File created: 15-Jun-93 12:27:27 **{DSK}<python>release>loops>2.0>src>LOOPSEDIT.;4**

changes to: (FNS InstallClassSource InstallClassVariables GetSourceCVs)

previous date: 14-Jun-93 23:52:08 {DSK}<python>release>loops>2.0>src>LOOPSEDIT.;3

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ LOOPSEDITCOMS
       [
```

;;; Functions used to create classes and instances

```
        (FNS ChangeEditedClass ChangeEditedInstance ChangedClass CheckAlist CheckClassSource CheckMetaClass
             CheckSupers ComputeSupersList DefineClass FromAnEditor? GetClassSource RemoveClassDef
             GetInstanceSource GetSourceCVs GetSourceIVs GetSourceInhCVs GetSourceInhIVs GetSourceMeta
             GetSourceMethods GetSourceSupers GoodClassName InstallClassSource InstallClassVariables
             InstallInstanceVariables InstallMetaClass InstallMethods InstallSupers OddLengthList
             RenameClassVariable RenameVariable)
```

;;; Inform DEdit how to edit certain datatypes --- at some point, these should be moved to the file LOOPS-DEDIT

```
        (FNS ObjSource ObjInstallSource)
        (ADDVARS (DT.EDITMACROS (class ObjSource ObjInstallSource)
                        (instance ObjSource ObjInstallSource)))
```

;; The following is a patch to SEdit which fixes AR 8899; nuke it for Motown

```
        [P (if (GETD '\\seditA0001)
               then
               (CHANGENAME '\\seditA0001 'READP '\SYSBUFP]
```

;; This makes METHOD-FNS shadow METHODS and FNS definitions from SEdit Meta-O

```
        (ADDVARS (SHADOW-TYPES (METHOD-FNS METHODS FNS])
```

;;; Functions used to create classes and instances

```
(DEFINEQ
```

(ChangeEditedClass

```
  [LAMBDA (self expression type flg)                          ; Edited  7-Mar-88 11:26 by jrb:
                                                              (* What is done if an class is changed)
    (if (CL:SYMBOLP self)
        then (SETQ self ($! self)))
    (AND flg (_ self InstallEditSource expression))
    self])
```

(ChangeEditedInstance

```
  [LAMBDA (self expression type flg)                          ; Edited  7-Mar-88 17:10 by jrb:
                                                              (* What is done if an instance is changed)
    (if (CL:SYMBOLP self)
        then (SETQ self ($! self)))
    [COND
       (flg (_ self InstallEditSource expression)
            (_ self ObjectModified (GetObjectName self]
    self])
```

(ChangedClass

```
  [LAMBDA (classOrName)                                       (* smL "17-Apr-86 13:42")

          (* Fn to mark a class as having been edited. Called when class is edited or defined.
          See the Cleanup fns for informing the file package about changed definitions.)
                                                              (* dgb%: "24-FEB-82 10:36")
    (MARKASCHANGED (SELECTQ (TYPENAME classOrName)
                        (class (ClassName classOrName))
                        classOrName)
           'CLASSES
           'CHANGED])
```

(CheckAlist

```
  [LAMBDA (type form)                                         ; Edited 27-May-87 17:22 by smL
```

;;;

;;; Called from CheckClassSource to check formatting of lists in source. Checks the list form to make sure that each element is a list starting with an
;;; atom. Complains on violations, and returns T if any errors were found, NIL otherwise

```
;;;

    (for pair in form bind errorOccurred do (COND
                                                ((NLISTP pair)
                                                 (printout T %,pair "must be of form (name prop1 val1 ...)" T)
                                                 (RETURN T))
                                                ((NOT (LITATOM (CAR pair)))
                                                 (printout T type %, .P2 (CAR pair)
                                                        " should be atomic to be a name" T)
                                                 (SETQ errorOccurred T))
                                                ((NULL (CDR pair))
                                                 (printout T type %, .P2 (CAR pair)
                                                        " should have a value" T)
                                                 (SETQ errorOccurred T))
                                                ((OddLengthList (CDDR pair))
                                                 (printout T (CAR pair)
                                                        " has a property list which is not of even length." T)
                                                 (SETQ errorOccurred T)))
       finally (RETURN errorOccurred])
```

(CheckClassSource

```
  [LAMBDA (source className)                                    ; Edited 27-May-87 17:03 by smL

;;;

;;; Checks the list structure form of the definition to make sure that no simple problems exist.  If there are problems, notifies the user and calls ERROR.
;;; This should cause the user to end up back in the editor.

;;;

    (for item in source bind form ERRFLG do (SETQ form (CDR item))
                                            (SELECTQ (CAR item)
                                                 (MetaClass (SETQ ERRFLG (OR (CheckMetaClass form)
                                                                             ERRFLG)))
                                                 (Supers (SETQ ERRFLG (OR (CheckSupers form)
                                                                          ERRFLG)))
                                                 (ClassVariables
                                                      (SETQ ERRFLG (OR (CheckAlist "ClassVariable" form)
                                                                       ERRFLG)))
                                                 (InstanceVariables
                                                      (SETQ ERRFLG (OR (CheckAlist "InstanceVariable" form)
                                                                       ERRFLG)))
                                                 (Methods (SETQ ERRFLG (OR (CheckAlist "Method" form)
                                                                           ERRFLG)))
                                                 ((MethodFns IVsInherited CVsInherited)
                                                                  ; Do nothing.  There to aid in editing

                                                      )
                                                 (PROGN (printout T item %, " is not a recognized item for class
                                                                definition" T)
                                                        (SETQ ERRFLG T)))
       finally (RETURN ERRFLG])
```

(CheckMetaClass

```
  [LAMBDA (form)                                                (* mjs%: " 1-JUL-82 16:45")

          (* * Checks to see if the metaClass is a real class and the property list length is even.)

    (PROG (metaClass)
          (SETQ metaClass (CAR form))
          (COND
             ((NULL form)
              (RETURN))
             ((NULL metaClass)
              (RPLACA form 'Class))
             ((NULL (GoodClassName metaClass form))
              (printout T "MetaClass " metaClass " is not a defined class" T)
              (RETURN T)))
          (COND
             ((NULL (OddLengthList form))
              (printout T "Property list of class is not of even length" T)
              (RETURN T)))
          (RETURN NIL])
```

(CheckSupers

```
  [LAMBDA (supersForm)                                          (* smL "18-Sep-86 10:28")

          (* * Checks the supersForm of supersForm to make sure that each name is a defined class.
          -
          Return NIL if there was no problem, otherwise T)

    (if (NOT (OR (NULL supersForm)
                 (LISTP supersForm)))
```

```
      then T
    else (for tail on supersForm bind (errorOccurred _ NIL)
                                      super
            do (SETQ super (CAR tail))
               (OR (GoodClassName super tail)
                   (PROGN (printout T super " must be replaced in definition, or defined as a class." T)
                          (SETQ errorOccurred T)))
            finally (RETURN errorOccurred])
```

(ComputeSupersList

```
[LAMBDA (localSupers inheritList)                          ; Edited 14-Aug-90 15:56 by jds

        (* * Compute closure of localSupers, removing earlier duplicates)

  (COND
     ((NULL localSupers)
      inheritList)
     (T (LET* [(first (CAR localSupers))
               (temp (ComputeSupersList (fetch (class localSupers) of first)
                                        (ComputeSupersList (CDR localSupers)
                                                           inheritList]
           (COND
              ((FMEMB first temp)
               temp)
              (T (CONS first temp])
```

(DefineClass

```
[LAMBDA (name supers self)                                 ; Edited 29-Sep-87 10:27 by jrb:
```

;; Defines a new class. className is the name of the new class or metaClass. supers is a list of class designators as follows: Each class
;; designator is an atom which is a class designator -- treated recursively in inheritance --- or a singleton list of such a className -- treated as a
;; patch of behavior. If DefineClass was invoked via a (_ self New) message, --- self is the class to which the message was sent (either Class or
;; MetaClass or subclass of MetaClass.) If some super is not yet a defined class, DefineClass asks the user to correct the list. Default on supers is
;; (Object) if self is Class, and is Class if self is MetaClass or one of its subClasses.

```
  (if (NOT (AND name (LITATOM name)))
      then (ERROR name "must be a LITATOM to be a class name.")
    elseif (AND ErrorOnNameConflict ($! name))
      then (HELPCHECK name "is already used as a name for an object.  To continue, type OK.")

           ;; If we get past here, we want to stomp the old name

           (_ ($! name)
              UnSetName name))                             ; Default case is that new class is not a MetaClass.
  (LET [(self (OR self ($ Class]                           ; Set default Supers according to whether you are being created
                                                           ; by MetaClass itself
       [InstallClassSource name '((MetaClass ,self Edited%: ,(EDITDATE NIL INITIALS))
                                  [Supers ,@(COND
                                              [supers (COND
                                                         ((NULL (CheckSupers supers))
                                                          supers)
                                                         (T (ERROR supers "is a bad supers list"]
                                              ((EQ self ($ MetaClass))
                                               (LIST 'Class))
                                              (T (LIST 'Object]
                                  (InstanceVariables)
                                  (Methods)
                                  (ClassVariables]         ; PutClass will mark this class as changed
       (SETQ LASTCLASS name)
       (SETQ LASTWORD name)
       (ADDSPELL name 0)
       (GetClassRec name])
```

(FromAnEditor?

```
[LAMBDA NIL                                                ; Edited  4-Mar-88 11:13 by jrb:
```

;; This works under Lyric SEdit only- it will have to change for Medley to sedit::whatever it is.

```
  (AND (STKPOS '\\verify.structure)
       T])
```

(GetClassSource

```
[LAMBDA (className)                                        ; Edited  4-Mar-88 11:14 by jrb:
                                                           (* Computes an editable list structure which represents the
                                                           "source" for a class definition)

  (LET [(classRec (COND
                     ((LITATOM className)
                      (GetObjectRec className))
                     ((type? class className)              (* In this case it should be a class record itself)
                      (PROG1 className
                          (SETQ className (ClassName className)))]
       (COND
          ((NOT (type? class classRec))
           NIL)
          ((FromAnEditor?)                                 ; HACK: are we under SEdit?
           (_ classRec MakeEditSource))
```

```
            (T (_ classRec MakeFileSource])
```

(RemoveClassDef

```
  [LAMBDA (NAME)                                                    (* smL " 8-Apr-87 18:09")

          (* * The FILEPKG hook when doing a DELDEF on a class)

    (LET ((class ($ NAME)))
         (if (Class? class)
             then (_ class Destroy])
```

(GetInstanceSource

```
  [LAMBDA (self)                                                    ; Edited  7-Mar-88 14:14 by jrb:

          (* * Computes a list structure which can be edited, and which when evaluated will reset contents of instance)

    [COND
       ((ATOM self)
        (SETQ self (GetObjectRec self]
    (COND
       ((NOT (type? instance self))
        NIL)
       ((FromAnEditor?)                                             ; HACK: are we under SEdit?
        (_ self MakeEditSource))
       (T (_ self MakeFileSource])
```

(GetSourceCVs

```
  [LAMBDA (classRec dontInit/DontSave)                              ; Edited 15-Jun-93 12:18 by sybalsky:mv:envos
    ;; Gets part of source for class -- list of local CVs values and properties
    ;; JDS 6/15/93:  Make :initForm work for this, and DontSave, also.  NB that DontSave Any is pretty stupid, but....
    ;; If dontInit/DontSave is T (as it is coming from MakeEditSource), don't bother with creaming the value.
    (for varName in (fetch (class cvNames) of classRec) as varDescr in (fetch (class cvDescrs) of classRec)
       collect (LET ((DESC (CONS varName varDescr))
                     Init)
                    (COND
                       (dontInit/DontSave
                              ;; Do nothing.
                              )
                       ((SETQ Init (LISTGET DESC :initForm))
                        (SETQ DESC (COPY DESC))
                        ;; The (LIST... below can't be a back-quote -- that prints wrong into the source file!
                        (RPLACA (CDR DESC)
                               (LIST '\LOOPS-PRINT-EVAL-MARKER Init)))
                       ((EQ 'Any (LISTGET DESC 'DontSave))
                        (SETQ DESC (COPY DESC))
                        (RPLACD DESC NIL))
                       ((EQ 'Value (LISTGET DESC 'DontSave))
                        (SETQ DESC (COPY DESC))
                        (RPLACA (CDR DESC)
                               NIL)))
                    DESC])
```

(GetSourceIVs

```
  [LAMBDA (classRec)                                                (* dgb%: "26-NOV-82 10:32")
    (for varName in (fetch (class localIVs) of classRec) collect (CONS varName (FetchCIVDescr classRec varName])
```

(GetSourceInhCVs

```
  [LAMBDA (self)                                                    ; Edited 17-Jul-87 18:25 by jrb:
    ;; Get a full description of all the IVs and their properties
    ;; JRB - changed GetClassValue to GetClassValueOnly to not trigger any AVs that may be there.
    (for iv in (_ self ListAttribute! 'CVs) bind (locals _ (_ self ListAttribute 'CVs))
       when (NOT (FMEMB iv locals)) collect (CONS iv (CONS (GetClassValueOnly self iv)
                                                           (for prop in (_ self ListAttribute! 'CVProps iv)
                                                              join (LIST prop (GetClassValueOnly self iv prop])
```

(GetSourceInhIVs

```
  [LAMBDA (self)                                                    (* smL "11-Apr-86 15:01")
                                                                    (* Get a full description of all the IVs and their properties)
    (for iv in (_ self ListAttribute! 'IVs) bind (locals _ (_ self ListAttribute 'IVs))
       when (NOT (FMEMB iv locals)) collect (NCONC (LIST iv (GetClassIV self iv))
                                                   (for prop in (_ self ListAttribute! 'IVProps iv)
                                                      join (LIST prop (GetClassIV self iv prop])
```

(GetSourceMeta

```
  [LAMBDA (classRec)                                                ; Edited 14-Aug-90 15:57 by jds
```

```
    (CONS (ClassName (fetch (class metaClass) of classRec))
          (fetch (class otherClassDescription) of classRec])
```

(GetSourceMethods

```
  [LAMBDA (classRec)                                          ; Edited 14-Aug-90 15:57 by jds
    (LET ((sels (fetch (class selectors) of classRec))
          (meths (fetch (class methods) of classRec)))
         (if sels
             then (SORT (for I from 0 by 2 bind sel until (NULL (SETQ sel (\GetNthEntry sels I)))
                           collect `(,sel ,(\GetNthEntry meths I)
                                          args
                                          ,(GetMethod classRec sel 'args)
                                          doc
                                          ,(GetMethod classRec sel 'doc])
```

(GetSourceSupers

```
  [LAMBDA (classRec)                                          (* dgb%: "23-APR-83 16:56")
    (for s in (fetch (class localSupers) of classRec) collect (ClassName s])
```

(GoodClassName

```
  [LAMBDA (classNameOrClass tail errFlg)                      ; Edited 14-Aug-90 15:57 by jds

          (* * Checks classNameOrClass to see if defines a class. If not tries to make a spelling correction, or define a new class.
          If tail is specified, it will stuff any corrections into the (CAR tail)%.
          If errFlg is T, will cause an error class name is invalid and it can not fix it by spelling correction.
          Returns a class classNameOrClass or NIL)

    (COND
       ((type? class classNameOrClass)
        (fetch (class className) of classNameOrClass))
       (T (OR (AND (GetClassRec classNameOrClass)
                   classNameOrClass)
              (OR (AND errFlg (ERROR classNameOrClass "is not a defined class"))
                  (COND
                     ((EQ 'Y (ASKUSER NIL 'Y (LIST "Should" classNameOrClass "be defined as a new class")
                                      NIL))
                      (_ ($ Class)
                         New classNameOrClass)
                      classNameOrClass])
```

(InstallClassSource

```
  [LAMBDA (className source)                                  ; Edited 15-Jun-93 12:02 by sybalsky:mv:envos
    ;; Called by DEFCLASS to actually create the class record from the Source.  There is no error checking here, so the class source had better be
    ;; correct.

    (LET ((oldClass? (GetClassRec className))
          (classRec (OR (GetClassRec className)
                        (NewClass className)))
          sourceForm)
         [COND
            ((SETQ sourceForm (FASSOC 'MetaClass source))
             (InstallMetaClass classRec (CDR sourceForm]
         [COND
            ((OR (SETQ sourceForm (FASSOC 'Supers source))
                 oldClass?)
             (InstallSupers classRec (CDR sourceForm]
         [COND
            ((OR (SETQ sourceForm (FASSOC 'ClassVariables source))
                 oldClass?)
             (InstallClassVariables classRec (CDR sourceForm]
         [COND
            ((SETQ sourceForm (FASSOC 'Methods source))
             (InstallMethods classRec (CDR sourceForm]
         [COND
            ((OR (SETQ sourceForm (FASSOC 'InstanceVariables source))
                 oldClass?)
             (InstallInstanceVariables classRec (CDR sourceForm]
         (COND
            ((NOT oldClass?)
             (MARKASCHANGED className 'CLASSES 'DEFINED))
            (T (MARKASCHANGED className 'CLASSES 'CHANGED])
```

(InstallClassVariables

```
  [LAMBDA (classRec form)                                     ; Edited 15-Jun-93 12:14 by sybalsky:mv:envos
    (/replace (class cvNames) of classRec with (MAPCAR form 'CAR))
    (/replace (class cvDescrs) of classRec with (MAPCAR form 'CDR))
    (for cvdesc in (ffetch (class cvDescrs) of classRec) do (LET (Init)
                                                                 (CL:WHEN (SETQ Init (LISTGET (CDR cvdesc)
                                                                                              :initForm))
                                                                     (PROMPTPRINT Init)
                                                                     (RPLACA cvdesc (EVAL Init)))])
```

(InstallInstanceVariables

```
  [LAMBDA (classRec form)                                    ; Edited 14-Aug-90 15:57 by jds

          (* * Starting with form of ivName decription pairs, this installs the new descriptions in the class.
          Causes an update of IVDescrs, and of subs of this class.)

    (PROG [(varNames (for pair in form collect (CAR pair]

          (* * In case there is some significant change in IVs. Set up for updating by putting only local description in ivDescrs)

          (/replace (class localIVs) of classRec with varNames)
          (/replace (class ivNames) of classRec with varNames)
          [/replace (class ivDescrs) of classRec with (for pair in form collect (OR (CDR pair)
                                                                                    (LIST NIL]
          (UpdateIVDescrs classRec])
```

(InstallMetaClass

```
  [LAMBDA (classRec form)                                    ; Edited 14-Aug-90 15:57 by jds
    (/replace (class metaClass) of classRec with (GetClassRec (CAR form)))
    (/replace (class otherClassDescription) of classRec with (CDR form])
```

(InstallMethods

```
  [LAMBDA (classRec form)                                    ; Edited 14-Aug-90 15:57 by jds
                                                             (* Called by InstallClassSource for Methods only.)
    (/replace (class selectors) of classRec with (\BlockFromList form (FUNCTION CAR)))
    (/replace (class methods) of classRec with (\BlockFromList form (FUNCTION CADR)))
    (for methForm in form bind file files methName fnName sel (cName _ (ClassName classRec))
       do (DefMethObj cName (SETQ sel (pop methForm))
                  (SETQ fnName (pop methForm))
                  (LISTGET methForm 'args)
                  (LISTGET methForm 'doc)
                  (for X by (CDDR X) on methForm when (AND (NEQ 'doc (CAR X))
                                                           (NEQ 'args (CAR X)))
                     join (LIST (CAR X)
                                (CADR X])
```

(InstallSupers

```
  [LAMBDA (classRec form)                                    ; Edited 14-Aug-90 15:57 by jds

;;;

;;; Install the list of super classes in the classRec.  Special case check for Tofu and NULL supers list

;;;

    (PROG [class addList deleteList (currentSupers (fetch (class localSupers) of classRec))
                (newSupers (for name inside (OR form (LIST 'Tofu)) collect (GetClassRec name]
          [COND
             ((EQ 'Tofu (fetch (class className) of classRec))
              ;; Can't change Tofu's supers
              (RETURN 'NoUpdateRequired))
             ((EQUAL currentSupers newSupers)
              ;; no change in supers
              (RETURN 'NoUpdateRequired))
             ([OR (MEMB classRec newSupers)
                  (for newSuper in newSupers thereis (MEMB classRec (fetch (class supers) of newSuper]
              ;; Can't make a Loop in the inheritence lattice
              (RETURN 'NoUpdateRequired]
          (FlushMethodCache)
          (replace (class localSupers) of classRec with newSupers)
          (replace (class supers) of classRec with (ComputeSupersList newSupers))
          (SETQ addList (LDIFFERENCE newSupers currentSupers))
          (SETQ deleteList (LDIFFERENCE currentSupers newSupers))
       ;; For new items on newSupers, add back pointers, and for deleted items on current supers delete back pointers
          [for super in addList do (/replace (class subClasses) of super with (CONS classRec (fetch (class subClasses
                                                                                                    )
                                                                                                of super]
          [for super in deleteList do (/replace (class subClasses) of super with (REMOVE classRec
                                                                                    (fetch (class subClasses)
                                                                                       of super]
       ;; Now put new supers list in class, and update the instance variables
          (UpdateClassIVs classRec])
```

(OddLengthList

```
  [LAMBDA (pairList)                                         (* dgb%: "18-JUN-82 09:39")

          (* * Syntax checking function used by CheckAList and CheckMetaClass)
```

```
    (for p on pairList by (CDDR p) do (COND
                                          ((NULL (CDR p))
                                           (RETURN T)))
       finally (RETURN NIL])
```

(RenameClassVariable

```
  [LAMBDA (className oldVarName newVarName)                    (* dgb%: "18-NOV-82 03:20")
```

(* Renames the variable in the class, but does NOT look for references of the variable in the methods. Returns newVarName if successful, NIL otherwise)

```
    (PROG (source varList)
          (OR (SETQ source (GetClassSource className))
              (RETURN NIL))
          (OR (SETQ varList (FASSOC oldVarName (FASSOC 'ClassVariables source)))
              (RETURN NIL))
          (RPLACA varList newVarName)
          (EVAL source)
          (RETURN newVarName])
```

(RenameVariable

```
  [LAMBDA (className oldVarName newVarName classVarFlg)        (* dgb%: "10-NOV-82 16:06")
```

(* Renames the variable in the class, but does NOT look for references of the variable in the methods. Returns newVarName if successful, NIL otherwise)

```
    (PROG (source varList)
          (OR (SETQ source (GetClassSource className))
              (RETURN NIL))
          (OR (SETQ varList (FASSOC oldVarName (FASSOC (COND
                                                          (classVarFlg 'ClassVariables)
                                                          (T 'InstanceVariables))
                                                       source)))
              (RETURN NIL))
          (RPLACA varList newVarName)
          (EVAL source)
          (RETURN newVarName])

)

;;; Inform DEdit how to edit certain datatypes --- at some point, these should be moved to the file LOOPS-DEDIT

(DEFINEQ

## (ObjSource

  [LAMBDA (self)                                               (* dgb%: "11-NOV-83 18:21")
    (_ self MakeEditSource])

## (ObjInstallSource

  [LAMBDA (obj expression)                                     (* dgb%: "11-NOV-83 18:21")
    (_ obj InstallEditSource expression])

)

(ADDTOVAR DT.EDITMACROS (class ObjSource ObjInstallSource)
                        (instance ObjSource ObjInstallSource))

;; The following is a patch to SEdit which fixes AR 8899; nuke it for Motown

(if (GETD '\\seditA0001)
    then (CHANGENAME '\\seditA0001 'READP '\SYSBUFP))

;; This makes METHOD-FNS shadow METHODS and FNS definitions from SEdit Meta-O

(ADDTOVAR SHADOW-TYPES (METHOD-FNS METHODS FNS))

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

FUNCTION INDEX

VARIABLE INDEX