CSE 1530

Introduction to Computer Use II: Programming

Winter 2006 (Section M)

Topic F: External Files and Databases – Using Classes and Objects

Monday, March 27 2006

Bill Kapralos

CSE 1530, Winter 2006, Bill Kapralos

Overview (1):

- **Before We Begin**
  - Some administrative details
  - Some questions to consider
- **Reading/Displaying the Contents of a File**
  - Overview
  - Opening a file
  - References
  - Adding a reference to a project
  - Working with references

Before We Begin

Administrative Details (1):

- **Exercise 7-6**
  - Due Monday, April 3 2006 before noon
- **Test Annulment Forms**
  - Now available from the Computer Science Engineering Undergrad Office located in CSEB 1003
    - Office hours → 10:00am – 12:00pm & 2:00-4:30pm
  - Must be completed if you wish to drop either of your test grades (Test 1 and/or Test 2)
- **Last Lecture is Monday, April 3**
  - Entire lecture will be review for exam

Some Questions to Consider (1):

- What is the Filter property and how is it used ?
- What is InitDir property and how is it used ?
- What is the Cancel Error property and how is it used ?
- What is the Flags property and how is it used ?

Reading & Displaying the Contents of a File

Overview (1):

- **We Now Know How to Obtain the File Name**
  - In the previous lectures we learned how we can specify a particular file (e.g., obtain the name of the file) using the common file dialog via the ShowOpen method
    - Of course specifying the file is only the beginning!
    - Once the file has been specified and "located" by Visual Basic, we must "open" the file in order to perform any of the required operations → in particular, read the contents of a file
  - We will now examine how to read the contents of a file

Opening a File (1):

- **Files Will be Opened Using Two "New" Classes**
  - We will create instances of these classes (e.g., objects), open the file and access its contents using the methods of the class via the objects
  - The two classes are
    - FileSystemObject
    - TextStream

As An Aside – References (1):

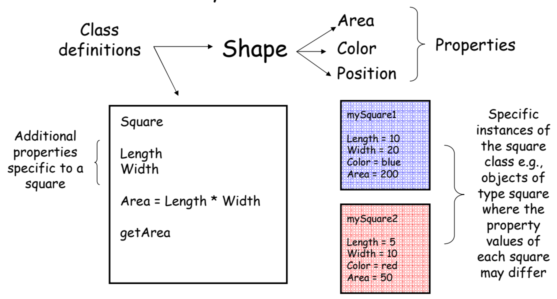
- **Until Now We Know That Objects are an Instance of a Class**
  - We know what a class and what an object is
  - We have worked with various objects
    - Control objects (text boxes, buttons, labels etc.)
    - We created these objects by simply choosing the desired one from the toolbox and placing it on the form → once on the form, any methods and properties of the object are available to us
    - But how do we "jump" from class to object → how are objects created (instantiated) ???

As An Aside – References (2):

- **Creating (Instantiating) an Object**
  - Recall
    - A class is nothing more than a definition → provides a definition of the properties/methods that are available to any object of that class
  - Cannot use the class definition on its own
    - We must have an object of the particular class
    - Once we create an object of a particular class we set aside space in the computer's memory for that object → space will be set aside to hold the values of all properties etc.

As An Aside – References (3):

- **Creating (Instantiating) an Object (cont.)**
  - Recall from early on in the course

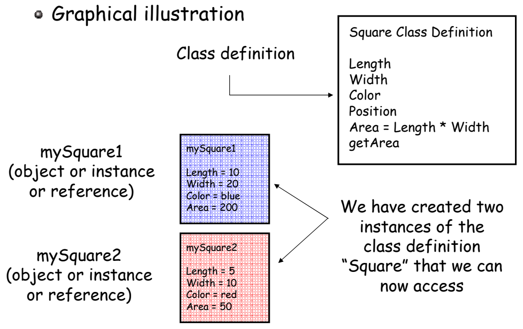


As An Aside – References (4):

- **Creating (Instantiating) an Object (cont.)**
  - Once we create (instantiate) an object of some class, memory for the object has been set aside and we can access the object (e.g., refer to the object)
    - Of course, we refer to the object by a name that we provide
    - We say that we have a reference to the object → the name we provide for the object lets us reference the properties and methods of the object

As An Aside – References (5):

- **Creating (Instantiating) an Object (cont.)**
  - Graphical illustration

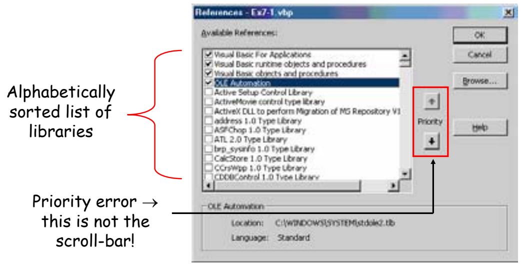


As An Aside – References (6):

- **Creating (Instantiating) an Object (cont.)**
  - As mentioned, in order to open and read the contents of a file, we will use the FileSystemObject and TextStream classes
    - This implies that we will have to instantiate an instance of each class → create an object of each class that we can use
    - These two classes are part of another library called the Scripting library
    - Just as we had to instruct Visual Basic to include the Common Dialog control, we have to do the same with the Scripting library

Adding a Reference to a Project (1):

- **Incorporating a New Reference**
  - From the Projects menu, choose References
    - The following window appears



☑ Visual Basic For Applications
☑ Visual Basic runtime objects and procedures
☑ Visual Basic objects and procedures
☑ OLE Automation
☐ Active Setup Control Library
☐ ActiveMovie control type library
☐ ActiveX DLL to perform Migration of MS Repository V1
☐ address 1.0 Type Library
☐ ASFChop 1.0 Type Library
☐ ATL 2.0 Type Library
☐ brp_sysinfo 1.0 Type Library
☐ CalcStore 1.0 Type Library
☐ CCrsWpp 1.0 Type Library
☐ CDDBControl 1.0 Type Library

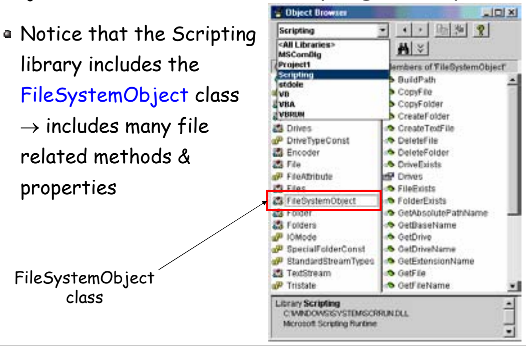
OLE Automation
Location: C:\WINDOWS\SYSTEM\stdole2.tlb
Language: Standard

Adding a Reference to a Project (2):

- **Incorporating a New Reference (cont.)**
  - Scroll down the alphabetic list of libraries and choose the Microsoft Scripting Runtime library
    - Check the box next to it and then click "Ok"
    - Do not un-check any other references that are already checked!
  - Unlike the Common Dialog component, no new control object icon will be added in the toolbox to indicate the addition of the Scripting library
    - You can of course obtain confirmation that it has been added through the Object Browser

Adding a Reference to a Project (3):

- **Object Browser and the Scripting Library**
  - Notice that the Scripting library includes the FileSystemObject class → includes many file related methods & properties



Adding a Reference to a Project (4):

- **Incorporating a New Reference (cont.)**
  - Once we have installed the Scripting library, essentially what we have done is to make available to our program, a large number of class definitions
    - By default, a Visual Basic project already has references to the collection of classes necessary to create forms and controls
    - But, there are many other class definitions available for a wide variety of "more complex" programs that must be explicitly added such as the file related class definitions we just added
    - We will now focus on creating instances of objects

Working With References (1):

- **Creating a FileSystemObject Object**
  - We want to create an object of this class so that we may be able to access the properties and methods it contains in order to work with our file
  - We will separate the process of creating an object into two steps
    1. Must specify a name for the object (e.g., a reference that will allow us to refer to it)
    2. Assign a "new" object to the reference → think of the object as the actual location in memory where space is set aside for the reference

Working With References (2):

- **Specifying a Name For the Reference**
  - This part is similar to any variable declaration except that the variable type (e.g., Integer, String, Single etc.) is replaced by a class type → familiar to you
    - However, you can think of a class as being a particular type as well
  - To specify an reference in general:
    - Dim referenceName As className
  - More specifically, to generate a FileSystemObject reference (that we will call "myFile")
    - Dim myFile As FileSystemObject

Working With References (3):

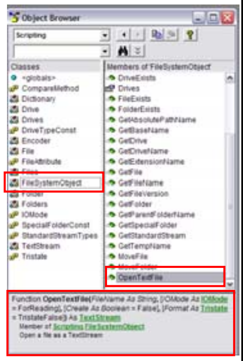
- **Assigning a New Object to the Reference**
  - Think of this as setting aside the actual memory space for the reference
  - To assign a new object to a reference
    - Set referenceName = New className
  - More specifically, to assign a FileSystemObject object (that we will call "myFile") to the "myFile" reference variable that we previously declared
    - Set myFile = New FileSystemObject
  - Note the key-words Set and New
    - Always used when new object is created

Working With References (4):

- **Using the Newly Created Object**
  - We now created an instance of the FileSystemObject class and can therefore access its many methods and properties
    - We must of course understand how to use the object!
  - How can we determine what methods it does contain and furthermore, how can we know how many and what type of arguments (if any) the methods require?
    - The answer lies in the familiar Object Browser!

Working With References (5):

- **Using the Newly Created Object (cont.)**
  - From the Object Browser, you can easily determine that the FileSystemObject contains a method called OpentextFile



Function **OpenTextFile**(*FileName As String*, [*IOMode As* IOMode = ForReading], [*Create As Boolean* = False], [*Format As* Tristate = TristateFalse]) As TextStream
Member of Scripting.FileSystemObject
Open a file as a TextStream

Working With References (6):

- **Using the Newly Created Object (cont.)**
  - Lets take a closer look at the description for the OpentextFile method

```
Function OpenTextFile(FileName As String, _
  [IOMode As IOMode = ForReading], _
  [Create As Boolean = False], _
  [Format As Tristate = TristateFalse]) _
  As TextStream
  Member of Scripting.FileSystemObject
  Open a file as a TextStream
```

Working With References (7):

- **Using the Newly Created Object (cont.)**
  - The description essentially tells you how to use the method
    - Provides a description of the arguments and their type → knowing how many and the type of each argument, you can call the method as you would call any other method!
    - The first argument is of type String and it denotes the name of the file you wish to open (e.g., the name of the file that was obtained using the Open File dialog)