File created: 14-Aug-90 15:53:11 **{DSK}<usr>local>lde>loops>src>SYSTEM>LOOPSDEBUG.;3**

changes to: (CLASSES BreakOnPut BreakOnPutOrGet TraceOnPut TraceOnPutOrGet)
(VARS LOOPSDEBUGCOMS)

previous date: 13-Aug-90 16:04:23 {DSK}<usr>local>lde>loops>src>SYSTEM>LOOPSDEBUG.;2

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ LOOPSDEBUGCOMS
       ((* * Breaking and tracing vars and methods)
        (CLASSES BreakOnPut BreakOnPutOrGet TraceOnPut TraceOnPutOrGet)
        (METHODS BreakOnPut.PutWrappedValue BreakOnPutOrGet.GetWrappedValue Object.BreakIt Object.TraceIt
               TraceOnPut.PutWrappedValue TraceOnPutOrGet.GetWrappedValue)
        (FNS BreakMethod TraceMethod BreakIt TraceIt UnBreakIt)
        (DECLARE%: EVAL@COMPILE DONTCOPY (EXPORT (RECORDS brkRec))
               DONTEVAL@COMPILE DOCOPY (INITRECORDS brkRec))
        (VARS (BrokenVariables NIL))
        (GLOBALVARS BrokenVariables)
        (* * Clearing caches, just in case)
        (FNS)))
```

(* * Breaking and tracing vars and methods)

```
(DEFCLASSES BreakOnPut BreakOnPutOrGet TraceOnPut TraceOnPutOrGet)

(DEFCLASS BreakOnPut (MetaClass Class Edited%:                    (* smL " 5-May-86 17:51")
                           doc "This is the default metaClass for all classes")
       (Supers LocalStateActiveValue))

(DEFCLASS BreakOnPutOrGet (MetaClass Class Edited%:               (* smL " 5-May-86 17:53")
                               doc "This is the default metaClass for all classes")
       (Supers BreakOnPut))

(DEFCLASS TraceOnPut (MetaClass Class Edited%:                    (* smL " 5-May-86 17:55")
                           doc "This is the default metaClass for all classes")
       (Supers LocalStateActiveValue))

(DEFCLASS TraceOnPutOrGet (MetaClass Class Edited%:               (* smL " 5-May-86 17:55")
                               doc "This is the default metaClass for all classes")
       (Supers TraceOnPut))

(\BatchMethodDefs)

(METH BreakOnPut PutWrappedValue (containingObj varName newValue propName type)
      "Replace the value wrapped in the active value"
      (category (ActiveValue)))

(METH BreakOnPutOrGet GetWrappedValue (containingObj varName propName type)
      "Fetch the value wrapped in the active value"
      (category (ActiveValue)))

(METH Object BreakIt (varName propName type brkOnGetAlsoFlg)
      "makes an active value which will cause break when the on this value is to be changed. If
      brkOnGetAlsoFlg=T then will also break when value is fetched. Message on Object"
      (category (Object)))

(METH Object TraceIt (varName propName type traceGetAlsoFlg)
      "Makes an active value which will cause tracing when this variable is changed. Will also trace on
      fetches if traceGetAlsoFlg=T. Message on Object"
      (category (Object)))

(METH TraceOnPut PutWrappedValue (containingObj varName newValue propName type)
      "Replace the value wrapped in the active value"
      (category (ActiveValue)))

(METH TraceOnPutOrGet GetWrappedValue (containingObj varName propName type)
      "Fetch the value wrapped in the active value"
      (category (ActiveValue)))

(Method ((BreakOnPut PutWrappedValue) self containingObj varName newValue propName type)
                                                                  ; smL 7-Apr-87 18:14
   "Replace the value wrapped in the active value"
   (LET ((oldValue (GetLocalState self containingObj varName newValue propName)))
        (if propName
            then (CL:BREAK "~&Setting property ~A of ~A ~A of ~S~&Old value: ~S~&New value: ~S" propName
                       (OR type "IV")
                       varName containingObj oldValue newValue)
```

```
          else (CL:BREAK "~&Setting ~A ~A of ~S~&Old value: ~S~&Value: ~S" (OR type "IV")
                      varName containingObj oldValue newValue)))
```

;; jrb - Maybe this is the right thing...

```
   (_Super))

(Method ((BreakOnPutOrGet GetWrappedValue) self containingObj varName propName type)
                                                                    ; smL  5-May-86 18:02
   "Fetch the value wrapped in the active value"
   (LET ((value (_Super)))
        (DECLARE (SPECVARS value))
        (if propName
            then (CL:BREAK "~&Fetching property ~A of ~A ~A of ~S~&Value: ~S" propName (OR type "IV")
                        varName containingObj value)
          else (CL:BREAK "~&Fetching ~A ~A of ~S~&Value: ~S" (OR type "IV")
                      varName containingObj value))
        value))

(Method (((Object BreakIt) self varName propName type brkOnGetAlsoFlg)
                                                                    ; smL  5-Aug-86 16:56
   "makes an active value which will cause break when the on this value is to be changed. If
   brkOnGetAlsoFlg=T then will also break when value is fetched. Message on Object"
   (SETQ type (OR type 'IV))
   (push BrokenVariables (create brkRec
                                 brkSelf _ self
                                 brkVarName _ varName
                                 brkAv _ (if brkOnGetAlsoFlg
                                             then (_New
                                                    ($ BreakOnPutOrGet)
                                                    AddActiveValue self varName propName type)
                                           else (_New
                                                  ($ BreakOnPut)
                                                  AddActiveValue self varName propName type))
                                 brkPropName _ propName
                                 brkType _ type))
   self)

(Method (((Object TraceIt) self varName propName type traceGetAlsoFlg)
                                                                    ; smL  5-Aug-86 16:57
   "Makes an active value which will cause tracing when this variable is changed. Will also trace on
   fetches if traceGetAlsoFlg=T. Message on Object"
   (SETQ type (OR type 'IV))
   (push BrokenVariables (create brkRec
                                 brkSelf _ self
                                 brkVarName _ varName
                                 brkAv _ (if traceGetAlsoFlg
                                             then (_New
                                                    ($ TraceOnPutOrGet)
                                                    AddActiveValue self varName propName type)
                                           else (_New
                                                  ($ TraceOnPut)
                                                  AddActiveValue self varName propName type))
                                 brkPropName _ propName
                                 brkType _ type))
   self)

(Method (((TraceOnPut PutWrappedValue) self containingObj varName newValue propName type)
                                                                    ; smL  7-Apr-87 18:18
   "Replace the value wrapped in the active value"
   (LET ((oldValue (GetLocalState self containingObj varName newValue propName)))
        (if propName
            then (CL:FORMAT *TRACE-OUTPUT* "~&Setting property ~A of ~A ~A of ~S~&Old value: ~S~&New value:
                        ~S~&" propName (OR type "IV")
                        varName containingObj oldValue newValue)
          else (CL:FORMAT *TRACE-OUTPUT* "~&Setting ~A ~A of ~S~&Old value: ~S~&Value: ~S~&" (OR type "IV")
                      varName containingObj oldValue newValue)))
   (_Super))

(Method (((TraceOnPutOrGet GetWrappedValue) self containingObj varName propName type)
                                                                    ; smL 15-Jan-87 16:31
   "Fetch the value wrapped in the active value"
   (LET ((value (_Super)))
        (DECLARE (SPECVARS value))
        (if propName
            then (CL:FORMAT *TRACE-OUTPUT* "Fetching property ~A of ~A ~A of ~S~&Value: ~S~%%" propName
                        (OR type "IV")
                        varName containingObj value)
          else (CL:FORMAT *TRACE-OUTPUT* "Fetching ~A ~A of ~S~&Value: ~S~%%" (OR type "IV")
                      varName containingObj value))
        value))

(\UnbatchMethodDefs)

(DEFINEQ

(BreakMethod
  [LAMBDA (className selector)                                      (* smL "29-May-86 18:26")
```

```
    (LET [(class (GetObjectRec (GoodClassName className NIL T]
         (if (type? class class)
             then [APPLY* 'BREAK (OR (FetchMethod (GetObjectRec (GoodClassName className NIL T))
                                                  selector)
                                     (ERROR selector (CONCAT " not found in " className]
          else (ERROR className "not the name of a class"])
```

(TraceMethod

```
  [LAMBDA (className selector)                                        (* smL "29-May-86 18:25")
    (LET [(class (GetObjectRec (GoodClassName className NIL T]
         (if (type? class class)
             then [APPLY* 'TRACE (OR (FetchMethod class selector)
                                     (ERROR selector (CONCAT " not found in " className]
          else (ERROR className "not the name of a class"])
```

(BreakIt

```
  [LAMBDA (self varName propName type brkOnGetAlsoFlg)                (* mjs%: " 2-AUG-82 15:36")

          (* makes an active value which will cause break when the on this value is to be changed.
          If brkOnGetAlsoFlg=T then will also break when value is fetched.
          Sends message to self)

    (_ self BreakIt varName propName type brkOnGetAlsoFlg])
```

(TraceIt

```
  [LAMBDA (self varName propName type traceGetAlsoFlg)                (* mjs%: " 2-AUG-82 15:34")

          (* makes an active value which will cause tracing when this variable is changed.
          Will also trace on fetches if traceGetAlsoFlg=T.)

    (_ self TraceIt varName propName type traceGetAlsoFlg])
```

(UnBreakIt

```
  [LAMBDA (self varName propName type)                                ; Edited 13-Aug-90 16:03 by jds

          (* Finds the active value which has been used to Break or trace this value on the list BrokenVariables, extracts the old
          value, and removes same from list. Does all of them if self=NIL)

    (LET [(type (OR type 'IV]
         (COND
            [(NULL self)
             (for V in BrokenVariables do (_ (fetch (brkRec brkAv) of V)
                                             DeleteActiveValue
                                             (fetch (brkRec brkSelf) of V)
                                             (fetch (brkRec brkVarName) of V)
                                             (fetch (brkRec brkPropName) of V)
                                             (fetch (brkRec brkType) of V))
                 finally (RETURN (PROG1 BrokenVariables (SETQ BrokenVariables NIL]
            (T (for V in BrokenVariables when (AND (EQ self (fetch (brkRec brkSelf) of V))
                                                   (EQ varName (fetch (brkRec brkVarName) of V))
                                                   (EQ propName (fetch (brkRec brkPropName) of V))
                                                   (EQ type (fetch (brkRec brkType) of V)))
                  do (_ (fetch (brkRec brkAv) of V)
                        DeleteActiveValue
                        (fetch (brkRec brkSelf) of V)
                        (fetch (brkRec brkVarName) of V)
                        (fetch (brkRec brkPropName) of V)
                        (fetch (brkRec brkType) of V))
                     (SETQ BrokenVariables (DREMOVE V BrokenVariables))
                     (RETURN (LIST self varName propName))
                  finally (HELPCHECK (LIST self varName propName)
                                "not broken.  Type OK to go on."])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(RECORD brkRec (brkSelf brkVarName brkAv brkPropName brkType)
       (SYSTEM))
)
DOCOPY)

;; END EXPORTED DEFINITIONS

(RPAQQ BrokenVariables NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY
```

```
(GLOBALVARS BrokenVariables)
)
```

(* * Clearing caches, just in case)

```
(PUTPROPS LOOPSDEBUG COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1990))
```

FUNCTION INDEX

VARIABLE INDEX

RECORD INDEX