File created: 6-Nov-91 15:41:40 **{DSK}<python>RELEASE>loops>2.0>src>LOOPSACCESS.;2**

changes to: (FNS Cached-GetIVValue Cached-PutIVValue)

previous date: 27-Jul-90 08:02:31 {DSK}<python>RELEASE>loops>2.0>src>LOOPSACCESS.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ LOOPSACCESSCOMS
       [
```

;;; Access functions which don't know the structure of the underlying datatypes

;;; The basic GetValue and PutValue functions

```
        (OPTIMIZERS GetValue PutValue)
        (FNS GetValue PutValue)
                                                                    ; Internal stuff
        (FNS GetIVValue GetIVProp PutIVProp PutIVValue)
        (FNS Cached-GetIVValue Cached-GetIVProp Cached-PutIVValue Cached-PutIVProp)
```

;;; Generic Get and Put functions

```
        (FNS GetIt GetItHere GetItOnly PutIt PutItOnly)
```

;;; Misc other access fns

```
        (FNS DeleteCIV DeleteCV)
        (FNS FetchCIVDescr FetchCIVValueOnly FetchCVOnly FetchMethodClass FetchMethodDescr)
        (FNS GetCVHere GetClassIV GetClassIVHere GetClassValue GetClassValueOnly GetLocalState GetLocalStateOnly
             )
        (FNS PutCIVHere PutCVHere PutClassIV PutClassValue PutClassValueOnly)
```

;;; IVs that are FIXPs

```
        (FNS GetNthValue FetchNthValue FetchNthValueOnly FetchNthDescr! FetchNthDescr PutNthValue StoreNthValue
             StoreNthValueOnly)
```

;;; Class property stuff

```
        (FNS GetClass GetClassOnly GetClassHere)
        (FNS DeleteClassProp)
        (FNS PutClass PutClassOnly)
```

;;; Method property stuff

```
        (FNS GetMethod GetMethodOnly GetMethodHere)
        (FNS PutMethod PutMethodOnly)
```

;;; CHANGETRAN stuff for IV access

```
        (PROPS (GetClassValueOnly SETFN)
               (GetValueOnly SETFN)
               (GetClassValue SETFN)
               (@ SETFN)
               (GetValue SETFN))
        (MACROS PtClassValueOnly PtValueOnly PutClsValue PtValue)
        (FNS SwitchPutArgs)
```

;;; UNDOable versions

```
        (FNS /PutValue)
        (DECLARE%: DONTEVAL@LOAD (P (NEW/FN 'PutValue])
```

;;; Access functions which don't know the structure of the underlying datatypes

;;; The basic GetValue and PutValue functions

```
(DEFOPTIMIZER GetValue (self varName &OPTIONAL (propName NIL))
                       [if (AND (LISTP varName)
                                (EQ (CAR varName)
                                    'QUOTE))
                           then                                    ; We have a fixed IV name
                                [if (NULL propName)
                                    then `(Cached-GetIVValue ,self ,varName (LOADTIMECONSTANT (
                                                                                \Make-IV-Cache-Entry
                                                                                 )))
                                  else `(Cached-GetIVProp ,self ,varName ,propName (LOADTIMECONSTANT (
                                                                                \Make-IV-Cache-Entry
                                                                                       ]
                         else                                      ; Variable IV name
                              (if (NULL propName)
                                  then `(GetIVValue ,self ,varName)
                                else `(GetIVProp ,self ,varName ,propName])


(DEFOPTIMIZER PutValue (self varName newValue &OPTIONAL (propName NIL))
                       [if (AND (LISTP varName)
                                (EQ (CAR varName)
                                    'QUOTE))
                           then [if (NULL propName)
                                    then `(Cached-PutIVValue ,self ,varName ,newValue (LOADTIMECONSTANT (
                                                                                \Make-IV-Cache-Entry
                                                                                            )))
                                  else `(Cached-PutIVProp ,self ,varName ,newValue ,propName (LOADTIMECONSTANT
                                                                                (
                                                                                \Make-IV-Cache-Entry
                                                                                 ]
                         else (if (NULL propName)
                                  then `(PutIVValue ,self ,varName ,newValue)
                                else `(PutIVProp ,self ,varName ,newValue ,propName])

(DEFINEQ

(GetValue
  [LAMBDA (self varName propName)                                  (* smL "23-May-86 11:04")

          (* * Used to return the value of an "instance variable" for an object.
          Calls general Get function for non standard objects -
          GetValue activates getFn if the value is an activevalue.)

    (GetIVProp self varName propName])

(PutValue
  [LAMBDA (self varName newValue propName)                         (* smL "27-May-86 18:04")

          (* * Replace the IV or IVProp value)

    (PutIVProp self varName newValue propName])

)

;; Internal stuff

(DEFINEQ

(GetIVValue
  [LAMBDA (obj ivName)                                             (* smL "28-May-86 15:27")

          (* * Get an IV value from an instance. Implements GetValue when there is no property.)

    (COND
       ((type? instance obj)
        (GetInstanceIVValue obj ivName))
       (T (GetIt obj ivName NIL 'IV])


(GetIVProp
  [LAMBDA (obj ivName propName)                                    (* edited%: "10-Jun-86 14:58")

          (* * Get an IV property from an object. Implements GetValue when there is a property.)

    (COND
       ((NOT (type? instance obj))
        (GetIt obj ivName propName 'IV))
       [propName (WithIVPropDescr obj ivName [LAMBDA (self ivName propDescr)
                                             (ExtractRealValue self ivName (InstGetProp propDescr propName)
                                                    propName
                                                    'IV]
                         (LAMBDA (self ivName)
                           (_ self IVMissing ivName propName 'GetValue]
       (T (GetInstanceIVValue obj ivName])
```

(PutIVProp

```
  [LAMBDA (self varName newValue propName)                    (* smL " 5-Aug-86 17:13")

          (* * Replace the IV or IVProp value)

    (COND
       ((NOT (type? instance self))
        (PutIt self varName newValue propName 'IV))
       [propName (WithIVPropDescr! self varName [LAMBDA (self varName propDescr)
                                                  (LET ((oldValue (InstGetProp propDescr propName)))
                                                       (COND
                                                          ((type? annotatedValue oldValue)
                                                           (_ (fetch annotatedValue of oldValue)
                                                              PutWrappedValue self varName newValue propName
                                                              'IV))
                                                          (T (InstPutProp propDescr propName newValue]
                         (LAMBDA (self varName)
                           (_ self IVMissing varName propName 'PutValue newValue]
       (T (WithIVValue self varName [LAMBDA (self varName oldValue loc)
                                      (COND
                                         ((type? annotatedValue oldValue)
                                          (_ (fetch annotatedValue of oldValue)
                                             PutWrappedValue self varName newValue NIL 'IV))
                                         (T (ChangeIVValue self varName loc newValue]
                  (LAMBDA (self varName)
                    (_ self IVMissing varName NIL 'PutValue newValue])
```

(PutIVValue

```
  [LAMBDA (self varName newValue)                             (* smL " 5-Aug-86 17:13")

          (* * Replace the IV or IVProp value)

    (COND
       ((NOT (type? instance self))
        (PutIt self varName newValue NIL 'IV))
       (T (WithIVValue self varName [LAMBDA (self varName oldValue loc)
                                      (COND
                                         ((type? annotatedValue oldValue)
                                          (_ (fetch annotatedValue of oldValue)
                                             PutWrappedValue self varName newValue NIL 'IV))
                                         (T (ChangeIVValue self varName loc newValue]
                  (LAMBDA (self varName)
                    (_ self IVMissing varName NIL 'PutValue newValue])

)

(DEFINEQ
```

(Cached-GetIVValue

```
  [LAMBDA (obj ivName cache)                                  (* edited%: " 3-Jun-86 13:07")

          (* * Get an IV value from an instance. Implements GetValue when there is no property.)

    (COND
       ((NOT (type? instance obj))
        (GetIt obj ivName NIL 'IV))
       (T (WithIVValue obj ivName [LAMBDA (self varName value)
                                    (ExtractRealValue self varName value NIL 'IV]
                  [LAMBDA (self varName)
                    (_ self IVMissing varName NIL 'GetValue]
                  cache])
```

(Cached-GetIVProp

```
  [LAMBDA (obj ivName propName cache)                         (* smL " 9-Jun-86 19:23")

          (* * Get an IV property from an object. Implements GetValue when there is a property.)

    (COND
       ((NOT (type? instance obj))
        (GetIt obj ivName propName 'IV))
       (propName (WithIVPropDescr obj ivName [LAMBDA (self ivName propDescr)
                                               (ExtractRealValue self ivName (InstGetProp propDescr propName)
                                                      propName
                                                      'IV]
                         [LAMBDA (self ivName)
                           (_ self IVMissing ivName propName 'GetValue]
                         cache))
       (T (WithIVValue obj ivName [LAMBDA (self varName value)
                                    (ExtractRealValue self varName value NIL 'IV]
                  [LAMBDA (self varName)
                    (_ self IVMissing varName NIL 'GetValue]
                  cache])
```

(Cached-PutIVValue

```
  [LAMBDA (self varName newValue cache)                    (* smL " 5-Aug-86 17:14")

          (* * Replace the IV or IVProp value)

    (COND
       ((NOT (type? instance self))
        (PutIt self varName newValue NIL 'IV))
       (T (WithIVValue self varName [LAMBDA (self varName oldValue loc)
                                      (COND
                                         ((type? annotatedValue oldValue)
                                          (_ (fetch annotatedValue of oldValue)
                                             PutWrappedValue self varName newValue NIL 'IV))
                                         (T (ChangeIVValue self varName loc newValue]
                [LAMBDA (self varName)
                  (_ self IVMissing varName NIL 'PutValue newValue]
                cache])
```

(Cached-PutIVProp

```
  [LAMBDA (self varName newValue propName cache)           (* smL " 5-Aug-86 17:14")

          (* * Replace the IV or IVProp value)

    (COND
       ((NOT (type? instance self))
        (PutIt self varName newValue propName 'IV))
       (propName (WithIVPropDescr! self varName [LAMBDA (self varName propDescr)
                                                  (LET ((oldValue (InstGetProp propDescr propName)))
                                                       (COND
                                                          ((type? annotatedValue oldValue)
                                                           (_ (fetch annotatedValue of oldValue)
                                                              PutWrappedValue self varName newValue propName
                                                              'IV))
                                                          (T (InstPutProp propDescr propName newValue]
                         [LAMBDA (self varName)
                           (_ self IVMissing varName propName 'PutValue newValue]
                         cache))
       (T (WithIVValue self varName [LAMBDA (self varName oldValue loc)
                                      (COND
                                         ((type? annotatedValue oldValue)
                                          (_ (fetch annotatedValue of oldValue)
                                             PutWrappedValue self varName newValue NIL 'IV))
                                         (T (ChangeIVValue self varName loc newValue]
                [LAMBDA (self varName)
                  (_ self IVMissing varName NIL 'PutValue newValue]
                cache])

)

;;; Generic Get and Put functions

(DEFINEQ
```

(GetIt

```
  [LAMBDA (self varOrSelector propName type)               (* smL " 8-Apr-87 14:56")

          (* * "General Get fn")

    (SELECTQ type
        ((NIL IV)
             (COND
                [(type? instance self)
                 (COND
                    ((NUMBERP varOrSelector)
                     (FetchNthValue self varOrSelector propName))
                    (T (GetValue self varOrSelector propName]
                ((type? class self)
                 (GetClassIV self varOrSelector propName))
                (T (_ self GetValue varOrSelector propName))))
        (CV (GetClassValue self varOrSelector propName))
        (CLASS                                             ; ambiguous which arg should be used for class propName
               (_ self GetClassProp (OR varOrSelector propName)))
        (METHOD (GetMethod self varOrSelector propName))
        (ERROR type "not part of Class.  Use one of NIL (for instance variables), CV, CLASS, METHOD"])
```

(GetItHere

```
  [LAMBDA (self varOrSelector propName type)               (* dgb%: " 9-Apr-84 12:24")

          (* * General Get fn with no inheritance or active Values)

    (SELECTQ type
        ((NIL IV)
             [COND
                ((type? instance self)
                 (GetIVHere self varOrSelector propName))
```

```
               ((type? class self)
                (GetClassIVHere self varOrSelector propName))
               (T (PROG ((objClass (GetLispClass self)))
                        (COND
                           ((NULL objClass)
                            (LoopsHelp self "has no instance variables."))
                           (T (RETURN (_ objClass GetValueHere self varOrSelector propName])
        (CV (GetCVHere self varOrSelector propName))
        (CLASS                                                    (* ambiguous which arg should be used for class propName)
               (GetClassHere self (OR varOrSelector propName)))
        (METHOD (GetMethodHere self varOrSelector propName))
        (ERROR type "not part of Class.  Use one of NIL (for instance variables), CV, CLASS, METHOD"])
```

(GetItOnly

```
  [LAMBDA (self varOrSelector propName type)                      (* dgb%: "26-Nov-84 17:24")

          (* * Like GetIt except doesn't check for active values.)

    (SELECTQ type
         ((NIL IV)
              (COND
                 ((type? instance self)
                  (GetValueOnly self varOrSelector propName))
                 ((type? class self)
                  (GetClassIV self varOrSelector propName))
                 (T (_ (GetLispClass self)
                       GetValueOnly self varOrSelector propName))))
         (CV (GetClassValueOnly self varOrSelector propName))
         (CLASS                                                   (* ambiguous which arg should be used for propName for class
                property)
                (GetClassOnly self (OR varOrSelector propName)))
         (METHOD (GetMethodOnly self varOrSelector propName))
         (ERROR type "not part of Class.  Use one of NIL (for instance variables), CV, CLASS, METHOD"])
```

(PutIt

```
  [LAMBDA (self varOrSelector newValue propName type)             (* smL "29-Oct-86 15:28")

;;;

;;; General put fn

;;;

    (SELECTQ type
         ((NIL IV)
              (COND
                 ((type? class self)
                  ;;
                  ;; no activation of active values when putting into iv description in class
                  ;;
                  (PutClassIV self varOrSelector newValue propName))
                 [(type? instance self)
                  (COND
                     ((NUMBERP varOrSelector)
                      (StoreNthValue self varOrSelector newValue propName))
                     (T (PutValue self varOrSelector newValue propName]
                 (T (_ self PutValue varOrSelector newValue propName))))
         (CV (PutClassValue self varOrSelector newValue propName))
         (CLASS                                                   ; ambiguous which arg should be used for class propname
                (PutClass self newValue (OR varOrSelector propName)))
         (METHOD (PutMethod self varOrSelector newValue propName))
         (ERROR type "not part of Class.
                Use one of NIL (for instance variables), CV, CLASS, METHOD"])
```

(PutItOnly

```
  [LAMBDA (self varOrSelector newValue propName type)             (* dgb%: "12-JAN-83 14:30")

          (* * Like PutIt except that does not check for active values.)

    (SELECTQ type
         ((NIL IV)
              (COND
                 ((type? class self)
                  (PutClassIV self varOrSelector newValue propName))
                 (T (PutValueOnly self varOrSelector newValue propName))))
         (CV (PutClassValueOnly self varOrSelector newValue propName))
         (CLASS                                                   (* ambiguous which arg should have been used as name of
                                                                  prop for class properties)
                (PutClassOnly self newValue (OR varOrSelector propName)))
         (METHOD (PutMethodOnly self varOrSelector newValue propName))
         (ERROR type "not part of Class.
                Use one of NIL (for instance variables), CV, CLASS, METHOD"])
```

```
)

;;; Misc other access fns

(DEFINEQ

(DeleteCIV
  [LAMBDA (class varName prop)                                    ; Edited 29-Sep-87 10:15 by jrb:

          (* * Deletes an IV from a class, or a property from its propList if prop is given.)

    (if (Class? class)
        then (PROG1 [COND
                    ((NULL (_ class HasIV varName))
                     (HELPCHECK varName "is not a local instance variable of class " (ClassName class)
                            "
                            Type OK to ignore error and go on."))
                    (prop (ObjRemProp (FetchCIVDescr class varName)
                                  prop))
                    (T (InstallInstanceVariables class (DELASSOC varName (GetSourceIVs class]
                  (ChangedClass class))
      else (PROMPTPRINT class " is not a class")
           NIL])

(DeleteCV
  [LAMBDA (class varName prop)                                    ; Edited 29-Sep-87 10:14 by jrb:

          (* * Deletes a classVariable or one of its properties)

    (if (Class? class)
        then (PROG1 [COND
                    (prop (ObjRemProp (FetchCVDescr class varName)
                                  prop))
                    (T (for var varFoundFlg in (fetch cvNames of class) as descr
                          in (fetch cvDescrs of class) bind newNames newDescr first (SETQ newNames (CONS))
                                                                                    (SETQ newDescr (CONS))
                          do (COND
                                 ((NEQ var varName)
                                  (TCONC newNames var)
                                  (TCONC newDescr descr))
                                 (T (SETQ varFoundFlg T)))
                          finally (replace cvNames of class with (CAR newNames))
                                  (replace cvDescrs of class with (CAR newDescr))
                                  (RETURN (AND varFoundFlg varName]
                  (ChangedClass class))
      else (PROMPTPRINT class " is not a class")
           NIL])

)

(DEFINEQ

(FetchCIVDescr
  [LAMBDA (self varName)                                          (* dgb%: "28-APR-83 17:47")

          (* * Find the description list for the named variable in a class, returning NIL if none is there.)

    (PROG NIL

          (* Short circuit the GetNth with embedded RETURN if no index is found, i.e.
          the variable does not exist)

          (RETURN (GetCIVNth self (OR (FindVarIndex varName self)
                                      (RETURN NIL])

(FetchCIVValueOnly
  [LAMBDA (classRec varName propName)                             (* smL "21-Apr-86 17:10")

          (* * Used to fetch the default IV value from the class. Returns value of NoValueFound if none found)

    (PROG (supers val (class classRec))
          (COND
             ((NULL class)
              (RETURN NIL)))
          (SETQ supers (Supers class))
      LP  (SETQ val (ObjGetProp (FetchCIVDescr class varName)
                            propName))
          (COND
             ((NOT (NotSetValue val))
              (RETURN val)))
          (SETQ class (pop supers))
          (COND
             (class (GO LP))
```

```
            (T (RETURN NoValueFound])
```

(FetchCVOnly

```
[LAMBDA (self varName propName typeFlag)                    (* smL " 5-Jun-86 12:28")

          (* * Maps through a class and its supers and returns property value with no activations.
          Returns NotSetValue if none found.)

    (for class in-supers-of self bind descr value foundCV? eachtime (SETQ descr (FetchCVDescr class varName))
       when descr do (SETQ value (ObjGetProp descr propName))
                     (COND
                        ((NOT (NotSetValue value))
                         (RETURN value))
                        (T (SETQ foundCV? T)))
       finally (COND
                  (foundCV? (RETURN NotSetValue))
                  (T (RETURN (_ (Class self)
                                CVMissing self varName propName typeFlag])
```

(FetchMethodClass

```
[LAMBDA (class selector)                                    (* dgb%: "28-Feb-84 17:25")
  (PROGN                                                    (* dgb%: "12-JAN-82 15:20")

          (* * MapSupersForm is used to look up the supers chain for class)

           (MapSupersForm (COND
                             ((FindSelectorIndex class selector) (* First search the selectors stored with this class.)
                              (RETURN class)))
                  class])
```

(FetchMethodDescr

```
[LAMBDA (self selector)                                     ; Edited 27-May-87 16:04 by smL

          (* * Computes Method descr from method object)

    (if (FindSelectorIndex self selector)
        then (LET ((meth (GetMethodObj self selector)))
                  (if meth
                      then `(args NIL doc NIL ,.(WithIVPropDescr meth 'method [LAMBDA (meth varName propDescr)
                                                                                (fetch IVPropList of propDescr]
                                                                    (LAMBDA (meth varName)
                                                                      NIL])

)

(DEFINEQ
```

(GetCVHere

```
[LAMBDA (classRec varName propName)                         (* smL "19-Nov-85 14:51")

          (* Gets the class property here without inheritance or activeValues.
          Returns NotSetValue if not found here)

    (ObjGetProp (FetchCVDescr classRec varName)
           propName])
```

(GetClassIV

```
[LAMBDA (self varName prop)                                 (* mjs%: "30-Oct-85 11:10")
                                                            (* Get the value of an IV from the class)
    (COND
       ((type? class self)
        (for class in (CONS self (Supers self)) bind val when [NOT (NotSetValue (SETQ val (ObjGetProp (
                                                                                                  FetchCIVDescr
                                                                                                   class
                                                                                                   varName)
                                                                                        prop]
           do (RETURN val) finally (RETURN NotSetValue)))
       (T (ERROR self "should be a class to use GetClassIV"])
```

(GetClassIVHere

```
[LAMBDA (self varName prop)                                 (* smL "15-Jan-87 16:49")

          (* * Get the value of an IV from the class)

    (COND
       ((type? class self)
        (COND
           ((FMEMB varName (fetch localIVs of self))
            (ObjGetProp (FetchCIVDescr self varName)
                   prop))
           (T NotSetValue)))
       (T (ERROR self "should be a class to use GetClassIVHere"])
```

(GetClassValue

```
  [LAMBDA (self varName prop)                                    (* smL "27-May-86 14:01")

          (* * Returns the value of a class variable, activating getFn of an ActiveValue.)

    (ExtractRealValue self varName (FetchCVOnly self varName prop 'GetClassValue)
           prop
           'CV])
```

(GetClassValueOnly

```
  [LAMBDA (classRec varName prop)                                (* dgb%: "19-Nov-84 15:57")
                                                                 (* Returns the localState of a class variable.)
    (FetchCVOnly classRec varName prop])
```

(GetLocalState

```
  [LAMBDA (av self varName propName type)                        (* smL " 5-Aug-86 17:50")

          (* * Returns the local state of an activeValue -
          can handle either an ActiveValue object or an annotatedValue datatype)

    (_ (if (AnnotatedValue? av)
           then (fetch annotatedValue of av)
         else av)
       GetWrappedValueOnly self varName propName type])
```

(GetLocalStateOnly

```
  [LAMBDA (av)                                                   (* smL " 5-May-86 17:18")

          (* * Returns the localState of an activeValue. Does not check whether the value is itself active.
          -
          Can handle either an ActiveValue object or an annotatedValue datatype)

    (_ (if (type? annotatedValue av)
           then (fetch annotatedValue of av)
         else av)
       GetWrappedValueOnly])

)

(DEFINEQ
```

(PutCIVHere

```
  [LAMBDA (self varName value prop)                              (* dgb%: "17-Apr-84 12:56")
                                                                 (* Put a value locally even if not a local variable of class)
    [AddCIV self varName (COND
                            ((NULL prop)
                             value)
                            (T (GetClassIV self varName)))
           (COND
              (prop (LIST prop value]
    value])
```

(PutCVHere

```
  [LAMBDA (self varName value)                                   (* dgb%: "22-DEC-83 22:43")
                                                                 (* Put a ClassVariable locally with value if if not local now)
    (AddCV self varName value)
    value])
```

(PutClassIV

```
  [LAMBDA (self varName newValue propName)                       (* smL "18-Sep-86 14:08")

          (* * Store value in a class description of an IV. One may only store in values of local variables.)

    (COND
       ((NOT (Class? self))
        (ERROR self "not a class"))
       ([NOT (FMEMB varName (_ self ListAttribute 'IVs]
        (ERROR varName (CONCAT "is not a local IV, so cannot be changed in " self)))
       (T (ObjPutProp (FetchCIVDescr self varName)
                 propName newValue])
```

(PutClassValue

```
  [LAMBDA (self varName newValue propName)                       (* smL "10-Apr-87 10:31")

          (* * Puts value in a class variable. Activates putFn if an ActiveValue.)

    (LET [(classRec (if (type? class self)
                        then self
                      elseif (type? instance self)
```

```
                    then (Class self)
                  else (GetLispClass self]
        (for class in-supers-of classRec bind descr index oldValue eachtime (SETQ index
                                                                                (FindIndex varName
                                                                                       (fetch cvNames of class)))
           when index do (SETQ descr (GetNth (fetch cvDescrs of class)
                                             index))
                         (COND
                            ((NULL descr)                          (* Here if no value previously set for class variable.)
                             (SETQ descr (create IVDescr))
                             (PutNth (fetch cvDescrs of class)
                                     index descr)))
                         (SETQ oldValue (ObjGetProp descr propName))
                         (RETURN (ObjSetValue class varName newValue descr oldValue propName 'CV))
           finally                                                 (* Never found the CV)
                 (RETURN (_ classRec CVMissing self varName propName 'PutClassValue newValue])
```

(PutClassValueOnly

```
  [LAMBDA (self varName newValue propName)                         (* smL "15-Apr-87 14:04")

          (* * Stores a localState for an class variable. Never activates attached procedures or checks type of value.)

    (LET [(classRec (if (type? class self)
                        then self
                      elseif (type? instance self)
                        then (Class self)
                      else (GetLispClass self]
        (for class in-supers-of classRec bind descr index oldValue eachtime (SETQ index
                                                                                (FindIndex varName
                                                                                       (fetch cvNames of class)))
           when index do (SETQ descr (GetNth (fetch cvDescrs of class)
                                             index))
                         (COND
                            ((NULL descr)                          (* Here if no value previously set for class variable.)
                             (SETQ descr (create IVDescr))
                             (PutNth (fetch cvDescrs of class)
                                     index descr)))
                         (SETQ oldValue (ObjGetProp descr propName))
                         (RETURN (ObjPutProp descr propName newValue))
           finally                                                 (* Never found the CV)
                 (RETURN (_ classRec CVMissing self varName propName 'PutClassValueOnly newValue])

)

;;; IVs that are FIXPs

(DEFINEQ

## (GetNthValue

  [LAMBDA (self varIndex prop)                                     (* dgb%: "20-Nov-84 14:05")
    (COND
       ((FIXP varIndex)
        (FetchNthValue self varIndex prop))
       (T (GetValue self varIndex prop])

## (FetchNthValue

  [LAMBDA (self index propName)                                    (* smL "27-May-86 14:01")

          (* * Used to fetch value of indexed variable for the mixin class VarLength.)

    (LET [value (descr (CAR (FNTH (@ indexedVars)
                                  index]
         (COND
            ((NULL descr)
             NotSetValue)
            (T [SETQ value (COND
                              (propName (ObjGetProp descr propName))
                              (T (CAR descr]
               (ExtractRealValue self index value propName])

## (FetchNthValueOnly

  [LAMBDA (self index propName)                                    (* smL "27-May-86 13:04")

          (* * Used to fetch value of indexed variable for the mixin class VarLength.
          Like FetchNthValue except ignores activeValues.)

    (LET [(descr (CAR (FNTH (@ indexedVars)
                            index]
         (COND
            ((NULL descr)
             NotSetValue)
            ((ObjGetProp descr propName])
```

```
(FetchNthDescr!
  [LAMBDA (self index)                                          (* smL "17-Oct-85 11:48")
    (OR (FetchNthDescr self index)
        (PROG [last numVars (varList (LCONC NIL (GetValue self 'indexedVars]
              (SETQ numVars (LENGTH (CAR varList)))
              [for N from 1 to (IDIFFERENCE index numVars) do (SETQ varList (TCONC varList (SETQ last
                                                                                          (create IVDescr]
              (PutValueOnly self 'indexedVars (CAR varList))
              (RETURN last])

(FetchNthDescr
  [LAMBDA (self index)                                          (* dgb%: " 2-DEC-82 22:14")
    (CAR (FNTH (GetValue self 'indexedVars)
               index])

(PutNthValue
  [LAMBDA (self varIndex newValue propName)                     (* dgb%: "23-NOV-82 00:11")
                                                                (* Store away a value for an indexed variable)
    (COND
       ((NUMBERP varIndex)
        (StoreNthValue self varIndex newValue propName))
       (T (PutValue self varIndex newValue propName])

(StoreNthValue
  [LAMBDA (self index newValue propName)                        (* dgb%: " 2-DEC-82 18:35")

          (* * Store value for nth indexed variable. Used by objects having a VarLength mixin.)

    (PROG ((descr (FetchNthDescr! self index)))
          (RETURN (ObjSetValue self index newValue descr (ObjGetProp descr propName)
                          propName])

(StoreNthValueOnly
  [LAMBDA (self index newValue propName)                        (* edited%: "22-Dec-84 18:29")

          (* * Store away value for nth indexed variable. Used by objects having a VarLength mixin.
          Same as StoreNthValue except ignores activeValues.)

    (PROG [(descr (CAR (FNTH (@ indexedVars)
                             index]
          (RETURN (COND
                     ((NULL descr)                              (* Here if no value set yet.)
                      (PROG (varLst numVars)
                            (SETQ varLst (GetValueOnly self 'indexedVars))
                            (SETQ numVars (LENGTH varLst))      (* First allocate space for any vars of lower index.)
                            [SETQ varLst (APPEND varLst (for i from 1 to (SUB1 (IDIFFERENCE index numVars))
                                                          collect (CONS NotSetValue]
                                                                (* Then stick the newValue on the end.)
                            [SETQ varLst (NCONC1 varLst (COND
                                                          (propName (LIST NotSetValue propName newValue))
                                                          (T (CONS newValue]
                            (PutValueOnly self 'indexedVars varLst)
                            (RETURN newValue)))
                     (T                                         (* Usual case.)
                        (ObjPutProp descr propName newValue])

)


;;; Class property stuff

(DEFINEQ

(GetClass
  [LAMBDA (classRec propName)                                   (* dgb%: " 5-Dec-84 14:47")

          (* 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.)

    (_ classRec GetClassProp propName])

(GetClassOnly
  [LAMBDA (classRec propName)                                   (* smL "24-Sep-85 10:36")

          (* Maps through a class and its supers and returns property value with no activations.
          Returns NotSetValue if none found. If firstFoundFlg=T then returns CONS of value and flg indicating whether prop was
          found past first Class in inheritance chain)

    (COND
       ((NULL propName)                                         (* Return metaClass if no prop specified.)
        (Class classRec))
```

```
       (T (for c bind v in (CONS classRec (Supers classRec)) unless (NotSetValue (SETQ v (ObjGetProp (
                                                                                                GetClassDescr
                                                                                                c)
                                                                                             propName)))
             do
          (* * Return the first place wehre there is a real value (not = NotSetValue))
             (RETURN v)
          finally (RETURN NotSetValue])
```

(GetClassHere

```
  [LAMBDA (classRec propName)                                    (* dgb%: "27-NOV-82 04:05")
                                                                 (* Gets the class property here without activeValues or
                                                                 inheritance)
    (ObjGetProp (GetClassDescr classRec)
           propName])

)

(DEFINEQ
```

(DeleteClassProp

```
  [LAMBDA (class prop)                                           (* smL "18-Sep-86 14:44")
                                                                 (* Removes property from the class property list.)
    (ChangedClass class)
    (LET [(fakedDescr (create IVPropDescr
                              IVPropList _ (fetch otherClassDescription of class]
          (* Create a faked IVDescr so as to be able to use ObjRemProp.
          -
          To be changed later if MetaClass field is reformated to contain a full-fledge descr.)
         (PROG1 (ObjRemProp fakedDescr prop)
             (replace otherClassDescription of class with (fetch IVPropList of fakedDescr)))])

)

(DEFINEQ
```

(PutClass

```
  [LAMBDA (classRec newValue propName)                           (* smL "17-Apr-87 09:15")
          (* * Changes the class-properities; that is the ones seen after the metaclass)
    [COND
       [(NULL propName)                                          (* Change class if no propName given.)
        (replace metaClass of classRec with (OR (GetClassRec newValue)
                                                (AND (HELPCHECK newValue "is not a class. Type OK to replace
                                                                    metaclass of " classRec "with $Class")
                                                     ($ Class]
       (T (LET ((oldValue (GetClassOnly classRec propName)))
               (COND
                  ((type? annotatedValue oldValue)
                   (_ (fetch annotatedValue of oldValue)
                      PutWrappedValue classRec NIL newValue propName 'CLASS)
                   (ChangedClass classRec))
                  (T (PutClassOnly classRec newValue propName]
    newValue])
```

(PutClassOnly

```
  [LAMBDA (classRec newValue propName)                           (* smL "17-Apr-87 09:14")
    [COND
       [(NULL propName)
        (replace metaClass of classRec with (OR (GetClassRec newValue)
                                                (AND (HELPCHECK newValue "not a class. Type OK to replace
                                                                    metaclass of " classRec " with $Class.")
                                                     ($ Class]
       (T (PROG ((ocd (fetch otherClassDescription of classRec)))
               (COND
                  (ocd (LISTPUT ocd propName newValue))
                  (T (replace otherClassDescription of classRec with (LIST propName newValue]
    newValue])

)
```

;;; Method property stuff

```
(DEFINEQ
```

(GetMethod

```
  [LAMBDA (classRec selector propName)                           ; Edited 17-Jun-87 18:43 by smL
    ;; Maps through a classRec and its supers in order to find the value of a property on the method specified by selector.  Returns if property is set, or
    ;; NoValueFound if none found.  Invokes activeValues.
```

```
    (COND
       ((NULL propName)                                    ; Here to return a method name
        (FetchMethod classRec selector))
       (T (PROG (supers value index methObj (class classRec))
                (COND
                   ((NULL class)
                    (RETURN NIL)))
                (SETQ supers (Supers class))
            LP  [COND
                   ((SETQ index (FindSelectorIndex class selector))
                                                           ; Method is in class
                    (SETQ methObj (GetMethodObj class selector T))
                    [SETQ value (COND
                                   ((FMEMB propName (GetClassValue methObj 'ivProperties))
                                    (GetValue methObj propName))
                                   (T (GetValue methObj 'method propName]
                    (COND
                       ((ValueFound value)                 ; There is a value for property
                        (RETURN value]
                ON  (COND
                       ((SETQ class (pop supers))          ; If there is a Super, iterate around the Loop
                        (GO LP)))                          ; Returns NotSetValue if not found
                (RETURN NoValueFound])
```

(GetMethodOnly

```
  [LAMBDA (classRec selector propName)                     (* smL "30-Sep-86 10:31")
```

(* * Maps through a class and its supers and returns property value of method with no activations.
Returns NoValueFound if none found.)

```
    (for class in-supers-of classRec bind value index do (SETQ index (FindSelectorIndex class selector))
                                                         [COND
                                                            (index (SETQ value (GetMethodHere classRec selector
                                                                                              propName))
                                                                   (COND
                                                                      ((NOT (NotSetValue value))
                                                                       (RETURN value]
       finally (RETURN NoValueFound])
```

(GetMethodHere

```
  [LAMBDA (classRec selector propName)                     ; Edited 17-Jun-87 18:43 by smL
```

;;; Gets method property here or function if prop is NIL. Returns NotSetValue if not present in this class.

```
    (COND
       ((NULL propName)
        (FindLocalMethod classRec selector))
       (T (LET (methObj (index (FindSelectorIndex classRec selector)))
               (COND
                  ((NULL index)
                   NotSetValue)
                  (T (SETQ methObj (GetMethodObj classRec selector T))
                                                           ; Find or create method object
                     (COND
                        ((FMEMB propName (GetClassValue methObj 'ivProperties))
                         (GetIVHere methObj propName))
                        (T (GetIVHere methObj 'method propName])

)

(DEFINEQ
```

(PutMethod

```
  [LAMBDA (classRec selector newValue propName)            ; Edited 17-Jun-87 18:36 by smL
```

;; Puts a value for a property on of a local method for selector. Activates activeValues if found on methodObject

```
    (PROG (methObj (index (FindSelectorIndex classRec selector)))
          (RETURN (COND
                     ((NULL propName)
                      (DefineMethod classRec selector newValue))
                     ((NULL index)
                      (ERROR selector (CONCAT "not a local method of " classRec)))
                     (T (SETQ methObj (GetMethodObj classRec selector T))
                                                           ; Find or create method object
                        (COND
                           ((FMEMB propName (GetClassValue methObj 'ivProperties))
                            (PutValue methObj propName newValue))
                           (T (PutValue methObj 'method newValue propName)))
                        newValue])
```

(PutMethodOnly

```
  [LAMBDA (classRec selector newValue propName)            ; Edited 17-Jun-87 18:38 by smL
```

;; Puts a value for a property on of a local method for selector.

```
    (COND
       ((NULL propName)
        (DefineMethod classRec selector newValue))
       ((NULL (FindSelectorIndex classRec selector))
        (ERROR selector (CONCAT "not a local method of " classRec)))
       (T (LET ((methObj (GetMethodObj classRec selector T)))    ; Find or create method object
               (COND
                  ((FMEMB propName (GetClassValue methObj 'ivProperties))
                   (PutValueOnly methObj propName newValue))
                  (T (PutValueOnly methObj 'method newValue propName)))
               newValue])

)

;;; CHANGETRAN stuff for IV access

(PUTPROPS GetClassValueOnly SETFN PtClassValueOnly)

(PUTPROPS GetValueOnly SETFN PtValueOnly)

(PUTPROPS GetClassValue SETFN PutClsValue)

(PUTPROPS @ SETFN _@)

(PUTPROPS GetValue SETFN PtValue)

(DECLARE%: EVAL@COMPILE

(PUTPROPS PtClassValueOnly MACRO (args (SwitchPutArgs 'PutClassValueOnly args)))

(PUTPROPS PtValueOnly MACRO (args (SwitchPutArgs 'PutValueOnly args)))

(PUTPROPS PutClsValue MACRO (args (SwitchPutArgs 'PutClassValue args)))

(PUTPROPS PtValue MACRO (args (SwitchPutArgs 'PutValue args)))
)

(DEFINEQ

(SwitchPutArgs
  [LAMBDA (putFn args)                                    (* dgb%: "24-Oct-84 14:47")
    (LIST* putFn (CAR args)
           (CADR args)
           (COND
              ((CDDDR args)                               (* has four)
               (LIST (CADDDR args)
                     (CADDR args)))
              (T (CDDR args])

)

;;; UNDOable versions

(DEFINEQ

(/PutValue
  [LAMBDA (self varName newValue propName)                (* smL "18-Sep-86 13:03")

          (* * UNDOable version of PutValue)

    (UNDOSAVE (LIST '/PutValue self varName (GetValue self varName propName)
                    propName))
    (PutValue self varName newValue propName])

)

(DECLARE%: DONTEVAL@LOAD

(NEW/FN 'PutValue)
)

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

FUNCTION INDEX

PROPERTY INDEX

MACRO INDEX

OPTIMIZER INDEX