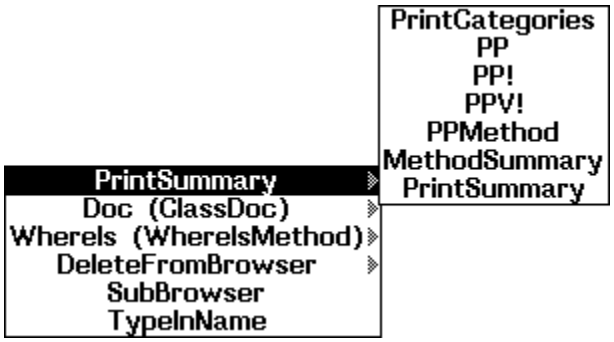
10.3.2.1 PrintSummary and its Suboptions

PrintSummary provides a quick way to see object and method definitions. For all printing that occurs as a result of selecting this option or one of its suboptions, the output is sent to the value of the variable **PPDefault**, which is by default the Common Lisp Executive Window.

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



PrintCategories Prints the categories and associated methods for the selected class, as shown in the following window:

```
  #,($ DestroyedObject)

Object
     Destroy!
```

PP Produces a standard **PrettyPrint** of class, as shown in the following window:

```
(DEFCLASS DestroyedObject
   (MetaClass Class Edited%:
  (* --) )
   (Supers Object))
```

PP! Produces a formatted **Print** of class, as shown in the following window:

```
#,($ DestroyedObject)
MetaClass and its Properties
  Class Edited: (*
TheCollaborators;
15-Oct-84 16:23)
Supers
  (Object Tofu)
Instance Variable Descriptions
Class Variables
Methods
  Destroy! DestroyedObject.Destroy!
 doc NIL args NIL
```

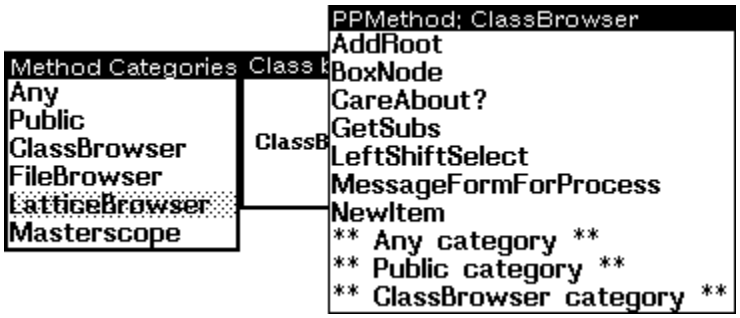
Information that is defined locally within the class is printed in the bold font. Inherited information is printed in the regular font. Inherited information from the classes **Object** and **Tofu** is not printed.

PPV! Same as **PP!**, but does not include **Methods**, as shown in the following window:

```
#,($ DestroyedObject)
MetaClass and its Properties
   Class Edited: (*
TheCollaborators:
15-Oct-84 16:23)
Supers
   (Object Tofu)
Instance Variable Descriptions
Class Variables
```

PPMethod Brings up a menu of the methods for this class followed by a list of the known categories. The menu is influenced by the shaded options on the **Method Categories** menu (see Section 10.3.1.3, "Add Category Menu"), whether or not it is opened. Selecting a category will include any methods under that category in the menu. After selecting one of the methods, the Lisp function for that method is prettyprinted.

For example, selecting **PPMethod** from the node **ClassBrowser** with the **Method Categories** menu as shown results in the following menu of methods:



MethodSummary Prints a summary of the methods defined for the class, as shown in the following window:

```
((Destroy! DestroyedObject.Destroy! args
        NIL doc NIL))
```

PrintSummary Similar to **PP**, but default values or properties associated with variables and methods are not printed, as shown in the following window:

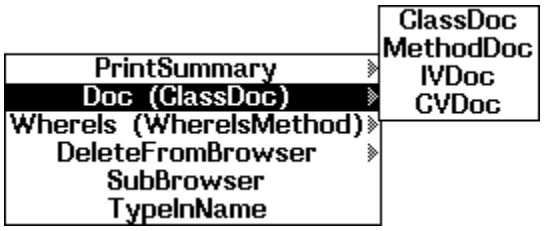
```
   #,($ DestroyedObject)
Supers
   Object
IVs


CVs


Methods
   Destroy!
```

10.3.2.2 Doc (ClassDoc) and its Suboptions

Each part of an object's definition has a **doc** property containing strings. This menu option is a quick way to see the string for a specific part of a definition. For all printing that occurs as a result of selecting this option or one of its

suboptions, the output is sent to the value of the variable **PPDefault**, which is by default the Common Lisp Executive Window.

Selecting the **Doc (ClassDoc)** option and dragging the mouse to the right causes the following submenu to appear:



ClassDoc Prints documentation for class, that is, the **doc** property.

MethodDoc Causes a menu to appear in the same manner as **PPMethod** described in Section 10.3.2.1, "PrintSummary and its Suboptions." After selecting a method, information about that method is printed, as shown in the following window:

```
class: ClassBrowser selector: BoxNode
args: NIL
doc: NIL
```

The menu of methods reappears until a selection is made from outside the menu.

IVDoc Causes a menu to appear showing instance variables associated with the class. After selecting one, its documentation is printed, as shown in the following window:

```
ClassBrowser:menus:
Cache For Saved Menus. Will Cache Menus only
 if value is T
```

The menu of instance variables reappears until a selection is made from outside the menu.

CVDoc Causes a menu to appear showing class variables associated with the class. After selecting one, its documentation is printed, as shown in the following window:

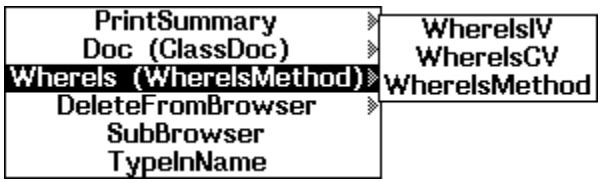
```
ClassBrowser::RightButtonItems:
Items to be done if Right button is selected
```

The menu of class variables reappears until no selection is made from the menu.

10.3.2.3 Wherels and its Suboptions

Wherels describes where each part of the object comes from. For all printing that occurs as a result of selecting this option or one of its suboptions, the output is sent to the value of the variable **PPDefault**, which is by default the Common Lisp Executive Window.

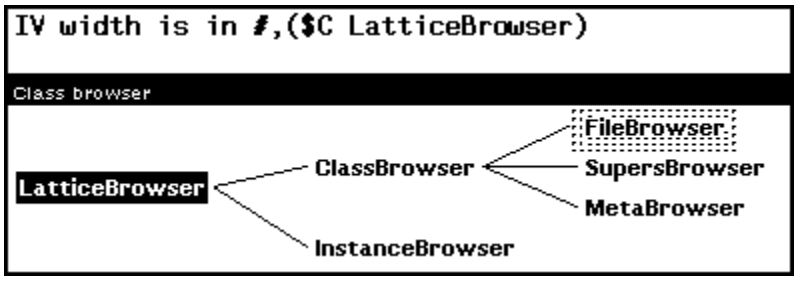
Selecting the **Wherels** option and dragging the mouse to the right causes the following submenu to appear:



WhereIsIV Causes a menu to appear at the current cursor position, showing the local and inherited instance variables. A sample menu appears here:

When you select an instance variable from this menu, these actions occur, as shown in the following window:

- A search starts with the selected class and then proceeds upwards through its supers. The first class that contains the selected instance variable flashes three times.
- All other classes in the browser that contain, that is, specialize, the instance variable are shaded.
- The name of the topmost class name is printed across the top of the window.

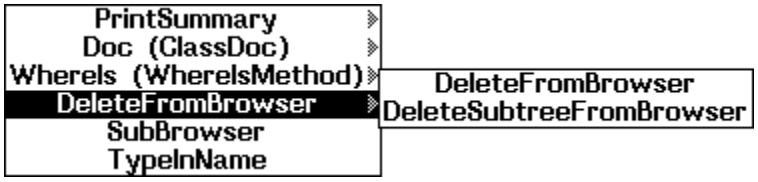


WhereIsCV Same as above, but for class variables.

WhereIsMethod Same as above, but for **Methods**. The menu of methods is not filtered by any category information.

10.3.2.4 DeleteFromBrowser and its Suboptions

Browsers show all of the lattice or tree from the root to the leaves. The **DeleteFromBrowser** option and its suboptions, shown here, allow you to prune the tree.

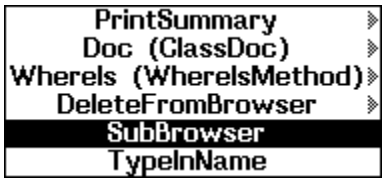


DeleteFromBrowser Deleting a class from a browser adds it to that browser's instance variable **badList**. Items listed in a browser's **badList** are not displayed by the browser. If a class on the **badList** has subclasses, these are made into roots. To redisplay a class once it has been deleted, refer to the command **RemoveFromBadList** in Section 10.3.1.2, "AddRoot and its Suboptions."

DeleteSubtreeFromBrowser Deleting a class with this command places this class on the browser's **badList**. Additionally, any subclasses of the deleted class are also placed on the **badList**. If one uses the command **RemoveFromBadList** to redisplay the deleted class, only that class is redisplayed. Its subclasses remain on the **badList** until they are explicitly removed from it.

10.3.2.5 SubBrowser

The following window shows the selection of this option:



For class and supers browsers, the SubBrowser option opens a new browser of the same type (for example, a **SubBrowser** of a supers browser is a supers browser) with the class selected becoming the root object of the browser. For file browsers, the **SubBrowser** becomes a class browser.

10.3.2.6 TypeInName

The following window shows the selection of this option:

This option puts the class name in the type-in buffer.

10.3.2.7 Extending Functionality with the Left Mouse Button

Using various keys in conjunction with the left mouse button extends the available options.

- **SHIFT** key

Pressing the **SHIFT** key while selecting a node with the left button causes the name of the class to be typed into the current type-in point. If a node is not selected, but the cursor is in the background of the browser, the entire graph is copied. This can be used to insert browser images into TEdit documents, for example. See the Lisp Library documentation on Grapher for more details.

- Control (**CTRL**) key

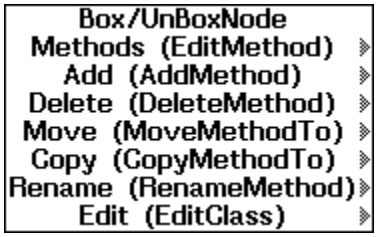
  Pressing the **CRL** key while selecting a node with the left button causes the node to track movement by the cursor. This allows you to temporarily change the layout of the nodes in the graph. The next update of the browser recomputes the node positions.

- **META** key

  Pressing the **META** key while selecting a node with the left button is the same as a **PrintSummary** selection.

10.3.3 Selecting Options in the Middle Menu

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



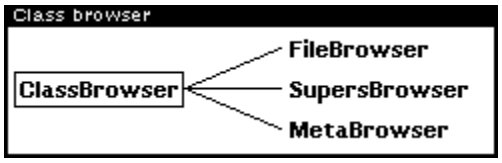
This section describes the actions that occur when you select an option from this menu.

Except for **BoxNode**, all options are followed by a parenthetical suboption. This suboption appears in the option's submenu and performs the same operation as the option itself. For example, selecting **Methods (EditMethod)** performs the same operation as selecting **EditMethod** from its submenu.

The prompt for these options usually appears in a small window at the top of the browser, unless otherwise stated.

10.3.3.1 Box/UnBoxNode

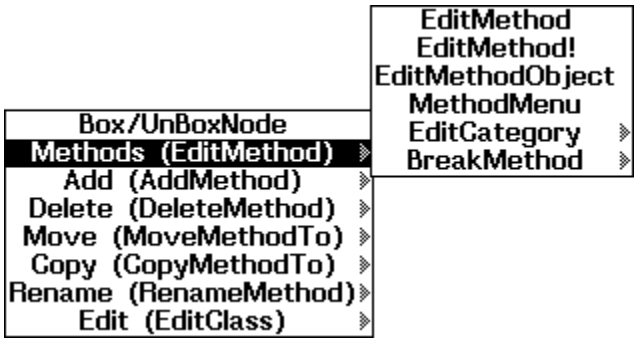
Draws a box around the node, as shown here:



This also selects the node as a target for **Move** and **Copy** options. If a node is already boxed, **Box/UnBoxNode** unboxes it. Only one node in a class browser can be boxed at a time with this menu option.

10.3.3.2 Methods (EditMethod) and its Suboptions

Selecting the **Methods (EditMethod)** option and dragging the mouse to the right causes the following submenu to appear:

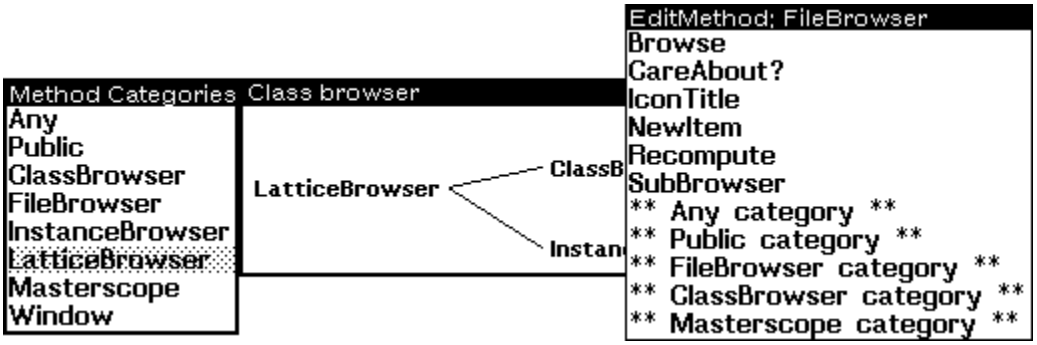


EditMethod Edits a method of class, selected from a menu of local methods. The menu that appears contains some or all of the methods of the class and some category choices. The menu options that appear depend on the shaded options on the **Method Categories** menu (see Section 10.3.1.3, "Add Category Menu").

For example, assume the following actions occur:

- A class browser is opened on the class **LatticeBrowser**.
- A category menu is added and the category **LatticeBrowser** is selected.
- **FileBrowser** under the class **ClassBrowser** is selected.
- The **EditMethod** suboption is selecte.

The following menu appears:



If you select one of the method names, an edit window appears containing the source code for the method (assuming the source code was loaded). If you select one of the categories, a similar menu appears showing the changed methods and categories. The method/category menu continues to appear until you select a method or press a mouse button outside of the menu. This operation is provided by the method **PickSelector**.

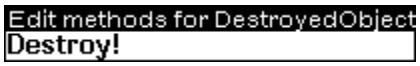
EditMethod! Edits a method selected from a menu of all the inherited methods, making the method local if necessary.

EditMethodObject Edits the object representing the method, as shown in this display editor window:

```
SEdit #,($& Method (NEW0.1Y;.;h.eN6 . 453
((className FileBrowser)
 (selector Recompute)
 (method FileBrowser.Recompute)
 (args NIL)
 (doc NIL)
 (category (LatticeBrowser)))
```

This suboption uses the method **PickSelector** to provide the same menu interface for choosing a selector that **EditMethod** uses. Within this edit window, the only items you should change are the **doc** and **category** properties.

MethodMenu Creates a permanent menu of methods of this class, for example,



After you have placed the menu, pressing the left mouse when the cursor is over a menu option will cause the method to be printed; pressing the middle button will cause the method to be edited.

EditCategory This option has three suboptions:

- **EditCategory**

  Opens a menu of available categories, similar to the following menu example:

  After you select a category, another menu appears containing the methods of that category and a list of categories not chosen (using the method **PickSelector**). When a method is selected, it appears in an edit window.

- **ChangeMethodCategory**

  Using the method **PickSelector**, this prompts you for a method. When one is selected, another menu appears containing the option ***other*** with all of the categories of the selected class.

  If one of the categories is chosen, the category for the method is changed to this value.

  If ***other*** is chosen, this prompts you in the **PROMPTWINDOW** for a symbol or a list of symbols that will become the new category or categories for the method.

  See the method **Class.ChangeMethodCategory**.

- **CategorizeMethods**

  Edits an association list of categories and the local methods they contain via the editor.

  This is an example of the edit window that appears:

```
SEdit Package: INTERLISP
((Any (CreateClass DestroyInstance New NewWithValues))
 (Public (CreateClass DestroyInstance New NewWithValues))
 (Internal NIL)
 (MetaClass (CreateClass))
 (Class (DestroyInstance New NewWithValues)))
```

BreakMethod This option has three suboptions:

- **BreakMethod**

  Places a break on a method of class, selected from a menu of local methods, by sending the message **BreakMethod**.

- **TraceMethod**

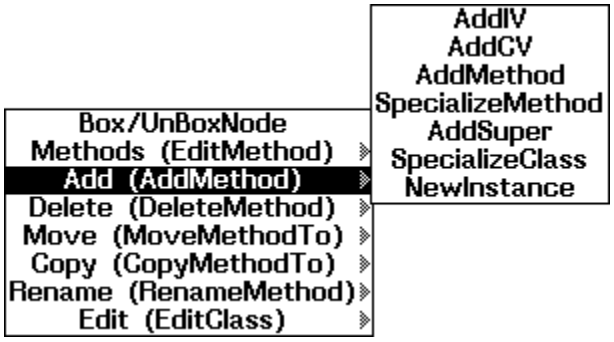
  Places a trace on a method of class, selected from a menu of local methods, by sending the message **TraceMethod**.

- **UnbreakMethod**

  Brings up a menu of methods local to class that have been broken, and removes any breaks or traces on the one selected by sending the message **UnbreakMethod**.

10.3.3.3 Add (AddMethod) and its Suboptions

Selecting the **Add (AddMethod)** option and dragging the mouse to the right causes the following submenu to appear:



AddIV Prompts you for the name of a new instance variable to be added to a class, and opens an editor as shown here:

```
* SEdit DestroyedObject Package: INTERLISP
(newIV **DefaultValue doc (* IV added by MCGILL))
```

From here, change the default value of the instance variable to the desired value, and change the documentation to add any other properties and values if necessary. When you exit from the editor, the instance variable is added to the class.

AddCV Same as **AddIV**, except that you add a class variable.

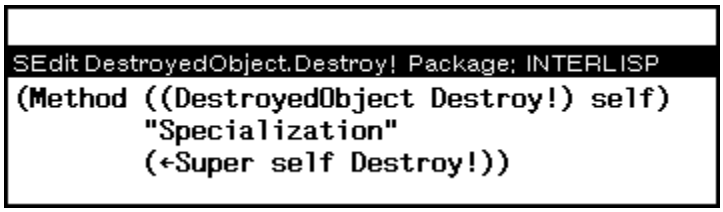
AddMethod Allows you to create and edit a new method for this class. You are prompted to type a selector for the new method. Next, an edit window appears with the following template:

Replace the form (SubclassResponsibility) with the functionality you want.

SubclassResponsibility is a macro that causes a call to **HELPCHECK,** so if you forget to remove it, or want to leave it in for debugging, it will break when the new method is invoked.

SpecializeMethod Causes a menu to appear, containing the selectors of inherited methods, the option ** **Generic Methods** **, and available categories.

If one of the selectors is chosen, a display editor window appears with a template that contains a **_Super** and the same comment as the specialized method. An example of this window appears here:



If ** **Generic methods** ** is chosen this causes a menu to appear, listing the methods inherited from **Object**, **Class**, or **Tofu**.

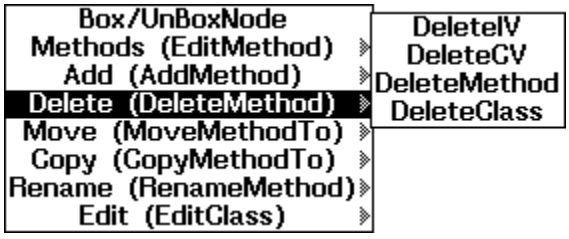
AddSuper Prompts you for a class name to be added to the beginning of the **Supers** list.

SpecializeClass Defines a subclass of this class. This subclass is initialized with no locally defined instance variables, class variables, or methods. You are prompted to give a name for the new class. The new class is added to the browser.

NewInstance Creates a new instance of this class, calls **PutSavedValue** with the new instance as an argument, and prints the instance.

10.3.3.4 Delete (DeleteMethod) and its Suboptions

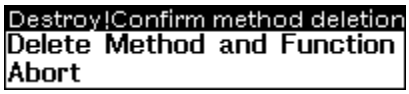
Selecting the **Delete (DeleteMethod)** option and dragging the mouse to the right causes the following submenu to appear:



DeleteIV Opens a menu containing the local instance variables, that is, those defined in the class. Selecting one removes it from the class.

DeleteCV Same as **DeleteIV**, but for class variables.

DeleteMethod Opens a menu containing the local selectors. Choosing one opens the following menu for confirmation:

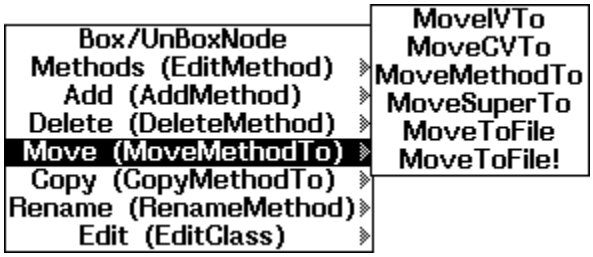


DeleteClass Deletes this class. Opens a menu similar to the following for confirmation:

If the class has no subclasses it will be deleted. If it does have subclasses a **HELPCHECK** break occurs; you can then abort or type OK to destroy the class and all of its subclasses.

10.3.3.5 Move (MoveMethodTo) and its Suboptions

Selecting the **Move (MoveMethodTo)** option and dragging the mouse to the right causes the following submenu to appear:



Moving methods and variables requires that you first have used **BoxNode** (see Section 10.3.3.1, "BoxNode") on the class which is to receive whatever is moved.

CAUTION

Moving methods and variables can have profound effects on the classes that inherit them.

MoveIVTo Opens a menu with the instance variables for the class. Choosing one causes it to be moved to the class that is boxed in the browser. When this operation is completed, the menu opens again prompting you for another instance variable to move, if desired.

MoveCVTo Similar to **MoveIVTo**, but for class variables.

MoveMethodTo Similar to **MoveIVTo**, but for methods.

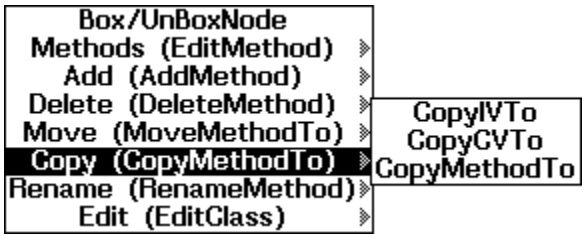
MoveSuperTo Opens a menu of the Supers for the class. Choosing one of these will cause it to be removed as a super class, and the boxed class to be added to the Supers list.

MoveToFile Opens a menu of files on **FILELST** along with ***newFile***. (See Section 10.2.1.2, "Command Summary," for more information on ***newFile***.) The class and its methods are moved to the chosen file.

MoveToFile! Same as **MoveToFile**, but includes subclasses and their methods of the classes.

10.3.3.6 Copy (CopyMethodTo) and its Suboptions

Selecting the **Copy (CopyMethodTo)** option and dragging the mouse to the right causes the following submenu to appear:



Copy options operates similarly to Move options, but leave the original method or variable in its place while adding it to the destination. You can then specialize the original or copy.

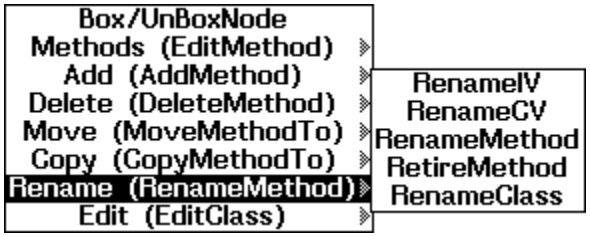
CopyIVTo Similar to **MoveIVTo,** but copies the instance variables instead of moving them.

CopyCVTo Similar to **MoveCVTo**, but copies the class variables instead of moving them.

CopyMethodTo Similar to **MoveMethodTo**, but copies the methods instead of moving them.

10.3.3.7 Rename (RenameMethod) and its Suboptions

Selecting the **Rename (RenameMethod)** option and dragging the mouse to the right causes the following submenu to appear:



RenameIV A menu appears, showing the instance variables of the class. After selecting one of these, you are prompted to give a new name for that instance variable.

RenameCV Similar to **RenameIV**, but for class variables.

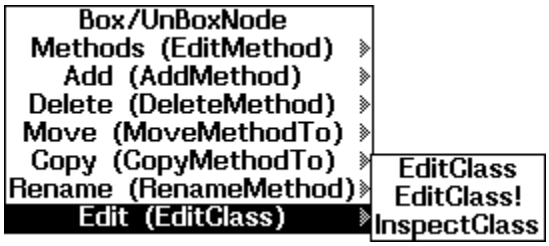
RenameMethod Similar to **RenameIV**, but for methods. You are prompted to give a new name for the selector of the method. The method function name is changed to reflect the change in the name of the selector.

RetireMethod A menu appears, showing the selectors of the class. After selecting one of these, the selector is changed by adding the prefix "Old" to it. The method function is also renamed appropriately.

RenameClass You are prompted for a new name for the class. After the new name is entered, the browser is updated to reflect the change. In addition, the method functions associated with the class are also renamed. For example, given a class **Foo** with selector **Fie**, when the class is renamed to **Fum**, the method function is automatically renamed from **Foo.Fie** to **Fum.Fie**.

10.3.3.8 Edit (EditClass) and its Suboptions

Selecting the **Edit (EditClass)** option and dragging the mouse to the right causes the following submenu to appear:



Editing options allow you to make quick, massive changes to object descriptions, and are sometimes the only menu-driven way to change certain items. See Chapter 13, Editing, for more details.

EditClass Edits the class definition of the class showing only the locally defined class variables, instance variables, and methods.

EditClass! Edits the class definition of the class showing both the locally defined and inherited class variables, instance variables, and methods. Changes to the inherited information that are done during the edit have no effect. The inherited information is included for informational purposes only. For example, you may want to copy the definition of an instance variable from the list of inherited instance variables to the list of local instance variables.

InspectClass Opens an inspector window on the class. See Chapter 18, User Input/Output Modules, for more details.

10.4 Using File Browsers

File browsers are a specialization of class browsers. In addition to the capabilities of class browsers, file browsers allow you to manipulate files. This section explains the file browser menu options not available from class browser menus, or those that have been modified from the class browser menu options. This section lists all file browser menu options; references are made to class browser menus where appropriate.

Multiple files can be associated with a file browser. Thus, one of those files can be designated as the "selected" file. There are various options as to which classes should be displayed in a file browser. See Section 10.4.1.4, "Change display mode and its Suboptions," for more information.

When a file browser is opened, the window title displays the selected file:

The icon for a shrunken file browser contains the name of the selected file, as shown here: