10. BROWSERS

As described in Chapter 1, Introduction, one of the key components in the LOOPS system is inheritance, in which structures have well-defined relationships to other structures. Class inheritance is a typical example of this relationship.

Since inheritance can be described by a two-dimensional graph, it is natural to create a user interface for LOOPS built on the Lisp Library Module, Grapher. This user interface is called a browser. Browsers are tools to assist in the development cycle of a product or vehicles for building user interfaces within a final product.

Much development time is spent building, examining, and modifying the relationships between classes. These tasks include specifying the contents of various classes: class variables, instance variables, properties, and methods. The location of the class within the inheritance structure must also be determined. After a number of classes have been built, the relationships between the classes may need to be reviewed. Often, the initial design is flawed and requires the following changes:

- Moving parts of one class to another class.
- Adding, substracting, or changing data or functionality within classes.
- Adding new classes, or merging different classes.

Browsers are the facility within LOOPS which support these types of operations. This chapter describes how to use browsers both interactively with the mouse, and programmatically.

Browsers are most commonly used on the classes defined for an application. Many of the examples here browse objects which LOOPS uses internally; the functionality is exactly the same.

0.1 Types of Built-in Browsers

A number of different types of browsers are already built into LOOPS:

- Lattice browsers
- Class brwsers
- File browsers
- Supers browsers
- Metaclass browsers

This section describes these browsers in detail.

10.1.1 Lattice Browsers

The most general class is called **LatticeBrowser**. Figure 10-1 shows a class inheritance lattice with the subclasses of **LatticeBrowser.**

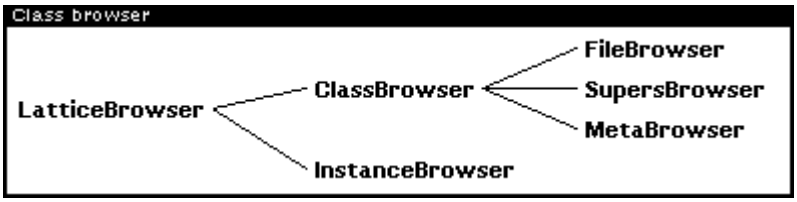


Figure 10-1. Sample Lattice Browser

10.1.2 Class Browsers

A class browser shows the linkages between a class or classes and their subclasses. Super classes are shown on the left (or top) side of the browser window. Subclasses of these are connected by links moving to the right (or down). An example of a class browser is shown in the previous section. The class **LatticeBrowser** is the root object of this example. Subclasses of **LatticeBrowser** are **ClassBrowser** and **InstanceBrowser**. Subclasses of **ClassBrowser** are **FileBrowser**, **SupersBrowser**, and **MetaBrowser**.

10.1.3 File Browsers

A file browser is a class browser containing all classes defined within a file. Additionally, file browsers contain a menu interface to common operations on files.

10.1.4 Supers Browers

A supers browser is an inverted class browser. A class browser is built by following subclass links from a class. A supers browser is built by following superclass links from a class. An example of a supers browser is shown in Figure 10-2.

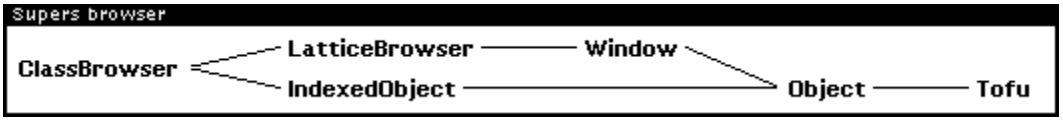


Figure 10-2. Sample Supers Browser

10.1.5 Metaclass Browsers

A metaclass browser is like a supers browsers, but is built by following metaclass links. Figure 10-3 shows two root classes: **ActiveValue** and **ClassBrowser.**

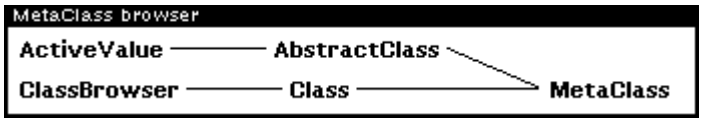


Figure 10-3. Sample Metaclass Browser

10.1.6 Instance Browsers

An instance browser shows the relationships between instances. These relationships may be more dynamic than the inheritance relationships shown by the class browsers. Typically, the relationships are not defined until runtime, and may be changed often. By specializing the instance browser, you can show several relationships between a fixed set of objects.

10.2 OPENING BROWSERS

10.2 Opening Browsers

A browser can be opened in several ways:

- Selecting a menu option from the Background Menu.
- Selecting a menu option from the LOOPS icon.
- Sending a **Browse** message to an instance of a browser.
- Calling either of the functions **Browse** or **FileBrowse**.

10.2.1 Using Menu Options to Open Browsers

Since browsers are an important part of LOOPS, you can use menus in several ways to create standard browsers. Once opened, via menu or program, any browser can be operated from both the appropriate menus and programmatic commands.

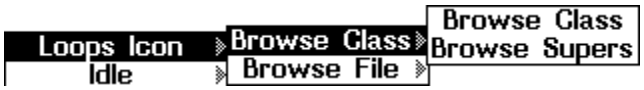
10.2.1.1 Overview of Background Menu and LOOPS Icon

When LOOPS is loaded, the option **Loops Icon** is added to the background menu, as shown in this window:

The **Loops Icon** option has two suboptions:

- **Browse Class**
- **Browse File**

These suboptions are shown in the following windows:



Selecting **Loops Icon** puts a LOOPS icon on the screen; you are prompted to place the icon after it is created. The other commands are discussed in Section 10.2.1.2, "Command Summary."

The LOOPS icon, which appears in Figure 10-4, is a prototype instance of the class **LoopsIcon**. It is provided to give you another convenient menu interface to typical programming operations.

Figure 10-4. LOOPS Icon

Pressing the left button while the mouse is on the icon causes the following menu to appear with options appropriate for class browsers:

Pressing the middle button while the mouse is on the icon causes the following menu to appear with choices appropriate for file browsers:

Pressing the right button while the mouse is on the icon causes the following menu to appear with two options for operations on the icon itself:

Close removes the icon from the screen. It can be restored at any time from the background menu. **Move** lets the icon be moved to another location on the screen, just as any icon is moved in Lisp.

10.2.1.2 Command Summary

The background menu and the LOOPS icon provide the same functionalities. This section describes the commands available.

Browse Class, Browse Supers

Selecting either of these options causes the following prompt to appear in the prompt window.

```
Please tell me the name of the root object >
```

Enter the name of a class without using the "$" notation. The system builds the appropriate type of browser and prompts you to move the window containing the browser.

Browse File

Selecting this option causes the following menu to appear:

The top option on this menu is ***newFile*** which has three suboptions; the remaining options are the names of the files that are on **FILELST** (Lisp remembers what files are loaded and how they were loaded; **FILELST** files were loaded normally. See the *Interlisp-D Reference Manual* for a full explanation). Selecting one of the filenames will open a file browser on that file.

newFile Prompts you in the prompt window with the following prompt :

```
Please type in file name: >.
```

Enter the name of a file to create. The system checks to determine if a filecoms exists for that file name. If one exists, the system asks for confirmation before destroying the value of that filecoms and opening up an empty browser window. If no filecoms exists for that filename, an empty file browser window is opened.

loadFile Prompts you with:

```
Please type in file name to load: >
```

The system loads that file and opens a browser on it.

hiddenFile Causes a menu to appear with files that have been loaded but not **SYSLOAD**ed and are not on **FILELST**; that is, the files are on **LOADEDFILELST**, but not on **FILELST**. The LOOPS files, for example, the .LCOMs that add LOOPS to Lisp, are on this list.

Edit Filecoms

Selecting this option brings up the same menu as the option **Browse File**. Instead of opening a browser on the file, a display editor window is opened on the filecoms of that file. If ***newFile*** is selected, you are prompted to enter a file name and an SEdit window is opened with a template containing the File Manager commands **CLASSES, METHODS, FNS, VARS**, and **INSTANCES**.

CleanUp File

Selecting this option first calls **FILES?** and then builds a menu of files in **FILELST** that have changed. From this menu, select a file to be cleaned up; this calls **CLEANUP**.

10.2.2 Using Commands to Open Browsers

You can use the following methods and functions for opening browsers programmatically and from the Lisp Executive window.

Name	Type	Description
Browse	Method	Opens a browser showing the relationships between classes.

BrowseFile	Method	Opens a browser showing the relationships between classes on a file.
Browse	Function	Provides a short way to create a class browser.
FileBrowse	Function	Provides a short way to create a file browser.

(← *self* **Browse** *browseList windowOrTitle goodList position*) [Method of LatticeBrowser]

Purpose/Behavior: Opens a browser showing inheritance relationships between classes.

Arguments:

self An instance of **ClassBrowser**, **MetaBrowser**, or **SupersBrowser.**

browseList A LOOPS class name, a LOOPS pointer to a class name, or a list of those.

windowOrTitle A title to appear on the browser or a window to use (but which will be reshaped to fit the browser.) Title defaults to "Class browser."

goodList A **goodList** other than the *browselist*. (See Section 10.5.1, "Instance Variables of Class LatticeBrowser," for more information on **goodList**.)

position Lower left corner of browser. If NIL, position the window with the mouse.

Returns: Pointer to the browser object.

Examples: The following command opens a class browser on **Window .**

```
(←New ($ ClassBrowser) Browse 'Window)
```

The following command opens a supers browser on **InstanceBrowser** and **ClassBrowser**.

```
(←New ($ SupersBrowser) Browse (LIST 'InstanceBrowser ($ ClassBrowser)))
```

(← *self* **BrowseFile** *fileName*) [Method of FileBrowser]

Purpose: Opens a browser showing relationships between classes on a file.

Behavior: Classes defined within *fileName* are displayed within the browser. If *fileName* is NIL, a menu of files on **FILELST** opens. The selected file has a file browser opened on it.

Arguments:

self An instance of the class **FileBrowser**

fileName File to browse; should not be a list.

Returns: *self*

Categories: FileBrowser

Example: The following command opens a file browser on the file **LoopsWindow**.

```
(←New ($ FileBrowser) BrowseFile 'LoopsWindow)
```

(**Browse** *classes title goodClasses position*) [Function]

Purpose: Provides a short way to create a class browser.

Behavior: Sends a **Browse** message to a new instance of a **ClassBrowser** passing *classes*, *title*, *goodClasses*, and *position* as arguments. If *goodClasses* is T, it is rebound to the value of *classes* before the message is sent.

Arguments:

classes A LOOPS class name, a LOOPS pointer to a class name, or a list of those.

title A title to appear on the browser. Title defaults to "Class browser."

goodClassses A **goodList** other than *classes*. (See Section 10.5.1, "Instance Variables of Class LatticeBrowser," for more information on **goodList**.)

position Lower left corner of browser. If NIL, position the window with the mouse.

Returns: A new instance of **ClassBrowser**.

Example: The following command creates a class browser on the class **ActiveValue** and all its subclasses.

```
11← (Browse 'ActiveValue)
```

(**FileBrowse** *filename*) [Function]

Purpose: Provides a short way to create a file browser.

Behavior: Sends a **BrowseFile** message to a new instance of a **FileBrowser** passing *filename* as the argument.

Arguments: *filename* File to browse.

Returns: New instance of **FileBrowser**.

Example: The following command creates a file browser on the file **LoopsWindow**.

```
12← (FileBrowse 'LoopsWindow)
```


10.3 USING CLASS BROWSERS, META BROWSERS, AND SUPERS BROWSERS

10.3 Using Class Browsers, Meta Browsers, and Supers Browsers

Instances of **ClassBrowser, SupersBrowser, MetaBrowser** all have the same menu interface. This section shows examples of the various menus followed by descriptions of the actions performed after selecting particular options.

Three pop-up menus are associated with browsers:

- One menu appears by positioning the mouse on the title bar of the browser window and pressing either the left or the middle mouse button. This menu contains options that control the appearance of the browser.
- A second menu appears by positioning the mouse on one of the nodes in a browser and pressing the left mouse button. This menu contains informational options.

- A third menu appears by positioning the mouse on one of the nodes in a browser and pressing the middle mouse button. This menu contains editing options.

These menus differ depending on the browser type. The following sections describe the menus associated with class browsers, supers browsers, and metaclass (or, more simple, meta) browsers. Sections then describe the additional functionality associated with file browser menus.

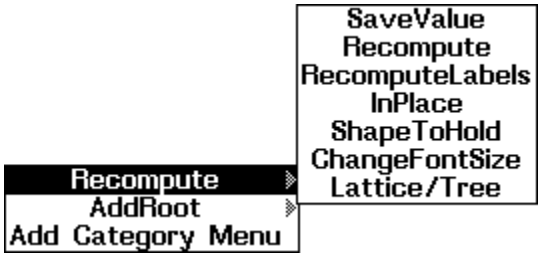
10.3.1 Selecting Options in the Title Bar Menu

The following menu appears when you position the mouse on the title bar of the browser menu and press either the left or the middle mouse button:

This section describes each menu option.

10.3.1.1 Recompute and its Suboptions

Selecting the **Recompute** option and dragging the mouse to the right causes the following submenu to appear:



Most of the **Recompute** suboptions change the appearance of a browser but not its contents. For example, **SaveValue** provides a pointer to the browser without changing anything in it.

SaveValue The browser instance is stored in the instance variable **savedValue** of the prototype instance of **LoopsIcon** and in the value of **IT** (see the *Interlisp-D Reference Manual*). This value is returned from the function call **SavedValue**.

Recompute Recomputes the entire browser structure from the starting objects. It does not recompute the labels for each item if those labels have been cached in the property **objectLabels** of the instance variable menus.

RecomputeLabels Recomputes the entire browser structure from the starting objects and recomputes the labels for each item.

InPlace Recomputes the browser without affecting the scrolled location of the lattice within the window. This may be necessary for a browser containing a large lattice structure.

ShapeToHold Makes the window for the browser just large enough to hold all of the nodes in the browser, up to a maximum size. Browser windows may also be changed interactively or programmatically with **SHAPEW**.

ChangeFontSize Causes a menu to appear containing 8, 10, 12, and 16. Selecting one changes the font size used to display the nodes to that value. The font family is

```
(@ self browseFont:,FontFamily)
```

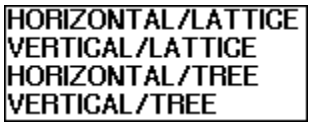
The font face is

```
(@ self browseFont:,FontFace)
```

Note: An alternative way to change the font of a browser is to enter:

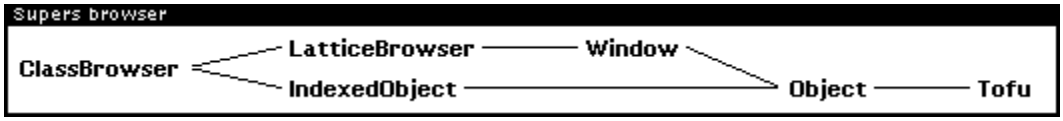
```
[PROGN (←@ ($ InstanceOfBrowser) browseFont
(FONTCREATE .....))(← ($ InstanceOfBrowser)
RecomputeLabels)]
```

Lattice/Tree Causes the following menu to appear:

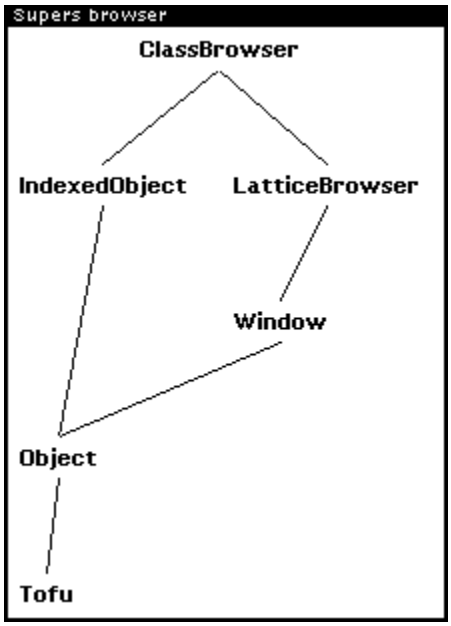


Using the example of a supers browser for the class **ClassBrowser**, this browser is drawn for each of the formatting options. A tree does not show branches recombining; a lattice does. A boxed node in a tree indicates the node shows up in more than one location in a tree. When a browser is constructed by the system the default formatting style is HORIZONTAL/LATTICE.

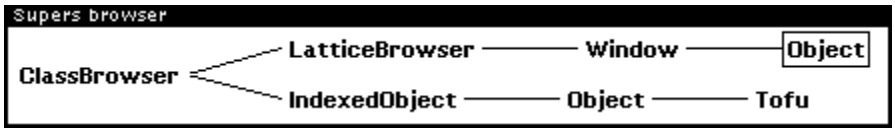
- HORIZONTAL/LATTICE



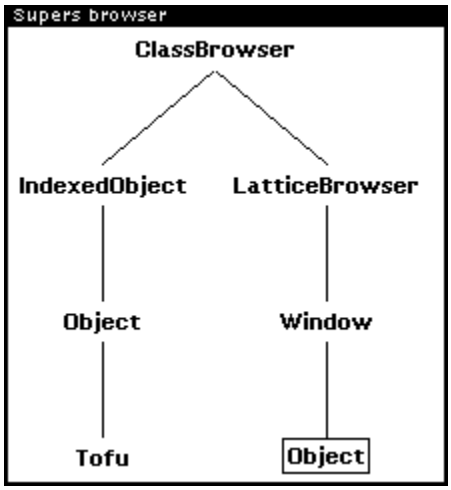
- VERTICAL/LATTICE



- HORIZONTAL/TREE



- VERTICAL/TREE



10.3.1.2 AddRoot and its Suboptions

The **AddRoot** options add items or subtrees to the browser.

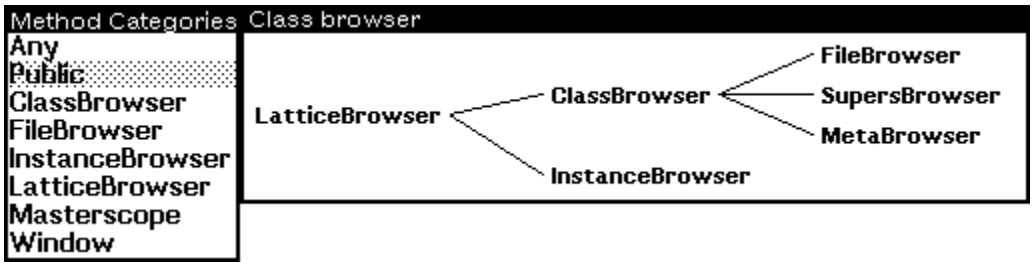
Selecting the **AddRoot** option and dragging the mouse to the right causes the following submenu to appear:

AddRoot A prompt appears in an attached window to enter the name of a class to be added to the browser. If the entered item is not an object, a message that nothing was added to the browser is printed. If the entered item is already in the browser, nothing occurs. If the entered name does not correspond to a class, nothing occurs.

RemoveFromBadList Objects within a browser can be put on the instance variable **badList**. This can be done by positioning the mouse on the node in a browser, pressing the left mouse button, and selecting an option from the menu that appears. Items on the **badList** are not displayed in the browser. If you select the option **RemoveFromBadList,** a menu appears showing any objects on the **badList**. Selecting one of those objects removes it from the **badList** and causes it to be redisplayed in the browser.

10.3.1.3 Add Category Menu

The system searches all methods in all classes shown in the browser and computes the categories for these. These categories are made into a sorted menu with the categories **Any** and **Public** included at the top. This menu is attached to the left side of the browser. Selecting options in this menu acts as a toggle, either highlighting them or returning them to their normal display.

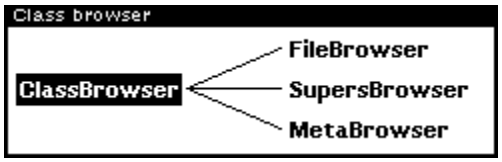


Selected options are stored on the browser instance variable **viewingCategories.** Options on this menu interact with the browser interface for editing methods as described in Section 10.3.2, "Selecting Options in the Left Menu."

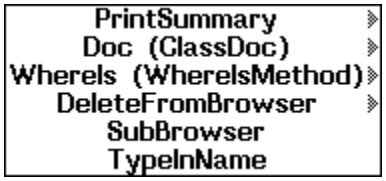
Note: Very often when using a browser, you ask to see what items a class inherits from classes above it in the inheritance lattice. To keep this inherited information more manageable, information inherited from the classes **Tofu**, **Object**, and **Class** are filtered out from the information presented to you. As an example, see the description of **PP** in the following section.

10.3.2 Selecting Options in the Left Menu

When the mouse is inside a browser and you hold down the left mouse button, nodes within the browser become inverted when the cursor moves over them, as shown in the following window:



If you release the left mouse button while the cursor is over a node, the following menu appears:



The options shown on the menu operate on the node (class) selected. Several of these options have associated submenus. Common options are in the main menu, and less common ones are menu suboptions. The actions that occur as a result of selecting one of these options are described in the following subsections. An additional subsection describes extended functionality available with the left mouse button.