Introduction to objects

The entire .NET framework is based on objects. Every ASP.NET page written has actually made use of objects.

Learning object oriented programming (OOP) and object oriented design (OOD) is not an easy task. However once understood it is a vital skill for a software developer. It is necessary for all OOP languages such as VB.NET, C++ and Java.

To create an application built from objects you need to learn how to create objects (OOP) and learn which objects are needed to be created (OOD). Deciding which objects to create and how they will interact is the hard part. We will be using UML (unified modelling language) to help us design our object oriented applications.

This section introduces objects and classes. We will create a class and test it

Terms

Class – Template for creating an object. Defines the properties, methods and events.

Property – An attribute of an object

Method – An action an object can implement

Event – An action an object implements in response to something

Member – Properties, methods and events of a class

Public – Accessible from outside the file

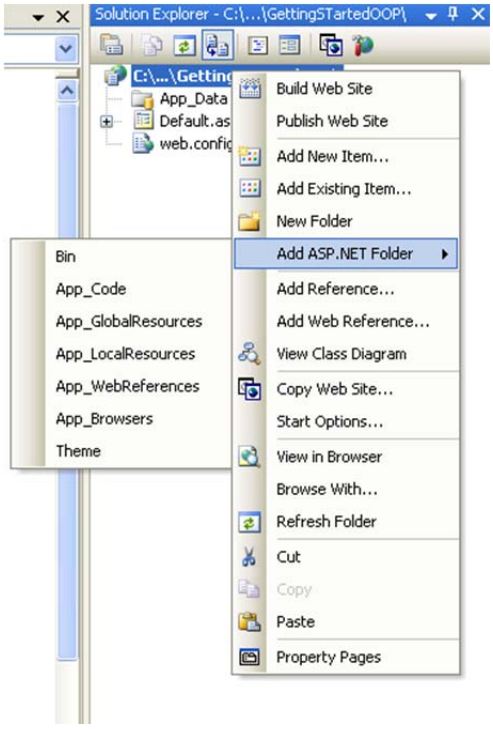
Private – Accessible only from within the file

Protected - Members which are only accessible by the immediate children of a class

Activity 1. Defining a class

A class is the template or blue print of an object. It defines the characteristics of the object and the actions the object can take.

All the classes in your web applications need to be created in the App_code folder. To add a new App_code folder right click on the web application name in the solution explorer and select “Add ASP.NET Folder” from the sub menu select “App_code”.



This creates a new folder called App_code. Right click on this new folder and select “Add new item” This will open the Add New Item window. We are creating a new class.

Give the class a name of “Student.cs”. By convention always begin a class’s name identifier with a capital letter and start each subsequent word in the identifier with a capital letter. The following code will be generated for you:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

/// <summary>
/// Summary description for Student1
/// </summary>
public class Student1
{
	public Student1()
	{
		//
		// TODO: Add constructor logic here
		//
	}
}
```

This creates a class called student.

First few lines imports the namespaces listed. This allows us to have access to the classes in these namespaces.

A comment is then included this is where you should place a description of the class.

After the comment we have the class declaration the word public identifies the access of this class we will learn latter other options. Class specifies that we are creating a new class. Student is the name of the class.

The class declaration needs to be surrounded by opening and closing braces { }.

Inside the class we have a constructor defined. The code for the constructor will be placed here. (We will learn about this later).

We will now add some variables to the class:

```
…
public class Student
{
	public string strFirstName;
	public string strLastName;
	public string strStudentNumber;

	public Student()
	{
		//
		// TODO: Add constructor logic here
		//
	}
}
```

This adds 3 variables.

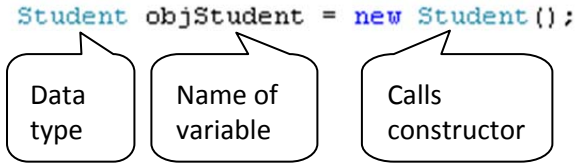
A class is made up of just C# code there is “front end” or visual part to a class. A user cannot directly interact with it. It has no user interface. For this reason when we test the code in a class we need to test it by creating a web page which will use the class (interact with it).

To test the class create the following web page:

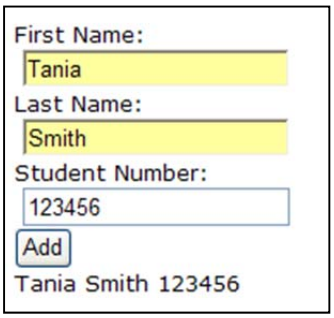
Add the following code to the click event handler:

```
protected void btnAdd_Click(object sender, EventArgs e)
    {
        //create a new instance of the student class
        Student objStudent = new Student();

        //allocate the values
        objStudent.strFirstName = txtFirstName.Text;
        objStudent.strLastName = txtLastName.Text;
        objStudent.strStudentNumber = txtStudentNumber.Text;

        //display the values to the label
        lblMessage.Text = objStudent.strFirstName + " " + objStudent.strLastName + " " +
objStudent.strStudentNumber;
    }
```

In this code we create an instance of the Student class. The file Student.cs declares how the student class will be (the blue print or template) a variable, objStudent is created, which will be of type Student (our user defined class) and it calls the constructor to create the instance. objStudent is now an instance of the Student class or an object.



The output should look something like this:



Activity 2. Creating classes

Q1. Create a car class provide variables to store the make, colour, year, model. Test this class by creating a web form which collects the values from textboxes.

References

C# for programmers second edition by Harvey M. Deitel and Paul J. deitel