... from <http://hobbes.nmsu.edu/pub/os2/dev/orexx/> - **orx7.zip/class_ref.cmd** ...

```
/* maintains an anchor manager per class/given leadin:
   - created anchor name is a unique string, hence "surrogate-string"
   - sort of creating hash-values for any object  */
:: CLASS anchor                    PUBLIC
:: METHOD init                  CLASS
:: METHOD AnchorDir             CLASS ATTRIBUTE /* keeps anchor objects for diff. classes  */
:: METHOD init
:: METHOD getAnchorName                    /* returns a surrogate string for given object*/
:: METHOD ObjCounter                       /* count of number of objects                 */
:: METHOD AnchObjTable                ATTRIBUTE /* AnchObjTable[ Object ]=surrogate-string */


/* an interface class for geting/seting anchor names:
   - all methods are class methods, hence no instances necessary */
:: CLASS ref                       PUBLIC          /* an Anchor-manager    */
:: METHOD INIT CLASS
:: METHOD setOfAnchorNames      CLASS  ATTRIBUTE /* explicitly created via createReference */
:: METHOD setOfReferences       CLASS  ATTRIBUTE /* explicitly asked for via getReference  */
:: METHOD getAnchorObject       CLASS  PRIVATE   /* retrieves/creates anchor object        */
:: METHOD Reference             CLASS  /* retrieve/create surrogate-string for object      */
:: METHOD createReference       CLASS  /* tracking of explicitly created surrogate-strings */
:: METHOD getReference          CLASS  /* tracking of explicitly referred to objects       */

... cut ...
```

... non-public: **i2html.md** ...

```
/* define a class for HTML-list handling (un/numbered and definition lists)     */
:: CLASS html.reference SUBCLASS ref    /* subclass the anchor-manager             */

:: METHOD A_Name        CLASS           /* retrieve a reference name, else create it       */
   USE ARG anObject, text

   RETURN '<A NAME="' || self ~ createReference( anObject, anObject ~ xShortHand ) || '">',
          text  "</A>"

:: METHOD A_HRef        CLASS           /* retrieve a reference name, else create it      */
   USE ARG anObject, text, htmlFile

   IF \ VAR( "htmlFile" ) THEN htmlFile = ""    /* supply default empty string  */
   RETURN '<A HREF="' || htmlFile || "#" || ,
          self ~ getReference( anObject, anObject ~ xShortHand ) || '">' || text || "</A>"

... cut ...

/* gets or sets up an array containing: */

        arr[ 1 ] = tmpLongName    ... MO-xLongName (plain ASCII )
        arr[ 2 ] = tmpAnchorFile  ... HTML-file, which contains anchor
        arr[ 3 ] = tmpDots        ... visual feedback in form of dots (HTML)
        arr[ 4 ] = tmpRefText     ... HTML-surrogate-string for referencing
        arr[ 5 ] = tmpAnchorText  ... HTML-text for anchor, SmartCapped for ME/MR/SA,
                                      normal text for MA
```

... non-public, from **setCheckMM.cmd** ...

```
/* create and initialize the xfer_cdif environment  (a directory)     */
:: ROUTINE initializeXferCDIF   PUBLIC

xfer_cdif = .directory ~ new           /* store results in Xfer_CDIF  */
     /* MetaMetaRelationships        */
     /* -- CMO.IsUsedIn.SA ...     IsUsedIn[ CMO ] = SA               */
xfer_cdif ~ IsUsedIn                 = .relation ~ new   /* CMO.iui.SA   */
xfer_cdif ~ IsLocalMetaAttributeOf   = .relation ~ new   /* MA.ilmao.AMO */
xfer_cdif ~ HasSubtype               = .relation ~ new   /* AMO.hs.AMO   */
xfer_cdif ~ HasSource                = .relation ~ new   /* MR.hs.ME     */
xfer_cdif ~ HasDestination           = .relation ~ new   /* MR.hd.ME     */
    /* lookup relations                    */
               /* ALL NAME2MO-entries! nameRel[ name ] = object */
xfer_cdif ~ nameRel   = .relation  ~ new
    /* index: CDIFMetaIdentifier 4 SA, ME, MR, MA       */
               /* CDIF2MO: cdifRel[ cdifMetaIdentifier ]=object        */
xfer_cdif ~ cdifRel   = .relation  ~ new
    /* MetaMetaEntities (instantiable) stored in sets   */
xfer_cdif ~ SA_set       = .set ~ new   /* SubjectArea                  */
xfer_cdif ~AMO_set       = .set ~ new   /* instance "RootObject" only   */
xfer_cdif ~ ME_set       = .set ~ new   /* MetaEntity                   */
xfer_cdif ~ MR_set       = .set ~ new   /* MetaRelationship             */
xfer_cdif ~ MA_set       = .set ~ new   /* MetaAttribute                */

    /* additional objects needed for making life easier ;-) */
xfer_cdif ~ roots          = .set ~ new /* set of roots (should have 1 entry only) */
xfer_cdif ~ leaves         = .set ~ new /* set of leaves                           */
xfer_cdif ~ IsMandatoryFor    = .relation ~ new  /* IsMandatoryFor[ ME ] = MR      */
xfer_cdif ~ IsMandatoryForSet = .set ~ new /* set of MEs participating in mand. MR */
xfer_cdif ~ MultInherit    = .set ~ new /* set of AMOs with multiple inheritance   */

/* determines the resolving order once and forever ... */
                                        /* points to next super:    */
                     /* ResolveSuper[ AMO ] = .array~of( list, index )    */
xfer_cdif ~ ResolveSuper = .relation ~ new
xfer_cdif ~ MO_missing   = .set ~ new   /* set of missing MOs (ME, MR, SA)  */
xfer_cdif ~ Error_set    = .set ~ new   /* objects with errors              */
xfer_cdif ~ Warning_set  = .set ~ new   /* objects with errors              */
xfer_cdif ~ Mandatory_MA = .set ~ new   /* set with AMOs containing mandatory MAs*/
xfer_cdif ~ SA_object    = .nil /* object to contain SA, this all is defined for        */
xfer_cdif ~ missing.nr   = 0    /* error number for creating unique names and for CMI   */
xfer_cdif ~ duplicate.nr = 0    /* error number for creating duplicate MO   */
RETURN xfer_cdif
```

```
... non-public, from cdif_instantiable.cmd ...
:: CLASS "MetaObject" /* would work: SUBCLASS class  */
:: METHOD       INIT
        self ~ Aliases                  = .nil  /* MMAs         */
        self ~ CDIFMetaIdentifier       = .nil
        self ~ Constraints              = .nil
        self ~ Description              = .nil
        self ~ Name                     = .nil
        self ~ Usage                    = .nil
        self ~ xCdifDir                 = .directory ~ new /* auxiliary    */
        self ~ xErrorSet                = .set ~ new
        self ~ xHasWarning              = .false
        self ~ xHasError                = .false
        self ~ xWarningSet              = .set ~ new
        self ~ xNodeLevel               = .nil
        self ~ xNode2Root               = .nil
        self ~ xUserSlot                = .directory ~ new
:: METHOD       "Aliases"               ATTRIBUTE
:: METHOD       "CDIFMetaIdentifier"    ATTRIBUTE
:: METHOD       "Constraints"           ATTRIBUTE
:: METHOD       "Description"           ATTRIBUTE
:: METHOD       "Usage"                 ATTRIBUTE
:: METHOD       "Name"
   EXPOSE Name
   RETURN Name
:: METHOD       "Name="                  /* set xLongName to Name by default     */
   EXPOSE Name xLongName
   Name      = ARG( 1 )
   IF self ~ xShortHand <> "MR" THEN     /* set xLongName to Name                */
      xLongName = Name
:: METHOD       "xLongName"             ATTRIBUTE
:: METHOD       "xCDIFDir"              ATTRIBUTE       /* directory of attributes with
                                                           CDIF-datatypes */
:: METHOD       "xErrorSet"             ATTRIBUTE       /* set of error-dirs    */
:: METHOD       "xHasError"             ATTRIBUTE       /* boolean, error associated ?, e.g.
                                                           MA-error, set flag for AMO too */
:: METHOD       "xHasWarning"           ATTRIBUTE       /* boolean, error associated ?, e.g.
                                                           MA-error, set flag for AMO too */
:: METHOD       "xWarningSet"           ATTRIBUTE       /* set of warning-dirs   */
:: METHOD       "xShortHand"                            /* return xShortHand string      */
   RETURN       "MO"
:: METHOD       "xNodeLevel"            ATTRIBUTE       /* will contain the level of node*/
:: METHOD       "xNode2Root"            ATTRIBUTE       /* will contain node--->root path*/
:: METHOD       "xUserSlot"             ATTRIBUTE       /* directory for user stuff     */
:: METHOD UNKNOWN
   USE ARG messName, args
   call sayerror "arrived in UNKNOWN instance message of [MetaObject]"
   call sayerror "MetaObject::Unknown:" pp( messName ) "for" pp( self ~ xLongName ),
                  pp( self ~ cdifMetaIdentifier )
   "@pause"
:: METHOD UNKNOWN CLASS
   USE ARG messName, args
   call sayerror "arrived in UNKNOWN **CLASS** message of [MetaObject]"
   call sayerror "MetaObject::Unknown:" pp( messName ) "for" pp( self )
   call sayerror pp( self ~ xShortHand ) pp( self ~ name ) pp( self ~ cdifMetaIdentifier )
   call trace ?i

:: CLASS "SubjectArea"                  SUBCLASS MetaObject     PUBLIC
:: METHOD       INIT            CLASS
   IF self ~ xOrderedMMA ~ class <> .list THEN  /* not initialised as of yet ?  */
   DO
      self ~ xOrderedMMA = .list ~ of( "Name" , "CDIFMetaIdentifier" , "Description" ,,
                                 "Usage" , "Aliases" , "Constraints" , "VersionNumber" )
      self ~ xMand_MMA_set = .set  ~ of( "Name",  "CDIFMetaIdentifier", "Description",,
                                 "VersionNumber" )
   END
   FORWARD CLASS ( super )                      /* make sure superclass initializes too */
:: METHOD       "xOrderedMMA"    CLASS   ATTRIBUTE       /* determines order for dumping MAs */
:: METHOD       "xMand_MMA_set"  CLASS   ATTRIBUTE       /* determines non-optional MAs      */
:: METHOD       INIT
        self ~ "VersionNumber"          = .nil
        FORWARD CLASS ( super )         /* make sure superclass initializes too */
:: METHOD       "VersionNumber"         ATTRIBUTE
:: METHOD       "xShortHand"                            /* return xShortHand string      */
   RETURN       "SA"
```

... cut from **fnd.4.overview.html** ...

```
<HR><!-- empty -->
<H2>AttributableMetaObject Hierarchy</H2>

<!-- Begin of [OL type=1 compact] list # [5]... -->
 <OL type=1 compact>
 <!-- list entry # 1 -->
 <LI><A HREF="fnd.5.detailed.html#AMO_1">RootObject</A>
     <IMG SRC="det_RGF.gif" WIDTH=14 HEIGHT=14 ALIGN=TOP>  <br>

   <!-- Begin of [OL type=1 compact] list # [6]... -->
   <OL type=1 compact>
   <!-- list entry # 1 -->
     <LI><A HREF="fnd.5.detailed.html#ME_1">RootEntity</A>
     <IMG SRC="det_RGF.gif" WIDTH=14 HEIGHT=14 ALIGN=TOP> </LI>

   <!-- list entry # 2 -->
     <LI><A HREF="fnd.5.detailed.html#ME_1">RootEntity</A>.<em><A
     HREF="fnd.5.detailed.html#MR_1">IsRelatedTo</A></em>.<A
     HREF="fnd.5.detailed.html#ME_1">RootEntity</A>
     <IMG SRC="det_RGF.gif" WIDTH=14 HEIGHT=14 ALIGN=TOP> </LI>
   </OL>
   <!-- end of [OL type=1 compact] list # [6]. -->
 </LI>
</OL>
<!-- end of [OL type=1 compact] list # [5]. -->

<HR SIZE=10><!-- empty -->
```