## 10.4.1 Selecting Options in the Title Bar Menu

The title bar menu in a **FileBrowser**, shown here, is like the title bar menu in a **ClassBrowser**, but has additional entries for file system and Masterscope functions.


### 10.4.1.1 Recompute and its Suboptions

Operates identically to the class browser. See Section 10.3.1.1, "Recompute and its Suboptions," for more information.

### 10.4.1.2 AddRoot and its Suboptions

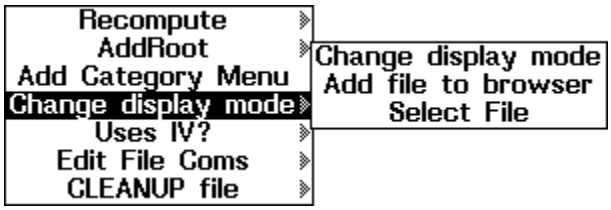
Prompts you for the name of a class to add. If the class already exists, it is added to the browser and shaded. If the class is not contained on the file, it will appear shaded in the browser. If the class does not exist, it is created and added to the selected file.

### 10.4.1.3 Add Category Menu

Operates identically to the class browser. See Section 10.3.1.3, "Add Category Menu and its Suboptions," for more information.

### 10.4.1.4 Change display mode and its Suboptions

Selecting the **Change display mode** option and dragging the mouse to the right causes the following submenu to appear:




A **FileBrowser** logically includes more display options than a **ClassBrowser**. A **FileBrowser** can display a class hierarchy as it is stored in the file, or as it exists in combination with other files and the system as a whole.

**Change display mode** Selecting this option causes a sub-submenu to appear showing three options:

- **selectedFile**

  Displays only the classes contained within the selected file or classes that have been added to the browser by **AddRoot** or **AddSubs** (by setting the browser's instance variable **goodList** to the appropriate value). See

Section 10.5.3, "Methods for the Class LatticeBrowser," for an explanation of **AddSubs**.

- **associatedFiles**

  Same as **selectedFile**, but the browser also includes any classes defined in files associated with the browser. The instance variable **goodList** is bound to this list, slightly differently than the use of **goodList** in a class browser.

- **all**

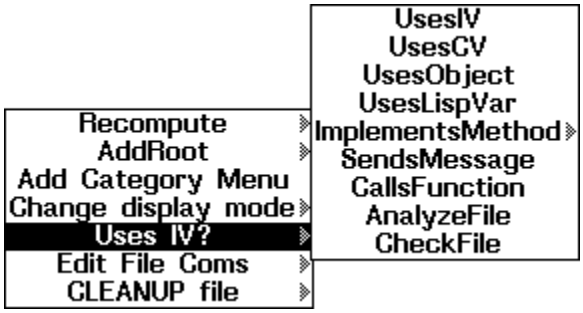
  Same as **associatedFiles,** but any subclasses, even if not defined in the files, are also displayed because the instance variable **goodList** is bound to NIL.

**Add file to browser** Prompts you with a menu that is similar to the menu that is displayed when you select **Browse File** from the LOOPS Icon or the background menu, except that files already associated with the browser are not displayed on the menu.

**Select file** Causes a menu to appear, showing the files associated with the browser. Selecting one causes that file to become the "selected file" of the browser. It clears the browser of any classes added to the browser by the **AddRoot** and **AddSubs** menu commands. That is, it resets the starting list of the browser to be only those classes contained within the selected file. The instance variable **badList** of the browser is set to NIL.

#### 10.4.1.5 Uses IV? and its Suboptions

Selecting the **Uses IV?** option and dragging the mouse to the right causes the following submenu to appear:




These menu options trigger various Masterscope operations. Most of these operations prompt you for information that is used in a Masterscope query. The results of this query are used to build a second menu. If the situation occurs that the second menu is empty, a message is printed in the prompt window of the browser similar to "someCV not used as a CV."

---

**CAUTION**

Source files being displayed in the file browser must be available or these functions cannot work. In addition, the LOOPS Library Module LOOPSMS must also be loaded (see the *LOOPS Library Modules Manual* for details).

---

Additionally, the first time one of these options is selected there may be a pause while Masterscope analyzes the file. A window will open, and fill with

"blips" as the analysis proceeds, until the file is analyzed and the original question is answered.

**UsesIV** This first opens a menu of instance variables defined in classes contained in the selected file of the browser. Two additional options are placed at the top of the menu:

- ***other***

  Selecting ***other*** causes a prompt to enter the name of an instance variable.

- ***any***

  Selecting ***any*** creates a menu with all methods that reference any instance variable.

After an instance variable has been chosen, you are prompted to place a menu that contains the following options:

- A list of methods on the selected file that use that instance variable
- A list of classes on the selected file that contain that instance variable.
- ***EditAll***

If one of the methods or classes is selected, it is edited. Suboptions from the methods or classes include:

- **Edit**

  Edits the method.

- **Substitute**

  Prompts for a new name for the instance variable. Changes the instance variable name to the new name in the method and then brings up the display editor for you to edit the method.

- **Check**

  Executes the Masterscope command CHECK <file> on the file associated with the LOOPS FileBrowser. See the *Lisp Library Modules Manual* for details.

***EditAll*** has the following two suboptions:


- ***EditAll***

  Edits each method and class in succession.

- ***SubstituteAll***

  Prompts you for a new name for the instance variable. Substitutes this new name for the old name in all methods and classes listed in the menu.

**UsesCV** Same as **UsesIV**, but for class variables.

**UsesObject** Opens a menu of classes or instances defined on the selected file that are used by any of the methods or functions on the selected file. After you choose one of the objects, a menu similar to the one created for **UsesIV** is created, but contains the methods and functions that use the chosen objects.

**UsesLispVar** Same as **UsesIV**, but the initial menu displays Lisp variables instead of objects.

**ImplementsMethod** This option has three suboptions:

- **ImplementsMethod**

  Opens a menu of all of the selectors in the selected file. When one is chosen, a menu is created showing the methods that use that selector and the classes that are associated with those methods.

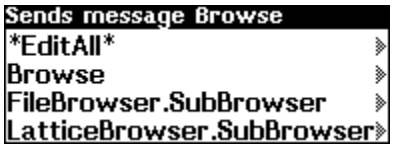
- **OverridesMethod**

  Generates a menu of methods and classes that override (that is, does not invoke _**Super**) the selected method.

- **SpecializesMethod**

  Generates a menu of methods and classes that specialize (that is, invoke _**Super**) the selected method.

**SendsMessage** Opens a menu of all of the selectors in the selected file. When one is chosen, a menu is created that lists the methods and functions that send messages using that selector. The following window shows a sample of this menu.



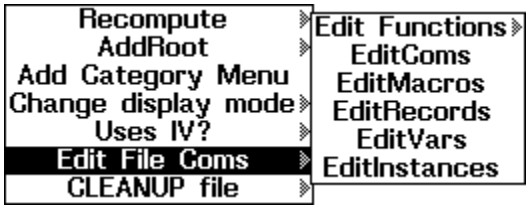

**CallsFunction** Opens a menu of all of the functions that are called by functions or methods in the selected file. After one is chosen, a menu is opened that contains the methods and/or functions that call the chosen function; the last option on the menu is the chosen function.

**AnalyzeFile** Begins a separate process analyzing the selected file. When the analysis is completed, "Done analyzing" is printed in the browser's prompt window.

**CheckFile** Begins a separate process checking the selected file. When the checking is completed, "Done checking" is printed in the browser's prompt window.

#### 10.4.1.6 Edit FileComs and its Suboptions

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



The filecoms are variables that describe the contents of a file, for example, methods, classes, and Lisp functions and variables. LOOPS extends the File Manager to handle object oriented code and data, and the **FileBrowser** gives users a menu driven interface to deal with this extended file functionality.

**Edit Functions** Opens a sub-submenu giving options dealing with filecoms and with some of the items listed (functions in particular). This sub-submenu contains five suboptions:

- **EditFns**

  Opens a menu of the functions contained within the selected file (those listed under FNS in the filecoms) and the option ***NewFunction***. Selecting one of the functions calls the editor on that function.

  Selecting ***NewFunction*** causes a prompt for a name for the new function. An edit window then opens containing a template for a lambda expression. This newly defined function is added to the FNS list of the selected file's filecoms.

- **MakeFunctionMenu**

  Does an **ADDMENU** (the Interlisp function which adds a permanent menu to the screen) of a menu containing functions on the selected file. Selecting one of the functions opens an editor on it.

- **BreakFunction**

  Opens a menu containing functions on the selected file that are not on **BROKENFNS** (see the *Lisp Release Notes* and the *Interlisp-D Reference Manual*). Selecting one of the functions causes it to break next time it is invoked.

- **TraceFunction**

  Same as **BreakFunction**, except that the selected function is traced.

- **UnbreakFunction**

  Creates a menu of functions that are members of **BROKENFNS** and are contained in the selected file. The selected file is unbroken.

**EditComs** Edits the filecoms of the selected file.

**EditMacros** Creates a menu of macros contained in the selected file. Selecting one of them opens an editor on it.

**EditRecords** Same as **EditMacros,** but for records.

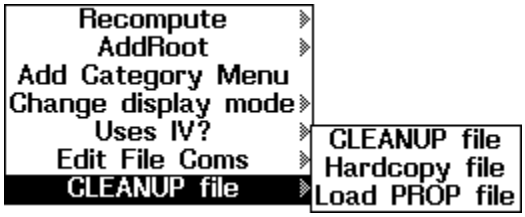
**EditVars** Same as **EditMacros,** but for variables.

**EditInstances** Same as **EditMacros,** but for instances.

#### 10.4.1.7 CLEANUP file and its Suboptions

This option invokes some or all of **CLEANUP**, which is the automatic file maintenance utility of Medley.

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




**CLEANUP file** Calls **FILES?** and then calls **CLEANUP** on the selected file.

**Hardcopy file** Sends the selected file to the **DEFAULTPRINTINGHOST**.

**Load PROP file** Loads the selected file with **LDFLG** set to **PROP**, making sources available to Masterscope, but leaving any compiled code in place to execute.

### 10.4.2 Selecting Options in the Left Menu

When the cursor is inside of a file browser and you press the left mouse button, nodes within the browser are inverted when the cursor moves over them. If you release the left mouse button while the cursor is over a node, the following menu appears:


These options include those in a class browser and add **AddSubs**, an option that expands the lattice of a file browser to look more like that of a class browser by showing related classes not stored in the browsed file.

#### 10.4.2.1 PrintSummary and its Suboptions

Operates identically to the class browser. See Section 10.3.2.1, "PrintSummary and its Suboptions," for more information.

#### 10.4.2.2 Doc (ClassDoc) and Its Suboptions

Operates identically to the class browser. See Section 10.3.2.2, "Doc (ClassDoc) and its Suboptions," for more information.

#### 10.4.2.3 Wherels (WhereIsMethod) and Its Suboptions

Operates identically to the class browser. See Section 10.3.2.3, "Wherels (WhereIsMethod) and its Suboptions", for more information.

#### 10.4.2.4 DeleteFromBrowser and Its Suboptions

Operates identically to the class browser. See Section 10.3.2.4, "DeleteFromBrowser and its Suboptions," for more information.

#### 10.4.2.5 SubBrowser

Creates an instance of a class browser with the selected class as the root node, not a file browser. See Section 10.3.2.5, "SubBrowser," for more information.

#### 10.4.2.6 TypeInName

Operates identically to the class browser. See Section 10.3.2.6, "TypeInName," for more information.

#### 10.4.2.7 AddSubs and its Suboptions

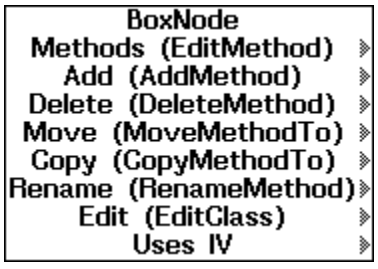
**AddSubs** fills out the class lattice in a file browser window. This shows classes, the file they are from (if any) and the inherited methods and variables from classes which are in the file.

**AddSubs** Adds the immediate subclasses of the class to the browser and shades the new subclasses, as shown here:


**AddSubs!** Adds all subclasses of the class to the browser and shades the new subclasses.

### 10.4.3 Selecting Options in the Middle Menu

When the cursor is over a node and you press the middle mouse button, the following menu appears:




The middle button commands are the same as those on a **ClassBrowser**, with some new functionality for **Add (AddMethod)**, and with Masterscope options added under the option **UsesIV**.

#### 10.4.3.1 BoxNode

Operates identically to the class browser. See Section 10.3.3.1, "BoxNode/UnBoxNode," for more information.

#### 10.4.3.2 Methods (EditMethod) and its Suboptions

Operates identically to the class browser. See Section 10.3.3.2, "Methods (EditMethod) and its Suboptions," for more information.

#### 10.4.3.3 Add (AddMethod) and its Suboptions

Operates identically to the class browser, but with additional functionality to keep the added items associated with the file being browsed in the following suboptions.

**SpecializedClass** Prompts you to enter a name for the new subclass for the chosen class. If the chosen class is on the selected file, the new subclass is added to that file. If the chosen class is on another file, choose a file from a menu of files to which the new subclass is added.

**NewInstance** Creates a new instance of the class and calls **PutSavedValue** with the new instance as an argument. You are prompted to give the new instance a name, and the new instance is added to the selected file.

#### 10.4.3.4 Delete (DeleteMethod) and its Suboptions

Operates identically to the class browser. See Section 10.3.3.4, "Delete (DeleteMethod) and its Suboptions," for more information.

#### 10.4.3.5 Move (MoveMethodTo) and its Suboptions

Operates identically to the class browser. See Section 10.3.3.5, "Move (MoveMethodTo) and its Suboptions," for more information.

#### 10.4.3.6 Copy (CopyMethodTo) and its Suboptions

Operates identically to the class browser. See Section 10.3.3.6, "Copy (CopyMethodTo) and its Suboptions," for more information.

#### 10.4.3.7 Rename (RenameMethod) and its Suboptions

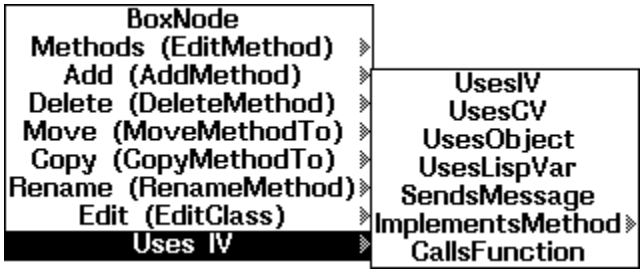
Operates identically to the class browser. See Section 10.3.3.7, "Rename (RenameMethod) and its Suboptions," for more information.

#### 10.4.3.8 Edit (EditClass) and its Suboptions

Operates identically to the class browser. See Section 10.3.3.8, "Edit (EditClass) and its Suboptions," for more information.

#### 10.4.3.9 UsesIV and its Suboptions

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




These commands operate similarly to the **Uses IV?** commands in the title bar menu. See Section 10.4.1.5, "UsesIV? and its Suboptions," for more information. Here, however, the Masterscope queries are limited to the class in question instead of the entire file.


10.5 PROGRAMMER'S INTERFACE TO LATTICE BROWSERS

## 10.5 Programmer's Interface to Lattice Browsers

LOOPS browsers are standard LOOPS objects, so their functionality can be exercised programmatically by messages which invoke their methods. Many browser functions are based on the Lisp Library Module, Grapher, but browsers apply only to dealing with LOOPS objects. All of the functionality in the menu-driven interface to the browsers is available programmatically.

Note: Data not a part of LOOPS data can be graphed with LOOPS calls to the Lisp Library Module, Grapher.

**ClassBrowsers** show the inheritance structure of object classes, a relationship defined at application design time. Browsers can also be used to show dynamic information, not computed until runtime. The LOOPS class **InstanceBrowser** does this, showing links between instances defined at runtime.

**InstanceBrowser** is derived from **LatticeBrowser** by specializing just two methods, **GetSubs** and **NewPath**. In general, you can specialize browsers to your own purposes by specializing these methods and **GetLabel**. An example is given in Section 10.7, "Class Instance Browser Example," producing a class browser which also shows instances, connected with dashed lines.

### 10.5.1 Instance Variables for the Class LatticeBrowser

Instance variables appear in alphabetical order.

**badList** A list of objects that are not displayed in the browser window.

**boxedNode** The last object boxed, if any.

**browseFont** The font used for labels. This has two properties: **FontFamily** and **FontFace** which are referenced in the method **ChangeFontSize**.

**lastSelectedObject** The last object selected.

**goodList** A list of objects that are displayed in the browser window; if NIL, no objects are displayed.

**graphFormat** A list indicating the style of layout for the graph. See the method **ChangeFormat** and the Lisp Library Module, Grapher.

**LabelMaxCharsWidth** Affects the way labels are generated. This limits the width of a label to **LabelMaxCharsWidth** times the width of the character "A". See the method **ChangeMaxLabelSize**. Default value is NIL, which puts no restrictions on label size.

**LabelMaxLines** Affects the way labels are generated. This limits the number of lines in a label to **LabelMaxLines**. Refer to the method **ChangeMaxLabelSize**. Default value is NIL which puts no restrictions on label size.

**startingList** List of objects used to compute this browser.

**title** Title passed to Grapher module.

**topAlign** This flag is used to indicate whether the graph should be aligned with the top or bottom of the window. If **topAlign** = T (the default), then the Grapher module aligns the graph to the top of the window.

## 10.5.2 Class Variables for the Class LatticeBrowser

Except for **BoxLineWidth**, the following class variables determine the menus that appear when a mouse is positioned over a node within a browser and the left or middle button is pressed. The default behavior is to send a message to the object represented by the node with the selector returned from the menu selection. The form for the values that these class variables can have is described in Chapter 20, Windows.

Class variables appear in alphabetical order.

**BoxLineWidth** The width of line that is drawn around a node when it is boxed. See the method **BoxNode** in Section 10.5.3, "Methods for the Class LatticeBrowser."

**LeftButtonItems** Items for the menu that appears when the mouse is on a node in the browser and the left button is pressed. When an item is selected from a menu, the returned value is sent as a message to an object represented by the node. See **LocalCommands**, below.

**LocalCommands** When the cursor is positioned over a node in a browser and you press the left or middle mouse button, the default behavior is to bring up a menu from which you select an item. The value returned from that item specifies the selector of a message that is sent to the object which is represented by the node in the browser.

The class variable **LocalCommands** provides a way to override that behavior. If the value returned from the menu selection is on the list that is the value of **LocalCommands**, the message is not sent to the object, but is sent to the browser instead. The object is passed as an argument in that message.

**MiddleButtonItems** Options for the menu that appears when the mouse is on one of the nodes in a browser and the middle button is pressed. When an option is selected from a menu, the returned value is sent as a message to the object represented by the node. See **LocalCommands**, above.

**TitleItems** Options for the menu that appears when the mouse is on the title bar in a browser and the left or middle button is pressed. When an option is selected from the menu, the returned value is sent as message to the browser.

The following selectors are associated with this menu:

- **SaveInIT**
- **Recompute**
- **RecomputeInPlace**

- **ShapeToHold**
- **ChangeFontSize**
- **ChangeFormat**
- **AddRoot**
- **RemoveFromBadList**

Examine the class **LatticeBrowser** for more information.

### 10.5.3 Methods for the Class LatticeBrowser

The following table shows the methods and variables for the class **LatticeBrowser**.

| Name | Type | Description |
|---|---|---|
| **AddRoot** | Method | Adds a LOOPS name or an object to the starting list of the browser. |
| **BoxNode** | Method | Puts a box around the node representing the object. |
| **Browse** | Method | Uses a lattice or tree graph to display the relationship between a number of objects. |
| **BrowserObjects** | Method | Returns the list of objects currently in the graph. |
| **ChangeFontSize** | Method | Changes the size of the characters in the labels. |
| **ChangeFormat** | Method | Changes between lattice and tree graphs. |
| **ChangeMaxLabelSize** | Method | Changes how labels are printed. |
| **ClearLabelCache** | Method | Recomputes labels. |
| **DeleteFromBrowser** | Method | Prunes branches in a graph. |
| **DeleteSubtreeFromBrowser** | Method | Prunes branches in a graph. |
| **FlashNode** | Method | Changes the label of a node from black-on-white to white-on-black several times. |
| **FlipNode** | Method | Changes the label of a node that represents an object from black-on-white to white-on-black. |
| **GetDisplayLabel** | Method | Finds the label for a node. |
| **GetLabel** | Method | Computes a label for an object. |
| **GetSubs** | Method | Computes a list of subnodes of an object. |
| **GraphFIts** | Method | Determines if the graph in the browser can be contained within the browser window. |
| **HasObject** | Method | Returns T if an object is in the graph. |
| **HighlightNode** | Method | Changes the way a node is displayed. |
| **IconTitle** | Method | Computes the title to write in the icon. |
| **LeftSelection** | Method | Controls the effect of using the left mouse button. |

| | | |
|---|---|---|
| **LeftShiftSelect** | Method | Sends the message **PP!** to an object. |
| **MiddleSelection** | Method | Controls the effect of using the middle mouse button. |
| **MiddleShiftSelect** | Method | Invoked by the mouse operations to edit an object in the TTY process context. |
| **NewItem** | Method | Gets an object. |
| **NodeRegion** | Method | Returns the region occupied in an object in the browser. |
| **ObjectFromLabel** | Method | Returns the object in the graph that has a specified label. |
| **PositionNode** | Method | Places a node at a particular position in the browser window. |
| **Recompute** | Method | Recomputes the browser graph in the same window. |
| **RecomputeInPlace** | Method | Recomputes the browser graph in the same window, trying to maintain the same scroll position in the window. |
| **RecomputeLabels** | Method | Recomputes the labels in a browser. |
| **RemoveHighlights** | Method | Removes all highlights in any node in the graph. |
| **RemoveShading** | Method | Removes all shading in any node in the graph. |
| **SaveInIt** | Method | Places a pointer in a browser where it can be accessed. |
| **ShadeNode** | Method | Adds shading to a node. |
| **ShapeToHold** | Method | Reshapes the window to all items. |
| **MaxLatticeWidth** | Variable | Restricts the maximum width of a browser window. |
| **MaxLatticeHeight** | Variable | Restricts the maximum height of a browser window. |
| **Show** | Method | Displays items and their subitems in a browser window. |
| **Shrink** | Method | Shrinks a browser window. |
| **SubBrowser** | Method | Creates a browser that is an instance of the same class as *self* with a specified object as the root. |
| **TitleSelection** | Method | Invokes an action when the mouse is in the title bar on a browser window and the left or middle button is pressed. |
| **UnmarkNodes** | Method | Sends the messages **RemoveHighlights** and **RemoveShading** to *self*. |

(← *self* **AddRoot** *newItem*) [Method of LatticeBrowser]

Purpose: Adds *newItem*, which is a LOOPS name, to the starting list of the browser.

Behavior: First determines if the name *newItem* points to a LOOPS object. If it does not, a message is printed that nothing has been added to the browser. If *newItem* is NIL, you are prompted to enter a name through the method **NewItem**. If the object pointed to by *newItem* is on the browser's instance variable **badList**, it is removed from **badList.** If the instance variable **goodList** has a value, *newItem* is added to it.

Arguments: *newItem* LOOPS name.

Returns: Class object or NIL.

Categories: LatticeBrowser

Example: The following command adds class **Datum** to a class browser instance named **CB1**:

```
55←(← ($ CB1) AddRoot ($ Datum))
```

(← *self* **BoxNode** *object objName unboxPrevious*) [Method of LatticeBrowser]

Purpose: Puts a box around the node in the graph representing the object.

Behavior: First checks to make sure *object* points to a LOOPS object. If not, nothing happens. The previous value of the instance variable **boxedNode** is returned.

- If the instance variable **boxedNode** is NIL, then a box is drawn around the node with a line width equal to the value of the class variable **BoxLineWidth**. The instance variable **boxedNode** is assigned the value of *object*, and *object* is returned.
- If *object* is EQ to the instance variable **boxedNode**, then the box is erased. That is, calling **BoxNode** twice in succession will draw and then erase the box. The instance variable **boxedNode** is assigned the value NIL, and NIL is returned.
- If none of the above conditions hold, the flag **unboxPrevious** is checked. If it is non-NIL, the previously boxed node is unboxed, and the node represented by *object* is boxed. The instance variable **boxedNode** is assigned the value of *object*, and *object* is returned.

It is possible that *object* is not a node in *self*.

Arguments:
*object* LOOPS name or object.

*objName* Used internally; can be NIL.

*unboxPrevious*
Can be NIL or T.

Returns: The object in the browser that is currently boxed, or NIL if nothing is currently boxed.

Categories: LatticeBrowser

Specializations: ClassBrowser

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

Purpose: Uses a lattice or tree graph to display the relationships between a number of objects. **Browse** is the proper message to use for initializing browsers.

Behavior: Sends the message **Show** to *self* passing the arguments *browseList, windowOrTitle*, and *goodList*. It next sends **ShapeToHold** to *self*. Finally it sends **Move** to *self* with the argument *position*.

Arguments:
*browseList* A list, elements of which can be a LOOPS name or an object, or a single item which can be a LOOPS name or an object. Used as the starting node(s) of a browser. See the **Show** message later in this section for details.

*windowOrTitle*
If a window, the browser is displayed in this window. If not a window, this becomes the title of the browser window.

*goodList* A list, elements of which can be a LOOPS name or an object. See the **Show** message later in this section for details.

*position* A position to which the lower left corner of the browser is moved. Can be NIL.

Returns: Used for side effect only.

Categories: LatticeBrowser

Example: The following command gets the class browser instance **CB1** to browse class **Datum** and its subclasses:

```
57←(← ($ CB1) Browse 'Datum)
```

(← *self* **BrowserObjects**) [Method of LatticeBrowser]

Purpose/Behavior: Returns the list of objects currently in the graph of the browser.

Categories: LatticeBrowser

(← *self* **ChangeFontSize** *size*) [Method of LatticeBrowser]

Purpose: Changes the size of the characters of the labels.

Behavior: Changes the font used to display labels in a browser. The browser is redrawn and the window shaped to fit. If no size is given, this lets you select the size from a menu. This menu is bound to the top level binding of the variable **MenuSize** (the first time it is called). The font family used is the value of the **FontFamily** property of the instance variable **browseFont**. The font face used is the value of the **FontFace** property of the instance variable **browseFont**.

This sends the message **RecomputeLabels** to *self*.

Arguments: *size* Integer size of font for node labels.

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **ChangeFormat** *format*) [Method of LatticeBrowser]

Purpose: Changes between lattice and tree graphs.

Behavior: If *format* is NIL, then select a format from a menu that appears. The items in this menu are determined by the value of the **choices** property of the instance variable **graphFormat**. Changes the value of the instance variable **graphFormat** to *format* or the value selected from the menu.

Arguments: *format* Describes the format layout. The argument *format* is an unordered list of atoms or lists. The following options control the structure of the graph:

- COMPACT, the default, which lays out the graph as a forest (that is, a set of disjoint trees) using the minimal amount of screen space.
- FAST, which lays out the graph as a forest, sacrificing screen space for speed.
- LATTICE, which lays out the graph as a directed acyclic graph, that is, a lattice.

In addition, the following options control the direction of the graph:

- HORIZONTAL, the default, has roots at the left and links that run left-to-right.
- VERTICAL has roots at the top and links that run top-to-bottom.

See the function **LAYOUTGRAPH** in the Grapher library module documentation for more information.

Returns: Used for side effect only.

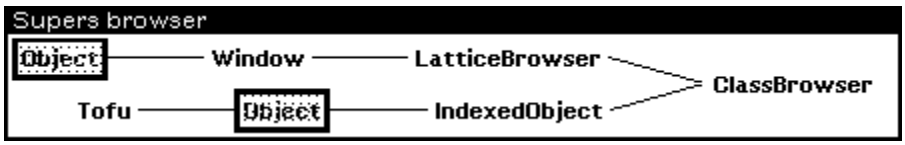
Categories: LatticeBrowser

Example: The commands:

```
58← (SETQ b1 (←New ($ SupersBrowser) Browse 'ClassBrowser))
#,($& SupersBrowser (NEW0.1Y%:.;h.eN6 . 506))

59←(← b1 ChangeFormat '(HORIZONTAL REVERSE (MARK BORDER 3 LABELSHADE 1)))
#,($& SupersBrowser (NEW0.1Y%:.;h.eN6 . 506))
```

results in:




(← *self* **ChangeMaxLabelSize** *newMaxWidth newMaxLines*) [Method of LatticeBrowser]

Purpose: Changes how labels are printed.

Behavior: Sets the maximum width of a node label. An argument value of zero means no maximum size, and NIL means no change.

By setting both *newMaxWidth* and *newMaxLines*, you get an abbreviation facility. This binds the values of the instance variables **LabelMaxCharsWidth** and **LabelMaxLines**. The default values for **LabelMaxCharsWidth** and **LabelMaxLines** are both 0, so to return a browser b1 to default performance, send

```
(← ($ b1) ChangeMaxLabelSize 0 0).
```

The resulting labels may be bitmaps or strings. The width of the bitmap is a product of *newMaxWidth* and the width of the character "A" in the current value of the instance variable **browsefont**.

Sends the message **RecomputeLabels** to *self*.

Arguments: *newMaxWidth*
Maximum number of characters per line; default is 0.

*newMaxLines*
Maximum number of lines; default is 0.

Returns: Used for side effect only.

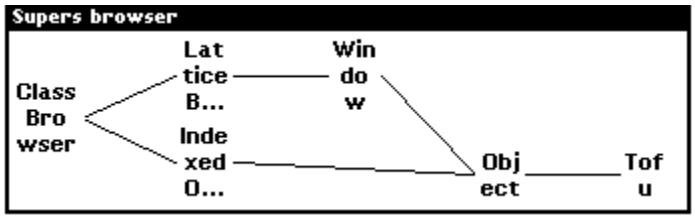
Categories: LatticeBrowser

Example: The commands:

```
60←(SETQ b1 (←New ($ SupersBrowser) Browse 'ClassBrowser))
#,($& SupersBrowser (NEW0.1Y%:.;h.eN6 . 506))
```

```
61←(← b1 ChangeMaxLabelSize 3 3)
#,($& SupersBrowser (NEW0.1Y%:.;h.eN6 . 506))
```

results in:




(← *self* **ClearLabelCache** *objects*) [Method of LatticeBrowser]

Purpose: Forgets cached labels in the browser.

Behavior: Clears the label cache for item(s) in *objects*. If *objects* is the symbol T, then this clears the entire label cache. The cache for the labels is on the **objectLabels** property of the instance variable **menus**.

Arguments: *objects* An object or a list of objects.

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **DeleteFromBrowser** *object objname*) [Method of LatticeBrowser]

Purpose: Prunes branches in a graph.

Behavior: Removes *object* from the browser by putting it on the instance variable **badList** and then sending the **Recompute** message to *self*. The object and its subtree are deleted from the browser.

Arguments: *object* An object in the browser.

*objname* Used internally; can be NIL.

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **DeleteSubtreeFromBrowser** *object*) [Method of LatticeBrowser]

Purpose: Prunes branches in a graph.

Behavior: Similar to **DeleteFromBrowser**, but the subnodes of *object* are also added to the instance variable **badList**.

Arguments: *object* An object in the browser.

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **FlashNode** *node N flashTime leaveFlipped?*) [Method of LatticeBrowser]

Purpose/Behavior: Changes the label of a node from black-on-white to white-on-black several times.

Arguments:
*node* LOOPS name or object.

*N* Number of times *node* will be flipped.

*flashTime* The amount of time in milliseconds that the node is held between transitions. If *flashTime* is NIL, this time defaults to 300 milliseconds.

*leaveFlipped?* Can be NIL or T. If T, *node* is left inverted from its original state.

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **FlipNode** *object*) [Method of LatticeBrowser]

Purpose: Inverts the label of a node.

Behavior: If the node is black-on-white then it is changed to white-on-black, and conversely.

Arguments:
*object* LOOPS name or object.

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **GetDisplayLabel** *object*) [Method of LatticeBrowser]

Purpose: Finds the label for a node in the graph.

Behavior: If there is a cached label on the **objectLabels** property of the instance variable **menus**, return it.

If not, this takes the result of **GetLabel** and breaks it into multiple lines to fit in the maximum label size defined by the instance variables **LabelMaxCharsWidth** and **LabelMaxLines**, if these are non-NIL. This placement tries to break the label after special characters such as . , : ; / or space, or at changes from lowercase to uppercase. The resulting bitmap is put into cache so that recomputing the graph is faster.

When a label is broken up into multiple lines, the label is changed from a string to a bitmap, thus causing shading not to work as described later in this section in the method **ShadeNode**.

Arguments:
*object* An object.

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **GetLabel** *object*) [Method of LatticeBrowser]

Purpose: Computes a label for *object*. A label may be a symbol or a bitmap; bitmap labels should be freshly created since the method **ShadeNode** may smash them. (The method **GetDisplayLabel** is used internally to fetch labels for display; it caches label bitmaps to minimize the use of **GetLabel.**)

Behavior: Returns (**GetObjectName** *object*).

Arguments: *object* LOOPS name or object, which can be a bitmap.

Returns: (**GetObjectName** *object*)

Categories: LatticeBrowser

(← *self* **GetSubs** *object*) [Method of LatticeBrowser]

Purpose: Computes a list of subnodes of *object*.

Behavior: Determines next level of nodes in lattice. Specializations of **LatticeBrowser** typically specialize this method.

Arguments: *object* A LOOPS object.

Returns: NIL or the value of the instance variable **subs** of *object*.

Categories: LatticeBrowser

Specializations: ClassBrowser, InstanceBrowser, MetaBrowser, SupersBrowser

(← *self* **GraphFits** *snugly*) [Method of LatticeBrowser]

Purpose/Behavior: Determines if the graph in the browser can be contained within the window of the browser.

Arguments: *snugly* If snugly? is non-NIL the graph must fit in the window leaving less than twice the **FONTHEIGHT** of the browser's browseFont as empty space around it.

Returns: T if the entire graph can be displayed within the window; else NIL.

Categories: LatticeBrowser

(← *self* **HasObject** *object*) [Method of LatticeBrowser]

Purpose/Behavior: Returns T if *object* is in the graph of the browser.

Arguments: *object* LOOPS name or object.

Categories: LatticeBrowser

(← *self* **HighlightNode** *object width shade*) [Method of LatticeBrowser]

Purpose: Changes the way a node is displayed.

Behavior: Draws a box around a node for *object* using a given *width* and *shade* for the lines of the box. A shade is a 16-bit number representing a 4x4 bitmap. See **EDITSHADE** in the *Interlisp-D Reference Manual*.

Arguments: *object* LOOPS name or object.

*width* Integer width of box.

*shade* 16-bit number representing a 4x4 bitmap.

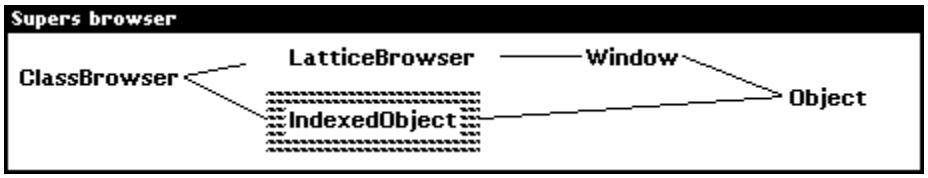
Returns: Used for side effect only.

Categories: LatticeBrowser

Example: The command

```
64←(← b1 HighlightNode 'IndexedObject 10 123)
```

results in the following window:




(← *self* **IconTitle**) [Method of LatticeBrowser]

Purpose/Behavior: Computes the title to write in the icon.

Returns: The label of the first root entry in the lattice, that is, the **CAR** of the instance variable **startingList**. If this is NIL, then use "Browser". If **AddRoot** is called, the new root becomes the first entry on **startingList**.

Categories: LatticeBrowser

Specializations: FileBrowser

(← *self* **LeftSelection**) [Method of LatticeBrowser]

Purpose: Controls the effect of using the left mouse button. LatticeBrowser provides defaults, but allows these methods to be overwritten or specialized.

Behavior: The instance variable **lastSelectedObject** is bound to the object of the node that the left button selected.

The remaining behavior varies according to the key pressed.

- If the **Move** key or the **Control** key is down, this allows you to move the node the mouse is over when you press the left mouse button.
- If the left shift key or **Copy** key is down while the cursor is over a node, the label of the node is copied to the system buffer. If the cursor is not over a node, the entire graph is copied. This allows you to copy browsers into TEdit documents.
- If the **Meta** key is pressed, the message **LeftShiftSelect** is sent to *self* passing as an argument the object the cursor is over.
- If none of the above keys are down, a menu pops up. This may trigger additional functionality to be evaluted in the TTY process context.

This method is generally not called directly by the user, but is invoked by mouse operations.

Returns: Used for side effect only.

Categories: Window

Specializes: Window

(← *self* **LeftShiftSelect** *object objectName*) [Method of LatticeBrowser]

Purpose/Behavior: Sends the message **PP!** to *object*. LatticeBrowser provides defaults, but allows these methods to be overwritten or specialized.

This is generally not called directly by the user, but is invoked by mouse operations.

Arguments: *object* An object.

*objectName* Used internally; can be NIL.

Returns: Used for side effect only.

Categories: LatticeBrowser

Specializations: ClassBrowser

(← *self* **MiddleSelection**) [Method of LatticeBrowser]

Purpose: Controls the effect of using the middle mouse button. LatticeBrowser provides defaults, but allows these methods to be overwritten or specialized.

Behavior: If no node is selected, then returns NIL.

The instance variable **lastSelectedObject** is bound to the object of the node that the middle button selected.

If the **Meta** key is down, the message **MiddleShiftSelect** is sent to *self*.

If the **Meta** key is not down, a menu pops up. This may trigger additional functionality to be evaluated in the TTY process context.

This is generally not called directly by the user, but is invoked by mouse operations.

Returns: Used for side effect only.

Categories: Window

Specializes: Window

(← *self* **MiddleShiftSelect** *object objname*) [Method of LatticeBrowser]

Purpose/Behavior: Edits *object* in the TTY process context. LatticeBrowser provides defaults, but allows these methods to be overwritten or specialized.

This is generally not called directly by the user, but is invoked by mouse operations.

Arguments: *object* LOOPS object.

*objname* Used internally; can be NIL.

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **NewItem** *newItem*) [Method of LatticeBrowser]

Purpose: Gets an object.

Behavior: If *newItem* is NIL, a prompt appears in an attached prompt window for the name of the item to be added.

Arguments: *newItem* LOOPS name or object.

Returns: An object pointed to by *newItem* or the name entered by the user at the prompt.

Categories: LatticeBrowser

Specializes: None.

Specializations: ClassBrowser, FileBrowser

(← *self* **NodeRegion** *object*) [Method of LatticeBrowser]

Purpose/Behavior: Returns the region occupied by *object* in the browser.

Arguments: *object* LOOPS name or object.

Returns: A region defined in terms of the coordinates of the browser's window.

Categories: LatticeBrowser

(← *self* **ObjectFromLabel** *label*) [Method of LatticeBrowser]

Purpose/Behavior: Returns the object displayed in the browser that has the given *label*, or NIL if no object labelled with *label* is visible in the browser.

Arguments: *label* Symbol or bitmap as it appears in the *objectLabels* property of the instance variable *menus*.

Returns: The object in the graph that has the given *label*, or NIL if there is no such object.

Categories: LatticeBrowser

Example: The following command gets the object which is being displayed in browser **CB1** as Datum:

```
65←(← ($ CB1) ObjectFromLabel 'Datum)
```

(← *self* **PositionNode** *object windowX windowY*) [Method of LatticeBrowser]

Purpose: Places a node at a particular position in a scrollable browser window.

Behavior: When the browser is too large for its window and has become scrollable, **PositionNode** scrolls the graph so that the node for the given object is at the position (*windowX . windowY*) within the restrictions of the window property **SCROLLEXTENTUSE** (see the *Interlisp-D Reference Manual*). As in the **ScrollWindow** method of the class **Window**, if either of the arguments *windowX* or *windowY* is a **FLOATP**, it is taken to be a proportional position. For example,

- (0.0, 0.0) is the lower left corner.
- (1.0, 1.0) is the upper right corner.
- (0.5, 0.5) is the center of the window.

If either is a **FIXP**, it is a position in the displayStream coordinate system. Any null argument is taken to be 0. **PositionNode** may only be sent to browsers which are scrollable.

Arguments: *object* LOOPS name or object represented by a node.

*windowX* X-coordinate for new node position. If type **FLOATP**, then a relative position; if type **FIXP**, then absolute position.

*windowY* Y-coordinate for new node position. If type **FLOATP**, then a relative position; if type **FIXP**, then absolute position.

Returns: Position of **CLIPPINGREGION** of window after scrolling.

Categories: LatticeBrowser

(← *self* **Recompute** *dontReshapeFlg*) [Method of LatticeBrowser]

Purpose: Recomputes the browser graph in the same window. Typically used after adding or deleting from the lattice.

Behavior: Sends the **Show** message with the value of the instance variable **startingList** to *self*. If *dontReshapeFlg* is NIL or if the graph does not fit the window(as determined by **GraphFits** with *snugly* flag set to T), then send *self* the message **ShapeToHold**.

Arguments: *dontReshapeFlg*
If non-NIL, do not reshape browser.

Returns: *self*

Categories: LatticeBrowser

Specializations: FileBrowser

(← *self* **RecomputeInPlace**) [Method of LatticeBrowser]

Purpose/Behavior: Recomputes the graph, trying to maintain the same scroll position in the window.

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **RecomputeLabels**) [Method of LatticeBrowser]

Purpose: Recomputes the labels in a browser.

Behavior: Performs the following sequence of expressions:

```
(← self ClearLabelCache T)
(← self Recompute)
```

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **RemoveHighlights**) [Method of LatticeBrowser]

Purpose/Behavior: Removes all highlights on any node in the graph, including nodes that are boxed (see the method **HighlightNode**, which is described earlier in this section). The method **Recompute** maintains shading and boxing and does not do a **RemoveHighlights**. **RemoveHighlights** does not do a Recompute automatically.

Sets the value of the instance variable **boxedNode** to NIL.

Returns: Used for side effect only. **RemoveHighlights** does not do a **Recompute** automatically.

Categories: LatticeBrowser

(← *self* **RemoveShading**) [Method of LatticeBrowser]

Purpose/Behavior: Removes all shading on any node in the graph (see the method **ShadeNode**, which is described later in this section). Does not automatically do a Recompute.

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **SaveInIT**) [Method of LatticeBrowser]

Purpose: Places the pointer to *self* where it can be accessed by `(SavedValue).`

Behavior: Calls (**PutSavedValue** *self*).

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **ShadeNode** *object shade*) [Method of LatticeBrowser]

Purpose: Adds shading to a node.

Behavior: Shades the inside of node with a given *shade* if the node is defined with a string **nodeLabel**, which is the usual case in lattice browsers. Six shades are available by name and are shown in Figure 10-5.

- **WHITESHADE**
- **GRAYSHADE1**
- **GRAYSHADE2**
- **GRAYSHADE3**
- **GRAYSHADE4**
- **BLACKSHADE**

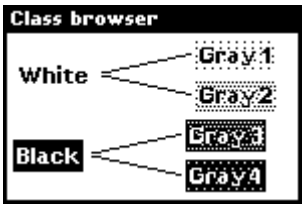



*Figure 10-5. Shading Available for a Node*

If the label displayed for *object* is a bitmap, **ShadeNode** will destructively shade that bitmap.

Arguments: *object* LOOPS name or object.

*shade* A texture (See the *Interlisp-D Reference Manual*).

Returns: Used for side effect only.

Categories: LatticeBrowser

Example: The following command shades the node for **Gray1** with GRAYSHADE1:

```
66←(← ($ CB1) ShadeNode 'Gray1 GRAYSHADE1)
```

(← *self* **ShapeToHold**) [Method of LatticeBrowser]

Purpose/Behavior: Reshapes the window to hold all the items comfortably, unless they would fill up the screen or more.

The window is not shaped larger than **MaxLatticeWidth** by **MaxLatticeHeight** (see below).

Returns: Used for side effect only.

Categories: LatticeBrowser

**MaxLatticeWidth** [Variable]

Purpose: Restricts the maximum width of a browser window.

Behavior: Initialized to 900.

**MaxLatticeHeight** [Variable]

Purpose: Restricts the maximum height of a browser window.

Behavior: Initialized to 750.

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

Purpose: Displays items and their subitems in a browser window. In general, uses the method **Browse** which calls **Show.**

Behavior: The instance variable **startingList** is assigned the value of objects referred to in *browseList*. If the argument *goodList* is provided, the instance variable **goodList** is assigned the value of objects referred to in it.

Arguments:

*browseList* A list, elements of which can be a LOOPS name or an object, or a single item which can be a LOOPS name or an object. Used as the starting node(s) of a browser.

*windowOrTitle* If a window, the browser is displayed in this window. If not a window, this becomes the title of the browser window.

*goodList* Optional. If provided, it is a list of LOOPS objects or object names which are the only items the browser will display as nodes. As opposed to *badList*, which excludes nodes from display, *goodList* gives a population that nodes must be members of to be displayed.

Returns: Used for side effect only.

Categories: LatticeBrowser

(← *self* **Shrink**) [Method of LatticeBrowser]

Purpose: Shrinks a browser window to its icon.

Behavior: If the window already has an icon, this is used. Otherwise, builds an icon (see example below) that has (_ *self* **IconTitle**) as a title. When the icon is expanded, the browser uses a **RecomputeInPlace**.

When the mouse is positioned on an icon and the left or middle button is pressed when the **META** key is down, this sends the message **TitleSelection** to the browser the icon represents.

The browser icon bitmap template is stored on the variable **BrowserIconBM**.

Returns: The icon for *self*.

Categories: Window

Specializes: Window

Example: All browser classes use the same icon:


(← *self* **SubBrowser** *obj objName*) [Method of LatticeBrowser]

Purpose/Behavior: Creates a browser that is an instance of the same class as *self* with *object* as the root node.

Arguments: *obj* LOOPS name or object .

*objName* Used internally; can be NIL.

Returns: The new browser.

Categories: LatticeBrowser

Specializations: FileBrowser

Example: The following command creates a new browser on just the **ClassBrowser** subtree of a browser **CB1** showing the entire **LatticeBrowser** lattice:

```
67←(← ($ CB1) SubBrowser 'ClassBrowser)
```

(← *self* **TitleSelection**) [Method of LatticeBrowser]

Purpose: This message is sent to a browser when the mouse is positioned on its title bar and either the left or middle mouse buttons are pressed. It should not be sent directly by users.

Behavior: Opens a menu, created from the class variable *TitleItems*, from which you pick an entry. The resulting action that this causes is evaluated in the context of the TTY process.

Returns: Used for side effect only.

Categories: Window

Specializes: Window

TitleItems [Class Variable]

Purpose: Holds the menu list used by the TitleSelection method.

(← *self* **UnmarkNodes**) [Method of LatticeBrowser]

Purpose/Behavior: Sends the messages **RemoveHighlights** and **RemoveShading** to *self*.

Returns: Used for side effect only.

Categories: LatticeBrowser



## 10.6 INSTANCE BROWSERS

## 10.6 Instance Browsers

Instance browsers show linkages between instances. That is, each node in an instance browser represents an instance. The links are determined by the value of a particular instance variable in each instance; the value should point to the subitems of an instance.

This section includes the instance variables, methods, and an example of instance browsers.

### 10.6.1 Instance Variables for the Class InstanceBrowsers

**subIV** Nodes within an instance browser have subitems determined by the value of a particular instance variable within each instance. The value of **subIV** is the name of that particular instance variable. Initialized to NIL.

### 10.6.2 Methods for the Class InstanceBrowsers

The **GetSubs** and **NewPath** methods are available for the class **InstanceBrowsers**.

(← *self* **GetSubs** *object*) [Method of InstanceBrowser]

Purpose: Computes the subitems for a node in a browser.

Behavior: If *self* has a value for the instance variable *subIV*, which should be a symbol, and *object* has an instance variable named with that symbol, then return the value of that instance variable in *object*.

Arguments: *object* An object in the browser.

Returns: The subitems of *object,* which should be a list.

Categories: LatticeBrowser

Specializes: LatticeBrowser

Example: The following commands perform these actions:

- Create an instance of **InstanceBrowser** and call it **IB1**.
- Give the value of its instance variable **subIV** the symbol **nextWindow**.
- Create a LOOPS window and call it **W1**.
- Add the instance variable **nextWindow** to it and give it the value of another LOOPS window, instance **W2**.
- Send the **Browse** message to **IB1** with **W1** as the root node.

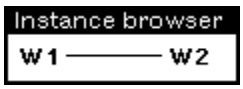
```
68←(← ($ InstanceBrowser) New 'IB1)
#,($& InstanceBrowser (NEW0.1Y%:.;h.eN6 . 515))

69←(←@ ($ IB1) subIV 'nextWindow)
nextWindow

70←(← ($ Window) New 'W1)
#,($& Window (NEW0.1Y%:.;h.eN6 . 516))

71←(← ($ W1) AddIV 'nextWindow (LIST (← ($ Window) New 'W2)))
(#,($& Window --))

72←(← ($ IB1) Browse 'W1)
(407 . 623)
```




(← *self* **NewPath** *subName*) [Method of InstanceBrowser]

Purpose: Specifies which of the instance variables in objects point to subitems.

Behavior: If *subName* is NIL, display a prompt for a value in an attached window.

If *subName* or the value entered is non-NIL, change the value of the instance variable **subIV** of *self* to that value.

Changes the instance variable **title** of the browser window to (CONCAT *subName* " instance browser").

If the browser is open, send the **Recompute** message to it.

Arguments: *subName* A symbol that should be an instance variable within each object of the browser.

Returns: Used for side effect only.

Categories: InstanceBrowser

Example: The following command causes an instance browser **IB1** to look in **W1**'s instance variable **nextPointer** instead of in **nextWindow** to get its subnodes:

```
73_(_ ($ IB1) NewPath 'nextPointer)
```

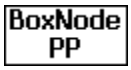
## 10.6.3 Selecting Options in the Title Bar Menu

The title bar menu for instance browsers is a subset of that for class browsers, as shown here:


See Section 10.3.1, "Selecting Options in the Title Bar Menu," for details.

## 10.6.4 Selecting Options in the Left Menu

When you position the cursor on a node of an instance browser and press the left mouse button, the following menu appears:




**BoxNode** is the same as in the class browsers (see Section 10.3.3.1, "BoxNode"). **PP** prints out the instance class, name, and UID.

### 10.6.5 Selecting Options in the Middle Menu

When you position the cursor on a node of an instance browser and press the middle mouse button, the following menu appears:


This menu is a small subset of that for class browsers (see Section 10.3.3, "Selecting Options in the Middle Menu").

- **Inspect** opens an inspector window on the instance.
- **Edit** calls the instance into the editor.
- **DeleteFromBrowser** removes a node from display via the **badList** mechanism.



## 10.7 Automatic Updates of Class Browsers

LOOPS advises the File Manager **LOAD** function to guarantee that all class browsers are updated whenever a file is loaded. The updating is performed by the function **UpdateClassBrowsers** and controlled by the setting of the variable **UpdateClassBrowsers?**.

(**UpdateClassBrowsers** *newLabels?*) [Function]

Purpose: Updates instances of **ClassBrowse**r and its subclasses.

Behavior: Called whenever a new class is defined (including loading from a file) or destroyed. If the variable **UpdateClassBrowsers?** is NIL, then do nothing.

For browsers that have been marked as needing updating and having windows that are opened, this sends the message **Recompute** or **RecomputeLabels** if *newLabels?* is non-NIL.

Arguments: *newLabels?*Can be NIL or T.

Returns: Used for side effect only.

**UpdateClassBrowsers?** [Variable]

Behavior: See the function **UpdateClassBrowsers**, above. Initialized to T.

Values: NIL Do not update browsers.

| | |
|---|---|
| T | Update browsers with each change. |
| SHADE | Shade browsers that need to change. |

[This page intentionally left blank]