13. EDITING

LOOPS has an interface to the display-based editor, SEdit. This editor is most often used for modifying classes, functions, and methods, but it can also be invoked to modify instances. Instances are typically modified through the inspector interface (see Chapter 18, User Input/Output Modules).

The technique for editing methods is exactly the same as that for editing functions. LOOPS uses Method, an extension of the lambda form that is documented in Chapter 6, Methods, which also gives details about the form of methods and how to invoke the editor upon them.

The process of editing classes and instances is different from editing methods in that you are not editing structure directly. The data structures representing the objects are translated into list structures, those list structures are then edited, and, finally, on exiting from the editor, the list structure is translated back into LOOPS objects. Because of this process, changes in objects do not take effect until you have exited from the editor.

13.1 Editing Classes

The **Edit** and **Edit!** methods provide a screen-based way to modify class structure. You can quickly add and delete local class and instance variables, make inherited variables local, and change initial values. The other methods listed are used to interface LOOPS to the display editor.

Name	Type	Description
Edit	Method	Edits the structure of a class.
Edit!	Method	Edits a class in a form that includes inherited information.
InstallEditSource	Method	Makes a class conform to a description.
MakeEditSource	Method	Makes a list structure for editing a class.
MakeFullEditSource	Method	Makes a list structure, including inherited information, for editing a class.

(← *self* **Edit** *commands*) [Method of Class]

Purpose: Edits the structure of a class.

Behavior: Translates *self* from a data type to a list structure that is then passed to the editor through **EDITE** (see the *Lisp Release Notes* and the *Interlisp-D Reference Manual*). If *commands* is non-NIL, this is passed as the second argument to **EDITE**.

The variable **LASTCLASS** is bound to the class name of *self*.

Generally, *commands* is NIL, which causes you to enter the editor interactively. From this point, you can perform the following actions:

- Change the metaclass of *self*
- Add or delete class properties
- Add or delete class variables, instance variables, and their properties.

Also, as a user convenience, the edited form has a list of methods that you can select and edit, although you cannot delete items from this **MethodFns** list and have that action disassociate the methods from the class.

Arguments: *commands* A list of editing commands to be passed to EDITE.

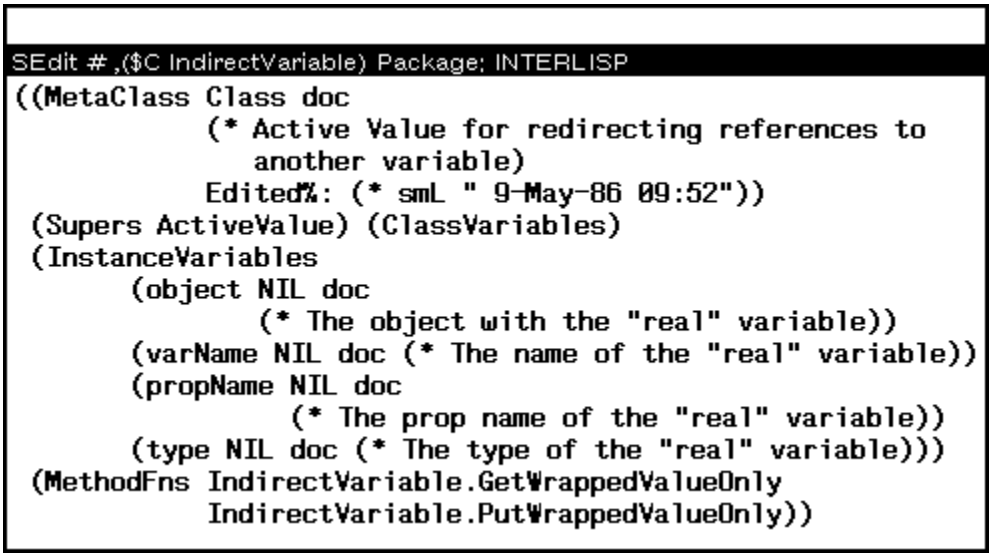
Returns: The class name of *self*.

Categories: Object

Specializes: Object

Example: The following editing window is generated as a result of

```
(← ($ IndirectVariable) Edit)
```



The following information describes this window:

- The title of the window contains the name of the class being edited and package it uses for displaying symbols. This package should be INTERLISP when using LOOPS.
- It has the metaclass **Class**.
- It has the two class properties **doc** and **Edited%:**.
- It has one super class, the class **ActiveValue**.
- It has no class variables.
- It has four instance variables: **object, varName**, **propName**, and **type**. Each instance variable has a **doc** property.
- It has two local methods: **GetWrappedValueOnly** and **PutWrappedValueOnly**.

(← *self* **Edit!** *commands*) [Method of Class]

Purpose: Edits a class in a form that includes inherited information.

Behavior: This is similar is behavior to the method **Class.Edit**.

The form you are editing includes not only items local to *self* but also class variables and instance variables that are inherited. This allows you to easily move inherited information into *self*. Editing operations that modify the inherited values have no effect.

Arguments: *commands* A list of editing commands to be passed to **EDITE**.

Returns: The class name of *self*.

Categories: Class

Example: The following command puts you into the display editor. Compare this display with the previous one.

```
(← ($ IndirectVariable) Edit!)
```

```
SEdit #,($C IndirectVariable) Package: INTERLISP
((MetaClass Class doc
            (* Active Value for redirecting references to
               another variable)
            Edited%: (* smL " 9-May-86 09:52"))
 (Supers ActiveValue) (ClassVariables) (CVsInherited)
 (InstanceVariables
       (object NIL doc
               (* The object with the "real" variable))
       (varName NIL doc (* The name of the "real" variable))
       (propName NIL doc
                 (* The prop name of the "real" variable))
       (type NIL doc (* The type of the "real" variable)))
 (IVsInherited))
```

(← *self* **InstallEditSource** *editedDescription*) [Method of Class]

Purpose: Makes a class conform to a description.

Behavior: Called by the system to change a class data structure to correspond to a list structure you have edited. If there are errors in the structure, the editor is activated again. If there are errors in the edited structure, an error message is printed in the prompt window and you are returned to the editor to fix it.

If there are no errors in the structure, this successfully translates the structure into the class data type structure. In addition, a class property **Edited:** is added to *self* with the value returned by (**EDITDATE** NIL NIL INITIALS).

Arguments: *editedDescription*
A list structure similar to that returned by the message **MakeEditSource**.

Returns: Used for side effect only.

Categories: Object

Specializes: Object

(← *self* **MakeEditSource**) [Method of Class]

Purpose: Makes a list structure for editing a class.

Behavior: This builds a list structure containing metaclass, super, class variable and instance variable information. In addition, the method function names are included in this list.

Returns: List expression of class structure.

Categories: Object

Specializes: Object

Example: The command

```
(← ($ BreakOnPutOrGet) MakeEditSource)
```

returns

```
((MetaClass Class Edited%: (* nbm  " 5-May-87 17:53")
doc "This is the default metaClass for all classes")
 (Supers BreakOnPut)
 (ClassVariables)
 (InstanceVariables)
 (MethodFns BreakOnPutOrGet.GetWrappedValue))
```

(← *self* **MakeFullEditSource**) [Method of Class]

Purpose: Makes a list structure, including inherited information, for editing a class.

Behavior: This is similar to **MakeEditSource**. The constructed list also includes instance variables and class variables that are inherited.

The list does not contain the method functions associated with self that **MakeEditSource** includes.

Returns: List expression of class structure.

Categories: Class

Example: The command

```
(← ($ BreakOnPutOrGet) MakeFullEditSource)
```

returns:

```
((MetaClass Class Edited%: (* nbm  " 5-May-87 17:53")
doc "This is the default metaClass for all classes")
 (Supers BreakOnPut)
 (ClassVariables)
 (CVsInherited)
 (InstanceVariables)
 (IVsInherited
   (localState NIL doc (* The local state of the active value))))
```


13.1 EDITING INSTANCES

13.2 Editing Instances

LOOPS instances can also be edited by the standard Medley code editor. From this editor, you can change the values of instance variables and properties, and add new instance variables. Instances follow the same basic editing protocol that classes do.

The following table shows the methods in this section.

Name	Type	Description
Edit	Method	Allows you to change the values contained in an instance.
InstallEditSource	Method	Makes an instance conform to a description.
MakeEdit	Method	Makes a list structure for editing an instance.

(← *self* **Edit** *commands*) [Method of Object]

Purpose: Allows you to change the values contained in an instance.

Behavior: Changes the data structure of *self* to a list and passes that list to **EDITE** (see the *Lisp Release Notes* and the *Interlisp-D Reference Manual*.) If *commands* is non-NIL, this is passed as the second argument to **EDITE**.

Deleting a variable does not delete it from the instance.

Arguments: *commands* A list of editing commands to be passed to **EDITE**.

Returns: *self*

Categories: Object

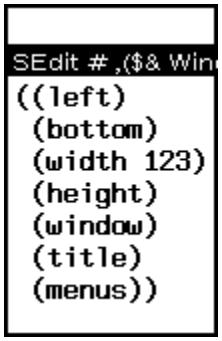
Specializations: Class

Example: The following commands create the edit window shown.

```
71←(← ($ Window) New 'w1)
#,($& Window (NEW0.1Y%:.;h.eN6 . 495))

72←(←@ ($ w1) width 123)
123

73←(←($ w1) Edit)
```



(←*self* **MakeEditSource**) [Method of Object]

Purpose: Makes a list structure for editing an instance.

Behavior: Returns a list showing all instance variables, values, and properties.

Returns: A list showing all instance variables, values, and properties.

Categories: Object

Specializations: Class

Example: The following shows **MakeEditSource** results as values are assigned to the instance variables of ($ w1).

```
38←(← ($ Window) New 'w1)
#,($& Window (NEW0.1Y%:.;h.eN6 . 495))

39←(← ($ w1) MakeEditSource)
((left) (bottom) (width) (height) (window) (title) (menus))

40←(←@ ($ w1) width 123)
123

41←(← ($ w1) MakeEditSource)
((left) (bottom) (width 123) (height) (window) (title) (menus))

42←(←@ ($ w1) menus Title T)
T

43←(← ($ w1) MakeEditSource)
((left) (bottom) (width 123) (height) (window) (title) (menus
#,NotSetValue Title T))

44←(@ ($ w1) window)
{WINDOW}#74,25554

45←(← ($ w1) MakeEditSource)
((left 31) (bottom 407) (width 123) (height 12) (window #,($AV
LispWindowAV ((NEW0.1Y%:.;h.eN6 . 495)) (localState {WINDOW}#74,25554)))
(title) (menus #,NotSetValue Title T))
```

(← *self* **InstallEditSource** *editedDescription*) [Method of Object]

Purpose: Makes an instance conform to a description.

Behavior: This is called by the system to change an instance data structure to correspond to a list structure you have edited.

Arguments: *editedDescription*
A list structure similar to that returned by the message **MakeEditSource**.

Returns: Used for side effect only.

Categories: Object

[This page intentionally left blank]