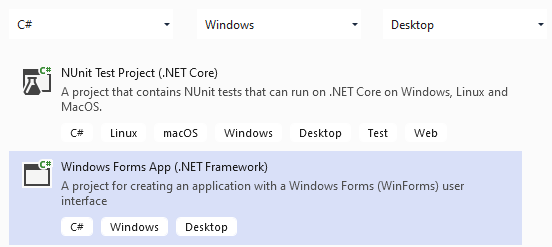
CPSC1012 – Windows Forms Demo

Introduction

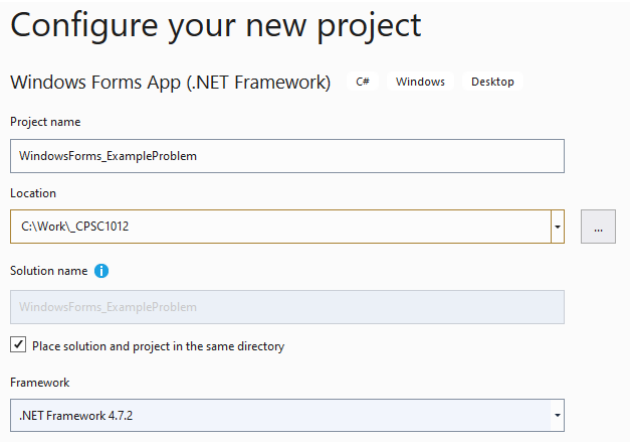
This demo is not part of the content for CPSC1012 and is provided here as a segue to more advanced programming courses. The core concepts taught in CPSC1012 are pure console applications, but those types of applications are very limited. One of the issues we had to overcome was to continue to change user input to perform the calculations many times; we did this using looping structures. A Windows Forms application can bypass the use of looping in many instances.

Creating the Project

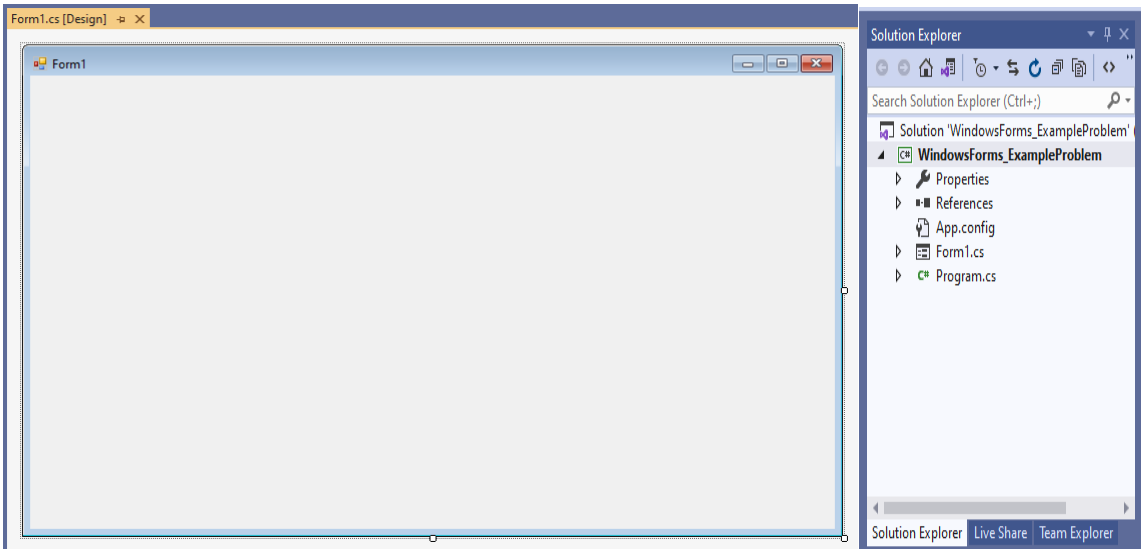
1. Select the correct project type. This will be:



2. Configure your project:

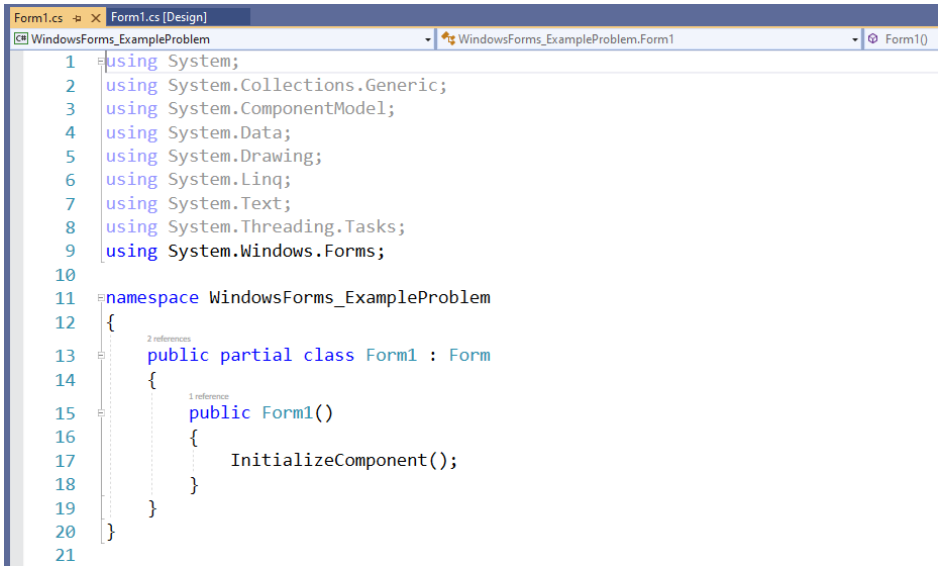


3. Once created, you should see something like:



Components of the Form

Notice that there is a new file, **Form1.cs**, in the Solution Explorer. This is the code for the form. This file is two parts (although not shown in the Solution Explorer): one part is the design code, and the other is the code for functionality. To see both, you will need to show the code. In the upper right of the Solution Explorer window click on the **< >** to reveal the code:



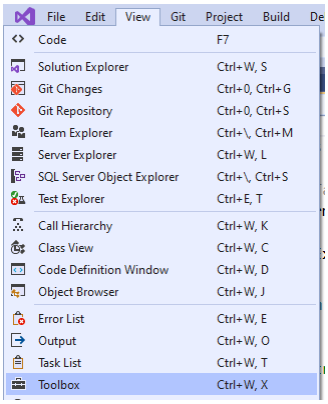
Another thing to make note of is the code for **Program.cs**:

```
Program.cs  Form1.cs  Form1.cs [Design]
WindowsForms_ExampleProblem                WindowsForms_ExampleProblem.Program
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Threading.Tasks;
 5  using System.Windows.Forms;
 6
 7  namespace WindowsForms_ExampleProblem
 8  {
 9      static class Program
10      {
11          /// <summary>
12          /// The main entry point for the application.
13          /// </summary>
14          [STAThread]
15          static void Main()
16          {
17              Application.EnableVisualStyles();
18              Application.SetCompatibleTextRenderingDefault(false);
19              Application.Run(new Form1());
20          }
21      }
22  }
23
```

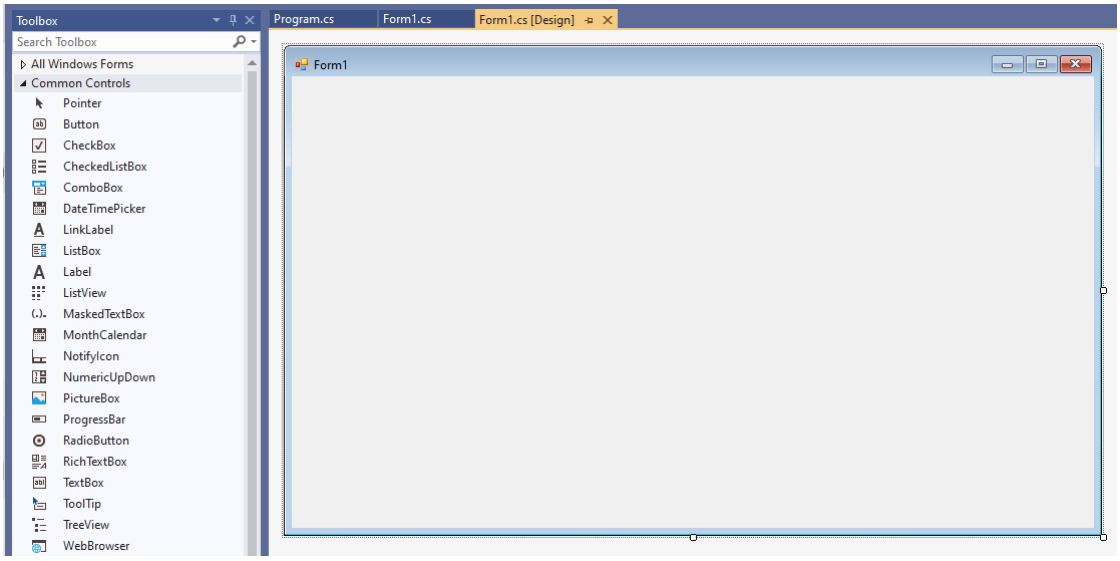
You will **NOT** modify this code!

Design the Form

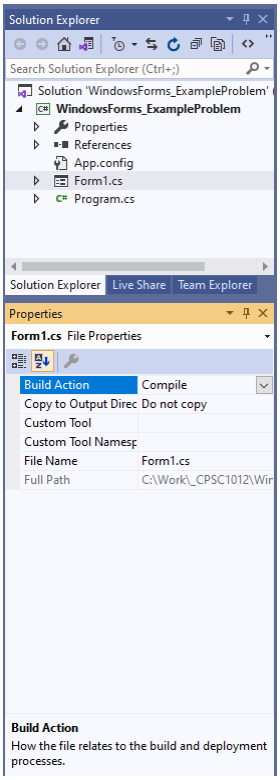
1. As this is a Graphical User Interface (GUI), you will need the graphical tools to add the form elements to the form:



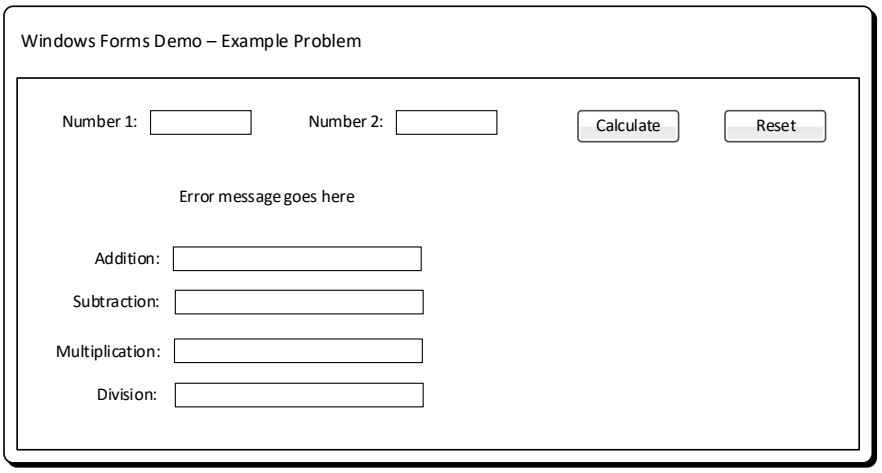
2. Set up your work environment to be:



3. You will also need to make sure the **Properties** window is visible and expanded to see the properties you will need to set for the form controls:

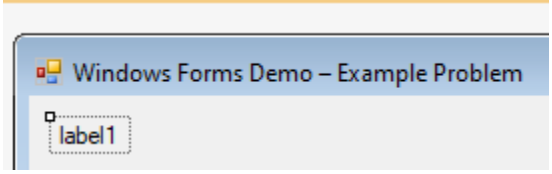


4. The form you need to design is to look like:

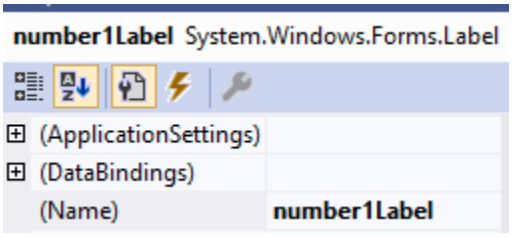


5. Change the form title from **Form1** to **Windows Forms Demo – Example Problem**:
    a. Click on the form window in the designer
    b. Go to the Properties tab and set the properties to be alphabetical (second icon at the top of the Properties tab
    c. Scroll down to fine the **Text** property and change the value in the right column

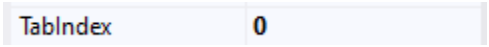
6. Add the first **Label** control (Number 1:):
    a. From the Toolbox, drag a **Label** control to the form



    b. Change the **Text** property to **Number 1:**
    c. Change the **(Name)** property to **number1Label**



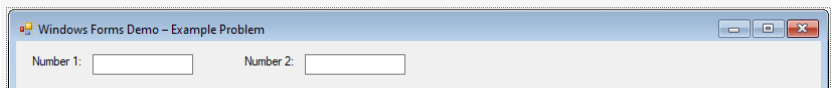
    d. Make sure the label's **Tabindex** is **0**



7. Add the first **TextBox** control:
    a. Drag a **TextBox** control from the **Toolbox** and place it to the right of the label you added
    b. Change the **(Name)** to **number1Textbox**

8. Repeat steps 6 and 7 to add the other controls:
    a. Label text, **Number 2:**, and name to **number2Label**
    b. TextBox name: **number2Textbox**

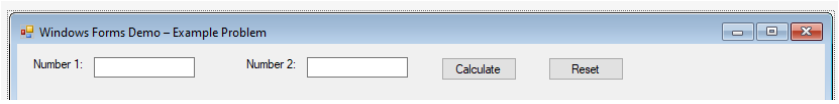
You should now have the form looking like:



9. Add the first **Button** control (**Calculate)**:
    a. Drag a **Button** control and place it to the right of the last TextBox
    b. Change the **Text** property to **Calculate**
    c. Change the **(Name)** to **calculateButton**

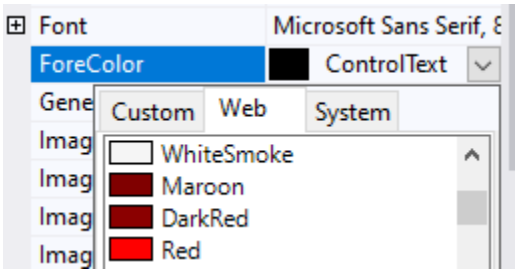
10. Add the second **Button** control (**Reset**) using step 9 as a sample:
    a. Text property is **Reset**
    b. Name property is **resetButton**

    Your form should now look like:

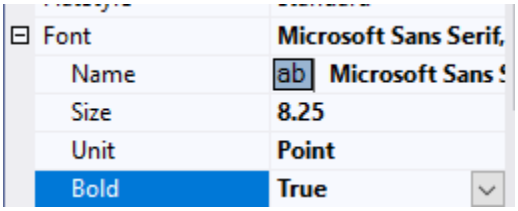


11. You now need to make sure the **Tabindex** properties of the controls are sequential from **0** to **5**.

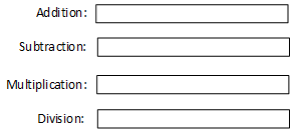
12. Now add a **Label** control that will be used to display any error messages:
    a. Text property is **error**
    b. Font colour (ForeColor) is red (select the red colour from the Web colours):



    c. Name is **errorLabel**
    d. Optionally, you can make the label bold



13. Below the error label add the web controls to match the design:



    a. **Addition**:
        i. **Label**: Name is **addLabel**

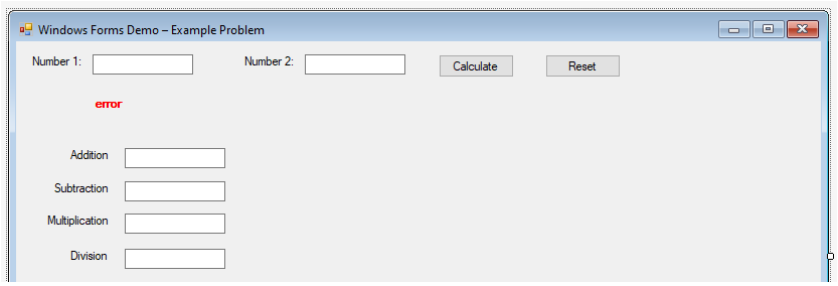
ii. **TextBox**: Name is **addTextbox**

b. **Subtraction**:
   i. **Label**: Name is **subtractLabel**
   ii. **TextBox**: Name is **subtractTextbox**

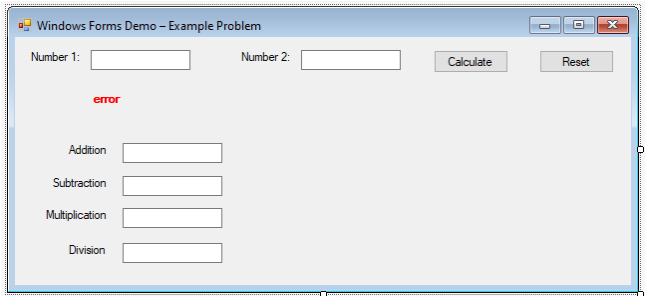
c. **Multiplication**:
   i. **Label**: Name is **multiplyLabel**
   ii. **TextBox**: Name is **multiplyTextbox**

d. **Division**:
   i. **Label**: Name is **divideLabel**
   ii. **TextBox**: Name is **divideTextbox**

Your design should now look like:



14. Now that all the controls are on the form, you need to resize the form by dragging the form box control (lower right corner of the form) to get the size correct:



15. Once again, it is prudent that you check the **Tabindex** of the controls to make sure they are in the correct sequence. Alternatively, you can run your form and tab through each TextBox and Button control to verify the sequence is correct.

Code the Form

1. Open the **Form1.cs** code file and place the following code after the **`InitializeComponent();`** line:

```
errorLabel.Text = "";
```

This will make sure the word **`error`** does not appear when the program is first run.

2. You will need a way to validate that the user enters a number, and not any other character(s), in the TextBox controls. For this it is best to have a method that validates if the string value in the TextBox can be converted to a `double`. Use the code below:

```
#region Input Validations
public static bool IsDouble(string inputString)
{
    bool isValid;
    double temp;
    try
    {
        temp = double.Parse(inputString);
        isValid = true;
    }
    catch (Exception)
    {
        isValid = false;
    }
    return isValid;
}//end of IsDouble
#endregion
```

Notice that this code is different from our **`GetSafeDouble(string prompt)`** method but its function is to determine if the string can be converted to a `double` value.

3. The code for the **Calculate** button is done by:
    a. Switch to the **Form1.cs [Design]** view
    b. Double-click the **Calculate** button; this will generate an event method in the **Form1.cs** code file
    c. Switch to the **Form1.cs** code file to see the event method stub

```
private void calculateButton_Click(object sender, EventArgs e)
{

}
```

The details of how this works is beyond the scope of this demo. Suffice it to say that when the user clicks the button, this event method will be called.

    d. Add the following code to this method:

```
double number1,
    number2,
    addition,
    subtraction,
    product,
    quotient;
if (IsDouble(number1Textbox.Text.Trim()))
{
    if (IsDouble(number2Textbox.Text.Trim()))
    {
```

```
            number1 = double.Parse(number1Textbox.Text.Trim());
            number2 = double.Parse(number2Textbox.Text.Trim());
            addition = number1 + number2;
            subtraction = number1 - number2;
            product = number1 * number2;
            quotient = number1 / number2;
            //put these values in the appropriate textboxes
            addTextbox.Text = addition.ToString();
            subtractTextbox.Text = subtraction.ToString();
            multiplyTextbox.Text = product.ToString();
            divideTextbox.Text = quotient.ToString();
            errorLabel.Text = "";
        }
        else
        {
            errorLabel.Text = "Invalid Number 2.";
        }
    }
    else
    {
        errorLabel.Text = "Invalid Number 1.";
    }
```

4. Now to code the **Reset** button. The **Reset** button will simply clear the form. Do this by:
    a. Switch to the **Form1.cs [Design]** view
    b. Double-click the **Reset** button
    c. In the event method stub created, add the following code:

```
number1Textbox.Text = "";
number2Textbox.Text = "";
addTextbox.Text = "";
subtractTextbox.Text = "";
multiplyTextbox.Text = "";
divideTextbox.Text = "";
errorLabel.Text = "";
```

Test the Form

Now just run the form and see how it works. Debug and fix any errors.