I101 Introduction to Informatics

By:

Dr. Hossein Hakimzadeh
Computer Science and Informatics
IU South Bend

(c) Copyright 2007, H. Hakimzadeh
What do we need to learn in order to write computer programs?

- Fundamental programming constructs:
  - Variables,
  - Arithmetic operators,
  - Input and output
  - Conditionals,
  - Loops,
  - Procedures and functions,
  - Arrays (Multi-Dimensional Arrays),
  - Structures, More on classes and objects,
  - Files
Access Modifiers:

- Class properties and methods are preceded with ‘Access Modifiers’.

- Public:
  - Make the property or behavior accessible to the outside clients.

- Private:
  - Make the property or behavior usable by the class methods only.
Access Modifiers:

Public Class Student

Public Name As String
Public Email As String
Private Password As String

' set the private data of a given object
Public Sub setPassword(ByVal ThePassword As String)
    Password = ThePassword
End Sub

' return the private data from the object
Public Function getPassword() As String
    Return (Password)
End Function

End Class

' Declare a new object of type student
Dim MyStudent As New Student

' Initializing the public property of the object
MyStudent.Name = "Jack Clark"
MyStudent.Email = "jclark@iusb.edu"

' Incorrect method for setting the password
MyStudent.Password = "XYZ"

' Correct method to set the private parameter
MyStudent.setPassword("XYZ")
What is a Constructor?

- A constructor is a method that is called when a new object is instantiated. The job of the constructor is to setup the initial state of the object. (e.g. initialize the class variables etc.)

- Visual Basic will automatically create an empty constructor for each class. However the user can create her own constructor.

- The method name of the constructor is “New()”

- **Example:**

  ```vbs```
  ' Constructor for this class will be called whenever
  ' a new object is instantiated.

  Public Sub New()
      Name = "No Name"
      Email = "No Email"
      Password = "No Password"
  End Sub
  ```vbs```
Class with a user-defined Constructor:

Public Class Student

    Public Name As String
    Public Email As String
    Private Password As String

    ' set the private data of a given object
    Public Sub setPassword(ByVal ThePassword As String)
        Password = ThePassword
    End Sub

    ' return the private data from the object
    Public Function getPassword() As String
        Return (Password)
    End Function

    ' Constructor for this class will be called whenever
    ' a new object is instantiated.
    Public Sub New()
        Name = "No Name"
        Email = "No Email"
        Password = "No Password"
    End Sub

End Class
Polymorphism

- Polymorphism comes from the Greek: "having many shapes".
Polymorphism

- Two methods for achieving Polymorphism:
  - Overloading
  - Overriding
Overloading

- Ability to take many shapes or forms.

- In Object Oriented Programming, polymorphism refers to methods that have identical names but have different implementations depending on their parameters.

- A single class can have more than one method with the same name. This is called OVERLOADING.
Example of Overloading

The example below shows two methods for setting the password property of the student class.

'set the “Password” to a value passed to the method
Public Sub setPassword(ByVal ThePassword As String)
    Password = ThePassword
End Sub

'set the “Password” to a default value (No parameters!!)
Public Sub setPassword()
    Password = "Default"
End Sub
Complete Example:

Public Class Student

    Public Name As String
    Public Email As String
    Private Password As String

    '----------------------------------------------
    Public Sub setPassword(ByVal ThePassword As String)
        Password = ThePassword
    End Sub

    Public Sub setPassword()
        Password = "Default"
    End Sub

    'return the private data from the object
    Public Function getPassword() As String
        Return (Password)
    End Function

    '----------------------------------------------
    Public Sub New()
        Name = "No Name"
        Email = "No Email"
        Password = "No Password"
    End Sub

    Public Sub New(ByVal TheName As String, ByVal TheEmail As String)
        Name = TheName
        Email = TheEmail
        Password = "*******"
    End Sub

End Class

Overloaded setPassword() methods.

Overloaded constructors.

New() methods
Instantiating and Using Overloaded Methods:

Public Class Student

    Public Name As String
    Public Email As String
    Private Password As String

    '----------------------------------------------
    Public Sub setPassword(ByVal ThePassword As String)
        Password = ThePassword
    End Sub
    Public Sub setPassword()
        Password = "Default"
    End Sub
    'return the private data from the object
    Public Function getPassword() As String
        Return (Password)
    End Function
    '----------------------------------------------
    Public Sub New()
        Name = "No Name"
        Email = "No Email"
        Password = "No Password"
    End Sub
    Public Sub New(ByVal TheName As String, ByVal TheEmail As String)
        Name = TheName
        Email = TheEmail
        Password = "*******"
    End Sub

End Class

Dim Another_Student As New Student("Jack Clark", "jclark@iusb.edu")
Dim A_Student As New Student
MyStudent.setPassword("*****")
MyStudent.setPassword()
Overriding

- Overriding is similar to Overloading, however overriding applies in the inherited classes.

- Refers to methods that have the same name as a method in its BASE CLASS.

- The method in the subclass or derived class takes precedence, or overrides the identically named method in the base class.
Example of Overriding

Base Class
Student

Derived Class
GradStudent
Example of Overriding

Base Class

Student

Public **Overridable** Sub PrintName()
    Console.WriteLine("Student: {0,-20}", Name)
End Sub

Derived Class

GradStudent

**Overrides** Sub PrintName()
    Console.WriteLine("GradStudent: {0,20}", Name)
End Sub
Complete Example:

Public Class Student
    Public Name As String
    Public Email As String
    Private Password As String

    ' set the private data of a given object
    Public Sub setPassword(ByVal ThePassword As String)
        Password = ThePassword
    End Sub

    ' Overloaded Method, set the private to a default value
    Public Sub setPassword()
        Password = "Default"
    End Sub

    ' return the private data from the object
    Public Function getPassword() As String
        Return (Password)
    End Function

    ' Constructor for this class will be called whenever
    ' a new object is instantiated.
    Public Sub New()
        Name = "No Name"
        Email = "No Email"
        Password = "No Password"
    End Sub

    ' Overloaded Constructor
    Public Sub New(ByVal TheName As String, ByVal TheEmail As String)
        Name = TheName
        Email = TheEmail
        Password = "*******"
    End Sub

    ' To be overridden by a methods in a derived class
    Public Overridable Sub PrintName()
        Console.WriteLine("Student: {0,-20}", Name)
    End Sub
End Class

PrintName() is an Overridable method.
Complete Example:

```csharp
Public Class GradStudent
    Inherits Student

    Public ThesisTopic As String
    Public ThesisAdvisor As String

    Public Sub New()
        MyBase.New()
        ThesisTopic = "Unknown"
        ThesisAdvisor = "Graduate Director"
    End Sub

    Public Sub New(ByVal TheName As String, ByVal TheEmail As String,
                    ByVal TheTopic As String, ByVal TheAdvisor As String)
        MyBase.New(TheName, TheEmail)
        ThesisTopic = TheTopic
        ThesisAdvisor = TheAdvisor
    End Sub

    ' Override the PrintName() method in the base class
    Overrides Sub PrintName()
        Console.WriteLine("GradStudent: {0,20}", Name)
    End Sub

End Class
```

Overrides the PrintName() method of its base class.
**Instantiating and Using Overridden Methods:**

- Create a new graduate student object:
  
  Dim A_Graduate_Student As New GradStudent

- Call the PrintName() method:
  
  A_Graduate_Student.PrintName()

- Which PrintName() method gets called? The one in the base class or the on the subclass?

- What if the subclass did not override the PrintName() method? Which method would get the call?