File created: 15-Jun-93 12:34:16 **{DSK}<python>release>loops>2.0>src>LOOPSKERNEL.;3**

changes to: (METHODS Class.MakeEditSource)

previous date: 6-Nov-91 13:03:28 {DSK}<python>release>loops>2.0>src>LOOPSKERNEL.;2

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ LOOPSKERNELCOMS
       [(DECLARE%: DONTCOPY (PROP MAKEFILE-ENVIRONMENT LOOPSKERNEL)
              (PROP FILETYPE LOOPSKERNEL))
```

;;; Functions called by kernel classses

```
        (FNS * KERNELFNS)
```

;;; The kenel classes themselves

```
        (CLASSES * KERNELCLASSES)
        (INITVARS (Viewed-Categories '(Public))
               (DumpMethodsInClass))
        (METHODS Class.AllInstances Class.AllMethodCategories Class.CVMissing Class.CVValueMissing
               Class.CategorizeMethods Class.ChangeMethodCategory Class.CreateInstance Class.DefMethod
               Class.DelFromFile Class.EM! Class.Edit Class.Edit! Class.EditMethod Class.EditMethodObject
               Class.FetchMethod Class.FileIn Class.FileOut Class.Fringe Class.GetClassProp Class.HasAttribute
               Class.HasAttribute! Class.HasItem Class.IndexedInstances Class.Initialize Class.InstallEditSource
               Class.ListAttribute Class.ListAttribute! Class.MakeEditSource Class.MakeFileSource
               Class.MakeFullEditSource Class.MethodCategories Class.MoveToFile Class.MoveToFile! Class.New
               Class.NewClass Class.NewWithValues Class.Old Class.PickSelector Class.Prototype Class.Rename
               Class.RenameMethod Class.ReplaceSupers Class.SelectorsInCategories Class.SelectorsWithBreak
               Class.SetName Class.Specialize Class.SpecializeMethod Class.SubClasses Class.Subclass
               Class.TraceMethod Class.UnSetName DestroyedClass.Destroy DestroyedClass.Destroy!
               DestroyedClass.DestroyClass DestroyedClass.DestroyInstance DestroyedClass.SubClasses
               DestroyedObject.Destroy! MetaClass.CreateClass MetaClass.DestroyInstance MetaClass.New
               MetaClass.NewWithValues Method.ChangeClassName Method.ChangeName Method.DelFromFile
               Method.EditMethod Method.FileOut Method.MakeFileSource Method.ObjectModified Method.OldInstance
               Method.UnSetName Object.ChangeClass Object.Class Object.ClassName Object.ConformToClass
               Object.DelFromFile Object.Destroy Object.Destroy! Object.DoMethod Object.Edit Object.FileOut
               Object.HasAttribute Object.HasAttribute! Object.IVMissing Object.IVValueMissing
               Object.InstallEditSource Object.InstallFileSource Object.ListAttribute Object.ListAttribute!
               Object.MakeEditSource Object.MakeFileSource Object.MessageNotUnderstood Object.MoveToFile
               Object.NewInstance Object.ObjectModified Object.OldInstance Object.OnFile Object.Rename
               Object.SaveInstance Object.SaveInstance? Object.SetName Object.UnSetName
               Tofu.MessageNotUnderstood Tofu.MethodNotFound Tofu.SuperMethodNotFound)
        (FNS MakeMethodMenu MethodMenuWhenSelectedFn SelectFile)
        (DECLARE%: EVAL@COMPILE DONTCOPY (EXPORT (MACROS \PutValueOnly))
               (FILES (LOADCOMP)
                      LOOPSDATATYPES LOOPSACTIVEVALUES LOOPSMETHODS))
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA METHCOM)
                                                                              (NLAML OldClass)
                                                                              (LAMA SelectFile])

(DECLARE%: DONTCOPY

(PUTPROPS LOOPSKERNEL MAKEFILE-ENVIRONMENT (:PACKAGE "IL" :READTABLE "INTERLISP" :BASE 10))

(PUTPROPS LOOPSKERNEL FILETYPE :COMPILE-FILE)
)
```

;;; Functions called by kernel classses

```
(RPAQQ KERNELFNS (AddCIV AddCV AddIV AllSubClasses ClassName CopyCVToIV CopyDeepDescr CopyInstance
                         CopyLoopsStruc DeleteIV DumpInstanceFacts EnsureFnLoaded FixSelectorSpelling
                         \LoopsFixSpell \LoopsDwim \FixSelectorSpelling GetMethodObj GetMethodObj! IVSublis
                         METHCOM MapIVs MapIVs! NewWithValues OldClass SendMessageNotUnderstood SubsTree
                         TypeInMethods WhoHas))

(DEFINEQ

(AddCIV
  [LAMBDA (class varName defaultValue otherProps)              ; Edited 25-Jun-87 14:02 by smL
    ;; Add an instance variable to the class, if needed, and add properties of otherProps
    (COND
       ([AND (NULL varName)
             (NULL (SETQ varName (PromptRead "Please type the name of the new IV: "]
```

```
        NIL)
       [(FMEMB varName (_ class ListAttribute 'IVs))            ; Variable is local
        (PutClassIV class varName defaultValue)
        (for p on otherProps by (CDDR p) do (PutClassIV class varName (CADR p)
                                                    (CAR p]
       ((OddLengthList otherProps)
        (ERROR "Odd length property list"))
       (T [InstallInstanceVariables class (NCONC1 (GetSourceIVs class)
                                                  (CONS varName (CONS defaultValue otherProps]
          (OR (FMEMB 'doc (_ class ListAttribute! 'IVPROPS varName))
              (PutClassIV class varName `(,COMMENTFLG IV added by ,(USERNAME NIL T))
                     'doc))
          varName])
```

(AddCV

```
  [LAMBDA (class varName newValue)                                  ; Edited 25-Jun-87 14:00 by smL
```

;;; 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

```
    (COND
       ([AND (NULL varName)
             (NULL (SETQ varName (PromptRead "Please type name of new CV: "]
        NIL)
       ((FetchCVDescr class varName)
        (AND newValue (PutClassValueOnly class varName newValue))
        NIL)
       (T (InstallClassVariables class (NCONC1 (GetSourceCVs class)
                                               (LIST varName newValue)))
          (OR (_ class HasCV varName 'doc)
              (PutClassValue class varName `(,COMMENTFLG CV added by ,(USERNAME NIL T))
                     'doc))
          varName])
```

(AddIV

```
  [LAMBDA (self name value prop)                                    ; Edited 14-Aug-90 16:23 by jds
```

(* * Adds an IV to instance. If it is not in regular set, puts it in assoc List on otherIVs)

```
    [COND
       ((NULL name)
        (ERROR "A name must be given to add an IV"))
       ((NOT (_ self HasIV name))
        (push (fetch (instance otherIVs) of self)
              (CONS name NotSetValue]
    (PutValueOnly self name value prop)
    value])
```

(AllSubClasses

```
  [LAMBDA (class currentSubs)                                       ; Edited 14-Aug-90 16:22 by jds
                                                                    (* Gets all subclasses recursively, making sure there are no
                                                                    duplicates. Called from Class.List!)
    (for SUB in (fetch (class subClasses) of class) do [OR (FMEMB (SETQ SUB (OR (CAR (LISTP SUB))
                                                                               SUB))
                                                                  currentSubs)
                                                           (SETQ currentSubs (AllSubClasses SUB (CONS SUB
                                                                                                      currentSubs]
       finally (RETURN currentSubs])
```

(ClassName

```
  [LAMBDA (self)                                                    ; Edited 14-Aug-90 16:23 by jds
                                                                    (* Returns className of class of object)
    (COND
       ((type? class self)
        (ffetch (class className) of self))
       ((type? instance self)
        (ffetch (class className) of (ffetch (instance class) of self)))
       (T (LET ((class (GetLispClass self)))
               (COND
                  (class (ClassName class))
                  (T (LoopsHelp self "has no class name"])
```

(CopyCVToIV

```
  [LAMBDA (self varName)                                            ; Edited 14-Aug-90 16:27 by jds
```

(* * Used by the IVMissing protocol to copy a CV down to an IV when there is an %:allocation property)

```
    (LET ((initForm (GetClassValue self varName '%:initForm))
          (classValue (GetClassValueOnly self varName)))
         (AddIV self varName (if (ValueFound initForm)
                                 then (EVAL initForm)
                              elseif (type? annotatedValue classValue)
```

```
                                then (_AV
                                      classValue CopyActiveValue classValue)
                              else classValue)))
    (for p on (fetch (IVDescr IVProps) of (FetchCVDescr (_ self Class)
                                                varName))
       by (CDDR p) do (PutValueOnly self varName (if (type? annotatedValue (CADR p))
                                                     then (_AV
                                                           (CADR p)
                                                           CopyActiveValue
                                                           (CADR p))
                                                   else (CADR p))
                          (CAR p])
```

(CopyDeepDescr

```
[LAMBDA (descr newObjAlist)                                    ; Edited 14-Jun-88 12:52 by TAL
  (DECLARE (LOCALVARS . T))
```

;; Copies instances active values and lists, but bottoms out on anything else

```
  (SELECTQ (TYPENAME descr)
       (instance (OR (CDR (FASSOC descr newObjAlist))
                     (_ descr CopyDeep newObjAlist)))
       (annotatedValue
            (create annotatedValue
                    annotatedValue _ (CopyDeepDescr (fetch annotatedValue of descr)
                                                    newObjAlist)))
       (LISTP (bind t2 val for valTail on descr do [COND
                                                     [t2 (FRPLACD t2 (SETQ t2 (LIST (CopyDeepDescr (CAR valTail
                                                                                                   )
                                                                                   newObjAlist]
                                                     (T (SETQ val (SETQ t2 (LIST (CopyDeepDescr (CAR valTail)
                                                                                 newObjAlist]
                                                  [COND
                                                     ((AND (CDR valTail)
                                                           (NLISTP (CDR valTail)))
                                                      (FRPLACD t2 (CopyDeepDescr (CDR valTail)
                                                                                 newObjAlist]
                                                  yield val))
       descr])
```

(CopyInstance

```
[LAMBDA (oldInstance)                                          ; Edited 16-Sep-88 17:26 by TAL
```

;;; make a new instance with the same contents as self, or copy into an instance if given

```
  (LET ((newInstance (_ (Class oldInstance)
                        CreateInstance)))
```

;; Creating UID for copy loses big. E.g., AVs as default IV value in class generally have UID. When IV is first accessed, AV is copied and
;; stored in instance. If copy has UID it will never go away, and in the case of LispWindowAV this causes the window, bitmap, stream, etc. to
;; stay around also.

#|(COND ((AND (fetch OBJUID of oldInstance) (NULL (fetch OBJUID of newInstance))) (* ; "Old one not temporary, but new one has non OBJUID yet") (UID newInstance)))|#

;; Copy IVSource down one layer of list structure.

```
       (FillIVs newInstance (Class oldInstance)
              (MAPCAR (IVSource oldInstance)
                      (FUNCTION APPEND)))
       newInstance])
```

(CopyLoopsStruc

```
[LAMBDA (desc)                                                 (* dgb%: "11-NOV-82 02:29")
  (SELECTQ (TYPENAME desc)
       (instance (_ desc CopyDeep))
       (LISTP (CONS (CopyLoopsStruc (CAR desc))
                    (CopyLoopsStruc (CDR desc))))
       desc])
```

(DeleteIV

```
[LAMBDA (self varName propName)                                ; Edited 14-Aug-90 16:23 by jds
```

(* * Removes an IV from an Instance. No longer shares IVName List with class.
Some programs which depend on IV may not work.)

```
  [COND
     ((NULL (_ self HasIV varName))
      (ERROR varName "Not instance variable in this instance"))
     [propName (WithIVPropDescr self varName [LAMBDA (self varName propDescr)
                                               (InstRemProp propDescr propName]
                       (LAMBDA (self varName)
                         NIL]
     ((_ (Class self)
         HasIV! varName)
      (ERROR varName "in class. Cannot be deleted from instance"))
```

```
       (T (change (fetch (instance instIVProps) of self)
                  (DREMOVE [WithIVPropDescr self varName [LAMBDA (self varName propDescr)
                                                           propDescr]
                                   (LAMBDA (self varName)
                                     NIL]
                           DATUM))
          (FillIVs self (Class self)
                  (DELASSOC varName (IVSource self]
    self])
```

(DumpInstanceFacts

```
  [LAMBDA (instanceRec fileHandle)                                    ; Edited 14-Aug-90 16:23 by jds
```

;;; This prints an expression on the file which specifies the contents of an instance record. Called by (_ object DumpFacts)

```
    (PROG ((filePos (GETFILEPTR fileHandle)))
          (PRIN1 'i fileHandle)
          (PRINT (CONS (fetch (instance class) of instanceRec)
                       (NCONC [for name exceptions descr in (fetch (instance iNames) of instanceRec) as i
                                 from 0 when [NEQ 'Any (SETQ exceptions (GetValueOnly instanceRec name
                                                                                      'DontSave]
                                 collect (SETQ descr (GetVarNth instanceRec i))
                                      ;; Collect a list of properties, omitting those on the list which is the value of the property DontSave.
                                      ;; Value should be on that list if the value is not to be dumped.
                                        (CONS name (COND
                                                     ((EQ NotSetValue exceptions)
                                                      descr)
                                                     ((NULL (CDR descr))
                                                      (COND
                                                         ((FMEMB 'Value exceptions)
                                                          NIL)
                                                         (T descr)))
                                                     (T (CONS (COND
                                                                ((FMEMB 'Value exceptions)
                                                                 ; value is to be omitted
                                                                  NotSetValue)
                                                                (T (CAR descr)))
                                                              (for pair on (CDR descr) by (CDDR pair)
                                                                 when (NOT (FMEMB (CAR pair)
                                                                                  exceptions))
                                                                 join (LIST (CAR pair)
                                                                            (CADR pair]
                                 (fetch (instance otherIVs) of instanceRec)))
                  fileHandle)
          (RETURN filePos])
```

(EnsureFnLoaded

```
  [LAMBDA (fn)                                                        ; Edited 17-Jun-87 16:35 by smL
    (OR [GETDEF fn 'FNS 'CURRENT '(NOERROR NOCOPY 'NODWIM]
        [GETDEF fn 'METHOD-FNS 'CURRENT '(NOERROR NOCOPY 'NODWIM]
        (AND (WHEREIS fn 'FNS)
             (LOADFNS fn NIL 'PROP))
        (AND (WHEREIS fn 'METHOD-FNS)
             (LOADVARS fn NIL 'PROP))
        (HELPCHECK "Can't find source for " fn])
```

(FixSelectorSpelling

```
  [LAMBDA (self selector)                                             (* smL " 8-Apr-87 17:50")
```

(* * Attempt the correct the spelling of a selector -
If we can, and the containing form can be found, smash it to contain the fixed selector)

```
    (LET ((containingForm (if (AND (BOUNDP '\SendForm)
                                   (EQ \Obj self)
                                   (EQ \Selector selector))
                              then \SendForm)))
         (\LoopsFixSpell selector (_ (Class self)
                                     ListAttribute!
                                     'METHODS NIL 'verboseFlg)
                (CONS '_ containingForm)
                (CDR containingForm])
```

(\LoopsFixSpell

```
  [LAMBDA (originalValue possibleValues containingForm tail)          (* smL "13-Aug-86 16:50")
```

(* * Try to correct the originalValue spelling.)

```
    (if (NULL DWIMFLG)
        then                                                          (* DWIM disabled, so don't even try to correct the spelling)
             NIL
      else (\LoopsDwim originalValue (LET ((FIXSPELL.UPPERCASE.QUIET T))
                                       (DECLARE (SPECVARS FIXSPELL.UPPERCASE.QUIET))
```

```
                                        (FIXSPELL originalValue NIL possibleValues 'NO-MESSAGE NIL NIL
                                               'PICKONE T))
                containingForm tail])
```

(\LoopsDwim

```
[LAMBDA (originalValue correctValue containingForm tail)       (* smL "13-Aug-86 16:56")

        (* * Make the change if we should, according to DWIM setting)

   (if correctValue
       then

        (* * Print out a msg about the {proposed} translation)

           (printout NIL originalValue)
           (if containingForm
               then (printout NIL " {in ")
                    (LVLPRIN1 containingForm NIL 3 4)
                    (printout NIL "}"))
           (printout NIL " -> " correctValue)
           (if (NULL APPROVEFLG)
               then (printout NIL T))

        (* * Find out if we should make the translation)

           (if (OR (NULL APPROVEFLG)
                   (EQ 'Y (ASKUSER DWIMWAIT FIXSPELLDEFAULT " ? ")))
               then

        (* * Make the correction)

                    (if tail
                        then (/RPLACA tail correctValue))

        (* * Return the corrected value if we want to make the correction)

                    correctValue])
```

(\FixSelectorSpelling

```
[LAMBDA (original possibleValues containingForm tail)          (* smL "13-Aug-86 16:44")

        (* * Try to correct the original spelling.)

   (if (NULL DWIMFLG)
       then

        (* * DWIM disabled, so don't even try to correct the spelling)

          NIL
     else (LET [(correctedValue (LET ((FIXSPELL.UPPERCASE.QUIET T))
                                     (DECLARE (SPECVARS FIXSPELL.UPPERCASE.QUIET))
                                     (FIXSPELL original NIL possibleValues 'NO-MESSAGE NIL NIL 'PICKONE T]
                (if correctedValue
                    then

        (* * Print out a msg about the {proposed} translation)

                         (printout NIL original)
                         (if containingForm
                             then (printout NIL " {in ")
                                  (LVLPRIN1 (CONS '_ containingForm)
                                         NIL 3 4)
                                  (printout NIL "}"))
                         (printout NIL " -> " correctedValue)
                         (if (NULL APPROVEFLG)
                             then (printout NIL T))

        (* * Find out if we should make the translation)

                         (if (OR (NULL APPROVEFLG)
                                 (EQ 'Y (ASKUSER DWIMWAIT FIXSPELLDEFAULT " ? ")))
                             then

        (* * Make the correction)

                                  (if tail
                                      then (/RPLACA tail correctedValue))
                                  correctedValue])
```

(GetMethodObj

```
[LAMBDA (class selector createIfNotFoundFlg)                   ; Edited 17-Jun-87 16:08 by smL

;;; Method objects have names of form className.selector.  If not found, and createIfNotFoundFlg=T then create a new one, filling in className and
;;; selector
```

```
    (LET ((methName (MethName class selector)))
         (OR ($! methName)
             (if createIfNotFoundFlg
                 then (LET ((obj (_ ($ Method)
                                    New methName)))
                           (PutValueOnly obj 'className (ClassName class))
                           (PutValueOnly obj 'selector selector)
                           (PutValueOnly obj 'category (LET [(superCategory (GetMethod class selector
                                                                                       'category]
                                                             (if (NoValueFound superCategory)
                                                                 then (LIST (ClassName class))
                                                               else superCategory)))
                           obj])
```

(GetMethodObj!

```
  [LAMBDA (class selector)                                          (* smL "30-Oct-86 17:15")
```

;;;

;;; Return the method object for this class and selector,

;;; no matter where the method is inherieted from

;;;

```
    (LET [(holding-class (for c in-supers-of class thereis (GetMethodObj c selector]
         (if holding-class
             then (GetMethodObj holding-class selector)
           else NIL])
```

(IVSublis

```
  [LAMBDA (value alist)                                             ; Edited 14-Aug-90 16:22 by jds
                                                                    (* Copy value putting in substitutions for items on alist.
                                                                    Called from Object.Sublis)
    (PROG ((pair (FASSOC value alist)))
          (RETURN (COND
                     (pair [COND
                              ((NULL (CDR pair))
                               (COND
                                  ((type? instance value)           (* This will fix up alist as a side effect)
                                   (_ value Sublis alist))
                                  (T (RPLACD pair (LIST (IVSublis value alist]
                           (CADR pair))
                     [(LISTP value)
                      (COND
                         ((EQ '* (CAR value))                       (* A comment)
                          (APPEND value))
                         (T (CONS (IVSublis (CAR value)
                                            alist)
                                  (IVSublis (CDR value)
                                            alist]
                     ((type? annotatedValue value)
                      (create annotatedValue
                              annotatedValue _ (IVSublis (fetch (annotatedValue annotatedValue) of value)
                                                         alist)))
                     (T value])
```

(METHCOM

```
  [NLAMBDA MS                                                       ; Edited 30-Sep-87 16:14 by smL
```

;;; Computes file package commands for METHODS

```
    (LET ((instList (for M in MS when (OR ($! M)
                                          (PROGN (printout \TopLevelTtyWindow M " in METHODS list not an object.
                                                           ")
                                                 NIL))
                       collect M)))
```

;; Don't need to (explicity) dump out the method-object unless it contains info that isn't also contained in the method-body.

```
         `((P (\BatchMethodDefs))
           (INSTANCES ,@(for M in instList
                           when [LET* [(method-object ($! M))
                                       (method-ivs (AND method-object (_ method-object ListAttribute!
                                                                         'IVS]
                                      (for iv in method-ivs
                                         thereis (AND [NOT (MEMB iv '(className selector method args doc]
                                                      (NOT (NotSetValue (GetIVHere method-object iv]
                           collect M))
           (METHOD-FNS ,@(for M in instList when (EQ M (@ ($! M)
                                                          method))
                            collect M))
           (P (\UnbatchMethodDefs])
```

(MapIVs

```
[LAMBDA (self mapfn)                                    (* smL "15-Jan-87 16:37")

        (* * maps through self applying (mapfn self ivName propName) for all IVnames and all props, including NIL for the value
        itself)

    (for ivName in (_ self ListAttribute 'IVs) do (for propName in (CONS NIL (_ self ListAttribute 'IVPROPS ivName)
                                                                            )
                                                      do (APPLY* mapfn self ivName propName])
```

(MapIVs!

```
[LAMBDA (self mapfn)                                    (* smL "11-Apr-86 15:02")

        (* maps through self applying (mapfn self ivName propName) for all IVnames and all props including inherited ones and NIL
        for the value itself)

    (for ivName in (_ self ListAttribute! 'IVs) do (for propName in (CONS NIL (_ self ListAttribute! 'IVPROPS
                                                                                ivName))
                                                       do (APPLY* mapfn self ivName propName])
```

(NewWithValues

```
[LAMBDA (class description)                             (* dgb%: "24-DEC-83 12:37")

        (* * Creates a new instance, substituting values given explicitly in description Does not initialize variables in the usual way.)

    (FillIVs NIL class description])
```

(OldClass

```
[NLAMBDA (name)                                         (* edited%: "19-Dec-84 21:59")
    (OR (GetObjectRec name)
        (NewClass name])
```

(SendMessageNotUnderstood

```
[LAMBDA (messageArguments selector)                     (* dgb%: "14-Dec-84 10:07")

        (* * message arguments include object as first of messageArguments.
        These are the arguments pased to the function implementing the method.
        The selector is not included)

    (COND
       ((EQ selector 'MessageNotUnderstood)
        (HELP "MessageNotUnderstood not understood"))
       (T (_ (CAR messageArguments)
             MessageNotUnderstood selector messageArguments])
```

(SubsTree

```
[LAMBDA (class currentList)                             (* smL "11-Apr-86 14:52")
                                                        (* Compute the SubsTree starting at class given, adding
                                                        elements to currentList)
    [for cl in (_ (GetClassRec class)
                  ListAttribute
                  'Subs)
       do (COND
             ((NOT (FMEMB cl currentList))
              (SubsTree cl (SETQ currentList (NCONC1 currentList cl]
    currentList])
```

(TypeInMethods

```
[LAMBDA (com name type)                                 ; Edited  2-Jun-87 19:14 by smL
    ;; This function is part of the implementation of METHODS as a file package type.  See page 11.31 of the October 83 Interlisp-D manual.
    (LET [(methList (COND
                        ((EQ (CADR com)
                             '*)
                         (EVAL (CADDR com)))
                        (T (CDR com]
         (SELECTQ type
              ((METHODS INSTANCES FNS METHOD-FNS)
                   [SELECTQ name
                        ((NIL T)
                            methList)
                        (COND
                           ((LITATOM name)
                            (FMEMB name methList))
                           (T (INTERSECTION name methList])
              NIL])
```

(WhoHas

```
[LAMBDA (name type files editFlg)                       (* smL "15-Aug-86 14:35")
```

```
        (* * Collect all classes on the files that contain name as a type -
        type is one of Method IV or -
        if editFlg is true, edit the classes/methods)

    (for f in (MKLIST (OR files FILELST)) join (for cl in (FILECOMSLST f 'CLASSES)
                                                  collect [COND
                                                     (editFlg (COND
                                                                 ((FMEMB type '(NIL Method METHOD))
                                                                  (_ ($! cl)
                                                                     EditMethod name))
                                                                 (T (_ ($! cl)
                                                                       Edit]
                                                           cl
                                                  when (SELECTQ type
                                                       ((NIL Method METHOD)
                                                            (FindLocalMethod ($! cl)
                                                                  name))
                                                       (IV (_ ($! cl)
                                                              HasIV name))
                                                       (CV (_ ($! cl)
                                                              HasCV name))
                                                       NIL])

)

;;; The kenel classes themselves

(RPAQQ KERNELCLASSES (AbstractClass Class DestroyedClass DestroyedObject MetaClass Method Object Tofu))

(DEFCLASSES AbstractClass Class DestroyedClass DestroyedObject MetaClass Method Object Tofu)

(DEFCLASS AbstractClass (MetaClass MetaClass doc

        (* * Abstract classes are placeholders in the inheritance network, which cannot themselves be instantiated.)

                                Edited%:                                    (* mjs%: "30-JUN-82 16:41")
                                )
    (Supers MetaClass))

(DEFCLASS Class (MetaClass MetaClass doc

        (* * This is the default metaClass for all classes)

                            Edited%:                                        (* smL "18-Sep-86 15:04")
                            )
    (Supers Object))

(DEFCLASS DestroyedClass (MetaClass AbstractClass Edited%:                  (* kmk%: "13-Dec-84 15:46")
                                 doc                                        (* Becomes the class for any destroyed class)
                                 )
    (Supers AbstractClass))

(DEFCLASS DestroyedObject (MetaClass Class Edited%:                         (* TheCollaborators%: "15-Oct-84 16:23"))
    (Supers Object))

(DEFCLASS MetaClass (MetaClass MetaClass Edited%:                           (* mjs%: "30-JUN-82 16:38"))
    (Supers Class))

[DEFCLASS Method (MetaClass Class doc                                       (* Connects class to function implementing method, plus
                                                                            properties)
                         Edited%:                                           (* smL " 9-May-86 14:40")
                         )
    (Supers Object)
    (ClassVariables (ivProperties (doc args category)
                           doc                                              (* names of IVs which should be made properties of the method)
                           ))
    (InstanceVariables (className NIL doc                                   (* name of class in which this method appears))
           (selector NIL doc                                                (* An atom which is the selector for the method;))
           (method NIL doc

        (* Atom name of function which does the work other properties of this IV are properties of the method)
)
           (args NIL doc                                                    (* arguments of the method))
           (doc NIL doc                                                     (* documentation of the method))
           (category NIL doc                                                (* if a LITATOM, a public method.
                                                                            If a LIST, internal)]

(DEFCLASS Object (MetaClass Class doc                                       (* Default behavior stored here.)
                         Edited%:                                           (* dgb%: "16-Nov-84 13:46")
                         )
    (Supers Tofu))

(DEFCLASS Tofu (MetaClass AbstractClass doc                                 (* Minimum super for objects in system.)
                      Edited%:                                              ; Edited 30-Nov-87 09:24 by jrb:
```

```
                        )
       (Supers))

(RPAQ? Viewed-Categories '(Public))

(RPAQ? DumpMethodsInClass )

(\BatchMethodDefs)

(METH Class AllInstances NIL "Find all instances that you can. Used IndexedObject if possible" (category (Class)
                                                                                                        ))

(METH Class AllMethodCategories (includeCategories okSelectors)
      "Return a list of all categories for methods of this class"
      (category (Class)))

(METH Class CVMissing (object varName propName typeFlag newValue)
      "Reference to an Undefined CV. Generate an error."
      (category (Class)))

(METH Class CVValueMissing (object varName propName typeFlag)
      "Returns NotSetValue if a value is not found in a CV"
      (category (Class)))

(METH Class CategorizeMethods (categorization)
      "Change the categorization according to the categorization argument, which must be in format: ((category
      (selectors ...)) ...) --- If this argument isn't provided, then prompt the user to EDIT a form in this
      syntax that represents the current categorization --- Note that a selector can be in more than one
      category"
      (category (Class)))

(METH Class ChangeMethodCategory (selector newCategory)
      "Change the category of a selected method"
      (category (Class)))

(METH Class CreateInstance (oldObject oldInstanceFlg)
      "Creates the data structure for an instance based on the class. If oldObject is given, then just makes
      it blank . If oldInstanceFlg=T, then it does not mark the object as modified."
      (category (Class)))

(METH Class DefMethod (selector args exp file methodType)
      "Adds a method for selector to class. If args and expr are NIL, puts user into editor"
      (category (Class)))

(METH Class DelFromFile NIL "Delete a class from a file" (category (Object)))

(METH Class EM! (selector)
      "Edit in place, make local or specialize method"
      (category (Class)))

(METH Class Edit (commands)
      "Use Interlisp editor on source of class"
      (category (Object)))

(METH Class Edit! (commands)
      "Use Interlisp editor on source of class including inherited values"
      (category (Class)))

(METH Class EditMethod (selector commands okCategories)
      "Finds the function associated with selector in class, and calls editor on it"
      (category (Class)))

(METH Class EditMethodObject (selector)
      "Edit the object corresponding to the method"
      (category (Class)))

(METH Class FetchMethod (selector)
      "Find the name of the function which implements this method in this class"
      (category (Class)))

(METH Class FileIn (fileSource)
      "Create an instance from expr, which was read from a file"
      (category (Class)))

(METH Class FileOut (file)
      "Print out a class definition to a file."
      (category (Object)))

(METH Class Fringe NIL "List classes which have now subclasses" (category (Class)))

(METH Class GetClassProp (propName)
      "Maps through a class and its metaClasses in order to find the value of a property on the class itself.
       Returns if property is set, or NotSetValue if none found. If propName is NIL, then returns the
      metaClass of the class."
      (category (Class)))

(METH Class HasAttribute (type name propname)
```

```
      "Similar to HasItem, but with right semantics from start."
      (category (Class)))

(METH Class HasAttribute! (type name propname)
      "Similar to HasItem!, but with right semantics from start."
      (category (Class)))

(METH Class HasItem (itemName prop itemType)
      "Generalized Has predicate for IVS, CVS, METHODS."
      (category (Class)))

(METH Class IndexedInstances NIL "Find IndexedInstances of this class" (category (Class)))

(METH Class Initialize (self)
      "Run initial expression for IVs with active value defaults with ls = INITIAL or gfn = AtCreation. In
      that case, makes a value which is the expression in GetFn. Other active values are copied to instance
      by PutValue"
      (category (Class)))

(METH Class InstallEditSource (editedDescription)
      "Make class conform to new edited description"
      (category (Object)))

(METH Class ListAttribute (type name)
      "Fn to list local parts of a class."
      (category (Object)))

(METH Class ListAttribute! (type name verboseFlg)
      "Recursive version of ListAttribute message. Omits things inherited from Object and Class unless
      verboseFlg is T. Sets it to T for Class and Object"
      (category (Object)))

(METH Class MakeEditSource NIL "Make a source for editing the class" (category (Object)))

(METH Class MakeFileSource (file)
      "Creates a list structure source of a class to be dumped on a file"
      (category (Object)))

(METH Class MakeFullEditSource NIL "Make source including inherited values" (category (Class)))

(METH Class MethodCategories (selector)
      "Return the category list of a method"
      (category (Class)))

(METH Class MoveToFile (file)
      "Move this class to a file"
      (category (Object)))

(METH Class MoveToFile! (file fromfiles)
      "Move this class and all its subs to file"
      (category (Class)))

(METH Class New (name arg1 arg2 arg3 arg4 arg5)
      "Creates an instance of a particular class. The variable name if given is used to name the object."
      (category (Class)))

(METH Class NewClass (init1 init2 init3)
      "Just returns newly created class"
      (category (Class)))

(METH Class NewWithValues (description)
      "Create a new instance of the class, with initial IV values given by the description."
      (category (Class)))

(METH Class Old (fileSource)
      "Find an old object or create a new one with this uid"
      (category (Class)))

(METH Class PickSelector (title okCategories okSelectors includeGenerics?)
      "Let the user pick a defined method selector for this class"
      (category (Class)))

(METH Class Prototype (newProtoFlg)
      "Find an instance of class on CV Prototype, or create an puts one there. Used to send messages for
      effect to a prototype object If newProtoFlg=T then make sure a new prototype is created"
      (category (Class)))

(METH Class Rename (newName)
      "Same as SetName.  Classes can have only one name"
      (category (Object)))

(METH Class RenameMethod (oldSelector newSelector)
      "Rename selector, and change function name"
      (category (Class)))

(METH Class ReplaceSupers (supers)
      "replace supers of class by new supers list"
```

```
      (category (Class)))

(METH Class SelectorsInCategories (okCategories okSelectors)
      "Return a sorted list of selectors for the class that match the indicated categories"
      (category (Class)))

(METH Class SelectorsWithBreak NIL "Returns a list of selectors whose implementations have a BREAK"
      (category (Class)))

(METH Class SetName (newClassName)
      "Change the newClassName of the class, forgetting old name. Change the names of all methods which are
      of the form oldName.selector"
      (category (Object)))

(METH Class Specialize (newName)
      "Creates a class with name newName with self as its only super. If newName is NIL, then makes up an
      unused name consisting of current name followed by integer"
      (category (Class)))

(METH Class SpecializeMethod (selector file)
      "Specialize method for selector given"
      (category (Class)))

(METH Class SubClasses NIL "Returns a list of immediate subclasses currently known for this class."
      (category (Class)))

(METH Class Subclass (super)
      "Is self a subclass of super? If it is, return super, else NIL."
      (category (Class)))

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

(METH Class UnSetName (name)
      "Unname entity"
      (category (Object)))

(METH DestroyedClass Destroy NIL "you don't have to do anything to destroy a destroyed class" (category (Object)
                                                                                                        ))

(METH DestroyedClass Destroy! NIL "Similar to DestroyedObject.Destroy! -- Nothing to do once one is dstroyed"
      (category (Object)))

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

(METH DestroyedClass DestroyInstance (self)
      "smash back pointer to entity rec, the list of vars and var descriptions"
      (category (Class)))

(METH DestroyedClass SubClasses NIL "Non subclasses" (category (Class)))

(METH DestroyedObject Destroy! NIL "Do nothing. I am already destroyed" (category (Object)))

(METH MetaClass CreateClass (name supers)
      "Create the data object for a class, checking the inputs"
      (category (MetaClass)))

(METH MetaClass DestroyInstance NIL "Destroy the class specified by smashing its contents" (category (Class)))

(METH MetaClass New (name supers init1 init2 init3)
      "New method for MetaClass. Since MetaClass is its own metaClass, this needs to work correctly whether
      the self is Class or MetaClass or a subClass of MetaClass. Work is done by DefineClass in LOOPS."
      (category (Class)))

(METH MetaClass NewWithValues (selector superFlg)
      "Create a new class, filled in with the given descriptor"
      (category (Class)))

(METH Method ChangeClassName (newClassName)
      "Change name of class -- called  when className is changed"
      (category (Method)))

(METH Method ChangeName (oldMethName newMethName newSelector)
      "Change the name of the method and update the file"
      (category (Method)))

(METH Method DelFromFile NIL "Delete from a file as a method" (category (Object)))

(METH Method EditMethod NIL "Edit the method defintion" (category (Method)))

(METH Method FileOut (file)
      "Print out filesource for methods"
      (category (Object)))
```

```
(METH Method MakeFileSource NIL "Return a form that will redefine the method object when read back in from a
      file - The form is (METH <className> <selector> <method> <args> <doc> . <otherProps>)" (category (Object))
      )

(METH Method ObjectModified (name reason)
      "sent when self modified in any way"
      (category (Object)))

(METH Method OldInstance NIL "Adds Method to those known in class." (category (Object)))

(METH Method UnSetName (name)
      "Unname entity"
      (category (Object)))

(METH Object ChangeClass (newClass)
      "Change object to be new class, keeping old IVs"
      (category (Object)))

(METH Object Class NIL "Returns class of object" (category (Object)))

(METH Object ClassName NIL "Get the name of the class of self" (category (Object)))

(METH Object ConformToClass NIL "Make object have only those IVs that are defined in class" (category (Object)))

(METH Object DelFromFile NIL "Remove object from any file it is on" (category (Object)))

(METH Object Destroy NIL "All the work is normally done by the class in DestroyInstance" (category (Object)))

(METH Object Destroy! NIL "Same as Object.Destroy except when self is a class" (category (Object)))

(METH Object DoMethod (selector class arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10)
      "Message form of DoMethod. Maximum of 10 arguments allowed"
      (category (Object)))

(METH Object Edit (commands)
      "Use Interlisp editor on source of object"
      (category (Object)))

(METH Object FileOut (file)
      "Print out file source on file so it can be reread"
      (category (Object)))

(METH Object HasAttribute (type name propname)
      "Similar to HasItem, but with right semantics from start."
      (category (Object)))

(METH Object HasAttribute! (type name propname)
      "Similar to HasItem!, but with right semantics from start."
      (category (Object)))

(METH Object IVMissing (varName propName typeFlg newValue)
      "Called from code like GetInstanceIV when there is no IV varName. If varName is an IV the class, or
      user requests, then Object.IVMisssing adds IV to the instance. Returns what GetInstanceIV (or whatever
      ) whould have returned. - - Added feature: if there is a CV by the same name with :allocation
      property, then an IV is automatically created..."
      (category (Object)))

(METH Object IVValueMissing (varName propName typeFlg newValue)
      "No value found in the instance or in locally in the class if self is a class"
      (category (Object)))

(METH Object InstallEditSource (editedDescription)
      "Blank instance and make it conform to new description"
      (category (Object)))

(METH Object InstallFileSource (fileSource)
      "Fill the given instance based on expression fileSource read from file, and name it"
      (category (Object)))

(METH Object ListAttribute (type name)
      "For type= IVs, list the iv names in instance. For IVProps lists IV properties for name found in
      instance. Otherwise lists properties inherited from class"
      (category (Object)))

(METH Object ListAttribute! (type name verboseFlg)
      "Recursive form of ListAttribute for objects. Omits things inherited from Object unless verboseFlg is
      T."
      (category (Object)))

(METH Object MakeEditSource NIL "Get a lst showing all instance variables, values, and properties for Editing"
      (category (Object)))

(METH Object MakeFileSource (file)
      "create a list structure source to be dumped on a file"
      (category (Object)))

(METH Object MessageNotUnderstood (selector messageArguments superFlg)
```

```
      "Invoked when a selector is not found for an object during a message sending operation. Attempts to do
      spelling correction on the selector. Forwards the message on to Tofu if it fails (causing a break we
      hope)"
      (category (Tofu)))

(METH Object MoveToFile (file)
      "Move this object to a file"
      (category (Object)))

(METH Object NewInstance (name arg1 arg2 arg3 arg4 arg5)
      "This allows initialization by the classes of objects themselves, rather than going to a metaClass"
      (category (Object)))

(METH Object ObjectModified (name reason)
      "sent when self modified in any way"
      (category (Object)))

(METH Object OldInstance NIL "Allow fixup of object after reading in." (category (Object)))

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

(METH Object Rename (newName oldNames)
      "Remove old name(s), and give it new name"
      (category (Object)))

(METH Object SaveInstance (name arg1 arg2)
      "Used to save the instance on a file. Justs marks it as changed as a default"
      (category (Object)))

(METH Object SaveInstance? (file outInstances)
      "Save this instance if referred to by another unless it is already on this list to be saved"
      (category (Object)))

(METH Object SetName (name)
      "Call on NameEntity"
      (category (Object)))

(METH Object UnSetName (name)
      "Unname entity"
      (category (Object)))

(METH Tofu MessageNotUnderstood (selector messageArguments superFlg)
      "Pretends it understands Understands and also returns NIL for PrintOn so that PrintInstance will use
      the default datatype printing mechanism."
      (category (Tofu)))

(METH Tofu MethodNotFound (selector)
      "Standard form for calling MessageNotUnderstood"
      (category (Tofu)))

(METH Tofu SuperMethodNotFound (selector classOfSendingMethod)
      "No super method found when starting search from classOfSendingMethod"
      (category (Tofu)))

(Method ((Class AllInstances) self)                                    ; smL 19-May-86 09:49
   "Find all instances that you can. Used IndexedObject if possible"
   [COND
      ((_ self Subclass ($ IndexedObject))
       (FindIndexedObjects (ClassName self)))
      (T (LET ((objList (CONS)))
              (DECLARE (SPECVARS objList))
              [MapObjectUID (FUNCTION (LAMBDA (VAL KEY)
                                      (COND
                                         ((_ VAL InstOf self)
                                          (TCONC objList VAL]
              (CAR objList])

(Method (((Class AllMethodCategories) self includeCategories okSelectors)
                                                                       ; smL 30-Oct-86 17:15
   "Return a list of all categories for methods of this class"
   (OR (NULL includeCategories)
       (LISTP includeCategories)
       (ERROR "Not a list of categories:" includeCategories))
   (for sel in (OR okSelectors (_ self ListAttribute 'Selectors)) bind (allCategories _ (COPY includeCategories))
      do (for cat inside (@ (GetMethodObj! self sel)
                            category)
            when (NOT (MEMB cat allCategories)) do (SETQ allCategories (NCONC1 allCategories cat)))
      finally (RETURN allCategories)))

(Method (((Class CVMissing) self object varName propName typeFlag newValue)
                                                                       ; smL 21-Apr-86 10:32
   "Reference to an Undefined CV. Generate an error."
   (LoopsHelp varName " not a CV of " self))

(Method (((Class CVValueMissing) self object varName propName typeFlag)
```

```
                                                                    ; smL  5-May-86 11:26
   "Returns NotSetValue if a value is not found in a CV"
   (if propName
       then NoValueFound
     else NotSetValue))

(Method ((Class CategorizeMethods) self categorization)             ; smL 31-Oct-86 14:01
   "Change the categorization according to the categorization argument, which must be in format: ((category
   (selectors ...)) ...) --- If this argument isn't provided, then prompt the user to EDIT a form in this syntax
   that represents the current categorization --- Note that a selector can be in more than one category"

;;;
   ;;
   ;; First, make sure we have a legit categorization description
   ;;
   (if (NULL categorization)
       then [SETQ categorization (for cat in (_ self AllMethodCategories '(Any Public Internal))
                                    bind (selectors _ (_ self ListAttribute 'Selectors))
                                    collect (LIST cat (_ self SelectorsInCategories cat selectors]
            (EDITE categorization))                                 ; Strip out any pseudo-categories
   (SETQ categorization (for cat-descr in categorization when [NOT (MEMB (CAR cat-descr)
                                                                         '(Any Public]
                              collect cat-descr))
   ;;
   ;; Now apply that categorization
   ;;
   [for sel in (_ self ListAttribute 'Selectors) when (for each in categorization thereis (MEMB sel (CADR each)))
      do (_ self ChangeMethodCategory sel (for each in categorization when (MEMB sel (CADR each))
                                             collect (CAR each])

(Method ((Class ChangeMethodCategory) self selector newCategory)
                                                                    ; smL 21-Aug-86 15:11
   "Change the category of a selected method"
   [SETQ selector (OR selector (_ self PickSelector (CONCAT "Change Method Category: " (ClassName self]
   (if (AND selector (GetMethodObj self selector))
       then [SETQ newCategory (OR newCategory
                                  (LET [(choice (MENU (create MENU
                                                              ITEMS _ (CONS '*other*
                                                                            (for cat
                                                                               in (SORT (_ self
                                                                                           AllMethodCategories))
                                                                               collect (LIST cat)))
                                                              CENTERFLG _ T
                                                              TITLE _ "New method category"]
                                        (if (EQ choice '*other*)
                                            then (PromptRead "New Category: " PROMPTWINDOW)
                                          else (CAR choice]
            (if newCategory
                then (COND
                         ((LISTP newCategory)                       ; Lists replace old categories
                          ;; Categories can only be atoms
                          (if (for c in newCategory thereis (NOT (CL:SYMBOLP c)))
                              then (ERROR "Non-symbol in Category list")
                            else (change (@ (GetMethodObj self selector)
                                            category)
                                         newCategory)))
                         ((CL:SYMBOLP newCategory)                  ; Atoms are added
                          (push (@ (GetMethodObj self selector)
                                   category)
                                newCategory))
                         (T (ERROR "Non-symbol in Category list")))
                       (_ (GetMethodObj self selector)
                          ObjectModified)
                       newCategory)))

(Method ((Class CreateInstance) self oldObject oldInstanceFlg)  ; dgb: 13-OCT-83 22:06
   "Creates the data structure for an instance based on the class. If oldObject is given, then just makes
   it blank . If oldInstanceFlg=T, then it does not mark the object as modified."
   (BlankInstance self oldObject oldInstanceFlg))

(Method ((Class DefMethod) self selector args exp file methodType)
                                                                    ; smL 11-Apr-86 14:48
   "Adds a method for selector to class. If args and expr are NIL, puts user into editor"
   (PROG NIL
         (OR selector [AND (SETQ selector (PromptRead "Type the selector for the new method: "))
                           (OR [NOT (FMEMB selector (_ self ListAttribute 'Methods]
                               (MOUSECONFIRM (CONCAT selector " already exists as a method. Do you want to
                                                     overwrite?"]
             (RETURN (PROMPTPRINT "No method defined.")))
         (RETURN (DefineMethod self selector args exp file methodType))))

(Method ((Class DelFromFile) self)                                  ; smL  8-Apr-87 13:05
```

```
   "Delete a class from a file"
   (LET [(files (WHEREIS (ClassName self)
                         'CLASSES]
        (if files
            then (DELFROMFILES (ClassName self)
                               'CLASSES files)
          else T)))

(Method ((Class EM!) self selector)                                    ; smL 15-Jan-87 16:55
   "Edit in place, make local or specialize method"
   (PROG (items classForMethod)
         (OR selector (SETQ selector (_ self PickSelector (CONCAT "EM!: " (ClassName self))
                                        NIL
                                        (_ self ListAttribute! 'Selectors)
                                        T))
             (RETURN))
         (SETQ classForMethod (_ self WhereIs selector))
         [COND
            ((NULL classForMethod)
             (RETURN (PROMPTPRINT "No class for method")))
            ((NEQ self classForMethod)
             (SELECTQ [MENU (MenuGetOrCreate MakeLocalMethodMenu '(("Make method local" 'MakeLocal "Copy method
                                                                          to this class and edit")
                                                                   ("Edit Method in Place" 'Edit "Edit method in
                                                                          class where found")
                                                                   ("Specialize Method" 'SpecializeMethod
                                                                          "Define a template for a method
                                                                          specialization"]
                      (MakeLocal (PROGN (PROMPTPRINT "Making " selector " local method of " self)
                                        (_ self MakeLocalMethod selector))
                                 (SETQ classForMethod self))
                      (Edit                                            (* Edit in classForMethod))
                      (SpecializeMethod
                          (_ self SpecializeMethod selector))
                      (RETURN NIL]
         (RETURN (_ classForMethod EditMethod selector))))

(Method ((Class Edit) self commands)                                   ; smL 13-May-86 13:14
   "Use Interlisp editor on source of class"
   [PROG (editResult (editSource (_ self MakeEditSource)))
     LP  [SETQ editResult (NLSETQ (EDITE editSource commands (ClassName self)
                                         'CLASSES
                                         'ChangeEditedClass]
         (COND
            ((NULL editResult)                                         (* Returned with STOP)
             NIL)
            ((NULL (CAR editResult))
             (SETQ commands NIL)
             (GO LP)))
         (RETURN (SETQ LASTCLASS (ClassName self])

(Method ((Class Edit!) self commands)                                  ; dgb: 31-OCT-83 09:11
   "Use Interlisp editor on source of class including inherited values"
   [PROG ((editSource (_ self MakeFullEditSource)))
     LP  (COND
            ((NULL (EDITE editSource commands self 'CLASSES 'ChangeEditedClass))
             (SETQ commands NIL)
             (GO LP)))
         (RETURN (SETQ LASTCLASS (ClassName self])

(Method ((Class EditMethod) self selector commands okCategories)
                                                                       ; smL 11-Sep-86 10:06
   "Finds the function associated with selector in class, and calls editor on it"
   (LET* [(selector (OR selector (_ self PickSelector (CONCAT "EditMethod: " (ClassName self))
                                    okCategories)))
          (method (AND selector (FindLocalMethod self selector]
         [if (NULL selector)
             then NIL
           elseif (NULL method)
             then (LET [(allSelectors (_ self ListAttribute! 'Selectors]
                       (if (MEMB selector allSelectors)
                           then                                        (* The method is an inherited one)
                                (if (EQ 'Y (ASKUSER NIL NIL (CONCAT selector " is not a local method of " self
                                                                    ". Should I make it local for editing? ")))
                                    then (SETQ method (_ self MakeLocalMethod selector)))
                         else (LET* [(containingForm (if (AND (BOUNDP '\SendForm)
                                                              (EQ \Obj self)
                                                              (EQ selector 'EditMethod))
                                                         then \SendForm))
                                     (correctedSelector (\LoopsFixSpell selector allSelectors (CONS '_
                                                                                                    containingForm)
                                                                        (if (EQ 'QUOTE (CAR (CADDR containingForm)))
                                                                            then (CDR (CADDR containingForm]
                                (if correctedSelector
                                    then (SETQ selector correctedSelector)
                                         (SETQ method (OR (FindLocalMethod self correctedSelector)
                                                          (_ self MakeLocalMethod correctedSelector]
```

```
         (if (NULL method)
             then NIL
           elseif (NULL (CheckMethodForm self selector method))
             then (PROMPTPRINT method " is not a known function.")
           else (if (GETDEF method 'METHOD-FNS NIL '(NOERROR NOCOPY NODWIM))
                    then (EDITDEF method 'METHOD-FNS NIL commands)
                  else (PROMPTPRINT "Can't find source for " method T)
                       NIL))))

(Method ((Class EditMethodObject) self selector)                ; smL 29-Jan-86 14:04
   "Edit the object corresponding to the method"
   (PROG NIL
         (OR selector [SETQ selector (_ self PickSelector (CONCAT "EditMethodObject: " (ClassName self]
             (RETURN NIL))
         (_ (OR (GetMethodObj self selector)
                (RETURN NIL))
            Edit)))

(Method ((Class FetchMethod) self selector)                     ; dgb: 29-Feb-84 08:50
   "Find the name of the function which implements this method in this class"
   (FetchMethod self selector))

(Method ((Class FileIn) self fileSource)                        ; smL 19-May-86 19:15
   "Create an instance from expr, which was read from a file"
   (LET ((obj (_ self Old fileSource)))
        (_ obj InstallFileSource fileSource)
        (_ obj OldInstance)))

(Method ((Class FileOut) self file)                             ; dgb: 24-Sep-84 12:57
   "Print out a class definition to a file."
   [RESETVAR FIRSTCOL 16 (LET ((source (_ self MakeFileSource)))
                              (COND
                                 ((NULL source)
                                  (HELPCHECK className " is not defined as a class.
                                         Type OK to ignore this class and go on."))
                                 (T                             ; Bind the flag to stop PPing embedded objects in a class
                                    (LET ((ObjectDontPPFlag T))
                                         (printout file "(" .P2 'DEFCLASS " " .FONT LAMBDAFONT .P2 (CADR source)
                                                .FONT DEFAULTFONT 3)
                                         (if PRETTYFLG
                                             then (printout file .PPFTL (CDDR source)
                                                            ")" T T)
                                           else (printout file .PPVTL (CDDR source)
                                                          ")" T]
   self)

(Method ((Class Fringe) self)                                   ; smL 11-Apr-86 15:01
   "List classes which have now subclasses"
   (for C in (_ self ListAttribute! 'Subs) when (NULL (_ (GetClassRec C)
                                                         ListAttribute
                                                         'Subs))
      collect C))

(Method ((Class GetClassProp) self propName)                    ; smL 27-May-86 14:11
   "Maps through a class and its metaClasses in order to find the value of a property on the class itself.
    Returns if property is set, or NotSetValue if none found. If propName is NIL, then returns the
   metaClass of the class."
   [COND
      ((NULL propName)
       (_ self Class))
      (T (bind (class _ self)
               value do (COND
                           [[NOT (NotSetValue (SETQ value (GetClassHere class propName]
                            (RETURN (ExtractRealValue class NIL value propName 'CLASS]
                           ((EQ class (SETQ class (_ class Class)))

          (* * Lots of class in this code. SETQ replaces value of class. Stops at any tight loop such as Metaclass)

                            (RETURN NotSetValue])

(Method ((Class HasAttribute) self type name propname) "Similar to HasItem, but with right semantics from
   start."
   (SELECTQ (U-CASE type)
        ((IV IVPROP NIL)
             [AND (FMEMB name (fetch (class localIVs) of self))
                  (OR (NULL propname)
                      (for N in (CDR (FetchCIVDescr self name)) by (CDDR N) when (EQ N propname)
                         do (RETURN T])
        ((CV CVPROP)
             [AND (FMEMB name (fetch cvNames of self))
                  (OR (NULL propname)
                      (for N in (CDR (FetchCVDescr self name)) by (CDDR N) when (EQ N propname)
                         do (RETURN T])
        ((METHOD SELECTOR)
             (FindLocalMethod self name))
        (PROGN (printout T "Don't know how to look up attribute " type)
               NIL)))
```

```
(Method ((Class HasAttribute!) self type name propname) "Similar to HasItem!, but with right semantics from
   start."
   (OR (_ self HasAttribute type name propname)
       (for super in (Supers self) when (_ super HasAttribute type name propname) do (RETURN T))))

(Method ((Class HasItem) self itemName prop itemType)                    ; mjs: 4-Dec-85 14:26
   "Generalized Has predicate for IVS, CVS, METHODS."
   (SELECTQ itemType
        ((IV IVS)
            (_ self HasIV itemName prop))
        ((CV CVS)
            (_ self HasCV itemName prop))
        ((SELECTOR METHOD SELECTORS METHODS)
            (FindLocalMethod self itemName))
        NIL))

(Method ((Class IndexedInstances) self)                                  ; dgb: 16-May-84 19:56
   "Find IndexedInstances of this class"
   (FindIndexedObjects (ClassName self)))

(Method ((Class Initialize) class self)                                  ; dgb: 18-JAN-83 17:25
   "Run initial expression for IVs with active value defaults with ls = INITIAL or gfn = AtCreation. In
   that case, makes a value which is the expression in GetFn. Other active values are copied to instance
   by PutValue"
   ;; Clean slow code (for varName value in (_ self List (QUOTE IVs)) do (* * for all properties in IV,
   ;; including NIL for IV value, Fire initialization function which exist.) (for prop in (CONS NIL (_ self
   ;; List! (QUOTE IVPROPS) varName)) when (NEQ NotSetValue (SETQ value (FireInit self varName (GetValueOnly
   ;;  self varName prop)))) do (PutValueOnly self varName value prop)))
   (FastClassInitialize class self)
   self)

(Method ((Class InstallEditSource) self editedDescription)               ; smL 13-May-86 13:03
   "Make class conform to new edited description"
   [LET ((className (ClassName self)))
        (COND
           ((CheckClassSource editedDescription className)               ; Dont't install the class if there are errors.  Bounce back to editor
            (RINGBELLS 1)
            ;; JRB - changed from PROMPTPRINT to two PRIN1s because PROMPTPRINT clears the window,so you never get to see error
            ;; message printed by CheckClassSource
            (PRIN1 className PROMPTWINDOW)
            (PRIN1 " not defined  -- bad form
                   " PROMPTWINDOW)
            ;; JRB - This clause knows how to punt from the editor; SEdit punts differently from DEdit (which may not punt this way anymore; need
            ;; to ask Woz about it...)
            (SELECTQ (EDITMODE)
                 (SEDIT (ERROR "Bad Class form
                               Type ^ to return to editor "))
                 (RETFROM 'EDITE NIL)))
           (T (InstallClassSource className editedDescription)
              (PutClass self (EDITDATE NIL INITIALS)
                     'Edited%:])

(Method ((Class ListAttribute) self type name)                           ; smL 23-May-86 10:29
   "Fn to list local parts of a class."
   [SELECTQ (SETQ type (U-CASE type))
        (IVS (APPEND (fetch (class localIVs) of self)))
        (CVS (APPEND (fetch cvNames of self)))
        ((METHODS SELECTORS)
            (\ListFromBlock (fetch selectors of self)))
        (METHODOBJECTS
            (for s in (_ self ListAttribute 'Selectors) collect (MethName self s)))
        (FUNCTIONS (\ListFromBlock (fetch methods of self)))
        ((SUPERS SUPERCLASSES)
            (for x in (fetch localSupers of self) collect (ClassName x)))
        ((SUBS SUBCLASSES)
            (for sub in (fetch subClasses of self) collect (ClassName sub)))
        ((META METACLASS)
            (CONS (ClassName (fetch metaClass of self))
                  (APPEND (fetch otherClassDescription of self))))
        (PROG [(descr (SELECTQ type
                              ((IV IVPROPS NIL)
                                  (FetchCIVDescr self name))
                              ((CV CVPROPS)
                                  (FetchCVDescr self name))
                              ((CLASS)
                                  (fetch otherClassDescription of self))
                              ((METHOD)
                                  (FetchMethodDescr self name))
                              (LoopsHelp type "not recognized part of class"]
              (RETURN (SELECTQ type
                              ((CLASS METHOD)
                                  (for x in descr by (CDDR x) collect x))
                              (for x in (CDR descr) by (CDDR x) collect x])
```

```
(Method ((Class ListAttribute!) self type name verboseFlg)          ; smL 29-Sep-86 11:27
   "Recursive version of ListAttribute message. Omits things inherited from Object and Class unless
   verboseFlg is T. Sets it to T for Class and Object"
   (COND
      ((FMEMB (ClassName self)
              '(Class Object Tofu))
       (SETQ verboseFlg T)))
   (SETQ type (U-CASE type))
   [SELECTQ type
       ((META METACLASS)
           (_ self ListAttribute type))
       ((IVS NIL)
           (APPEND (fetch ivNames of self)))
       ((SUPERS SUPERCLASSES)
           (PROG (name (nameList (CONS)))
                 (MapSupersForm (COND
                                   ((NOT (FMEMB (SETQ name (ClassName class))
                                                (CAR nameList)))
                                    (TCONC nameList name)))
                                self)
                 (RETURN (CDAR nameList))))
       ((SUBS SUBCLASSES)                                           (* List all subclasses of self)
           (SubsTree self))
       (PROG (attList)                                              (* Here if need to recur to collect items.)
             (MapSupersUnlessBadList [COND
                                        (verboseFlg NIL)
                                        (T '(Class Object Tofu]
                   (for item in (_ class ListAttribute type name) do (pushnew attList item))
                   self)
             (RETURN (SELECTQ type
                         (CLASS (DREVERSE attList))
                         attList])

(Method ((Class MakeEditSource) self)                               ; smL 14-May-86 14:56
   "Make a source for editing the class"
   (DECLARE (GLOBALVARS EditClassMethodsFlg))
   [LIST* (CONS 'MetaClass (GetSourceMeta self))
          (CONS 'Supers (GetSourceSupers self))
          (CONS 'ClassVariables (GetSourceCVs self T))
          (CONS 'InstanceVariables (GetSourceIVs self))
          (COND
             (EditClassMethodsFlg (LIST (CONS 'MethodFns (SORT (for I from 0 by 2
                                                                  bind sel (sels _ (fetch selectors of self))
                                                                       (meths _ (fetch (class methods)
                                                                                       of self))
                                                                  first (if (NULL sels)
                                                                            then (RETURN NIL))
                                                                  eachtime (SETQ sel (\GetNthEntry sels I))
                                                                  until (NULL sel) collect (\GetNthEntry meths I])

(Method ((Class MakeFileSource) self file)                          ; smL  4-Apr-86 17:47
   "Creates a list structure source of a class to be dumped on a file"
   (LET (tail (cvs (GetSourceCVs self))
              (ivs (GetSourceIVs self)))
        [SETQ tail (NCONC [AND cvs `((ClassVariables %,. cvs]
                          (AND ivs `((InstanceVariables %,. ivs]
        `(DEFCLASS %, (ClassName self)
                (MetaClass %,. (GetSourceMeta self))
                (Supers %,. (GetSourceSupers self))
                %,. tail)))

(Method ((Class MakeFullEditSource) self)                           ; mjs: 25-Oct-85 15:59
   "Make source including inherited values"
   (LIST (CONS 'MetaClass (GetSourceMeta self))
         (CONS 'Supers (GetSourceSupers self))
         (CONS 'ClassVariables (GetSourceCVs self))
         (CONS 'CVsInherited (GetSourceInhCVs self))
         (CONS 'InstanceVariables (GetSourceIVs self))
         (CONS 'IVsInherited (GetSourceInhIVs self))))

(Method ((Class MethodCategories) self selector) "Return the category list of a method"
   (@ (GetMethodObj self selector)
      category))

(Method ((Class MoveToFile) self file)                              ; smL 17-Apr-86 14:12
   "Move this class to a file"
   [COND
      ([NULL (OR file (SETQ file (SelectFile]
       (printout PROMPTWINDOW "Nothing moved!"))
      (T (MOVEITEM file (ClassName self)
                'CLASSES)
         (for method in (_ self ListAttribute 'METHODS) do          (* First make sure we have the source loaded)
                                                       (LET ((methName (MethName self method)))
                                                            (EnsureFnLoaded methName)
                                                            (MOVEITEM file methName 'METHODS])
```

```
(Method ((Class MoveToFile!) self file fromfiles)              ; smL 11-Apr-86 14:49
   "Move this class and all its subs to file"
   (if (OR file (SETQ file (SelectFile)))
       then (_ self MoveToFile file)
            (for s in (_ self ListAttribute 'Subs) when [OR (NOT fromfiles)
                                                            (for F in fromfiles
                                                               thereis (MEMBER s (FILECOMSLST F 'CLASSES]
              do (_ ($! s)
                    MoveToFile! file))))

(Method ((Class New) self name arg1 arg2 arg3 arg4 arg5)       ; smL 22-May-86 14:30
   "Creates an instance of a particular class. The variable name if given is used to name the object."
   (_ (_ self CreateInstance)
      NewInstance name arg1 arg2 arg3 arg4 arg5))

(Method ((Class NewClass) self init1 init2 init3)              ; dgb: 22-SEP-83 14:19
   "Just returns newly created class"
   self)

(Method ((Class NewWithValues) self description)               ; smL 25-Apr-86 14:30
   "Create a new instance of the class, with initial IV values given by the description."
   (NewWithValues self description))

(Method ((Class Old) self fileSource)                          ; smL 19-May-86 19:14
   "Find an old object or create a new one with this uid"
   [LET ((names (CAR fileSource)))
        (NewObject self (COND
                           ((UIDP names)
                            names)
                           (T (CAR (LAST names])

(Method ((Class PickSelector) self title okCategories okSelectors includeGenerics?)
                                                               ; smL 30-Oct-86 17:55
   "Let the user pick a defined method selector for this class"

;;;

   (DECLARE (SPECVARS Viewed-Categories))
   [LET* ((okCategories (OR okCategories Viewed-Categories))
          [okSelectors (OR okSelectors (_ self ListAttribute 'Selectors]
          (allCategories (_ self AllMethodCategories '(Any Public)
                            okSelectors))
          (otherCategories (LDIFFERENCE allCategories okCategories))
          (selectors (_ self SelectorsInCategories okCategories okSelectors)))
         (if (OR (AND (NULL okSelectors)
                      (NULL includeGenerics?))
                 (AND (NULL selectors)
                      (NULL otherCategories)))
             then (PROMPTPRINT "No methods for class" self)
          else (LET ((selector (NiceMenu (NCONC selectors [for cat in otherCategories
                                                             collect (LIST (CONCAT "** " cat " category" " **")
                                                                           (KWOTE (LIST cat]
                                                 (if includeGenerics?
                                                     then '[("** Generic methods **" '(*generics*]
                                                   else NIL))
                                         title)))
                     (if (LITATOM selector)
                         then selector
                       elseif (EQ (CAR selector)
                                  '*generics*)
                         then [_ self PickSelector title okCategories (SORT (LDIFFERENCE (_ self ListAttribute!
                                                                                           'Methods NIL T)
                                                                                    (_ self ListAttribute!
                                                                                       'Methods]
                         else (_ self PickSelector title (LIST (CAR selector))
                                 okSelectors])

(Method ((Class Prototype) self newProtoFlg)                   ; smL  4-Dec-85 15:43
   "Find an instance of class on CV Prototype, or create an puts one there. Used to send messages for
   effect to a prototype object If newProtoFlg=T then make sure a new prototype is created"
   (PROG (proto descr)
         [COND
            ((SETQ descr (FetchCVDescr self 'Prototype))
             (SETQ proto (CAR descr]
         [COND
            ((OR newProtoFlg (NEQ self (Class proto)))
             (PutCVHere self 'Prototype (SETQ proto (_ self CreateInstance]
         (RETURN proto)))

(Method ((Class Rename) self newName)                          ; smL 31-Oct-86 09:48
   "Same as SetName.  Classes can have only one name"
   (until (NOT (NULL newName)) do (SETQ newName (HELPCHECK "Can't rename a class without specifying name. Type
                                                   RETURN '<newName>
                                                   to continue and rename class: " self)))
   [LET* ((oldName (ClassName self))
          (SELFNAMECATMETHODS (_ self SelectorsInCategories oldName)))
         (_ self SetName newName)
```

```
         (UNMARKASCHANGED oldName 'CLASSES)                    ; so it won't show up in (FILES?)
         (for S in SELFNAMECATMETHODS do (_ self ChangeMethodCategory S (DSUBST newName oldName
                                                                        (_ self MethodCategories S])

(Method ((Class RenameMethod) self oldSelector newSelector)    ; dgb: 18-MAR-83 16:30
   "Rename selector, and change function name"
   (RenameMethod (GoodClassName self)
          oldSelector newSelector))

(Method ((Class ReplaceSupers) self supers)                    ; dgb: 27-AUG-82 13:05
   "replace supers of class by new supers list"
   (OR (EQ 'NoUpdateRequired (InstallSupers self supers))
       (ChangedClass self)))

(Method ((Class SelectorsInCategories) self okCategories okSelectors)
                                                               ; smL 31-Oct-86 13:54
   "Return a sorted list of selectors for the class that match the indicated categories"

;;;

   (SORT (for sel inside (OR okSelectors (_ self ListAttribute 'Selectors))
            when (LET ((cat (@ (GetMethodObj! self sel)
                               category)))
                      (if (EQMEMB 'Any okCategories)
                          then                                 ; every method is in category Any
                               T
                        elseif (AND (EQMEMB 'Public okCategories)
                                    (NOT (EQMEMB 'Internal cat)))
                          then                                 ; Public matches all except Internal
                               T
                        elseif [NOT (NULL (for c inside cat thereis (EQMEMB c okCategories]
                          then                                 ; A category for the selector is present in the list of okCategories
                               T))
            collect sel)))

(Method ((Class SelectorsWithBreak) self)                      ; smL 15-Mar-85 19:07
   "Returns a list of selectors whose implementations have a BREAK"
   (for fn bind pos (cn _ (CONCAT (ClassName self)
                                  "."))
      in BROKENFNS first (SETQ pos (ADD1 (NCHARS cn))) when (STRPOS cn fn 1 NIL T) collect (SUBATOM fn pos)))

(Method ((Class SetName) self newClassName)                    ; smL 12-May-87 18:40
   "Change the newClassName of the class, forgetting old name. Change the names of all methods which are
   of the form oldName.selector"
   (LET* [(oldName (ClassName self))
          (files (WHEREIS oldName 'CLASSES]
         (COND
            ((NULL newClassName)
             (ERROR "NIL not a legal class name"))
            ((EQ oldName newClassName)
             NIL)
            (T (_ self UnSetName oldName)
               (NameEntity self newClassName)
               (replace className of self with newClassName)
               (for class in (fetch subClasses of self) do (change (fetch className of class)
                                                                   (DSUBST newClassName oldName DATUM))
                  when (LISTP (fetch className of class)))
               (ChangedClass self)
               (for file in files do (ADDTOFILE newClassName 'CLASSES file)
                                     (OR [NLSETQ (EDITCALLERS oldName file
                                                              `((ORR (R ($ %, oldName)
                                                                        ($ %, newClassName))
                                                                     NIL]
                                         (printout NIL T "Can't rename " oldName " to " newClassName " on file "
                                                   file T)))
               (for selector in (_ self ListAttribute 'Selectors) do (_ ($! (MethName oldName selector))
                                                                        ChangeClassName newClassName))
               (if (_ self Subclass ($ IndexedObject))
                   then (RenameIndexList oldName newClassName))
               self))))

(Method ((Class Specialize) self newName)                      ; mjs: 21-FEB-83 07:57
   "Creates a class with name newName with self as its only super. If newName is NIL, then makes up an
   unused name consisting of current name followed by integer"
   [OR newName (PROG ((N 0)
                      (myName (ClassName self)))
                 LP   (COND
                         ([GetObjectRec (SETQ newName (PACK* myName (SETQ N (ADD1 N]
                          (GO LP]
   (_ (Class self)
      New newName (LIST (ClassName self))))

(Method ((Class SpecializeMethod) self selector file)          ; smL 30-Oct-86 18:13
   "Specialize method for selector given"
   ;; First, get the selector that we should work on
   (if (NULL selector)
```

```
       then (SETQ selector (_ self PickSelector (CONCAT "SpecializingMethod: " (ClassName self))
                              NIL
                              [SORT (LDIFFERENCE (_ self ListAttribute! 'Methods)
                                                 (_ self ListAttribute 'Methods]
                              T)))
   ;; Now specialize it
   [if selector
       then (LET [(superMethodObj (if (FetchMethod self selector)
                                      then (for class in (Supers self) thereis (GetMethodObj class selector)
                                                finally (RETURN (GetMethodObj class selector]
                  (if superMethodObj
                      then (DefineMethod self selector (COPY (@ superMethodObj args))
                                         [COPY `(,(OR (@ superMethodObj doc)
                                                      (CONCAT "Specialization"))
                                                  (_Super
                                                   self
                                                   ,selector
                                                   ,@(COPY (@ superMethodObj args]
                                         file)
                             (change (@ (GetMethodObj self selector)
                                        category)
                                     (@ superMethodObj category))
                             (_ self EditMethod selector)
                    else (PROMPTPRINT "No method for selector"])

(Method ((Class SubClasses) self)                                    ; smL 11-Oct-85 16:29
   "Returns a list of immediate subclasses currently known for this class."
   (COPY (fetch subClasses of self)))

(Method ((Class Subclass) self super)                                ; smL 18-Mar-85 14:14
   "Is self a subclass of super? If it is, return super, else NIL."
   (MapSupersForm (COND
                     ((EQ class superClass)
                      (RETURN class)))
          self
          (superClass (GetClassRec super))))

(Method ((Class TraceMethod) self selector)                          ; smL  8-Apr-87 19:38
   "Trace selected method, or give choice if selector is NIL"
   [SETQ selector (OR selector (_ self PickSelector (CONCAT "Trace Method: " (ClassName self))
                                  NIL
                                  (LDIFFERENCE (_ self ListAttribute 'Selectors)
                                               (_ self SelectorsWithBreak]
   (COND
      [selector (PROMPTPRINT (CONS 'Tracing (TraceMethod (ClassName self)
                                                         selector]
      (T 'NothingTraced)))

(Method ((Class UnSetName) self name)                                ; smL  8-Apr-87 13:14
   "Unname entity"
   [LET ((names (DeleteObjectName self name)))
        (if names
            then (for name inside names collect (LET [(files (WHEREIS name 'CLASSES]
                                                     (UNMARKASCHANGED name 'CLASSES)
                                                     (if files
                                                         then (DELFROMFILES name 'CLASSES files)
                                                       else T])

(Method ((DestroyedClass Destroy) self)                              ; edited: 20-Dec-84 11:31
   "you don't have to do anything to destroy a destroyed class"
   self)

(Method ((DestroyedClass Destroy!) self)                             ; edited: 19-Dec-84 09:21
   "Similar to DestroyedObject.Destroy! -- Nothing to do once one is dstroyed"
   self)

(Method ((DestroyedClass DestroyClass) self classToDestroy)          ; smL 21-May-86 11:35
   "Destroy the class specified by smashing its contents"            ; First delete from knowledge of file system
   (DELDEF (ClassName classToDestroy)
           'CLASSES)                                                 ; Remove from subClasses lists of each super.
   (for super in-supers-of classToDestroy do (replace subClasses of super
                                                with (for sub in DATUM when (NEQ classToDestroy (COND
                                                                                                   ((LISTP sub)
                                                                                                    (CAR sub))
                                                                                                   (T sub)))
                                                                                                   collect sub)))
                                                                     ; smash back pointer to the list of vars and var descriptions
   (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 supers of classToDestroy with (LIST ($ DestroyedObject)))
   (replace metaClass of classToDestroy with ($ DestroyedClass))
   (DeleteObjectUID classToDestroy)
   'DestroyedClass)

(Method ((DestroyedClass DestroyInstance) class self)                ; smL 21-May-86 11:21
```

```
   "smash back pointer to entity rec, the list of vars and var descriptions"
   (replace class of self with ($ DestroyedObject))
   (replace iNames of self with NIL)
   (replace iDescrs of self with NIL)
   (replace otherIVs of self with NIL)
   (DeleteObjectUID self))

(Method ((DestroyedClass SubClasses) self)                    ; dgb:  5-OCT-83 07:56
   "Non subclasses"
   NIL)

(Method ((DestroyedObject Destroy!) self)                     ; dgb: 27-MAY-83 11:44
   "Do nothing. I am already destroyed"
   self)

(Method ((MetaClass CreateClass) self name supers)            ; dgb: 22-SEP-83 14:17
   "Create the data object for a class, checking the inputs"
   (DefineClass name supers self))

(Method ((MetaClass DestroyInstance) classToDestroy)          ; smL 21-May-86 11:36
   "Destroy the class specified by smashing its contents"     ; First delete from knowledge of file system
   (DELDEF (ClassName classToDestroy)
           'CLASSES)                                          ; 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 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 supers of classToDestroy with (LIST ($ DestroyedObject)))
   (replace metaClass of classToDestroy with ($ DestroyedClass))
   (DeleteObjectUID classToDestroy)
   'DestroyedClass)

(Method ((MetaClass New) self name supers init1 init2 init3)  ; dgb: 22-SEP-83 14:20
   "New method for MetaClass. Since MetaClass is its own metaClass, this needs to work correctly whether
   the self is Class or MetaClass or a subClass of MetaClass. Work is done by DefineClass in LOOPS."
   (_ (_ self CreateClass name supers)
      NewClass init1 init2 init3))

(Method ((MetaClass NewWithValues) self selector superFlg)    ; smL 25-Apr-86 14:32
   "Create a new class, filled in with the given descriptor"
   (NewWithValues self selector superFlg))

(Method ((Method ChangeClassName) self newClassName)          ; smL 14-Feb-86 16:06
   "Change name of class -- called  when className is changed"
   (PROG (newMethName (oldMethName (GetObjectName self))
              (selector (@ selector)))

;;; JRB - removed references to method type and changed arg finding

         (SETQ newMethName (MethName newClassName selector))
         [COND
            ((EQ oldMethName (@ method))
             [LET* [[def (GETDEF oldMethName 'METHOD-FNS NIL '(NOCOPY]
                    (args (COND
                              ((LISTP (CAADR def))            ; new form for Methods.  (Method (ClassName selector)  . args)
                               (CDADR def))
                              (T (CAR def]
                   (_ (GetClassRec newClassName)
                      DefMethod selector args (CDDR def)
                      NIL)
                   (EVAL (CL:MULTIPLE-VALUE-BIND (cname sel args decls forms doc quals)
                             (PARSE-METHOD-BODY def)
                           (PACK-METHOD-BODY newClassName sel args decls forms doc quals))]
             (_@
              method newMethName)
             (PUTD oldMethName NIL))
            (T (AddMethod (GetClassRec newClassName)
                      selector
                      (@ method]
         (_@
          className newClassName)
         (_ self ChangeName oldMethName newMethName selector)
         (RETURN newMethName)))

(Method ((Method ChangeName) self oldMethName newMethName newSelector)
                                                              ; smL 15-Aug-86 16:47
   "Change the name of the method and update the file"
   (LET [(files (WHEREIS oldMethName 'METHODS]
        (UNMARKASCHANGED (@ method)
               'METHODS)                                      ; So old method won't show up in (FILES?)
```

```
        (_ self UnSetName oldMethName)
        (_ self SetName newMethName)
        (FlushMethodCache)
        (change (@ selector)
               newSelector)
        (for file in files do (ADDTOFILE newMethName 'METHODS file))
        self))

(Method ((Method DelFromFile) self)                                    ; smL  8-Apr-87 13:06
   "Delete from a file as a method"
   (LET [(files (WHEREIS (GetObjectName self)
                         'METHODS]
        (if files
            then (DELFROMFILES (GetObjectName self)
                               'METHODS files)
          else T)))

(Method ((Method EditMethod) self)                                     ; dgb: 27-NOV-83 16:28
   "Edit the method defintion"
   (_ ($! (@ className))
      EditMethod
      (@ selector)))

(Method ((Method FileOut) self file)                                   ; dgb: 30-OCT-83 11:24
   "Print out filesource for methods"
   (PROG (pos (source (_ self MakeFileSource)))
         (printout file "(" .FONT DEFAULTFONT .P2 (CAR source)
                %,)
         (SETQ pos (POSITION file))
         (printout file .P2 (CADR source)
                .FONT LAMBDAFONT %,, .P2 (CADDR source)
                %, .P2 (CADDDR source)
                .FONT DEFAULTFONT .TAB pos .PPVTL (CDDDDR source)
                ")" T))
   self)

(Method ((Method MakeFileSource) self)                                 ; smL 19-Aug-86 15:01
   "Return a form that will redefine the method object when read back in from a file - The form is (METH
   <className> <selector> <method> <args> <doc> . <otherProps>)"
   (LET ((className (ClassName self))
         (name (GetObjectName self))
         (source (IVSource self T)))
        [COND
           ((NEQ className 'Method)
            (PutValue self 'method className 'methodClass)
            (PutValue self 'method (UID self)
                   'UID]
        (for iv in '(className selector args doc) do (SETQ source (DELASSOC iv source)))
        [COND
           ([AND (EQ name (@ method))
                 (NULL (CDDR (FASSOC 'method source]                   ; Has default name and no properties
            (SETQ source (DELASSOC 'method source]
        (CONS 'METH (NCONC (LIST (@ className)
                                 (@ selector)
                                 (@ args)
                                 (@ doc))
                           source))))

(Method ((Method ObjectModified) self name reason)                     ; smL 21-Aug-86 10:40
   "sent when self modified in any way"
   (MARKASCHANGED (OR name (GetObjectName self))
          'METHODS reason))

(Method ((Method OldInstance) self)                                    ; smL 27-Sep-85 11:20
   "Adds Method to those known in class."
   (LET ((class (GetClassRec (@ className)))
         methClass)
        [OR class (AND (HELPCHECK (@ className)
                                  "not a currently defined class.
                                  Cannot add method to class.  Type OK to create class and go on.")
                       (SETQ class (_ ($ Class)
                                      New
                                      (@ className]
        (AddMethod class (@ selector)
               (@ method))
        (AND [SETQ methClass ($! (GetValue self 'method 'methodClass]
             (NEQ (Class self)
                  methClass)
             (_ self ChangeClass methClass))))

(Method ((Method UnSetName) self name)                                 ; smL  8-Apr-87 13:16
   "Unname entity"
   [LET ((names (DeleteObjectName self name)))
        (if names
            then (for name inside names collect (LET [(files (WHEREIS names 'METHODS]
                                                     (UNMARKASCHANGED name 'METHODS)
                                                     (if files
```

```
                                                    then (DELFROMFILES name 'METHODS files)
                                                  else T])

(Method ((Object ChangeClass) self newClass)                    ; dgb: 16-May-84 20:16
   "Change object to be new class, keeping old IVs"
   (PROG [(source (IVSource self))
          (classRec (COND
                        ((type? class newClass)
                         newClass)
                        (T (OR (GetClassRec newClass)
                               (ERROR newClass " not a class for ChangeClass"]
          (FillInst source (BlankInstance newClass self))
          (_ self OldInstance))
   self)

(Method ((Object Class) self)                                   ; dgb: 27-AUG-82 13:07
   "Returns class of object"
   (Class self))

(Method ((Object ClassName) self)                               ; smL 25-Apr-86 14:28
   "Get the name of the class of self"
   (ClassName self))

(Method ((Object ConformToClass) self)                          ; smL 11-Apr-86 15:07
   "Make object have only those IVs that are defined in class"
   (FillInst (for descr in (IVSource self) bind (ivs _ (_ (Class self)
                                                          ListAttribute!
                                                          'IVS))
                when (FMEMB (CAR descr)
                            ivs)
                collect descr)
          (BlankInstance NIL self)))

(Method ((Object DelFromFile) self)                             ; smL  8-Apr-87 13:07
   "Remove object from any file it is on"
   (if (OR (HasUID? self)
           (GETHASH self ObjNameTable))
       then (for name in (GetObjectNames self) do (LET [(files (WHEREIS name 'INSTANCES]
                                                        (if files
                                                            then (DELFROMFILES name 'INSTANCES files)
                                                          else T)))
     else                                                       (* No names, so can't be on any file)
          NIL))

(Method ((Object Destroy) self)                                 ; dgb: 26-DEC-83 22:44
   "All the work is normally done by the class in DestroyInstance"
   (_ (Class self)
      DestroyInstance self))

(Method ((Object Destroy!) self)                                ; dgb: 28-Jan-85 22:35
   "Same as Object.Destroy except when self is a class"
   (_ self Destroy))

(Method ((Object DoMethod) self selector class arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10)
                                                                ; smL 29-May-86 18:00
   "Message form of DoMethod. Maximum of 10 arguments allowed"
   (DoMethod self selector class arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10))

(Method ((Object Edit) self commands)                           ; dgb:  4-OCT-83 11:34
   "Use Interlisp editor on source of object"
   (LET ((EDITSOURCE (_ self MakeEditSource)))
        (if EDITSOURCE
            then (EDITE EDITSOURCE commands (OR (GetObjectName self)
                                                self)
                        'INSTANCES
                        'ChangeEditedInstance)
          else (PROMPTPRINT self "has no IVs, can't edit")))
   self)

(Method ((Object FileOut) self file)                            ; dgb: 30-OCT-83 11:25
   "Print out file source on file so it can be reread"
   (PROG ((source (_ self MakeFileSource))
          pos)
         (if PRETTYFLG
             then ;; Pretty-print it

                  ;; Always bold the third thing in the source.  Assume first is a function to install instance e.g.  DEFINST and second is a
                  ;; clasName The third is a critical identifier.

                  (printout file "(" .FONT DEFAULTFONT .P2 (CAR source)
                         %,)
                  (SETQ pos (POSITION file))
                  (printout file .P2 (CADR source)
                         .FONT LAMBDAFONT %, .P2 (CADDR source)
                         .FONT DEFAULTFONT .TAB pos .PPVTL (CDDDR source)
                         ")" T)
           else ;; Don't pretty-print it
```

```
               (printout file .P2 source T)))
   self)

(Method ((Object HasAttribute) self type name propname) "Similar to HasItem, but with right semantics from
   start."
   (SELECTQ (U-CASE type)
        ((IV IVPROP)
            (AND (OR (FMEMB name (fetch iNames of self))
                     (FASSOC name (fetch otherIVs of self)))
                 [OR (NULL propname)
                     (WithIVPropDescr self name [LAMBDA (self name propDescr)
                                                  (AND propDescr (for pList on (fetch IVPropList of propDescr)
                                                                    by (CDDR pList)
                                                                    thereis (EQ propname (CAR pList]
                               (LAMBDA (self name)
                                 NIL]
                 T))
        ((CV CVPROP)
            (LET* [(class (Class self))
                   (index (FindIndex name (fetch cvNames of class]
                  (AND index [OR (NULL propname)
                                 (for tail on (CDR (GetNth (fetch cvDescrs of class)
                                                           index))
                                    by (CDDR tail) do (COND
                                                         ((EQ propname (CAR tail))
                                                          (RETURN T]
                             T)))
        (_ (Class self)
           HasAttribute type name propname)))

(Method ((Object HasAttribute!) self type name propname)
   "Similar to HasItem!, but with right semantics from start."
   (OR (_ self HasAttribute type name)
       (for super in (Supers (Class self)) thereis (_ super HasAttribute type name))))

(Method ((Object IVMissing) self varName propName typeFlg newValue)
                                                                    ; smL 10-Sep-86 15:32
   "Called from code like GetInstanceIV when there is no IV varName. If varName is an IV the class, or
   user requests, then Object.IVMissing adds IV to the instance. Returns what GetInstanceIV (or whatever
   ) whould have returned. - - Added feature: if there is a CV by the same name with :allocation
   property, then an IV is automatically created..."
   [PROG (fixedName)
     Loop

          (* If the IV is now defined in the class, copy it down to the instance.
          This can happen if the class was changed after the instance was created.)

         (COND
            ((FMEMB varName (_ (Class self)
                               ListAttribute!
                               'IVs))
             (FillIVs self (Class self)
                    (for ivDescr in (IVSource self) when (NOT (EQLENGTH ivDescr 1)) collect ivDescr)
                    T)
             (GO Finish)))

          (* If there is a CV by that name, use it instead of the IV. The method of use is determined by the %:allocation CVProp, one
          of dynamicCached, dynamic, or class. In that case, the initial value is determined by the CVProp %:initForm.)

         [COND
            ((_ (Class self)
                HasCV varName)                                  (* Use the class variable, as dictated by its %:allocation property)
             (RETURN (LET ((allocation (GetClassValue self varName '%:allocation))
                           (initForm (GetClassValue self varName '%:initForm))
                           initValue)
                          [COND
                             ((NoValueFound allocation)
                              (SETQ allocation 'class]
                          (SELECTQ allocation
                               (dynamicCached                   (* Create the IV on first access)
                                    (CopyCVToIV self varName)
                                    (SELECTQ typeFlg
                                         ((PutValue PutValueOnly)
                                              (APPLY* typeFlg self varName newValue propName))
                                         ((GetValue GetValueOnly)
                                              (APPLY* typeFlg self varName propName))
                                         (SHOULDNT "Error in IVMissing")))
                               (dynamic                         (* Create the IV if a put, otherwise get the class value)
                                         (SELECTQ typeFlg
                                              ((PutValue PutValueOnly)
                                                   (CopyCVToIV self varName)
                                                   (APPLY* typeFlg self varName newValue propName))
                                              (GetValue (if (AND (NULL propName)
                                                                 (ValueFound initForm))
                                                            then (EVAL initForm)
                                                          else (GetClassValue self varName propName)))
                                              (GetValueOnly (if (AND (NULL propName)
```

```
                                                                    (ValueFound initForm))
                                                               then (EVAL initForm)
                                                              else (GetClassValueOnly self varName propName)))
                                                    (SHOULDNT "Error in IVMissing")))
                                          (class (SELECTQ typeFlg
                                                      (PutValue (PutClassValue self varName newValue propName))
                                                      (PutValueOnly (PutClassValueOnly self varName newValue propName))
                                                      (GetValue (if (AND (NULL propName)
                                                                         (ValueFound initForm))
                                                                    then (EVAL initForm)
                                                                   else (GetClassValue self varName propName)))
                                                      (GetValueOnly (if (AND (NULL propName)
                                                                             (ValueFound initForm))
                                                                        then (EVAL initForm)
                                                                     else (GetClassValueOnly self varName propName)))
                                                      (SHOULDNT "Error in IVMissing")))
                                          (ERROR ":allocation property not one of (dynamic dynamicCached class)" allocation
                                                 ]
                 (COND
                    ((SETQ fixedName (\LoopsFixSpell varName (_ self ListAttribute 'IVs)
                                           (BLIPVAL '*FORM* 'Object.IVMissing 1)))
                                                                    (* It was misspelled)
                     (SETQ varName fixedName)
                     (GO Loop)))
                 (LoopsHelp varName " not an IV of " self)
              Finish
                 (RETURN (SELECTQ typeFlg
                             ((GetValue GetValueOnly)
                                  (APPLY* typeFlg self varName propName))
                             ((PutValue PutValueOnly)
                                  (APPLY* typeFlg self varName newValue propName))
                             (SHOULDNT "Error in IVMissing"])

(Method ((Object IVValueMissing) self varName propName typeFlg newValue)
                                                                    ; smL 29-May-86 12:07
   "No value found in the instance or in locally in the class if self is a class"
   (LET ((classValue (FetchCIVValueOnly (Class self)
                             varName propName)))
        (SELECTQ typeFlg
            (GetValueOnly classValue)
            (GetValue                                               (* NOTE%: This won't inherit for NotSetValue inside an
                                                                    activeValue in a class)
                       (COND
                          ((type? annotatedValue classValue)        (* Copy the annotated value down into the instance and send it
                                                                    the appropriate msg)
                           (_AV
                            (\PutValueOnly self varName propName (_AV
                                                                  classValue CopyActiveValue classValue))
                            GetWrappedValue self varName propName 'IV))
                          (T classValue)))
            (PutValueOnly (\PutValueOnly self varName propName newValue))
            (PutValue (COND
                          ((type? annotatedValue classValue)        (* Copy the annotated value down into the instance and send it
                                                                    the appropriate msg)
                           (_AV
                            (\PutValueOnly self varName propName (_AV
                                                                  classValue CopyActiveValue classValue))
                            PutWrappedValue self varName newValue propName 'IV))
                          (T (\PutValueOnly self varName propName newValue))))
            (SHOULDNT "Error in Put or GetValue"))))

(Method ((Object InstallEditSource) self editedDescription)         ; dgb:  4-OCT-83 11:33
   "Blank instance and make it conform to new description"
   (BlankInstance (Class self)
          self)
   (FillInst editedDescription self))

(Method ((Object InstallFileSource) self fileSource)                ; dgb: 13-OCT-83 22:06
   "Fill the given instance based on expression fileSource read from file, and name it"
   (NameObject self (LISTP (CAR fileSource)
                           NIL))
   (FillInst (CDR fileSource)
          self)
   self)

(Method ((Object ListAttribute) self type name)                     ; smL 18-Jun-86 14:05
   "For type= IVs, list the iv names in instance. For IVProps lists IV properties for name found in
   instance. Otherwise lists properties inherited from class"
   (SELECTQ (U-CASE type)
        ((IV IVPROPS NIL)
            [WithIVPropDescr self name [LAMBDA (self name propDescr)
                                       (for x in (fetch IVPropList of propDescr) by (CDDR x) collect x]
                   (LAMBDA (self name)
                     NIL])
        (IVS (APPEND (if (EQ (fetch ivNames of (Class self))
                             (fetch iNames of self))
                         then (fetch iNames of self)
```

```
                    else (UNION (fetch ivNames of (Class self))
                                (fetch iNames of self)))
                (for vl in (fetch otherIVs of self) collect (CAR vl))))
       (_ (Class self)
          ListAttribute type name)))

(Method ((Object ListAttribute!) self type name verboseFlg)       ; smL 17-Apr-87 10:14
   "Recursive form of ListAttribute for objects. Omits things inherited from Object unless verboseFlg is
   T."
   (SELECTQ (U-CASE type)
        (IVS (_ self ListAttribute type name))
        ((IV IVPROPS NIL)
             (UNION (_ self ListAttribute type name)
                    (_ (Class self)
                       ListAttribute! type name)))
        (_ (Class self)
           ListAttribute! type name verboseFlg)))

(Method ((Object MakeEditSource) self)                            ; smL  9-Sep-86 09:53
   "Get a lst showing all instance variables, values, and properties for Editing"
   (for ivDescr in (IVSource self) collect ivDescr))

(Method ((Object MakeFileSource) self file)                       ; smL 20-Dec-84 15:49
   "create a list structure source to be dumped on a file"
   [CONS 'DEFINST (CONS (ClassName self)
                        (CONS (GetObjectNames self)
                              (IVSource self T])

(Method ((Object MessageNotUnderstood) self selector messageArguments superFlg)
                                                                  ; dgb: 17-Dec-84 18:31
   "Invoked when a selector is not found for an object during a message sending operation. Attempts to do
   spelling correction on the selector. Forwards the message on to Tofu if it fails (causing a break we
   hope)"
   [LET ((correctSelector (FixSelectorSpelling self selector)))
        (COND
           (correctSelector (\ApplyMethod correctSelector messageArguments))
           (T (_Super])

(Method ((Object MoveToFile) self file)                           ; smL  9-Apr-87 15:22
   "Move this object to a file"
   (LET [(file (OR file (SelectFile]
        (if file
            then (LET ((name (GetObjectName self))
                       (uid (UID self)))
                      [if (AND name (WHEREIS uid 'INSTANCES))
                          then (DELFROMFILES uid 'INSTANCES (WHEREIS uid 'INSTANCES]
                      (MOVETOFILE file (OR name uid)
                                  'INSTANCES))
          else (FRESHLINE PROMPTWINDOW)
               (printout PROMPTWINDOW "No target file supplied, so nothing moved!")
               NIL)))

(Method ((Object NewInstance) self name arg1 arg2 arg3 arg4 arg5)
                                                                  ; smL  1-May-86 11:27
   "This allows initialization by the classes of objects themselves, rather than going to a metaClass"
   (AND name (_ self SetName name))
   (FastClassInitialize (Class self)
          self)
   (AND name (_ self SaveInstance name arg1 arg2))
   self)

(Method ((Object ObjectModified) self name reason)                ; smL 15-Aug-86 16:55
   "sent when self modified in any way"
   (DECLARE (SPECVARS \Default-Object-Modified-Reason))
   (if name
       then (MARKASCHANGED name 'INSTANCES reason))
   self)

(Method ((Object OldInstance) self)                               ; smL  5-Sep-86 14:26
   "Allow fixup of object after reading in."
   self)

(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 Rename) self newName oldNames)                   ; smL  4-Jan-85 16:28
   "Remove old name(s), and give it new name"
   ;; JRB - made it return self as docmented. Also
   (LET (files badName)
        (COND
           ((OR (NOT newName)
                (NOT (LITATOM newName)))
```

```
            (ERROR "Illegal LOOPS name" newName))
           ((NOT (CL:LISTP oldNames))
            (ERROR "Not a list of names" oldNames))
           (T [if oldNames
                  then (if (NOT (for N in oldNames always (SETQ badName N)
                                                          (EQ ($! N)
                                                              self)))
                           then (ERROR "Not a name for self" badName))
                    else (SETQ oldNames (CDR (REVERSE (GetObjectNames self]
              [if oldNames
                  then [SETQ files (for oldName in oldNames join (PROG1 (WHEREIS oldName 'INSTANCES)
                                                                        (_ self UnSetName oldName]
                       ;; And so the oldNames won't show up in (FILES?)...
                       (for oldName in oldNames do (UNMARKASCHANGED oldName 'INSTANCES]
              (_ self SetName newName)
              (for f in files do (ADDTOFILE newName 'INSTANCES f))
              self))))

(Method ((Object SaveInstance) self name arg1 arg2)            ; smL 15-Aug-86 16:23
   "Used to save the instance on a file. Justs marks it as changed as a default"
   (_ self ObjectModified name))

(Method ((Object SaveInstance?) self file outInstances)        ; edited: 26-Oct-84 15:36
   "Save this instance if referred to by another unless it is already on this list to be saved"
   [COND
      ((type? instance self)
       (NOT (FMEMB self outInstances)))
      (T (NOT (MEMBER self outInstances]))

(Method ((Object SetName) self name)                           ; smL  2-Apr-86 14:55
   "Call on NameEntity"
   (AND (NameEntity self name)
        (_ self ObjectModified name))
   self)

(Method ((Object UnSetName) self name)                         ; smL  8-Apr-87 13:13
   "Unname entity"
   [LET ((names (DeleteObjectName self name)))
        (if names
            then (for name inside names collect (LET [(files (WHEREIS name 'INSTANCES]
                                                     (UNMARKASCHANGED name 'INSTANCES)
                                                     (if files
                                                         then (DELFROMFILES name 'INSTANCES files)
                                                       else T])

(Method ((Tofu MessageNotUnderstood) self selector messageArguments superFlg)
                                                               ; smL 13-Aug-86 17:28
   "Pretends it understands Understands and also returns NIL for PrintOn so that PrintInstance will use
   the default datatype printing mechanism."
   (SELECTQ selector
        ((PrintOn Understands)
            NIL)
        ((GetValue PutValue GetValueOnly PutValueOnly GetIVHere)
            (ERROR (LIST* selector self (CDR messageArguments))
                   "not possible."))
        (ERROR (LIST (COND
                        (superFlg '_Super)
                        (T '_))
                     self selector '--)
               "not understood")))

(Method ((Tofu MethodNotFound) self selector)                  ; smL  5-Aug-86 17:22
   "Standard form for calling MessageNotUnderstood"
   `[CL:LAMBDA (&REST ArgumentsForMessageNotUnderstood)
           (SendMessageNotUnderstood ArgumentsForMessageNotUnderstood ',selector])

(Method ((Tofu SuperMethodNotFound) self selector classOfSendingMethod)
                                                               ; smL 16-Oct-85 17:12
   "No super method found when starting search from classOfSendingMethod"
   [if (GETD selector)
       then                                                    ; Use the function by the same name
            selector
     else `(CL:LAMBDA (&REST ArgumentsForMessageNotUnderstood)
                 (SendMessageNotUnderstood ArgumentsForMessageNotUnderstood ',selector T])

(\UnbatchMethodDefs)

(DEFINEQ

(MakeMethodMenu
  [LAMBDA (class xPos yPos)                                    ; Edited 13-Jun-88 19:00 by TAL

          (* * Puts up a menu for editing the selectors of a given class.)

    [COND
       ((LITATOM class)
```

```
        (SETQ class ($! class]
    (LET* [columns [selectors (SORT (_ class ListAttribute 'Methods]
                [menu (AND selectors (SETQ columns (ADD1 (IQUOTIENT (FLENGTH selectors)
                                                                    35)))
                           (create MENU
                                   ITEMS _ (ColumnateMenuItems selectors columns)
                                   MENUCOLUMNS _ columns
                                   WHENSELECTEDFN _ 'MethodMenuWhenSelectedFn
                                   TITLE _ (CONCAT "Edit methods for " (ClassName class]
                (window (AND menu (ADDMENU menu NIL (AND xPos (create POSITION
                                                                      XCOORD _ (IDIFFERENCE xPos
                                                                                            (fetch (MENU IMAGEWIDTH)
                                                                                                   of menu))
                                                                      YCOORD _ (IDIFFERENCE yPos
                                                                                            (fetch (MENU IMAGEHEIGHT)
                                                                                                   of menu]

        (* * place relative to the upper right hand corner)

        (AND window (WINDOWPROP window 'Class class))
        window])
```

(MethodMenuWhenSelectedFn

```
  [LAMBDA (selector menu key)                                   (* dgb%: " 5-Jun-84 14:36")
    (SELECTQ key
         (LEFT (ApplyMethodInTtyProcess (WINDOWPROP (WFROMMENU menu)
                                                    'Class)
                     'PPM
                     (LIST selector)))
         (MIDDLE (ApplyMethodInTtyProcess (WINDOWPROP (WFROMMENU menu)
                                                      'Class)
                       'EditMethod
                       (LIST selector)))
         NIL])
```

(SelectFile

```
  [LAMBDA prompts                                               ; Edited 14-Jun-88 16:30 by TAL
    ;; Select a file name, or specify a new one.  Return NIL if no file selected.  Create a COMS list and add to FILELST if it is new.
    (DECLARE (GLOBALVARS SelectFileMenu SelectFileMenuItems))
    (APPLY 'PROMPTPRINT (for I from 1 to prompts collect (ARG prompts I)))
    (LET (columns)
         (PROG [fileVar fullFileName
                     (file (MENU (COND
                                    ((AND (type? MENU SelectFileMenu)
                                          (EQUAL SelectFileMenuItems FILELST))
                                     SelectFileMenu)
                                    (T (SETQ columns (ADD1 (IQUOTIENT (FLENGTH FILELST)
                                                                      35)))
                                       (SETQ SelectFileMenuItems (APPEND FILELST))
                                       (SETQ SelectFileMenu
                                        (create MENU
                                                TITLE _ "File List"
                                                ITEMS _ (ColumnateMenuItems [NCONC1 (SORT (APPEND FILELST))
                                                                                    '(*newFile* '*newFile*
                                                                                            "Create a new file"
                                                                                            (SUBITEMS *newFile*
                                                                                                   *loadFile*
                                                                                                   *hiddenFile*]
                                                                                    columns)
                                                MENUCOLUMNS _ columns]
               (SELECTQ file
                    (*newFile* (if [NULL (SETQ file (U-CASE (PromptRead "Please type in file name: "]
                                   then (CLRPROMPT)
                                        (PROMPTPRINT "No file entered.")
                                        (RETURN))
                               (SETQ fileVar (FILECOMS file))
                               [COND
                                  ((BOUNDP fileVar)             ; Check if filecoms exists
                                   (OR (MOUSECONFIRM (CONCAT fileVar " has a value. Do  you want the value
                                                             smashed?")
                                                     "Trying to create fileComs for new file")
                                       (RETURN (PROMPTPRINT "No File Created"]
                               [SETTOPVAL fileVar (COPY `([,COMMENTFLG ; ,(CONCAT "File created by "
                                                                                  (USERNAME NIL T]
                                                          (CLASSES)
                                                          (METHODS)
                                                          (FNS)
                                                          (VARS)
                                                          (INSTANCES]
                               (ADDFILE file)
                               (RETURN file))
                    (*loadFile* (if [NULL (SETQ file (U-CASE (PromptRead "Please type in file name to load: "]
                                    then (CLRPROMPT)
                                         (PROMPTPRINT "No file entered.")
```

```
                                    (RETURN))
                                [SETQ fullFileName (OR (FINDFILE (PACKFILENAME 'BODY file 'EXTENSION
                                                                          'LCOM))
                                                       (FINDFILE (PACKFILENAME 'BODY file 'EXTENSION
                                                                          'DFASL))
                                                       (FINDFILE (PACKFILENAME 'BODY file 'EXTENSION ""]
                                (if fullFileName
                                    then (LOAD fullFileName)
                                         (RETURN (ROOTFILENAME fullFileName T))
                                  else (CLRPROMPT)
                                       (PROMPTPRINT "No such file")
                                       (RETURN)))
                          (*hiddenFile* [SETQ file
                                         (MENU (create MENU
                                                       ITEMS _ (OR (for file in LOADEDFILELST
                                                                      when (AND (NOT (FMEMB (ROOTFILENAME file T)
                                                                                            FILELST))
                                                                                (BOUNDP (FILECOMS file)))
                                                                      collect (ROOTFILENAME file T))
                                                                   (PROGN (PROMPTPRINT "No hidden files.")
                                                                          (RETURN]
                                        (if file
                                            then (push FILELST file)
                                                 (RETURN file)
                                          else (RETURN)))
                          (RETURN file])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(PUTPROPS \PutValueOnly MACRO [OPENLAMBDA (self varName propName newValue)
                                (COND
                                   [propName (WithIVPropDescr! self varName [LAMBDA (self varName propDescr)
                                                                              (InstPutProp propDescr propName
                                                                                     newValue]
                                                                    (LAMBDA (self varName)
                                                                      (SHOULDNT]
                                   (T (WithIVValue self varName [LAMBDA (self varName oldValue loc)
                                                                  (ChangeIVValue self varName loc newValue]
                                                             (LAMBDA (self varName)
                                                               (SHOULDNT])
)

;; END EXPORTED DEFINITIONS

(FILESLOAD (LOADCOMP)
       LOOPSDATATYPES LOOPSACTIVEVALUES LOOPSMETHODS)
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA METHCOM)

(ADDTOVAR NLAML OldClass)

(ADDTOVAR LAMA SelectFile)
)

(PUTPROPS LOOPSKERNEL COPYRIGHT ("Venue & Xerox Corporation" 1983 1984 1985 1986 1987 1988 1990 1991 1993))
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX

PROPERTY INDEX