File created: 16-Sep-88 17:51:17 **{DSK}<LISPFILES2>SYSTEM>MLOOPSPATCH.;14**

changes to: (VARS MLOOPSPATCHCOMS)
(FNS CopyInstance)
(OPTIMIZERS _ SEND)

previous date: 27-Jul-88 15:28:51 {DSK}<LISPFILES2>SYSTEM>MLOOPSPATCH.;13

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1988 by Xerox Corporation.  All rights reserved.

(RPAQQ MLOOPSPATCHCOMS
       [(FUNCTIONS LOOPS-FUNCALL Method _Super _Super? _SuperFringe SubclassResponsibility)
        (OPTIMIZERS LOOPS-FUNCALL _ SEND)
        (FNS CopyDeepDescr CopyInstance DualSubItems GetSuperClassValue INSTALL-METHOD-FN LatticeBrowserExpandFn
             MakeMethodMenu NiceMenu SelectFile)
        (INITVARS (SelectFileMenu NIL))
        (METHODS Class.MakeEditSource LatticeBrowser.GetNodeList Object.SaveInstance? Window.ItemMenu)
        (VARIABLES EditClassMethodsFlg)
        (ADVISE LAYOUTGRAPH)
        (FNS Cached-GetIVValue Cached-PutIVValue Cached-FetchMethodOrHelp)
        (PROP FILETYPE MLOOPSPATCH)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA SelectFile])

(DEFMACRO LOOPS-FUNCALL (FN &REST ARGS)
   ;; The optimizer for this doesn't make sure FN is evaluated first, but that's ok for loops and save a GENSYM and binding
   `(APPLY* ,FN ,@ARGS))

(DEFDEFINER (Method [:NAME (CL:LAMBDA (method-body)
                                 (CL:MULTIPLE-VALUE-BIND (class-name selector)
                                     (PARSE-METHOD-BODY method-body)
                                   (MethName class-name selector))])
    METHOD-FNS (&WHOLE method-body)
   (CL:MULTIPLE-VALUE-BIND (class-name selector args declarations forms doc qualifiers method-type)
       (PARSE-METHOD-BODY method-body)
     (CL:ASSERT (Class? ($! class-name)))
     [LET (function-name function-type body)
          ;; Compute the name of the function
          (SETQ function-name (MethName class-name selector))
          ;; Compute the type of the function
          (SETQ function-type (OR (LISTGET qualifiers :FUNCTION-TYPE)
                                  :IL))
          (CL:CHECK-TYPE function-type (CL:MEMBER :CL :IL))
          ;; Get the body of the function, with the top level comments removed
          [SETQ body `(CL:COMPILER-LET [(*ArgsOfMethodBeingCompiled* ',args)
                                        (*ClassNameOfMethodOwner* ',class-name)
                                        (*SelectorOfMethodBeingCompiled* ',selector)
                                        (*SelfOfMethodBeingCompiled* ',(CAR args]
                       ,.forms]
          ;; Build the function definition form
          `(PROGN ,(CL:ECASE function-type
                      (:CL `(CL:DEFUN ,function-name ,args
                               ,@declarations ,body))
                      (:IL `(DEFINEQ [,function-name (LAMBDA ,args
                                                       ,@declarations
                                                       ,body])))
                   (INSTALL-METHOD-FN ',class-name ',selector ',function-name ',(CDR args)
                         ',doc)
                   ',function-name]))

(DEFMACRO _Super (&REST Send-Super-Args)
   (DECLARE (SPECIAL *ArgsOfMethodBeingCompiled* *ClassNameOfMethodOwner* *SelectorOfMethodBeingCompiled*
                     *SelfOfMethodBeingCompiled*))
   [COND
      [(NULL Send-Super-Args)                                    ; Args default to args of the method
       `(LOOPS-FUNCALL [FindSuperMethod ,*SelfOfMethodBeingCompiled* ',*SelectorOfMethodBeingCompiled*
                               (LOADTIMECONSTANT (OldClass ,*ClassNameOfMethodOwner*]
              ,.*ArgsOfMethodBeingCompiled*]
      ((NEQ *SelectorOfMethodBeingCompiled* (CADR Send-Super-Args))
                                                                 ; Selectors must match
       (ERROR "Selector to _Super does not match method selector" (CADR Send-Super-Args)))
```

```
      ((NEQ *SelfOfMethodBeingCompiled* (CAR Send-Super-Args))   ; Self must match
       (ERROR "Can't _Super to other than first arg of method" (CDR Send-Super-Args)))
      (T                                                        ; Args differ
         (APPEND `(LOOPS-FUNCALL [FindSuperMethod ,*SelfOfMethodBeingCompiled* '
                                                                              ,
                                                                              *SelectorOfMethodBeingCompiled*
                                       (LOADTIMECONSTANT (OldClass ,*ClassNameOfMethodOwner*]
                             ,*SelfOfMethodBeingCompiled*)
                 (CDDR Send-Super-Args])


(DEFMACRO _Super? (&REST Send-Super-Args)
   (DECLARE (SPECIAL *ArgsOfMethodBeingCompiled* *ClassNameOfMethodOwner* *SelectorOfMethodBeingCompiled*
                     *SelfOfMethodBeingCompiled*))
   [COND
      [(NULL Send-Super-Args)                                   ; Args default to args of the method
       `(LOOPS-FUNCALL (FindSuperMethod ,*SelfOfMethodBeingCompiled* ',*SelectorOfMethodBeingCompiled*
                               (LOADTIMECONSTANT (OldClass ,*ClassNameOfMethodOwner*))
                               (FUNCTION NILL))
              ,.*ArgsOfMethodBeingCompiled*]
      ((NEQ *SelectorOfMethodBeingCompiled* (CADR Send-Super-Args))
                                                                ; Selectors must match
       (ERROR "Selector to _Super does not match method selector" (CADR Send-Super-Args)))
      ((NEQ *SelfOfMethodBeingCompiled* (CAR Send-Super-Args))   ; Self must match
       (ERROR "Can't _Super? to other than first arg of method" (CDR Send-Super-Args)))
      (T                                                        ; Args differ
         (APPEND `(LOOPS-FUNCALL (FindSuperMethod ,*SelfOfMethodBeingCompiled* '
                                                                              ,
                                                                              *SelectorOfMethodBeingCompiled*
                                       (LOADTIMECONSTANT (OldClass ,*ClassNameOfMethodOwner*))
                                       (FUNCTION NILL))
                             ,*SelfOfMethodBeingCompiled*)
                 (CDDR Send-Super-Args])


(DEFMACRO _SuperFringe (&REST Send-Super-Args)
   (DECLARE (SPECIAL *ArgsOfMethodBeingCompiled* *ClassNameOfMethodOwner* *SelectorOfMethodBeingCompiled*
                     *SelfOfMethodBeingCompiled*))
   [COND
      [(NULL Send-Super-Args)                                   ; Args default to args of the method
       `(for cls in [fetch localSupers of (LOADTIMECONSTANT (OldClass ,*ClassNameOfMethodOwner*]
           do (LOOPS-FUNCALL (OR (FetchMethod cls ',*SelectorOfMethodBeingCompiled*)
                                 (FUNCTION NILL))
                    ,.*ArgsOfMethodBeingCompiled*]
      ((NEQ *SelectorOfMethodBeingCompiled* (CADR Send-Super-Args))
                                                                ; Selectors must match
       (ERROR "Selector to _Super does not match method selector" (CADR Send-Super-Args)))
      ((NEQ *SelfOfMethodBeingCompiled* (CAR Send-Super-Args))   ; Self must match
       (ERROR "Can't _SuperFringe to other than first arg of method" (CDR Send-Super-Args)))
      (T `(for cls in [fetch localSupers of (LOADTIMECONSTANT (OldClass ,*ClassNameOfMethodOwner*]
             bind (argList _ (MAPCAR '(,(CAR Send-Super-Args)
                                       ,.(CDDR Send-Super-Args))
                                     (FUNCTION EVAL)))
             do (APPLY (OR (FetchMethod cls ',*SelectorOfMethodBeingCompiled*)
                           (FUNCTION NILL))
                       argList])


(DEFMACRO SubclassResponsibility ()
   (DECLARE (SPECIAL *ArgsOfMethodBeingCompiled* *ClassNameOfMethodOwner* *SelectorOfMethodBeingCompiled*
                     *SelfOfMethodBeingCompiled*))
   `(HELPCHECK (CONCAT "Method " ,*SelectorOfMethodBeingCompiled* " of class " ,*ClassNameOfMethodOwner* " needs
                       to be defined in class ")
          (_ ,*SelfOfMethodBeingCompiled* ClassName)))


(DEFOPTIMIZER LOOPS-FUNCALL (FN &REST ARGS)
                                   ;; This version doesn't make sure FN is evaluated first, but that's ok for loops and save a GENSYM and
                                   ;; binding
                                   `((OPCODES APPLYFN)
                                     ,@ARGS
                                     ,(LENGTH ARGS)
                                     ,FN))


(DEFOPTIMIZER _ (object selector &REST args)
              [if (AND *Compile-Local-Message-Cache* *BYTECOMPILER-IS-EXPANDING*)
                  then `(LET [(/\obj/\ ,object)
                              (*LOOPS-INLINE-METHOD-CACHE* (LOADTIMECONSTANT (\Make-Method-Cache-Entry]
                             (DECLARE (LOCALVARS /\obj/\ *LOOPS-INLINE-METHOD-CACHE*))
                             (LOOPS-FUNCALL (COND
                                                ((AND (Object? /\obj/\)
                                                      (EQ (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE* 0)
                                                          (fetch (OBJECT CLASS) of /\obj/\)))
                                                                ; A cache hit
                                                 (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE* 2))
```

```
                                        (T              ; A cache miss
                                           (Cached-FetchMethodOrHelp /\obj/\ ',selector
                                                  *LOOPS-INLINE-METHOD-CACHE*)))
                                  /\obj/\
                                  ,@args))
                ;; `((OPENLAMBDA (/\obj/\ *LOOPS-INLINE-METHOD-CACHE*) (DECLARE (LOCALVARS . T)) (CL:FUNCALL (if
                ;; (AND (Object? /\obj/\) (EQ (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE* 0) (fetch (OBJECT CLASS) of
                ;; /\obj/\))) then ; A cache hit (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE* 2) else ; A cache miss
                ;; (Cached-FetchMethodOrHelp /\obj/\ ',selector *LOOPS-INLINE-METHOD-CACHE*)) /\obj/\ ,@args)) ,object
                ;; (LOADTIMECONSTANT (\Make-Method-Cache-Entry)))

          elseif *BYTECOMPILER-IS-EXPANDING*
            then `(LET ((/\obj/\ ,object))
                       (DECLARE (LOCALVARS /\obj/\))
                       (LOOPS-FUNCALL (FetchMethodOrHelp /\obj/\ ',selector)
                              /\obj/\
                              ,@args))
          else
          (LET* [(obj (if (LITATOM object)
                          then object
                        else (GENSYM)))
                 [bindings (if (EQ obj object)
                               then NIL
                             else `((,obj ,object]
                 (localVars (for binding in bindings collect (CAR binding]
                (if *Compile-Local-Message-Cache*
                    then `(LET (,@bindings)
                               ;; ,@(if localVars then `((DECLARE (LOCALVARS ,@localVars))) else NIL)
                               (DECLARE (LOCALVARS . T))
                               (LOOPS-FUNCALL (LET [(*LOOPS-INLINE-METHOD-CACHE* (LOADTIMECONSTANT
                                                                                   (
                                                                                 \Make-Method-Cache-Entry
                                                                                   ]
                                              ;; This bogus SPECVARS stuff is here to prevent the compiler from thinking that the in-line cache is a
                                              ;; quoted constant.  Note that (almost) no user code gets called within this binding, so it is pretty safe.
                                              ;; (The potential exception is when the user has redefined the FetchMethodOrHelp method).
                                                    ;; (DECLARE (SPECVARS *LOOPS-INLINE-METHOD-CACHE*))
                                                    (if [AND (Object? ,obj)
                                                             (EQ (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE*
                                                                         0)
                                                                 (fetch (OBJECT CLASS) of ,obj]
                                                      then
                                                       ; A cache hit
                                                        (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE* 2)
                                                    else
                                                       ; A cache miss
                                                       (Cached-FetchMethodOrHelp ,obj
                                                              ',selector *LOOPS-INLINE-METHOD-CACHE*)
                                                    ))
                                              ,obj
                                              ,@args))
                      elseif bindings
                        then `(LET (,@bindings)
                                   (DECLARE (LOCALVARS ,@localVars))
                                   (LOOPS-FUNCALL (FetchMethodOrHelp ,obj ',selector)
                                          ,obj
                                          ,@args))
                      else `(LOOPS-FUNCALL (FetchMethodOrHelp ,obj ',selector)
                                  ,obj
                                  ,@args])


(DEFOPTIMIZER SEND (object selector &REST args)
          [if (AND *Compile-Local-Message-Cache* *BYTECOMPILER-IS-EXPANDING*)
              then `(LET [(/\obj/\ ,object)
                          (*LOOPS-INLINE-METHOD-CACHE* (LOADTIMECONSTANT (\Make-Method-Cache-Entry]
                         (DECLARE (LOCALVARS /\obj/\ *LOOPS-INLINE-METHOD-CACHE*))
                         (LOOPS-FUNCALL (COND
                                           ((AND (Object? /\obj/\)
                                                 (EQ (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE* 0)
                                                     (fetch (OBJECT CLASS) of /\obj/\)))
                                                 ; A cache hit
                                            (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE* 2))
                                           (T       ; A cache miss
                                              (Cached-FetchMethodOrHelp /\obj/\ ',selector
                                                     *LOOPS-INLINE-METHOD-CACHE*)))
                                  /\obj/\
                                  ,@args))
                ;; `((OPENLAMBDA (/\obj/\ *LOOPS-INLINE-METHOD-CACHE*) (DECLARE (LOCALVARS . T)) (CL:FUNCALL
                ;; (if (AND (Object? /\obj/\) (EQ (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE* 0) (fetch (OBJECT
                ;; CLASS) of /\obj/\))) then ; A cache hit (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE* 2) else ; A cache
                ;; miss (Cached-FetchMethodOrHelp /\obj/\ ',selector *LOOPS-INLINE-METHOD-CACHE*)) /\obj/\ ,@args))
                ;; ,object (LOADTIMECONSTANT (\Make-Method-Cache-Entry)))
```

```
                        elseif *BYTECOMPILER-IS-EXPANDING*
                          then `(LET ((/\obj/\ ,object))
                                     (DECLARE (LOCALVARS /\obj/\))
                                     (LOOPS-FUNCALL (FetchMethodOrHelp /\obj/\ ',selector)
                                            /\obj/\
                                            ,@args))
                        else
                        (LET*
                         [(obj (if (LITATOM object)
                                   then object
                                 else (GENSYM)))
                          [bindings (if (EQ obj object)
                                        then NIL
                                      else `(,obj ,object]
                          (localVars (for binding in bindings collect (CAR binding]
                         (if *Compile-Local-Message-Cache*
                             then `(LET (,@bindings)
                                        ;; ,@(if localVars then '((DECLARE (LOCALVARS ,@localVars))) else NIL)
                                        (DECLARE (LOCALVARS . T))
                                        (LOOPS-FUNCALL (LET [(*LOOPS-INLINE-METHOD-CACHE* (LOADTIMECONSTANT
                                                                                              (
                                                                                            \Make-Method-Cache-Entry
                                                                                              ]
                                        ;; This bogus SPECVARS stuff is here to prevent the compiler from thinking that the in-line cache is a
                                        ;; quoted constant.  Note that (almost) no user code gets called within this binding, so it is pretty safe.
                                        ;; (The potential exception is when the user has redefined the FetchMethodOrHelp method).
                                                    ;; (DECLARE (SPECVARS *LOOPS-INLINE-METHOD-CACHE*))
                                                    (if [AND (Object? ,obj)
                                                             (EQ (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE*
                                                                              0)
                                                                 (fetch (OBJECT CLASS) of ,obj]
                                                        then
                                                          ; A cache hit
                                                          (\GETBASEPTR *LOOPS-INLINE-METHOD-CACHE* 2)
                                                      else
                                                          ; A cache miss
                                                          (Cached-FetchMethodOrHelp ,obj
                                                                 ',selector *LOOPS-INLINE-METHOD-CACHE*)
                                                     ))
                                                  ,obj
                                                  ,@args))
                           elseif bindings
                             then `(LET (,@bindings)
                                        (DECLARE (LOCALVARS ,@localVars))
                                        (LOOPS-FUNCALL (FetchMethodOrHelp ,obj ',selector)
                                               ,obj
                                               ,@args))
                           else `(LOOPS-FUNCALL (FetchMethodOrHelp ,obj ',selector)
                                       ,obj
                                       ,@args])

(DEFINEQ

(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])
```

(DualSubItems

```
  [LAMBDA (menu item)                                    ; Edited 21-Jun-88 17:30 by TAL
```

;; A menu WHENSELECTEDFN which allows differential selection on left and middle button. For such differential selection item should be of form ;; --- (itemSeenInMenu (leftValue midValue)) --- where midValue can be an atom which is directly returned when item is selected with middle, or ;; midValue can be an itemList, which will be displayed in a subselection menu ; Made this recognize standard subitem format also -- TAL

```
    (PROG (it it1)
          (RETURN (COND
                     [[OR (NLISTP item)
                          (NLISTP (SETQ it (CADR item)))
                          (EQ (SETQ it1 (CAR it))
                              'QUOTE)
                          (EQ it1 'PROGN)
                          (NLISTP (SETQ it1 (CADR it]
                      (COND
                         ((AND (LISTP item)
                               (LISTP (SETQ it (CADDDR item)))
                               (EQ (CAR it)
                                   'SUBITEMS))
                          (CDR it]
                     (T it1])
```

(GetSuperClassValue

```
  [LAMBDA (self varName propName activeVal)              ; Edited 14-Jun-88 15:47 by TAL
    (LET* [(class (COND
                     ((type? instance self)
                      (Class self))
                     (T self)))
           (superList (for sup on (CONS class (Supers class))
                         do (if (AND (_ (CAR sup)
                                        HasCV varName)
                                     (HasAV (GetCVHere (CAR sup)
                                                       varName propName)
                                            activeVal))
                                then (RETURN (CDR sup]
          (for c in superList bind value do [if [AND (_ c HasCV varName)
                                                     (NOT (NotSetValue (SETQ value (GetCVHere c varName propName]
                                                then (RETURN (ExtractRealValue c varName value propName
                                                                               'CV]
            finally (RETURN NotSetValue])
```

(INSTALL-METHOD-FN

```
  [LAMBDA (class-name selector fn-name args doc)         ; Edited 21-Jun-88 17:22 by TAL
    (LET* ((class ($! class-name))
           (methObj (GetMethodObj class selector T))
           whereisClasses)
          (AddMethod class selector fn-name)
          (change (@ methObj method)
                  fn-name)
          (change (@ methObj args)
                  args)
          (change (@ methObj doc)
                  doc)
          (if (NOT (\Loading-File?))
              then (SETQ LASTWORD fn-name)
                   ;; TAL - Removed (COND ((AND (NULL (WHEREIS fn-name 'METHODS)) (SETQ whereisClasses (WHEREIS class-name
                   ;; 'CLASSES))) (ADDTOFILE fn-name 'METHODS (CAR whereisClasses))))
                   )
          ;; JRB - Removed (MARKASCHANGED fn-name 'METHODS 'DEFINED); I think in the world of METHOD-FNS this marking is not needed
          ;; (this DOES worry me, though).
          (UNMARKASCHANGED fn-name 'FNS)
          (UNMARKASCHANGED fn-name 'INSTANCES)
          fn-name])
```

(LatticeBrowserExpandFn

```
  [LAMBDA (window)                                       ; Edited 14-Jun-88 12:54 by TAL
```

```
;; When a browser window is expanded, it should be recomputed
(LET [(self (WINDOWPROP window 'LoopsWindow]
     (AND UpdateClassBrowsers? (_ self RecomputeInPlace])
```

(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])
```

(NiceMenu

```
[LAMBDA (items title)                                       ; Edited 13-Jun-88 18:56 by TAL
  (COND
     ((NULL items)
      (PROMPTPRINT "No items for " title)
      NIL)
     (T (LET [(columns (ADD1 (IQUOTIENT (FLENGTH items)
                                        35]
              (MENU (create MENU
                            CHANGEOFFSETFLG _ T
                            ITEMS _ (ColumnateMenuItems items columns)
                            TITLE _ title
                            MENUCOLUMNS _ columns])
```

(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])

)

(RPAQ? SelectFileMenu NIL)

(\BatchMethodDefs)

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

(METH LatticeBrowser GetNodeList (browseList goodList)
      "Compute the node data structures of the tree starting at browseList. If goodList is given, only
      include elements of it. If goodList=T make it be browseList."
      (category (LatticeBrowser)))

(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 Window ItemMenu (items title)
      "Create a simple (one level) menu which will not overflow height of screen"
      (category (Window)))

(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))
          (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 ((LatticeBrowser GetNodeList) self browseList goodList)          ; smL 21-Mar-85 14:09
   "Compute the node data structures of the tree starting at browseList. If goodList is given, only
   include elements of it. If goodList=T make it be browseList."
   (DECLARE (GLOBALVARS WHITESHADE))
   (COND
      ((EQ goodList T)
       (SETQ goodList browseList)))
   (PROG (subs pair node [oldNodes (fetch GRAPHNODES of (WINDOWPROP (@ window)
```

```
                                                                'GRAPH]
            (objList (CONS)))
```

;; first make objList which is a list of pairs (object . objName). objName will be used as a title for a node in the browser. This structure will be ;; replaced by a graphNode when it is processed. The nodeID of the graphNode will be the object, and the label will be the name.

```
      (for objOrName in browseList do (AND (SETQ pair (_ self ObjNamePair objOrName))
                                           (NOT (FASSOC (CAR pair)
                                                        (CAR objList)))
                                           (TCONC objList pair)))
```

;; Now MAP ON list so pair can be replaced by graphNode

```
      (for pair name obj subObjs on (CAR objList) when (NLISTP (SETQ name (CDAR pair)))
         do (SETQ subObjs (CONS))
            [for sub objPair obj1 in (_ self GetSubs (SETQ obj (CAAR pair)))
               do ;; ObjNamePair returns NIL for destroyed objects.  include only members of goodList in subs if given.  Add to objList only
                  ;; once
                  (SETQ obj1 (COND
                                ((EQ (CAR sub)
                                     'Link% Parameters)
                                 (CADR sub))
                                (T sub)))
                  (COND
                     ((SETQ objPair (_ self ObjNamePair obj1))
                      (COND
                         ((NOT (FASSOC obj1 (CAR objList)))
                          (TCONC objList objPair)))
                      (TCONC subObjs sub]
            [RPLACA pair (SETQ node (OR (FASSOC obj oldNodes)
                                        (create GRAPHNODE
                                               NODEID _ obj
                                               NODEBORDER _ (PROGN (LIST (ADD1 (@ |::BoxLineWidth|))
                                                                         WHITESHADE)
                                                                   ; This makes graphs too big -- TAL
                                                                   NIL]
            (replace TONODES of node with (CAR subObjs))
            (replace NODELABEL of node with name)
            (replace NODEFONT of node with (@ browseFont))
            (replace NODEWIDTH of node with NIL)
            (replace NODEHEIGHT of node with NIL))
      (RETURN (CAR objList))))
```

```
(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 ((Window ItemMenu) self items title)                     ; dgb: 21-Apr-84 09:31
   "Create a simple (one level) menu which will not overflow height of screen"
   (LET [(columns (ADD1 (IQUOTIENT (ITIMES (FONTHEIGHT MENUFONT)
                                           (LENGTH items))
                                   750]
        (create MENU
               ITEMS _ (ColumnateMenuItems items columns)
               MENUCOLUMNS _ columns
               TITLE _ title
               CHANGEOFFSETFLG _ T)))

(\UnbatchMethodDefs)


(CL:DEFVAR EditClassMethodsFlg NIL
   "Include method list in class editor if T")

[XCL:REINSTALL-ADVICE 'LAYOUTGRAPH :BEFORE '((:LAST (COND
                                                      ((NULL PERSONALD)
                                                       (SETQ PERSONALD -2]

(READVISE LAYOUTGRAPH)

(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-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-FetchMethodOrHelp

```
  [LAMBDA (self selector classCache)                       (* smL "20-Jun-86 13:13")

          (* * Get the method for this instance and selector, and update the cache.)

    (DECLARE (LOCALVARS class method))
    (LET* ((class (Class self))
           (method (FetchMethod class selector)))
          (if method
              then (\PUTBASEPTR classCache 0 class)
                   (\PUTBASEPTR classCache 2 method)
            else (_ self MethodNotFound selector])

)

(PUTPROPS MLOOPSPATCH FILETYPE :FAKE-COMPILE-FILE)

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA SelectFile)
)

(PUTPROPS MLOOPSPATCH COPYRIGHT ("Xerox Corporation" 1988))
```

FUNCTION INDEX

MACRO INDEX

OPTIMIZER INDEX

VARIABLE INDEX

PROPERTY INDEX

ADVICE INDEX

DEFINER INDEX