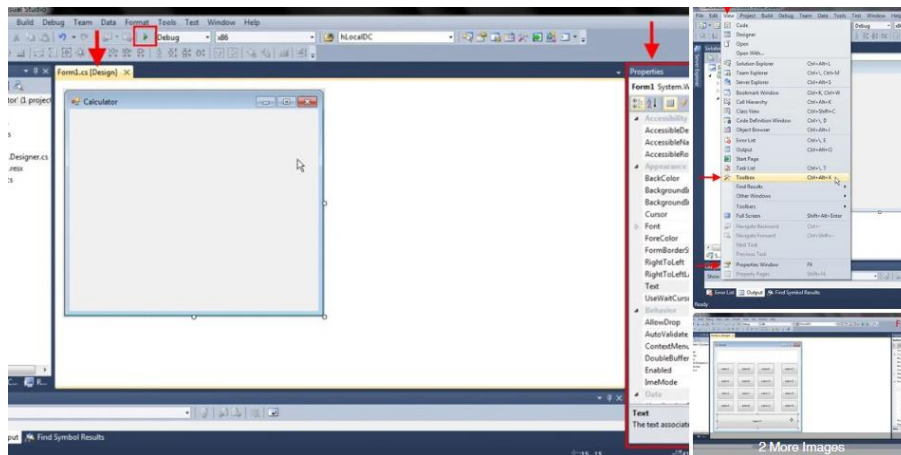


Calculator – Project

Step 01- Designing the application



This step will help you create an intuitive calculator design for your users.

1.1. Select the "Form1.cs [Design]" tab if not already selected. This Form is the window that will display when your program is run. Clicking the green arrow in the toolbar on top of the screen will run the program as an application (opening it in a new window). Try it. Close the new window with the X when you are done.

On the right side of the screen*, there are two important menus (Toolbox and Properties). The Toolbox menu allows you to add new items (like buttons) to your interface, while the Properties menu allows you to change the specifics of your currently selected item on the screen.

***Note: These menus may not be located on the right for everyone. They can be placed wherever the programmer desires by clicking and dragging around the screen. If they are not visible, open the View menu, and choose "Toolbox" or "Properties Window." This will make the menus visible on the screen.**

1.2. In the Toolbox menu, click and drag a Button and a TextBox onto your Form. You can change the size of the Button/TextBox by clicking and dragging any of the dots surrounding it.

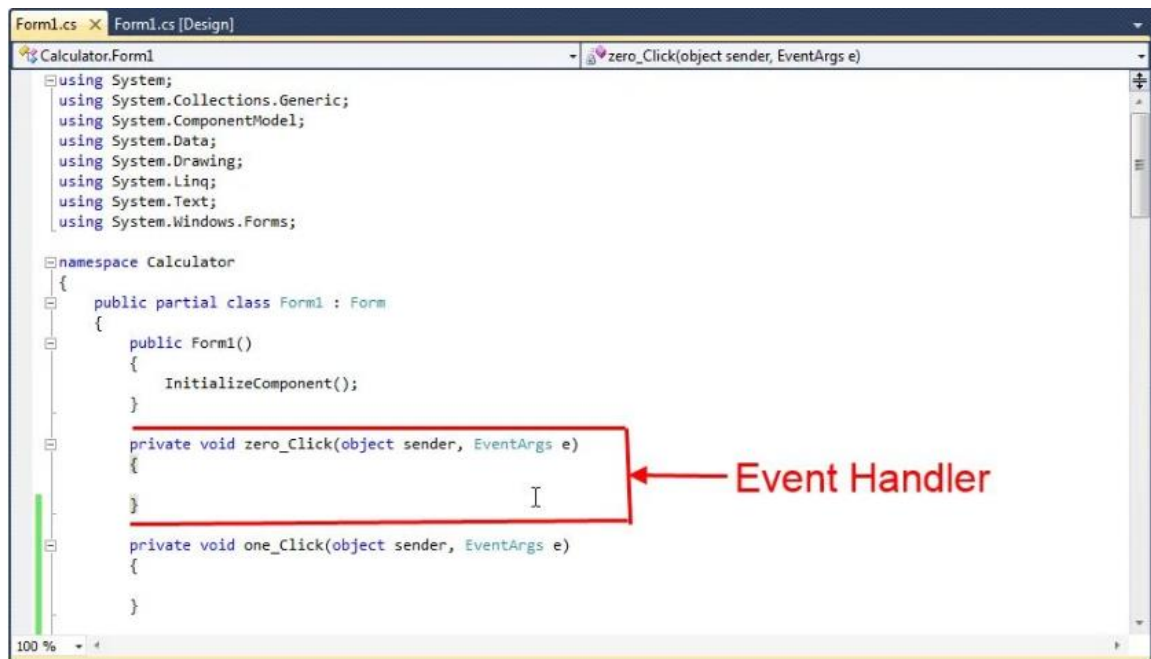
1.3. Add** 16 more buttons to the Form and arrange them to look like a calculator Position them around the Form by clicking and dragging.

1.4. Select a button. Its properties will be displayed in the properties menu off to the side. In the Appearance section, you can edit an item's appearance. Use the Text field to change the text on a button. Have some fun picking different BackColors and types of font.

1.5. Scroll down to the Design section in the Properties menu and edit the name field for each button. (This isn't required, but it will make coding much easier). Name each button something you can easily keep track of. Eg: Name the 1 button "one," that way you can keep track of it in the code later.

1.6. Click Save All and move on to step 2!

Step 2 : Creating Event Handlers

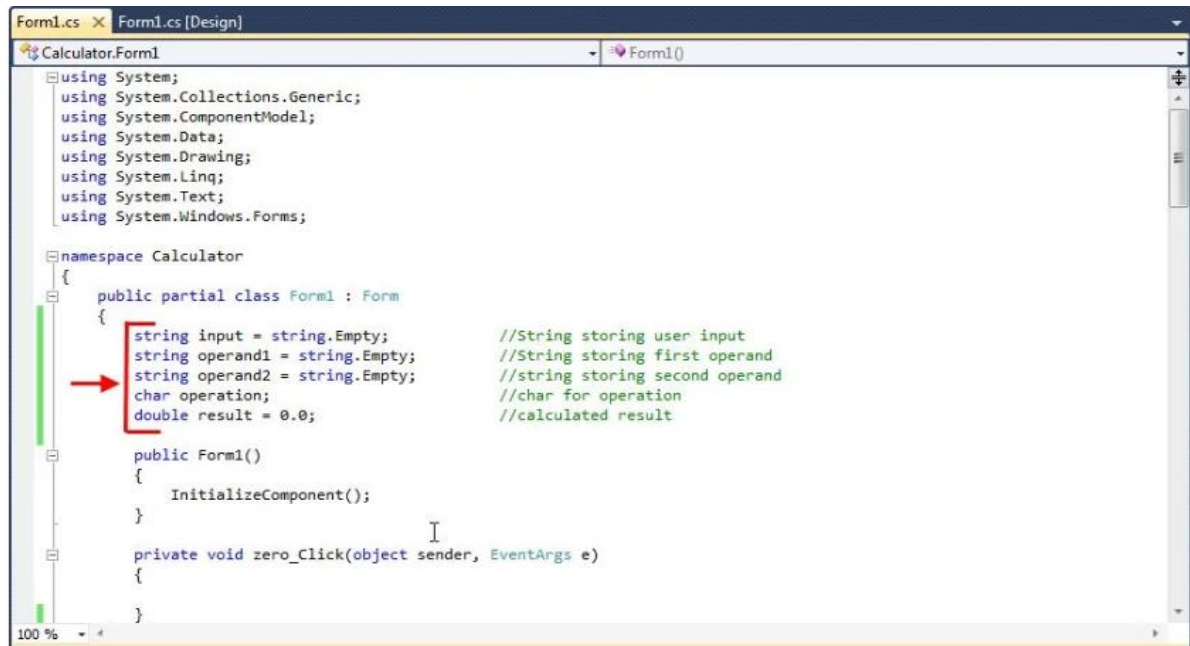


***Note: Make sure all of your buttons are named correctly before you start this step!**

2.1. Double click a button. This will bring you to the "Form1.cs" tab. This is where all of your code will be written. Notice a section has been pre-written for you based off of the "Click" event. This section of code will be executed when that specific button is clicked. For the sake of this project, we will call these Event Handlers.

2.2. Double click all of your buttons so you have an event handler for each button (use the tabs to switch back and forth).

2.3. Click Save All and move on to step 3.



Step 3: Declare Variables

This project requires three different types of variables, char, double, and String.

char - data type for holding any character like the letter 'c' or a star '*'.
double - Data type for holding number values with a decimal. The value 3.14 is an example of a double.

String - A string is not a data type, it is an Object that handles chains of characters. We call these chains strings. Ex: 'c' is a character, "ccc" is a character string.

Variables are declared in the following form: **<Data Type> <Variable Name> = <Initial Value>;**

Ex: To declare a new char initialized to contain 'c', the code would look like this: **char mycharacter = 'c';**

3.1. Declare 5 new variables above the Form1() method as shown in the image above. This will create the needed variables to store and process the input from the user.

***Note: It isn't required to initialize a variable. I do not initialize the operation variable because a value will be placed into it later depending on which operation the user chooses. I do initialize the strings to be empty (using string.Empty) however just to be sure that the string will be completely empty before it is used.**

3.2. Click the Save All button and move on to the next step when you are ready.

Step 4 : Recording User Inputs

Form1.cs [Design]*

```

tor.Form1
- dot_Click(object sender, EventArgs e)
{
    input += "5";
}

private void six_Click(object sender, EventArgs e)
{
    input += "6";
}

private void seven_Click(object sender, EventArgs e)
{
    input += "7";
}

private void eight_Click(object sender, EventArgs e)
{
    input += "8";
}

private void nine_Click(object sender, EventArgs e)
{
    input += "9";
}

private void dot_Click(object sender, EventArgs e)
{
    input += ".";
}

```

Form1.cs [Design]*

```

tor.Form1
- minus_Click(object sender, EventArgs e)

private void divide_Click(object sender, EventArgs e)
{
    operand1 = input;
    operation = '/';
    input = string.Empty;
}

private void star_Click(object sender, EventArgs e)
{
    operand1 = input;
    operation = '*';
    input = string.Empty;
}

private void plus_Click(object sender, EventArgs e)
{
    operand1 = input;
    operation = '+';
    input = string.Empty;
}

private void minus_Click(object sender, EventArgs e)
{
    operand1 = input;
    operation = '-';
    input = string.Empty;
}

```

Things are looking good. The shell is set up and ready to go. But in order for a user to interact with the program, we need to store the data given from the user clicking the buttons.

4.1. Type the line `input += "0";` between the curly braces for the `zero_Click` event handler. This will add a zero to the input character string.

***Note:** Each time the user clicks the zero button, another zero will be added to the input string. One click will have the string contain one zero like so: "0". Clicking the zero button again will add another making the string value "00", and so on.

****Note:** `+=` is a shortcut for writing `input = input + "0";` Essentially, it is adding a zero to the end of what is already existing in the string.

4.2. Repeat step 4.1 for all the numerical input event handlers, as well as the decimal button event handler. Change the "0" value to match which button handler you are in. (Don't want to add a 0 to the string when a 1 is pressed).

4.3. Next add the following lines of code for each of your operand button event handlers (+, -, *, /).

```

operand1 = input;
operation = '+';
input = string.Empty;

```

This code will store the contents of the input string into the `operand1` string, and set the operation character accordingly. The input string gets cleared at the end so it can be ready to record the next value from the user.

4.4. Repeat step 4.3 for all operand input event handlers.

4.5. Click the Save All button and move on to step 5.

Step 5: Viewing Input

```

Form1.cs [Design]*
tor.Form1
three_Click(object sender, EventArgs e)

private void zero_Click(object sender, EventArgs e)
{
    this.textBox1.Text = "";
    input += "0";
    this.textBox1.Text += input;
}

private void one_Click(object sender, EventArgs e)
{
    this.textBox1.Text = "";
    input += "1";
    this.textBox1.Text += input;
}

private void two_Click(object sender, EventArgs e)
{
    this.textBox1.Text = "";
    input += "2";
    this.textBox1.Text += input;
}

private void three_Click(object sender, EventArgs e)
{
    this.textBox1.Text = "";
    input += "3";
    this.textBox1.Text += input;
}
  
```



Users would probably like to see what they are inputting into the calculator. This step adds that feature.

***Note: Don't forget code is executed in a top down manner!**

5.1. Add the line `this.textBox1.Text += input;` underneath the input line as shown. This will add the input to the TextBox that was dragged to the screen earlier.

5.2. Repeat this step for every numerical input handler.

5.3. Next, add the line `this.textBox1.Text = "";` before the input line. This is essential because it clears the TextBox before the input string is added to it. Forgetting this step will leave the user with a mess on the display (feel free to try it out by running the code without this step).

5.4. Run the program and see what happens!

5.5. Click Save All and move on to step 6.

Step 6 : Clearing Input

From the last step, you have probably noticed that we can't clear the input off the screen. This step will fix that issue!

6.1. Add the following lines of code in the clear button event handler.

```

this.textBox1.Text = "";
this.input = string.Empty;
  
```

```
this.operand1 = string.Empty;
this.operand2 = string.Empty;
```

Code Explained: We have seen all of these lines before, but here is a quick refresher. The first line clears the TextBox, and the three lines after empty our input, operand1, and operand2 strings. This will effectively clear everything.

6.2. Run the program and see what happens.

6.3. Click Save All and move on to step 7.

Step 7: Calculating Input

```
private void equals_Click(object sender, EventArgs e)
{
    operand2 = input;
    double num1, num2;
    double.TryParse(operand1, out num1);
    double.TryParse(operand2, out num2);

    if (operation == '+')
    {
        result = num1 + num2;
        textBox1.Text = result.ToString();
    }
    else if (operation == '-')
    {
        result = num1 - num2;
        textBox1.Text = result.ToString();
    }
    else if (operation == '*')
    {
        result = num1 * num2;
        textBox1.Text = result.ToString();
    }
    else if (operation == '/')
    {
        if (num2 != 0)
        {
            result = num1 / num2;
            textBox1.Text = result.ToString();
        }
        else
        {
            textBox1.Text = "DIV/Zero!";
        }
    }
}
```

Now time
best, calculate output!

to do what a calculator does

7.1. Insert the following code into the equals button event handler. (Try not to just paste it, try and understand what is happening).

```
operand2 = input;
double num1, num2;
double.TryParse(operand1, out num1);
double.TryParse(operand2, out num2);

if (operation == '+')
{
    result = num1 + num2;
    textBox1.Text = result.ToString();
}
else if (operation == '-')
```

```
{
    result = num1 - num2;
    textBox1.Text = result.ToString();
}
else if (operation == '*')
{
    result = num1 * num2;
    textBox1.Text = result.ToString();
}
else if (operation == '/')
{
    if (num2 != 0)
    {
        result = num1 / num2;
        textBox1.Text = result.ToString();
    }
    else
    {
        textBox1.Text = "DIV/Zero!";
    }
}
}
```

Code Explained: We first set our input string equal to our second operand string (operand2). We do this because we are assuming that the user has clicked the "=" button. Next we create some more doubles (num1 and num2). These will store the numerical values that have been stored in operand1 and operand2. The next lines (**double.TryParse(operand1, out num1);** and **double.TryParse(operand2, out num2);**) will convert the string that is in operand1 or operand1, into a double type, and then place it in num1 or num2, depending on what we specified in the method. This will allow the computer to be able to perform mathematical functions on the numbers.

If() and else if() explained: The functions if() and else if() are used for data comparisons. The first if() statement can be read as "if the character in operation is equal to the + sign." If this condition is true, the code will execute whatever code is inside of the if statement's curly braces({}). Else if() statements are used if multiple if statements are being used to check a condition. If the previous if() or else if() is not true, then the code in the brackets will be skipped and the program will jump to the next if() or else if() in line, and so on.

If our condition is satisfied based off of what operation was set to, we tell the computer to execute the correct math by placing the correct functions in their corresponding if() statements. The code **result = num1 + num2;** will add the variables num1 and num2 together, and store the result in the result variable.

Displaying the result: The code line **textBox1.Text = result.ToString();** Will convert the result variable to a string type, and then place it in the TextBox to display the output.

7.2. Run your code and see what happens!

7.3 Click Save All, and move on to step 8.

Step 8: Debugging

Debugging! Time to look for errors in our code and see what we can do to fix it!

Reference

www.instructables.com