The Browser: An Exploration Tool for ObjTalk Inheritance Structures

by

Christian Rathke

CU-CS-331-86 May, 1986

Department of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309-0430
USA

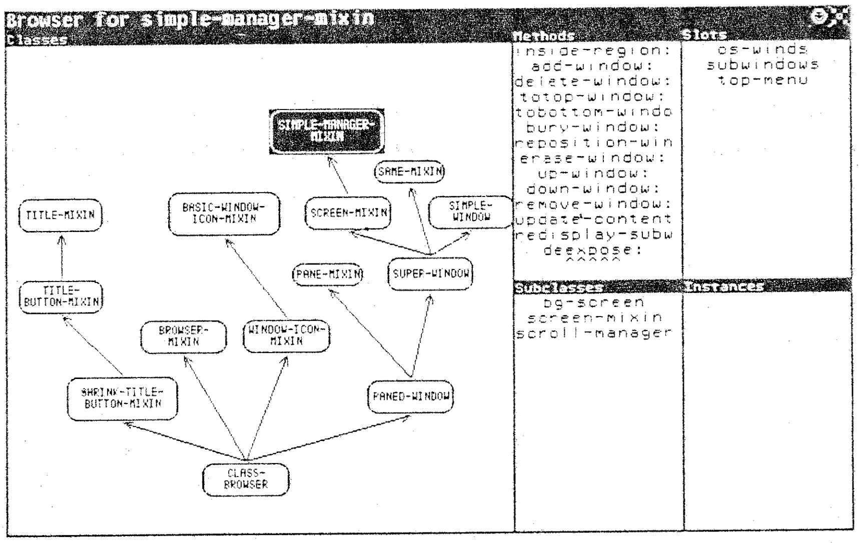
This work was supported by grants from the U.S. Department of Energy numbered DE-FG02-84ER13283.

The Browser

An Exploration Tool for ObjTalk Inheritance Structures

Christian Rathke

The Browser is part of the programming environment of ObjTalk. It serves as an exploration tool for existing ObjTalk systems. Inheritance hierarchies of ObjTalk classes can be shown using graphical representation^c^. The definition of slots and methods can be inspected (Fig. 1).



```
Methods              Slots
inside-region:       os-winds
add-window:          subwindows
delete-window:       top-menu
totop-window:
tobottom-windo
bury-window:
reposition-win
erase-window:
up-window:
down-window:
remove-window:
update-content
redisplay-subw
deexpose:

Subclasses           Instances
bg-screen
screen-mixin
scroll-manager
```

Figure 1: The Browser with windows to show the class hierarchy, methods, slots, instances and subclasses of a selected class

1. A Programming Environment for ObjTalk

The Browser is an integrated part of the user's programming environment. The programming environment is built upon Wlisp, a user interface management system. Wlisp provides a multiple window environment and defines a variety of screen objects. Screen objects have common properties:

- They can be created dynamically.
- They are able to change their position and size.
- They can overlap.
- They can be stored as pixel images inside the terminal's memory, etc.

The Wlisp system itself is implemented in ObjTalk, an objectoriented extension to LISP. Objects communicate by sending messages to each other. They *behave* according to the *interpretation of messages* and their *internal state*. Reactions on messages are defined in *methods*. The internal state is defined in *slots*.

Objects are organized in *classes*. Objects of the same class are called the *instances* of that class. They have the same methods and slots and show the same type of behavior. Classes are arranged in hierarchies. They share the properties, i.e. methods and slots, of their superordinated classes. In this way properties are *inherited* among classes. All classes together form an *inheritance hierarchy* with a special class at the root called OBJECT.

2. The User Interface to the Browser

The Browser window is divided into several subwindows some of which overlap (Fig. 2-1).

The Wlisp system offers the possibility to place windows within windows. The layout of the Browser's subwindows is specified by using a formal layout description language.

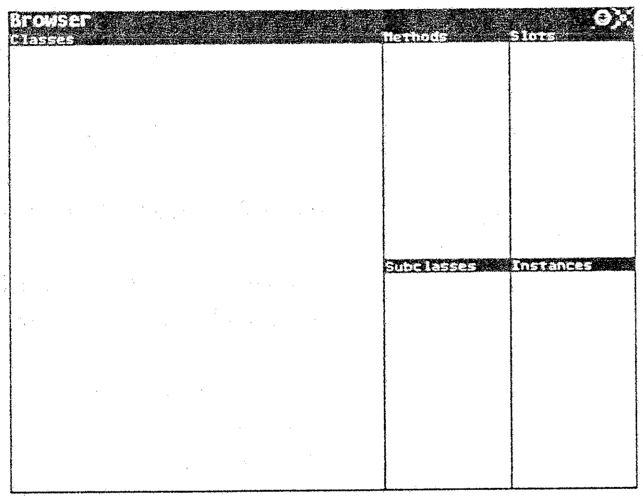


Figure 2-1: The initial state of the Browser

*Inside the **Classes** window a special menu can be brought to top by pressing the right mouse button. The name of an ObjTalk class can be entered or the selection of a window's class can be initiated (Fig. 2-2).*

Mouse buttons have a predefined meaning within windows. Usually by pressing the left mouse button a window will be *selected*. To be selected may have different semantics depending on the class of the window. Normally the window will be brought to top. In menus menu selection is initiated. An icon would start the program that is associated with it.

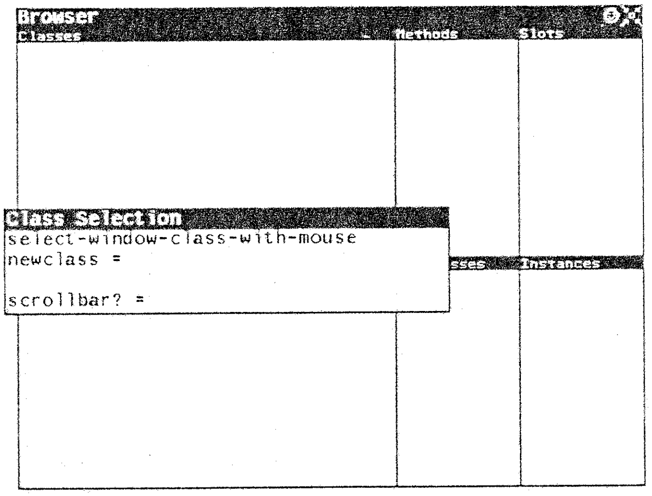


Figure 2-2: Selecting a class to be inspected

Pressing the right mouse button causes a menu to appear on the screen at mouse position. It contains operations for the window.

The consistency of interaction is an important property of the user interface. Consistency, although not enforced, is simplified by the system's architecture.

Within the ***Classes*** window the menu contains two operation with which the user can specify a certain class:

1. The name of the class may be entered from the keyboard. The user has to know its name.
2. If the user wants to inspect a window class, one of whose instances is visible on the screen, s/he can identify the window by pointing to it with the mouse.

Since the Browser is an exploration tool for unknown systems the knowledge of class names cannot be expected.[1] Not all of the classes can be easily referred to by pointing at some object on the screen. In order to identify them additional facilities have to be provided.[2]

*The selected class is visualized by an icon that can be moved inside the **Classes** window (Fig. 2-3).*

In general icons are complex information structures with components for visualization and operation. Within the context of the Browser they are associated with classes.

In the ***Classes*** window icons contain graphic and text. The text is the name of a class. The

[1] An exeption would be a user's behavior in which s/he starts at the root class OBJECT and follows the subclasses links.

[2] e.g. Rabbit like systems

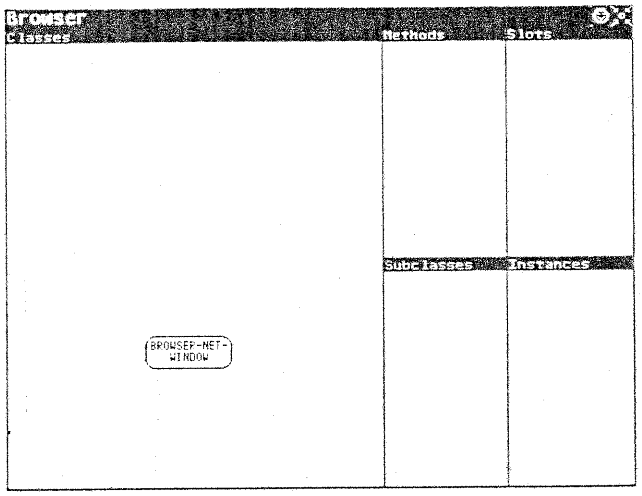


Figure 2-3: A visualized class

graphic component forms the icon's border and is one of three predefined pixel images which is selected according to the length of the text.

Icons for superclasses may be created by pressing the right *mouse button on one of the class icons (Fig. 2-4).*

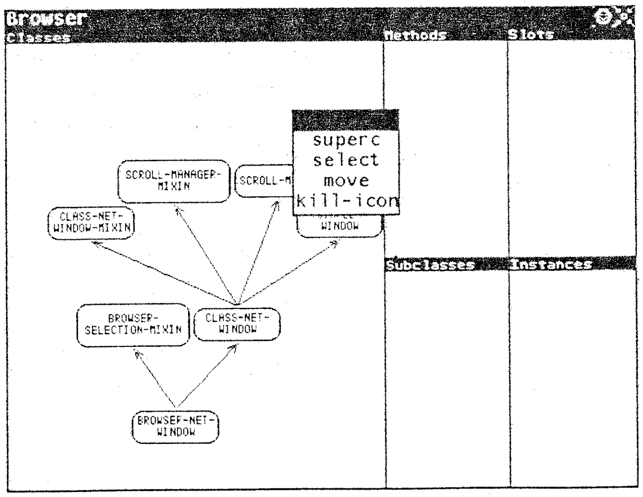


Figure 2-4: Creating icons for the superclasses

Icons are screen objects by themselves. They have similarities with other screen objects like windows or menus. Pressing the right mouse button causes a menu with several useful operations to appear. By selecting the ***superc*** entry the user creates icons that visualize the superclasses.

Selecting an icon with the left mouse button causes the Browser's menus to be filled with slots, methods, subclasses and instances of the selected class (Fig. 2-5).

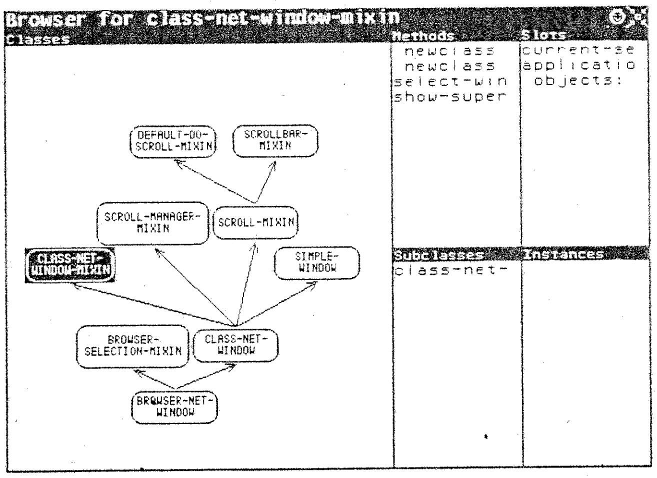


Figure 2-5: Selection of a class icon

Selection in the ***Subclasses*** *menu causes the appearance of the corresponding class icon in the* ***Classes*** *window.*

This is the fourth alternative (besides typing in the class name, selecting one of the window's instances with the mouse and selecting the superc entry in the icon's menu) to identify a class to be shown. In this way the class hierarchy can be explored "downward" starting at some known class.

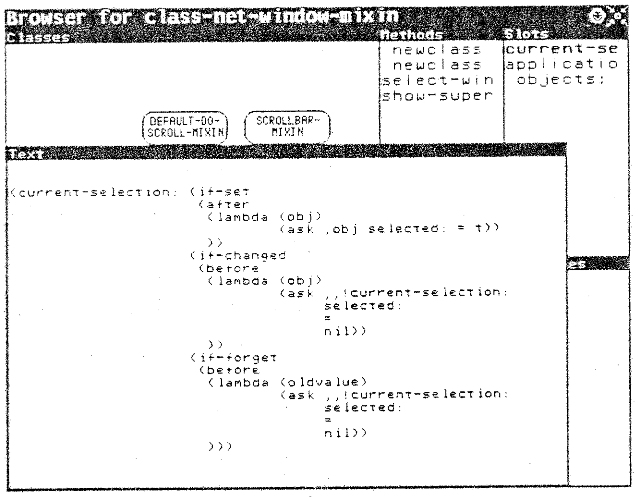
Selection within one of the other menus causes the textual representation of slots, methods and instances to appear in the Browser's ***Text*** *window (Fig. 2-6).*

The Browser's menus are specialized scroll menus. Their entries consist of methods, slots and object names. The number of items is not restricted. Because their size is fixed only some of the names are shown. Others can be accessed by scrolling.

After menu selection the selected entry is shown in boldface.

The representation of a method, slot or instance as LISP s-expressions is shown inside a window that covers the ***Classes*** window for that purpose.

The properties of the user interface for the Browser depend in some part of preexisting systems and



```
(current-selection: (if-set
                      (after
                       (lambda (obj)
                               (ask ,obj selected: = t))
                       ))
                     (if-changed
                      (before
                       (lambda (obj)
                               (ask ,,!current-selection:
                                    selected:
                                    =
                                    nil))
                       ))
                     (if-forget
                      (before
                       (lambda (oldvalue)
                               (ask ,,!current-selection:
                                    selected:
                                    =
                                    nil))
                       )))
```

Figure 2-6: The description of a selected slot is shown in the *text* window

tools:

1. the window system (implemented by F.Fabian, 1984)
2. the menu system (F.Fabian and C.Rathke, 1983)
3. the icon system (M.Herczeg, 1984)
4. the net editor (W.F.Riekert, 1985)
5. the window panes (F.Fabian and C.Rathke, 1984)
6. application sheets (W.F.Riekert, 1985)

All of the systems are implemented in ObjTalk and are part of the Wlisp environment. The Browser adds the functionality of showing classes in an inheritance network and being able to explore an ObjTalk based system in detail.

3. Conclusions

The Browser is an exploration tool for unknown ObjTalk systems. The user must have some knowledge about ObjTalk in order to make use of the information provided by the Browser. Beeing part of the ObjTalk programming environment the Browser helps an expert programmer to identify components of any ObjTalk system.

For its user interface the Browser uses the window-, menu- and icon-subsystem of the Wlisp environment in several ways:

1. by inheriting methods and slots from existing classes,
2. by instantiation of existing classes and

3. by filling slots of existing objects with specific values.

This implies consistency to other systems in terms of

- object properties and
- object behavior.

As software engineering becomes an activity in which new systems are not built totally from scratch but use components of already existing ones, the process of acquiring knowledge about those components becomes important. The Browser is a tool for this purpose. It helps in exploring hierarchical systems by providing visualizations of structure and content of the system's objects.