Course: EGDF20

Module: EGE202 Application Programming

Practical 1: Familiarization with WinForm Application Development

Objectives: At the end of this lab, the student should be familiarized with some of the basic C# syntaxes and OOP concepts used in WinForm development. Student will also be introduced to Microsoft Docs documentation.

Part 1: Understanding Visual Studio IDE (Integrated Development Environment)

1. Reopen the previous ***HelloWorld*** solution from SDL 1.
2. Under the ***File*** menu, click ***Open->Project/Solution***. Then navigate to the folder where the previous HelloWorld project is located. Choose the file “HelloWorld.sln” (Microsoft Visual Studio Solution).

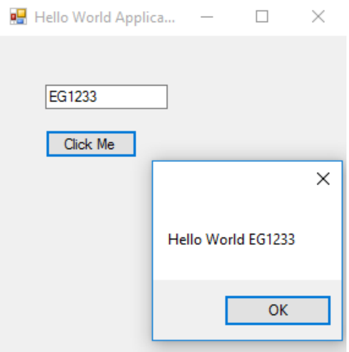
No	Actions	Observation
1	Run the program using <F5> key or from the menu ***Debug->Start Debugging***. Type in your name in the Textbox and click the “Click Me” Button	
2	What is the equivalent of `Console.WriteLine(str)` In Python Programming?	
3	Run the program using <Ctrl + F5> key or from the menu ***Debug->Start without Debugging***. Type in your name in the Textbox and click the “Click Me” Button	

Analyze and explain the C# codes for *button1_Click (…)*

```
private void button1_Click(object sender, EventArgs e)
{
    string str;
    str = txtInput.Text;
    str = "Hello World " + str;
    Console.WriteLine(str);
}
```

No	Actions	Observation / Action
1	`string str;`	
	`str = txtInput.Text;`	
	`str = "Hello World " + str;`	
	`Console.WriteLine(str);`	
2	**<u>Coding Task:</u>** Modify the codes to add an additional exclamation mark ! when printed on the Output window **Hello World EGE202!**	
3	**<u>Coding Task:</u>** The current codes for button1_Click (…) requires 4 line of codes. Optimize and modify your code to just 1 line! **Note:** In programming we need to balance between concise and readability of codes.	

3. Next, we will improve the application to use a **MessageBox** object instead of **Console** object. At the code editor for "Form1.cs" modify *button1_Click (…)* to the following codes:

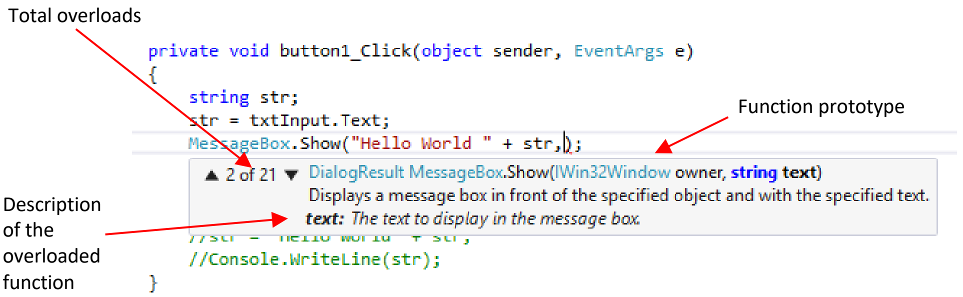


```
private void button1_Click(object sender, EventArgs e)
{
    string str;
    str = txtInput.Text;
    MessageBox.Show("Hello World " + str);
}
```

4. Build and debug your application by hitting <F5> key or from the menu ***Debug->Start Debugging***. Type in some words in the Textbox and click the "Click Me" Button. **Will you see the same observation with** <Ctrl + F5> key or from the menu ***Debug->Start without Debugging***
5. **Form**, **Button**, **TextBox**, **Console** and **MessageBox** are examples of objects defined in .NET class library. These objects have attributes (*TextBox.**Text***, *Button.**Name*** …) and methods (*MessageBox.**Show ()*** …). These attributes (variables in C) and methods (functions in C) enable programmers to interact with the objects.

Most methods (functions) in .NET class library implements "function overloading". Function overloading or method overloading is a programming concept that allows definition of two or more functions with the same name. In step 4, we used one of the overloaded function *MessageBox.Show("Hello World " + str)* that will print out the message on a popup window call **MessageBox**. This function only accepts one input parameter. Alternatively we can use another overloaded function which takes in two input parameter, *MessageBox.Show("Hello World" + str, "My Output")*

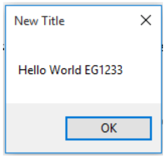
6. At the code editor for "Form1.cs", search for the *button1_Click (…).* At the line showing *MessageBox.Show ("Hello World " + str)*, use the mouse and place the cursor right before the closing ')' and type in a comma (",") symbol. A pop-up window appear showing the available overloaded function. This is known as the Visual Studio Intellisense feature. Intellisense will show the total number of overloaded functions for *Show(…)*, a brief description of the function and the function prototype.



```
private void button1_Click(object sender, EventArgs e)
{
    string str;
    str = txtInput.Text;
    MessageBox.Show("Hello World " + str,);
    //str = "Hello World " + str;
    //Console.WriteLine(str);
}
```

▲ 2 of 21 ▼ DialogResult MessageBox.Show(IWin32Window owner, **string** **text**)
Displays a message box in front of the specified object and with the specified text.
text: *The text to display in the message box.*

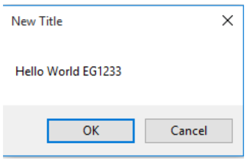
7. Modify the *button1_Click (…).* as follows:


New Title

Hello World EG1233

```
private void button1_Click(object sender, EventArgs e)
{
    string str;
    str = txtInput.Text;
    MessageBox.Show("Hello World " + str,"New Title");
}
```

8. Build and test your application by hitting <F5> key or from the menu ***Debug->Start Debugging*** or by clicking the Start button

No	Actions	Observation / Action
1	Run the program using <F5> key or from the menu ***Debug->Start Debugging***. What are the changes to the MessageBox?	
2	**Coding Task:** Based on the step 7, look for the function overload that enables the **MessageBox** to have both *OK* and *Cancel* buttons. Implement the required codes.	 New Title Hello World EG1233
3	Put the mouse cursor on the *Show(…)* method. Press the <F1> key.	