```
'Design_Patterns_Demo:

Option Declare

%INCLUDE "lsconst.lss"


'/**************************************************************************
' * <b>Abstract class</b>: represents an entry in a directory.
' *
' * @author             Thomas Bahn <tbahn@assono.de>
' * @version            2007-10-07
' **************************************************************************/
Public Class DirectoryEntry

        '/**************************************************************************
        ' * Name of the entry.
        ' **************************************************************************/
        Private entryName As String

        '/**************************************************************************
        ' * Getter for the entryName attribute.
        ' *
        ' * @return                          current value of the atrribute.
        ' *
        ' * @author             Thomas Bahn <tbahn@assono.de>
        ' * @version            2007-10-05
        ' **************************************************************************/
        Public Function GetEntryName As String
                GetEntryName = Me.entryName
        End Function ' DirectoryEntry.GetEntryName


        '/**************************************************************************
        ' * Timestamp of the last modification to the entry.
        ' **************************************************************************/
        Private lastModified As Variant

        '/**************************************************************************
        ' * Getter for the lastModified attribute.
        ' *
        ' * @return                          current value of the atrribute.
        ' *
        ' * @author             Thomas Bahn <tbahn@assono.de>
        ' * @version            2007-10-05
        ' **************************************************************************/
        Public Function GetLastModified As Variant
                GetLastModified = Me.lastModified
        End Function ' DirectoryEntry.GetLastModified
```

```
'/*************************************************************************
' * directory entry is hidden.
' *************************************************************************/
Private hidden As Boolean

'/*************************************************************************
' * Getter for the hidden attribute.
' *
' * @return                          current value of the atrribute.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-06
' *************************************************************************/
Public Function IsHidden As Boolean
        IsHidden = Me.hidden
End Function ' DirectoryEntry.IsHidden


'/*************************************************************************
' * parent directory.
' *************************************************************************/
Private parent As Directory

'/*************************************************************************
' * Getter for the parent attribute.
' *
' * @return                          current value of the atrribute.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-06
' *************************************************************************/
Public Function GetParent As Directory
        Set GetParent = Me.parent
End Function ' DirectoryEntry.GetParent



'/*************************************************************************
' * Constructor - sets all read-only attributes.
' *
' * @param           entryName Name of the entry.
' * @param           lastModified Timestamp of the last modification to the entry.
' * @param           hidden directory entry is hidden.
' * @param           parent parent directory.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-07
```

```
' ****************************************************************************/
Public Sub New(entryName As String, lastModified As Variant, hidden As Boolean, parent As Directory)
	If Typename(Me) = "DIRECTORYENTRY" Then
		Error 32001, |Abstract class: you cannot create objects with this class ("| & Typename(Me) & |") directly, only with concrete subclasses!|
	End If

	Me.EntryName = entryName
	Me.LastModified = lastModified
	Me.Hidden = hidden
	Set Me.Parent = parent
End Sub ' DirectoryEntry.New



'/*****************************************************************************
' * returns a string representation of this object.
' *
' * @return			string representation of the current object.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' ****************************************************************************/
Public Function ToString() As String
	Dim result As String

	result = Typename(Me) & |[|
	result = result & |entryName: "| & Me.entryName & |"|
	If Isnull(lastModified) Then
		result = result & |, lastModified: unknown|
	Else
		result = result & |, lastModified: | & Me.lastModified & ||
	End If
	result = result & |, hidden: | & Cstr(Me.hidden)
	If Me.parent Is Nothing Then
		result = result & |, parent: unknown|
	Else
		result = result & |, parent: | & Typename(Me.parent) & |[location: " | & Me.parent.GetLocation() & |", ...]|
	End If
	ToString = result & |]|
End Function ' DirectoryEntry.ToString


End Class ' DirectoryEntry



'/*****************************************************************************
```

```
' * represents an folder's entry in a directory.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' ************************************************************************/
Public Class FolderEntry As DirectoryEntry

	'/************************************************************************
	' * directory of this FolderEntry.
	' ************************************************************************/
	Private directory As Directory

	'/************************************************************************
	' * Getter for the parent attribute.
	' *
	' * @return			current value of the atrribute.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-07
	' ************************************************************************/
	Public Function GetDirectory As Directory
		Set GetDirectory = Me.directory
	End Function ' FolderEntry.GetDirectory


	'/************************************************************************
	' * Constructor - sets all read-only attributes.
	' *
	' * @param		EntryName Name of the entry.
	' * @param		LastModified Timestamp of the last modification to the entry.
	' * @param		Hidden directory entry is hidden.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' ************************************************************************/
	Public Sub New(entryName As String, lastModified As Variant, hidden As Boolean, parent As Directory), DirectoryEntry(entryName, lastModified, hidden, parent)
		Set directory = New DirectoryProxy(parent.GetLocation & "/" & entryName, parent)
	End Sub ' FolderEntry.New


End Class ' FolderEntry




'/************************************************************************
' * represents an file's entry in a directory.
' *
```

```
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' ****************************************************************************/
Public Class FileEntry As DirectoryEntry

	'/*****************************************************************************
	' * Size of the entry in bytes or empty, if not applicable,
	' *****************************************************************************/
	Private Size As Variant

	'/*****************************************************************************
	' * Getter for the Size attribute.
	' *
	' * @return			current value of the atrribute.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-05
	' *****************************************************************************/
	Public Function GetSize As Variant
		GetSize = Me.Size
	End Function ' FileEntry.GetSize



	'/*****************************************************************************
	' * Constructor - sets all read-only attributes.
	' *
	' * @param		EntryName Name of the entry.
	' * @param		LastModified Timestamp of the last modification to the entry.
	' * @param		Hidden directory entry is hidden.
	' * @param		Size Size of the entry in bytes or empty, if not applicable.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' *****************************************************************************/
	Public Sub New(EntryName As String, LastModified As Variant, Hidden As Boolean, Parent As Directory, Size As Long), DirectoryEntry(EntryName, LastModified, Hidden, Parent)
		Me.Size = Size
	End Sub ' FileEntry.New


	'/*****************************************************************************
	' * Returns a string representation of this object.
	' *
	' * @return			string representation of the current object.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' *****************************************************************************/
	Public Function ToString() As String
```

```
                Dim result As String

                result = DirectoryEntry..ToString()
                result = Left(result, Len(result) - 1) ' chop off the closing ], which is always the last character
                result = result & |, Size: | & Me.size & | Bytes|
                ToString = result & |]|
        End Function ' FileEntry.ToString


End Class ' FileEntry




'/**************************************************************************
' * represents a directory in a file system.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-05
' **************************************************************************
Public Class Directory

        '/**************************************************************************
        ' * Array of the entries of this directory.
        ' **************************************************************************/
        Private entries() As DirectoryEntry

        '/**************************************************************************
        ' * Clears the array of entries.
        ' *
        ' * @author          Thomas Bahn <tbahn@assono.de>
        ' * @version         2007-10-06
        ' **************************************************************************/
        Private Sub ClearEntries()
                Redim Me.entries(0 To 0)
                Set Me.entries(0) = Nothing ' empty array has Nothing as first and only entry
        End Sub ' Directory.ClearEntries

        '/**************************************************************************
        ' * appends a new DirectoryEntry to the array.
        ' *
        ' * @param           newEntry DirectoryEntry to be appended.
        ' *
        ' * @author          Thomas Bahn <tbahn@assono.de>
        ' * @version         2007-10-06
        ' **************************************************************************/
        Public Sub AppendEntry(newEntry As DirectoryEntry)
                If Not (Me.entries(0) Is Nothing) Then ' expand array, if necessary
```

```
                Redim Preserve Me.entries(Lbound(Me.entries) To Ubound(Me.entries) + 1)
        End If
        Set Me.entries(Ubound(Me.entries)) = newEntry
End Sub ' Directory.AppendEntry

'/**************************************************************************
' * returns DirectoryEntry at given array index.
' *
' * @param       index index in entries array.
' * @return              DirectoryEntry at given index or nothing, if index is out of range.
' *
' * @author      Thomas Bahn <tbahn@assono.de>
' * @version     2007-10-06
' **************************************************************************/
Public Function GetEntry(index As Integer) As DirectoryEntry
        Set GetEntry = Nothing

        If Lbound(Me.entries) <= index And index <= Ubound(Me.entries) Then
                Set GetEntry = Me.entries(index)
        End If
End Function ' Directory.GetEntry


'/**************************************************************************
' * Location of the directory.
' **************************************************************************/
Private location As String

'/**************************************************************************
' * Getter for the location attribute.
' *
' * @return              current value of the atrribute.
' *
' * @author      Thomas Bahn <tbahn@assono.de>
' * @version     2007-10-06
' **************************************************************************/
Public Function GetLocation As String
        GetLocation = Me.location
End Function ' Directory.GetLocation


'/**************************************************************************
' * parent directory.
' *
' * Parent points to current directory object, if this directory is the root directory.
' * Parent is Nothing, when the parent is still unknown.
' **************************************************************************/
Private parent As Directory
```

```
'/**************************************************************************
' * Getter for the parent attribute.
' *
' * @return                          current value of the atrribute.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-06
' **************************************************************************/
Public Function GetParent As Directory
        Set GetParent = Me.parent
End Function ' Directory.GetParent


'/**************************************************************************
' * Setter for the parent attribute.
' *
' * @param           new value of the atrribute.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-06
' **************************************************************************/
Public Sub SetParent(parent As Directory)
        Set Me.parent = parent
End Sub ' Directory.SetParent



'/**************************************************************************
' * DirectoryReader to be used to read/refresh this directory.
' *
' * Default: LocalDirectoryReader
' **************************************************************************/
Private directoryReader As DirectoryReader


'/**************************************************************************
' * Setter for the directoryReader attribute.
' *
' * @param           new value of the atrribute.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-06
' **************************************************************************/
Public Sub SetDirectoryReader(directoryReader As DirectoryReader)
        Set Me.directoryReader = directoryReader
End Sub ' Directory.SetDirectoryReader
```

```
'/**************************************************************************
' * Array of registered observers for this object.
' **************************************************************************/
Private observers() As DirectoryObserver


'/**************************************************************************
' * clears the observers array.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-07
' **************************************************************************/
Private Sub ClearObservers()
        Redim Me.observers(0 To 0)
        Set Me.observers(0) = Nothing ' empty array has Nothing as first and only entry
End Sub ' Directory.ClearObservers


'/**************************************************************************
' * adds an observer to the array
' *
' * @param           observer new observer to be registered.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-07
' **************************************************************************/
Public Sub AddObserver(observer As DirectoryObserver)
        If Not(Me.observers(0) Is Nothing) Then ' expand array, if necessary
                Redim Preserve Me.observers(Lbound(Me.observers) To Ubound(Me.observers) + 1)
        End If
        Set Me.observers(Ubound(Me.observers)) = observer
End Sub ' Directory.AddObserver


'/**************************************************************************
' * removes an observer from the array
' *
' * @param           observer observer to be removed.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-07
' **************************************************************************/
Public Sub RemoveObserver(observer As DirectoryObserver)
        Dim index As Integer
        Dim observerIndex As Integer

        ' search the observer, which should be removed
        observerIndex = -1
```

```
		For index = Lbound(Me.observers) To Ubound(Me.observers)
			If Me.observers(index) Is observer Then
				observerIndex = index
				Exit For
			End If
		Next

		If observerIndex > -1 Then ' if observer was found
			For index = observerIndex To (Ubound(Me.observers) - 1) ' move all entries above this array entry on position down
				Set Me.observers(index) = Me.observers(index + 1)
			Next
			Set Me.observers(Ubound(Me.observers)) = Nothing ' clear topmost array entry
			If Ubound(Me.observers) > Lbound(Me.observers) Then ' shorten array, if no emtpy
				Redim Preserve Me.observers(Lbound(Me.observers) To Ubound(Me.observers) - 1)
			End If
		End If
End Sub ' Directory.RemoveObserver


'/**************************************************************************
' * notifies the registered observers
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' **************************************************************************/
Public Sub NotifyObservers
		Dim index As Integer

		If Me.observers(Lbound(Me.observers)) Is Nothing Then Exit Sub ' emtpy array --> nothing to do

		For index = Lbound(Me.observers) To Ubound(Me.observers)
			Call Me.observers(index).Update()
		Next
End Sub ' Directory.NotifyObservers


'/**************************************************************************
' * marks current directory as modified and notifies the registered iterators.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' **************************************************************************/
Public Sub MarkModified()
		Call NotifyObservers()
		Call ReadDirectory(False) ' refresh content
End Sub ' Directory.MarkModified
```

```
'/*************************************************************************
' * Constructor - initializes attributes and reads directory if instructed to.
' *
' * @param		location location of the directory.
' * @param		readNow read directory now
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' *************************************************************************/
Public Sub New(location As String, readNow As Boolean)
	Me.location = location
	Set Me.parent = Nothing
	Set Me.directoryReader = Nothing

	Call ClearObservers()

	If readNow Then Call ReadDirectory(False)
End Sub ' Directory.New



'/*************************************************************************
' * Returns a string representation of this object.
' *
' * @return			string representation of the current object.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-05
' *************************************************************************/
Public Function ToString() As String
	Dim result As String
	Dim i As Integer

	result = Typename(Me) & |[|
	result = result & |location: "| & Me.location & |"|
	If Me.parent Is Nothing Then
		result = result & |, parent: unknown|
	Else
		result = result & |, parent: | & Typename(Me.parent) & |[location: " | & Me.parent.GetLocation() & |", ...]|
	End If
	If entries(0) Is Nothing Then
		result = result & |, no directory entries|
	Else
		result = result & |, | & (Ubound(entries) + 1) & | directory entries:|
		For i = Lbound(Me.entries) To Ubound(Me.entries)
			result = result & Chr$(10) & "  Entry " & i & |: | & Me.entries(i).ToString()
		Next
		result = result & Chr$(10)
	End If
```

```
		ToString = result & |]|
	End Function ' Directory.ToString


	'/**************************************************************************
	' * Reads (or refreshes) the directory and stores the current entries.
	' *
	' * @param		withModificationMark if true, MarkModification() is called
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-07
	' **************************************************************************/
	Public Sub ReadDirectory(withModificationMark As Boolean)
		Dim currentDirectoryReader As DirectoryReader

		Call ClearEntries()

		If Me.directoryReader Is Nothing Then ' if Me.directoryReader is not set, use a LocalDirectoryReader
			Set currentDirectoryReader = New LocalDirectoryReader()
		Else
			Set currentDirectoryReader = Me.directoryReader
		End If

		Call currentDirectoryReader.ReadDirectory(Me)

		If withModificationMark Then Call MarkModified() ' just to be sure, since we cannot "know", if it is really modified
	End Sub ' Directory.ReadDirectory


End Class ' Directory




'/**************************************************************************
' * proxy of a real Directory object, which is created, when needed, but not before.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' **************************************************************************/
Public Class DirectoryProxy As Directory


	'/**************************************************************************
	' * Directory object, this proxy stands for.
	' **************************************************************************/
```

```
Private directory As Directory

'/**************************************************************************
' * creates Directory object, if it is not already created.
' *
' * @author              Thomas Bahn <tbahn@assono.de>
' * @version             2007-10-07
' **************************************************************************/
Private Sub CreateDirectory()
        If Me.directory Is Nothing Then ' do it only, when Directory object is not created yet
                Set directory = New Directory(Me.location, True)
                Call directory.SetParent(parent)
        End If
End Sub



'/**************************************************************************
' * Clears the array of entries.
' *
' * @author              Thomas Bahn <tbahn@assono.de>
' * @version             2007-10-07
' **************************************************************************/
Private Sub ClearEntries()
        Call CreateDirectory
        Call directory.ClearEntries
End Sub ' DirectoryProxy.ClearEntries


'/**************************************************************************
' * appends a new DirectoryEntry to the array.
' *
' * @param               newEntry DirectoryEntry to be appended.
' *
' * @author              Thomas Bahn <tbahn@assono.de>
' * @version             2007-10-07
' **************************************************************************/
Public Sub AppendEntry(newEntry As DirectoryEntry)
        Call CreateDirectory
        Call directory.AppendEntry(newEntry)
End Sub ' DirectoryProxy.AppendEntry


'/**************************************************************************
' * returns DirectoryEntry at given array index.
' *
' * @param               index index in entries array.
' * @return                      DirectoryEntry at given index or nothing, if index is out of range.
' *
```

```
' * @author             Thomas Bahn <tbahn@assono.de>
' * @version            2007-10-07
' ***************************************************************************/
Public Function GetEntry(index As Integer) As DirectoryEntry
        Call CreateDirectory
        Set GetEntry = directory.GetEntry(index)
End Function ' DirectoryProxy.GetEntry


'/****************************************************************************
' * Getter for the location attribute.
' *
' * @return                     current value of the atrribute.
' *
' * @author             Thomas Bahn <tbahn@assono.de>
' * @version            2007-10-07
' ***************************************************************************/
Public Function GetLocation As String
        GetLocation = Me.location
End Function ' DirectoryProxy.GetLocation


'/****************************************************************************
' * Getter for the parent attribute.
' *
' * @return                     current value of the atrribute.
' *
' * @author             Thomas Bahn <tbahn@assono.de>
' * @version            2007-10-07
' ***************************************************************************/
Public Function GetParent As Directory
        Set GetParent = Me.parent
End Function ' DirectoryProxy.GetParent


'/****************************************************************************
' * Setter for the parent attribute.
' *
' * @param              new value of the atrribute.
' *
' * @author             Thomas Bahn <tbahn@assono.de>
' * @version            2007-10-07
' ***************************************************************************/
Public Sub SetParent(parent As Directory)
        Call CreateDirectory
        Set Me.parent = parent
        Call directory.SetParent(parent)
End Sub ' DirectoryProxy.SetParent
```

```
'/**************************************************************************
' * Setter for the directoryReader attribute.
' *
' * @param           new value of the atrribute.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-07
' **************************************************************************/
Public Sub SetDirectoryReader(directoryReader As DirectoryReader)
        Call CreateDirectory
        Call directory.SetDirectoryReader(directoryReader)
End Sub ' DirectoryProxy.SetDirectoryReader


'/**************************************************************************
' * adds an observer to the array
' *
' * @param           observer new observer to be registered.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-07
' **************************************************************************/
Public Sub AddObserver(observer As DirectoryObserver)
        Call CreateDirectory
        Call directory.AddObserver(observer)
End Sub ' DirectoryProxy.AddObserver


'/**************************************************************************
' * removes an observer from the array
' *
' * @param           observer observer to be removed.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-07
' **************************************************************************/
Public Sub RemoveObserver(observer As DirectoryObserver)
        Call CreateDirectory
        Call directory.RemoveObserver(observer)
End Sub ' DirectoryProxy.RemoveObserver


'/**************************************************************************
' * notifies the registered observers
' *
' * @author          Thomas Bahn <tbahn@assono.de>
```

```
' * @version           2007-10-07
' ************************************************************************************/
Public Sub NotifyObservers
        Call CreateDirectory
        Call directory.NotifyObservers()
End Sub ' DirectoryProxy.NotifyObservers


'/************************************************************************************
' * marks current directory as modified and notifies the registered iterators.
' *
' * @author            Thomas Bahn <tbahn@assono.de>
' * @version           2007-10-07
' ************************************************************************************/
Public Sub MarkModified()
        Call CreateDirectory
        Call directory.MarkModified()
End Sub ' DirectoryProxy.MarkModified



'/************************************************************************************
' * Constructor - initializes attributes and reads directory if instructed to.
' *
' * @param             location location of the directory.
' *
' * @author            Thomas Bahn <tbahn@assono.de>
' * @version           2007-10-07
' ************************************************************************************/
Public Sub New(location As String, parent As Directory), Directory(location, False)
        Me.location = location
        Set Me.parent = parent
        Set Me.directory = Nothing ' nothing for now
End Sub ' DirectoryProxy.New



'/************************************************************************************
' * Returns a string representation of this object.
' *
' * @return                      string representation of the current object.
' *
' * @author            Thomas Bahn <tbahn@assono.de>
' * @version           2007-10-07
' ************************************************************************************/
Public Function ToString() As String
        Call CreateDirectory
        ToString = directory.ToString()
End Function ' DirectoryProxy.ToString
```

```
		'/*****************************************************************************
		' * Reads (or refreshes) the directory and stores the current entries.
		' *
		' * @param		withModificationMark if true, MarkModification() is called
		' *
		' * @author		Thomas Bahn <tbahn@assono.de>
		' * @version		2007-10-07
		' *****************************************************************************/
		Public Sub ReadDirectory(withModificationMark As Boolean)
			Call CreateDirectory
			Call directory.ReadDirectory(withModificationMark)
		End Sub ' DirectoryProxy.ReadDirectory


End Class ' DirectoryProxy




'/*****************************************************************************
' * <b>Abstract class</b>: reads an directory from its source.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' *****************************************************************************
Public Class DirectoryReader

		'/*****************************************************************************
		' * Constructor - initializes attributes.
		' *
		' * @author		Thomas Bahn <tbahn@assono.de>
		' * @version		2007-10-07
		' *****************************************************************************/
		Public Sub New()
			If Typename(Me) = "DIRECTORYREADER" Then
				Error 32001, |Abstract class: you cannot create objects with this class ("| & Typename(Me) & |") directly, only with concrete subclasses!|
			End If
		End Sub ' DirectoryReader.New



		'/*****************************************************************************
		' * <b>Abstract method</b>: reads an directory from its source.
		' *
		' * @param		currentDirectory directory to be read.
```

```
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' ***************************************************************************/
	Public Sub ReadDirectory(currentDirectory As Directory)
		Error 32000, "Abstract method; has to be implemented in a subclass!"
	End Sub ' DirectoryReader.ReadDirectory

End Class ' DirectoryReader



'/***************************************************************************
' * reads an directory from the local file system.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' ***************************************************************************/
Public Class LocalDirectoryReader As DirectoryReader

	'/***************************************************************************
	' * reads an directory from the local file system.
	' *
	' * @param		currentDirectory directory to be read.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' ***************************************************************************/
	Public Sub ReadDirectory(currentDirectory As Directory)
		Dim entryName As String
		Dim newEntry As DirectoryEntry

		entryName = Dir$(currentDirectory.GetLocation(), ATTR_NORMAL + ATTR_HIDDEN + ATTR_SYSTEM + ATTR_DIRECTORY)
		Do While entryName <> ""
			Set newEntry = CreateDirectoryEntry(currentDirectory, entryName)
			Call currentDirectory.AppendEntry(newEntry)
			entryName = Dir$()
		Loop
	End Sub ' LocalDirectoryReader.ReadDirectory


	'/***************************************************************************
	' * creates a new DirectoryEntry.
	' *
	' * @param		currentDirectory directory, to which new entry belongs to.
	' * @param		entryName name of the entry.
	' * @return			new DirectoryEntry.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
```

```
	' * @version		2007-10-06
	' ********************************************************************************/
	Private Function CreateDirectoryEntry(currentDirectory As Directory, entryName As String) As DirectoryEntry
		Dim newEntry As DirectoryEntry
		Dim entryLocation As String
		Dim lastModifiedText As String
		Dim lastModified As Variant
		Dim attributes As Integer
		Dim hidden As Boolean
		Dim folder As Boolean
		Dim size As Long

		On Error 70 Resume Next ' Error #70: Permission denied (thrown by FileDateTime())
		On Error 53 Resume Next ' Error #53: File not found (which is thrown by GetFileAttr() and FileLen(), when in truth access permission was denied!)

		entryLocation = currentDirectory.GetLocation() & "/" & entryName ' (full) location of the entry: path, directory separator and file name
		lastModifiedText = "" ' default if FileDateTime fails (e. g. due to access restrictions)
		lastModified = Null ' Null means unknown; default if FileDateTime fails
		lastModifiedText = Filedatetime(entryLocation)
		If lastModifiedText <> "" Then
			lastModified = Cdat(lastModifiedText)
		End If
		attributes = ATTR_NORMAL ' default if GetFileAttr fails (e. g. due to access restrictions)
		attributes = Getfileattr(entryLocation)
		hidden = (attributes And ATTR_HIDDEN)
		folder = (attributes And ATTR_DIRECTORY)

		If folder Then
			Set CreateDirectoryEntry = New FolderEntry(entryName, lastModified, hidden, currentDirectory)
		Else
			size = 0 ' default if FileLen fails (e. g. due to access restrictions)
			size = Filelen(entryLocation)
			Set CreateDirectoryEntry = New FileEntry(entryName, lastModified, hidden, currentDirectory, size)
		End If

		On Error Goto 0
	End Function ' LocalDirectoryReader.CreateDirectoryEntry

End Class ' LocalDirectoryReader



/********************************************************************************
' * reads an directory from a remote file system.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' ********************************************************************************/
Public Class RemoteDirectoryReader As DirectoryReader
```

```
	'/*************************************************************************
	' * reads an directory from the local file system.
	' *
	' * @param		currentDirectory directory to be read.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' *************************************************************************/
	Public Sub ReadDirectory(currentDirectory As Directory)
		Error 32002, "Method not implemented!"

		' A remote directory could be read by e. g. FTP, HTTP, a Web service or any other means.
		' I would propose a specific subclass for each protocol implemented: FTPRemoteDirectoryReader, ....
	End Sub ' RemoteDirectoryReader.ReadDirectory

End Class ' RemoteDirectoryReader




'/*************************************************************************
' * <b>Abstract class</b>: Observer of a Directory or DirectoryIterator.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' *************************************************************************
Public Class DirectoryObserver

	'/*************************************************************************
	' * Constructor - initializes attributes.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-07
	' *************************************************************************/
	Public Sub New()
		If Typename(Me) = "DIRECTORYOBSERVER" Then
			Error 32001, |Abstract class: you cannot create objects with this class ("| & Typename(Me) & |") directly, only with concrete subclasses!|
		End If
	End Sub ' DirectoryObserver.New



	'/*************************************************************************
	' * <b>Abstract method</b>: is called by the Observant (Subject) in case of a change.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-07
```

```
	' ************************************************************************************/
	Public Sub Update()
		Error 32000, "Abstract method; has to be implemented in a subclass!"
	End Sub ' DirectoryObserver.Update


End Class ' DirectoryObserver




'/**************************************************************************************
' * iterates through a Directory.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' **************************************************************************************
Public Class DirectoryIterator As DirectoryObserver

	'/**************************************************************************************
	' * Directory to be iterated through.
	' **************************************************************************************/
	Private currentDirectory As Directory

	'/**************************************************************************************
	' * Getter for the currentDirectory attribute.
	' *
	' * @return				current value of the atrribute.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' **************************************************************************************/
	Public Function GetCurrentDirectory As Directory
		Set GetCurrentDirectory = Me.currentDirectory
	End Function ' DirectoryEntry.GetCurrentDirectory


	'/**************************************************************************************
	' * Index of the current entry in the entries array of the directory.
	' **************************************************************************************/
	Private currentEntryIndex As Integer

	'/**************************************************************************************
	' * Getter for the currentEntryIndex attribute.
	' *
	' * @return				current value of the atrribute.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
```

```
' * @version           2007-10-06
' ************************************************************************************/
Public Function GetCurrentEntryIndex As Integer
        GetCurrentEntryIndex = Me.currentEntryIndex
End Function ' DirectoryIterator.GetCurrentEntryIndex



'/************************************************************************************
' * DirectoryIterator is valid only until base directory/iterator has changed.
' ************************************************************************************/
Private Valid As Boolean

'/************************************************************************************
' * is called by the Observant (Subject) in case of a change and invalidates current object.
' *
' * @author            Thomas Bahn <tbahn@assono.de>
' * @version           2007-10-07
' ************************************************************************************/
Public Sub Update()
        Me.Valid = False ' this iterator is now invalid
        Call NotifyObservers() ' notify dependend iterators
End Sub ' DirectoryIterator.Update



'/************************************************************************************
' * Array of registered observers for this object.
' ************************************************************************************/
Private observers() As DirectoryObserver


'/************************************************************************************
' * clears the observers array.
' *
' * @author            Thomas Bahn <tbahn@assono.de>
' * @version           2007-10-07
' ************************************************************************************/
Private Sub ClearObservers()
        Redim Me.observers(0 To 0)
        Set Me.observers(0) = Nothing ' empty array has Nothing as first and only entry
End Sub ' Directory.ClearObservers


'/************************************************************************************
' * adds an observer to the array
' *
' * @param             observer new observer to be registered.
' *
```

```
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' ************************************************************************************/
Public Sub AddObserver(observer As DirectoryObserver)
	If Not Me.Valid Then Error 32003, "DirectoryIterator is no longer valid"

	If Not(Me.observers(0) Is Nothing) Then ' expand array, if necessary
		Redim Preserve Me.observers(Lbound(Me.observers) To Ubound(Me.observers) + 1)
	End If
	Set Me.observers(Ubound(Me.observers)) = observer
End Sub ' Directory.AddObserver


'/************************************************************************************
' * removes an observer from the array
' *
' * @param		observer observer to be removed.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' ************************************************************************************/
Public Sub RemoveObserver(observer As DirectoryObserver)
	Dim index As Integer
	Dim observerIndex As Integer

	' search the observer, which should be removed
	observerIndex = -1
	For index = Lbound(Me.observers) To Ubound(Me.observers)
		If Me.observers(index) Is observer Then
			observerIndex = index
			Exit For
		End If
	Next

	If observerIndex > -1 Then ' if observer was found
		For index = observerIndex To (Ubound(Me.observers) - 1) ' move all entries above this array entry on position down
			Set Me.observers(index) = Me.observers(index + 1)
		Next
		Set Me.observers(Ubound(Me.observers)) = Nothing ' clear topmost array entry
		If Ubound(Me.observers) > Lbound(Me.observers) Then ' shorten array, if no emtpy
			Redim Preserve Me.observers(Lbound(Me.observers) To Ubound(Me.observers) - 1)
		End If
	End If
End Sub ' Directory.RemoveObserver


'/************************************************************************************
' * notifies the registered observers
' *
```

```
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' ************************************************************************************/
Public Sub NotifyObservers
	Dim index As Integer

	If Me.observers(Lbound(Me.observers)) Is Nothing Then Exit Sub ' emtpy array --> nothing to do

	For index = Lbound(Me.observers) To Ubound(Me.observers)
		Call Me.observers(index).Update()
	Next
End Sub ' Directory.NotifyObservers



'/************************************************************************************
' * Constructor - initializes attributes.
' *
' * @param		currentDirectory Directory to be iterated through.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' ************************************************************************************/
Public Sub New(currentDirectory As Directory)
	Set Me.currentDirectory = currentDirectory
	Me.currentEntryIndex = -1
	Me.Valid = True

	Call ClearObservers()
	Call currentDirectory.AddObserver(Me)
End Sub ' DirectoryIterator.New



'/************************************************************************************
' * returns the current DirectoryEntry of the Directory.
' *
' * @return			current DirectoryEntry of the Directory or Nothing, if there is no current entry (e. g. before first call to GetFirstEntry).
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' ************************************************************************************/
Public Function GetCurrentEntry() As DirectoryEntry
	If Not Me.Valid Then Error 32003, "DirectoryIterator is no longer valid"

	If Me.currentEntryIndex = -1 Then ' no current entry
		Set GetCurrentEntry = Nothing
	Else
		Set GetCurrentEntry = currentDirectory.GetEntry(Me.currentEntryIndex)
```

```
                End If
        End Function ' DirectoryIterator.GetCurrentEntry

        '/**************************************************************************
        ' * returns first DirectoryEntry of the Directory.
        ' *
        ' * @return                          first DirectoryEntry of the Directory or Nothing, if there is no entry in the directory.
        ' *
        ' * @author          Thomas Bahn <tbahn@assono.de>
        ' * @version         2007-10-06
        ' **************************************************************************/
        Public Function GetFirstEntry() As DirectoryEntry
                Me.currentEntryIndex = 0
                Me.Valid = True ' starting from the beginning makes iterator valid again
                Set GetFirstEntry = GetCurrentEntry()
        End Function ' DirectoryIterator.GetFirstEntry

        '/**************************************************************************
        ' * returns next DirectoryEntry of the Directory.
        ' *
        ' * @return                          next DirectoryEntry of the Directory or Nothing, if there is next entry in the directory.
        ' *
        ' * @author          Thomas Bahn <tbahn@assono.de>
        ' * @version         2007-10-06
        ' **************************************************************************/
        Public Function GetNextEntry() As DirectoryEntry
                If Not Me.Valid Then Error 32003, "DirectoryIterator is no longer valid"

                Me.currentEntryIndex = Me.currentEntryIndex + 1
                Set GetNextEntry = GetCurrentEntry()
        End Function ' DirectoryIterator.GetNextEntry

End Class ' DirectoryIterator



'/**************************************************************************
' * <b>Abstract class</b>: iterates through all DirectoryEntries, which fulfill the filter condition.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-07
' **************************************************************************/
Public Class FilteredDirectoryIterator As DirectoryIterator

        '/**************************************************************************
        ' * base DirectoryIterator to iterate through.
        ' **************************************************************************/
        Private baseIterator As DirectoryIterator
```

```
'/**************************************************************************
' * current DirectoryEntry
' **************************************************************************/
Private currentEntry As DirectoryEntry



'/**************************************************************************
' * Constructor - initializes attributes.
' *
' * @param           baseIterator base DirectoryIterator to iterate through.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-07
' **************************************************************************/
Public Sub New(baseIterator As DirectoryIterator), DirectoryIterator(baseIterator.GetCurrentDirectory())
        If Typename(Me) = "FILTEREDDIRECTORYITERATOR" Then
                Error 32001, |Abstract class: you cannot create objects with this class ("| & Typename(Me) & |") directly, only with concrete subclasses!|
        End If

        Set Me.baseIterator = baseIterator
        Set currentEntry = Nothing
End Sub ' FilteredDirectoryIterator.New



'/**************************************************************************
' * <b>Abstract method</b>: returns if current directory entry fulfills filter condition.
' *
' * @return                  true, if and only if current directory entry fulfills filter condition.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-06
' **************************************************************************/
Private Function CurrentEntryIsValid() As Boolean
        Error 32000, "Abstract method; has to be implemented in a subclass!"
End Function ' FilteredDirectoryIterator.CurrentEntryIsValid



'/**************************************************************************
' * returns the current DirectoryEntry of the Directory.
' *
' * @return                  current DirectoryEntry of the Directory or Nothing, if there is no current entry (e. g. before first call to GetFirstEntry).
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-06
' **************************************************************************/
Public Function GetCurrentEntry() As DirectoryEntry
```

```
	If Not Me.Valid Then Error 32003, "DirectoryIterator is no longer valid"

	Set GetCurrentEntry = currentEntry
End Function ' FilteredDirectoryIterator.GetCurrentEntry


'/**************************************************************************
' * returns next valid DirectoryEntry of the Directory starting from Me.currentEntry.
' *
' * @return			next DirectoryEntry of the Directory or Nothing, if there is no next valid entry in the directory.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' **************************************************************************/
Private Function GetNextValidEntry() As DirectoryEntry
	Do Until Me.currentEntry Is Nothing ' not after the end yet
		If CurrentEntryIsValid() Then
			Me.currentEntryIndex = Me.baseIterator.GetCurrentEntryIndex()
			Set GetNextValidEntry = Me.currentEntry
			Exit Function
		End If
		Set Me.currentEntry = Me.baseIterator.GetNextEntry() ' current entry is hidden, try next
	Loop

	' since we got here, currentEntry Is Nothing
	Me.currentEntryIndex = Me.baseIterator.GetCurrentEntryIndex()
	Set GetNextValidEntry = Me.currentEntry
End Function ' FilteredDirectoryIterator.GetNextValidEntry


'/**************************************************************************
' * returns first valid DirectoryEntry of the Directory, that is the first entry, which fulfills the filter condition.
' *
' * @return			first DirectoryEntry of the Directory or Nothing, if there is no valid entry in the directory.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' **************************************************************************/
Public Function GetFirstEntry() As DirectoryEntry
	Set Me.currentEntry = Me.baseIterator.GetFirstEntry()
	Me.Valid = True ' starting from the beginning makes iterator valid again
	Set GetFirstEntry = GetNextValidEntry()
End Function ' FilteredDirectoryIterator.GetFirstEntry


'/**************************************************************************
' * returns next valid DirectoryEntry of the Directory.
' *
' * @return			next DirectoryEntry of the Directory or Nothing, if there is no next valid entry in the directory.
```

```
		' *
		' * @author		Thomas Bahn <tbahn@assono.de>
		' * @version		2007-10-06
		' ************************************************************************/
		Public Function GetNextEntry() As DirectoryEntry
			If Not Me.Valid Then Error 32003, "DirectoryIterator is no longer valid"

			Set Me.currentEntry = Me.baseIterator.GetNextEntry()
			Set GetNextEntry = GetNextValidEntry()
		End Function ' FilteredDirectoryIterator.GetNextEntry

End Class ' FilteredDirectoryIterator




'/************************************************************************
' * iterates through all visible DirectoryEntries.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' ************************************************************************/
Public Class VisibleOnlyDirectoryIterator As FilteredDirectoryIterator

		'/************************************************************************
		' * Constructor - initializes attributes.
		' *
		' * @param		baseIterator base DirectoryIterator to iterate through.
		' *
		' * @author		Thomas Bahn <tbahn@assono.de>
		' * @version		2007-10-06
		' ************************************************************************/
		Public Sub New(baseIterator As DirectoryIterator), FilteredDirectoryIterator(baseIterator)

		End Sub ' VisibleOnlyDirectoryIterator.New



		'/************************************************************************
		' * returns if current directory entry is visible.
		' *
		' * @return			true, if and only if current directory entry is visible.
		' *
		' * @author		Thomas Bahn <tbahn@assono.de>
		' * @version		2007-10-06
		' ************************************************************************/
		Private Function CurrentEntryIsValid() As Boolean
			CurrentEntryIsValid = Not(currentEntry.IsHidden())
```

```
	End Function ' VisibleOnlyDirectoryIterator.CurrentEntryIsValid

End Class ' VisibleOnlyDirectoryIterator




'/**************************************************************************
' * iterates through all DirectoryEntries, which are files.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' **************************************************************************
Public Class FilesOnlyDirectoryIterator As FilteredDirectoryIterator

	'/**************************************************************************
	' * Constructor - initializes attributes.
	' *
	' * @param		baseIterator base DirectoryIterator to iterate through.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' **************************************************************************/
	Public Sub New(baseIterator As DirectoryIterator), FilteredDirectoryIterator(baseIterator)

	End Sub ' FilesOnlyDirectoryIterator.New



	'/**************************************************************************
	' * returns if current directory entry is a file.
	' *
	' * @return			true, if and only if current directory entry is a file.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' **************************************************************************/
	Private Function CurrentEntryIsValid() As Boolean
		CurrentEntryIsValid = (Typename(currentEntry) = "FILEENTRY")
	End Function ' FilesOnlyDirectoryIterator.CurrentEntryIsValid

End Class ' FilesOnlyDirectoryIterator




'/**************************************************************************
```

```
' * iterates through all DirectoryEntries, which are folders.
' *
' * @author            Thomas Bahn <tbahn@assono.de>
' * @version           2007-10-06
' ************************************************************************
Public Class FoldersOnlyDirectoryIterator As FilteredDirectoryIterator

	'/**********************************************************************
	' * Constructor - initializes attributes.
	' *
	' * @param             baseIterator base DirectoryIterator to iterate through.
	' *
	' * @author            Thomas Bahn <tbahn@assono.de>
	' * @version           2007-10-06
	' **********************************************************************/
	Public Sub New(baseIterator As DirectoryIterator), FilteredDirectoryIterator(baseIterator)

	End Sub ' FoldersOnlyDirectoryIterator.New



	'/**********************************************************************
	' * returns if current directory entry is a folder.
	' *
	' * @return                          true, if and only if current directory entry is a folder.
	' *
	' * @author            Thomas Bahn <tbahn@assono.de>
	' * @version           2007-10-06
	' **********************************************************************/
	Private Function CurrentEntryIsValid() As Boolean
		CurrentEntryIsValid = (Typename(currentEntry) = "FOLDERENTRY")
	End Function ' FoldersOnlyDirectoryIterator.CurrentEntryIsValid

End Class ' FoldersOnlyDirectoryIterator




'/**************************************************************************
' * <b>Abstract class</b>: represents a command (operation) on a DirectoryEntry.
' *
' * @author            Thomas Bahn <tbahn@assono.de>
' * @version           2007-10-07
' **************************************************************************
Public Class DirectoryEntryCommand

	'/**********************************************************************
	' * current DirectoryEntry
```

```
	' ****************************************************************************/
	Private currentEntry As DirectoryEntry


	'/*****************************************************************************
	' * Constructor - initializes attributes.
	' *
	' * @param		currentEntry DirectoryEntry, this command operates on.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-07
	' ****************************************************************************/
	Public Sub New(currentEntry As DirectoryEntry)
		If Typename(Me) = "DIRECTORYENTRYCOMMAND" Then
			Error 32001, |Abstract class: you cannot create objects with this class ("| & Typename(Me) & |") directly, only with concrete subclasses!|
		End If
		Set Me.currentEntry = currentEntry
	End Sub ' DirectoryEntryCommand.New



	'/*****************************************************************************
	' * <b>Abstract method</b>: executes command.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' ****************************************************************************/
	Public Sub Execute()
		Error 32000, "Abstract method; has to be implemented in a subclass!"
	End Sub ' DirectoryEntryCommand.Execute


End Class ' DirectoryEntryCommand




'/*******************************************************************************
' * <b>Abstract class</b>: represents a command (operation) on a DirectoryEntry, which can be reversed (= is undoable).
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' ******************************************************************************/
Public Class ReversibleDirectoryEntryCommand As DirectoryEntryCommand

	'/*****************************************************************************
	' * Constructor - initializes attributes.
	' *
```

```
		' * @param		currentEntry DirectoryEntry, this command operates on.
		' *
		' * @author		Thomas Bahn <tbahn@assono.de>
		' * @version		2007-10-07
		' ************************************************************************/
		Public Sub New(currentEntry As DirectoryEntry), DirectoryEntryCommand(currentEntry)
			If Typename(Me) = "REVERSIBLEDIRECTORYENTRYCOMMAND" Then
				Error 32001, |Abstract class: you cannot create objects with this class ("| & Typename(Me) & |") directly, only with concrete subclasses!|
			End If
		End Sub ' ReversibleDirectoryEntryCommand.New



		'/***********************************************************************
		' * <b>Abstract method</b>: reverses command execution.
		' *
		' * @author		Thomas Bahn <tbahn@assono.de>
		' * @version		2007-10-06
		' ***********************************************************************/
		Public Sub Undo()
			Error 32000, "Abstract method; has to be implemented in a subclass!"
		End Sub ' ReversibleDirectoryEntryCommand.Undo


End Class ' ReversibleDirectoryEntryCommand




'/***************************************************************************
' * represents a delete operation on a DirectoryEntry.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' ***************************************************************************/
Public Class DeleteDirectoryEntryCommand As DirectoryEntryCommand

		'/***********************************************************************
		' * Constructor - initializes attributes.
		' *
		' * @param		currentEntry DirectoryEntry, this command operates on.
		' *
		' * @author		Thomas Bahn <tbahn@assono.de>
		' * @version		2007-10-06
		' ***********************************************************************/
		Public Sub New(currentEntry As DirectoryEntry), DirectoryEntryCommand(currentEntry)

		End Sub ' RenameDirectoryEntryCommand.New
```

```
	'/**************************************************************************
	' * executes command.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' **************************************************************************/
	Public Sub Execute()
		If currentEntry Isa "FileEntry" Then
			Kill currentEntry.GetParent().GetLocation() & "/" & currentEntry.GetEntryName()
		Elseif currentEntry Isa "FolderEntry" Then
			Rmdir currentEntry.GetParent().GetLocation() & "/" & currentEntry.GetEntryName()
		End If

		Call currentEntry.GetParent().MarkModified() ' marks directory as modified
	End Sub ' DeleteDirectoryEntryCommand.Execute


End Class ' DeleteDirectoryEntryCommand




'/**************************************************************************
' * represents a renaming operation on a DirectoryEntry.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' **************************************************************************/
Public Class RenameDirectoryEntryCommand As ReversibleDirectoryEntryCommand

	'/**************************************************************************
	' * name including location of the DirectoryEntry before renaming.
	' **************************************************************************/
	Private oldName As String

	'/**************************************************************************
	' * new name including location of the DirectoryEntry after renaming.
	' **************************************************************************/
	Private newName As String


	'/**************************************************************************
	' * Constructor - initializes attributes.
	' *
	' * @param		currentEntry DirectoryEntry, this command operates on.
```

```
        ' * @param          newName new name of the DirectoryEntry.
        ' *
        ' * @author         Thomas Bahn <tbahn@assono.de>
        ' * @version        2007-10-06
        ' ************************************************************************/
        Public Sub New(currentEntry As DirectoryEntry, newName As String), ReversibleDirectoryEntryCommand(currentEntry)
                Me.oldName = currentEntry.GetParent().GetLocation() & currentEntry.GetEntryName()
                Me.newName = currentEntry.GetParent().GetLocation() & newName
        End Sub ' RenameDirectoryEntryCommand.New



        '/************************************************************************
        ' * executes command.
        ' *
        ' * @author         Thomas Bahn <tbahn@assono.de>
        ' * @version        2007-10-06
        ' ************************************************************************/
        Public Sub Execute()
                Name oldName As newName

                Call currentEntry.GetParent().MarkModified() ' marks directory as modified
        End Sub ' RenameDirectoryEntryCommand.Execute



        '/************************************************************************
        ' * reverses command execution.
        ' *
        ' * @author         Thomas Bahn <tbahn@assono.de>
        ' * @version        2007-10-06
        ' ************************************************************************/
        Public Sub Undo()
                Name newName As oldName

                Call currentEntry.GetParent().MarkModified() ' marks directory as modified
        End Sub ' RenameDirectoryEntryCommand.Undo


End Class ' RenameDirectoryEntryCommand



'/************************************************************************
' * represents a move operation on a DirectoryEntry.
' *
' * @author         Thomas Bahn <tbahn@assono.de>
' * @version        2007-10-06
```

```
' ****************************************************************************
Public Class MoveDirectoryEntryCommand As ReversibleDirectoryEntryCommand

	'/*****************************************************************************
	' * name including location of the DirectoryEntry before moving.
	' *****************************************************************************/
	Private oldName As String

	'/*****************************************************************************
	' * new name including location of the DirectoryEntry after moving.
	' *****************************************************************************/
	Private newName As String


	'/*****************************************************************************
	' * Constructor - initializes attributes.
	' *
	' * @param		currentEntry DirectoryEntry, this command operates on.
	' * @param		newName new name of the DirectoryEntry.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' *****************************************************************************/
	Public Sub New(currentEntry As DirectoryEntry, newFolderName As String), ReversibleDirectoryEntryCommand(currentEntry)
		Me.oldName = currentEntry.GetParent().GetLocation() & currentEntry.GetEntryName()
		Me.newName = newFolderName & currentEntry.GetEntryName()
	End Sub ' MoveDirectoryEntryCommand.New



	'/*****************************************************************************
	' * executes command.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' *****************************************************************************/
	Public Sub Execute()
		Name oldName As newName

		Call currentEntry.GetParent().MarkModified() ' marks directory as modified
	End Sub ' MoveDirectoryEntryCommand.Execute



	'/*****************************************************************************
	' * reverses command execution.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
```

```
        ' ************************************************************************************/
        Public Sub Undo()
                Name newName As oldName

                Call currentEntry.GetParent().MarkModified() ' marks directory as modified
        End Sub ' MoveDirectoryEntryCommand.Undo


End Class ' MoveDirectoryEntryCommand




'/************************************************************************************
' * manages a LILO storage aka Queue.
' *
' * Simple and very inefficient implementation, use with care!
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-07
' ************************************************************************************
Public Class Queue

        '/************************************************************************************
        ' * array to store the elements of the queue.
        ' ************************************************************************************/
        Private queueArray() As Variant


        '/************************************************************************************
        ' * Constructor - initializes attributes.
        ' *
        ' * @author          Thomas Bahn <tbahn@assono.de>
        ' * @version         2007-10-07
        ' ************************************************************************************/
        Public Sub New()
                Call Me.Emtpy()
        End Sub ' Queue.New



        '/************************************************************************************
        ' * empties the queue.
        ' *
        ' * @author          Thomas Bahn <tbahn@assono.de>
        ' * @version         2007-10-06
        ' ************************************************************************************/
        Public Sub Emtpy()
```

```
        Redim Me.queueArray(0 To 0)
        Set Me.queueArray(0) = Nothing ' empty stack has Nothing as first and only entry
End Sub ' Queue.Emtpy


'/*****************************************************************************
' * returns if queue is empty.
' *
' * @return                      true, if and only if queue is empty.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-06
' *****************************************************************************/
Public Function IsEmpty As Boolean
        Me.IsEmpty = (Ubound(Me.queueArray) = 0) And (Me.queueArray(0) Is Nothing)
End Function ' Queue.IsEmpty


'/*****************************************************************************
' * Enques an element at the end of the queue.
' *
' * @param           newElement element to be put into the queue.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-06
' *****************************************************************************/
Public Sub Enque(newElement As Variant)
        If Not Me.IsEmpty Then ' expand array, if necessary
                Redim Preserve Me.queueArray(Lbound(Me.queueArray) To Ubound(Me.queueArray) + 1)
        End If

        If Isobject(newElement) Then ' put new element into topmost array entry
                Set Me.queueArray(Ubound(Me.queueArray)) = newElement
        Else
                Me.queueArray(Ubound(Me.queueArray)) = newElement
        End If
End Sub ' Queue.Enque


'/*****************************************************************************
' * returns the frontmost element from the queue.
' *
' * @return                      frontmost element from the queue.
' *
' * @author          Thomas Bahn <tbahn@assono.de>
' * @version         2007-10-06
' *****************************************************************************/
Public Function Front As Variant
        If Me.IsEmpty Then
```

```
                Set Front = Nothing
            Elseif Isobject(Me.queueArray(Lbound(Me.queueArray))) Then ' get frontmost array entry
                Set Front = Me.queueArray(Lbound(Me.queueArray))
            Else
                Front = Me.queueArray(Lbound(Me.queueArray))
            End If
        End Function ' Queue.Front


        '/*****************************************************************************
        ' * returns the frontmost element and removes it from the queue.
        ' *
        ' * @return                  frontmost element from the queue.
        ' *
        ' * @author          Thomas Bahn <tbahn@assono.de>
        ' * @version         2007-10-07
        ' *****************************************************************************/
        Public Function Deque As Variant
            Dim index As Integer

            If Me.IsEmpty Then
                Set Deque = Nothing
            Elseif Isobject(Me.queueArray(Lbound(Me.queueArray))) Then ' get frontmost array entry
                Set Deque = Me.queueArray(Lbound(Me.queueArray))
            Else
                Deque = Me.queueArray(Lbound(Me.queueArray))
            End If

            ' move the elements downward by copying them
            ' VERY INEFFICIENT
            ' Don't do this at home! ;-)
            For index = Lbound(Me.queueArray) To (Ubound(Me.queueArray) - 1)
                If Isobject(Me.queueArray(index)) Then
                    Set Me.queueArray(index) = Me.queueArray(index + 1)
                Else
                    Me.queueArray(index) = Me.queueArray(index + 1)
                End If
            Next

            Set Me.queueArray(Ubound(Me.queueArray)) = Nothing ' clear topmost array entry
            If Ubound(Me.queueArray) > Lbound(Me.queueArray) Then ' shorten array, if no emtpy
                Redim Preserve Me.queueArray(Lbound(Me.queueArray) To Ubound(Me.queueArray) - 1)
            End If
        End Function ' Queue.Deque


    End Class ' Queue
```

```
'/**************************************************************************
' * manages a LIFO storage aka Stack.
' *
' * Simple and inefficient implementation, use with care!
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' **************************************************************************
Public Class Stack

	'/**************************************************************************
	' * array to store the elements of the stack.
	' **************************************************************************/
	Private stackArray() As Variant


	'/**************************************************************************
	' * Constructor - initializes attributes.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-07
	' **************************************************************************/
	Public Sub New()
		Call Me.Emtpy()
	End Sub ' Stack.New



	'/**************************************************************************
	' * emtpies the stack.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
	' * @version		2007-10-06
	' **************************************************************************/
	Public Sub Emtpy()
		Redim Me.stackArray(0 To 0)
		Set Me.stackArray(0) = Nothing ' empty stack has Nothing as first and only entry
	End Sub ' Stack.Emtpy


	'/**************************************************************************
	' * returns if stack is empty.
	' *
	' * @return			true, if and only if stack is empty.
	' *
	' * @author		Thomas Bahn <tbahn@assono.de>
```

```
' * @version           2007-10-06
' ************************************************************************************/
Public Function IsEmpty As Boolean
        Me.IsEmpty = (Ubound(Me.stackArray) = 0) And (Me.stackArray(0) Is Nothing)
End Function ' Stack.IsEmpty


'/************************************************************************************
' * Pushes an element onto the top of the stack.
' *
' * @param             newElement element to be put onto the stack.
' *
' * @author            Thomas Bahn <tbahn@assono.de>
' * @version           2007-10-06
' ************************************************************************************/
Public Sub Push(newElement As Variant)
        If Not Me.IsEmpty Then ' expand array, if necessary
                Redim Preserve Me.stackArray(Lbound(Me.stackArray) To Ubound(Me.stackArray) + 1)
        End If

        If Isobject(newElement) Then ' put new element into topmost array entry
                Set Me.stackArray(Ubound(Me.stackArray)) = newElement
        Else
                Me.stackArray(Ubound(Me.stackArray)) = newElement
        End If
End Sub ' Stack.Push


'/************************************************************************************
' * returns the topmost element from the stack.
' *
' * @return                      topmost element from the stack.
' *
' * @author            Thomas Bahn <tbahn@assono.de>
' * @version           2007-10-06
' ************************************************************************************/
Public Function Top As Variant
        If Me.IsEmpty Then
                Set Top = Nothing
        Elseif Isobject(Me.stackArray(Ubound(Me.stackArray))) Then ' get topmost array entry
                Set Top = Me.stackArray(Ubound(Me.stackArray))
        Else
                Top = Me.stackArray(Ubound(Me.stackArray))
        End If
End Function ' Stack.Top


'/************************************************************************************
' * returns the topmost element and removes it from the stack.
```

```
		' *
		' * @return			topmost element from the stack.
		' *
		' * @author		Thomas Bahn <tbahn@assono.de>
		' * @version		2007-10-07
		' ************************************************************************************/
		Public Function Pop As Variant
			If Me.IsEmpty Then
				Set Pop = Nothing
			Elseif Isobject(Me.stackArray(Ubound(Me.stackArray))) Then ' get topmost array entry
				Set Pop = Me.stackArray(Ubound(Me.stackArray))
			Else
				Pop = Me.stackArray(Ubound(Me.stackArray))
			End If

			Set Me.stackArray(Ubound(Me.stackArray)) = Nothing ' clear topmost array entry
			If Ubound(Me.stackArray) > Lbound(Me.stackArray) Then ' shorten array, if no emtpy
				Redim Preserve Me.stackArray(Lbound(Me.stackArray) To Ubound(Me.stackArray) - 1)
			End If
		End Function ' Stack.Pop


End Class ' Stack




'/************************************************************************************
' * private variable to store the reference to the only DirectoryEntryCommandProcessor instance (cf. Singleton)
' ************************************************************************************
Private commandProcessor As DirectoryEntryCommandProcessor


'/************************************************************************************
' * manages DirectoryEntryCommands and enables batch execution/undo.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' ************************************************************************************
Public Class DirectoryEntryCommandProcessor

		'/************************************************************************************
		' * queue of DirectoryEntryCommand to be executed later.
		' ************************************************************************************/
		Private executionQueue As Queue


		'/************************************************************************************
```

```
' * Clears the execution queue.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' ************************************************************************************/
Public Sub ClearExecutionQueue()
	Call Me.executionQueue.Emtpy
End Sub ' DirectoryEntryCommandProcessor.ClearExecutionQueue


'/************************************************************************************
' * appends a DirectoryEntryCommand to the execution queue.
' *
' * @param		newCommand DirectoryEntryCommand to be appended.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' ************************************************************************************/
Private Sub AppendCommandToExecutionQueue(newCommand As DirectoryEntryCommand)
	Call Me.executionQueue.Enque(newCommand)
End Sub ' DirectoryEntryCommandProcessor.AppendCommandToExecutionQueue



'/************************************************************************************
' * stack of ReversibleDirectoryEntryCommand, which could be undone.
' ************************************************************************************/
Private undoStack As Stack


'/************************************************************************************
' * Clears the undo stack.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' ************************************************************************************/
Public Sub ClearUndoStack()
	Call Me.undoStack.Emtpy
End Sub ' DirectoryEntryCommandProcessor.ClearUndoStack


'/************************************************************************************
' * appends a DirectoryEntryCommand to the undo stack.
' *
' * @param		newCommand DirectoryEntryCommand to be appended.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' ************************************************************************************/
```

```
Private Sub AppendCommandToUndoStack(newCommand As DirectoryEntryCommand)
	Call Me.undoStack.Push(newCommand)
End Sub ' DirectoryEntryCommandProcessor.AppendCommandToUndoStack



'/**************************************************************************
' * Constructor - initializes attributes.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-07
' **************************************************************************/
Public Sub New()
	Set Me.executionQueue = New Queue()
	Set Me.undoStack = New Stack()
End Sub ' DirectoryEntryCommandProcessor.New



'/**************************************************************************
' * executes a DirectoryEntryCommand directly.
' *
' * @param		newCommand DirectoryEntryCommand to be executed.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' **************************************************************************/
Public Sub ExecuteCommand(newCommand As DirectoryEntryCommand)
	Call newCommand.Execute()

	If (newCommand Isa "ReversibleDirectoryEntryCommand") Then ' if command is reversible, push it onto the undo stack
		Call AppendCommandToUndoStack(newCommand)
	Else
		Call ClearUndoStack()
	End If
End Sub ' DirectoryEntryCommandProcessor.AppendCommand



'/**************************************************************************
' * appends a DirectoryEntryCommand to be processed later.
' *
' * @param		newCommand DirectoryEntryCommand to be appended.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' **************************************************************************/
Public Sub AppendCommand(newCommand As DirectoryEntryCommand)
	Call AppendCommandToExecutionQueue(newCommand)
```

```
End Sub ' DirectoryEntryCommandProcessor.AppendCommand


'/*****************************************************************************
' * executes the frontmost command in the execution queue.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' *****************************************************************************/
Public Sub ExecuteOne
	Dim currentCommand As DirectoryEntryCommand

	Set currentCommand = executionQueue.Deque

	Call currentCommand.Execute()

	If (currentCommand Isa "ReversibleDirectoryEntryCommand") Then ' if command is reversible, push it onto the undo stack
		Call AppendCommandToUndoStack(currentCommand)
	Else
		Call ClearUndoStack()
	End If
End Sub ' DirectoryEntryCommandProcessor.ExecuteOne


'/*****************************************************************************
' * executes all commands in the execution queue.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06
' *****************************************************************************/
Public Sub ExecuteAll

	Do Until executionQueue.IsEmpty
		Call ExecuteOne
	Loop
End Sub ' DirectoryEntryCommandProcessor.ExecuteAll


'/*****************************************************************************
' * undos the topmost command in the undo stack.
' *
' * @author		Thomas Bahn <tbahn@assono.de>
' * @version		2007-10-06Private commandProcessor As DirectoryEntryCommandProcessor
' *****************************************************************************/
Public Sub UndoOne
	Dim currentCommand As ReversibleDirectoryEntryCommand

	Set currentCommand = undoStack.Pop
```

```
                Call currentCommand.Undo()
        End Sub ' DirectoryEntryCommandProcessor.UndoOne


        '/**************************************************************************************
        ' * undos all commands in the undo stack.
        ' *
        ' * @author              Thomas Bahn <tbahn@assono.de>
        ' * @version             2007-10-06
        ' **************************************************************************************/
        Public Sub UndoAll

                Do Until undoStack.IsEmpty
                        Call UndoOne
                Loop
        End Sub ' DirectoryEntryCommandProcessor.UndoAll

End Class ' DirectoryEntryCommandProcessor

Public Function GetCommandProcessorInstance As DirectoryEntryCommandProcessor
        '/**************************************************************************************
        ' * creates one (and only one) instance of DirectoryEntryCommandProcessor if necessary and returns it.
        ' *
        ' * @return                          only instance of DirectoryEntryCommandProcessor.
        ' *
        ' * @author              Thomas Bahn <tbahn@assono.de>
        ' * @version             2007-10-09
        ' **************************************************************************************/

        If commandProcessor Is Nothing Then ' if commandProcessor is still not created
                Set commandProcessor = New DirectoryEntryCommandProcessor()
        End If

        Set GetCommandProcessorInstance = commandProcessor
End Function
```