LAB GOALS

To learn about object oriented programming:

- To create and use a class.
- Creating a class Constructors
- Overloading a Constructor
- Public vs. Private property and methods of a class

Step 1: To create a new Class called Student, first, create a new console application and type the following code in it:

```vbnet
Option Strict On
Option Explicit On

Module Module1

Sub Main()
    Console.ReadLine() 'pause the program
End Sub

End Module
```

Now, type the following code just below the line “Option Explicit On”:

```vbnet
Public Class Student
    Public Name As String
    Public Email As String
    Private Password As String 'Note that the Password is declared as a private property (variable).

    ' Constructor for this class will be called whenever
    ' a new object is instantiated (created).
    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

    Public Sub printStudent()
        Console.WriteLine("{0, -20}{1, -30}{2, -20}", Name, Email, Password)
    End Sub

End Class
```

Run: Compile and Run your program to make sure it does not have any syntax errors.

Step 2: Using the new Student class in a program. Type the following two lines in the main procedure (above the Console.ReadLine() statement).

```vbnet
Dim Student1 As New Student
Student1.printStudent()

Console.ReadLine() 'pause the program
```

Run: Compile and Run your program. Notice what is printed and see if you can figure out why.
Step 3:  Now, type the following two lines in the main procedure (once again above the Console.ReadLine() statement).

```
Dim Student2 As New Student("Mary", "mary@iusb.edu")
Student2.printStudent()
```

Run:  Compile and Run your program. Notice what is printed and see if you can figure out why.  What is the difference between Step 2 and Step 3?

Step 4:  To better understand the difference between step 2 and step 3.  Type the following 2 lines in the two New() constructors of this class.

```
' Constructor for this class will be called whenever
' a new object is instantiated (created).
Public Sub New()
    Console.WriteLine("Constructor: without any parameters is called")
    Name = "No Name"
    Email = "No Email"
    Password = "No Password"
End Sub

' Overloaded Constructor
Public Sub New(ByVal TheName As String, ByVal TheEmail As String)
    Console.WriteLine("Constructor: with two parameters is called")
    Name = TheName
    Email = TheEmail
    Password = "*******"
End Sub
```

Run:  Compile and Run your program. See if you now understand the difference between Step 2 and Step 3?  If not, ask me.

Step 5:  Ok, let’s do a little more. Type the following code in the main() sub procedure. (above the Console.ReadLine() statement)

```
Student1.Name = "Tom"
Student1.Email = "Tom@gmail.com"
```

Run:  Compile and Run your program.

Now type the following line of code in the main() sub procedure. (above the Console.ReadLine() statement)

```
Student1.Password = "XYZ"
```

Run:  Compile your program.  Any problems?  If so, do you know why?

Step 6:  As you noticed in the above step, you can’t seem to print the Password property of the student. This is because the password is declared as a private variable and it is only visible inside the class. To fix this problem we have two choices:

a)  Change the declaration and make the password public, so that others outside the class can have access to it.
b)  Create a procedure (inside the class) that can be called by outsiders to set password for the student.

Let’s try option (b). Let’s create a procedure called SetPassword() which takes an argument such as “XYZ” and copies it into the password property of the student.  Add the following procedure at the end of the Student class.  (right before the “End Class” line.

```
Public Sub setPassword(ByRef ThePassword As String)
    Password = ThePassword
End Sub
```
Now type the following lines of code in the main() sub procedure. (above the Console.ReadLine() statement)

```csharp
Student1.setPassword("XYZ")
Student1.printStudent()
```

Run: Compile and Run your program. You should see the following output.

```
Constructor: without any parameters is called
No Name       No Email                        No Password
Constructor: with two parameters is called
Mary          mary@iusb.edu                     ******
Tom           Tom@gmail.com                    XYZ
```

```csharp
```
Here is the complete Program.

Option Strict On
Option Explicit On

Public Class Student
    Public Name As String
    Public Email As String
    Private Password As String  'Note that the Password is declared as a private property (variable).

    ' Constructor for this class will be called whenever
    ' a new object is instantiated (created).
    Public Sub New()
        Console.WriteLine("Constructor: without any parameters is called")
        Name = "No Name"
        Email = "No Email"
        Password = "No Password"
    End Sub

    ' Overloaded Constructor
    Public Sub New(ByVal TheName As String, ByVal TheEmail As String)
        Console.WriteLine("Constructor: with two parameters is called")
        Name = TheName
        Email = TheEmail
        Password = "*******"
    End Sub

    Public Sub printStudent()
        Console.WriteLine("{0, -20}{1, -30}{2, -20}", Name, Email, Password)
    End Sub

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

End Class

Module Module1
    Sub Main()
        Dim Student1 As New Student
        Student1.printStudent()

        Dim Student2 As New Student("Mary", "mary@iusb.edu")
        Student2.printStudent()

        Student1.Name = "Tom"
        Student1.Email = "Tom@gmail.com"

        'Student1.Password = "XYZ"    'Note that this line causes a syntax error

        Student1.setPassword("XYZ")
        Student1.printStudent()

        Console.ReadLine()   'pause the program
    End Sub
End Module