the OMX-FS system, a mechanism had to be found that allows the Java Virtual Machine to communicate with the remote OMX-FS talking a different language and vice versa, i.e. some kind of Esperanto understandable by the two applications implemented in different programming languages and running on different platforms! On the one hand GOMES should be able to make remote procedure calls (RPC) and on the other hand the OMX-FS server must have the possibility to inform the graphical user interface about changes on its objects by invoking certain remote methods provided by GOMES guaranteeing consistency of remote objects.

We decided to choose the relatively new XML-RPC protocol[1], a standard protocol that uses the *Extensible Markup Language* (XML) to encode the remote procedure calls, for the whole inter-application communication between the Java virtual machine and the OMX-FS server, well knowing that we would never achieve the same performance as with an optimized proprietary protocol using stable sockets. An advantage of using the XML-RPC protocol is that the whole communication is based on the well established HTTP networking protocol. At the same time the use of the HTTP protocol may also a disadvantageous since each simple remote method call will have to build up a new HTTP connection (since HTTP/1.1 it is possible to solve this problem using persistent connections). This will become very time consuming if many remote procedure calls are required to perform a single GOMES task.

[1] http://Frontier.UserLand.Com/tree$2.8.2.1

By encoding the remote procedure calls and their results with the help of XML, the XML-RPC protocol furthermore allows us to view the content of the OMX-FS system using one of the latest network browser releases already able to show extensible markup language documents. Last but not least applications implementing the XML-RPC protocol will be able to make use of the OMX-FS independently of the language they are implemented in and the platform they are running on!

In this chapter we first will outline how the chosen XML-RPC protocol employs the extensible markup language to encode its data types. In a second part we describe how the protocol is used to model our own remote objects on the Java client side which can be used like normal Java objects but are in reality wrappers of the real Oberon objects residing on the machine running the OMX-FS.

Actually, the usage of the extensible markup language to encode remote procedure calls is only an implementation detail. Every other language allowing to define new meta information tags to represent the data types of the XML-RPC protocol could be used instead. Building on the extensible markup language has the advantage that there already exist quite a few robust XML parsers for the Java programming language allowing to parse extensible markup language documents with a limited effort. The *simple application programming interface for the extensible markup language* (SAX) is a small standard Java interface (for its specification see the Java interface *org.xml.sax.Parser*) for event-based XML parsing implemented by different available Java parsers. Each Java parser relying on the SAX interface traverses the tree of document nodes contained in an extensible markup language document and reports parsing events like the beginning of a new XML meta tag, input strings or the end of an XML tag to the application using a call back mechanism. An application interested in some of the parsing events will have to implement the *org.xml.sax.DocumentHandler* interface, i.e. it has to implement the corresponding methods which will be called by the XML-parser every time the corresponding meta tag is found.

The described method of using an existing extensible markup language parser implementing the SAX interface enables us to process extensible markup language documents and construct an internal representation of them, i.e. it allows GOMES to read the extensible markup language document resulting from a remote procedure call processed by the OMX-FS server. To encode the information necessary to invoke a remote method, we further have to be able to generate new extensible markup language documents. This is exactly the task of the `XMLWriter` (see appendix B.9) which can be used similar to the Java output stream classes and will encode all its input data into an XML-document.

Based on the explained facilities to read and generate new extensible markup language documents, in a next step the XML-RPC protocol specified by *Frontier.UserLand.Com* was implemented. An XML-RPC call is just a HTTP-POST request containing a remote procedure call encoded in XML. As an example of an XML-RPC call Figure 5.1 shows what the multiplication of two double val-

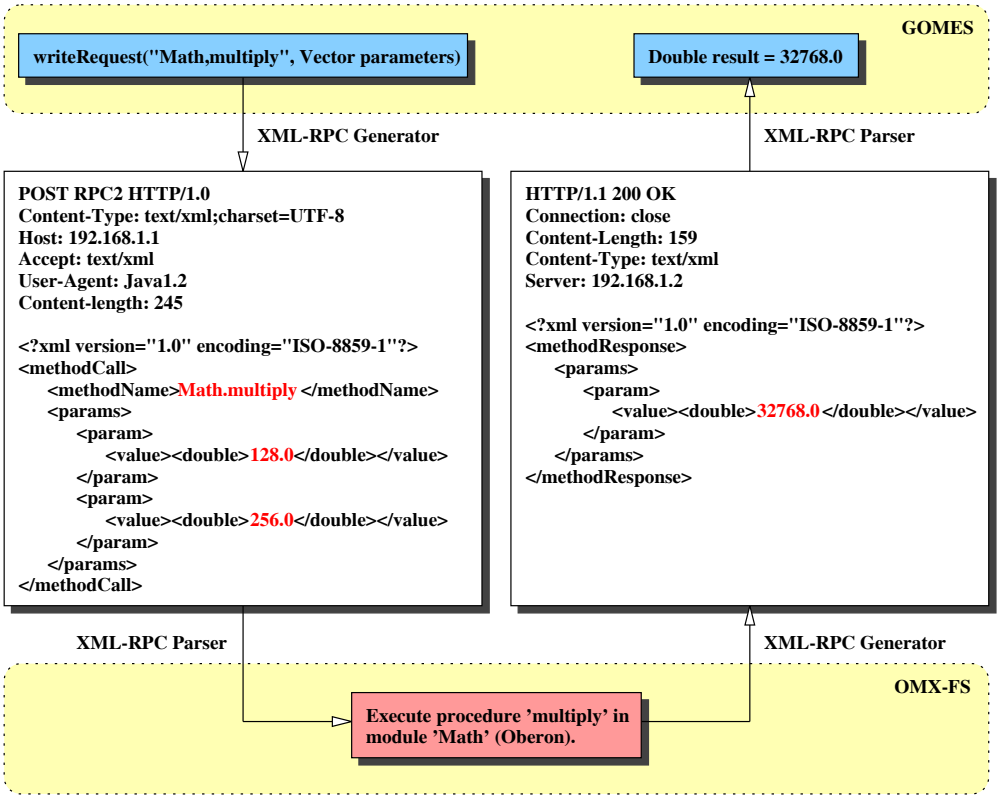

Figure 5.1: Example of an XML-RPC call multiplying two values

ues looks like. Each remote procedure call consists of two parameters: the name of the method to be invoked and a vector containing the parameters of the corresponding method. The parameters can either be scalars like integers, dates, etc. or composed values like structures and arrays (the scalar and composed values defined by the XML-RPC protocol are shown in table table 5.1 and table 5.2, respectively). In our example `Math.multiply` indicates that the procedure `multiply` in the module `Math` has to be executed. The parameter vector further contains two double values. The caller of the remote method invocation, in this case GOMES, will encode this information into an extensible markup language document using the XML-RPC generator. Here another advantage of the XML-RPC protocol becomes apparent, namely how easy XML documents are readable by humans. The generated document will be sent to the server using a simple HTTP-POST request. The OMX-FS server will parse the received document, execute the defined method and send the result as a new XML document to the caller. Finally GOMES will parse the XML document containing the result of the remote procedure call and return its value as result of the method invocation.

The implementation of the XML-RPC protocol allows for remote method invocations with scalar or composed return values. As shown in the previous example, we have to specify the method to be called by a module name and the

XML-Tag	Type	Java Type
<i4> or <int>	four-byte signed integer	java.lang.Integer
<boolean>	0 or 1	java.lang.Boolean
<string>	ASCII string	java.lang.String
<double>	double-precision signed floating point number	java.lang.Double
<dateTime.iso8601>	date/time	java.util.Date

Table 5.1: Scalar XML-RPC values

XML-Tag	Type	Java Type
<struct>	set of members each containing a name and a value	java.util.Hashtable
<array>	single <data> element which can contain any number of values	java.util.Vector

Table 5.2: Composed XML-RPC values

corresponding procedure name. Unfortunately, this only allows us to call static methods (procedures) but no methods bound to objects as needed. Furthermore it should also be possible that a remote procedure call returns a new object and not only a scalar value. Therefore, a new layer (Core Layer) which is responsible to construct and maintain remote objects had to be built on top of the existing XML-RPC layer. The Core Layer contains remote wrapper objects for all the corresponding main objects of the OMX-FS. Since each object has a unique object ID, we use the object ID instead of the module name if not a procedure but a method on a specific object has to be invoked. The response format for scalar and composed values stays the same as explained earlier. Additionally to this scalar format a new *object format* is introduced, which will always be used when an object has to be returned (see figure 5.2). An object return value

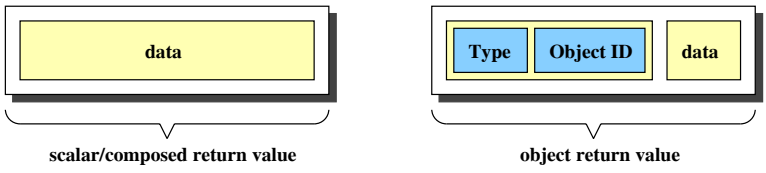

Figure 5.2: The two types of return values

is a vector containing two main parts: an *object descriptor* and a *data* part. The *object descriptor* contains a *type descriptor* string followed by the unique *object ID*. The *data* part optionally contains some values to initialize the object. Every time an XML-RPC caller receives the result of a remote procedure call, he first has to check if the result is a scalar value or a remote object (vector containing one element or vector containing two elements, respectively). In the

first case where a scalar value is returned, nothing special has to be done. If a remote object is returned, a special object loader will be invoked. This object loader maintains a mapping table of the OMX-FS types and their corresponding wrapping objects in the Core Layer of GOMES. The object loader will be invoked with the type descriptor string of the remote procedure call result as argument and will return a new Java wrapping object. As an invariant, each remote object always has a unique object ID and therefore the object ID of the result value will immediately be assigned to the new remote object. In a next step, the initialization method of the new remote object will be invoked with the data part resulting from the remote procedure call as argument. Since each remote object exactly knows the format of the data it has to receive, it will parse the data field and initialize the corresponding variables. The object load mechanism allows us, to dynamically extend the system by adding new classes to the mapping table. It is further possible to cascade the data parts of an object, e.g. if the returned object is an extension of another class type. In such a case, a supertype will first call its subtypes initialization method and in a second step parse its own additional data.

A last problem we had to solve is to guarantee consistency of the whole GOMES system. Since the OMX-FS is a multi-user system but our remote objects are some type of *local copies*, a mechanism had to be introduced allowing the OMX-FS to notify GOMES about object changes. Therefore GOMES will register all its remote objects by object ID in a central table. Every time an object in the OMX-FS changes, OMX-FS will send a message containing the object ID of the corresponding object to GOMES which will invalidate the corresponding remote object and reload it from the OMX-FS system.

Since XML-RPC uses the HTTP protocol and is therefore not very fast, some kind of caching had to be introduced in order to reduce the number of remote procedure calls. Every time an object is returned, not only the object itself but also the result of its most frequently used methods will be returned as part of the data field. This additional information is stored in the remote object and every time one of these cached methods will be called on the remote object, there will be no remote procedure call, since the data is already cached.

To illustrate the remote object loading and method caching mechanism, Figure 5.3 shows the loading of an `OMCore.Object` returned by the OMX-FS as the result of a remote procedure call. In a first step (1) the `CoreObjectLoader` extracts the *object descriptor* part from the remote procedure call result. It searches the remote object class for the Oberon type `OMCore.Object` in its class mapping table and dynamically loads the corresponding `CoreObject`. After assigning the unique *Object ID* to the new `CoreObject` the object loader invokes the object's initialization method with the data part of the remote procedure call result as parameter (2). Since the `CoreObject` has a superclass (`CoreOMObject`) we first have to initialize this superclass. The data part of the remote procedure call result contains two elements. The first vector entry is the initialization data for the superclass of `CoreObject` while the second part

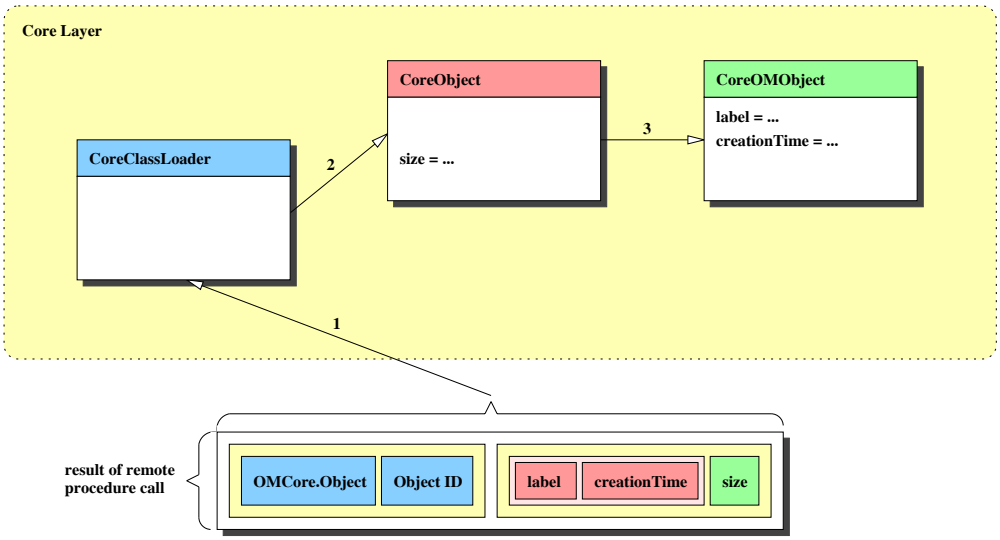

Figure 5.3: Processing of an object return value

contains its own data. The `CoreObject` invokes the initialization method of its superclass `CoreOMObject` (3) with the first data vector entry as parameter, resulting in the initialization of the *label* and *creationTime* variables. In a final step the initialization method of the `CoreObject` processes the second entry of the result data vector and assigns its value to the *size* variable

The *label*, *creationTime* and *size* values normally have to be fetched by a remote method invocation every time they are needed. Since they are frequently used by GOMES, they are already transfered at the construction time of a new object. This allows the wrapping remote object to cache these values and reduce the number of necessary remote procedure calls. As an example, if the `getLabel()` method is invoked on a remote `CoreObject` the return value *label* will be fetched from the cache and no remote procedure call has to be made.

Finally, it has to be mentioned that the XML-RPC protocol is a small and quite efficient protocol. It is much simpler than other standards like CORBA [16] and DCOM and still has the power to become an important remote procedure call protocol in the future. The magic of the protocol is that the whole object serialization mechanism can be expressed by only two simple tags, the <struct> and the <array> tag.

Chapter 6

Future Work

Due to the modular architecture of the GOMES system, it is possible to easily replace a layer of the system without affecting the others. To improve the performance when running GOMES and the OMX-FS system on the same machine, the implementation of an alternative proprietary remote procedure protocol for the communication between GOMES and the OMX-FS, replacing the XML-RPC layer, could be considered.

Chapter 7

Conclusion

The goal of this diploma thesis was the design and implementation of GOMES, an object-oriented graphical user interface for the object model muti-user extended file system (OMX-FS) offering easy access to the full strength of the OMX-FS file system.

The Java application GOMES was realized based on a case study about how computer users generally work with graphical user-interfaces, especially with user-interfaces of file management systems.

The main design principles of GOMES are its strict use of the object-action paradigm supported by a drag and drop mechanism. For the base functionality of the OMX-FS we tried to use a visualization similar to existing file managers, allowing novice users to profit quickly of GOMES. Additional components, such as the association view, enable the expert to make use of the full strength of the underlying OMX-FS file system.

The XML-RPC protocol has been proved to support the communication between GOMES and the OMX-FS quite well. Modeling its own remote object mechanism allows GOMES to get access to the full functionality of the OMX-FS system it is based on. GOMES' method caching technique makes it possible to keep the performance penalty of the HTTP based XML-RPC protocol within acceptable boundaries.

Acknowledgments
I am very grateful to my supervising assistant Gabrio Rivera for always having the time to answer my questions and beeing flexible in adding additional functionality to the OMX-FS file system interface. Thanks also to the other members of the GlobIS group, the members of the OMS-Lab and especially to Prof. Moira C. Norrie, for the opportunity of my diploma thesis.

Appendix A

Glossary

clipboard	A storage place for a single object which can be inserted using the *copy* command, while a *paste* allows to insert the clipboard's content into other components.
drag	Moving the mouse while holding down a mouse button.
drag and drop	To drag a component in order to apply an operation with the drop target.
drop	Releasing of the mouse button after a drag was initiated.
drop target	The component over which a drop occurs.
drop-down menu	A menu appearing when a user selects a menu title in the menu bar.
icon	A symbol graphically representing an object or a concept.
keyboard accelerator	A combination of keys that activates a menu item even if the corresponding menu is not currently displayed.
menu	A list of menu items logically grouped.
menu bar	The horizontal bar at the top of a window containing the titles of the drop-down menus.
menu item	A single choice in a menu. Generally menu items are commands that a user can select.
mnemonic	An underlined alphanumeric character, typically in a menu title, a menu item or the text of a component. A mnemonic allows the user to activate the corresponding command by pressing the *Alt* key and the underlined letter.

Oberon-2	Object-oriented version of the programming language Oberon, a successor of Pascal and Modula-2.
pop-up menu	A contextual menu appearing when a user presses the right mouse button while the mouse pointer is over an object associated with that menu. It offers only menu items applicable to the selected object.
separator	A graphical line used to logically group menu items and other components.
tooltip	A short message shown when a user moves the mouse pointer over a component associated with the corresponding tooltip.
XML	Extensible markup language allowing to define new markup tags or even new markup languages.
XML-RPC	Protocol developed by *Frontier.UserLand.Com* allowing to invoke remote methods by using XML.

Appendix B

API Reference

The whole GOMES system consists of nine packages which will be described in the following. The `gomes` package contains the main classes to start the GOMES client application. The maintenance of all remote objects is done by the `gomes.core` package. Package `gomes.event` provides some specific events and event listeners, respectively. The classes of the `gomes.model` package build the model for the whole visualization provided by the `gomes.view` package. A stub server supporting the system with data based on the local file system is implemented in the `gomes.server` package. The `gomes.util` package provides general utility classes while the `gomes.view.util` package contains view specific utility classes. Last but not least the `xmlrpc` package implements the XML-RPC protocol specified by *Frontier.UserLand.Com* allowing the Java virtual machine to communicate with the Oberon OMX-FS.

B.1 The *gomes* Package

The `gomes` package contains the startup classes of the GOMES system. `Desktop` is the main class of the whole system which will start the graphical user interface. `OMXFS` builds the entry point to the OMX-FS whereas `OMXFileSystem` provides the base functionality of the OMX-FS file system.

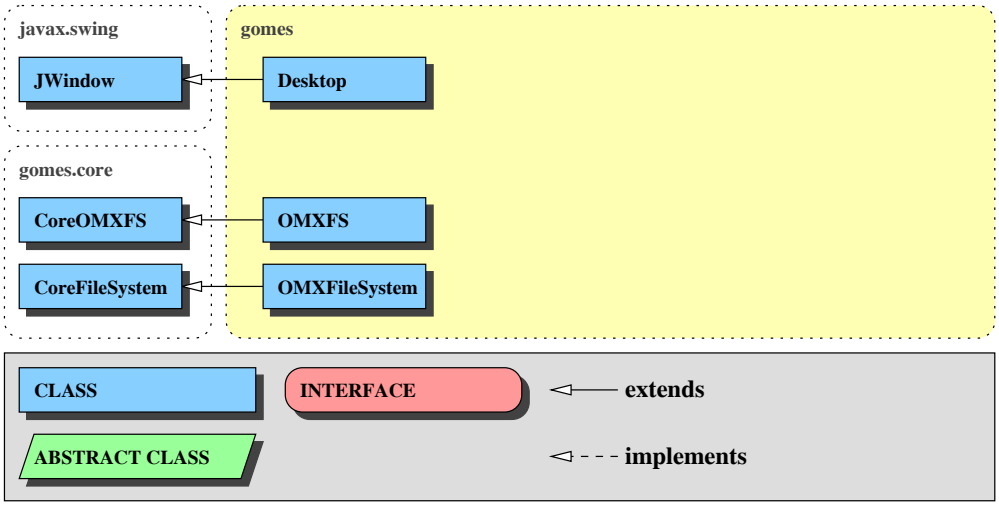

Figure B.1: The *gomes* package

gomes.Desktop

java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        javax.swing.JWindow

---

public *Desktop*
extends JWindow
implements GOMESObjectSelectionListener

Desktop and main application of the OMX file system.

Fields

Type	Description
public static final Integer	**ICON_LAYER**

Constructors

Description
Desktop() Constructs a new GOMES main desktop.

Methods

Returns	Description
public static GOMESObject	**getClipboard**() Returns the content of the clipboard.
public static OMXFileSystem	**getFileSystem**() Returns the currently opened file system.
public static JDesktopPane	**getMainDesktop**() Returns the main desktop.
public void	**initClipboard**() Initializes the clipboard.
public static void	**main**(String[] args) The GOMES main application.
public void	**objectSelected**(GOMESObjectSelectionEvent event) Up-called whenever a GOMESObject was selected.
protected void	**readPreferences**() Reads the preferences from a file.
public static void	**setClipboard**(GOMESObject object) Adds an object to the clipboard.
public static void	**showStatus**(String text) Adds a message to the status bar.
public static void	**showWarning**(String text) Adds a warning message to the status bar.
public void	**updatePreferences**() Updates the preferences.
public void	**writePreferences**() Writes the preferences to a file.

gomes.OMXFileSystem

java.lang.Object
  gomes.core.RPCObject
    gomes.core.CoreFileSystem

---

public *OMXFileSystem*
extends CoreFileSystem

Remote object of the 'OMXFS.Filesystem' type specified by the OMX-FS.

Methods

Returns	Description
public CoreCollection	**getFilesCollection**() Returns the root collection of the file system containing all the objects.
public CoreCollection	**getTrashCollection**() Returns a collection containing all the objects of the trash.
public CoreUser	**getUser**() Returns the user currently using the file system.
public CoreAssociation	**newAssociation**(String label, CoreCollection domainCollection, CoreCollection rangeCollection) Adds a new association to the file system.
public CoreCollection	**newCollection**(String label, CoreCollection supercollection) Adds a new collection to the file system.
public CoreObject	**newFile**(String filename) Adds a new file to the file system.
public CoreObject	**openFile**(CoreIdentifier identifier) Opens a file. A write lock will be installed.
public CoreSet	**openFiles**(String filename) Opens all the files with the specified filename.
public CoreObject	**openFirstFile**(String filename) Opens the first file with the specified filename.
public void	**setUser**(CoreUser currentUser) Sets the current user of the file system.

gomes.OMXFS

java.lang.Object
  gomes.core.RPCObject
    gomes.core.CoreOMXFS

---

public *OMXFS*
extends CoreOMXFS

Remote object of the 'OMXFS' static methods specified by the OMX-FS.

Methods

Returns	Description
public static void	**deleteFileSystem**(CoreUser user, CoreFileSystem fileSystem) Deletes the whole file system.
public static CoreUser	**getUser**(String username, String password) Returns a CoreUser for a specified user name and a corresponding password.
public static CoreFileSystem	**login**(CoreUser user) Login for a specified file system
public static CoreFileSystem	**login**(CoreUser user, int fileSystemNo) Login for a specified file system
public static void	**logout**(CoreUser user) Logout for a specified file system.
public static void	**logout**(CoreUser user, int fileSystemNo) Logout for a specified file system.
public static CoreFileSystem	**newFileSystem**(CoreUser creator) Creates a new file system.
protected static void	**readPreferences**() Reads the preferences from a file.
public static void	**updatePreferences**() Updates the preferences.
public static void	**writePreferences**() Writes the preferences to a file.

B.2 The *gomes.core* Package

The `gomes.core` package is responsible for the mapping of the Oberon OMX-FS objects to the Java GOMES system by maintaining the whole remote-object framework. `RPCObject` is the base class of all remote objects. The `RPC-`

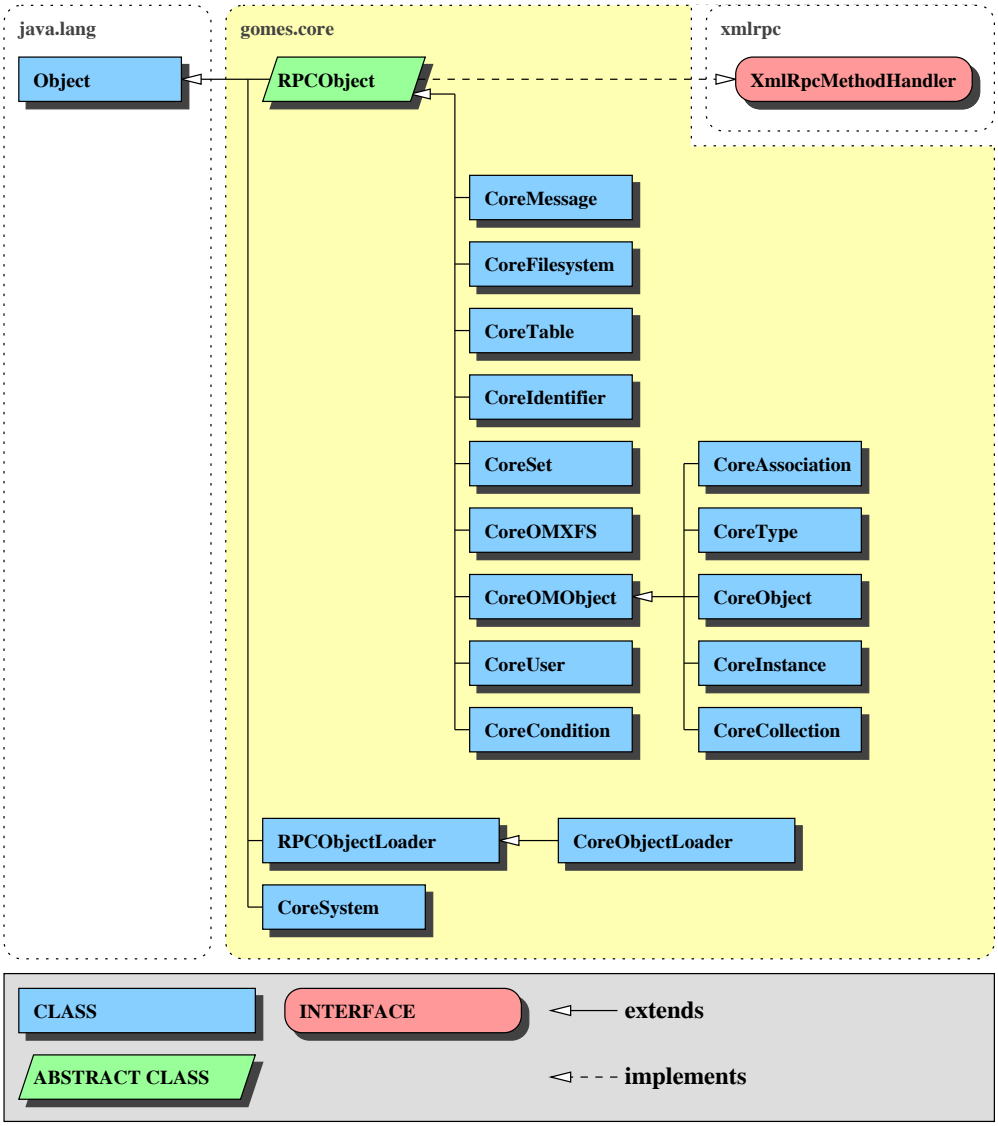

Figure B.2: The *gomes.core* package

`ObjectLoader` is a dynamic object loader invoked every time a new remote object has to be generated. The user can dynamically add new classes to a mapping table which will be used loading remote objects. The `CoreObject-Loader` is the specific `RPCObjectLoader` extension for the OMX-FS, providing the corresponding mapping of Oberon types to Java classes (e.g. the OMX-FS

type `OMCore.Association` is mapped to the Java class `gomes.core.CoreAssociation`). The `CoreSystem` maintains the `CoreObjectLoader` for the current session. The remaining extensions of the `RPCObject` class are the corresponding remote objects for the OMX-FS types.

gomes.core.CoreAssociation

java.lang.Object
  gomes.core.RPCObject
    gomes.core.CoreOMObject

public *CoreAssociation*
extends CoreOMObject

Remote object of the 'OMCore.Association' type specified by the OMX-FS.

Constructors

Description
CoreAssociation(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreAssociation.
CoreAssociation(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, Date creationDate, Date modificationDate) Constructs a new CoreAssociation.

Methods

Returns	Description
public void	**addSuperassociation**(CoreAssociation superassociation) Adds a superassociation to the association.
public boolean	**containPair**(CoreObject domainObject, CoreObject rangeObject) Returns true if the association contains the two associated objects domainObject and rangeObject, false otherwise.
public CoreAssociation	**copy**() Returns a copy of the association.
public CoreSet	**domainRestriction**(CoreObject rangeObject) Returns a set of all the pairs (domainObject, rangeObject) the association contains, whereas the range object has to be equivalent to the specified object.
public void	**enumerate**(CoreMessage message) Sends a message to all the objects the associations contains.
public CoreObject	**findFirst**(CoreCondition condition) Returns the first object fulfilling the specified condition.
public CoreSet	**getAllSubassociations**() Returns all subassociations of the association.
public CoreSet	**getAllSuperassociations**() Returns all superassociations of the association.
public CoreSet	**getDomain**(CoreObject rangeObject) Returns a set containing all the objects of the association's domain.
public CoreCollection	**getDomainCollection**() Returns the domain collection of the association.
public CoreTable	**getObjects**() Returns a table (2 columns) containing the objects of the association. Each row contains a pair of two associated objects.
public CoreSet	**getRange**(CoreObject domainObject) Returns a set containing all the objects of the association's range.
public CoreCollection	**getRangeCollection**() Returns the range collection of the association.
public CoreSet	**getSubassociations**() Returns a set containing the subassociations of the association (only one level in the hierarchy of subassociations).
public CoreSet	**getSuperassociations**() Returns a set containing the superassociations of the association (only one level in the hierarchy of superassociations).
public void	**insertPair**(CoreObject domainObject, CoreObject rangeObject) Associates two objects.
public CoreSet	**rangeRestriction**(CoreObject domainObject) Returns a set of all the pairs (domainObject, rangeObject) the association contains, whereas the domain object has to be equivalent to the specified object.
public void	**removePair**(CoreObject domainObject, CoreObject rangeObject) Removes the specified pair of objects from the association.
public void	**removeSuperassociation**(CoreAssociation superAssociation) Removes the specified association from the set of superassociations.

gomes.core.CoreObjectLoader

java.lang.Object
gomes.core.RPCObjectLoader

public *CoreObjectLoader*
extends RPCObjectLoader
implements XmlRpcMethodHandler

Object loader for dynamic loading of remote objects. The Oberon types are mapped to the corresponding Java classes to be used loading a new object.

Constructors

Description
CoreObjectLoader() Constructs a new CoreObjectLoader.

Methods

Returns	Description
public Object	**invokeMethod**(String method, Vector params) Invokes a method on the CoreObjectLoader to notify about an object modification.
public Object	**loadObject**(Vector result, XmlRpcMethodHandler methodHandler) Loads an 'RPCObject' object from the data encoded in the result vector. The result may either contain one or two elements. If it contains only one element the result is one of the defined XML-RPC standard types (not really a remote object, because no methods can be invoked on the result). If the result vector contains two elements, the first element contains the unique object ID and the corresponding class name of the object, whereas the second element contains object specific data.

gomes.core.CoreCollection

java.lang.Object
gomes.core.RPCObject
gomes.core.CoreOMObject

public *CoreCollection*
extends CoreOMObject

Remote object of the 'OMCore.Collection' type specified by the OMX-FS.

Constructors

Description
CoreCollection(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreCollection.
CoreCollection(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, int size, Date creationDate, Date modificationDate) Constructs a new CoreCollection.

Methods

Returns	Description
public void	**addSupercollection**(CoreCollection supercollection) Adds a supercollection to the collection.

Returns	Description
public boolean	**contain**(CoreObject object) Returns true if the collection contains the specified object, false otherwise.
public CoreCollection	**copy**() Returns a copy of the collection.
public Enumeration	**elements**() Returns all the objects of the collection.
public void	**enumerate**(CoreMessage message) Sends a message to all the objects the collection contains.
public CoreObject	**findFirst**(CoreCondition condition) Returns the first object fulfilling the specified condition.
public void	**fireRPCObjectChangePerformed**(RPCObjectChangeEvent event) Notifies all listeners that have registered interest for notification on this event type.
public CoreSet	**getAllSubcollections**() Returns all subcollections of the collection.
public CoreSet	**getAllSupercollections**() Returns all supercollections of the collection.
public CoreSet	**getAssocsByDomain**() Returns a set of all the associations having this collection as domain collection.
public CoreSet	**getAssocsByRange**() Returns a set of all the associations having this collection as range collection.
public CoreType	**getMembertype**() Returns the membertype of the collection.
public CoreSet	**getObjects**() Returns a set containing all the objects of the collection.
public int	**getSize**() Returns the size of the collection.
public CoreSet	**getSubcollections**() Returns the subcollections of the collection (only one level in the hierarchy of subcollections).
public CoreSet	**getSupercollections**() Returns the supercollections of the collection (only one level in the hierarchy of supercollections).
public void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part contains the elements of the table.
public void	**insert**(CoreObject object) Adds an object to the collection.
public void	**remove**(CoreObject object) Removes an object from the collection.
public void	**removeSupercollection**(CoreCollection supercollection) Removes a supercollection from the collection.
public void	**setMembertype**(CoreType membertype) Sets the membertype of the collection.
public void	**setSize**(int size) Sets the size of the collection.

gomes.core.CoreCondition

java.lang.Object
  gomes.core.RPCObject

---

public *CoreCondition*
extends RPCObject

Remote object of the 'OMCore.Condition' type specified by the OMX-FS.

Constructors

Description
CoreCondition(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreCondition.

Methods

Returns	Description
public void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part may be used to initialize the object.

gomes.core.CoreFileSystem

java.lang.Object
  gomes.core.RPCObject

public *CoreFileSystem*
extends RPCObject

Remote object of the 'OMXFS.Filesystem' type specified by the OMX-FS.

Constructors

Description
CoreFileSystem(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreFileSystem.

Methods

Returns	Description
public CoreObject	**associateFiles**(CoreAssociation association, CoreObject domainObject, CoreObject rangeObject) Adds a new associated pair of objects to specified association.
public void	**closeFile**(CoreObject file) Closes a file. The write lock on the file will be released.
public void	**deleteAssociation**(CoreAssociation association) Deletes an association from the file system.
public void	**deleteCollection**(CoreCollection collection) Deletes a collection from the file system.
public void	**deleteFile**(CoreObject file) Deletes a file from the file system.
public void	**emptyTrash**() Deletes all objects the trash contains.
public CoreCollection	**getFilesCollection**() Returns the root collection of the file system containing all the objects.
public int	**getFileSystemNo**() Returns the number of the file system.
public CoreCollection	**getTrashCollection**() Returns a collection containing all the objects of the trash.
public void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part may be used to initialize the object.
public CoreAssociation	**newAssociation**(String label, CoreCollection domainCollection, CoreCollection rangeCollection) Adds a new association to the file system.
public CoreCollection	**newCollection**(String label, CoreCollection supercollection) Adds a new collection to the file system.
public CoreObject	**newFile**(String filename) Adds a new file to the file system.
public CoreObject	**openFile**(CoreIdentifier identifier) Opens a file. A write lock will be installed.
public CoreSet	**openFiles**(String filename) Opens all the files with the specified filename.
public CoreObject	**openFirstFile**(String filename) Opens the first file with the specified filename.
public void	**saveFileSystem**() Stores the file system.

gomes.core.CoreIdentifier

java.lang.Object
  gomes.core.RPCObject

---

public *CoreIdentifier*
extends RPCObject

Remote object of the 'OMCore.Identifier' type specified by the OMX-FS.

Constructors

Description
CoreIdentifier(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreIdentifier.

Methods

Returns	Description
public void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part may be used to initialize the object.

gomes.core.CoreInstance

java.lang.Object
  gomes.core.RPCObject
    gomes.core.CoreOMObject

---

public *CoreInstance*
extends CoreOMObject

Remote object of the 'OMCore.Instance' type specified by the OMX-FS.

Constructors

Description
CoreInstance(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreInstance.
CoreInstance(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, Date creationDate, Date modificationDate) Constructs a new CoreInstance.

Methods

Returns	Description
public CoreObject	**getObject**() Returns the corresponding object.
public CoreType	**getType**() Returns the type of the instance.

gomes.core.CoreMessage

java.lang.Object
gomes.core.RPCObject

public *CoreMessage*
extends RPCObject

Remote object of the 'OMCore.Message' type specified by the OMX-FS.

Constructors

Description
CoreMessage(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreMessage.

Methods

Returns	Description
public void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part may be used to initialize the object.

gomes.core.CoreObject

java.lang.Object
gomes.core.RPCObject
gomes.core.CoreOMObject

public *CoreObject*
extends CoreOMObject

Remote object of the 'OMCore.Object' type specified by the OMX-FS.

Constructors

Description
CoreObject(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreObject.
CoreObject(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, int size, Date creationDate, Date modificationDate) Constructs a new CoreObject.

Methods

Returns	Description
public CoreObject	**copy**() Returns a copy of the object.
public CoreInstance	**findFirstInstance**(CoreCondition condition) Returns the first object fulfilling the specified condition.
public void	**fireRPCObjectChangePerformed**(RPCObjectChangeEvent event) Notifies all listeners that have registered interest for notification on this event type.
public CoreSet	**getAllAssocsByDomain**() Returns all possible associations the object can be domain part of.
public CoreSet	**getAllAssocsByRange**() Returns all possible associations the object can be range part of.
public CoreSet	**getAssocsByDomain**() Returns a set of all the associations containing a pair with this object as domain.

Returns	Description
public CoreSet	**getAssocsByRange**() Returns a set of all the associations containing a pair with this object as range.
public CoreSet	**getCollections**() Returns the collections the object is part of.
public CoreInstance	**getInstance**(CoreType context) Returns an instance for the corresponding context.
public CoreSet	**getInstances**() Returns the object's instances.
public int	**getSize**() Returns the size of an object.
public CoreSet	**getTypes**() Returns the object's types.
public void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part contains the elements of the object.
public void	**notifyInstances**(CoreMessage message) Sends a message to all instances.
public void	**setSize**(int size) Sets the size of an object.

gomes.core.CoreOMObject

java.lang.Object
  gomes.core.RPCObject

public *CoreOMObject*
extends RPCObject

Remote object of the 'OMCore.OMObject' type specified by the OMX-FS. Each CoreOMObject has a label, an access mode, a creation date and a modification date.

Fields

Type	Description
public static final int	**NO_ACCESS**
public static final int	**READ_ONLY_ACCESS**
public static final int	**READ_WRITE_ACCESS**

Constructors

Description
CoreOMObject(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreOMObject.
CoreOMObject(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, Date creationDate, Date modificationDate) Constructs a new CoreOMObject.

Methods

Returns	Description
public void	**fireRPCObjectChangePerformed**(RPCObjectChangeEvent event) Notifies all listeners that have registered interest for notification on this event type.
public int	**getAccessMode**(CoreUser user) Returns the access mode of the object.
public Date	**getCreationDate**() Returns the date the object was created.
public String	**getLabel**() Returns the label (description) of the object.
public Date	**getModificationDate**() Returns the date the object was last modified.
public void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part contains some attributes of the object (label, access mode, creation date and modification date).
public void	**setLabel**(String label) Sets the label (description) of the object.
public void	**setModificationDate**() Sets the date the object was last modified to the current date.

Returns	Description
public String	**toString**() Returns a string representation of the CoreOMObject's content.

gomes.core.CoreOMXFS

java.lang.Object
gomes.core.RPCObject

public *CoreOMXFS*
extends RPCObject

Remote object of the 'OMXFS' static methods specified by the OMX-FS.

Constructors

Description
CoreOMXFS() Constructs a new CoreOMXFS.

Methods

Returns	Description
public static void	**deleteFileSystem**(CoreUser user, CoreFileSystem filesystem, XmlRpcMethodHandler methodHandler) Deletes the whole file system.
public static CoreUser	**getUser**(String userName, String password, XmlRpcMethodHandler methodHandler) Returns a CoreUser for a specified user name and a corresponding password.
public void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part may be used to initialize the object.
public static CoreFileSystem	**login**(CoreUser user, int filesystemNo, XmlRpcMethodHandler methodHandler) Login for a specified file system.
public static void	**logout**(CoreUser user, int fileSystemNo, XmlRpcMethodHandler methodHandler) Logout for a specified file system.
public static CoreFileSystem	**newFileSystem**(CoreUser creator, XmlRpcMethodHandler methodHandler) Creates a new file system.

gomes.core.CoreSet

java.lang.Object
gomes.core.RPCObject

public *CoreSet*
extends RPCObject

Remote object of the 'ADTSets.Set' type specified by the OMX-FS.

Constructors

Description
CoreSet(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreSet.

Methods

Returns	Description
public void	**add**(Object object) Adds an object to the set.
public Enumeration	**elements**() Returns the objects of the set.
public Object	**get**(int index) Returns the object at the specified position.
public void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part contains the elements of the set.
public int	**size**() Returns the number of objects the set contains.

gomes.core.CoreSystem

java.lang.Object

---

public *CoreSystem*
extends Object

Maintains the CoreObjectLoader which will be used to load the remote objects.

Constructors

Description
CoreSystem()

Methods

Returns	Description
public static CoreObjectLoader	**getObjectLoader**() Returns the object loader to load the remote objects.

gomes.core.CoreTable

java.lang.Object
gomes.core.RPCObject

---

public *CoreTable*
extends RPCObject

Remote object of the 'ADTTables.Table' type specified by the OMX-FS.

Constructors

Description
CoreTable(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreTable.
CoreTable(String objectID, XmlRpcMethodHandler methodHandler, Object data) Constructs a new CoreTable.

Methods

Returns	Description
public int	**getColumnCount**() Returns the number of columns the table contains.
public int	**getRowCount**() Returns the number of rows the table contains.
public Object	**getValueAt**(int rowIndex, int columnIndex) Returns the value at position (rowIndex, columnIndex) in the table.
public void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part contains the elements of the table.

gomes.core.CoreType

java.lang.Object
  gomes.core.RPCObject
    gomes.core.CoreOMObject

---

public *CoreType*
extends CoreOMObject

Remote object of the 'OMCore.Type' type specified by the OMX-FS.

Constructors

Description
CoreType(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreType.
CoreType(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, Date creationDate, Date modificationDate) Constructs a new CoreType.

Methods

Returns	Description
public void	**addSupertype**(CoreType supertype) Adds a supertype to the type.
public CoreType	**copy**() Returns a copy of the type.
public CoreSet	**getCollections**() Returns a set containing the type's collections.
public CoreSet	**getInstances**() Returns a set containing the types's instances.
public CoreSet	**getObjects**() Returns a set containing the type's objects.
public CoreSet	**getSubtypes**() Returns a set containing the type's subtypes.
public CoreSet	**getSupertypes**() Returns a set containing the type's supertypes.
public void	**removeSupertype**(CoreType supertype) Removes a supertype from the type.

gomes.core.CoreUser

java.lang.Object
  gomes.core.RPCObject

---

public *CoreUser*
extends RPCObject

Remote object of the 'OMUtilities.User' type specified by the OMX-FS.

Constructors

Description
CoreUser(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new CoreUser.

Methods

Returns	Description
public void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part may be used to initialize the object.

gomes.core.RPCObjectLoader

java.lang.Object

---

public *RPCObjectLoader*
extends Object

Object loader for dynamic loading of remote objects. The user can add new classes to a mapping table which will be used loading remote objects.

Constructors

Description
RPCObjectLoader()

Methods

Returns	Description
public void	**addClass**(String originalClass, String mappedClass) Adds a new class to the mapping table of the object loader.
public Class	**getClass**(String originalClass) Returns the corresponding mapping class.
public RPCObject	**getInstance**(String className, String objectID, XmlRpcMethodHandler methodHandler) Constructs a new instance of an RPCObject.

gomes.core.RPCObject

java.lang.Object

---

public abstract *RPCObject*
extends Object
implements XmlRpcMethodHandler

Base class of all remote objects. Each remote object has a unique object identifier which will be used to identify operations on a specific object.

Fields

Type	Description
protected EventListenerList	**listenerList**
protected static final Vector	**NO_PARAMS**

Constructors

Description
RPCObject(String objectID, XmlRpcMethodHandler methodHandler) Constructs a new RPCObject.

Methods

Returns	Description
public void	**addRPCObjectChangeListener**(RPCObjectChangeListener listener) Adds an RPCObjectChangeListener to the RPCObject.
public void	**fireRPCObjectChangePerformed**(RPCObjectChangeEvent event) Notifies all listeners that have registered interest for notification on this event type.
public XmlRpcMethodHandler	**getMethodHandler**() Returns the method handler responsible to process operations on the RPCObject.
public String	**getObjectID**() Returns the unique object ID of the RPCObject.
public abstract void	**init**(Object data) Invoked by the RPCObjectLoader after generating a new object. The data part may be used to initialize the object.
public Object	**invokeMethod**(String methodName, Vector parameters) Invokes a method on the RPCObject.
public static Object	**invokeStaticMethod**(String moduleName, String methodName, Vector parameters, XmlRpcMethodHandler methodHandler) Invokes a static method.
public void	**removeRPCObjectChangeListener**(RPCObjectChangeListener listener) Removes an RPCObjectChangeListener from the RPCObject.
public void	**setMethodHandler**(XmlRpcMethodHandler methodHandler) Sets the method handler which will be responsible to process operations on the RPCObject.
public void	**setObjectID**(String objectID) Sets the unique object ID of the RPCObject.
public String	**toString**() Returns a string representation of the OMObject's content.

B.3 The *gomes.event* Package

The `gomes.event` package contains classes responsible for event handling. `Look-`

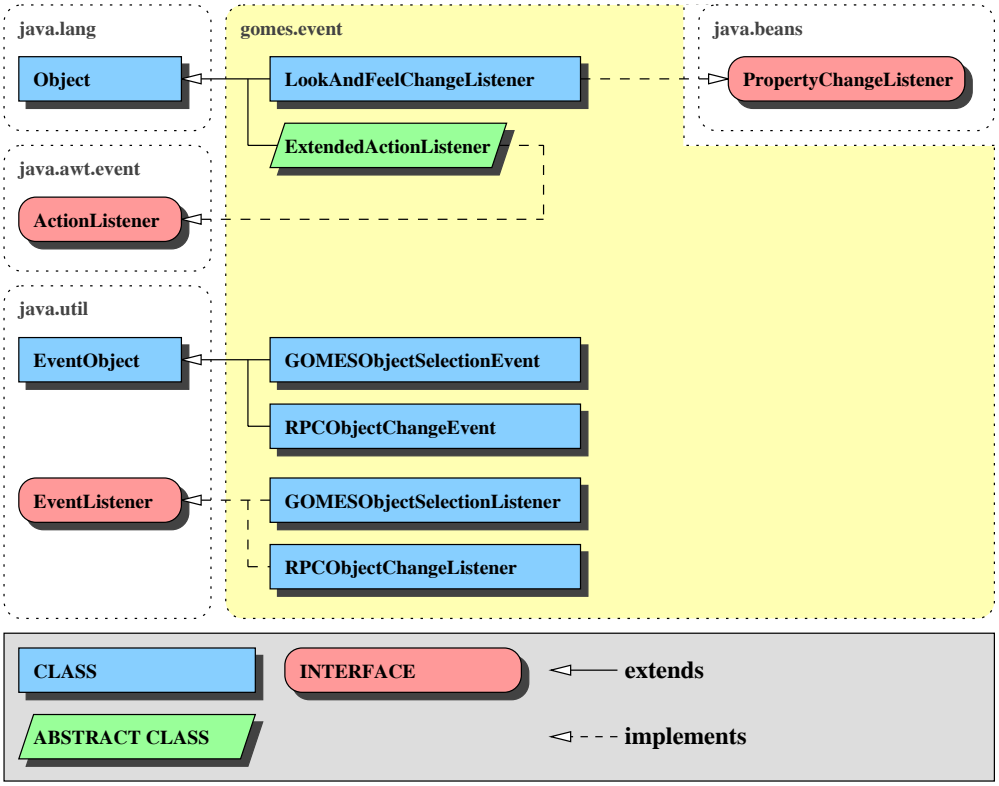

Figure B.3: The *gomes.event* package

`AndFeelChangeListener` is listening for changes to the look and feel of the desktop. `ExtendendActionListener` allows to bind an object to a specific action. A `GOMESObjectSelectionEvent` is fired each time an object is selected, while `GOMESObjectSelectionListener` is listening for these selection events. `RPCObjectChangeListener` and `RPCObjectChangeEvent` are used to guarantee GOMES' consistency to the OMX-FS.

gomes.event.ExtendedActionListener

java.lang.Object

public abstract *ExtendedActionListener*
extends Object
implements ActionListener

An action listener allowing to bind an object to an action.

Constructors

Description
ExtendedActionListener(Object producer) Constructs a new ExtendedActionListener.

Methods

Returns	Description
public Object	**getProducer**() Returns the object bound to the action.

gomes.event.GOMESObjectSelectionEvent

java.lang.Object
java.util.EventObject

public *GOMESObjectSelectionEvent*
extends EventObject

Event indicating that an GOMESObject was selected.

Fields

Type	Description
public static final int	**LEFT_BUTTON**
public static final int	**MIDDLE_BUTTON**
public static final int	**RIGHT_BUTTON**

Constructors

Description
GOMESObjectSelectionEvent(GOMESObject source, int clickCount, int mouseButton) Constructs a new GOMESObjectSelectionEvent.
GOMESObjectSelectionEvent(GOMESObject source, MouseEvent e) Constructs a new GOMES-ObjectSelectionEvent.

Methods

Returns	Description
public int	**getClickCount**() Returns the number of mouse clicks on the selected GOMESObject.
public GOMESObject	**getGOMESObject**() Returns the GOMESObject which produced the selection event.

Returns	Description
public boolean	**isLeftButton**() Returns true if the the GOMESObject was selected with the left mouse button, false otherwise.
public boolean	**isMiddleButton**() Returns true if the GOMESObject was selected with the middle mouse button, false otherwise.
public boolean	**isRightButton**() Returns true if the GOMESObject was selected with the right mouse button, false otherwise.
public String	**toString**() Returns a string representation of the GOMESObjectSelectionEvent.

gomes.event.GOMESObjectSelectionListener

public abstract interface *GOMESObjectSelectionListener*

implements EventListener

Interface that a GOMESObjectSelection listener has to implement to receive GOMESObjectSelection events.

Methods

Returns	Description
public void	**objectSelected**(GOMESObjectSelectionEvent event) Up-called whenever a GOMESObject was selected.

gomes.event.LookAndFeelChangeListener

java.lang.Object

public *LookAndFeelChangeListener*
extends Object
implements PropertyChangeListener

A look and feel change listener used to handle changes to the look and fell of the application.

Constructors

Description
LookAndFeelChangeListener(JComponent component) Constructs a new LookAndFeelChangeListener.

Methods

Returns	Description
public void	**propertyChange**(PropertyChangeEvent event) Up-called whenever a property changed.

gomes.event.RPCObjectChangeEvent

java.lang.Object
  java.util.EventObject

public *RPCObjectChangeEvent*
extends EventObject

Event indicating that an RPCObject has changed.

Fields

Type	Description
public static final int	**DELETED**
public static final int	**MODIFIED**

Constructors

Description
RPCObjectChangeEvent(RPCObject source, int changeType) Constructs a new RPCObject-ChangeEvent.

Methods

Returns	Description
public RPCObject	**getObject**() Returns the RPCObject which produced the change event.
public boolean	**isDeleted**() Returns true if the RPCObject was deleted.
public boolean	**isModified**() Returns true if the RPCObject was modified but not deleted, false otherwise.
public String	**toString**() Returns a string representation of the RPCObject-ChangeEvent.

gomes.event.RPCObjectChangeListener

public abstract interface *RPCObjectChangeListener*

implements EventListener

Interface that an RPCObjectChange listener has to implement to receive RPCObjectChange events.

Methods

Returns	Description
public void	**objectChanged**(RPCObjectChangeEvent event) Up-called whenever an RPCObject has changed.

B.4 The *gomes.model* Package

The package `gomes.model` contains the models used by the different views of GOMES. The `GOMESObject` interface ensures the base functionality of the OMX-FS `CoreOMObject`. `GOMESCollection` is an interface to be implemented by objects representing a collection of `GOMESObjects`. `GOMESTableModel` is an extension of the TableModel supporting easy access to the underlying objects. The remaining classes are the main models of GOMES based on the corresponding remote core classes.

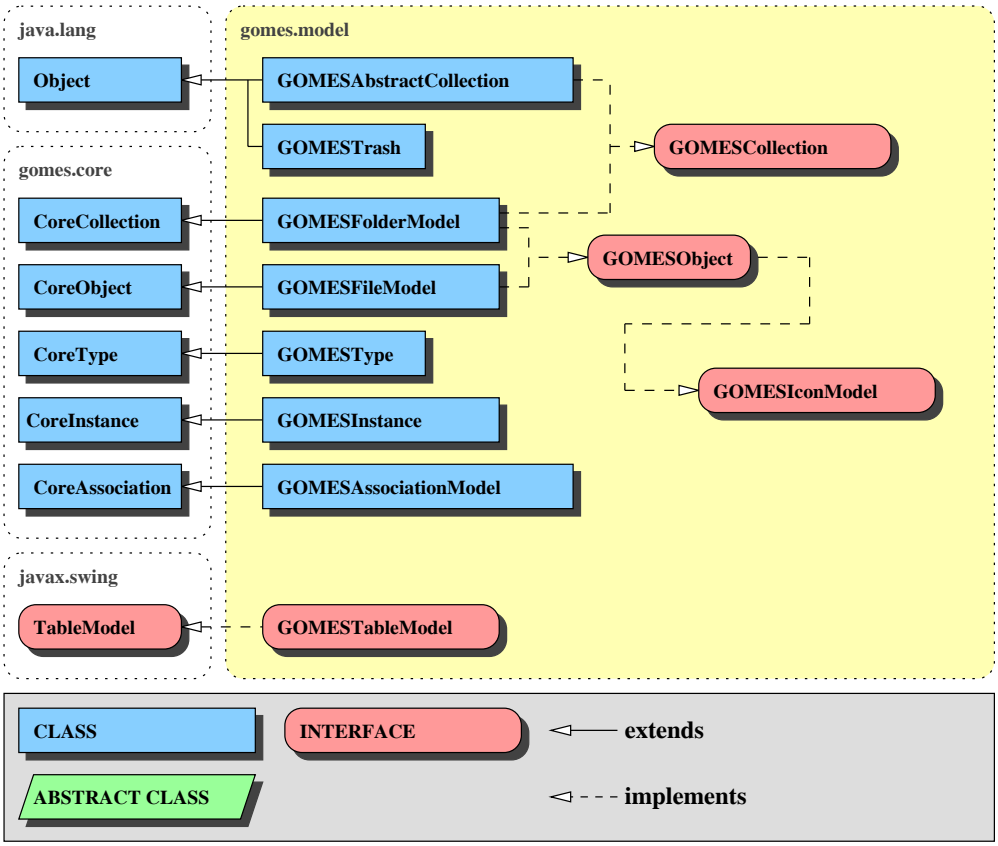

Figure B.4: The *gomes.model* package

gomes.model.GOMESAbstractCollection

java.lang.Object

---

public *GOMESAbstractCollection*
extends Object
implements GOMESCollection

Standard implementation of the GOMESCollection interface.

Constructors

Description
GOMESAbstractCollection()

Methods

Returns	Description
public void	**addElement**(GOMESObject object) Adds an object to the collection.
public Enumeration	**elements**() Returns all objects the collection contains.
public int	**getSize**() Returns the number of objects the collection contains.

gomes.model.GOMESAssociationModel

java.lang.Object
gomes.core.RPCObject
gomes.core.CoreOMObject
gomes.core.CoreAssociation

---

public *GOMESAssociationModel*
extends CoreAssociation
implements GOMESObject

Model of association views.

Constructors

Description
GOMESAssociationModel(CoreAssociation association) Constructs a new GOMESAssociationModel.
GOMESAssociationModel(CoreAssociation association, ImageIcon smallIcon, ImageIcon largeIcon) Constructs a new GOMESAssociationModel.
GOMESAssociationModel(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, Date creationDate, Date modificationDate) Constructs a new GOMESAssociationModel.
GOMESAssociationModel(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, Date creationDate, Date modificationDate, ImageIcon smallIcon, ImageIcon largeIcon) Constructs a new GOMESAssociationModel.

Methods

Returns	Description
public void	**addRPCObjectChangeListener**(RPCObjectChangeListener listener, int objectType, GOMESFileModel currentFile) Adds an RPCObject-ChangeListener to the GOMESFolderModel. The listener will also be informed about changes on objects the folder contains.
public CoreAssociation	**copy**() Returns a copy of the GOMESAssociationModel.

Returns	Description
public CoreObject	**findFirst**(CoreCondition condition) Returns the first GOMESFileModel fulfilling the specified condition.
public CoreSet	**getAllSubassociations**() Returns all subassociations of the association.
public CoreSet	**getAllSuperassociations**() Returns all superassociations of the association.
public String	**getCapitalizedTypeDescription**() Returns a description of the object's type (first letter is capitalized).
public CoreSet	**getDomain**(CoreObject rangeObject) Returns a set containing all the GOMESFileModels of the associations domain.
public CoreCollection	**getDomainCollection**() Returns the domain GOMESFolderModel of the association.
public ImageIcon	**getLargeIcon**() Returns the large icon to be used showing the GOMESAssociationModel.
public CoreTable	**getObjects**() Returns a table (2 columns) containing the GOMESFileModels of the association. Each row contains a pair of two associated GOMESFileModels.
public Enumeration	**getParents**() Returns the parents (superassociations) of the OMXAssociationsModel.
public CoreSet	**getRange**(CoreObject domainObject) Returns a set containing all the OMXfileModels of the associations range.
public CoreCollection	**getRangeCollection**() Returns the range GOMESFolderModel of the association.
public int	**getSize**() Returns the size of the GOMESAssociationModel.
public ImageIcon	**getSmallIcon**() Returns the small icon to be used showing the GOMESAssociationModel.
public CoreSet	**getSubassociations**() Returns a set containing the subassociations of the association (only one level in the hierarchy of subassociations).
public CoreSet	**getSuperassociations**() Returns a set containing the superassociations of the association (only one level in the hierarchy of superassociations).
public String	**getToolTipText**() Returns the text to be shown in the tooltip for the association object.
public String	**getTypeDescription**() Returns a description of the object's type.
public void	**removeRPCObjectChangeListener**(RPCObjectChangeListener listener, int objectType, GOMESFileModel currentFile) Removes an RPCObjectChangeListener from the GOMESAssociationModel and all the objects it contains.
public void	**setLargeIcon**(ImageIcon largeIcon) Sets the large icon to be used showing the GOMESAssociationModel.
public void	**setSmallIcon**(ImageIcon smallIcon) Sets the small icon to be used showing the GOMESAssociationModel.
public String	**toString**() Returns a string representation of the GOMESAssociationModel content.

gomes.model.GOMESCollection

public abstract interface *GOMESCollection*

Interface to be implemented by objects representing a collection of GOMESObjects.

Methods

Returns	Description
public Enumeration	**elements**() Returns all objects the collection contains.
public int	**getSize**() Returns the number of objects the collection contains.

gomes.model.GOMESFileModel

java.lang.Object
gomes.core.RPCObject
gomes.core.CoreOMObject
gomes.core.CoreObject

---

public *GOMESFileModel*
extends CoreObject
implements Comparable, GOMESObject

Model of file views.

Constructors

Description
GOMESFileModel(CoreObject coreObject) Constructs a new GOMESFileModel.
GOMESFileModel(CoreObject coreObject, ImageIcon smallIcon, ImageIcon largeIcon) Constructs a new GOMESFileModel.
GOMESFileModel(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, int size, Date creationDate, Date modificationDate) Constructs a new GOMESFileModel.
GOMESFileModel(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, int size, Date creationDate, Date modificationDate, ImageIcon smallIcon, ImageIcon largeIcon) Constructs a new GOMESFileModel.

Methods

Returns	Description
public int	**compareTo**(Object object) Compares the GOMESFileModel to another GOMESFileModel. Returns -1 if this FileModel is smaller than the FileModel it has to be compared to, 1 if it is larger and 0 if the two FileModels are equal.
public CoreObject	**copy**() Returns a copy of the GOMESFileModel.
public CoreInstance	**findFirstInstance**(CoreCondition condition) Returns the first GOMESInstance fulfilling the specified condition.
public CoreSet	**getAllAssocsByDomain**() Returns all possible GOMESAssociationModels the object can be domain part of.
public CoreSet	**getAllAssocsByRange**() Returns all possible GOMESAssociationModels the object can be range part of.
public CoreSet	**getAssocsByDomain**() Returns a set of all the GOMESAssociationModels containing a pair with this object as domain.
public CoreSet	**getAssocsByRange**() Returns a set of all the GOMESAssociationModels containing a pair with this object as range.
public String	**getCapitalizedTypeDescription**() Returns a description of the object's type (first letter is capitalized).
public CoreSet	**getCollections**() Returns the GOMESFolderModels the object is part of.
public CoreInstance	**getInstance**(CoreType context) Returns an GOMESInstance for the corresponding context.
public CoreSet	**getInstances**() Returns the objects GOMESInstances.
public ImageIcon	**getLargeIcon**() Returns the large icon to be used showing the GOMESFileModel.
public Enumeration	**getParents**() Returns the parents (folders) the GOMESFileModel is part of.
public ImageIcon	**getSmallIcon**() Returns the small icon to be used showing the GOMESFileModel.
public String	**getToolTipText**() Returns the text to be shown in the tooltip for the GOMESFileModel object.
public String	**getTypeDescription**() Returns a description of the object's type.
public CoreSet	**getTypes**() Returns the objects GOMESTypes.
public void	**setLargeIcon**(ImageIcon largeIcon) Sets the large icon to be used showing the GOMESFileModel.

Returns	Description
public void	**setSmallIcon**(ImageIcon smallIcon) Sets the small icon to be used showing the GOMESFileModel.

gomes.model.GOMESFolderModel

java.lang.Object
gomes.core.RPCObject
gomes.core.CoreOMObject
gomes.core.CoreCollection

public *GOMESFolderModel*
extends CoreCollection
implements GOMESObject, GOMESCollection, GOMESTableModel

Model of folder views.

Fields

Type	Description
protected static Class[]	**classTypes**
protected static String[]	**columnName**

Constructors

Description
GOMESFolderModel(CoreCollection coreCollection) Constructs a new GOMESFolderModel.
GOMESFolderModel(CoreCollection coreCollection, ImageIcon smallIcon, ImageIcon largeIcon) Constructs a new GOMESFolderModel.
GOMESFolderModel(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, int size, Date creationDate, Date modificationDate) Constructs a new GOMESFolderModel.
GOMESFolderModel(String objectID, XmlRpcMethodHandler methodHandler, String label, int accessMode, int size, Date creationDate, Date modificationDate, ImageIcon smallIcon, ImageIcon largeIcon) Constructs a new GOMESFolderModel.

Methods

Returns	Description
public void	**addChangeListener**(ChangeListener listener) Adds a new ChangeListener to the folder. The listener will be informed every time the data stored in the folder changes.
public void	**addRestrictedRPCObjectChangeListener**(RPCObjectChangeListener listener) Adds an RPCObjectChangeListener to the GOMESFolderModel (objects within the folder are not included).
public void	**addRPCObjectChangeListener**(RPCObjectChangeListener listener) Adds an RPCObjectChangeListener to the GOMESFolderModel. The listener will also be informed about changes on objects the folder contains.
public void	**addTableModelListener**(TableModelListener listener) Adds a listener to the list of objects being notified each time a change to the data model occurs.
public CoreCollection	**copy**() Returns a copy of the GOMESFolderModel.
public Enumeration	**elements**() Returns all objects the GOMESFolderModel contains.
public CoreObject	**findFirst**(CoreCondition condition) Returns the first GOMESFileModel fulfilling the specified condition.
protected void	**fireChange**() Sends a 'ChangeEvent' to all the ChangeListeners listening for changes to the data stored in the OMXFolder.

Returns	Description
public void	**fireTableCellUpdated**(int row, int column) Notifies all listeners that the value of the cell at (row, column) has been updated.
public void	**fireTableChanged**(TableModelEvent e) Forward the given notification event to all 'TableModelListeners' registerd for this table model.
public void	**fireTableDataChanged**() Notifies all listeners that all cell values in the table's rows may have changed.
public void	**fireTableRowsDeleted**(int firstRow, int lastRow) Notfies all listeners that rows in the (inclusive) range 'firstRow' - 'lastRow' have been deleted.
public void	**fireTableRowsInserted**(int firstRow, int lastRow) Notfies all listeners that rows in the (inclusive) range 'firstRow' - 'lastRow' have been inserted.
public void	**fireTableRowsUpdated**(int firstRow, int lastRow) Notfies all listeners that rows in the (inclusive) range 'firstRow' - 'lastRow' have been updated.
public void	**fireTableStructureChanged**() Notfies all listeners that the table's structure has changed.
public CoreSet	**getAllSubcollections**() Returns all subcollections of the GOMESFolderModel.
public CoreSet	**getAllSupercollections**() Returns all supercollections of the GOMESFolderModel.
public CoreSet	**getAssocsByDomain**() Returns a set of all the GOMESAssociationModels having this collection as domain collection.
public CoreSet	**getAssocsByRange**() Returns a set of all the GOMESAssociationModels having this collection as range collection.
public String	**getCapitalizedTypeDescription**() Returns a description of the object's type (first letter is capitalized).
public Class	**getColumnClass**(int column) Returns the lowest common denominator class (ancestor class) in the column which is used to by the table to set up a renderer and an editor for the column.
public int	**getColumnCount**() Returns the number of collumns managed by the data model.
public String	**getColumnName**(int columnIndex) Returns the name of the column at 'columnIndex' which is used to initialize the table's column header name.
public GOMESObject	**getElementAt**(int row) Returns the GOMESObject for the specified row.
public ImageIcon	**getLargeIcon**() Returns the large icon to be used showing the GOMESFolderModel.
public CoreType	**getMembertype**() Returns the membertype of the collection.
public CoreSet	**getObjects**() Returns a set containing all the GOMESFileModels of the collection.
public Enumeration	**getParents**() Returns the parents (folders) the GOMESFolderModel is part of.
public int	**getRowCount**() Returns the number of rows managed by the data model. This method should be fast, as it is used frequently by the 'JTable'.
public ImageIcon	**getSmallIcon**() Returns the small icon to be used showing the GOMESFolderModel.
public CoreSet	**getSubcollections**() Returns the subcollections of the GOMESFolderModel (only one level in the hierarchy of subcollections).
public CoreSet	**getSupercollections**() Returns the supercollections of the GOMESFolderModel (only one level in the hierarchy of supercollections).
public String	**getToolTipText**() Returns the text to be shown in the tooltip for the GOMESFolderModel object.
public String	**getTypeDescription**() Returns a description of the object's type.
public Object	**getValueAt**(int row, int column) Returns a value object for the cell at 'row' and 'column'.
public boolean	**isCellEditable**(int row, int column) Returns true is the cell at 'row' and 'column' is editable, false otherwise.
public void	**removeChangeListener**(ChangeListener listener) Removes a ChangeListener listening for changes to the data stored in the folder.
public void	**removeRestrictedRPCObjectChangeListener**(RPCObjectChangeListener listener) Removes an RPCObjectChangeListener from the GOMESFolderModel but not from the objects the folder contains.
public void	**removeRPCObjectChangeListener**(RPCObjectChangeListener listener) Removes an RPCObjectChangeListener from the GOMESFolderModel and all the objects it contains.

Returns	Description
public void	**removeTableModelListener**(TableModelListener listener) Removes a listener from the list of objects being notified each time a change to the data model occurs.
public void	**setLargeIcon**(ImageIcon largeIcon) Sets the large icon to be used showing the GOMESFolderModel.
public void	**setSmallIcon**(ImageIcon smallIcon) Sets the small icon to be used showing the GOMESFolderModel.
public void	**setValueAt**(Object value, int row, int column) Sets a value object for the cell at 'row' and 'column'.
public String	**toString**() Returns a string representation of the GOMESFolderModel content.

gomes.model.GOMESIconModel

public abstract interface *GOMESIconModel*

Interface for icon models.

Methods

Returns	Description
public String	**getLabel**() Returns the label (description) of the object.
public ImageIcon	**getLargeIcon**() Returns a large icon (32x32 pixels) for the object.
public ImageIcon	**getSmallIcon**() Returns a small icon (16x16 pixels) for the object.
public String	**getToolTipText**() Returns a tooltip text for the object.

gomes.model.GOMESInstance

java.lang.Object
  gomes.core.RPCObject
    gomes.core.CoreOMObject
      gomes.core.CoreInstance

public *GOMESInstance*
extends CoreInstance

Remote object of the 'OMCore.Instance' type specified by the OMX-FS.

Constructors

Description
GOMESInstance(CoreInstance coreInstance) Constructs a new GOMESInstance.

Methods

Returns	Description
public CoreObject	**getObject**() Returns the corresponding GOMESFileModel.
public String	**getToolTipText**() Returns the text to be shown in the tooltip for the GOMESInstance object.

Returns	Description
public CoreType	**getType**() Returns the instance's type.
public String	**toString**() Returns a string representation of the GOMESInstance content.

gomes.model.GOMESObject

public abstract interface *GOMESObject*

implements GOMESIconModel

Interface representing the base functionality of the CoreOMObject.

Methods

Returns	Description
public int	**getAccessMode**(CoreUser user) Returns the access mode of the object.
public String	**getCapitalizedTypeDescription**() Returns a description of the object's type (first letter is capitalized).
public Date	**getCreationDate**() Returns the creation date of the object.
public String	**getLabel**() Returns the label (description) of the object.
public Date	**getModificationDate**() Returns the date the object was last modified.
public Enumeration	**getParents**() Returns the parents of the object.
public int	**getSize**() Returns the size of the object.
public String	**getTypeDescription**() Returns a description of the object's type.
public void	**setLabel**(String label) Sets the label (description) of the object.
public void	**setModificationDate**() Sets the date of the last object modification to the current date.

gomes.model.GOMESTableModel

public abstract interface *GOMESTableModel*

implements TableModel

Extension of the TableModel allowing to get the object responsible for the data on a single row.

Methods

Returns	Description
public GOMESObject	**getElementAt**(int row) Returns the GOMESObject for the specified row.

gomes.model.GOMESTrash

java.lang.Object

---

public *GOMESTrash*
extends Object
implements GOMESObject

Model representing the trash.

Constructors

Description
GOMESTrash() Constructs a new GOMESTrash.
GOMESTrash(ImageIcon smallIcon, ImageIcon largeIcon) Constructs a new GOMESTrash.

Methods

Returns	Description
public int	**getAccessMode**(CoreUser user) Returns the access mode of the object.
public String	**getCapitalizedTypeDescription**() Returns a description of the object's type (first letter is capitalized).
public Date	**getCreationDate**() Returns the creation date of the object.
public String	**getLabel**() Returns the label (description) of the object.
public ImageIcon	**getLargeIcon**() Returns the large icon to be used showing the GOMESTrash.
public Date	**getModificationDate**() Returns the date the object was last modified.
public Enumeration	**getParents**() Returns the parents of the trash (the trash has no parents).
public int	**getSize**() Returns the size of the object.
public ImageIcon	**getSmallIcon**() Returns the small icon to be used showing the GOMESTrash.
public String	**getToolTipText**() Returns the text to be shown in the tooltip for the GOMESFileModel object.
public String	**getTypeDescription**() Returns a description of the object's type.
public void	**setLabel**(String label) Sets the label (description) of the object.
public void	**setLargeIcon**(ImageIcon largeIcon) Sets the large icon to be used showing the GOMESTrash.
public void	**setModificationDate**() Sets the date of the last object modification to the current date.
public void	**setSmallIcon**(ImageIcon smallIcon) Sets the small icon to be used showing the GOMESTrash.

gomes.model.GOMESType

java.lang.Object
  gomes.core.RPCObject
    gomes.core.CoreOMObject
      gomes.core.CoreType

---

public *GOMESType*
extends CoreType

Remote object of the 'OMCore.Type' type specified by the OMX-FS.

Constructors

Description
GOMESType(CoreType coreType) Constructs a new GOMESType.

Methods

Returns	Description
public CoreType	**copy**() Returns a copy of the GOMESType.
public CoreSet	**getCollections**() Returns a set containing the GOMESFolderModels of the GOMESType.
public CoreSet	**getInstances**() Returns a set containing the GOMESInstances of the GOMESType.
public CoreSet	**getObjects**() Returns a set containing the GOMESFileModels of the GOMESType.
public CoreSet	**getSubtypes**() Returns a set containing the supertypes of the GOMES-Type.
public CoreSet	**getSupertypes**() Returns a set containing the supertypes of the GOMESType.
public String	**getToolTipText**() Returns the text to be shown in the tooltip for the GOMESType object.
public String	**toString**() Returns a string representation of the GOMESFolderModel content.

B.5 The *gomes.server* Package

The `gomes.server` package is a stub for the OMX-FS since the OMX-FS does not yet implement the XML-RPC interface. The package simulates the behavior of the OMX-FS providing data based on the local file system. `Server` is the main class which will start a new server on the specified port. The extensions of `ServerRPCObject` are representing the corresponding types of the OMX-FS. As soon as the OMX-FS implements the XML-RPC interface, it will be used instead of the `gomes.server` package.

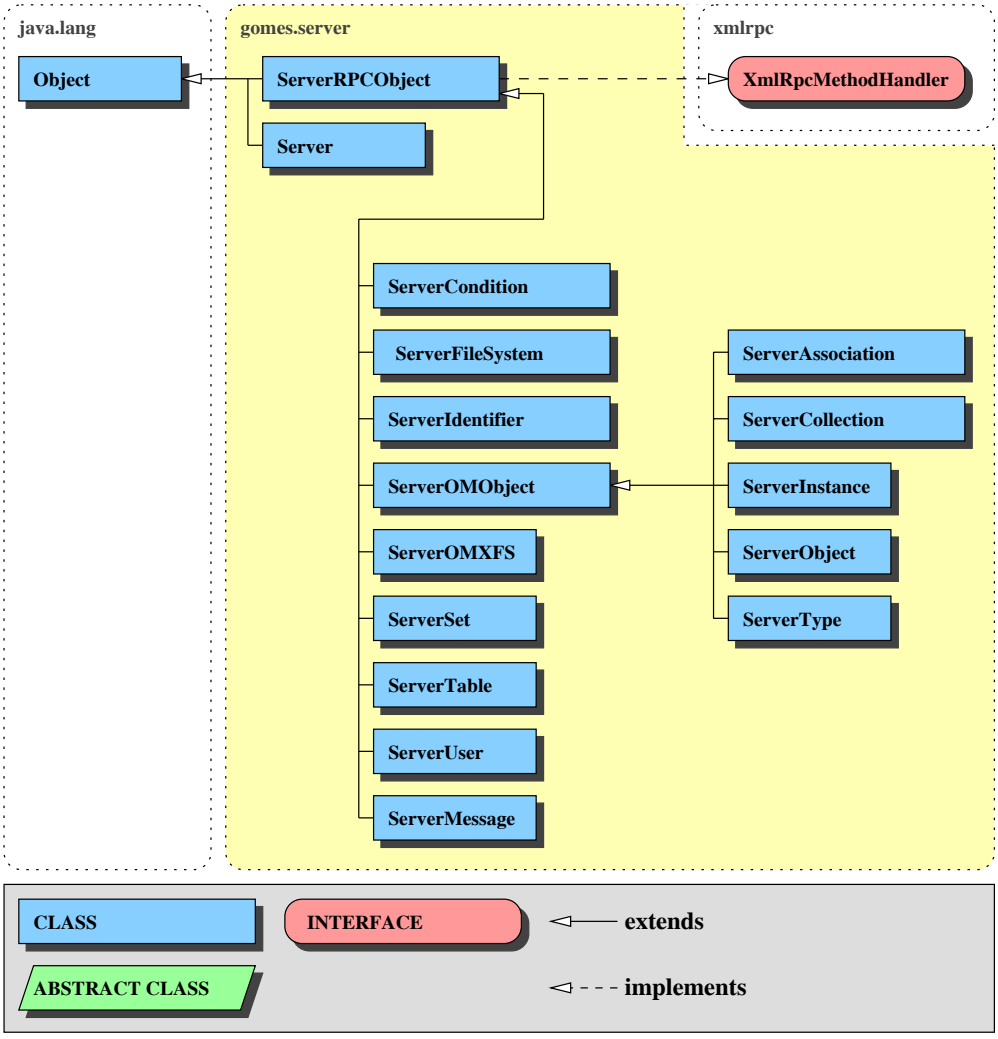

Figure B.5: The *gomes.server* package

gomes.server.Server

java.lang.Object

---

public *Server*
extends Object

Stub server providing data for GOMES.

Constructors

Description
Server()

Methods

Returns	Description
public static void	**addObject**(ServerRPCObject object)
public static String	**generateObjectID**()
public static XmlRpcMethodHandler	**getObject**(String objectID)
public static void	**main**(String[] args)

gomes.server.ServerAssociation

java.lang.Object
gomes.server.ServerRPCObject
gomes.server.ServerOMObject

---

public *ServerAssociation*
extends ServerOMObject

Server association.

Constructors

Description
ServerAssociation(String label, int accessMode, Date creationDate, Date modificationDate)

Methods

Returns	Description
public void	**addSuperassociation**(ServerAssociation superassociation)
public Boolean	**containPair**(ServerObject domainObject, ServerObject rangeObject)
public ServerAssociation	**copy**()
public ServerSet	**domainRestriction**(ServerObject rangeObject)
public void	**enumerate**(ServerMessage message)
public ServerObject	**findFirst**(ServerCondition condition)
public ServerSet	**getAllSubassociations**()
public ServerSet	**getAllSuperassociations**()
public Vector	**getData**()
public ServerSet	**getDomain**(ServerObject rangeObject)
public ServerCollection	**getDomainCollection**()
public ServerTable	**getObjects**()

Returns	Description
public ServerSet	**getRange**(ServerObject domainObject)
public ServerCollection	**getRangeCollection**()
public ServerSet	**getSubassociations**()
public ServerSet	**getSuperassociations**()
public String	**getType**()
public void	**insertPair**(ServerObject domainObject, ServerObject rangeObject)
public Object	**invokeMethod**(String method, Vector params)
public ServerSet	**rangeRestriction**(ServerObject domainObject)
public void	**removePair**(ServerObject domainObject, ServerObject rangeObject)
public void	**removeSuperassociation**(ServerAssociation superAssociation)

gomes.server.ServerCollection

java.lang.Object
  gomes.server.ServerRPCObject
    gomes.server.ServerOMObject

public *ServerCollection*
extends ServerOMObject

Server collection.

Constructors

Description
ServerCollection(OSFolder folder)

Methods

Returns	Description
public void	**addSupercollection**(ServerCollection supercollection)
public Boolean	**contain**(ServerObject object)
public ServerCollection	**copy**()
public void	**enumerate**(ServerMessage message)
public ServerObject	**findFirst**(ServerCondition condition)
public ServerSet	**getAllSubcollections**()
public ServerSet	**getAllSupercollections**()
public ServerSet	**getAssocsByDomain**()
public ServerSet	**getAssocsByRange**()
public Vector	**getData**()
public ServerType	**getMembertype**()
public ServerSet	**getObjects**()
public ServerSet	**getSubcollections**()
public ServerSet	**getSupercollections**()
public String	**getType**()
public void	**insert**(ServerObject object)
public Object	**invokeMethod**(String method, Vector params)
public void	**remove**(ServerObject object)
public void	**removeSupercollection**(ServerCollection collection)
public void	**setMembertype**(ServerType membertype)

gomes.server.ServerCondition

java.lang.Object
gomes.server.ServerRPCObject

public *ServerCondition*
extends ServerRPCObject

Server condition.

Constructors

Description
ServerCondition()

Methods

Returns	Description
public Vector	**getData**()
public String	**getType**()
public Object	**invokeMethod**(String method, Vector params)

gomes.server.ServerFileSystem

java.lang.Object
gomes.server.ServerRPCObject

public *ServerFileSystem*
extends ServerRPCObject

Server file system.

Constructors

Description
ServerFileSystem()

Methods

Returns	Description
public ServerObject	**associateFiles**(ServerAssociation association, ServerObject domainObject, ServerObject rangeObject)
public void	**closeFile**(ServerObject file)
public void	**deleteAssociation**(ServerAssociation association)
public void	**deleteCollection**(ServerCollection collection)
public void	**deleteFile**(ServerObject file)
public void	**emptyTrash**()
public Vector	**getData**()
public ServerCollection	**getFilesCollection**()
public Integer	**getFileSystemNo**()
public ServerCollection	**getTrashCollection**()
public String	**getType**()
public Object	**invokeMethod**(String method, Vector params)
public ServerAssociation	**newAssociation**(String label, ServerCollection domCollection, ServerCollection ranCollection)

Returns	Description
public ServerCollection	**newCollection**(String label, ServerCollection supercollection)
public ServerObject	**newFile**(String filename)
public ServerObject	**openFile**(ServerIdentifier identifier)
public ServerSet	**openFiles**(String filename)
public ServerObject	**openFirstFile**(String filename)
public void	**saveFileSystem**()

gomes.server.ServerIdentifier

java.lang.Object
  gomes.server.ServerRPCObject

---

public *ServerIdentifier*
extends ServerRPCObject

Server identifier.

Constructors

Description
ServerIdentifier()

Methods

Returns	Description
public Vector	**getData**()
public String	**getType**()
public Object	**invokeMethod**(String method, Vector params)

gomes.server.ServerInstance

java.lang.Object
  gomes.server.ServerRPCObject
    gomes.server.ServerOMObject

---

public *ServerInstance*
extends ServerOMObject

Server instance.

Constructors

Description
ServerInstance(String label, int accessMode, Date creationDate, Date modificationDate)

Methods

Returns	Description
public Vector	**getData**()
public ServerObject	**getObject**()
public String	**getType**()
public Object	**invokeMethod**(String method, Vector params)
public ServerType	**type**()

gomes.server.ServerMessage

java.lang.Object
  gomes.server.ServerRPCObject

---

public *ServerMessage*
extends ServerRPCObject

Server message.

Constructors

Description
ServerMessage()

Methods

Returns	Description
public Vector	**getData**()
public String	**getType**()
public Object	**invokeMethod**(String method, Vector params)

gomes.server.ServerObject

java.lang.Object
  gomes.server.ServerRPCObject
    gomes.server.ServerOMObject

---

public *ServerObject*
extends ServerOMObject

Server object.

Constructors

Description
ServerObject(OSFile file)
ServerObject(String label, int accessMode, int size, Date creationDate, Date modificationDate)

Methods

Returns	Description
public ServerObject	**copy**()
public ServerInstance	**findFirstInstance**(ServerCondition condition)
public ServerSet	**getAllAssocsByDomain**()
public ServerSet	**getAllAssocsByRange**()
public ServerSet	**getAssocsByDomain**()
public ServerSet	**getAssocsByRange**()
public ServerSet	**getCollections**()
public Vector	**getData**()
public ServerInstance	**getInstance**(ServerType context)
public ServerSet	**getInstances**()
public String	**getType**()
public ServerSet	**getTypes**()
public Object	**invokeMethod**(String method, Vector params)
public void	**notifyInstances**(ServerMessage message)

gomes.server.ServerOMObject

java.lang.Object
gomes.server.ServerRPCObject

---

public *ServerOMObject*
extends ServerRPCObject

Server OMObject.

Constructors

Description
ServerOMObject(String label, int accessMode, Date creationDate, Date modificationDate)

Methods

Returns	Description
public Integer	**getAccessMode**(ServerUser user)
public Date	**getCreationDate**()
public Vector	**getData**()
public String	**getLabel**()
public Date	**getModificationDate**()
public String	**getType**()
public Object	**invokeMethod**(String method, Vector params)
public void	**setLabel**(String label)
public void	**setModificationDate**()
public String	**toString**() Returns a string representation of the OMObject's content.

gomes.server.ServerOMXFS

java.lang.Object
gomes.server.ServerRPCObject

public *ServerOMXFS*
extends ServerRPCObject

Server OMXFS.

Constructors

Description
ServerOMXFS()

Methods

Returns	Description
public void	**deleteFileSystem**(ServerUser user, ServerFileSystem fileSystem)
public Vector	**getData**()
public String	**getType**()
public ServerUser	**getUser**(String userName, String password)
public Object	**invokeMethod**(String method, Vector params)
public ServerFileSystem	**login**(String userID, int fileSystemNo)
public void	**logout**(ServerUser user, int fileSystemNo)
public ServerFileSystem	**newFileSystem**(ServerUser creator)

gomes.server.ServerRPCObject

java.lang.Object

public abstract *ServerRPCObject*
extends Object
implements XmlRpcMethodHandler

Base type of all server objects.

Constructors

Description
ServerRPCObject()

Methods

Returns	Description
protected Object	**createAnswer**(Object object)
public abstract Vector	**getData**()
public String	**getObjectID**()
public abstract String	**getType**()
public abstract Object	**invokeMethod**(String methodName, Vector parameters)
public String	**toString**() Returns a string representation of the OMObject's content.

gomes.server.ServerSet

java.lang.Object
  gomes.server.ServerRPCObject

---

public *ServerSet*
extends ServerRPCObject

Server set.

Constructors

Description
ServerSet(Vector elements)

Methods

Returns	Description
public Vector	**getData**()
public String	**getType**()
public Object	**invokeMethod**(String method, Vector params)

gomes.server.ServerTable

java.lang.Object
  gomes.server.ServerRPCObject

---

public *ServerTable*
extends ServerRPCObject

Server table.

Constructors

Description
ServerTable()

Methods

Returns	Description
public Vector	**getData**()
public String	**getType**()
public Object	**invokeMethod**(String method, Vector params)

gomes.server.ServerType

java.lang.Object
  gomes.server.ServerRPCObject
    gomes.server.ServerOMObject

---

public *ServerType*
extends ServerOMObject

Server type.

Constructors

Description
ServerType(String label, int accessMode, Date creationDate, Date modificationDate)

Methods

Returns	Description
public void	**addSupertype**(ServerType type)
public ServerType	**copy**()
public ServerSet	**getCollections**()
public Vector	**getData**()
public ServerSet	**getInstances**()
public ServerSet	**getObjects**()
public ServerSet	**getSubtypes**()
public ServerSet	**getSupertypes**()
public String	**getType**()
public Object	**invokeMethod**(String method, Vector params)
public void	**removeSupertype**(ServerType supertype)

gomes.server.ServerUser

java.lang.Object
  gomes.server.ServerRPCObject

---

public *ServerUser*
extends ServerRPCObject

Server user.

Constructors

Description
ServerUser()

Methods

Returns	Description
public Vector	**getData**()
public String	**getType**()
public Object	**invokeMethod**(String method, Vector params)

B.6 The *gomes.util* Package

The `gomes.util` package contains a variety of tools. `BooleanComparator`, `DateComparator`, `IntegerComparator`, `LongComparator` and `StringComparator` compare objects of the corresponding type. Interface `DataString` is

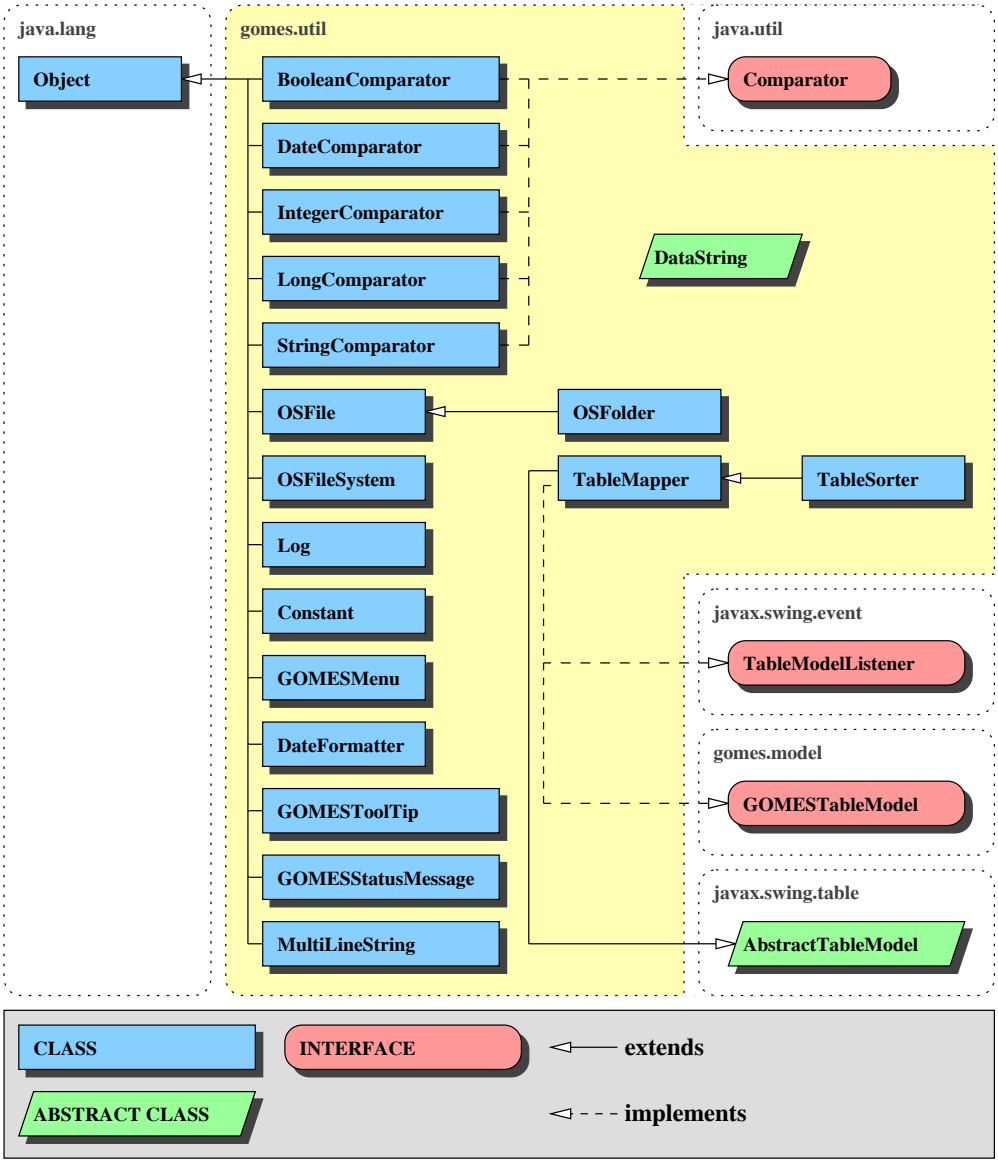

Figure B.6: The *gomes.util* package

implemented by classes providing a special string representation. The `Log` class should be used instead of printing to the standard `System.out` since it enhances the logging process. `Constant` defines some constants of GOMES. The `GOMESMenu` class is used to build all menu entries whereas `GOMESToolTip` is used

for tooltips and `GOMESStatusMessage` for the messages in the status line. The use of these three classes guarantees a consistent look of menus, tooltips and messages in the whole GOMES system. `DateFormatter` is used to format dates (e.g. it allows a special representation of the current day). MultiLineString divides a string into parts of a specified maximal length. `OSFile`, `OSFolder` and `OSFileSystem` are used to internalize and build a model of the local file system. `TableMapper` is a wrapper class for table models which can be used to modify (e.g. to sort) the data in a virtual model (uses the decorator design pattern, see [7]) whereas `TableSorter` is a specific extension of `TableMapper` allowing to sort a table using a quick sort algorithm.

gomes.util.BooleanComparator

java.lang.Object

public *BooleanComparator*
extends Object
implements Comparator

Comparator for boolean objects.

Constructors

Description
BooleanComparator()

Methods

Returns	Description
public int	**compare**(Object object1, Object object2) Compares two boolean values.

gomes.util.Constant

java.lang.Object

public *Constant*
extends Object

Some constants of the GOMES system.

Fields

Type	Description
public static final Date	**DATE_INIT**
public static final int	**DEFAULT_TOOLTIP_SIZE**
public static final ImageIcon	**ICON_INIT**
public static final int	**INT_INIT**
public static final long	**LONG_INIT**
public static final String	**STRING_INIT**

Constructors

Description
Constant()

gomes.util.DataString

public abstract interface *DataString*

String representation of an object.

Methods

Returns	Description
public String	**getString**(Object object) Returns the string representation of an object.

gomes.util.DateComparator

java.lang.Object

public *DateComparator*
extends Object
implements Comparator

Compares two date objects.

Constructors

Description
DateComparator()

Methods

Returns	Description
public int	**compare**(Object date1, Object date2) Compares two date values.

gomes.util.DateFormatter

java.lang.Object

public *DateFormatter*
extends Object

Formatted string representation of a date.

Constructors

Description
DateFormatter()

Methods

Returns	Description
public static String	**getString**(Date date) Returns a formatted string representation of a date. The date of the current day will be handled especially, i.e. not the date but the string 'today' will be returned.

gomes.util.GOMESMenu

java.lang.Object

---

public *GOMESMenu*
extends Object

Factory responsible to create the menu entries of the whole GOMES system. Guarantees consistency of all menu entries.

Fields

Type	Description
public static final int	**MNEMONIC_ABOUT**
public static final int	**MNEMONIC_ADOPT_TO_LEFT**
public static final int	**MNEMONIC_ADOPT_TO_RIGHT**
public static final int	**MNEMONIC_ASSOCIATION**
public static final int	**MNEMONIC_CLONE**
public static final int	**MNEMONIC_CLOSE**
public static final int	**MNEMONIC_COPY**
public static final int	**MNEMONIC_DELETE**
public static final int	**MNEMONIC_DELETE_FILE_SYSTEM**
public static final int	**MNEMONIC_EMPTY_TRASH**
public static final int	**MNEMONIC_EXIT**
public static final int	**MNEMONIC_HELP**
public static final int	**MNEMONIC_INFO_LOGGING**
public static final int	**MNEMONIC_LEFT_FOLDER**
public static final int	**MNEMONIC_LOGGING**
public static final int	**MNEMONIC_METHOD_LOGGING**
public static final int	**MNEMONIC_NEW_ASSOCIATION**
public static final int	**MNEMONIC_NEW_FILE**
public static final int	**MNEMONIC_NEW_FILE_MANAGER**
public static final int	**MNEMONIC_NEW_FILE_SYSTEM**
public static final int	**MNEMONIC_NEW_FOLDER**
public static final int	**MNEMONIC_OBJECT**
public static final int	**MNEMONIC_OPEN**
public static final int	**MNEMONIC_OPTIONS**
public static final int	**MNEMONIC_PASTE**
public static final int	**MNEMONIC_PROPERTY**
public static final int	**MNEMONIC_REMOVE**
public static final int	**MNEMONIC_RENAME**
public static final int	**MNEMONIC_RIGHT_FOLDER**
public static final int	**MNEMONIC_SYSTEM**
public static final int	**MNEMONIC_THEMES**
public static final int	**MNEMONIC_TOOLTIP**
public static final int	**MNEMONIC_XMLRPC_LOGGING**

Constructors

Description
GOMESMenu()

Methods

Returns	Description
public static JMenuItem	**getAboutItem**() Returns the menu item to be used to show about information.
public static JMenuItem	**getAdoptLeftFolderItem**() Returns the menu item to be used to adopt the left folder.
public static JMenuItem	**getAdoptRightFolderItem**() Returns the menu item to be used to adopt the right folder.
public static JMenu	**getAssociationMenu**() Returns the menu to be used for associations.
public static JMenuItem	**getCloneItem**() Returns the menu item to be used to clone an object.
public static JMenuItem	**getCopyItem**() Returns the menu item to be used to copy an object.
public static JMenuItem	**getDeleteFilesystemItem**() Returns the menu item to be used to delete a file system.
public static JMenuItem	**getDeleteItem**() Returns the menu item to be used to delete an object.
public static JMenu	**getDesktopThemesItem**() Returns the menu to be used for desktop themes.
public static JMenuItem	**getEmptyTrashItem**() Returns the menu item to be used to empty the trash.
public static JMenuItem	**getExitItem**() Returns the menu item to be used to exit the application.
public static JMenu	**getHelpMenu**() Returns the menu to be used for help.
public static JCheckBoxMenuItem	**getInformationLoggingItem**(boolean initValue) Returns the menu item to be used to enable/disable information logging.
public static JMenu	**getLeftFolderMenu**() Returns the menu to be used for operations on the left folder of the explorer view.
public static JMenu	**getLoggingMenu**() Returns the menu to be used for logging.
public static JCheckBoxMenuItem	**getMethodCallLoggingItem**(boolean initValue) Returns the menu item to be used to enable/disable method logging.
public static JMenuItem	**getNewAssociationItem**() Returns the menu utem to be used for the creation of new associations.
public static JMenuItem	**getNewFileItem**() Returns the menu item to be used for the creation of new files.
public static JMenuItem	**getNewFileManagerItem**() Returns the menu item to be used to open a new file manager.
public static JMenuItem	**getNewFilesystemItem**() Returns the menu item to be used to create a new file system.
public static JMenuItem	**getNewFolderItem**() Returns the menu item to be used for the creation of new folders.
public static JMenu	**getObjectMenu**() Returns the menu to be used for operations on objects.
public static JMenuItem	**getOpenItem**() Returns the menu item to be used to open an object.
public static JMenu	**getOptionMenu**() Returns the menu to be used to for options.
public static JMenuItem	**getPasteItem**() Returns the menu item to be used to paste an object.
public static JMenuItem	**getPropertyItem**() Returns the menu item to be used to show properties.
public static JMenuItem	**getRemoveItem**() Returns the menu item to be used to remove an object from another object.
public static JMenuItem	**getRenameItem**() Returns the menu item to be used to rename an object.
public static JMenu	**getRightFolderMenu**() Returns the menu to be used for operations on the right folder of the explorer view.
public static JMenu	**getSystemMenu**() Returns the menu to be used for system properties.
public static JCheckBoxMenuItem	**getToolTipItem**(boolean initValue) Returns the menu item to be used to enable/disable tool tips.
public static JCheckBoxMenuItem	**getXmlRpcLoggingItem**(boolean initValue) Returns the menu item to be used to enable/disable XML-RPC logging.

gomes.util.GOMESStatusMessage

java.lang.Object

public *GOMESStatusMessage*
extends Object

Factory responsible for the status messages of the GOMES system. Guarantees consistency of all status messages.

Constructors

Description
GOMESStatusMessage()

Methods

Returns	Description
public static final String	**changeLabel**(GOMESObject object, String newLabel) Returns the message to be used if the label of an object is changed.
public static final String	**changeToolTip**(boolean value) Returns the message to be used if the tooltip property is changed.
public static final String	**clipboardAssociationToAssociation**(GOMESAssociationModel superAssociation, GOMESAssociationModel association) Returns the message to be used if an association is copied from the clipboard to an association.
public static final String	**clipboardFileToFolder**(GOMESFolderModel folder, GOMESFileModel file) Returns the message to be used if a file is copied from the clipboard to another folder.
public static final String	**clipboardFolderToFolder**(GOMESFolderModel superFolder, GOMESFolderModel folder) Returns the message to be used if a folder is copied from the clipboard to a folder.
public static final String	**cloneObject**(GOMESObject object) Returns the message to be used if an object is cloned.
public static final String	**copyToClipboard**(GOMESObject object) Returns the message to be used if an object is copied to the clipboard.
public static final String	**deleteObject**(GOMESObject object) Returns the message to be used if an object is deleted.
public static final String	**newAssociation**(String associationLabel, String domainObjectLabel, String rangeObjectLabel) Returns the message to be used if two files are associated.
public static final String	**newFileImpossible**() Returns the message to be used if it is impossible to create a new file.
public static final String	**newFileToFolder**(String filename, GOMESFolderModel folder) Returns the message to be used if a new file is added to a folder.
public static final String	**newFolderImpossible**() Returns the message to be used if it is impossible to create a new folder.
public static final String	**newFolderToFolder**(String folderName, GOMESFolderModel folder) Returns the message to be used if a new folder is added to a folder.
public static final String	**openObject**(GOMESObject object) Returns the message to be used if an object is opened.
public static final String	**removeFileFromFolder**(GOMESFolderModel folder, GOMESFileModel file) Returns the message to be used if a file is removed from a folder.
public static final String	**removeFromAssociation**(GOMESAssociationModel association, GOMESFileModel domainFile, GOMESFileModel rangeFile) Returns the message to be used if two associated files are removed.
public static final String	**removeSubfolder**(GOMESFolderModel folder, GOMESFolderModel subfolder) Returns the message to be used if a subfolder is removed from its superfolder.
public static final String	**removeSuperfolder**(GOMESFolderModel folder, GOMESFolderModel superfolder) Returns the message to be used if a superfolder is removed from its subfolder.

gomes.util.GOMESToolTip

java.lang.Object

public *GOMESToolTip*
extends Object

Factory responsible for the tool tips of the GOMES system. Guarantees consistency of all tooltips.

Fields

Type	Description
public static final String	**ABOUT**
public static final String	**ADOPT_LEFT_FOLDER**
public static final String	**ADOPT_RIGHT_FOLDER**
public static final String	**ASSOCIATION**
public static final String	**CLONE**
public static final String	**COPY**
public static final String	**DELETE**
public static final String	**DELETE_FILESYSTEM**
public static final String	**DESKTOP_THEMES**
public static final String	**EMPTY_TRASH**
public static final String	**EXIT**
public static final String	**HELP**
public static final String	**INFORMATION_LOGGING**
public static final String	**LEFT_FOLDER**
public static final String	**LOGGING**
public static final String	**METHOD_CALL_LOGGING**
public static final String	**NEW_ASSOCIATION**
public static final String	**NEW_FILE**
public static final String	**NEW_FILE_MANAGER**
public static final String	**NEW_FILESYSTEM**
public static final String	**NEW_FOLDER**
public static final String	**OBJECT**
public static final String	**OPEN**
public static final String	**OPTION**
public static final String	**PASTE**
public static final String	**PROPERTIES**
public static final String	**REMOVE**
public static final String	**RENAME**
public static final String	**RIGHT_FOLDER**
public static final String	**SYSTEM**
public static final String	**TOOLTIP**
public static final String	**XML_RPC_LOGGING**

Constructors

Description
GOMESToolTip()

gomes.util.IntegerComparator

java.lang.Object

public *IntegerComparator*
extends Object
implements Comparator

Comparator for integer objects.

Constructors

Description
IntegerComparator()

Methods

Returns	Description
public int	**compare**(Object integer1, Object integer2) Compares two integer values.

gomes.util.Log

java.lang.Object

public *Log*
extends Object

Tool for additional outputs. A good compiler will eliminate the method calls if the corresponding constant is set to false. Exception logging can not be manipulated and will allways produce an output!

Constructors

Description
Log()

Methods

Returns	Description
public static final void	**printException**(String message, Exception e) Logging output of exceptions if EXCEPTION_LOGGING is true.
public static final void	**printInformation**(String output) Logging output of additional information if informationLogging is true.
public static final void	**printMethod**(String output) Logging output of method calls if methodDebugging is true.
public static final void	**printRPC**(Object object, String method, String params) Debugging output of remote procedure calls if rpcDebugging is true.
public static final void	**printSecurity**(String output) Logging output of security information if securityLogging is true.
public static void	**setInformationLogging**(boolean informationLogging) Sets the state of 'information logging'. If information logging is set to true, additional information will be logged.
public static void	**setLogging**(boolean logging) Sets the overall logging state. If this state is set true, the most possible logging will be performed.
public static void	**setMethodLogging**(boolean methodLogging) Sets the state of 'method logging'. If method logging is set to true, method calls will produce additional output.

Returns	Description
public static void	**setSecurityLogging**(boolean securityLogging) Sets the state of 'security logging'. If security logging is set to true, additional security information will be logged.
public static void	**setXmlRpcLogging**(boolean xmlRpcLogging) Sets the state of 'XML-RPC logging'. If XML-RPC logging is set to true, XML-RPC calls will be logged.

gomes.util.LongComparator

java.lang.Object

public *LongComparator*
extends Object
implements Comparator

Comparator for long objects.

Constructors

Description
LongComparator()

Methods

Returns	Description
public int	**compare**(Object long1, Object long2) Compares two long values.

gomes.util.MultiLineString

java.lang.Object

public *MultiLineString*
extends Object

Divides a string in parts of a specified maximal size. The parts will will be separated by a '\n'.

Constructors

Description
MultiLineString()

Methods

Returns	Description
public static String	**getString**(String string, int maxSize) Returns a 'multi line string' each line smaller than the specified maximum size. The lines will be separated by a "\n".

gomes.util.OSFile

java.lang.Object

public *OSFile*
extends Object

File based on the local file system used to build up the stub server's file system.

Constructors

Description
OSFile(String label, int size, Date creationDate, Date modificationDate) Creates a new OSFile.

Methods

Returns	Description
public void	**addParentFolder**(OSFolder folder) Adds a parent folder to the object.
public int	**compareTo**(Object object) Compares the label of this object to the specified object.
public int	**getAccessMode**() Returns the access mode of the object.
public Date	**getCreationDate**() Returns the creation date of the object.
public String	**getLabel**() Returns the label (description) of the object.
public Date	**getModificationDate**() Returns the date the object was last modificated.
public Enumeration	**getParentFolders**() Returns the parent folder of the object.
public int	**getSize**() Returns the size of the object.
public void	**setCreationDate**(Date creationDate) Sets the creation date of the object.
public void	**setLabel**(String label) Sets the label (description) for the object.
public void	**setModificationDate**(Date modificationDate) Sets the last modification date of the object.
public void	**setSize**(int size) Sets the size of the object.
public String	**toString**() Returns a string representation of the GOMESObject's content.

gomes.util.OSFileSystem

java.lang.Object

public *OSFileSystem*
extends Object

File system based on the local file system building up the stub server's file system.

Constructors

Description
OSFileSystem(String path) Constructs a new OSFileSystem.

Methods

Returns	Description
public OSFolder	**getRootFolder**() Returns the root folder of the OSFileSystem.

gomes.util.OSFolder

java.lang.Object
gomes.util.OSFile

public *OSFolder*
extends OSFile

Folder based on the local file system used to build up the stub server's file system.

Constructors

Description
OSFolder(String label, int size, Date creationDate, Date lastModifiedDate) Constructs a new OSFolder.

Methods

Returns	Description
public void	**addFile**(OSFile file) Adds an OSFile to this folder.
public void	**addSubfolder**(OSFolder folder) Adds a folder to this folder.
public Enumeration	**getFiles**() Returns all files the folder contains.
public Enumeration	**getSubfolders**() Returns the subfolders of the folder.

gomes.util.StringComparator

java.lang.Object

public *StringComparator*
extends Object
implements Comparator

Comparator for string objects.

Constructors

Description
StringComparator()

Methods

Returns	Description
public int	**compare**(Object object1, Object object2) Compares two strings. Returns -1 if the first string is smalller than the second one, 1 if it is larger and 0 if the strings are equal.

gomes.util.TableMapper

java.lang.Object
javax.swing.table.AbstractTableModel

public *TableMapper*
extends AbstractTableModel
implements GOMESTableModel, TableModelListener

Fields

Type	Description
protected GOMESTableModel	**model**

Constructors

Description
TableMapper()

Methods

Returns	Description
public Class	**getColumnClass**(int column) Redirects the 'getColumnClass' to the real model.
public int	**getColumnCount**() Redirects the 'getColumnCount' to the real model.
public String	**getColumnName**(int column) Redirects the 'getColumnName' to the real model.
public GOMESObject	**getElementAt**(int position) Redirects the 'getElementAt' to the real model.
public GOMESTableModel	**getModel**() Returns the real model for which the mapping is done.
public int	**getRowCount**() Redirects the 'getRowCount' to the real model.
public Object	**getValueAt**(int row, int column) Redirects the 'getValue' to the real model.
public boolean	**isCellEditable**(int row, int column) Redirects the 'isCellEditable' to the real mode.
public void	**setModel**(GOMESTableModel model) Sets the model for which the mapping has to be done.
public void	**setValueAt**(Object value, int row, int column) Redirects the 'setValueAt' to the real model.
public void	**tableChanged**(TableModelEvent event) Informs all the listeners about a change in the table.

gomes.util.TableSorter

java.lang.Object
javax.swing.table.AbstractTableModel
gomes.util.TableMapper

public *TableSorter*
extends TableMapper

Wrapper to sort a table (uses the decorator design pattern).

Constructors

Description
TableSorter(GOMESTableModel model) Constructs a new sorting table wrapper.

Methods

Returns	Description
public void	**addHeaderRenderer**(JTable table) Adds a special icon header renderer to each column header of the table.
public void	**addMouseListenerToHeaderInTable**(JTable table)
public GOMESObject	**getElementAt**(int row) Maps the 'getElementAt' request to the original model.
public boolean	**getTopFolder**() Retruns the 'topFolder' property indicating if folders always have to be on top.
public Object	**getValueAt**(int row, int column) Maps the 'getValueAt' request to the original model.
public void	**setModel**(GOMESTableModel model) Sets a new real model for the TableSorter and resets the mapping list (identity mapping).
public void	**setTopFolder**(boolean topFolder) Sets the 'topFolder' property indicating if folders always have to be on top.
public void	**setValueAt**(Object value, int row, int column) Maps the 'setValueAt' request to the original model.
public void	**sort**() Sorts the table using a quicksort algorithm.
public void	**sortByColumn**(int column, boolean ascending) Sorts the table by the specified column and order.
public void	**tableChanged**(TableModelEvent event) Redirects the TableModelEvent to the original model after resetting the mapping list.

B.7 The *gomes.view* Package

The `gomes.view` package contains the different views of GOMES. `GOMESObject-IconView` builds the base class of all icon views. `GOMESObjectSmallIconView` and `GOMESObjectLargeIconView` are two concrete implementations of icon views

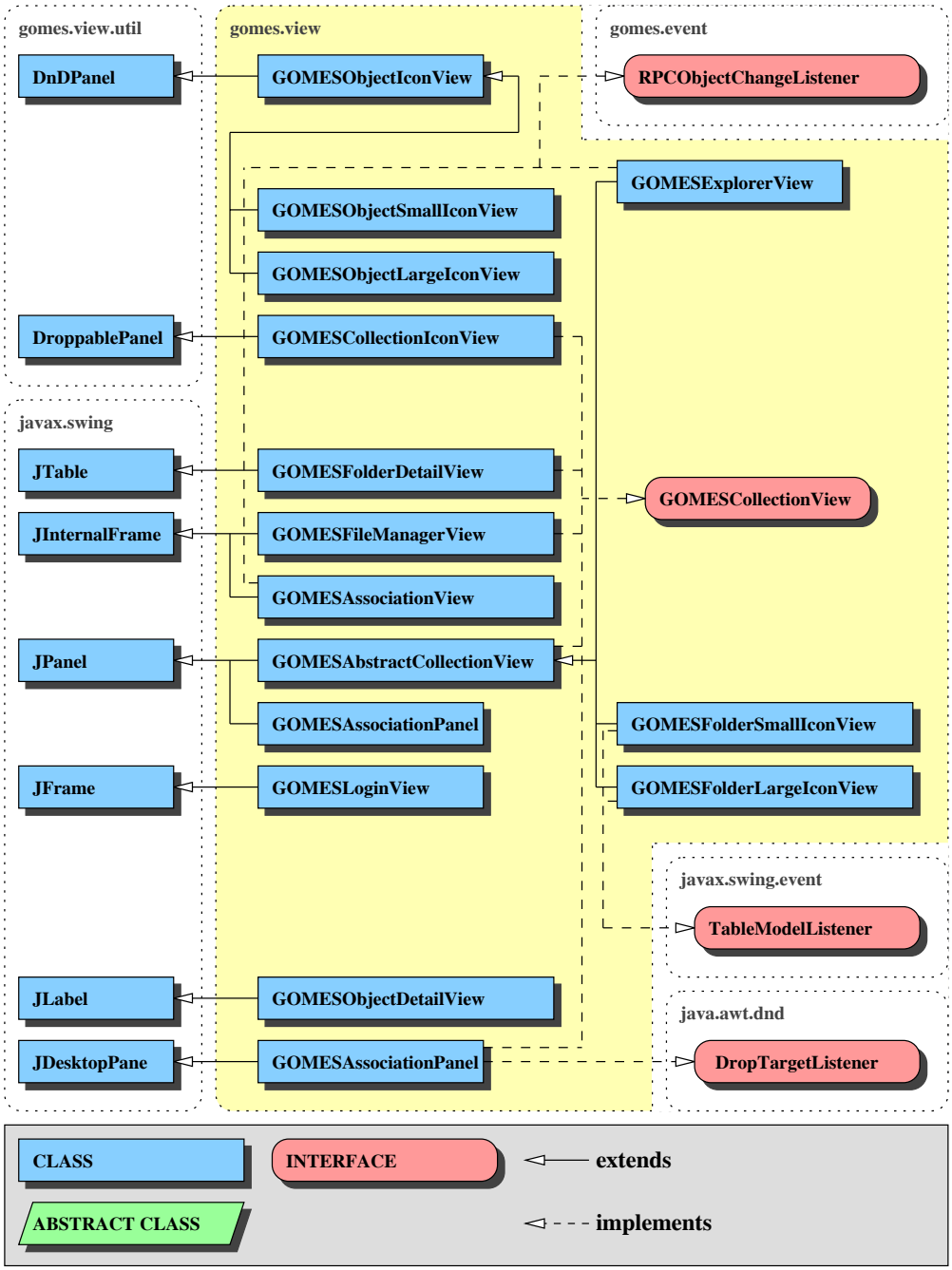

Figure B.7: The *gomes.view* package

to be used whenever a drag and drop icon is needed. The `DroppableDesktopPane` is a desktop pane with drop capabilities, i.e. icons can be placed on the desktop. The interface `GOMESCollectionInterface` guarantees a selection mechanism to be implemented by objects representing a view of a collection. `GOMESObjectDetailView` is shown whenever a file is opened. `GOMESLoginView` provides a login dialog asking for the user name and the corresponding password. `GOMESExplorerView` is the main view to show collections and files. `GOMESFolderSmallIconView`, `GOMESFolderLargeIconView` and `GOMESFolderDetailView` are main views of the `GOMESExplorerView` the user can choose. `GOMESFileManagerView` consists of two `GOMESExplorerViews`. `GOMESAssociationPanel` shows all the information concerning an association.

gomes.view.AssociationPanel

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel

---

public *AssociationPanel*
extends JPanel
implements GOMESCollectionView

Constructors

Description
AssociationPanel(GOMESObjectIconView domainIcon, GOMESObjectIconView associationIcon, GOMESObjectIconView rangeIcon)

Methods

Returns	Description
public void	**addGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Adds a GOMESObjectSelectionListener to the GOMESCollectionIconView.
public void	**fireGOMESObjectSelectionPerformed**(GOMESObjectSelectionEvent event) Notifies all listeners that have registered interest for notification on this event type.
public GOMESObject	**getSelection**() Returns the selected GOMESObject object.
public void	**initSelection**() Initializes the selection.
public boolean	**isSelected**() Returns true if the collection or one of its elements is selected.
public void	**paint**(Graphics g)
public void	**removeGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Removes a GOMESObjectSelectionListener from the GOMESCollectionIconView.
public void	**resetSelection**() Resets the current selection.

gomes.view.DroppableDesktopPane

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JLayeredPane
          javax.swing.JDesktopPane

---

public *DroppableDesktopPane*
extends JDesktopPane
implements DropTargetListener, GOMESCollectionView

Desktop pane providing drop functionality.

Constructors

Description
DroppableDesktopPane() Constructs a new DroppableDesktopPane.

Methods

Returns	Description
public void	**addGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Adds a GOMESObjectSelectionListener to the Desktop.
public void	**addToDesktop**(GOMESObjectIconView iconView, int x, int y) Adds an icon view to the desktop.
public void	**dragEnter**(DropTargetDragEvent e) A Drag operation has encountered the DropTarget.
public void	**dragExit**(DropTargetEvent e) The Drag operation has departed the DropTarget without dropping.
public void	**dragOver**(DropTargetDragEvent e) A Drag operation is ongoing on the DropTarget.
public void	**drop**(DropTargetDropEvent e) The Drag operation has terminated with a Drop on this DropTarget.
public void	**dropActionChanged**(DropTargetDragEvent e) The user has modified the current drop gesture.
public void	**fireGOMESObjectSelectionPerformed**(GOMESObjectSelectionEvent event) Notifies all listeners that have registered interest for notification on this event type.
public GOMESObject	**getSelection**() Returns the selected GOMESObject object.
public void	**initSelection**() Initializes the selection.
public boolean	**isSelected**() Returns true if the collection is selected, false otherwise.
public void	**removeGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Removes a GOMESObjectSelectionListener from the Desktop.
public void	**resetSelection**() Resets the current selection.

gomes.view.GOMESAbstractCollectionView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel

public abstract *GOMESAbstractCollectionView*
extends JPanel
implements GOMESCollectionView

Standard implementation of the GOMESCollectionView interface.

Constructors

Description
GOMESAbstractCollectionView() Constructs a new GOMESAbstractCollectionView.

Methods

Returns	Description
public void	**addGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Adds a GOMESObjectSelectionListener to the GOMES-AbstractCollectionView.
public void	**fireGOMESObjectSelectionPerformed**(GOMESObjectSelectionEvent event) Notifies all listeners that have registered interest for notification on this event type.
public void	**removeGOMESObjectSelectionListener**(GOMESObjectSelection-Listener listener) Removes a GOMESObjectSelectionListener from the GOMESAbstractCollectionView.

gomes.view.GOMESAssociationView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JInternalFrame

---

public *GOMESAssociationView*
extends JInternalFrame
implements RPCObjectChangeListener

View for associations.

Fields

Type	Description
public static final int	**DOMAIN_OBJECT**
public static final int	**RANGE_OBJECT**

Constructors

Description
GOMESAssociationView(GOMESAssociationModel association, int objectType, GOMESFileModel currentFile) Constructs a new GOMESAssociationView.

Methods

Returns	Description
public void	**initialize**() Initializes the association view.
public void	**newAssociationAction**()
public void	**newFolderAction**(GOMESFolderModel folder)
public void	**objectChanged**(RPCObjectChangeEvent event) Up-called whenever an RPCObject shown in the GOMEAssociationView has changed.

gomes.view.GOMESCollectionIconView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          gomes.view.util.DroppablePanel

---

public *GOMESCollectionIconView*
extends DroppablePanel
implements GOMESCollectionView

View of a GOMESCollection.

Fields

Type	Description
protected EventListenerList	**listenerList**

Constructors

Description
GOMESCollectionIconView(GOMESCollection model) Constructs a new view for an GOMESCollection.

Methods

Returns	Description
public void	**addGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Adds a GOMESObjectSelectionListener to the GOMESCollectionIconView.
public void	**fireGOMESObjectSelectionPerformed**(GOMESObjectSelectionEvent event) Notifies all listeners that have registered interest for notification on this event type.
public GOMESCollection	**getModel**() Returns the model of the GOMESCollectionIconView.
public GOMESObject	**getSelection**() Returns the selected icon view.
public void	**initSelection**() Initializes the selection.
public boolean	**isSelected**() Returns true is the collection or an element within it is selected.
public void	**removeGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Removes a GOMESObjectSelectionListener from the GOMESCollectionIconView.
public void	**resetSelection**() Resets the current selection.

gomes.view.GOMESCollectionView

public abstract interface *GOMESCollectionView*

Interface to be implemented by views showing a collection of objects.

Methods

Returns	Description
public void	**addGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Adds a GOMESObjectSelectionListener to the GOMESCollectionView.
public void	**fireGOMESObjectSelectionPerformed**(GOMESObjectSelectionEvent event) Notifies all listeners that have registered interest for notification on this event type.
public GOMESObject	**getSelection**() Returns the last recently selected GOMESObject (only single selection allowed).
public void	**initSelection**() Selects the first element in the collection.
public boolean	**isSelected**() Returns true if the collection is selected, false otherwise.
public void	**removeGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Removes an GOMESObjectSelectionListener from the GOMESCollectionView.
public void	**resetSelection**() Resets the selection state of the GOMESCollectionView.

gomes.view.GOMESExplorerView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          gomes.view.GOMESAbstractCollectionView

public *GOMESExplorerView*
extends GOMESAbstractCollectionView
implements RPCObjectChangeListener

View containing a GOMESFileManagerView and its super- and subfolders.

Fields

Type	Description
public static final int	**DETAIL_VIEW**
public static final int	**LARGE_ICON_VIEW**
public static final int	**SMALL_ICON_VIEW**

Constructors

Description
GOMESExplorerView(GOMESFolderModel folder, int viewType) Constructs a new GOMESExplorerView.

Methods

Returns	Description
public GOMESFolderModel	**getFolder**() Returns the folder shown in the main view.
public GOMESObject	**getSelection**() Returns the last recently selected GOMESObject (only single selection allowed).
public boolean	**getTopFolder**() Returns true if folders are shown on top, false otherwise.
public void	**initialize**() Initializes the GOMESExplorerView.
public void	**initSelection**() Selects the first element in the folder view.
public boolean	**isBottomCollectionSelected**() Returns true if the bottom collection is selected, false otherwise.
public boolean	**isFolderSelected**() Returns true if the main folder is selected, false otherwise.
public boolean	**isSelected**() Returns true if the collection is selected, false otherwise.
public boolean	**isTopCollectionSelected**() Returns true if the top collection is selected, false otherwise.
public void	**objectChanged**(RPCObjectChangeEvent event) Up-called whenever an RPCObject shown in the GOMESExplorer view has changed.
public void	**resetSelection**() Resets the selection state of the GOMESCollectionView.
public void	**setFolder**(GOMESFolderModel folder) Sets the folder viewe to the specified folder. The top view is also updated.
public void	**setTopFolder**(boolean topFolder) Sets the 'topFolder' property indicating if folders should be shown on top.
public void	**setViewType**(int viewType) Sets the type of the main view (small icons, large icons or detail view).
public void	**sortByCreationDate**() Sorts the objects of the main view by date of creation.
public void	**sortByModificationDate**() Sorts the objects of the main view by the date of their last modification.
public void	**sortByName**() Sorts the objects of the main view by filename.
public void	**sortBySize**() Sorts the objects of the main view by filesize.

gomes.view.GOMESFileManagerView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JInternalFrame

public *GOMESFileManagerView*
extends JInternalFrame
implements GOMESCollectionView

Main view for the visualization of collections containing two GOMESExplorerViews.

Constructors

Description
GOMESFileManagerView(GOMESFolderModel folder) Constructs a new GOMESFileManagerView.

Methods

Returns	Description
public void	**addGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Adds a GOMESObjectSelectionListener to the GOMESAbstractCollectionView.
public void	**fireGOMESObjectSelectionPerformed**(GOMESObjectSelectionEvent event) Notifies all listeners that have registered interest for notification on this event type.
public GOMESObject	**getSelection**() Returns the last recently selected GOMESObject (only single selection allowed).
public void	**initSelection**() Selects the first element in the folder view.
public void	**removeGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Removes a GOMESObjectSelectionListener from the GOMESAbstractCollectionView.
public void	**resetSelection**() Resets the selection state of the GOMESCollectionView.

gomes.view.GOMESFolderDetailView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JTable

---

public *GOMESFolderDetailView*
extends JTable
implements GOMESCollectionView, ChangeListener

Detailed folder view.

Fields

Type	Description
protected EventListenerList	**listenerList**

Constructors

Description
GOMESFolderDetailView(GOMESTableModel model) Constructs a new GOMESFolderDetailView.

Methods

Returns	Description
public void	**addGOMESObjectSelectionListener**(GOMESObjectSelectionListener listener) Adds a GOMESObjectSelectionListener to the GOMESFolderDetailView.
public void	**fireGOMESObjectSelectionPerformed**(GOMESObjectSelectionEvent event) Notifies all listeners that have registered interest for notification on this event type.
public int	**getPreferredColumnWidth**(TableColumn column) Returns the preferred width for column 'column'.

Returns	Description
public GOMESObject	**getSelection**() Returns the last recently selected GOMESObject (only single selection allowed).
public Point	**getToolTipLocation**(MouseEvent event) Returns the tooltip location in the receiving component coordinate system. If null is returned, Swing will choose a location.
public String	**getToolTipText**(MouseEvent event) Returns the tooltip for this GOMESFolderDetailView.
public void	**initColumnSizes**() Initializes the width of all columns.
public void	**initSelection**() Selects the first element in the GOMESFolderDetailView.
public boolean	**isSelected**() Returns true if the folder view is selected, false otherwise.
public void	**removeGOMESObjectSelectionListener**(GOMESObjectSelection-Listener listener) Removes a GOMESObjectSelectionListener from the GOMESFolderDetailView.
public void	**resetSelection**() Resets the selection state of the GOMESFolderDetail-View.
public void	**stateChanged**(ChangeEvent event) Invoked if the data in the modell changed. A new thread is generated which will update the view.

gomes.view.GOMESFolderLargeIconView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          gomes.view.GOMESAbstractCollectionView

public *GOMESFolderLargeIconView*
extends GOMESAbstractCollectionView
implements TableModelListener

Folder view using large icons.

Constructors

Description
GOMESFolderLargeIconView(GOMESTableModel model) Constructs a new GOMESFolderLarge-IconView.

Methods

Returns	Description
public GOMESTableModel	**getModel**() Returns the model containing the data of this view.
public GOMESObject	**getSelection**() Returns the selected icon view.
public void	**initSelection**() Initializes the selection.
public boolean	**isSelected**() Returns true if the collection is selected, false otherwise.
public void	**resetSelection**() Resets the current selection.
public void	**tableChanged**(TableModelEvent e) Invoked when data in the table has changed.

gomes.view.GOMESFolderSmallIconView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          gomes.view.GOMESAbstractCollectionView

---

public *GOMESFolderSmallIconView*
extends GOMESAbstractCollectionView
implements TableModelListener

Folder view using small icons.

Constructors

Description
GOMESFolderSmallIconView(GOMESTableModel model) Constructs a new view for a GOMESFolder.

Methods

Returns	Description
public GOMESTableModel	**getModel**() Returns the model containing the data of this view.
public GOMESObject	**getSelection**() Returns the selected icon view.
public void	**initSelection**() Initializes the selection.
public boolean	**isSelected**() Returns true if the collection is selected, false otherwise.
public void	**resetSelection**() Resets the current selection.
public void	**tableChanged**(TableModelEvent e) Invoked when data in the table has changed.

gomes.view.GOMESLoginView

java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          javax.swing.JFrame

---

public *GOMESLoginView*
extends JFrame

Login view of GOMES.

Constructors

Description
GOMESLoginView() Constructs a new GOMESLogonView.

Methods

Returns	Description
public Vector	**getLoginInfo**() Returns the information the user entered in the login field.

gomes.view.GOMESObjectDetailView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JLabel

---

public *GOMESObjectDetailView*
extends JLabel

Simple view of a GOMESObject.

Constructors

Description
GOMESObjectDetailView(GOMESObject model) Constructs a new view of a GOMESObject.

Methods

Returns	Description
public GOMESObject	**getModel**() Returns the model containing the data of this view.

gomes.view.GOMESObjectIconView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          gomes.view.util.DnDPanel

---

public abstract *GOMESObjectIconView*
extends DnDPanel

Base class for the visualization of a GOMESObject.

Fields

Type	Description
public static DataFlavor	**gomesObjectIconFlavor**

Constructors

Description
GOMESObjectIconView(GOMESObject model) Constructs a new GOMESObjectIconView.

Methods

Returns	Description
public void	**drop**(DropTargetDropEvent e) The Drag operation has terminated with a Drop on this DropTarget.
public GOMESIconModel	**getModel**() Returns the model containing the data of this view.
public Object	**getTransferData**(DataFlavor flavor) Returns an object which represents the data to be transferred. The class of the object returned is defined by the representation class of the flavor.
public DataFlavor[]	**getTransferDataFlavors**() Returns an array of DataFlavor objects indicating the flavors the data can be provided in. The array is ordered according to preference for providing the data (from most richly descriptive to least descriptive).
public boolean	**isDataFlavorSupported**(DataFlavor flavor) Returns whether the specified data flavor is supported or not for this object.
public boolean	**isSelected**() Indicates if the GOMESObjectIconView is selected.
public void	**setSelected**(boolean selected) Sets the selection mode of the GOMESObjectIconView. True if the GOMESObjectIconView should be selected, false otherwise.

gomes.view.GOMESObjectLargeIconView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          gomes.view.util.DnDPanel
            gomes.view.GOMESObjectIconView

---

public *GOMESObjectLargeIconView*
extends GOMESObjectIconView

Large icon view of a GOMESObject.

Constructors

Description
GOMESObjectLargeIconView(GOMESObject model) Constructs a new GOMESObjectLargeIconView.

Methods

Returns	Description
public Point	**getToolTipLocation**(MouseEvent event) Returns the tooltip location in the receiving component coordinate system. If null is returned, Swing will choose a location.
public String	**getToolTipText**() Returns the tooltip for this GOMESObjectSmallIconView.

Returns	Description
public void	**setSelected**(boolean selected) Sets the selection mode of the GOMESObjectLargeIconView. True if the GOMESObjectLargeIconView should be selected, false otherwise.

gomes.view.GOMESObjectSmallIconView

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          gomes.view.util.DnDPanel
            gomes.view.GOMESObjectIconView

public *GOMESObjectSmallIconView*
extends GOMESObjectIconView

Small icon view of a GOMESObject.

Constructors

Description
GOMESObjectSmallIconView(GOMESObject model) Constructs a new GOMESObjectSmallIconView.

Methods

Returns	Description
public Point	**getToolTipLocation**(MouseEvent event) Returns the tooltip location in the receiving component coordinate system. If null is returned, Swing will choose a location.
public String	**getToolTipText**() Returns the tooltip for this GOMESObjectSmallIconView.
public void	**setSelected**(boolean selected) Sets the selection mode of the GOMESObjectSmallIconView. True if the GOMESObjectSmallIconView should be selected, false otherwise.

B.8 The *gomes.view.util* Package

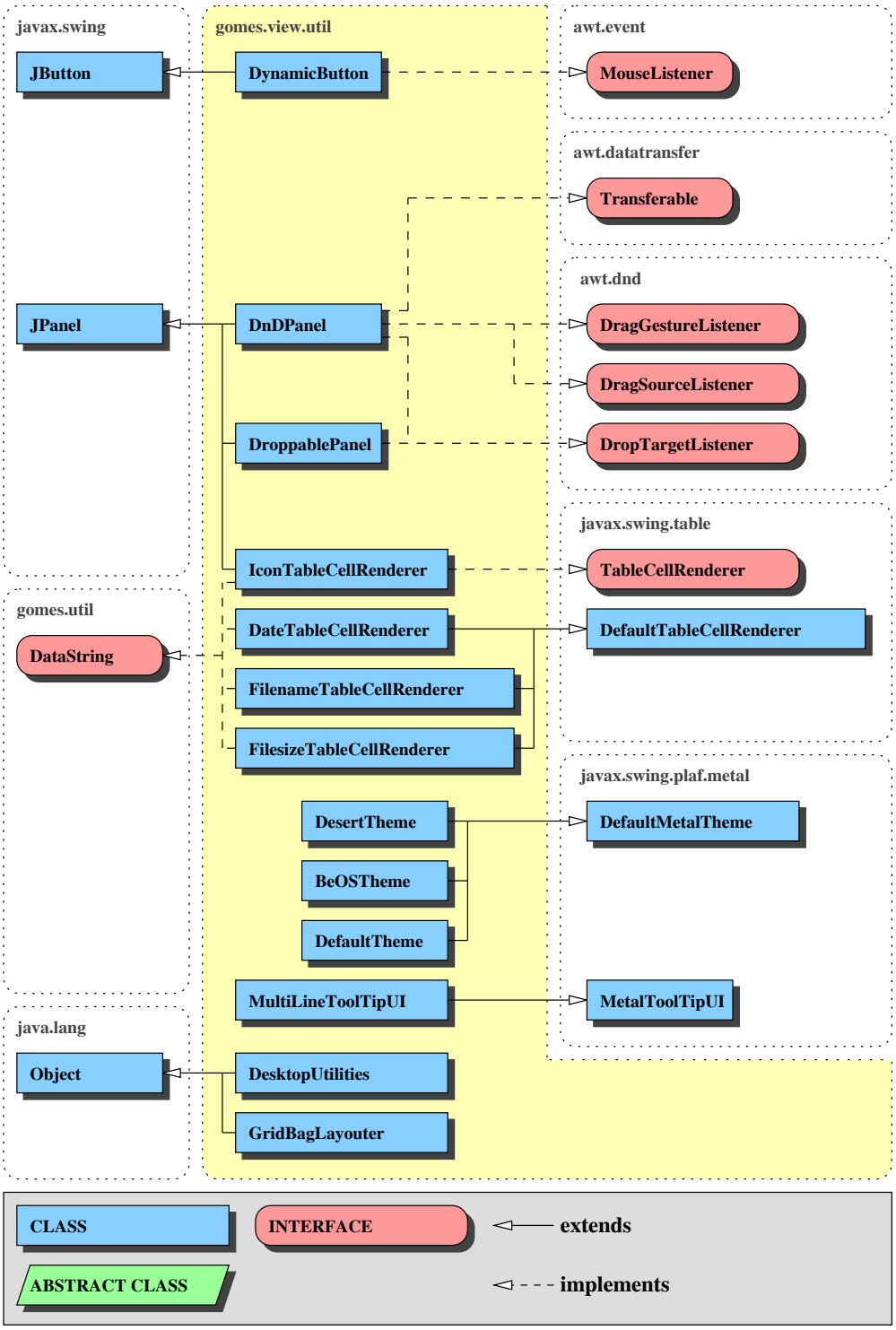

Figure B.8: The *gomes.view.util* package

The `gomes.view.util` package contains tools for the views of the GOMES system. `DynamicButton` is the button used for button bars. `DnDPanel` is the base class of views accepting drag and drop actions (e.g. icons) whereas `DroppablePanel` is the base class of views accepting drop actions only. `IconTableCellRenderer`, `DateTabelCellRenderer`, `FilenameTableCellRenderer` and `FilesizeTableCellRenderer` are customized cell renderers for the table entries. `DesktopUtilities` provides facilities to add frames to the desktop and arrange them. `GridBagLayouter` simplifies the usage of the gridbag layout. `MultiLineToolTipUI` is a customized tooltip renderer allowing to show multiple lines in a tooltip. `DesertTheme`, `BeOSTheme` and `DefaultTheme` are three desktop themes the user of GOMES can choose to customize his desktop.

gomes.view.util.BeOSTheme

java.lang.Object
  javax.swing.plaf.metal.MetalTheme
    javax.swing.plaf.metal.DefaultMetalTheme

---

public *BeOSTheme*
extends DefaultMetalTheme

Desktop theme in BeOS style defining the color schema of the desktop.

Constructors

Description
BeOSTheme()

Methods

Returns	Description
protected ColorUIResource	**getPrimary1**() Returns the primay1 color.
protected ColorUIResource	**getPrimary2**() Returns the primay2 color.
protected ColorUIResource	**getPrimary3**() Returns the primay3 color.
protected ColorUIResource	**getSecondary1**() Returns the secondary1 color.
protected ColorUIResource	**getSecondary2**() Returns the secondary2 color.
protected ColorUIResource	**getSecondary3**() Returns the secondary3 color.

gomes.view.util.DateTableCellRenderer

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JLabel
          javax.swing.table.DefaultTableCellRenderer

---

public *DateTableCellRenderer*
extends DefaultTableCellRenderer
implements DataString

Renderer for a date cell in a table.

Constructors

Description
DateTableCellRenderer() Constructs a new date table cell renderer.

Methods

Returns	Description
public String	**getString**(Object date) Returns the string representation of a 'date' object.
public Component	**getTableCellRendererComponent**(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) Returns the component to be used to draw dates by the table renderer.

gomes.view.util.DefaultTheme

java.lang.Object
  javax.swing.plaf.metal.MetalTheme
    javax.swing.plaf.metal.DefaultMetalTheme

---

public *DefaultTheme*
extends DefaultMetalTheme

Default desktop theme defining the color schema of the desktop.

Constructors

Description
DefaultTheme()

Methods

Returns	Description
protected ColorUIResource	**getPrimary1**() Returns the primay1 color.
protected ColorUIResource	**getPrimary2**() Returns the primay2 color.
protected ColorUIResource	**getPrimary3**() Returns the primay3 color.
protected ColorUIResource	**getSecondary1**() Returns the secondary1 color.
protected ColorUIResource	**getSecondary2**() Returns the secondary2 color.
protected ColorUIResource	**getSecondary3**() Returns the secondary3 color.

gomes.view.util.DesertTheme

java.lang.Object
  javax.swing.plaf.metal.MetalTheme
    javax.swing.plaf.metal.DefaultMetalTheme

---

public *DesertTheme*
extends DefaultMetalTheme

Desktop theme in desert style defining the color schema of the desktop.

Constructors

Description
DesertTheme()

Methods

Returns	Description
protected ColorUIResource	**getPrimary1**() Returns the primay1 color.
protected ColorUIResource	**getPrimary2**() Returns the primay2 color.
protected ColorUIResource	**getPrimary3**() Returns the primay3 color.
protected ColorUIResource	**getSecondary1**() Returns the secondary1 color.
protected ColorUIResource	**getSecondary2**() Returns the secondary2 color.
protected ColorUIResource	**getSecondary3**() Returns the secondary3 color.

gomes.view.util.DesktopUtilities

java.lang.Object

---

public *DesktopUtilities*
extends Object

Tool for the maintenance of the main desktop.

Constructors

Description
DesktopUtilities()

Methods

Returns	Description
public static void	**addToDesktop**(JDesktopPane desktop, JInternalFrame internalFrame) Adds an internal frame to the desktop. The added internal frame will be set to front.
public static JInternalFrame	**getInternalFrame**(Component component) Returns the internal frame the specified component resides in.
public static void	**setSelected**(JInternalFrame internalFrame) Sets the specified internal frame selected.
public static void	**toFront**(JInternalFrame internalFrame) Moves the specified internal frame to front.

gomes.view.util.DnDPanel

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel

---

public abstract *DnDPanel*
extends JPanel
implements DragSourceListener, DragGestureListener, DropTargetListener, Transferable

Panel providing drag and drop functionality.

Constructors

Description
DnDPanel() Constructs a new DnDPanel.

Methods

Returns	Description
public void	**dragDropEnd**(DragSourceDropEvent e) Invoked when the drag and drop operation completes.
public void	**dragEnter**(DragSourceDragEvent e) Invoked when the hotspot enters a platform dependent drop site.
public void	**dragEnter**(DropTargetDragEvent e) Invoked when a drag operation has encountered the DropTarget.
public void	**dragExit**(DragSourceEvent e) Invoked when the hotspot exits a platform dependent drop site.
public void	**dragExit**(DropTargetEvent e) Invoked when the drag operation has departed the DropTarget without dropping.
public void	**dragGestureRecognized**(DragGestureEvent e) When the DragGestureRecognizer recognizes a DnD action, it messages the registered DragGestureListener by invoking the 'dragGestureRecognized' method.
public void	**dragOver**(DragSourceDragEvent e) Invoked when the hotspot moves over a platform dependent drop site.
public void	**dragOver**(DropTargetDragEvent e) Invoked when a drag operation is ongoing on the DropTarget.
public void	**drop**(DropTargetDropEvent e) Invoked when the drag operation has terminated with a drop on this DropTarget.
public void	**dropActionChanged**(DragSourceDragEvent e) Invoked when the user has modified the drop gesture.
public void	**dropActionChanged**(DropTargetDragEvent e) Invoked when the user has modified the current drop gesture.

gomes.view.util.DroppablePanel

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel

public abstract *DroppablePanel*
extends JPanel
implements DropTargetListener

Panel providing drop functionality.

Constructors

Description
DroppablePanel() Constructs a new DroppablePanel.

Methods

Returns	Description
public void	**dragEnter**(DropTargetDragEvent e) Invoked when the hotspot enters a platform dependent drop site.
public void	**dragExit**(DropTargetEvent e) Invoked when the hotspot exits a platform dependent drop site.
public void	**dragOver**(DropTargetDragEvent e) Invoked when the hotspot moves over a platform dependent drop site.
public void	**drop**(DropTargetDropEvent e) Invoked when the drag operation has terminated with a drop on this DropTarget.
public void	**dropActionChanged**(DropTargetDragEvent e) Invoked when the user has modified the drop gesture.

gomes.view.util.DynamicButton

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.AbstractButton
          javax.swing.JButton

---

public *DynamicButton*
extends JButton
implements MouseListener

Dynamic image button changing its border on rollovers.

Constructors

Description
DynamicButton(String label, Icon image) Constructs a new dynamic button.
DynamicButton(String label) Constructs a new dynamic button without an image.
DynamicButton(String label, Icon image, String toolTip) Constructs a new dynamic button.

Methods

Returns	Description
public void	**mouseClicked**(MouseEvent event) Handles a clicked button.
public void	**mouseEntered**(MouseEvent event) Handles an entering of the cursor in the button area.
public void	**mouseExited**(MouseEvent event) Handles a leaving of the cursor from the button area.
public void	**mousePressed**(MouseEvent event) Handles a pressed button.
public void	**mouseReleased**(MouseEvent event) Handles a released button.

gomes.view.util.FilenameTableCellRenderer

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JLabel
          javax.swing.table.DefaultTableCellRenderer

---

public *FilenameTableCellRenderer*
extends DefaultTableCellRenderer
implements DataString

Renderer for a filename cell in a table.

Constructors

Description
FilenameTableCellRenderer() Constructs a new filename table cell renderer.

Methods

Returns	Description
public String	**getString**(Object filename) Returns the string representation of a 'filename' object.
public Component	**getTableCellRendererComponent**(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) Returns the component to be used to draw filenames by the table renderer.

gomes.view.util.FilesizeTableCellRenderer

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JLabel
          javax.swing.table.DefaultTableCellRenderer

---

public *FilesizeTableCellRenderer*
extends DefaultTableCellRenderer
implements DataString

Renderer for a filesize cell in a table.

Constructors

Description
FilesizeTableCellRenderer() Constructs a new filesize table cell renderer.

Methods

Returns	Description
public String	**getString**(Object object) Returns the string representation of a 'Long' object.
public Component	**getTableCellRendererComponent**(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) Returns the component to be used to draw filesizes by the table renderer.

gomes.view.util.GridBagLayouter

java.lang.Object

---

public *GridBagLayouter*
extends Object

Tool to use the GridBagLayout.

Constructors

Description
GridBagLayouter()

Methods

Returns	Description
public static void	**addComponent**(Container container, Component component, int top, int left, int bottom, int right) Adds a component to a container with a GridBagLayout.
public static void	**addComponent**(Container container, Component component, int gridx, int gridy, int gridwidth, int gridheight, int top, int left, int bottom, int right, double weightx, double weighty, int anchor, int fill) Adds a component to a container with a GridBagLayout.

gomes.view.util.IconTableCellRenderer

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel

---

public *IconTableCellRenderer*
extends JPanel
implements TableCellRenderer, DataString

Renderer for an ImageIcon in a table.

Constructors

Description
IconTableCellRenderer()

Methods

Returns	Description
public String	**getString**(Object object) Returns the string representation of an object.
public Component	**getTableCellRendererComponent**(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) Returns the component to be used to draw icons by the table renderer.
public void	**paint**(Graphics g) Draws the icon on the graphics context.

gomes.view.util.MultiLineToolTipUI

java.lang.Object
  javax.swing.plaf.ComponentUI
    javax.swing.plaf.ToolTipUI
      javax.swing.plaf.basic.BasicToolTipUI
        javax.swing.plaf.metal.MetalToolTipUI

---

public *MultiLineToolTipUI*
extends MetalToolTipUI

UI component to generate multi line tooltips.

Fields

Type	Description
public static final String	**LINE_SEPARATOR**
public static final int	**MARGIN**

Constructors

Description
MultiLineToolTipUI() Constructs a new MultiLineToolTipUI.

Methods

Returns	Description
public static ComponentUI	**createUI**(JComponent c) Creates a UI for a MultiLineTooltip.
public Dimension	**getPreferredSize**(JComponent component) Returns the preferred size of the multi line tooltip.
public void	**paint**(Graphics g, JComponent c) Draws the component to the specified graphics context.

B.9 The *xmlrpc* Package

The `xmlrpc` package implements the XML-RPC protocol specified by *Frontier.UserLand.Com*. It implements an object serialization and deserialization mechanism based on an existing XML parser. `XmlRpc` is the main class for

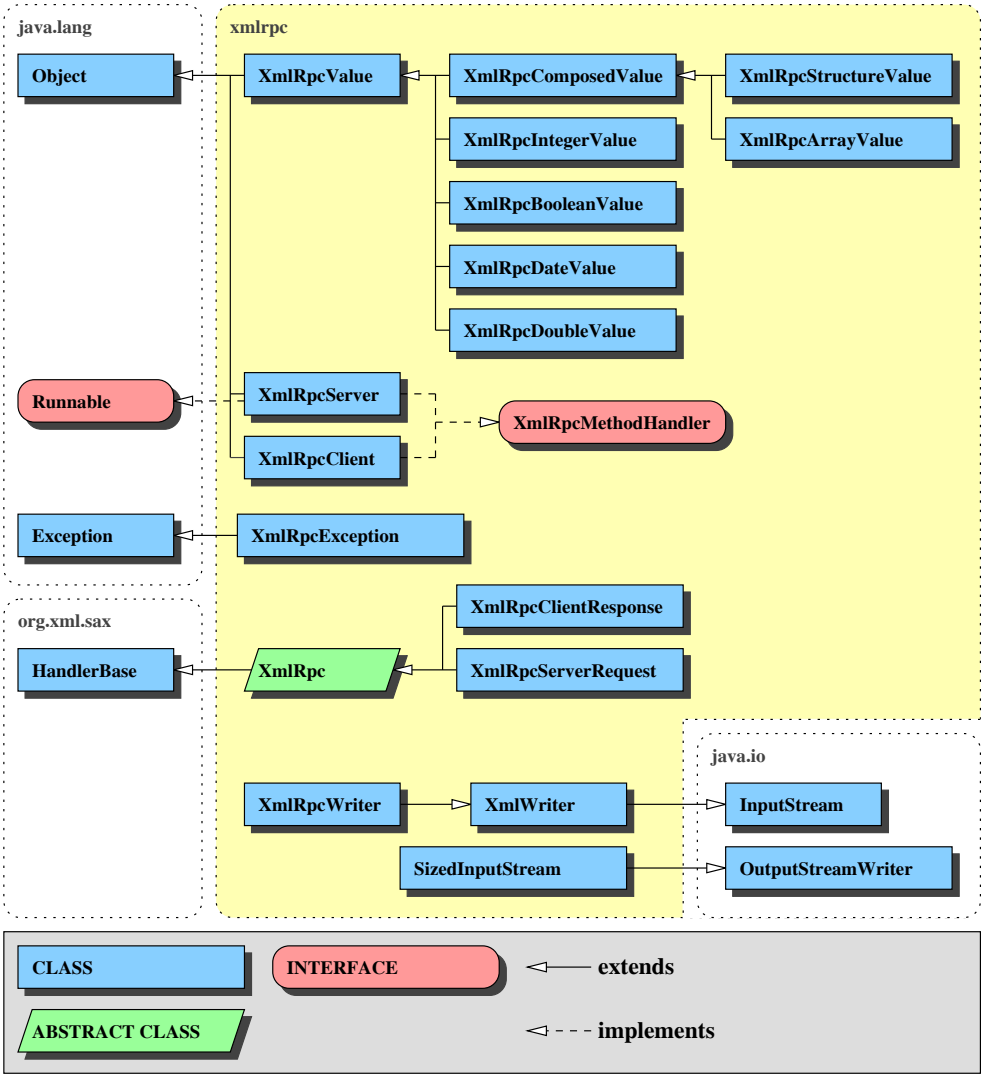

Figure B.9: The *xmlrpc* package

parsing an XML-RPC document. It is an extension of the `HandlerBase` of the corresponding SAX parser. `XmlRpcServerRequest` is used by the client to send an XML-RPC request to a server whereas `XmlRpcClientResponse` is used by the server to send the answer of an XML-RPC request to the client. `XmlWriter` is an output stream allowing to write XML documents. Based on the `XMLWriter`, the `XmlRpcWriter` builds an output stream allowing to write

XML-RPC documents. `SizedInputStream` is a tool class used by an XML-RPC server to extract the data part of an HTTP request. Every time an error occurs, an `XmlRpcException` is raised. The interface `XmlRpcMethodHandler` has to be implemented by classes acting as handlers for XML-RPC requests (it is possible to cascade several XML-RPC handlers). The `XmlRpcClient` allows to invoke a method on a remote object. On the server side an `XmlRpcServer` is listening for XML-RPC requests on a specific port. `XmlRpcValue` is the base class of all values defined by the XML-RPC protocol (the scalar values `XmlRpcIntegerValue`, `XmlRpcBooleanvalue`, `XmlRpcDateValue` and `XmlRpcDoubleValue` and the composed values `XmlRpcStructureValue` and `XmlRpcArrayValue`, respectively).

xmlrpc.HttpWriter

java.lang.Object
  java.io.Writer
    java.io.OutputStreamWriter

public *HttpWriter*
extends OutputStreamWriter

Tool to write HTTP protocol headers.

Constructors

Description
HttpWriter(OutputStream outputStream) Constructs a new HttpWriter.

Methods

Returns	Description
public void	**writeError**(String message) Writes a HTTP error response to the OutputStream.
public void	**writeResponse**(String result) Writes a HTTP response to the OutputStream.

xmlrpc.XmlRpc

java.lang.Object
  org.xml.sax.HandlerBase

public abstract *XmlRpc*
extends HandlerBase

Base class for the parsing of an XML-RPC document.

Fields

Type	Description
public static final String	**COMMAND_OK**

Constructors

Description
XmlRpc()

Methods

Returns	Description
public void	**characters**(char[] data, int start, int length) Up-call by the SAX parser returning the character data (content) of the current element.
public void	**endElement**(String name) Up-call by the SAX parser indicating the end of an element ('endTag').
public void	**error**(SAXParseException e) Receives notification of a recoverable parser error.

Returns	Description
public void	**fatalError**(SAXParseException e) Reports a fatal XML parsing error.
public synchronized void	**parse**(InputStream inputStream) Parses the XML InputStream. For every element of the XML InputStream the methods startElement(), characters() and endElement() are invoked and the corresponding Java object ID is built. For each root level object (parameter) the method objectParsed() is executed.
public void	**startElement**(String name, AttributeList attributes) Up-call by the SAX parser indicating the start of new element ('startTag'). Each element has a type identified by name and may have a set of attribute specifications.
public String	**toString**() Returns a string representation of the XmlRpc class.
public void	**warning**(SAXParseException e) Receives notification of a parser warning.

xmlrpc.XmlRpcArrayValue

java.lang.Object
  xmlrpc.XmlRpcValue
    xmlrpc.XmlRpcComposedValue

---

public *XmlRpcArrayValue*
extends XmlRpcComposedValue

Representation of an 'array' value specified by the XML-RPC specification from 'Frontier.UserLand.Com'.

Constructors

Description
XmlRpcArrayValue() Constructs a new XmlRpcArray Value.

Methods

Returns	Description
public void	**addElement**(XmlRpcValue child) Adds an element to the XmlRpcArrayValue.
public Object	**getContent**() Returns the content (in Java representation) of the 'array' value.
public void	**setContent**(Object content) The content of a XmlRpcArrayValue is not set by setContent(). Use addElement() to add elements to the content.
public String	**toString**() Returns a string representation of the XmlRpcArrayValue's content.

xmlrpc.XmlRpcBooleanValue

java.lang.Object
  xmlrpc.XmlRpcValue

---

public *XmlRpcBooleanValue*
extends XmlRpcValue

Representation of a 'boolean' value specified by the XML-RPC specification from 'Frontier.UserLand.Com'.

Constructors

Description
XmlRpcBooleanValue() Constructs a new XmlRpcBoolean value.
XmlRpcBooleanValue(Object content) Constructs a new XmlRpcBoolean value.

Methods

Returns	Description
public void	**setContent**(Object content) Constructs a new XmlRpcBoolean value.
public String	**toString**() Returns a string representation of the XmlRpcBooleanValue's content.

xmlrpc.XmlRpcClient

java.lang.Object

public *XmlRpcClient*
extends Object
implements XmlRpcMethodHandler

The XmlRpcClient allows to invoke a method defined by an XML-RPC 'methodName'-tag (the handler name followed by a method name separated by a dot) and the corresponding parameters.

Constructors

Description
XmlRpcClient(String host, int port) Constructs a new XmlRpcClient.

Methods

Returns	Description
public Object	**invokeMethod**(String methodName, Vector parameters) Generates an XML-RPC request and sends it to the server. After parsing the result the corresponding Java object is returned.

xmlrpc.XmlRpcComposedValue

java.lang.Object
xmlrpc.XmlRpcValue

public abstract *XmlRpcComposedValue*
extends XmlRpcValue

Base class of the composed XML-RPC values 'array' and 'struct'.

Constructors

Description
XmlRpcComposedValue()

Methods

Returns	Description
public abstract void	**addElement**(XmlRpcValue child) Adds an element to the composed value.

xmlrpc.XmlRpcDateValue

java.lang.Object
  xmlrpc.XmlRpcValue

---

public *XmlRpcDateValue*
extends XmlRpcValue

Representation of a 'dateTime.iso8601' value specified by the XML-RPC specification from 'Frontier.UserLand.Com'.

Constructors

Description
XmlRpcDateValue() Constructs a new XmlRpcDateValue.
XmlRpcDateValue(Object content) Constructs a new XmlRpcDateValue.

Methods

Returns	Description
public static Date	**decode**(String date) Decodes an XML-RPC representation of a date to a Date object.
public static String	**encode**(Date date) Encodes a Date object to an XML-RPC representation ('dateTime.iso8601').
public void	**setContent**(Object content) Sets the content of the XmlRpcDateValue.
public String	**toString**() Returns a string representation of the XmlRpcDateValue's content.

xmlrpc.XmlRpcDoubleValue

java.lang.Object
  xmlrpc.XmlRpcValue

---

public *XmlRpcDoubleValue*
extends XmlRpcValue

Representation of a 'double' value specified by the XML-RPC specification from 'Frontier.UserLand.Com'.

Constructors

Description
XmlRpcDoubleValue() Constructs a new XmlRpcDoubleValue.
XmlRpcDoubleValue(Object content) Constructs a new XmlRpcDoubleValue.

Methods

Returns	Description
public void	**setContent**(Object content) Sets the content of the XmlRpcDoubleValue.
public String	**toString**() Returns a string representation of the XmlRpcDoubleValue's content.

xmlrpc.XmlRpcException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception

public *XmlRpcException*
extends Exception

Used for exceptions occured executed the XML-RPC protocol.

Fields

Type	Description
public static final int	**DEFAULT_FAULT_CODE**

Constructors

Description
XmlRpcException(String message) Constructs a new XmlRpcException containing an error message.
XmlRpcException(int faultCode, String message) Constructs a new XmlRpcException containing an error message and a fault code.

Methods

Returns	Description
public int	**getFaultCode**() Returns the fault code of the exception.

xmlrpc.XmlRpcIntegerValue

java.lang.Object
  xmlrpc.XmlRpcValue

public *XmlRpcIntegerValue*
extends XmlRpcValue

Representation of an 'int' value specified by the XML-RPC specification from 'Frontier.UserLand.Com'.

Constructors

Description
XmlRpcIntegerValue() Constructs a new XmlRpcIntegerValue.
XmlRpcIntegerValue(Object content) Constructs a new XmlRpcDoubleValue.

Methods

Returns	Description
public void	**setContent**(Object content) Sets the content of the XmlRpcIntegerValue.
public String	**toString**() Returns a string representation of the XmlRpcIntegerValue's content.

xmlrpc.XmlRpcMethodHandler

public abstract interface *XmlRpcMethodHandler*

XmlRpcMethodHandler has to be implemented by classes acting as handlers for a remote procedure call.

Methods

Returns	Description
public Object	**invokeMethod**(String method, Vector parameters) Returns the result or throws an exception if something went wrong.

xmlrpc.XmlRpcServer

java.lang.Object

public *XmlRpcServer*
extends Object
implements Runnable, XmlRpcMethodHandler

A Server listening for XML-RPC requests on a specific port. Different handlers implementing the XmlRpcMethodHandler interface can be added to the server to process remote procedure calls.

Constructors

Description
XmlRpcServer(int port) Constructs a new XmlRpcServer.

Methods

Returns	Description
public void	**addMethodHandler**(String handlerName, XmlRpcMethodHandler methodHandler) Registers a method handler on the server. The different methods of the handler can be called over XML-RPC as 'handlerName.methodName'.
public XmlRpcMethodHandler	**getMethodHandler**(String handlerName) Returns the method handler for the corresponding handler name.
public Object	**invokeMethod**(String methodName, Vector parameters) Invokes the method of the corresponding handler.
public void	**removeMethodHandler**(String handlerName) Removes a method handler from the server.
public void	**run**() Starts the main thread listening for client requests.
public void	**stop**() Stops the server and closes the socket the server is listening at.

xmlrpc.XmlRpcServerRequest

java.lang.Object
  org.xml.sax.HandlerBase
    xmlrpc.XmlRpc

---

public *XmlRpcServerRequest*
extends XmlRpc

Used to send an XML-RPC request to the server.

Constructors

Description
XmlRpcServerRequest(URL server) Constructs a new Server request object.

Methods

Returns	Description
public Object	**invokeMethod**(String method, Vector parameters) Invokes a method on the server.
public void	**startElement**(String name, AttributeList attributes) Overrides the equivalent method in XmlRpc to handle also faulty responses.

xmlrpc.XmlRpcStringValue

java.lang.Object
  xmlrpc.XmlRpcValue

---

public *XmlRpcStringValue*
extends XmlRpcValue

Representation of an 'string'-value specified by the XML-RPC specification from 'Frontier.UserLand.Com'.

Constructors

Description
XmlRpcStringValue() Constructs a new XmlRpcStringValue.
XmlRpcStringValue(Object content) Constructs a new XmlRpcStringValue.

Methods

Returns	Description
public void	**setContent**(Object content) Sets the content of the XmlRpcStringValue.
public String	**toString**() Returns a string representation of the XmlRpcStringValue's content.

xmlrpc.XmlRpcStructureValue

java.lang.Object
  xmlrpc.XmlRpcValue
    xmlrpc.XmlRpcComposedValue

---

public *XmlRpcStructureValue*
extends XmlRpcComposedValue

Representation of a 'struct' value specified by the XML-RPC specification from 'Frontier.UserLand.Com'.

Constructors

Description
XmlRpcStructureValue() Constructs a new XmlRpcStructureVallue.

Methods

Returns	Description
public void	**addElement**(XmlRpcValue child) Adds an element to the XmlRpcStructureValue. Uses the name set by the last call of setName().
public Object	**getContent**() Returns the content (in Java representation) of the 'struct' value.
public void	**setContent**(Object content) The content of a XmlRpcStructureValue is not set by setContent(). Use addElement() to add elements to the content.
public void	**setName**(String name) Sets the name of the structure.
public String	**toString**() Returns a string representation of the XmlRpcStructureValue's content.

xmlrpc.XmlRpcValue

java.lang.Object

---

public *XmlRpcValue*
extends Object

Base class for all the 'value' types used by the XML-RPC procedure call protocol defined in the XML-RPC specification from 'Frontier.UserLand.Com'.

Constructors

Description
XmlRpcValue() Constructs a new value.
XmlRpcValue(Object content) Constructs a new value.

Methods

Returns	Description
public Object	**getContent**() Returns the content of the value.
public void	**setContent**(Object content) Sets the content of the value.
public String	**toString**() Returns a string representation of the XmlRpcValue's content.

xmlrpc.XmlRpcWriter

java.lang.Object
  java.io.Writer
    java.io.OutputStreamWriter
      xmlrpc.XmlWriter

public *XmlRpcWriter*
extends XmlWriter

Tool to write XML-RPC data to an OutputStream.

Fields

Type	Description
public static final String	**ARRAY_TAG**
public static final String	**BOOLEAN_TAG**
public static final String	**DATA_TAG**
public static final String	**DATE_TAG**
public static final String	**DOUBLE_TAG**
public static final String	**FAULT_CODE**
public static final String	**FAULT_STRING**
public static final String	**FAULT_TAG**
public static final String	**INTEGER_TAG_1**
public static final String	**INTEGER_TAG_2**
public static final String	**MEMBER_TAG**
public static final String	**METHOD_CALL_TAG**
public static final String	**METHOD_NAME_TAG**
public static final String	**METHOD_RESPONSE_TAG**
public static final String	**NAME_TAG**
public static final String	**PARAM_TAG**
public static final String	**PARAMS_TAG**
public static final String	**STRING_TAG**
public static final String	**STRUCTURE_TAG**
public static final String	**VALUE_TAG**

Constructors

Description
XmlRpcWriter(OutputStream outputStream) Constructs a new XmlRpcWriter.

Methods

Returns	Description
public void	**writeError**(int faultCode, String fault) Writes an XML-RPC error response to the OutputStream.
public void	**writeObject**(Object object) Writes the XML-RPC representation of a Java object to the OutputStream.

Returns	Description
public void	**writeRequest**(String method, Vector parameters) Writes an XML-RPC request to the OutputStream.
public void	**writeResponse**(Object param) Writes an XML-RPC response to the OutputStream.

xmlrpc.XmlWriter

java.lang.Object
  java.io.Writer
    java.io.OutputStreamWriter

---

public *XmlWriter*
extends OutputStreamWriter

Tool to write XML data to an OutputStream.

Constructors

Description
XmlWriter(OutputStream outputStream) Constructs a new XmlWriter.

Methods

Returns	Description
public void	**write**(String data) Writes an XML string to the output stream. There will be a special handling of some characters.
public void	**writeComment**(String comment) Writes an XML comment to the output stream.
public void	**writeEndTag**(String tag) Writes an XML end tag to the output stream.
public void	**writeStartTag**(String tag) Writes an XML start tag to the output stream.

Bibliography

[1] Apple Computer, Inc. *Macintosh Human Interface Guidelines*. Addison-Wesley, 1992.

[2] A. Fischer and H. Marais. *The Oberon Companion. A Guide to Using and Programming Oberon System 3*. VDF, 1998.

[3] David Flanagan. *Java in a Nutshell: A Desktop Quick Reference*. A Nutshell handbook. O'Reilly & Associates, Inc., second edition, 1997.

[4] Michael Foley and Mark McCulley. *JFC Unleashed: The Comprehensive Solution*. Sams, 1999.

[5] Susan Fowler and Victor Stanwick. *The GUI Style Guide*. Academic Press, 1994.

[6] Wilbert O. Galitz. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. John Wiley & Sons, 1996.

[7] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[8] David M. Geary. *Graphic Java 2: Mastering the JFC, Volume 2: Swing*. Prentice-Hall, 1999.

[9] Cay S. Horstmann and Gary Cornell. *Core Java 1.1: Volume 1: Fundamentals*. Prentice-Hall, third edition, 1997.

[10] Cay S. Horstmann and Gary Cornell. *Core Java 1.1: Volume 2: Advanced Features*. Prentice-Hall, first edition, 1998.

[11] Virginia Howlett. *Visual Interface Design for Windows: Effective User Interfaces for Windows 95, Windows NT, and Windows 3.1*. John Wiley & Sons, 1996.

[12] Sun Microsystems. *Java Look and Feel Design Guidelines*. Addison-Wesley, 1999.

[13] Moira C. Norrie. An extended entity-relationship approach to data management in object-oriented systems. In *Proceedings of the 12th International Conference on Entity-Relationship Approach*, 1993.

[14] Moira C. Norrie, Alain Wuergler, and M. Wunderli. A model for classification structures with evolution control. In *Proceedings of the 15th International Conference on Conceptual Modelling*, 1996.

[15] Gabrio Rivera and Moira C. Norrie. OMX-FS: A File System Architecture based on the OM Object Data Model. Technical report, Institute for Information Systems, ETHZ, 1999.

[16] Jon Siegel. *CORBA: Fundamentals and Programming*. John Wiley & Sons, 1996.

[17] Susan Weinschenk, Pamela Jamar, and Sarah C. Yeo. *GUI Design Essentials*. John Wiley & Sons, 1997.

[18] N. Wirth and J. Gutknecht. *Project Oberon*. Addison-Wesley, 1992.