# JIT121 Programming Tutorial 5

## Learning Objectives

1. Developing simple GUIs

## Assumptions

That you are already familiar with;

* Lectures 1 – 5 and Tutorials 1 - 4

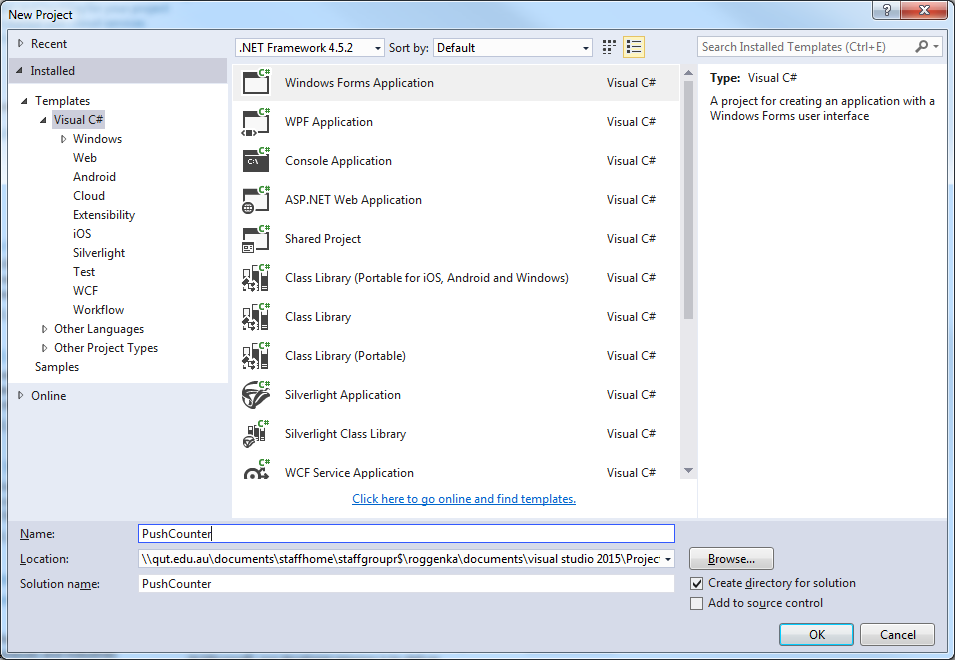
## Activity Overview

1. Simple GUI Development – Push Counter
2. Push Counter - continued
3. Currency Converter

**Activity 1: Simple GUI Development – Push Counter Part 1**

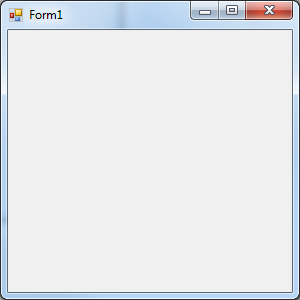
This activity will give you experience with using the **Windows Forms Designer** to create a simple GUI program.

1. Start Visual Studio and create a **Windows Forms application** named **PushCounter**



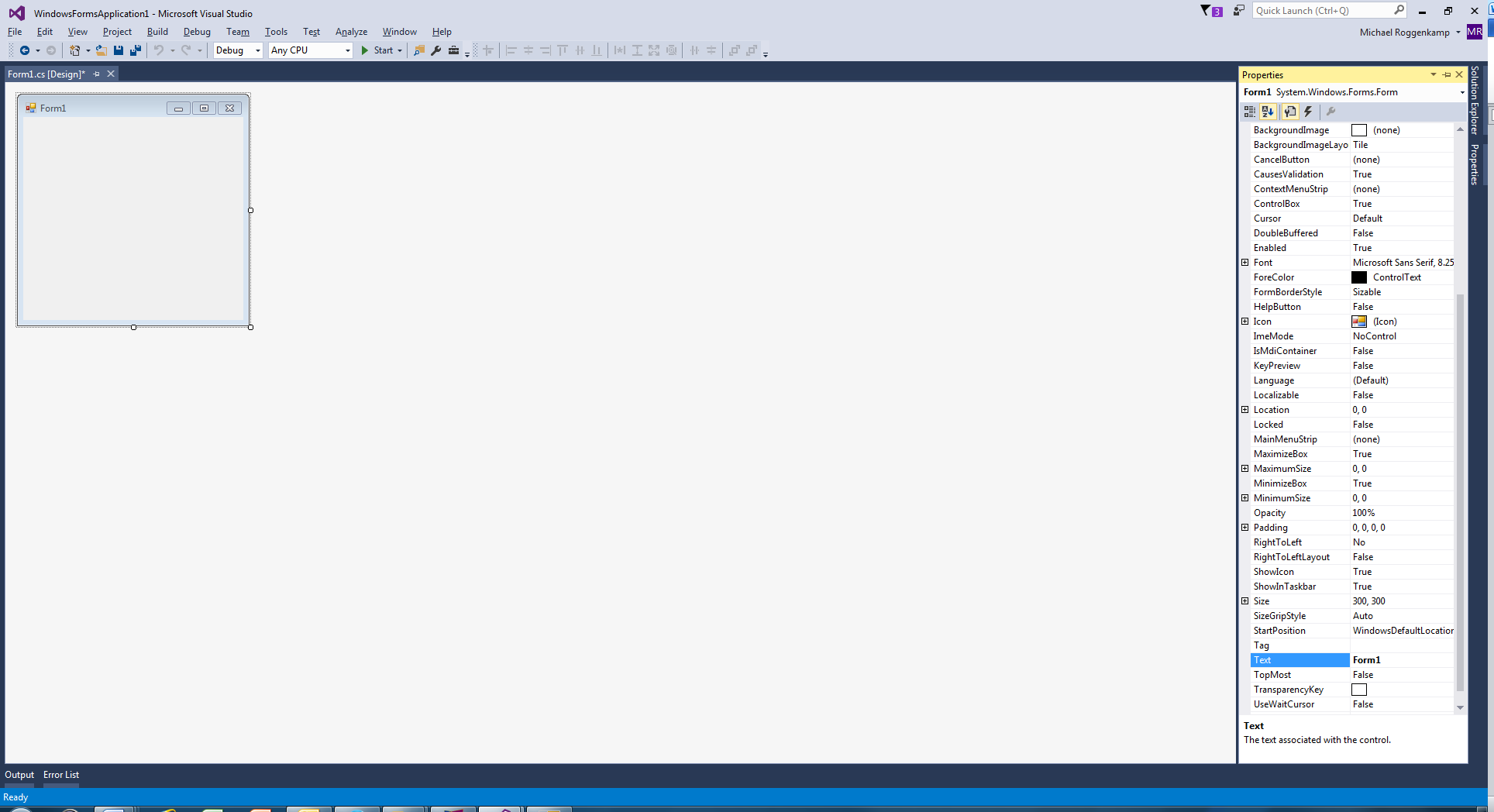
1. This will generate a default form named **Form1.** Open **Solution Explorer** to see that the project contains **Form1.cs**, this is the default form and the usual file named **Program.cs** which contains the **Main** method. Have a look at the code inside **Program.cs** which has been automatically generated.

At this stage you are not expected to understand the body of **Main** except that just as with a Console application, execution starts here and control is passed to the newly created **Form**, **Form1**. Run the program (Start or F5); the empty form will be displayed. Close the window.



1. Now to explore the **Forms Designer.** Open the **Properties Window** (tab). If the tab is not visible on right hand side, from **View** menu, select **Properties Window**.

If a list of various properties is not displayed similar to screenshot below, move cursor to anywhere within the form and click once, now reopen **Properties**.

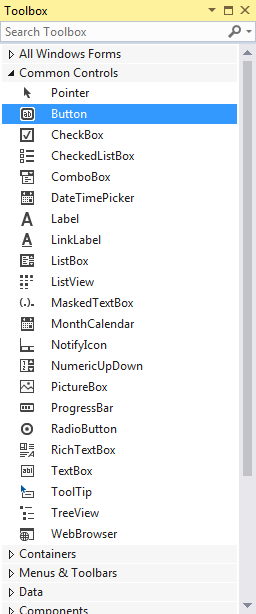
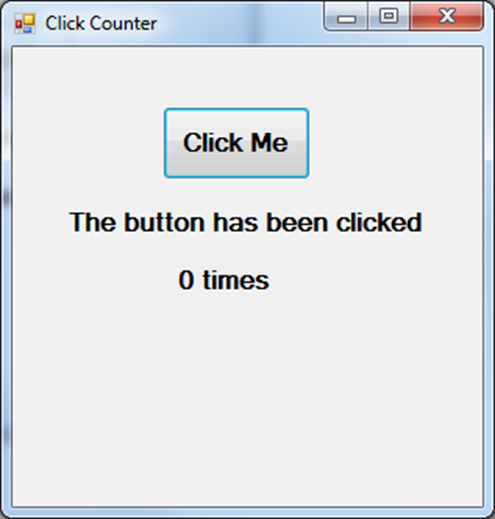


Scroll down and replace **Form1** with **Click Counter** as the **Text** property value.

Note: The title on Form1 is now changed.

1. Now add some controls to your form. This is done by dragging and dropping the controls from the Toolbox onto the form. Open the Toolbox using the **View 🡪 Toolbox** menu or the Toolbox icon (little black toolbox image)

Add a **Button** and two Labels to resemble the form below.



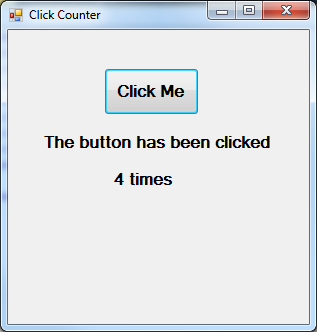
As you add a control to the form, change the **Name** and **Text** propertiesand perhaps **Font.**

For example change the **Name** of the **Button** from **button1** to **clickMeButton** and the **Text** to **Click Me** with **Font Size** of **12** and **Bold** set to **True.** Rename the Label which says **“0 times”** to **clickCounterLabel.** The other **Label Name** can be left as its default.

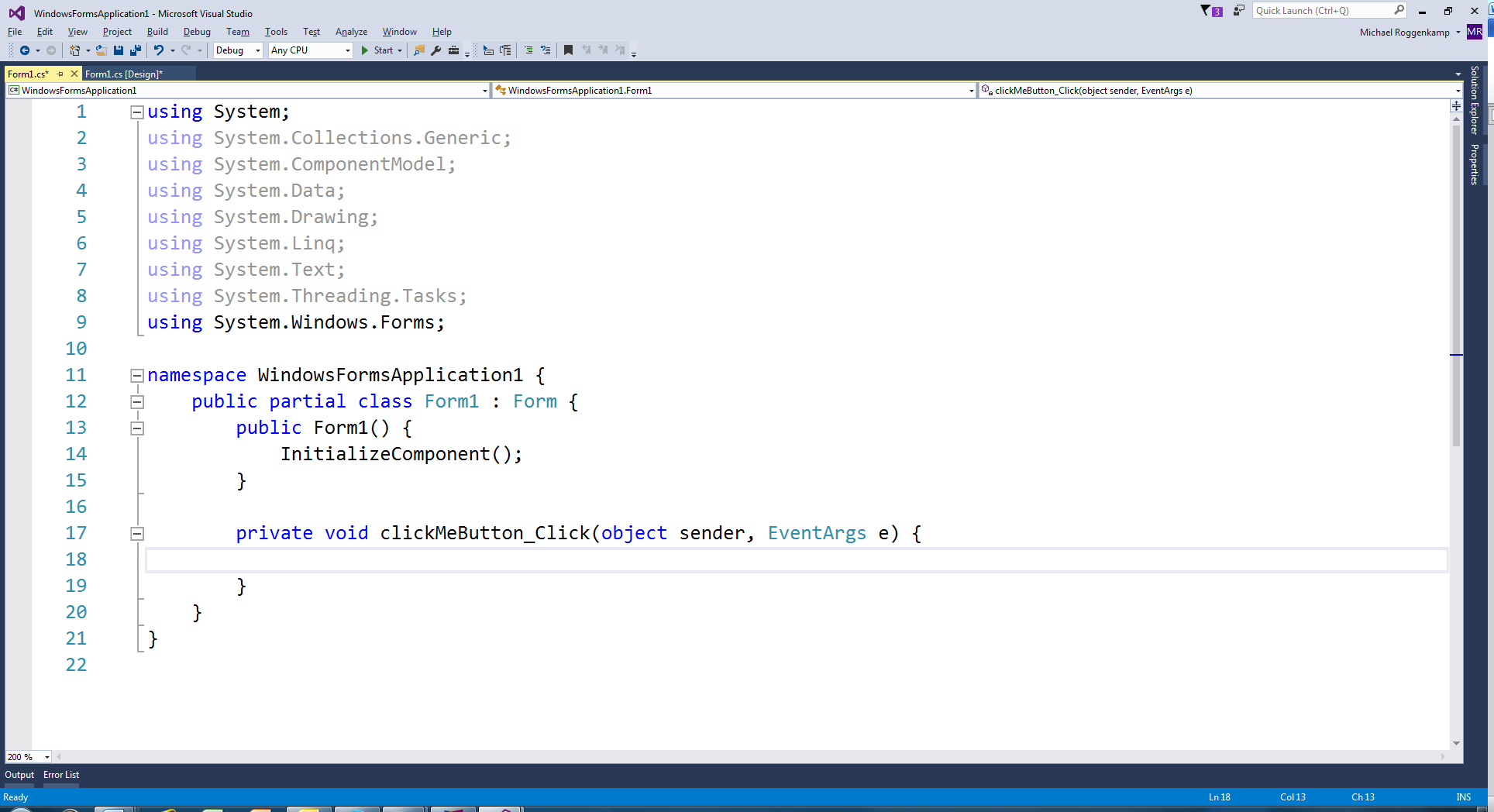
It is important when adding controls to a form to give the control a meaningful name because that will help you later when you code the event handler rather than staying with the default names of button1, label1, label 2 etc.

Run your program so you can see how the form looks. But remember the **Button** does not do anything yet as we have not written the event handler for the **Button**.

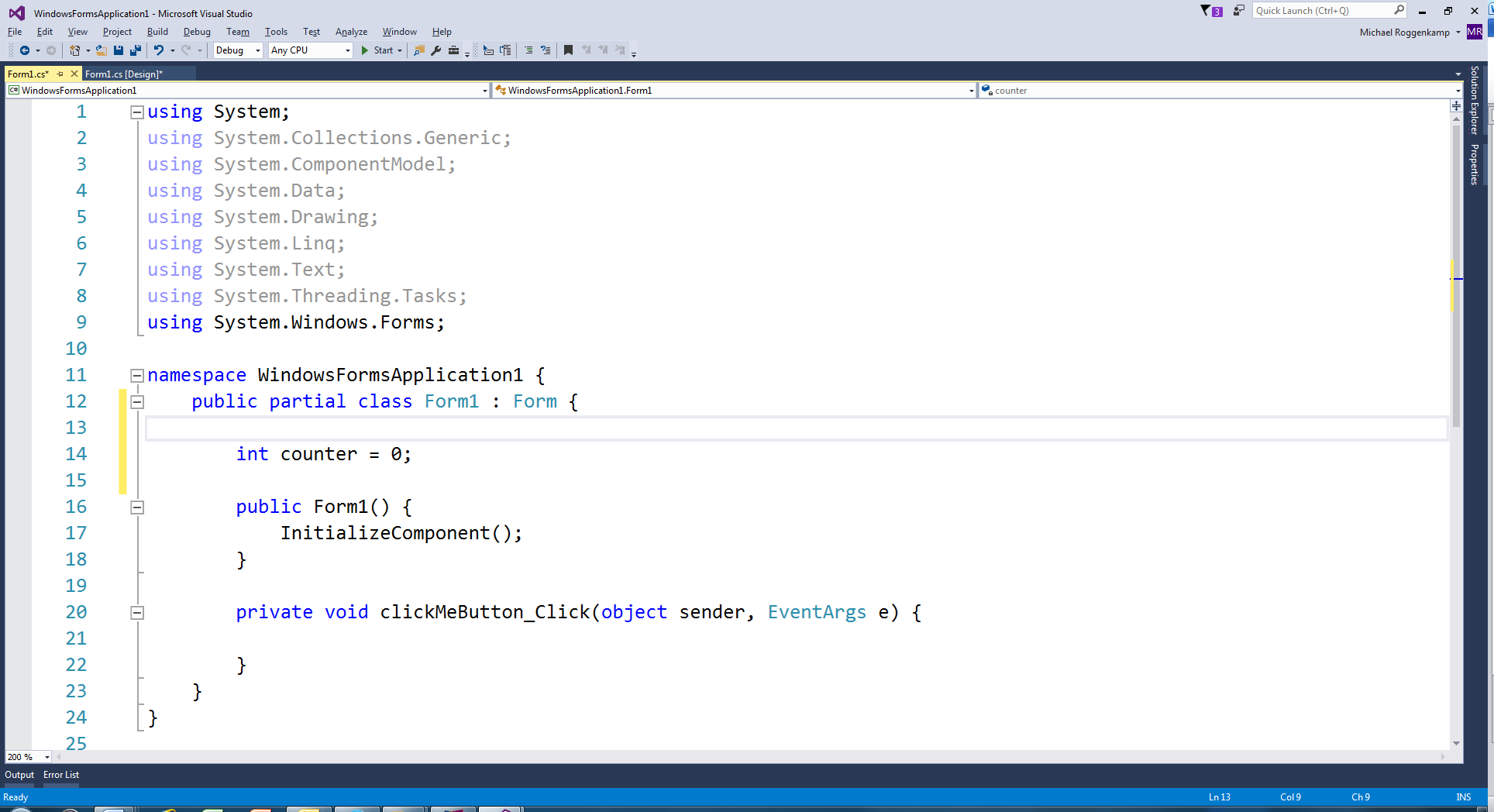
Feel free to play with the look of the **Form**, the **Button** and **Label**s.

1. What we want to happen each time the **Button** is clicked is for the text of **clickCounterLabel** to be updated to show how many times the **Button** has been clicked. For example, the following is the **Form** after four (4) clicks of the **Button**.

To add event handler code for the **Button**, double click on the **Button** in the **Form**. This will generate an empty event handler method. This method is called whenever the user clicks on the **Button**.



At this point, it is a simple programming exercise. We need a variable to hold the number of times the **Button** is clicked. For reasons to be explained at a later date, it is common to use "global" variables within the code manipulating a **Form**. See following screen shot with an **int** variable declared.

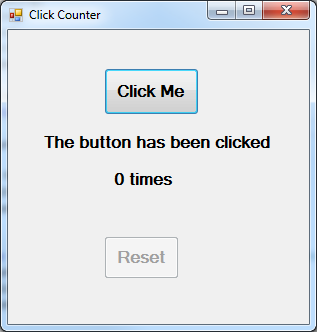


Remember all controls are just variables (objects) that can be referenced in our code and therefore we can access and update any Property field, eg clickCounterLabel.Text

So we just need to add two lines of code tothe **clickMeButton\_Click** method**,** the first to add one to the variable and the second to update the **Label**’s text. There is no need for a **return** statement in this **void** method as the program control will pass back to the **Form** once the method has executed the two lines.

Run your program and test that you have successfully coded the event handler. For each of the following steps run your program after each step has been implemented.

1. Once you have understood what you have done so far, you can add another **Button** to the **Form** which when pressed will reset the counter to zero and update the **Label**. Place the **Button** below the **Label**s. Write the event handler for this **Button** and run your program, testing that both **Button**s now perform as expected.
2. Using the Properties menu for the **Reset Button** to set the **Enabled** field to **false** so that when the program begins the user can only press the **Click Me** **Button**.

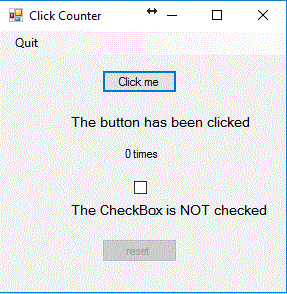


1. To the clickMeButton\_Clickmethod add an assignment statement which will set the **Enabled** field of the **Reset Button** to **true.** It makes no difference if this statement is the first or last statement in the method.
2. To the event handler for the **Reset Button**, add a statement to set the **Button**’s **Enabled** property to **false**.

For homework go back over this activity as it is essential you understand the basics of event driven programming as demonstrated in this activity.

**Activity 2: Push Counter Part 2**

Add more controls to your GUI as follows:



**MenuStrip**

**CheckBox**

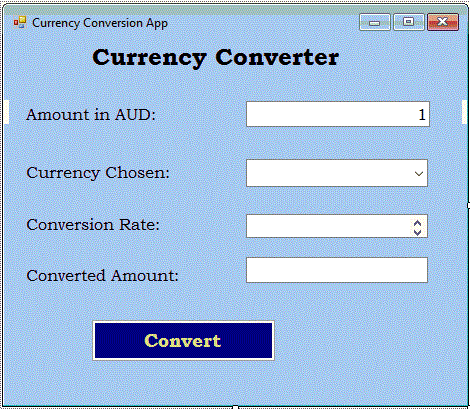
**Label**

Add an event handler to your code for the **CheckBox** by double-clicking that control in Design mode. When the **CheckBox** is checked (ticked) the new **Label** should show "The CheckBox is checked" and when it is not checked (unticked), the **Label** should show "The CheckBox is NOT checked" as shown in the diagram. You can test if the **CheckBox** is checked by accessing the value of the **Checked** property. If the **Checked** property is **true**, then the **CheckBox** is checked.

Now add an event handler for the menu item. Double-click on the **Quit** menu item and an empty **Click** method will automatically be generated. All we want to do when this menu is clicked is to end the program. In a GUI program, that is done by closing the **form**. So, the code for this method will be a call to the **Close**() method.

### Activity 3: Currency Converter App

Create a GUI to convert from Australian dollars into three other currencies of your choosing.



**Labels**

**TextBox**

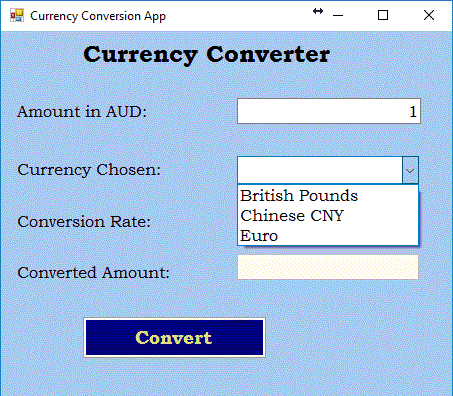
**ComboBox**

**ListBox**

**TextBox**

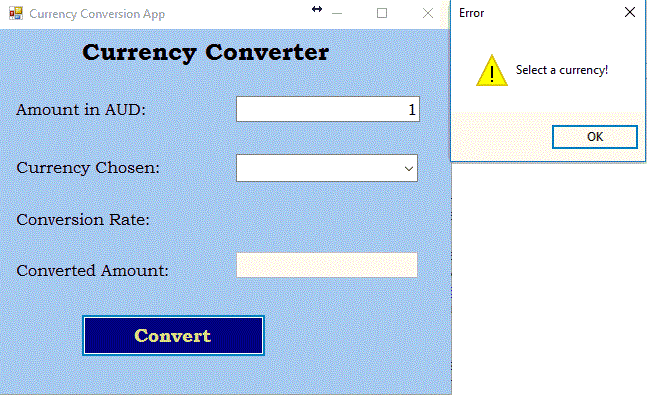
**Button**

**Label**

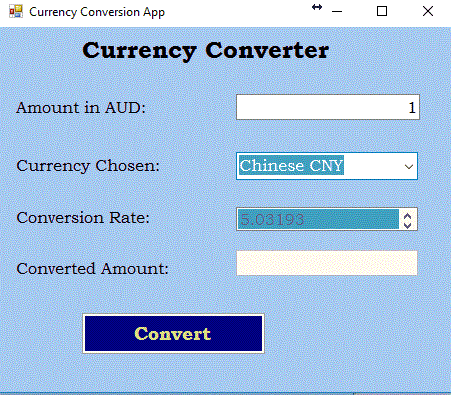


**Requirements**

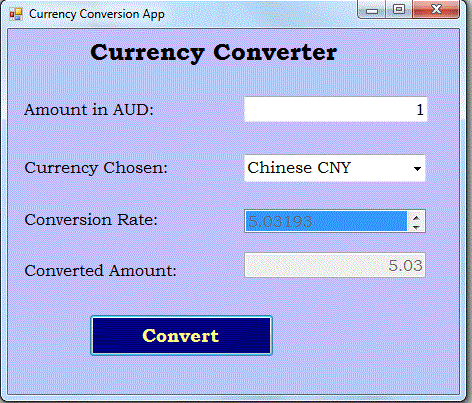
* Each control must have a meaningful name!
* The Amount in AUD **TextBox** should initially show the value "1".
* The Currency Chosen **ComboBox** should list the choice of three other currencies eg:
* The *Converted Amount* **TextBox** should remain disabled (read-only for the user).
* The *Conversion Rate* **ListBox** should initially be invisible.
* If the user hits the *Convert* **Button** before a currency is chosen, a **MessageBox** should inform them that a currency must be chosen first.



* When a currency is chosen, the **Listbox** should display the corresponding currency rate to the currency chosen by the user.



* When the **Button** is hit, the converted amount should appear. (This should be the amount in AUD multiplied by the conversion rate.)



* When the next currency is chosen the *Converted Amount* **TextBox** should be empty, until the convert **Button** is hit again.

**Events**

* When the user chooses a currency from the **ComboBox**, the correct currency conversion amount will appear in the *Conversion Rate* **ListBox**. (Look up the relevant conversion rates online eg: <https://www.oanda.com/currency/converter/> to populate the **ListBox** items). It may be helpful to have the **ComboBox** items and **ListBox** items in the same order. There are no calculations required in this method.
* The user will type an amount in Australian dollars. When they click on the *Convert* **Button**, the program will read the dollar amount, read the conversion rate and then write the converted amount in the **TextBox**.

**Implementation hints**

* Use an **enum** to describe the currencies available
* List the items in the **ListBox** and the **ComboBox** in the same order as the enum values
* For the purpose of this exercise, assume the user will only enter sensible data (exception handling will be introduced in future weeks).