VBA/ArcObjects

Slight Introduction

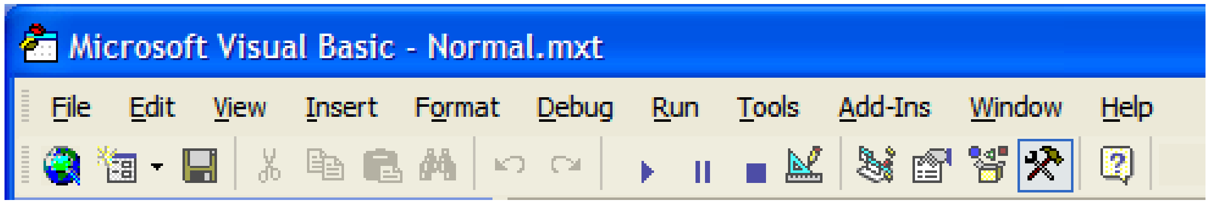
By José L. Flores

Objectives

- Introduce the concept of object programming to construct customized applications in the ArcGIS environment.
- Access of ArcObjects to accomplish the required task.
- Emphasis in the use of VisualBasic for Applications (VBA) programming language.

VBA - Menu

- Main menu used to access formatting, debugging, add-ins, and other functions
- Standard Toolbar used to run common functions



Return to ArcGIS

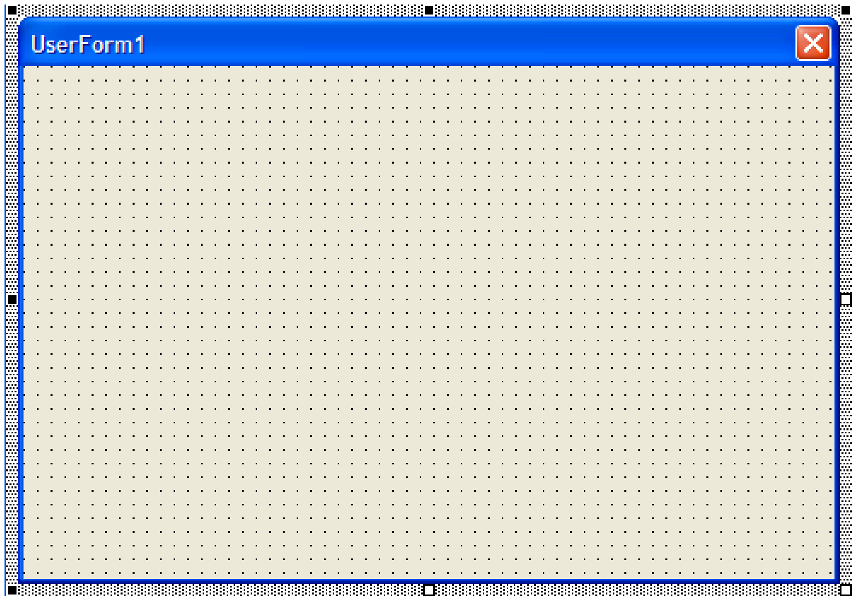
Save project

Run, pause or stop application

Access some functions

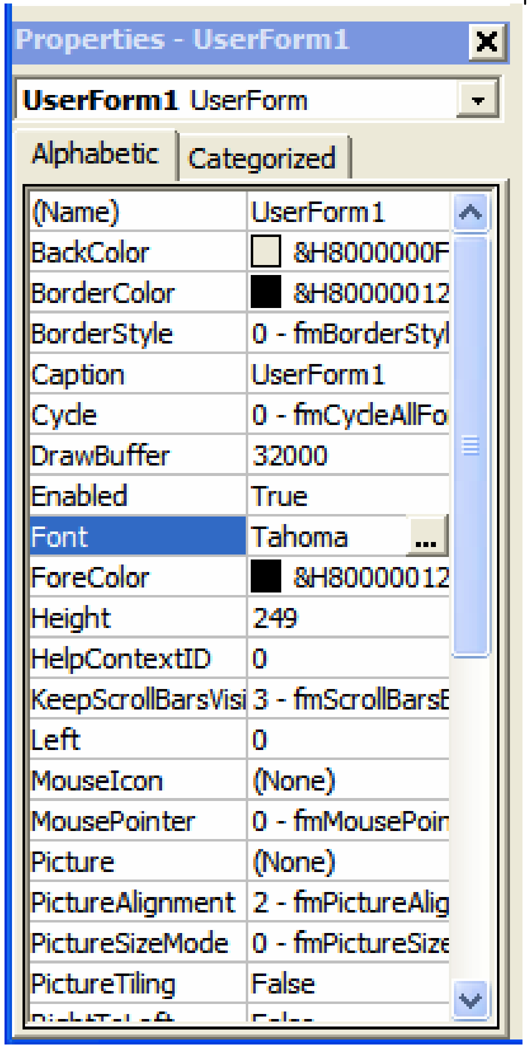
VBA – New Form window

- The new form window is used for the background in order to construct the interface of the intending application.
- It’s a blank canvas where you place the command objects like buttons, text boxes, etc.



VBA – Properties Windows

- Properties windows is a tool the access the properties of the objects in a form.
- Can modify properties like font type, font size, caption, object name, object size and placement, etc.


Properties - UserForm1

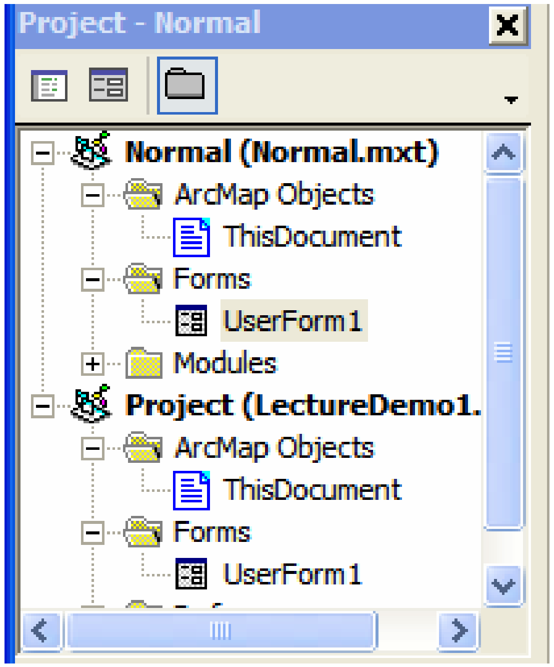
UserForm1 UserForm

Alphabetic | Categorized

(Name)	UserForm1
BackColor	⬜ &H8000000F
BorderColor	⬛ &H80000012
BorderStyle	0 - fmBorderStyl
Caption	UserForm1
Cycle	0 - fmCycleAllFo
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	⬛ &H80000012
Height	249
HelpContextID	0
KeepScrollBarsVisi	3 - fmScrollBarsE
Left	0
MouseIcon	(None)
MousePointer	0 - fmMousePoin
Picture	(None)
PictureAlignment	2 - fmPictureAlig
PictureSizeMode	0 - fmPictureSize
PictureTiling	False

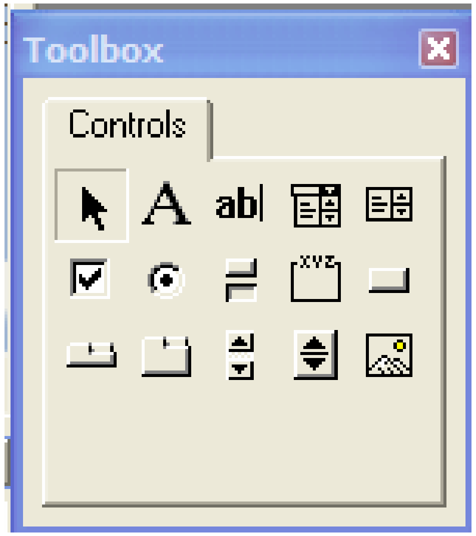
VBA – Project Window

- Project window allows you to visualize the documents, user forms, and modules that compose your project.
- It allows to keep track of these objects, since you may have more that one say user form or modules.
- You can delete and add user forms and modules.
- You can also view if the project is only accessible to current ArcMap file, or any ArcMap file (Normal.mxt).

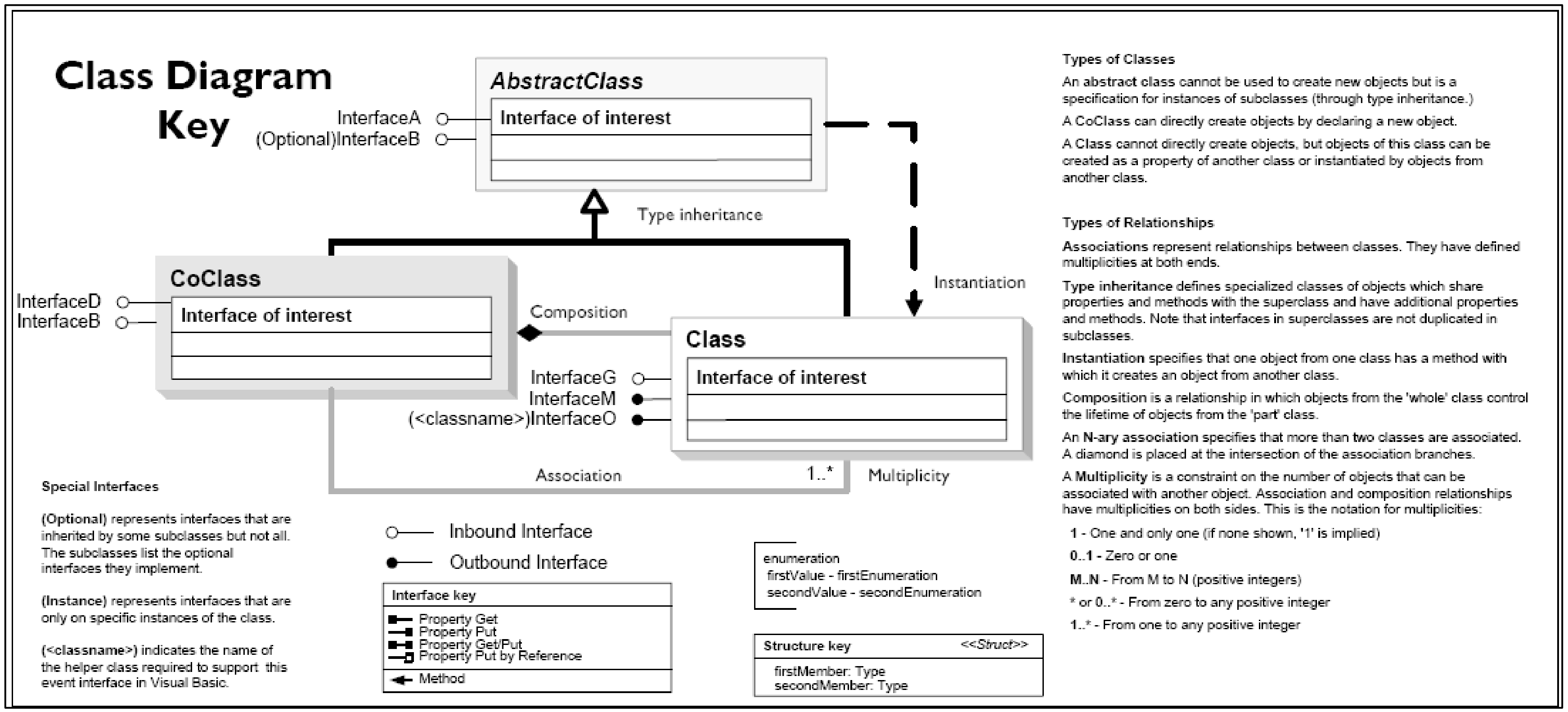


VBA – Toolbox

- Toolbox window allows you to access objects to be place in an user form.
- Some of the most common objects are label, textbox, combo box, check box, command button, radio button.
- These object have no functionality until you program them.



Class Diagram Key



Special Interfaces

(Optional) represents interfaces that are inherited by some subclasses but not all. The subclasses list the optional interfaces they implement.

(Instance) represents interfaces that are only on specific instances of the class.

(<classname>) indicates the name of the helper class required to support this event interface in Visual Basic.

Types of Classes

An **abstract class** cannot be used to create new objects but is a specification for instances of subclasses (through type inheritance.)

A **CoClass** can directly create objects by declaring a new object.

A **Class** cannot directly create objects, but objects of this class can be created as a property of another class or instantiated by objects from another class.

Types of Relationships

Associations represent relationships between classes. They have defined multiplicities at both ends.

Type inheritance defines specialized classes of objects which share properties and methods with the superclass and have additional properties and methods. Note that interfaces in superclasses are not duplicated in subclasses.

Instantiation specifies that one object from one class has a method with which it creates an object from another class.

Composition is a relationship in which objects from the 'whole' class control the lifetime of objects from the 'part' class.

An **N-ary association** specifies that more than two classes are associated. A diamond is placed at the intersection of the association branches.

A **Multiplicity** is a constraint on the number of objects that can be associated with another object. Association and composition relationships have multiplicities on both sides. This is the notation for multiplicities:

1 - One and only one (if none shown, '1' is implied)

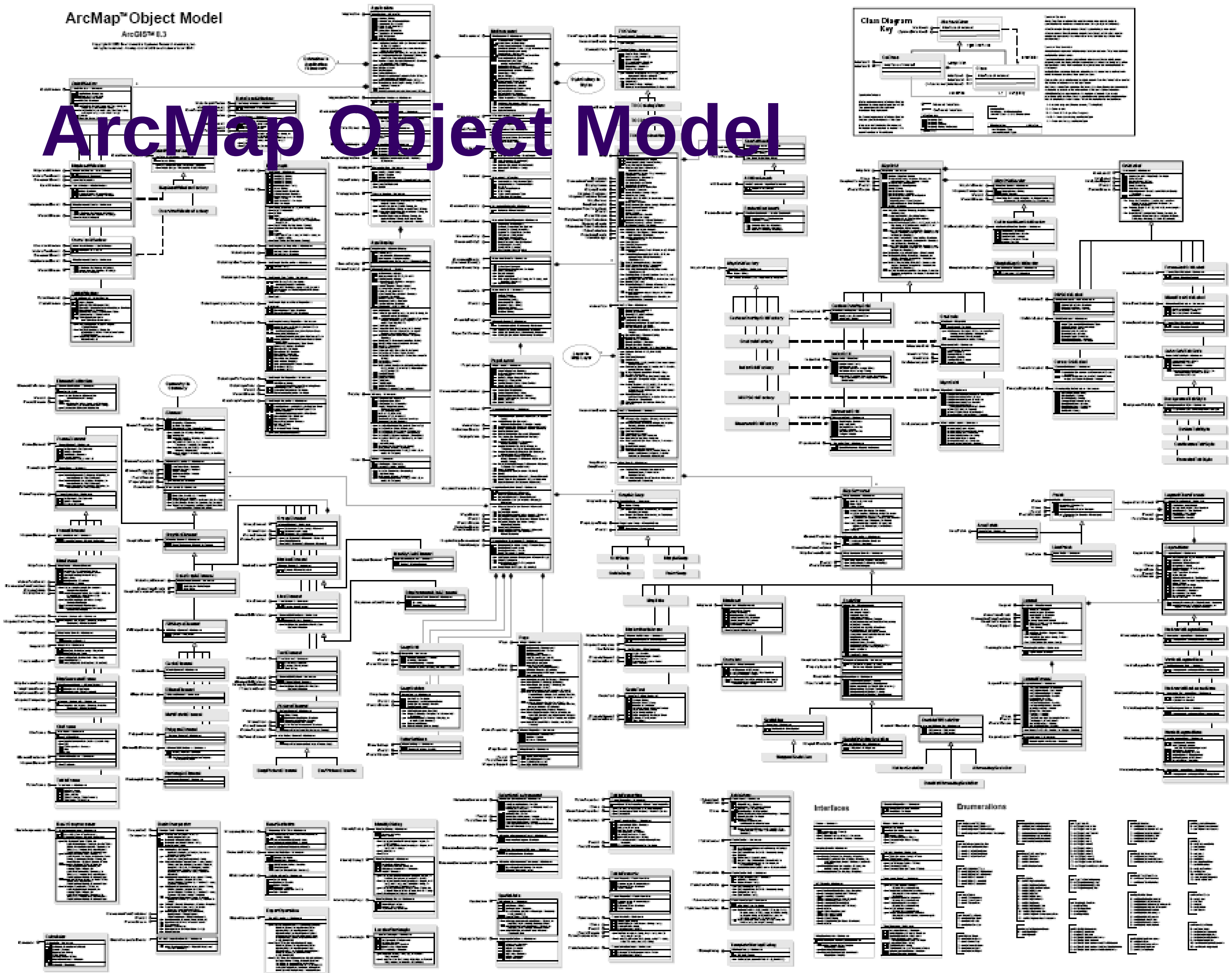
0..1 - Zero or one

M..N - From M to N (positive integers)

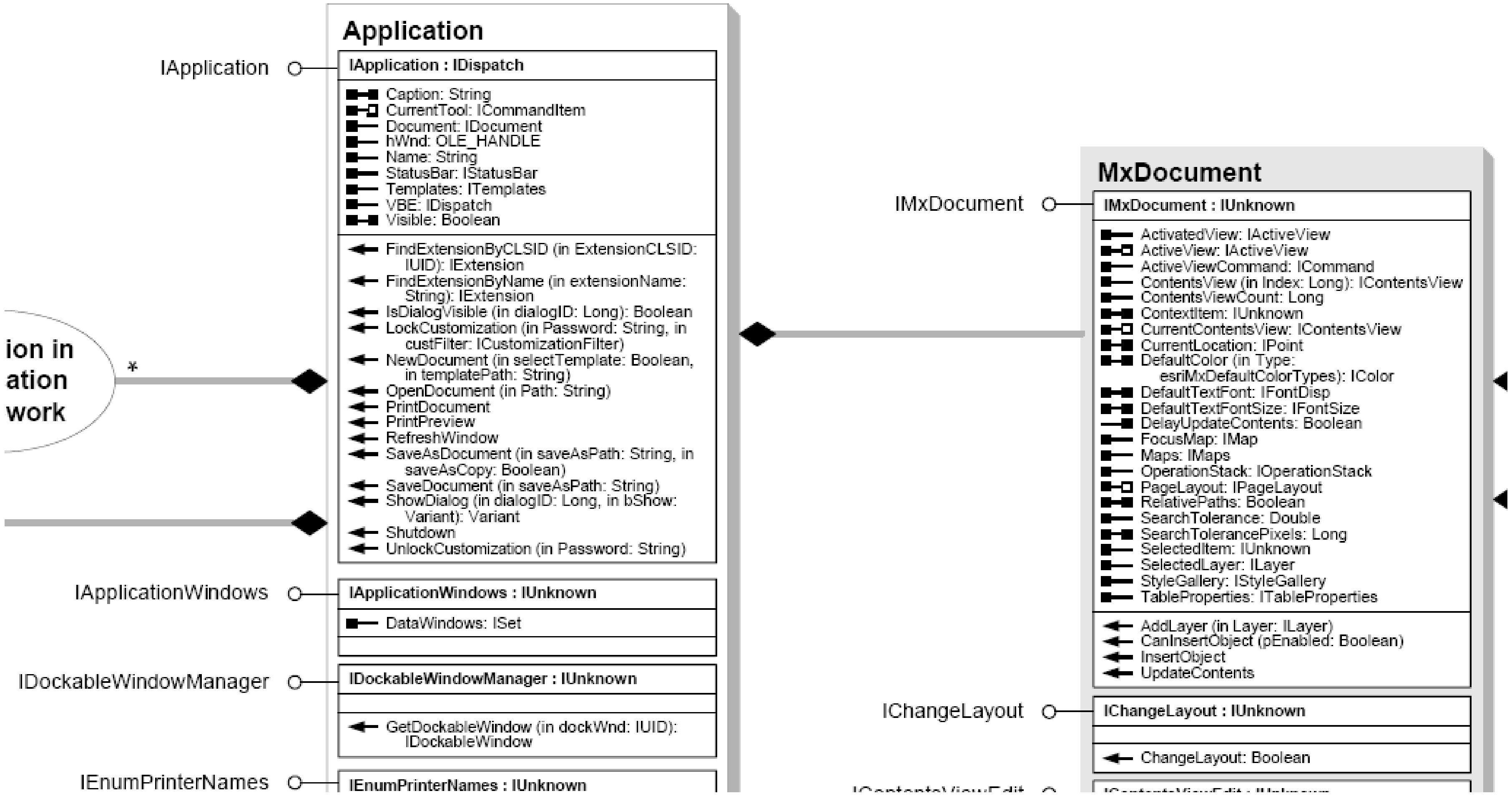
* or 0..* - From zero to any positive integer

1..* - From one to any positive integer

ArcMap Object Model

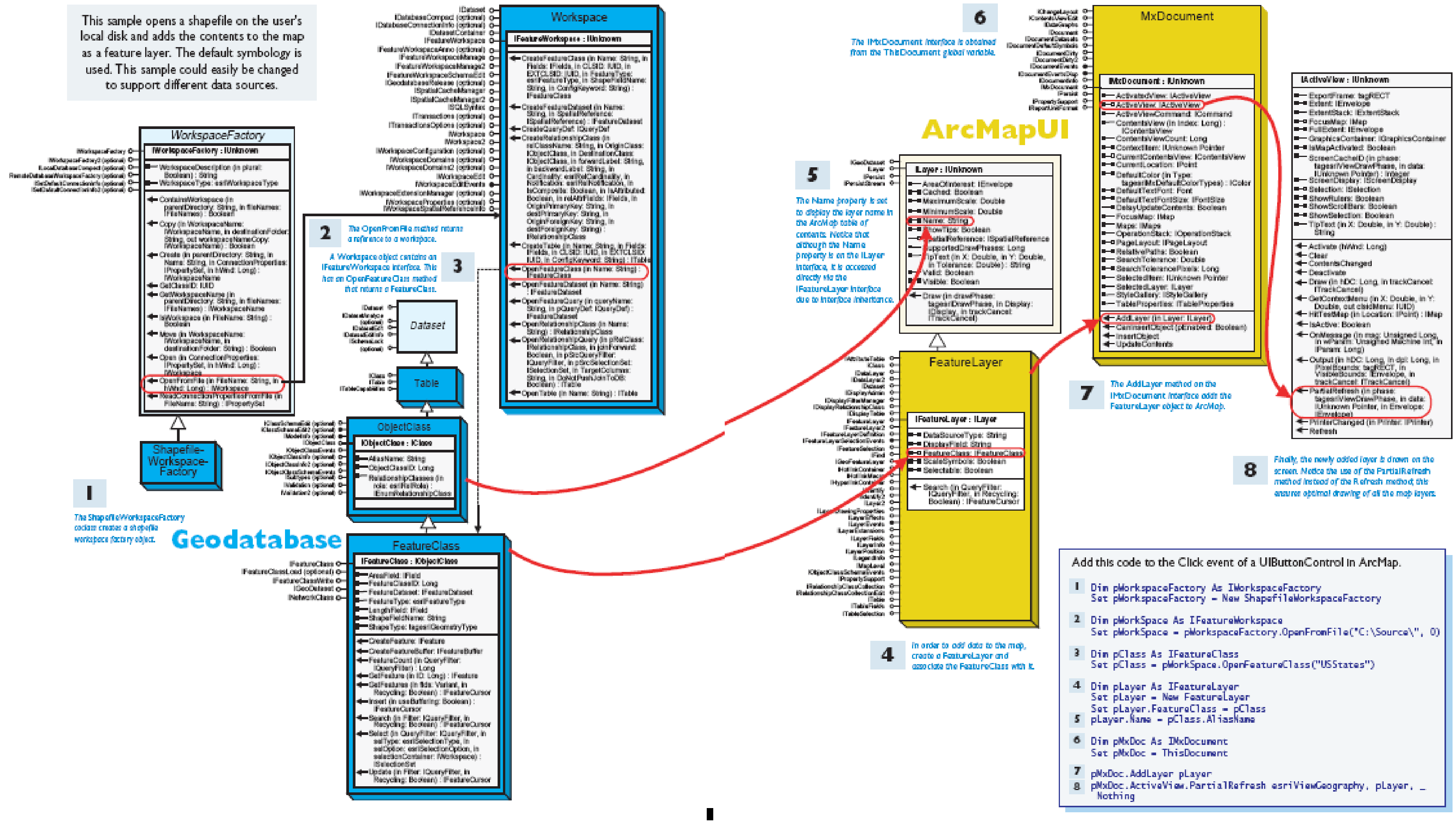


ArcMap Object Model Close Up



Application
IApplication
IApplication : IDispatch
Caption: String
CurrentTool: ICommandItem
Document: IDocument
hWnd: OLE_HANDLE
Name: String
StatusBar: IStatusBar
Templates: ITemplates
VBE: IDispatch
Visible: Boolean
FindExtensionByCLSID (in ExtensionCLSID: IUID): IExtension
FindExtensionByName (in extensionName: String): IExtension
IsDialogVisible (in dialogID: Long): Boolean
LockCustomization (in Password: String, in custFilter: ICustomizationFilter)
NewDocument (in selectTemplate: Boolean, in templatePath: String)
OpenDocument (in Path: String)
PrintDocument
PrintPreview
RefreshWindow
SaveAsDocument (in saveAsPath: String, in saveAsCopy: Boolean)
SaveDocument (in saveAsPath: String)
ShowDialog (in dialogID: Long, in bShow: Variant): Variant
Shutdown
UnlockCustomization (in Password: String)
IApplicationWindows
IApplicationWindows : IUnknown
DataWindows: ISet
IDockableWindowManager
IDockableWindowManager : IUnknown
GetDockableWindow (in dockWnd: IUID): IDockableWindow
IEnumPrinterNames
IEnumPrinterNames : IUnknown
ion in
ation
work
*
MxDocument
IMxDocument
IMxDocument : IUnknown
ActivatedView: IActiveView
ActiveView: IActiveView
ActiveViewCommand: ICommand
ContentsView (in Index: Long): IContentsView
ContentsViewCount: Long
ContextItem: IUnknown
CurrentContentsView: IContentsView
CurrentLocation: IPoint
DefaultColor (in Type: esriMxDefaultColorTypes): IColor
DefaultTextFont: IFontDisp
DefaultTextFontSize: IFontSize
DelayUpdateContents: Boolean
FocusMap: IMap
Maps: IMaps
OperationStack: IOperationStack
PageLayout: IPageLayout
RelativePaths: Boolean
SearchTolerance: Double
SearchTolerancePixels: Long
SelectedItem: IUnknown
SelectedLayer: ILayer
StyleGallery: IStyleGallery
TableProperties: ITableProperties
AddLayer (in Layer: ILayer)
CanInsertObject (pEnabled: Boolean)
InsertObject
UpdateContents
IChangeLayout
IChangeLayout : IUnknown
ChangeLayout: Boolean

Example Script & Object Model



WorkspaceFactory
IWorkspaceFactory : IUnknown
OpenFromFile (in FileName: String, in hWnd: Long) : IWorkspace
ShapefileWorkspaceFactory
Workspace
IFeatureWorkspace : IUnknown
OpenFeatureClass (in Name: String) : IFeatureClass
Dataset
Table
ObjectClass
IObjectClass : IClass
Geodatabase
FeatureClass
IFeatureClass : IObjectClass
ArcMapUI
ILayer : IUnknown
Name: String
FeatureLayer
IFeatureLayer : ILayer
FeatureClass: IFeatureClass
MxDocument
IMxDocument : IUnknown
ActiveView: IActiveView
AddLayer (in Layer: ILayer)
IActiveView : IUnknown
PartialRefresh (in phase: tagesriViewDrawPhase, in data: IUnknown Pointer, in Envelope: IEnvelope)

This sample opens a shapefile on the user's local disk and adds the contents to the map as a feature layer. The default symbology is used. This sample could easily be changed to support different data sources.

1 The ShapefileWorkspaceFactory coclass creates a shapefile workspace factory object.

2 The OpenFromFile method returns a reference to a workspace.

3 A Workspace object contains an IFeatureWorkspace interface. This has an OpenFeatureClass method that returns a FeatureClass.

4 In order to add data to the map, create a FeatureLayer and associate the FeatureClass with it.

5 The Name property is set to display the layer name in the ArcMap table of contents. Notice that although the Name property is on the ILayer interface, it is accessed directly via the IFeatureLayer interface due to interface inheritance.

6 The IMxDocument interface is obtained from the ThisDocument global variable.

7 The AddLayer method on the IMxDocument interface adds the FeatureLayer object to ArcMap.

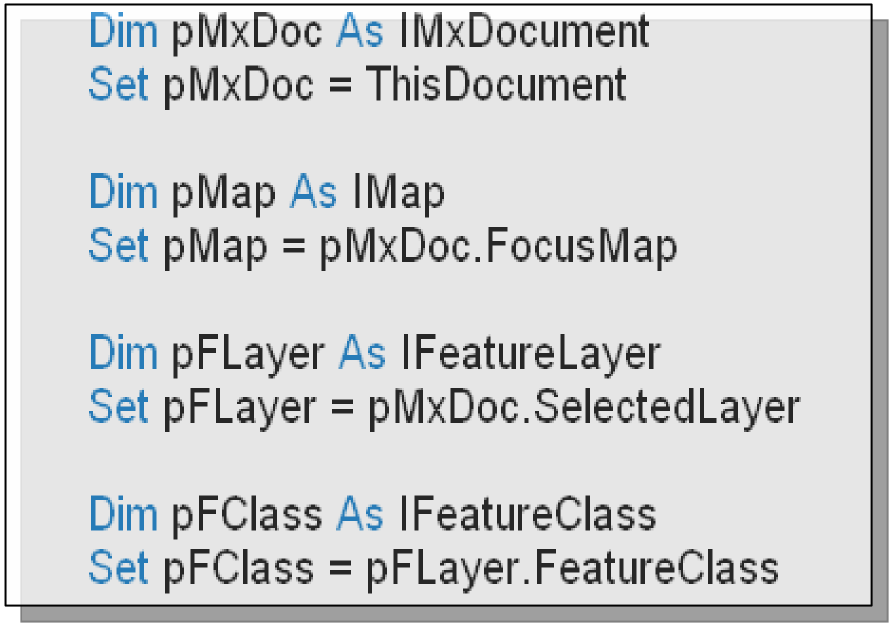
8 Finally, the newly added layer is drawn on the screen. Notice the use of the PartialRefresh method instead of the Refresh method; this ensures optimal drawing of all the map layers.

Add this code to the Click event of a UIButtonControl in ArcMap.

```
1  Dim pWorkspaceFactory As IWorkspaceFactory
   Set pWorkspaceFactory = New ShapefileWorkspaceFactory

2  Dim pWorkSpace As IFeatureWorkspace
   Set pWorkSpace = pWorkspaceFactory.OpenFromFile("C:\Source\", 0)

3  Dim pClass As IFeatureClass
   Set pClass = pWorkSpace.OpenFeatureClass("USStates")

4  Dim pLayer As IFeatureLayer
   Set pLayer = New FeatureLayer
   Set pLayer.FeatureClass = pClass
5  pLayer.Name = pClass.AliasName

6  Dim pMxDoc As IMxDocument
   Set pMxDoc = ThisDocument

7  pMxDoc.AddLayer pLayer
8  pMxDoc.ActiveView.PartialRefresh esriViewGeography, pLayer, _
    Nothing
```

Common SetUp Access Layer

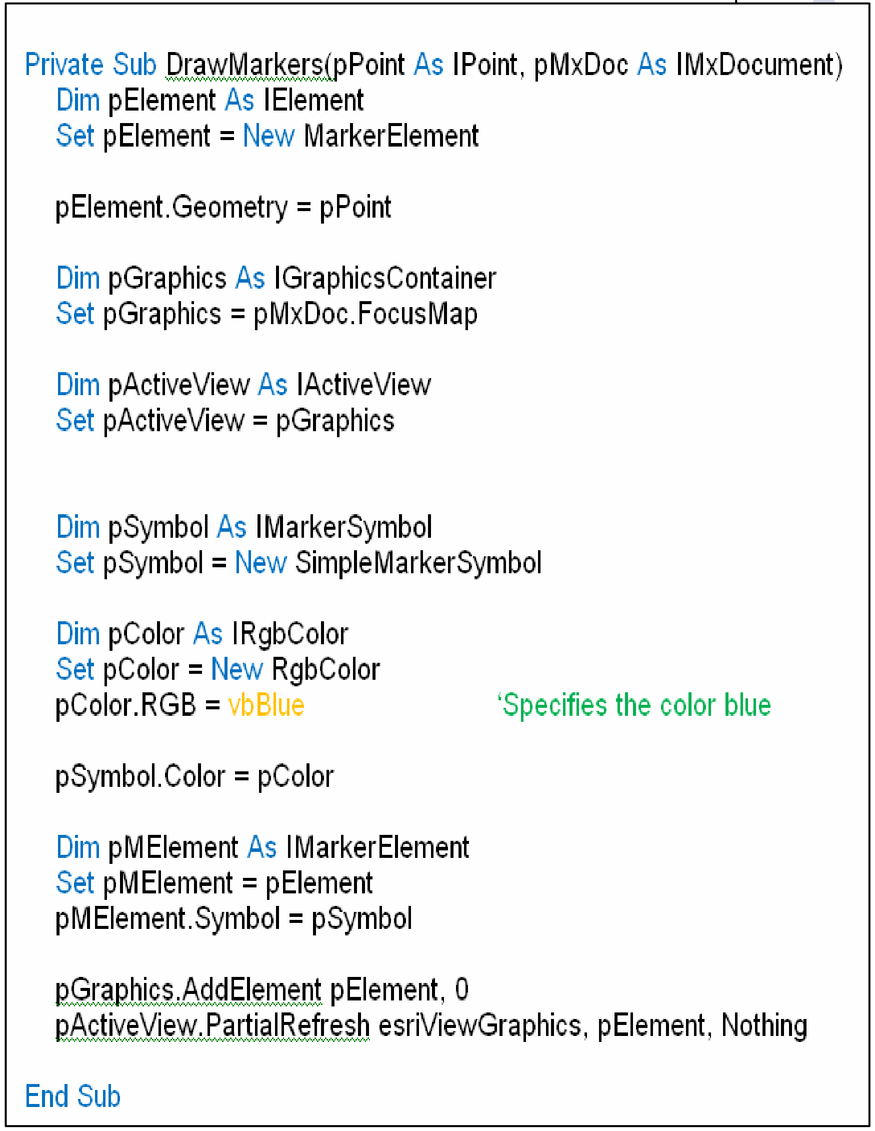
- A common setup is shown in the right.
- First two line to specify the document
- Second two lines to specify the map.
- Last four lines specifies the use of the highlighted layer.
- The option “pMxDoc.SelectLayer” can changed to “pMap.Layer(0)”
- If you replace the term “fearure” with “raster” you can access a raster layer.



```
Dim pMxDoc As IMxDocument
Set pMxDoc = ThisDocument

Dim pMap As IMap
Set pMap = pMxDoc.FocusMap

Dim pFLayer As IFeatureLayer
Set pFLayer = pMxDoc.SelectedLayer

Dim pFClass As IFeatureClass
Set pFClass = pFLayer.FeatureClass
```

Draw Marker Symbol

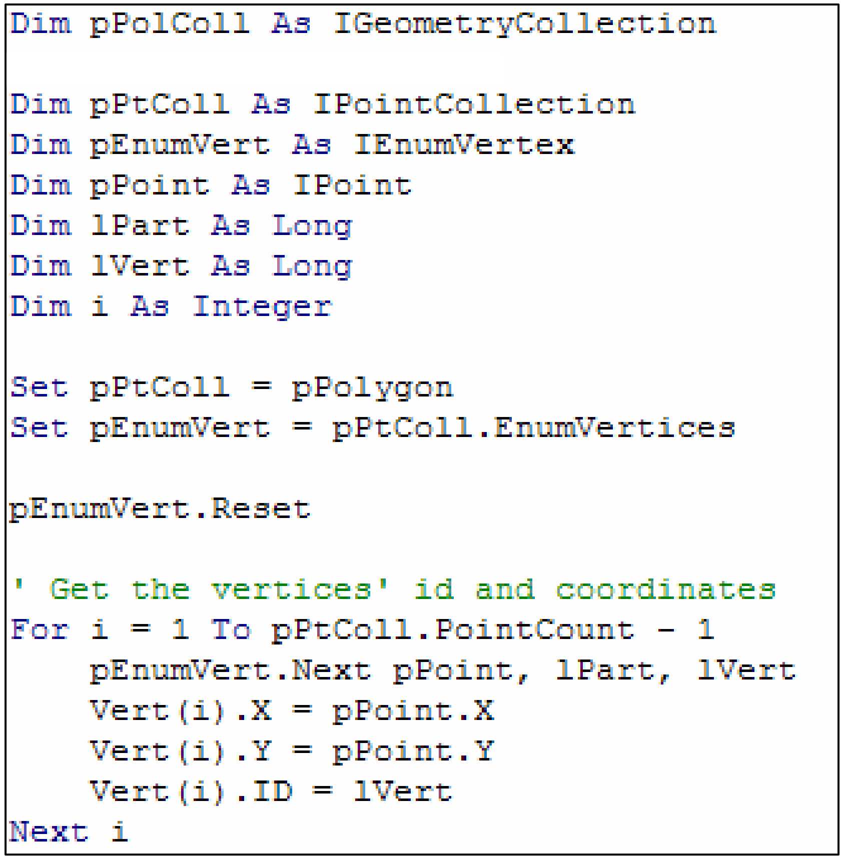
- This subroutine can be used to draw points (markers) in the select map.
- One arguments is pPoint, used for the X,Y of the point.
- Another argument pMxDoc to tell which map to draw it.
- This routine can be modified to allow flexibility, like color, size, and form of the marker.



```
Private Sub DrawMarkers(pPoint As IPoint, pMxDoc As IMxDocument)
   Dim pElement As IElement
   Set pElement = New MarkerElement

   pElement.Geometry = pPoint

   Dim pGraphics As IGraphicsContainer
   Set pGraphics = pMxDoc.FocusMap

   Dim pActiveView As IActiveView
   Set pActiveView = pGraphics


   Dim pSymbol As IMarkerSymbol
   Set pSymbol = New SimpleMarkerSymbol

   Dim pColor As IRgbColor
   Set pColor = New RgbColor
   pColor.RGB = vbBlue                  'Specifies the color blue

   pSymbol.Color = pColor

   Dim pMElement As IMarkerElement
   Set pMElement = pElement
   pMElement.Symbol = pSymbol

   pGraphics.AddElement pElement, 0
   pActiveView.PartialRefresh esriViewGraphics, pElement, Nothing

End Sub
```

Get Vertices' Data

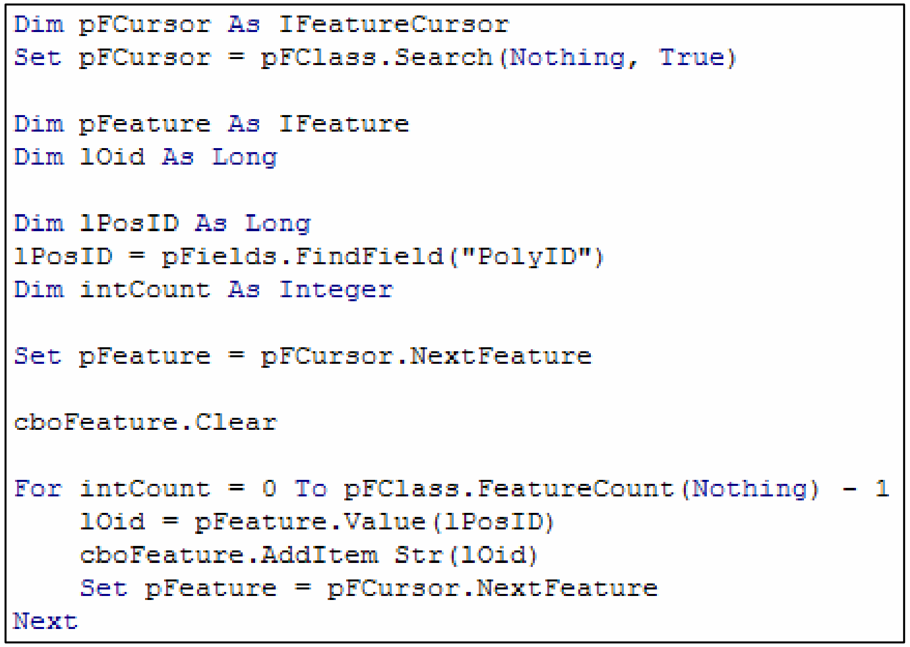
- To get the vertices coordinates and ID's there are two key interfaces:
  - iGeometryCollection
  - iPointCollection
  - iEnumVertex



```
Dim pPolColl As IGeometryCollection

Dim pPtColl As IPointCollection
Dim pEnumVert As IEnumVertex
Dim pPoint As IPoint
Dim lPart As Long
Dim lVert As Long
Dim i As Integer

Set pPtColl = pPolygon
Set pEnumVert = pPtColl.EnumVertices

pEnumVert.Reset

' Get the vertices' id and coordinates
For i = 1 To pPtColl.PointCount - 1
    pEnumVert.Next pPoint, lPart, lVert
    Vert(i).X = pPoint.X
    Vert(i).Y = pPoint.Y
    Vert(i).ID = lVert
Next i
```

List All Features on Layer

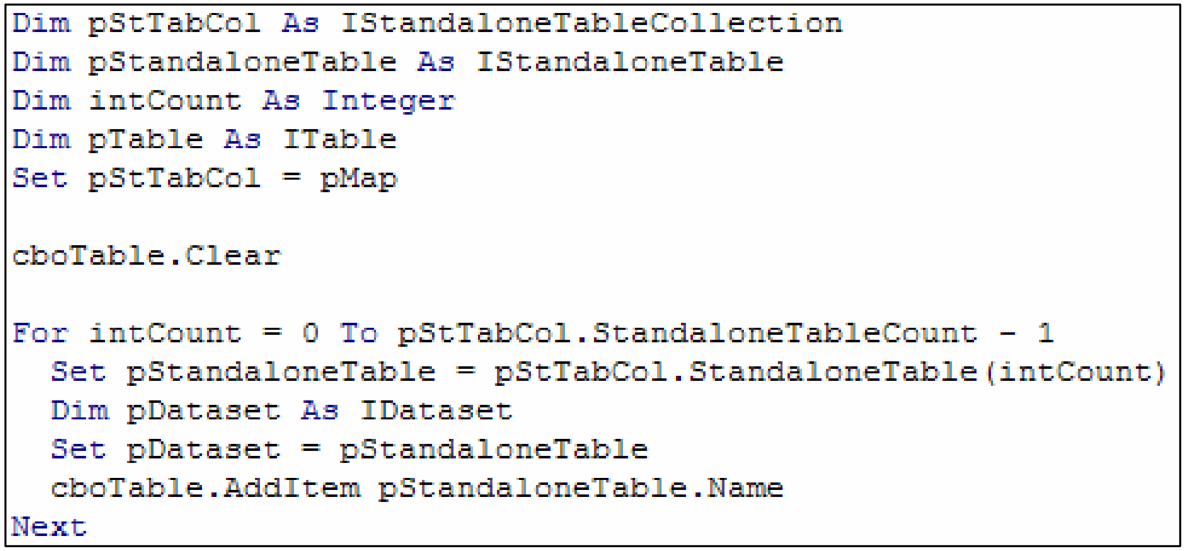
- The iFeatureCursor is the key in order to get all the features on a layer.
- In this case the items were loaded to a combo box on request.
- Now the list index on the combo box is the FID of the feature.



```
Dim pFCursor As IFeatureCursor
Set pFCursor = pFClass.Search(Nothing, True)

Dim pFeature As IFeature
Dim lOid As Long

Dim lPosID As Long
lPosID = pFields.FindField("PolyID")
Dim intCount As Integer

Set pFeature = pFCursor.NextFeature

cboFeature.Clear

For intCount = 0 To pFClass.FeatureCount(Nothing) - 1
    lOid = pFeature.Value(lPosID)
    cboFeature.AddItem Str(lOid)
    Set pFeature = pFCursor.NextFeature
Next
```

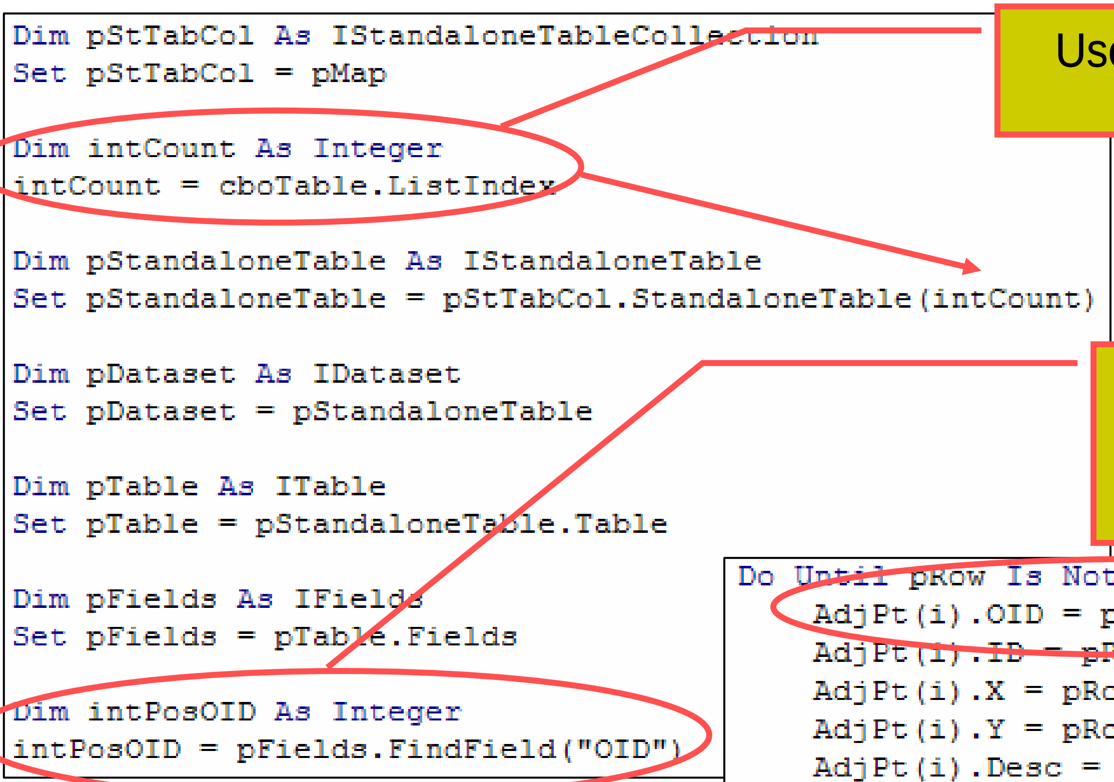
Get All Tables on Map

- The interfaces needed to get the available tables in the map are: iStandaloneTableCollection and IStandaloneTable.
- Similar to the features you may load them to combo box, and then it is possible to access the desired table by the use of list index from the combo box.



```
Dim pStTabCol As IStandaloneTableCollection
Dim pStandaloneTable As IStandaloneTable
Dim intCount As Integer
Dim pTable As ITable
Set pStTabCol = pMap

cboTable.Clear

For intCount = 0 To pStTabCol.StandaloneTableCount - 1
  Set pStandaloneTable = pStTabCol.StandaloneTable(intCount)
  Dim pDataset As IDataset
  Set pDataset = pStandaloneTable
  cboTable.AddItem pStandaloneTable.Name
Next
```

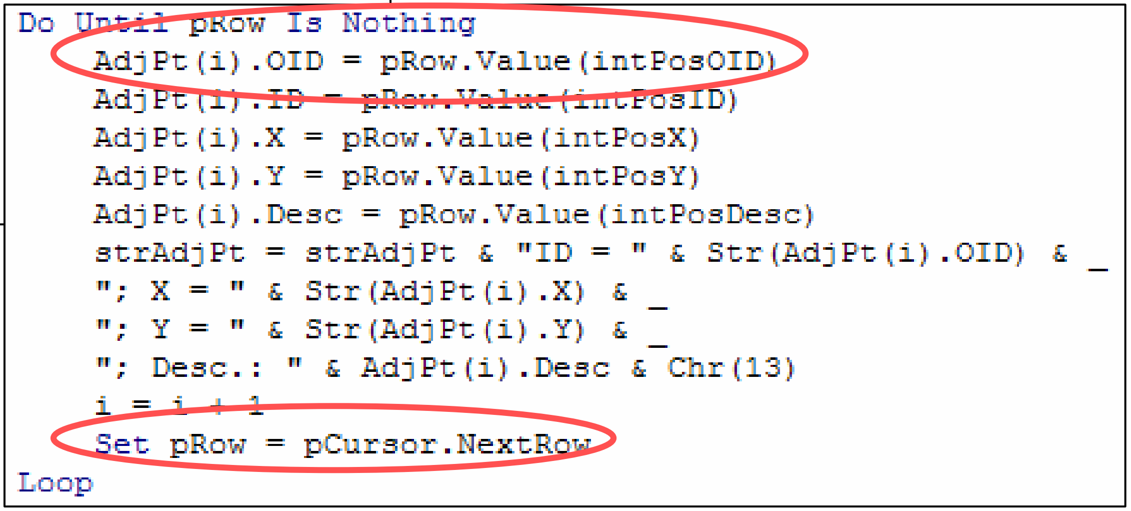
Put Information from Table to Array



```
Dim pStTabCol As IStandaloneTableCollection
Set pStTabCol = pMap

Dim intCount As Integer
intCount = cboTable.ListIndex

Dim pStandaloneTable As IStandaloneTable
Set pStandaloneTable = pStTabCol.StandaloneTable(intCount)

Dim pDataset As IDataset
Set pDataset = pStandaloneTable

Dim pTable As ITable
Set pTable = pStandaloneTable.Table

Dim pFields As IFields
Set pFields = pTable.Fields

Dim intPosOID As Integer
intPosOID = pFields.FindField("OID")
```

Use of ListIndex to access table.

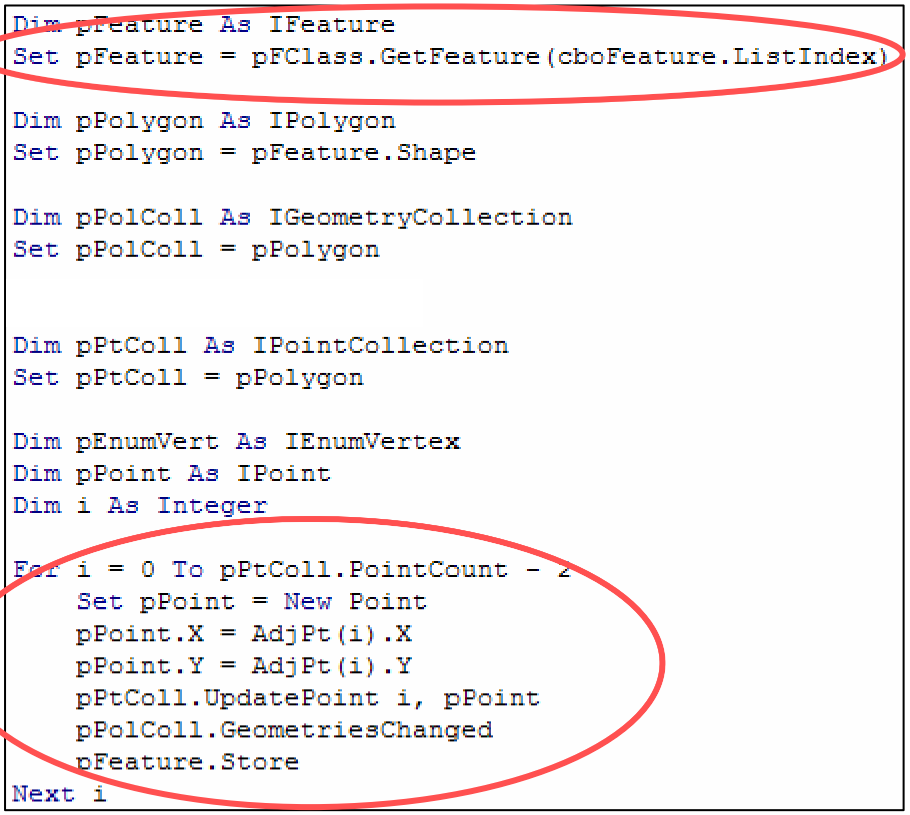
Find the position of the desired field, and the iCursor get the scroll through the rows to get the data.



```
Do Until pRow Is Nothing
    AdjPt(i).OID = pRow.Value(intPosOID)
    AdjPt(i).ID = pRow.Value(intPosID)
    AdjPt(i).X = pRow.Value(intPosX)
    AdjPt(i).Y = pRow.Value(intPosY)
    AdjPt(i).Desc = pRow.Value(intPosDesc)
    strAdjPt = strAdjPt & "ID = " & Str(AdjPt(i).OID) & _
    "; X = " & Str(AdjPt(i).X) & _
    "; Y = " & Str(AdjPt(i).Y) & _
    "; Desc.: " & AdjPt(i).Desc & Chr(13)
    i = i + 1
    Set pRow = pCursor.NextRow
Loop
```

Update the Polygon with New Coordinates

- To update the coordinates the setup is similar, except that now there is no need for iEnumVertex.
- Save the respective coordinate to the iPoint interface, and use UpdatePoint procedure on the ipointcollection interface to actualize the information.
- GeometriesChanged procedure in iPolygonCollection interface to tell the polygon it changed.
- Finally use the Store procedure on the iFeature interface to set to new coordinates.



```
Dim pFeature As IFeature
Set pFeature = pFClass.GetFeature(cboFeature.ListIndex)

Dim pPolygon As IPolygon
Set pPolygon = pFeature.Shape

Dim pPolColl As IGeometryCollection
Set pPolColl = pPolygon


Dim pPtColl As IPointCollection
Set pPtColl = pPolygon

Dim pEnumVert As IEnumVertex
Dim pPoint As IPoint
Dim i As Integer

For i = 0 To pPtColl.PointCount - 2
    Set pPoint = New Point
    pPoint.X = AdjPt(i).X
    pPoint.Y = AdjPt(i).Y
    pPtColl.UpdatePoint i, pPoint
    pPolColl.GeometriesChanged
    pFeature.Store
Next i
```

Create New Shapefile

```
Option Explicit

Public Function CreateShapefile(sPath As String, sName As String) As IFeatureClass ' Dont include .shp
extension

  ' Open the folder to contain the shapefile as a workspace
  Dim pFWS As IFeatureWorkspace
  Dim pWorkspaceFactory As IWorkspaceFactory
  Set pWorkspaceFactory = New ShapefileWorkspaceFactory
  Set pFWS = pWorkspaceFactory.OpenFromFile(sPath, 0)

  ' Set up a simple fields collection
  Dim pFields As IFields
  Dim pFieldsEdit As IFieldsEdit
  Set pFields = New Fields
  Set pFieldsEdit = pFields

  Dim pField As IField
  Dim pFieldEdit As IFieldEdit

  ' Make the shape field
  ' it will need a geometry definition, with a spatial reference
  Set pField = New Field
  Set pFieldEdit = pField
  pFieldEdit.Name = "Shape"
  pFieldEdit.Type = esriFieldTypeGeometry

  Dim pGeomDef As IGeometryDef
  Dim pGeomDefEdit As IGeometryDefEdit
  Set pGeomDef = New GeometryDef
  Set pGeomDefEdit = pGeomDef
  With pGeomDefEdit
    .GeometryType = esriGeometryPolygon
    Set .SpatialReference = New UnknownCoordinateSystem
  End With
  Set pFieldEdit.GeometryDef = pGeomDef
  pFieldsEdit.AddField pField

  ' Add another miscellaneous text field
  Set pField = New Field
  Set pFieldEdit = pField
  With pFieldEdit
      .Length = 30
      .Name = "MiscText"
      .Type = esriFieldTypeString
  End With
  pFieldsEdit.AddField pField

  ' Create the shapefile
  ' (some parameters apply to geodatabase options and can be defaulted as Nothing)
  Dim pFeatClass As IFeatureClass
  Set pFeatClass = pFWS.CreateFeatureClass(sName, pFields, Nothing, _
                                  Nothing, esriFTSimple, "Shape", "")

  Set CreateShapefile = pFeatClass
End Function
```

ArcObject Reference & Available Sample Codes Sites

- ArcGIS Developer Online: http://edndoc.esri.com/arcobjects/9.0
- EDN Code Exchange: http://edn.esri.com/index.cfm?fa=codeExch.gateway
- Getting to Know ArcObjects by Robert Burke
- Programming ArcObjects with VBA: A Task-Oriented Approach by Kang-Tsung Chang