Lab 1: Introduction to C# GUI

The intent of Lab 1 is to have you create a simple Windows GUI application with C# code.

Create a Visual Studio Solution

Visual Studio stores all applications in solutions. Contained inside of solutions are one or more projects. Both Solutions and Projects represent folders on the hard drive.

Note: Only one solution can be open in Visual Studio at one time. To work with multiple solutions, you simply need to open up multiple instances of Visual Studio.

Activity 1

New Project Dialog
Create a New Project called WindowsUI in Solution Lab1

1. From the File menu, choose New > Project
2. From the New Project dialog box, choose Visual C# Windows Forms Application
3. Enter WindowsUI as the project name.
4. Check Create directory for solution
5. Click OK

Read and explain the contents of C# Classes

C# is an object-oriented language very similar to Java. Classes are the fundamental organizing blocks for code. Projects typically contain one to many C# classes.

Files that contain C# classes use the .cs extension.

Visual Studio created two files that contain C# classes for the WindowsUI project

- Program.cs
- Form1.cs

To view the Program.cs class code, from Solution Explorer right-click Program.cs and select View Code
namespace WindowsUI {
    static class Program {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

Notes:

- The **Main** method (note the capital M) is always the starting point in a C# windows application.
- **Application** is a class defined in the .net framework.
- **EnableVisualStyles()** is a static method allow you to later set visual styles with your .net controls.
- **SetCompatibleTextRenderingDefault(false)** uses the newer GDI+ for rendering text in Forms.
- **Application.Run(new Form1())** displays the window generated by calling the constructor method for the **Form1** class, passing no-arguments to the constructor method.

---

**Understand the role of Window Forms in a GUI applications**

Graphical user interface **windows** are known are called **Forms** in .NET and are implemented as derived classes from the **System.Windows.Forms.Form** class.

Visual Studio added class **Form1** as the startup Form when you created your WindowsUI project.

From the Solution Explorer, right-click **Form1.cs** and select **View Code**.
8. using System.Windows.Forms;
9. namespace WindowsUI
10. {
11.     public partial class Form1 : Form
12.     {
13.         public Form1 ()
14.         {
15.             InitializeComponent();
16.         }
17.     }
18. }
19. }

Form1 is a public partial class that inherits from class Form. It is referenced by the namespace WindowsUI so its fully qualified name is WindowsUI.Form1.

By default, before the class definition, Visual Studio adds a number of using statements that allow us to reference classes (for example Form) without including the namespace prefix (System.Windows.Forms.Form).

At this point, class Form1 consists of a constructor method which is passed no arguments. Inside the constructor method is a call to method InitializeComponent(), which is defined in file Form1.Designer.cs. The InitializeComponent method contains C# code automatically generated by Visual Studio. You should avoid changing its contents because Visual Studio may change it back on you. Because some of the code for class Form1 is contained in an additional file, the class is labeled partial.

Add graphical components to Forms

Controls are graphical user interface widgets such as buttons, scroll bars, and menus that are displayed inside of Forms.
Simple Calculator using GUI

Use Visual Studio's refactoring tools, modify and add to the code to implement a simple two operand adding calculator.

Use **five** components:

1. TextBox operand1TextBox
2. TextBox operand2TextBox
3. Label plusLabel
4. Button calculateButton
5. Label resultLabel

**Scenario:** The user types into two text boxes two integer numbers, clicks Add-em Up, and the result are displayed in the label below the button.

To simplify the assignment, the code for the `calculateButton_Click` event handler is given below:

```csharp
private void calculateButton_Click (object sender, System.EventArgs e)
{
    try
    {
        int result = Convert.ToInt32(operand1TextBox.Text) + Convert.ToInt32(operand2TextBox.Text);
        resultLabel.Text = result.ToString();
    }
    catch (FormatException ex)
    {
        resultLabel.Text = "Error";
    }
}
```

Duplicate the snapshots shown below: