File created: 15-Aug-90 13:23:08 **{DSK}<usr>local>lde>SOURCES>loops>SYSTEM>LOOPSUTILITY.;2**

changes to: (VARS LOOPSUTILITYCOMS)
(FNS CalledFns)

previous date: 13-Jul-88 12:31:04 {DSK}<usr>local>lde>SOURCES>loops>SYSTEM>LOOPSUTILITY.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ LOOPSUTILITYCOMS
       (
```

;;; Indexed Objects

```
        (CLASSES IndexedObject)
        (FNS FindIndexList FindIndexedObjects IndexUid RemoveUidIndex RenameIndexList)
        (METHODS IndexedObject.ChangeClass IndexedObject.Destroy IndexedObject.NewInstance
               IndexedObject.OldInstance IndexedObject.PrintOn)
```

;;; Pattern matching functions

```
        (FNS MatchDescr MatchIVs MatchListDescr)
```

;;; User level fns for clearing caches

```
        (FNS ClearAllCaches)
        (INITVARS (ClearAllCaches NIL))
        (GLOBALVARS ClearAllCaches)
        (ADDVARS (GLOBALVARS ClearAllCaches))
```

;;; Misc fns

```
        (FNS CalledFns DELASSOC GetObjectNames MoveClassVariable MoveMethodsToFile MoveVariable RenameInClass
             RenameMethodFunction)
        (FUNCTIONS HELPCHECK)
```

;;; Misc methods

```
        (METHODS AbstractClass.New Class.Add Class.AddCV Class.AddIV Class.AllInstances! Class.BreakMethod
               Class.CommentMethods Class.Copy Class.CopyCV Class.CopyIV Class.CopyMethod Class.Delete
               Class.Destroy Class.Destroy! Class.DestroyClass Class.DestroyInstance Class.HasCV Class.HasIV
               Class.HasIV! Class.MakeLocalMethod Class.MethodDoc Class.MethodSummary Class.MoveMethod
               Class.OnFile Class.PP Class.PP! Class.PPM Class.PPMethod Class.PPV! Class.PrintSummary Class.Put
               Class.UnbreakMethod Object.AddIV Object.CopyDeep Object.CopyShallow Object.Get Object.HasCV
               Object.HasIV Object.DeleteIV Object.Inspect Object.InstOf Object.InstOf! Object.OnFile Object.PP
               Object.PP! Object.PPIVs Object.PPV! Object.PrintOn Object.Put Object.Sublis Object.Understands
               Object.VirtualCopy? Object.WhereIs)))
```

;;; Indexed Objects

```
(DEFCLASSES IndexedObject)

(DEFCLASS IndexedObject (MetaClass Class doc "This class keeps track of all its instances, both when they are
                                created, and when instances are read in" Edited%:
                                                                    (* dgb%: "28-May-84 10:03"))
       (Supers Object)
       (ClassVariables (IdentifierVar shortName doc "The name of an instance variable which will contain a
                               string which could provide some identification to the user Used in PrintOn if
                               variable is in object and filled. shortName is the default variable name which is
                               used.")))

(DEFINEQ
```

(**FindIndexList**

```
  [LAMBDA (key varName)                                             (* smL "13-May-86 14:05")

          (* * Find uid list under key)

    (OR varName (SETQ varName 'UIDINDEX))
    (CDR (FASSOC key (OR (LISTP (EVALV varName))
                         (SETTOPVAL varName (LIST (CONS key])
```

(**FindIndexedObjects**

```
  [LAMBDA (key varName)                                          (* dgb%: "25-Apr-84 15:58")
    (MAPCAR (FindIndexList key varName)
           (FUNCTION $!])
```

(IndexUid

```
  [LAMBDA (key uid varName)                                      (* dgb%: "16-May-84 20:20")
                                                                 (* Add uid to list under key (no check to see if it is a UID))
    (OR varName (SETQ varName 'UIDINDEX))
    [PROG [oldValue (indexList (OR (LISTP (EVALV varName))
                                   (SETTOPVAL varName (LIST (CONS key]
          (SETQ oldValue (CDR (FASSOC key indexList)))
          (COND
             ((NOT (MEMBER uid oldValue))
              (PUTASSOC key (CONS uid oldValue)
                     indexList]
    uid])
```

(RemoveUidIndex

```
  [LAMBDA (key uid varName)                                      (* smL "29-Sep-86 14:16")

          (* * Remove uid from list under key)

    (LET* [(varName (OR varName 'UIDINDEX))
           (keyList (FASSOC key (OR (LISTP (EVALV varName))
                                    (SETTOPVAL varName (LIST (CONS key]
          (AND keyList (RPLACD keyList (REMOVE uid (CDR keyList])
```

(RenameIndexList

```
  [LAMBDA (oldKey newKey varName)                                (* smL "10-Jun-86 10:19")

          (* * Change the oldKey on the index list to the newKey)

    (LET* [(varName (OR varName 'UIDINDEX))
           (indexList (FASSOC oldKey (OR (LISTP (EVALV varName))
                                         (SETTOPVAL varName (LIST (CONS oldKey]
          (if indexList
              then (change (CAR indexList)
                           newKey])

)

(\BatchMethodDefs)

(METH IndexedObject ChangeClass (newClass)
      "Remove the index and then let the super do its work"
      (category (Object)))

(METH IndexedObject Destroy NIL "Unindex before destroying" (category (Object)))

(METH IndexedObject NewInstance (name arg1 arg2 arg3 arg4 arg5)
      "Index newly created object"
      (category (Object)))

(METH IndexedObject OldInstance (arg1 arg2 arg3 arg4 arg5)
      "Index old object"
      (category (Object)))

(METH IndexedObject PrintOn NIL "Make print form with identification on file, including information specified by
      identifierVar" (category (Object)))

(Method ((IndexedObject ChangeClass) self newClass)              ; dgb: 16-May-84 20:19
   "Remove the index and then let the super do its work"
   (RemoveUidIndex (ClassName self)
          (UID self))
   (_Super
    self ChangeClass newClass))

(Method ((IndexedObject Destroy) self)                           ; dgb: 28-Nov-84 18:59
   "Unindex before destroying"
   (RemoveUidIndex (ClassName self)
          (UID self))
   (_Super
    self Destroy))

(Method ((IndexedObject NewInstance) self name arg1 arg2 arg3 arg4 arg5)
                                                                 ; dgb: 25-Apr-84 15:56
   "Index newly created object"
   (_Super
    self NewInstance name arg1 arg2 arg3 arg4 arg5)
   (IndexUid (ClassName self)
          (UID self))
   self)

(Method ((IndexedObject OldInstance) self arg1 arg2 arg3 arg4 arg5)
```

```
                                                                    ; dgb: 30-Apr-84 09:20
   "Index old object"
   (_Super
    self OldInstance arg1 arg2 arg3 arg4 arg5)
   (IndexUid (ClassName self)
          (UID self))
   self)

(Method ((IndexedObject PrintOn) self)                              ; smL  7-Feb-86 10:56
   "Make print form with identification on file, including information specified by identifierVar"
   [LET (id (iv (@ |::IdentifierVar|)))
        (COND
           [[AND (NULL (GetObjectName self))
                 (SETQ id (AND (_ self HasIV iv)
                               (@ !\iv]
            (LIST (CONCAT LoopsReadMacroChar ",")
                  '$&
                  (ClassName self)
                  (LIST id (UID self]
            (T (_Super])

(\UnbatchMethodDefs)


;;; Pattern matching functions

(DEFINEQ

(MatchDescr
  [LAMBDA (self description alist)                                  (* dgb%: "21-MAR-83 10:51")

          (* * match against a description. A List beginning with A is thought of as a description of an object.
          The next atom is the name of its class (or a superClass)%. Additional list elements are thought of as describing IVs, except
          for (=%: name descr) which defines this name as being the object just specified;
          and (= name) which mean that the object matched is eq to the named object.
          An IV description is (ivName ivValueDescription prop1 prop1Desc |...|) RETURNS the alist or NIL.)

    [OR alist (SETQ alist (CONS (CONS NIL self]                     (* So that when alist is returned it is not NIL)
    (AND (COND
            ((NULL description)                                     (* NIL matches anything)
             T)
            ((STRINGP description)                                  (* Test strings are a substring of self)
             (STRPOS description self))
            ((Object? description)                                  (* Objects which are descriptions follow a protocol for matching)
             (SETQ alist (_ description Match self)))
            ((LISTP description)
             (SETQ alist (MatchListDescr self description alist)))
            ((type? instance self)

          (* description had better be a LITATOM, and the name of a class.
          Everything else returns NIL)

               (_ self InstOf! description))
            (T                                                      (* self and description are two atoms, or two datatypes)
               (EQUAL self description)))
         alist])

(MatchIVs
  [LAMBDA (self ivDescrs alist)                                     (* dgb%: "21-MAR-83 10:53")

          (* Match each of the instance variable descriptions of self against the description in the list of ivDescrs.
          Each description there is of the form -
          (ivName ivValue propName1 propValue |...|) -
          Extra props on the iv, and ivs not mentioned are ignored)

    (for ivDescr ivName ivProp val in ivDescrs finally (RETURN alist)
       do [SETQ val (GetIVHere self (SETQ ivName (CAR ivDescr]
          (OR (SETQ alist (MatchDescr val (CADR ivDescr)
                                  alist))
              (RETURN NIL))
          (for ivTail on (CDDR ivDescr) do (OR (SETQ alist (MatchDescr (GetValue self ivName (CAR ivTail))
                                                                       (CADR ivTail)
                                                                       alist))
                                               (RETURN NIL])

(MatchListDescr
  [LAMBDA (self description alist)                                  (* dgb%: "21-MAR-83 11:45")
    (AND [SELECTQ (CAR description)
                  (=%:                                              (* Remember this item as named by CADR -
                                                                    then match to CADDR)
                       (SETQ alist (MatchDescr self (CADDR description)
                                               (CONS (CONS (CADR description)
                                                           self)
                                                     alist))))
                  (=                                                (* The same as a previosuly named item)
```

```
                (EQ self (CDR (FASSOC (CADR description)
                                      alist))))
            ((TEST Test test)                                    (* Force a functional test.)
                 (APPLY* (CADR description)
                         self))
            (EVAL                                                (* Evaluate the form and then use it for matching)
                  (SETQ alist (MatchDescr self (EVAL (CADR description))
                                          alist)))
            ((a an An A)                                         (* A class type, or for non objects, a test)
                 [COND
                    [(type? instance self)                       (* Match class and IV descriptions)
                     (AND (_ self InstOf! (CADR description))
                          (SETQ alist (MatchIVs self (CDDR description)
                                                alist]
                    (T (AND (GETD (CADR description))
                            (APPLY* (CADR description)
                                    self])
            ((NOT Not not)                                       (* Matches if description does not match)
                 (NOT (MatchDescr self (CADR description)
                                  alist)))
            ((OR Or or)                                          (* Matches if one matches. Returns alist from that match.)
                 [for descr al in (CDR description) do (AND (SETQ al (MatchDescr self descr alist))
                                                            (RETURN (SETQ alist al])
            ((AND And and)                                       (* Matches if all match -- probably not needed)
                 (for descr in (CDR description) finally (RETURN T) do (OR (SETQ alist (MatchDescr self descr alist
                                                                                                  ))
                                                                            (RETURN NIL))))
            (QUOTE                                               (* Quoted expressions must be EQUAL to object)
                  (EQUAL self (CADR description)))
            (COND
               ((LISTP self)                                     (* Recursive match)
                (AND (EQ (LENGTH description)
                         (LENGTH self))
                     (for descr in description as obj in self finally (RETURN T)
                        do (OR (SETQ alist (MatchDescr obj descr alist))
                               (RETURN NIL]
        alist])

)

;;; User level fns for clearing caches

(DEFINEQ

(ClearAllCaches
  [LAMBDA NIL                                                    (* smL " 5-Jun-86 12:57")

          (* * Clear all caches used by LOOPS)

    (CLRHASH CLISPARRAY)
    (FlushMethodCache)
    (FlushIVIndexCache)
    [for window in (OPENWINDOWS) bind loopsWindow expandedWindow
       do (if (SETQ loopsWindow (WINDOWPROP window 'LoopsWindow))
              then (_ loopsWindow ClearMenuCache))
          (if (SETQ expandedWindow (WINDOWPROP window 'ICONFOR))
              then (if (SETQ loopsWindow (WINDOWPROP expandedWindow 'LoopsWindow))
                       then (_ loopsWindow ClearMenuCache]
    (CLRHASH CLISPARRAY)
    (FlushMethodCache)
    (FlushIVIndexCache)
    (for form in ClearAllCaches do (EVAL form))
    (CLRHASH CLISPARRAY)
    (FlushMethodCache)
    (FlushIVIndexCache])

)

(RPAQ? ClearAllCaches NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS ClearAllCaches)
)

(ADDTOVAR GLOBALVARS ClearAllCaches)

;;; Misc fns

(DEFINEQ

(CalledFns
  [LAMBDA (classes definedFlg)                                   ; Edited 15-Aug-90 13:21 by jds

          (* * Finds names of all functions called from a set of classes. If definedFlg =NIL then gets all fns.
          If =T then defined fns. If =1 then undefined fns)
```

```
    [COND
       ((LITATOM classes)
        (SETQ classes (LIST classes]
    (for className in classes bind fns do (for fn in (\ListFromBlock (fetch (class methods) of (GetClassRec
                                                                                                  className)))
                                             do (AND [OR (NULL definedFlg)
                                                         (AND (EQ definedFlg T)
                                                              (FNTYP fn))
                                                         (AND (EQ definedFlg 1)
                                                              (NOT (FNTYP fn]
                                                    (pushnew fns fn)))
       finally (RETURN (SORT fns])
```

(DELASSOC

```
  [LAMBDA (key alist)                                          (* dgb%: "11-NOV-82 03:58")
    (for P in alist when (NOT (EQ key (CAR P))) collect P])
```

(GetObjectNames

```
  [LAMBDA (object)                                             (* smL " 2-Apr-86 15:04")
                                                               (* Returns the names of an object including its UID)
    (LET ((uid (LIST (UID object)))
          (names (GETHASH object ObjNameTable)))
         (COND
            ((NULL names)
             uid)
            ((LITATOM names)
             (CONS names uid))
            (T (APPEND names uid])
```

(MoveClassVariable

```
  [LAMBDA (oldClassName newClassName varName)                  (* smL " 6-May-86 14:38")

          (* * Moves a class variable and its properties to a new class and deletes it from the old class.)

    (LET [(oldClass (GetClassRec (GoodClassName oldClassName NIL T)))
          (newClass (GetClassRec (GoodClassName newClassName NIL T]
         (COND
            ([NOT (FMEMB varName (_ oldClass ListAttribute 'CVs]
             (HELPCHECK varName "is not a CV of" oldClass "so cannot be moved from there")))
         (AddCV newClass varName (GetClassValue oldClass vaName))
         (for propName in (_ oldClass ListAttribute 'CVPROPS varName) do (PutClassValue newClass varName
                                                                                        (GetClassValue oldClass varName
                                                                                               propName)
                                                                                        propName))
         (DeleteCV oldClass varName])
```

(MoveMethodsToFile

```
  [LAMBDA (file)                                               (* smL " 9-Apr-87 13:59")

          (* * Move a method to this file if it has the same name as a function on this file)

    (for FN in (FILEFNSLST file) bind obj (MYMETHODS _ (FILECOMSLST file 'METHODS))
       do (AND (SETQ obj (GetObjectRec FN))
               (_ obj InstOf! 'Method)
               (NOT (FMEMB FN MYMETHODS))
               (MOVEITEM file FN 'METHODS)
               (SETQ $$VAL T])
```

(MoveVariable

```
  [LAMBDA (oldClassName newClassName varName)                  (* smL "11-Apr-86 14:51")

          (* * Moves an instance variable and it description to new class, deleting it from old)

    (SETQ oldClassName (GoodClassName oldClassName NIL T))
    (SETQ newClassName (GoodClassName newClassName NIL T))
    (PROG (descr (oldC (GetClassRec oldClassName))
                 (newC (GetClassRec newClassName)))
          [COND
             ([NOT (FMEMB varName (_ oldC ListAttribute 'IVs]
              (ERROR varName (CONCAT "not a local IV of " oldC " so can not be moved from there"]
          (SETQ descr (FetchCIVDescr oldC varName))
          (AddCIV newC varName (CAR descr)
                 (CDR descr))
          (DeleteCIV oldC varName)
          (RETURN newC])
```

(RenameInClass

```
  [LAMBDA (className name newName place prop)                  (* dgb%: "30-NOV-81 13:36")

          (* * This is a general function for adding information to a class. It does this by modifying the tsource for the class.
          Called by DefineMethod DM and others of that ilk. place is one of Methods, InstanceVariables, etc prop is an optional
```

property name for variables e.g. CONSTRAINTS for the otherSlotDescription.
name is which one to add to e.g varName and newName is what is the primary associated quantity e.g.
default for variable or implementing function for Methods)

```
    (PROG (propForm valForm (source (GetClassSource className)))
          (COND
             ((NULL source)
              (ERROR className "not a defined class"))
             ([NULL (SETQ propForm (FASSOC (OR place 'InstanceVariables)
                                           (CDDR source]
              (ERROR place "not part of class definition"))
             (prop (SELECTQ place
                       (Supers (ERROR prop "is not available for this aspect of definition"))
                       NIL)))
          [SELECTQ place
              (MetaClass (COND
                             ((NULL prop)
                              (COND
                                 ((EQ (CADR propForm)
                                      name)
                                  (RPLACA (CDR propForm)
                                          newName)))
                              (GO OUT))
                             (T (SETQ valForm propForm))))
              (Supers (RPLACD propForm (DSUBST newName name (CDR propForm)))
                      (GO OUT))
              (SETQ valForm (FASSOC name (CDR propForm]
          [COND
             ((NULL prop)
              (RPLACA valForm newName))
             (T (for X on (CDDR valForm) by (CDDR X) do (COND
                                                           ((EQ (CAR X)
                                                                prop)
                                                            (RPLACA X newName)
                                                            (RETURN X]
      OUT (EVAL source])
```

(RenameMethodFunction

```
  [LAMBDA (className oldFnName newFnName)                    (* dgb%: "28-APR-83 18:40")
```

(* * Renames a function used as a method in className)

```
    (SETQ className (GoodClassName className NIL T))
    (PROG ((classRec (GetClassRec className))
           index)
          [COND
             ((NULL (SETQ index (FindSelectorIndex classRec oldFnName)))
              (ERROR oldFnName (CONCAT "not used as function in " className]
          (RENAME oldFnName newFnName NIL (WHEREIS oldFnName))
          (AddMethod classRec (GetNthMethod classRec index)
                 newFnName)
          (RETURN newFnName])

)


(CL:DEFUN HELPCHECK (&REST messages)
   [CL:APPLY #'CL:BREAK `("~&~S:~@{ ~A~}" HELPCHECK ,@messages])
```

;;; Misc methods

```
(\BatchMethodDefs)

(METH AbstractClass New NIL "Stop abstract class from being instantiated" (category (Class)))

(METH Class Add (type name value prop)
      "Type is one of IV IVPROP CV CVPROP META METHOD or SUPER - Adds the specified type to the class"
      (category (Class)))

(METH Class AddCV (varName newValue)
      "Adds a class variable with given newValue. Returns NIL if variable already is in class -- though it does
      change the value to newValue. Returns varName if variable was added"
      (category (Class)))

(METH Class AddIV (varName defaultValue otherProps)
      "Add a local IV to a class"
      (category (Object)))

(METH Class AllInstances! NIL "Get all objects indexed or known by UID which are of this class or any of its
      subclasses" (category (Class)))

(METH Class BreakMethod (selector)
      "Break selected method, or give choice if selector is NIL"
      (category (Class)))
```

```
(METH Class CommentMethods (selectors useDefaultComment)
      "Add args as property of method. Insert comment in class from function."
      (category (Class)))

(METH Class Copy (name)
      "Create a new class that is a copy of some existing class. Copies the variables and RuleSets. self is
      the class being copied."
      (category (Class)))

(METH Class CopyCV (cvName toClass)
      "Copy the cvName and properties to toClass."
      (category (Class)))

(METH Class CopyIV (ivName toClass)
      "Copy the ivName and properties to toClass."
      (category (Class)))

(METH Class CopyMethod (selector newClass newSelector)
      "Copy method from self to newClass. newSelector defaults to selector"
      (category (Class)))

(METH Class Delete (type name prop)
      "Deletes the specified type from class. type is one of IV IVPROP CV CVPROP META METHOD"
      (category (Class)))

(METH Class Destroy NIL "Destroys (deletes) a class by putting NIL as localRecord of entity." (category (Object)
                                                                                                        ))

(METH Class Destroy! NIL "Recursive version of Destroy. Destroys class and its subclasses." (category (Object)))

(METH Class DestroyClass (classToDestroy)
      "Destroy the class specified by smashing its contents"
      (category (Class)))

(METH Class DestroyInstance (instance)
      "smash the list of vars and var descriptions"
      (category (Class)))

(METH Class HasCV (CVName prop)
      "Tests if class has the specified class variable"
      (category (Object)))

(METH Class HasIV (IVName prop)
      "Tests if class has the specified instance variable"
      (category (Object)))

(METH Class HasIV! (IVName prop)
      "Tests if class has the specified instance variable"
      (category (Class)))

(METH Class MakeLocalMethod (selector dontComplain?)
      "make an inherited method be local"
      (category (Class)))

(METH Class MethodDoc (selector)
      "Show documentation for method in PPDefault window"
      (category (Class)))

(METH Class MethodSummary (dontPrintFlg printFile)
      "prints a summary of the methods in a class"
      (category (Class)))

(METH Class MoveMethod (newClass selector)
      "Move method specified by selector from this class to newClass"
      (category (Class)))

(METH Class OnFile (file)
      "See if class is on given file. Returns file if none given"
      (category (Class)))

(METH Class PP (file)
      "Prettyprint the class."
      (category (Object)))

(METH Class PP! (file)
      "Method for prettyPrinting self at all levels."
      (category (Object)))

(METH Class PPM (selector)
      "Pretty-print the named method for this class"
      (category (Class)))

(METH Class PPMethod (selector)
      "Prettyprint the function which implements selector in this class."
      (category (Class)))

(METH Class PPV! (file)
```

```
      "Method for prettyPrinting a self at all levels."
      (category (Object)))

(METH Class PrintSummary (file)
      "Print a summary of elements of class"
      (category (Class)))

(METH Class Put (type name value prop)
      "Type is one of IV IVPROP CV CVPROP META METHOD. Adds the specified type to the class"
      (category (Object)))

(METH Class UnbreakMethod (selector)
      "Break selected method, or give choice if selector is NIL"
      (category (Class)))

(METH Object AddIV (name value prop)
      "Make sure that self has an IV (or IVProp) by the given name, with the given value. If there is no such
       IV, add one to self."
      (category (Object)))

(METH Object CopyDeep (newObjAlist)
      "Copies the unit, sharing the iName list, copying instances, activeValues and lists - newObjAlist is an
       association list of old copied objects with there associated copies. Used to allow copying of
      circular structures."
      (category (Object)))

(METH Object CopyShallow NIL "Create a copy of self w/o initialization" (category (Object)))

(METH Object Get (varName propName)
      "Method indirection for GetValue"
      (category (Object)))

(METH Object HasCV (cvName prop)
      "Ask class"
      (category (Object)))

(METH Object HasIV (ivName prop)
      "Check to see if IV is on self. If so, return T"
      (category (Object)))

(METH Object DeleteIV (varName propName)
      "Delete the named IV from self"
      (category (Object)))

(METH Object Inspect (INSPECTLOC)
      "Inspect the object as class or instance, using INSPECTLOC as a region for the inspect window, if it is
       given"
      (category (Object)))

(METH Object InstOf (class)
      "test if self is an instance of class"
      (category (Object)))

(METH Object InstOf! (class)
      "Recursive version of InstOf."
      (category (Object)))

(METH Object OnFile (file)
      "See if an instance is on given file. Returns file if none given"
      (category (Object)))

(METH Object PP (file)
      "Pretty print an instance on a file"
      (category (Object)))

(METH Object PP! (file)
      "Method for prettyPrinting a non-class object at all levels"
      (category (Object)))

(METH Object PPIVs (file)
      "Called to prettyPrint a class definition on a file by the FILEPKGTYPE CLASSES and by the PP: method in
       Class"
      (category (Object)))

(METH Object PPV! (file)
      "Method for prettyPrinting a non-class object at all levels."
      (category (Object)))

(METH Object PrintOn (file)
      "This is the default printing function for object. It returns a form whose CAR is PRIN1ed and elements of
      CDR PRIN2ed."
      (category (Object)))

(METH Object Put (varName newValue propName)
      "Method indirection for PutValue"
      (category (Object)))
```

```
(METH Object Sublis (alist)
      "Copy this instance, substituting as specified by alist"
      (category (Object)))

(METH Object Understands (selector)
      "Tests if object will respond to selector"
      (category (Object)))

(METH Object VirtualCopy? NIL "no I'm not a virtual copy" (category (Object)))

(METH Object WhereIs (name type propName)
      "Searches the supers hierarchy until it finds the class from which type is inherited. type=NIL is
      METHODS"
      (category (Object)))

(Method ((AbstractClass New) self)                                    ; mjs: 21-JUL-82 09:26
   "Stop abstract class from being instantiated"
   (ERROR self "Abstract class cannot be instantiated"))

(Method ((Class Add) self type name value prop)                       ; smL  8-Apr-87 17:45
   "Type is one of IV IVPROP CV CVPROP META METHOD or SUPER - Adds the specified type to the class"
   (OR name (ERROR "You must specify a name to add a " type))
   [LET [(type (U-CASE type))
         (ERROR.MSG (CONSTANT (CONCAT "is not an alterable property of class." (CHARACTER (CHARCODE EOL))
                                      "Use one of" " " "IV (or IVPROP)" ", " "CV (or CVPROP)" ", " "CLASS (or
                                      METACLASS or META)" ", " "METHOD (or SELECTOR)" ", " "SUPER"]
        (DECLARE (LOCALVARS ERROR.MSG))
        (COND
           (prop                                                      (* here if property name given. Value is taken to be property
                                                                      value.)
                 (SELECTQ type
                      ((IVPROP IV NIL)
                           (PutCIVHere self name value prop))
                      ((CVPROP CV)
                           (PutClassValueOnly self name value prop))
                      ((META METACLASS CLASS)
                           (PutClassOnly self name value))
                      ((METHOD SELECTOR)
                           (PutMethod self name value prop))
                      (SUPER (ERROR "Adding a super is incompatible with a supplied prop"))
                      (ERROR type ERROR.MSG)))
           (T                                                         (* here if no property name given.
                                                                      Taken as regular value.)
              (SELECTQ type
                   (IV (PutCIVHere self name value))
                   (CV (AddCV self name value))
                   ((META METACLASS)
                        (PutClass self value))
                   ((METHOD SELECTOR)
                        (AddMethod self name value))
                   (SUPER (InstallSupers self (CONS name (GetSourceSupers self))))
                   ((CVPROP IVPROP)
                        (ERROR "Property name not supplied"))
                   (ERROR type ERROR.MSG])

(Method ((Class AddCV) self varName newValue)                         ; dgb:  9-NOV-83 11:15
   "Adds a class variable with given newValue. Returns NIL if variable already is in class -- though it does
   change the value to newValue. Returns varName if variable was added"
   (AddCV self varName newValue))

(Method ((Class AddIV) self varName defaultValue otherProps)          ; dgb:  9-NOV-83 11:17
   "Add a local IV to a class"
   (AddCIV self varName defaultValue otherProps))

(Method ((Class AllInstances!) self)                                  ; smL 11-Apr-86 15:01
   "Get all objects indexed or known by UID which are of this class or any of its subclasses"
   (for C in (CONS self (_ self ListAttribute! 'Subs)) join (_ ($! C)
                                                            AllInstances)))

(Method ((Class BreakMethod) self selector)                           ; smL  8-Apr-87 19:38
   "Break selected method, or give choice if selector is NIL"
   [AND (NULL selector)
        (SETQ selector (_ self PickSelector (CONCAT "BreakMethod: " (ClassName self))
                          NIL
                          (LDIFFERENCE (_ self ListAttribute 'Selectors)
                                 (_ self SelectorsWithBreak]
   (COND
      [selector (PROMPTPRINT (CONS 'Breaking (APPLY* 'BREAK (OR (FetchMethod self selector)
                                                                (ERROR selector (CONCAT " not found in "
                                                                                        (ClassName self]
      (T 'NothingBroken)))

(Method ((Class CommentMethods) class selectors useDefaultComment)
                                                                      ; smL 29-Sep-86 20:15
   "Add args as property of method. Insert comment in class from function."
   (SETQ class (GetClassRec class))
   [for sel inside (OR selectors (_ class ListAttribute 'Selectors)) bind method-name def comment
```

```
      eachtime (SETQ method-name (MethName class sel))
            [SETQ def (AND (FindLocalMethod class sel)
                           (OR (GETDEF method-name 'METHOD-FNS 'CURRENT '(NOERROR NOCOPY))
                               (GETDEF method-name 'METHOD-FNS 'SAVED '(NOERROR NOCOPY]
      when def do ;; Get the comment for the method.  --- If there is no comment, use the value of DefaultComment if useDefaultComment is T.  ---
                  ;; If there is still no comment, force the user to supply a comment.
                  (CL:MULTIPLE-VALUE-BIND (class-name selector args decls forms doc qualifiers type)
                      (PARSE-METHOD-BODY def)
                    (COND
                       (doc                                          ; Already commented
                            (SETQ comment doc))
                       [(AND useDefaultComment DefaultComment)       ; Insert DefaultComment in Method
                        (SETQ comment DefaultComment)
                        (PUTDEF method-name 'METHOD-FNS `(,(CAR def)
                                                          ,@qualifiers
                                                          ,args
                                                          ,comment
                                                          ,@forms]
                       (T (CL:WARN "No doc string in method ~A" method-name)))
                    [COND
                       ([NOT (EQUAL comment (GetMethod class sel 'doc]
                        (PutMethod class sel comment 'doc]           ; Put the arg list (without self) in the method object
                    (PutMethod class sel (CDR args)
                           'args))])

(Method ((Class Copy) self name)                                     ; smL 11-Apr-86 14:41
   "Create a new class that is a copy of some existing class. Copies the variables and RuleSets. self is
   the class being copied."
   (PROG (newClass supers ivs cvs oldRuleSetName oldRuleSet newRuleSetName newRuleSet)
         [COND
            ((NULL name)                                             (* Prompt for name if needed.)
             (SETQ name (MKATOM (PROMPTFORWORD "Name of new class: "]
                                                                     (* Make the new class.)
         (SETQ supers (_ self ListAttribute 'Supers))
         (SETQ newClass (_ (Class self)
                           New name supers))
         (for iv in (_ self ListAttribute 'IVS) do (_ self CopyIV iv newClass))
         (for cv in (_ self ListAttribute 'CVS) do (_ self CopyCV cv newClass))
         (for selector in (_ self ListAttribute 'Methods) do (_ self CopyMethod selector newClass selector))
         (RETURN newClass)))

(Method ((Class CopyCV) self cvName toClass)                         ; smL 11-Apr-86 14:41
   "Copy the cvName and properties to toClass."
   (_ toClass Add 'CV cvName (COPY (GetClassValueOnly self cvName)))
   (for prop in (_ self ListAttribute 'CV cvName) do (PutClassValueOnly toClass cvName
                                                            (COPY (GetClassValueOnly self cvName prop))
                                                            prop)))

(Method ((Class CopyIV) self ivName toClass)                         ; smL 11-Apr-86 14:41
   "Copy the ivName and properties to toClass."
   (_ toClass Add 'IV ivName (COPY (GetClassIV self ivName)))
   (for prop in (_ self ListAttribute 'IV ivName) do (PutClassIV toClass ivName (COPY (GetClassIV self ivName
                                                                                              prop))
                                                            prop)))

(Method ((Class CopyMethod) self selector newClass newSelector)
                                                                     ; smL 15-Jan-87 16:26
   "Copy method from self to newClass. newSelector defaults to selector"
   (OR newSelector (SETQ newSelector selector))
   (LET (def newFn (myMethName (FindLocalMethod (GetClassRec self)
                                    selector)))
        [COND
           ((NULL myMethName)
            (COND
               ((SETQ myMethName (FetchMethod self selector))
                (HELPCHECK selector " is not local for " self "
                       To copy anyway, type OK"))
               (T (ERROR selector (CONCAT "is not a selector for " self]
        [OR (type? class newClass)
            (SETQ newClass (OR (GetClassRec newClass)
                               (AND (HELPCHECK newClass " is not a class. Type OK to use oldClass: " self)
                                    self]                            ; Define the method
        (OR (SETQ def (GETDEF myMethName 'METHOD-FNS))
            (ERROR myMethName " is not a defined function"))
        [SETQ newFn (EVAL (CL:MULTIPLE-VALUE-BIND (cname sel args decls forms doc quals type)
                              (PARSE-METHOD-BODY def)
                            (PACK-METHOD-BODY (ClassName newClass)
                                  newSelector args decls forms doc quals type))]
        (for prop in (DREMOVE 'RuleSet (_ self ListAttribute 'Method selector))
           do                                                        ; Copy all the properties
              (PutMethod newClass newSelector (GetMethod self selector prop)
                     prop))
        newFn))

(Method ((Class Delete) self type name prop)                         ; dgb: 27-AUG-82 12:57
   "Deletes the specified type from class. type is one of IV IVPROP CV CVPROP META METHOD"
```

```
   (OR name (ERROR "You must specify a name to delete a " type))
   (SELECTQ (U-CASE type)
        ((IV IVPROP NIL)
            (DeleteCIV self name prop))
        ((CV CVPROP)
            (DeleteCV self name prop))
        ((META METACLASS CLASS)
            [COND
               (prop (DeleteClassProp self prop))
               (T (PutClass self 'Class])
        ((METHOD SELECTOR)
            (DeleteMethod self name prop))
        (ERROR type "is not an alterable part of class. Use one of
              IV CV META METHOD")))
```

```
(Method ((Class Destroy) class)                                    ; smL  8-Apr-87 19:13
   "Destroys (deletes) a class by putting NIL as localRecord of entity."
   (COND
      ((fetch subClasses of class)
       (HELPCHECK class "has subclasses. You cannot Destroy classes
             that have subclasses. Type OK to use Destroy! if that is what you want. ")
       (_ class Destroy!))
      (T (_ (Class class)
            DestroyClass class)))
   (FlushMethodCache))
```

```
(Method ((Class Destroy!) class)                                   ; dlsb:  3-JAN-83 14:46
   "Recursive version of Destroy. Destroys class and its subclasses."
   (PROG ((subClasses (fetch subClasses of class)))
         [COND
            (subClasses (replace subClasses of class with NIL)
                   (for sc in subClasses do (_ (OR (CAR (LISTP sc))
                                                   sc)
                                               Destroy!]
         (_ class Destroy)))
```

```
(Method ((Class DestroyClass) self classToDestroy)                 ; smL 21-May-86 12:37
   "Destroy the class specified by smashing its contents"
   (COND
      ((type? class classToDestroy)                                (* First delete from knowledge of file system)
       (LET ((className (ClassName classToDestroy)))
            (_ classToDestroy DelFromFile)
            ;; JRB - Must do this first, since it can call $! through ClassBrowserMarkAsChanged, which will recreate a dynamic mixin class (!).
            (MARKASCHANGED className 'CLASSES 'DELETED)
            (for M in (_ classToDestroy ListAttribute 'MethodObjects) do (_ ($! M)
                                                                            Destroy!))
            (DeleteObjectName classToDestroy className)            (* Remove from subClasses lists of each super.)
            (for superName in (_ classToDestroy ListAttribute 'Supers) bind super when (SETQ super (GetClassRec
                                                                                                    superName))
               do (replace subClasses of super with (for sub in (fetch subClasses of super)
                                                       when (NEQ classToDestroy (COND
                                                                                   ((LISTP sub)
                                                                                    (CAR sub))
                                                                                   (T sub)))
                                                       collect sub)))
                                                                   (* smash back pointer to entity rec, the list of vars and var descriptions)
            (replace className of classToDestroy with '*aDestroyedClass*)
            (replace otherClassDescription of classToDestroy with NIL)
            (replace VARNAMES of classToDestroy with NIL)
            (replace VARDESCRS of classToDestroy with NIL)         (* It is a classToDestroy so smash its list of subs and Supers)
            (replace localSupers of classToDestroy with (LIST ($ DestroyedObject)))
            (replace supers of classToDestroy with (LIST ($ DestroyedObject)
                                                         ($ Object)))
            (replace selectors of classToDestroy with NIL)
            (replace methods of classToDestroy with NIL)
            (replace cvNames of classToDestroy with NIL)
            (replace cvDescrs of classToDestroy with NIL)
            (replace localIVs of classToDestroy with NIL)
            (replace ivNames of classToDestroy with NIL)
            (replace ivDescrs of classToDestroy with NIL)
            (replace metaClass of classToDestroy with ($ DestroyedClass))
            (DeleteObjectUID classToDestroy)
            'DestroyedClass))
      (T (LoopsHelp classToDestroy "not a class for DestroyClass"))))
```

```
(Method ((Class DestroyInstance) self instance)                    ; smL  8-May-86 13:22
   "smash the list of vars and var descriptions"
   (COND
      ((type? class instance)
       (_ instance DelFromFile)
       (_ self DestroyClass instance))
      ((NULL (type? instance instance))
       (LoopsHelp instance "not instance for DestroyInstance"))
      (T (_ instance DelFromFile)
         ;; And to keep it off of (FILES?)...
```

```
         (for n in (GetObjectNames instance) when (CL:SYMBOLP n)  ; Avoid the UID... do (UNMARKASCHANGED n 'INSTANCES))
         (DeleteObjectName instance)
         (DeleteObjectUID instance)
         (replace class of instance with ($ DestroyedObject))
         (replace VARNAMES of instance with NIL)
         (replace VARDESCRS of instance with NIL)
         (replace otherIVs of instance with NIL))))

(Method ((Class HasCV) self CVName prop)                          ; smL 18-Apr-86 14:28
   "Tests if class has the specified class variable"
   (PROG (index (supers (Supers self))
                (class self))
     LP  [COND
            ((SETQ index (FindIndex CVName (fetch cvNames of class)))
             (COND
                ([OR (NULL prop)
                     (for tail on (CDR (GetNth (fetch cvDescrs of class)
                                               index))
                        by (CDDR tail) do (COND
                                               ((EQ prop (CAR tail))
                                                (RETURN T]
                 (RETURN T]                                       (* this is where the substitution goes)
         (COND
            ((SETQ class (pop supers))                            (* If there is a Super, iterate around the Loop)
             (GO LP)))                                            (* Returns NIL if not found)
         (RETURN NIL)))

(Method ((Class HasIV) self IVName prop)                          ; smL 17-Apr-87 09:24
   "Tests if class has the specified instance variable"
   [AND (FMEMB IVName (fetch localIVs of self))
        (OR (NULL prop)
            (FMEMB prop (_ self ListAttribute 'IVProps IVName])

(Method ((Class HasIV!) self IVName prop)                         ; smL 29-Sep-86 10:57
   "Tests if class has the specified instance variable"
   (COND
      (prop (AND (FMEMB IVName (fetch ivNames of self))
                 (FMEMB prop (SEND self ListAttribute! 'IVProps IVName))
                 T))
      (T (AND (FMEMB IVName (fetch ivNames of self))
              T))))

(Method ((Class MakeLocalMethod) self selector dontComplain?) ; edited:  3-Apr-86 18:10
   "make an inherited method be local"
   (LET ((methClass (FetchMethodClass self selector)))
        (COND
           ((NEQ self methClass)
            (_ methClass CopyMethod selector self selector))
           (dontComplain? NIL)
           (T (PROMPTPRINT selector " is already local in " self)
              NIL))))

(Method ((Class MethodDoc) self selector)                         ; smL  9-Apr-87 19:14
   "Show documentation for method in PPDefault window"

;;;

   [PROG (items menu sel (outFile (PPDefault NIL)))
         (COND
            [(NULL selector)                                      ; Let user pick the selector from a menu
             [SETQ items (SORT (_ self ListAttribute! 'Selectors]
             (COND
                (items (SETQ menu (create MENU
                                          ITEMS _ items
                                          CHANGEOFFSETFLG _ T)))
                (T (PROMPTPRINT "No methods in " self)
                   (RETURN NIL]
            ((NOT (FetchMethod self selector))
             (printout outFile T "No method for " selector " in class " (ClassName self)
                    T)
             (RETURN NIL)))
     LP  (SETQ sel (OR selector (MENU menu)))
         (COND
            (sel (WITH.PP.OUTPUT outFile (printout outFile T T "class: " .FONT LAMBDAFONT (ClassName self)
                                                   .FONT DEFAULTFONT [LET [(classWhere (ClassName
                                                                                        (_ self WhereIs sel
                                                                                           'Method]
                                                                                 (COND
                                                                                    ((EQ classWhere (ClassName self))
                                                                                     "")
                                                                                    (T (CONCAT " (from " classWhere ")"]
                                                   " selector: " .FONT LAMBDAFONT sel .FONT DEFAULTFONT "
                                                   args: " (GetMethod self sel 'args)
                                                   "
                                                   doc: "
                                                   (GetMethod self sel 'doc)
```

```
                                          T)))
            (T (RETURN NIL)))
         (COND
            (selector (RETURN NIL))
            (T (SETQ sel NIL)
               (GO LP])

(Method ((Class MethodSummary) self dontPrintFlg printFile)  ; smL 9-Apr-87 19:15
   "prints a summary of the methods in a class"
   [COND
      (dontPrintFlg (GetSourceMethods self))
      (T (LET ((outFile (PPDefault printFile)))
              (WITH.PP.OUTPUT outFile (PRINTDEF (GetSourceMethods self)
                                                NIL NIL NIL NIL outFile)
                     (TERPRI outFile])

(Method ((Class MoveMethod) self newClass selector)           ; smL 9-Apr-87 18:40
   "Move method specified by selector from this class to newClass"
   (MoveMethod (ClassName self)
          (if (Class? newClass)
              then (ClassName newClass)
            else newClass)
          selector))

(Method ((Class OnFile) self file)                            ; smL 8-Apr-87 17:54
   "See if class is on given file. Returns file if none given"
   (LET [(myfile (WHEREIS (ClassName self)
                          'CLASSES]
         (COND
            (file (MEMB file myfile))
            (T myfile))))

(Method ((Class PP) self file)                                ; smL 9-Apr-87 19:16
   "Prettyprint the class."
   (LET ((outFile (PPDefault file)))
        (WITH.PP.OUTPUT outFile (PrettyPrintClass (ClassName self)
                                        outFile))))

(Method ((Class PP!) self file)                               ; smL 9-Apr-87 19:17
   "Method for prettyPrinting self at all levels."
   (SETQ file (PPDefault file))
   (RESETLST
       (RESETSAVE FIRSTCOL 16)
       (RESETSAVE ([LAMBDA (X)
                     (DSPFONT X file]
                  NIL))
       (WITH.PP.OUTPUT file (printout file .FONT LAMBDAFONT self)
                                                              (* PP Class props)
              (printout file .FONT BOLDFONT T "MetaClass and its Properties" T)
              (printout file .FONT DEFAULTFONT "  " (ClassName (Class self)))
              (for cp in (_ self ListAttribute! 'CLASS) do (printout file .FONT COMMENTFONT "  " cp " "
                                                                    (GetClassOnly self cp)))
                                                              (* List supers)
              (printout file .FONT BOLDFONT T "Supers")
              (printout file .FONT DEFAULTFONT T "  " (_ self ListAttribute! 'SUPERS))
                                                              (* PP instance variables and props)
              (printout file .FONT BOLDFONT T "Instance Variable Descriptions" T)
              (for iv in (_ self ListAttribute! 'IVS) bind (locals _ (_ self ListAttribute 'IVS))
                 do (printout file .FONT (COND
                                            ((FMEMB iv locals)
                                             BOLDFONT)
                                            (T DEFAULTFONT))
                           "  " iv " " .FONT DEFAULTFONT (GetValueOnly self iv))
                    (for ivp in (_ self ListAttribute! 'IV iv) do (printout file .FONT COMMENTFONT " " ivp " "
                                                                         (GetValueOnly self iv ivp)))
                    (printout file .FONT BOLDFONT T))         (* PP Class Variables & props)
              (printout file .FONT BOLDFONT "Class Variables" T)
              (for cv in (_ self ListAttribute! 'CVS) bind (locals _ (_ self ListAttribute 'CVS))
                 do (printout file .FONT (COND
                                            ((FMEMB cv locals)
                                             BOLDFONT)
                                            (T DEFAULTFONT))
                           "  " cv " " .FONT DEFAULTFONT (GetClassValueOnly self cv))
                    (for cvp in (_ self ListAttribute! 'CV cv) do (printout file .FONT COMMENTFONT " " cvp " "
                                                                         (GetClassValueOnly self cv cvp)))
                    (printout file .FONT BOLDFONT T))         (* PP Methods and props)
              (printout file .FONT BOLDFONT "Methods" T)
              (for selector in (SORT (_ self ListAttribute! 'SELECTORS)) bind (locals _ (_ self ListAttribute
                                                                                           'SELECTORS))
                 do (printout file .FONT (COND
                                            ((FMEMB selector locals)
                                             BOLDFONT)
                                            (T DEFAULTFONT))
                           "  " selector " " (FetchMethod self selector))
                    (for methodProp in (_ self ListAttribute! 'METHOD selector)
                       do (printout file .FONT COMMENTFONT " " methodProp " " (GetMethodOnly self selector
                                                                                    methodProp)))
```

```
                    (printout file .FONT BOLDFONT T))
                (printout file .FONT DEFAULTFONT T))
        self))

(Method ((Class PPM) self selector)                                    ; smL 25-Apr-86 14:30
   "Pretty-print the named method for this class"
   (Class.PPMethod self selector))

(Method ((Class PPMethod) self selector)                               ; smL  9-Apr-87 19:18
   "Prettyprint the function which implements selector in this class."

;;;

   [LET [(selector (OR selector (_ self PickSelector (CONCAT "PPMethod: " (ClassName self]
        (COND
           (selector (LET ((fn (_ self FetchMethod selector))
                           (outFile (PPDefault NIL)))
                          (if fn
                              then [RESETFORM (OUTPUT outFile)
                                              (WITH.PP.OUTPUT outFile (TERPRI outFile)
                                                              (PRINTDEF (GETDEF fn 'METHOD-FNS]
                               else (PRINTOUT outFile "No method for selector " selector " in " self T])

(Method (((Class PPV!) self file)                                      ; smL  9-Apr-87 19:18
   "Method for prettyPrinting a self at all levels."
   (SETQ file (PPDefault file))
   (RESETLST
       (RESETSAVE FIRSTCOL 16)
       (RESETSAVE ([LAMBDA (X)
                     (DSPFONT X file]
                  NIL))
       (WITH.PP.OUTPUT file (printout file .FONT LAMBDAFONT self)
                                                                       (* PP Class props)
              (printout file .FONT BOLDFONT T "MetaClass and its Properties" T)
              (printout file .FONT DEFAULTFONT "  " (ClassName (Class self)))
              (for cp in (_ self ListAttribute! 'CLASS) do (printout file .FONT COMMENTFONT "  " cp " "
                                                                    (GetClassOnly self cp)))
                                                                       (* List supers)
              (printout file .FONT BOLDFONT T "Supers")
              (printout file T .FONT DEFAULTFONT "  " (_ self ListAttribute! 'SUPERS))
                                                                       (* PP instance variables and props)
              (printout file .FONT BOLDFONT T "Instance Variable Descriptions" T)
              (for iv in (_ self ListAttribute! 'IVS)
                 do (printout file .FONT DEFAULTFONT "  " iv " " (GetValueOnly self iv))
                    (for ivp in (_ self ListAttribute! 'IV iv) do (printout file .FONT COMMENTFONT " " ivp " "
                                                                          (GetValueOnly self iv ivp)))
                    (printout file .FONT BOLDFONT T))              (* PP Class Variables & props)
              (printout file .FONT BOLDFONT "Class Variables" T)
              (for cv in (_ self ListAttribute! 'CVS)
                 do (printout file .FONT DEFAULTFONT "  " cv " " (GetClassValueOnly self cv))
                    (for cvp in (_ self ListAttribute! 'CV cv) do (printout file .FONT COMMENTFONT " " cvp " "
                                                                          (GetClassValueOnly self cv cvp)))
                    (printout file .FONT BOLDFONT T))
              (printout file .FONT DEFAULTFONT T))
        self))

(Method (((Class PrintSummary) self file)                              ; smL  9-Apr-87 19:19
   "Print a summary of elements of class"
   [LET ((file (PPDefault file))
         temp)
        (RESETLST
            (RESETSAVE NIL (LIST (FUNCTION DSPFONT)
                                 (DSPFONT NIL file)
                                 file))
            (WITH.PP.OUTPUT file
                   (printout file T T .FONT LAMBDAFONT "   " self T .FONT BOLDFONT "Supers" T 3 .FONT
                         DEFAULTFONT .PARA 10 0 (_ self ListAttribute 'SUPERS))
                   [for type in '(IVs CVs Methods)
                      do (printout file T .FONT BOLDFONT type T 3 .PARA 3 0 (SORT (SETQ temp
                                                                                        (_ self ListAttribute type)))
                                   .FONT DEFAULTFONT T 2 .PARA 2 0 (SORT (LDIFFERENCE (_ self ListAttribute! type)
                                                                                      temp]
                   (TERPRI file)))]
   self)

(Method (((Class Put) self type name value prop)                       ; smL 18-Jun-86 13:09
   "Type is one of IV IVPROP CV CVPROP META METHOD. Adds the specified type to the class"
   (_ self Add type name value prop))

(Method (((Class UnbreakMethod) self selector)                         ; dgb:  1-Feb-85 13:40
   "Break selected method, or give choice if selector is NIL"
   [AND (NULL selector)
        (SETQ selector (NiceMenu (_ self SelectorsWithBreak)
                                 (CONCAT "Broken Methods For " (ClassName self]
   (COND
      [selector (APPLY* 'UNBREAK (OR (FetchMethod self selector)
                                     (ERROR selector (CONCAT " not found in " self]
```

```
      (T 'NothingUnbroken)))

(Method ((Object AddIV) self name value prop)                    ; smL 25-Apr-86 14:29
   "Make sure that self has an IV (or IVProp) by the given name, with the given value. If there is no such
    IV, add one to self."
   (AddIV self name value prop))

(Method ((Object CopyDeep) oldInstance newObjAlist)              ; smL 23-May-86 09:36
   "Copies the unit, sharing the iName list, copying instances, activeValues and lists - newObjAlist is an
    association list of old copied objects with there associated copies. Used to allow copying of
   circular structures."
   (LET ((newInstance (_ (Class oldInstance)
                         CreateInstance)))
        (COND
           ((fetch OBJUID of oldInstance)

             (* Has an OBJUID so this is not a temporary object. Create OBJUID for new object)

            (NewEntity newInstance)))
        [FillIVs newInstance (Class oldInstance)
               (CopyDeepDescr (IVSource oldInstance)
                      (NCONC1 newObjAlist (CONS oldInstance newInstance]
        newInstance))

(Method ((Object CopyShallow) self)                              ; edited: 19-Mar-85 13:56
   "Create a copy of self w/o initialization"
   (CopyInstance self))

(Method ((Object Get) self varName propName)                     ; smL 23-May-86 11:16
   "Method indirection for GetValue"
   (GetIVProp self varName propName))

(Method ((Object HasCV) self cvName prop)                        ; dgb: 13-Feb-85 11:58
   "Ask class"
   (_ (Class self)
      HasCV cvName prop))

(Method ((Object HasIV) self ivName prop)                        ; smL 27-May-86 13:13
   "Check to see if IV is on self. If so, return T"
   (AND (OR (FMEMB ivName (fetch iNames of self))
            (FASSOC ivName (fetch otherIVs of self)))
        [OR (NULL prop)
            [WithIVPropDescr self ivName [LAMBDA (self ivName propDescr)
                                           (AND propDescr (for pList on (fetch IVPropList of propDescr)
                                                             by (CDDR pList) thereis (EQ prop (CAR pList]
                  (LAMBDA (self ivName)
                    NIL]
            (for super in-supers-of (Class self) thereis (for pList on (fetch IVProps of (FetchCIVDescr super ivName
                                                                                          ))
                                                            by (CDDR pList) thereis (EQ prop (CAR pList]
        T))

(Method ((Object DeleteIV) self varName propName)                ; smL 25-Apr-86 14:29
   "Delete the named IV from self"
   (DeleteIV self varName propName))

(Method ((Object Inspect) self INSPECTLOC)                       ; sm: 12-SEP-83 15:54
   "Inspect the object as class or instance, using INSPECTLOC as a region for the inspect window, if it is
    given"
   (INSPECT self NIL INSPECTLOC))

(Method ((Object InstOf) self class)                             ; dgb: 12-JAN-83 15:58
   "test if self is an instance of class"
   [EQ (ClassName (Class self))
       (COND
          ((LITATOM class)
           class)
          (T (ClassName class])

(Method ((Object InstOf!) self class)                            ; dgb: 26-JAN-83 09:51
   "Recursive version of InstOf."
   (_ (Class self)
      Subclass class))

(Method (Object OnFile) self file) "See if an instance is on given file. Returns file if none given"
   (LET [(myfile (WHEREIS (ClassName self)
                          'INSTANCES]
        (COND
           (file (MEMB file myfile))
           (T myfile))))

(Method (Object PP) self file)                                   ; smL  9-Apr-87 19:20
   "Pretty print an instance on a file"
   (LET ((ObjectDontPPFlag T)
         (file (PPDefault file)))
        (WITH.PP.OUTPUT file (PrettyPrintInstance self file))))
```

```
(Method ((Object PP!) self file)                                    ; smL 9-Apr-87 19:19
   "Method for prettyPrinting a non-class object at all levels"
   (SETQ file (PPDefault file))
   (RESETLST
       (RESETSAVE FIRSTCOL 16)
       (RESETSAVE ([LAMBDA (X)
                     (DSPFONT X file]
                  NIL))
       (WITH.PP.OUTPUT file (PROG ((class (Class self)))         (* PP Class Variables & props)
                                  (printout file .FONT LAMBDAFONT self T)
                                  (printout file .FONT BOLDFONT "Instance Variables" T)
                                  (for iv in (_ self ListAttribute! 'IVS) unless (EQ iv 'indexedVars)
                                     do (printout file .FONT DEFAULTFONT "  " iv " " (GetValueOnly self iv))
                                        (for ivp in (_ self ListAttribute! 'IV iv)
                                           do (printout file .FONT COMMENTFONT " " ivp " " (GetValueOnly self iv
                                                                                                  ivp)))
                                        (TERPRI file))
                                  [COND
                                     ((_ self InstOf! 'VarLength)
                                                                   (* Special printing for indexed variables.)
                                      (printout file .FONT BOLDFONT "Indexed Variables" T)
                                      (for iv in (GetValueOnly self 'indexedVars) as index from 1
                                         to (_ self Length)
                                         do (printout file .FONT DEFAULTFONT "  " index .TAB0 4 (GetValueOnly
                                                                                                  self index))
                                            (for ivp in (_ self ListAttribute! 'IV index)
                                               do (printout file .FONT COMMENTFONT " " ivp " "
                                                            (GetValueOnly self index ivp)))
                                            (TERPRI file]
                                  (printout file .FONT BOLDFONT "Class Variables" T)
                                  (for cv in (_ class ListAttribute! 'CVS)
                                     do (printout file .FONT DEFAULTFONT "  " cv " " (GetClassValueOnly class cv
                                                                                                  ))
                                        (for cvp in (_ class ListAttribute! 'CV cv)
                                           do (printout file .FONT COMMENTFONT " " cvp " " (GetClassValueOnly
                                                                                            class cv cvp)))
                                        (TERPRI file))         (* PP instance variables and props)
                                                               (* PP Methods and props)
                                  (printout file .FONT BOLDFONT "Methods" T)
                                  (for selector in (SORT (_ class ListAttribute! 'SELECTORS))
                                     do (printout file .FONT DEFAULTFONT "  " selector " " (FetchMethod class
                                                                                                  selector))
                                        (for methodProp in (_ class ListAttribute! 'METHOD selector)
                                           do (printout file .FONT COMMENTFONT " " methodProp " "
                                                        (GetMethodOnly class selector methodProp)))
                                        (TERPRI file))
                                  (printout file .FONT DEFAULTFONT T))
            self)))

(Method ((Object PPIVs) self file)                                  ; smL 11-Apr-86 14:45
   "Called to prettyPrint a class definition on a file by the FILEPKGTYPE CLASSES and by the PP: method in
    Class"
   (PROG [(vars (for v in (_ self ListAttribute 'IVs) collect (LIST v (GetValueOnly self v]
         (printout file "[ " .FONT LAMBDAFONT self T .FONT DEFAULTFONT 3 .PPFTL vars "]" T T)))

(Method ((Object PPV!) self file)                                   ; smL 9-Apr-87 19:20
   "Method for prettyPrinting a non-class object at all levels."
   (SETQ file (PPDefault file))
   (RESETLST
       (RESETSAVE FIRSTCOL 16)
       (RESETSAVE ([LAMBDA (X)
                     (DSPFONT X file]
                  NIL))
       (WITH.PP.OUTPUT file (PROG ((class (Class self)))         (* PP Class Variables & props)
                                  (printout file .FONT LAMBDAFONT self T)
                                  (printout file .FONT BOLDFONT "Instance Variables" T)
                                  (for iv in (_ self ListAttribute! 'IVS) unless (EQ iv 'indexedVars)
                                     do (printout file .FONT DEFAULTFONT "  " iv " " (GetValueOnly self iv))
                                        (for ivp in (_ self ListAttribute! 'IV iv)
                                           do (printout file .FONT COMMENTFONT " " ivp " " (GetValueOnly self iv
                                                                                                  ivp)))
                                        (TERPRI file))
                                  [COND
                                     ((_ self InstOf! 'VarLength)
                                                                   (* Special printing for indexed variables.)
                                      (printout file .FONT BOLDFONT "Indexed Variables" T)
                                      (for iv in (GetValueOnly self 'indexedVars) as index from 1
                                         to (_ self Length)
                                         do (printout file .FONT DEFAULTFONT "  " index .TAB0 4 (GetValueOnly
                                                                                                  self index))
                                            (for ivp in (_ self ListAttribute! 'IV index)
                                               do (printout file .FONT COMMENTFONT " " ivp " "
                                                            (GetValueOnly self index ivp)))
                                            (TERPRI file]
                                  (printout file .FONT BOLDFONT "Class Variables" T)
                                  (for cv in (_ class ListAttribute! 'CVS)
                                     do (printout file .FONT DEFAULTFONT "  " cv " " (GetClassValueOnly class cv
```

```
                                                                                    ))
                                        (for cvp in (_ class ListAttribute! 'CV cv)
                                           do (printout file .FONT COMMENTFONT " " cvp " " (GetClassValueOnly
                                                                                            class cv cvp)))
                                        (TERPRI file))              (* PP instance variables and props)
                                                                    (* PP Methods and props)
                                  (printout file .FONT DEFAULTFONT T))
            self)))

(Method ((Object PrintOn) self file)                                ; smL 14-Jan-87 14:34
   "This is the default printing function for object. It returns a form whose CAR is PRIN1ed and elements of CDR
   PRIN2ed."
   [LET ((name (GetObjectName self))
         (uid (fetch OBJUID of self)))
        (CONS (CONCAT LoopsReadMacroChar ",")
              (COND
                 (name (LIST '$ name))
                 ((NULL uid)                                        ; A temporary object
                  (LIST '$& (ClassName self)
                        (EntityAddress self)))
                 (T (LIST '$& (ClassName self)
                          uid])

(Method ((Object Put) self varName newValue propName)               ; smL 27-May-86 11:26
   "Method indirection for PutValue"
   (PutValue self varName newValue propName))

(Method ((Object Sublis) self alist)                                ; smL 11-Apr-86 14:48
   "Copy this instance, substituting as specified by alist"
   ;; alist items are either one or two element lists (key substKey) . If a two element list, then the
   ;; substKey is substituted for key. If it contains only key, then the first time the key is found, a copy
   ;;  is made then, and used from then on. Sublis copies all lists and active values. It searches lists
   ;; recursively EXCEPT if they begin with *, to avoid searching comments.
   (PROG (myObj (myPair (FASSOC self alist)))
         [COND
            ((NULL (CDR myPair))                                    (* Need a copied object)
             (SETQ myObj (BlankInstance (Class self)))              (* If there is a pair, insert the new object)
             (AND myPair (RPLACD myPair (LIST myObj]
         [COND
            ((NULL myPair)                                          (* self is not on the alist)
             (SETQ alist (NCONC1 alist (SETQ myPair (LIST self myObj]
         [for iv val in (_ self ListAttribute 'IVs) do (for prop in (CONS NIL (_ self ListAttribute 'IVPROPS iv))
                                                          do        (* Subst for value (the NIL property) and all properties)
                                                             (SETQ val (IVSublis (GetIVHere self iv prop)
                                                                                 alist))
                                                             (COND
                                                                ((NEQ val NotSetValue)
                                                                 (PutValueOnly myObj iv val prop]
         (RETURN myObj)))

(Method ((Object Understands) self selector)                        ; smL 29-May-86 17:51
   "Tests if object will respond to selector"
   (AND (FetchMethod (Class self)
                     selector)
        T))

(Method ((Object VirtualCopy?) self)                                ; kmk: 13-Nov-84 15:57
   "no I'm not a virtual copy"
   NIL)

(Method ((Object WhereIs) self name type propName)                  ; smL 10-Dec-86 18:00
   "Searches the supers hierarchy until it finds the class from which type is inherited. type=NIL is
   METHODS"
   [PROG ((realType (U-CASE type)))
     LP  [SETQ realType (SELECTQ realType
                               ((METHOD METHODS T NIL)
                                   'METHODS)
                               ((IVPROP IVPROPS)
                                   'IVPROPS)
                               ((IV IVS)
                                   'IVS)
                               ((CV CVS)
                                   'CVS)
                               (PROGN (SETQ realType (HELPCHECK "Use one of METHODS IVS IVPROPS CVS for type.
                                                                 RETURN one of these atoms to go on"))
                                      (GO LP]
         (RETURN (COND
                    [propName (for class in-supers-of self thereis (FMEMB propName (_ class ListAttribute realType
                                                                                      name]
                    (T (for class in-supers-of self thereis (FMEMB name (_ class ListAttribute realType])

(\UnbatchMethodDefs)

(PUTPROPS LOOPSUTILITY COPYRIGHT ("Venue & Xerox Corporation" 1983 1984 1985 1986 1987 1988 1990))
```

FUNCTION INDEX

VARIABLE INDEX