9. DATA TYPE PREDICATES AND ITERATIVE OPERATORS

This chapter describes data type predicates and an operator for iterative statements.

9.1 Data Type Predicates

Data type predicates test the Lisp data type of some datum. For example, some predicates test whether a datum is an object, instance, or class.

LOOPS defines three Lisp data types: annotatedValue, class, and instance. LOOPS provides predicates that enable testing aspects of these types.

Name	Type	Description
Object?	Macro	Determines if a particular datum is a LOOPS object.
Class?	Macro	Determines if a particular datum is a class.
Instance?	Macro	Determines if a particular datum is an instance of a class.
AnnotatedValue?	Macro	Determines if a particular datum is an instance of the annotatedValue Lisp data type.
Understands	Method	Determines if an object will respond to a message.

To determine if a particular datum has an instance variable, class variable, or a property, use **HasIV, HasIV!,** or **HasCV** (see Chapter 2, Instances, and Chapter 3, Classes). To determine if a particular datum is an instance of a class or its superclasses, use **InstOf** or **InstOf!** (see Chapter 2, Instances).

(**Object?** *X*) [Macro]

Purpose/Behavior: Determines if *X* is a LOOPS object. **Object?** returns T for both instances and classes.

Arguments: *X* Possible object.

Returns: Returns T if a name is a pointer to a LOOPS object, and returns NIL otherwise.

Example: This example demonstrates the use of **Object?**.

```
3←(← ($ Window) New 'Window1)
#,($& Window (|OZW0.1Y:.;h.Qm:| . 495))

4←(Object? ($ Window1))
T
```

```
5←(Object? ($ Window))
T

6←(Object? ($ NotAnObject))
NIL

7←(Object? 'NotAnObject)
NIL
```

(**Class?** *X*) [Macro]

Purpose/Behavior: Determines if *X* is a class.

Arguments: *X* Possible class.

Returns: Returns T if *X* is a class; returns NIL otherwise.

Example: This example demonstrates the use of the predicate **Class?** Since **($ Window)** is a class, the function returns T. Since **Window1** and **NotClass** are not class names, NIL is returned. `(Class? X)` is equivalent to `(type? class X)`.

```
8←(Class? ($ Window))
T

9←(Class? ($ NotClass))
NIL

10←(Class? ($ Window1))
NIL
```

(**Instance?** *X*) [Macro]

Purpose/Behavior: Determines if *X* is an instance of some class.

Arguments: *X* Possible instance.

Returns: Returns T if *X* is an instance; returns NIL otherwise.

Example: This example shows the use of **Instance?** `(Instance? X)` is equivalent to `(type? instance X)`.

```
11←(Instance? ($ Window1))
T

12←(Instance? 'Unbound)
NIL

13←(Instance? ($ Window))
NIL
```

(**AnnotatedValue?** *X*) [Macro]

Purpose/Behavior: Determines if *X* is an instance of the annotatedValue data type. For a complete explanation of annotated values, see Chapter 8, Active Values.

Arguments: *X* Possible annotatedValue.

Returns: Returns T if *X* is an annotated value; returns NIL otherwise.

Example: Instances of class Window are created with an active value in the window instance variable. **AnnotatedValue** returns T for the annotatedValue which "wraps" an active value, not for the active value itself.

```
100←(← ($ Window) New 'Window3]
#,($& Window (|OZW0.1Y:.;h.Qm:| . 495))

1←(GetValue ($ Window3) 'window)
{WINDOW}#51,140000

2←(GetValueOnly ($ Window3) 'window)
#,($AV LispWindowAV ((|OZW0.1Y:.;h.Qm:| . 495))
(localState {WINDOW}#51,140000))

3←(AnnotatedValue? (GetValueOnly ($ Window3) 'window))
T

4←(AnnotatedValue? (GetValue ($ Window3) 'window))
NIL

5←(AnnotatedValue? (_ ($ LispWindowAV) New 'LWAV4]
NIL
```

(← *self* **Understands** *selector*) [Method of Object]

Purpose/Behavior: Determines if the object *self* will respond to a message with *selector.*

Arguments:

self	Instance or class in question.
selector	Selector in question.

Returns: T if *self* is a class or an instance of a class that understands message selector; NIL otherwise.

Note: If *self* is not a LOOPS object, you get NIL and not an error.

Categories: Object

Example: Given that **Window** is a class, **MyWindow** is an instance, and **SpinAround** is a method of **MyWindow**, **Window** returns NIL, and **MyWindow** returns T. Since **Shape** is a method of **Window**, this also returns T.

```
90←(← ($ Window) Understands 'SpinAround)
NIL

91←(← ($ MyWindow) Understands 'SpinAround)
T

91←(← ($ MyWindow) Understands 'Shape)
T
```

9.2 Iterative Operators

LOOPS provides an iterative operator to be used with Interlisp-D iterative statements.

in-supers-of *X* [Iterative Statement Operator]

Purpose / Behavior: Allows iteration up the supers chain of the object *X*. Used in an Interlisp-D iterative statement. (See the *Interlisp-D Reference Manual* for more information on iterative statements.)

Arguments:	*X*	A LOOPS class or an instance.

Example: This example shows one way to use this operator.

```
55←(FOR I in-supers-of ($ ClassBrowser) DO (PRINT (← I ClassName]
ClassBrowser
IndexedObject
LatticeBrowser
Window
Object
Tofu
NIL
```

[This page intentionally left blank]