

# Windows Form

Course Code: CSC 2210      Course Title: Object Oriented Programming 2



**Dept. of Computer Science**  
**Faculty of Science and Technology**

|                     |                        |                 |  |                  |  |
|---------------------|------------------------|-----------------|--|------------------|--|
| <b>Lecturer No:</b> |                        | <b>Week No:</b> |  | <b>Semester:</b> |  |
| <b>Lecturer:</b>    | <i>Noboranjana Dey</i> |                 |  |                  |  |

# Lecture Outline



- Basics of Windows Form Application.
- Using Different components of Windows Form Application.
- C# Graphical User Interface.
- Multiple Form Interaction.



# *Windows Forms Application*

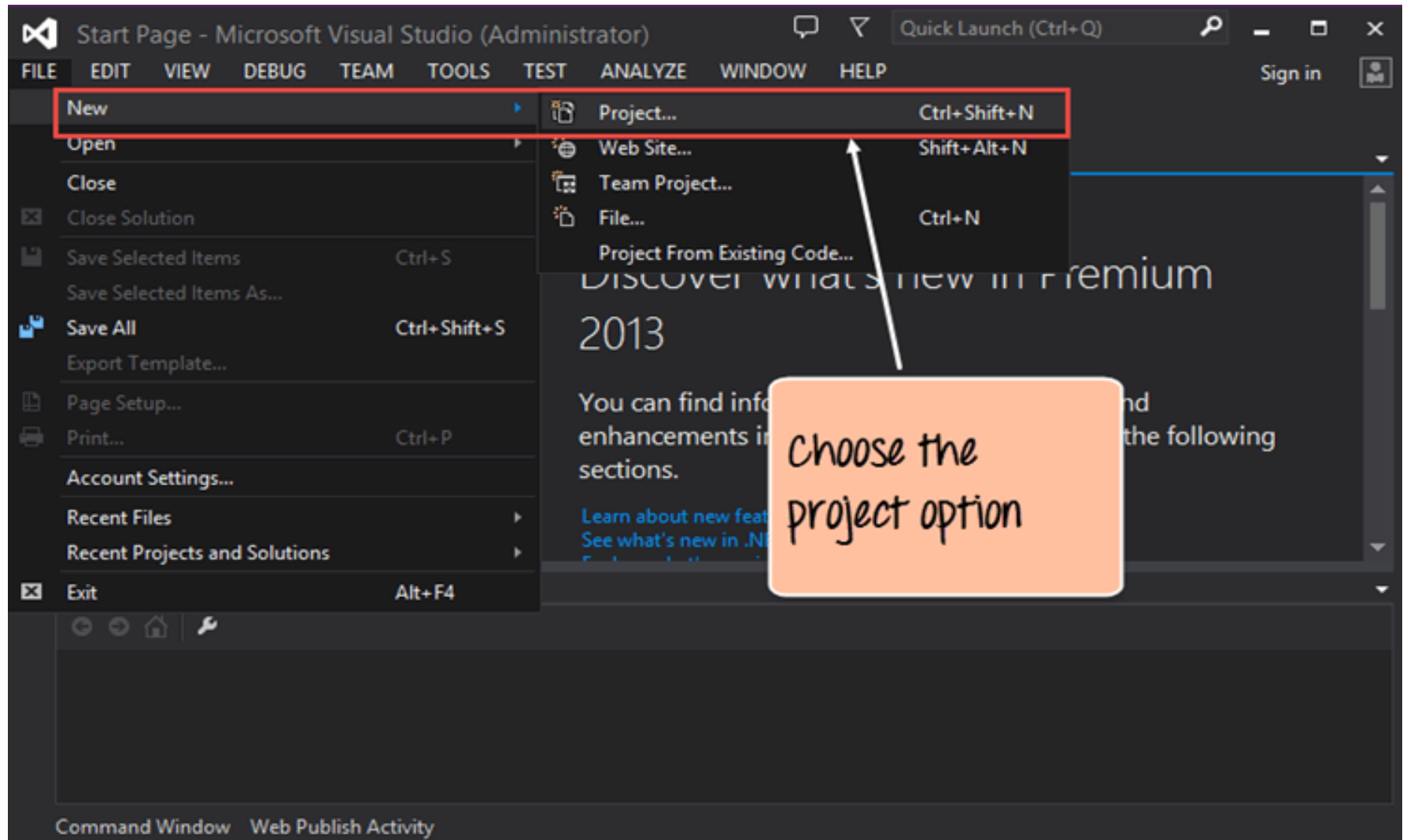
# Windows Forms Application

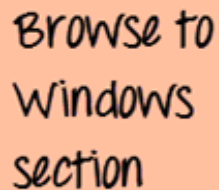


- Windows Forms is a Graphical User Interface(GUI) class library which is bundled in .Net Framework or Mono Framework .
- Its main purpose is to provide an easier interface to develop the applications for desktop, tablet, PCs.
- It is also termed as the **WinForms**.
- The applications which are developed by using Windows Forms or WinForms are known as the **Windows Forms Applications** that runs on the desktop computer.
- WinForms can be used only to develop the Windows Forms Applications not web applications.
- Providing a platform to write rich client applications.
- WinForms applications can contain the different type of controls like labels, list boxes, tooltip etc.
- Now lets work how to create an *Windows Forms Application*



**Step 1)** The first step involves the creation of a new project in Visual Studio. After launching Visual Studio, you need to choose the menu option New->Project.





## Choose Windows Forms Application

Give a name and location for the application

Click the OK button

Browse...

☒ Create directory for solution

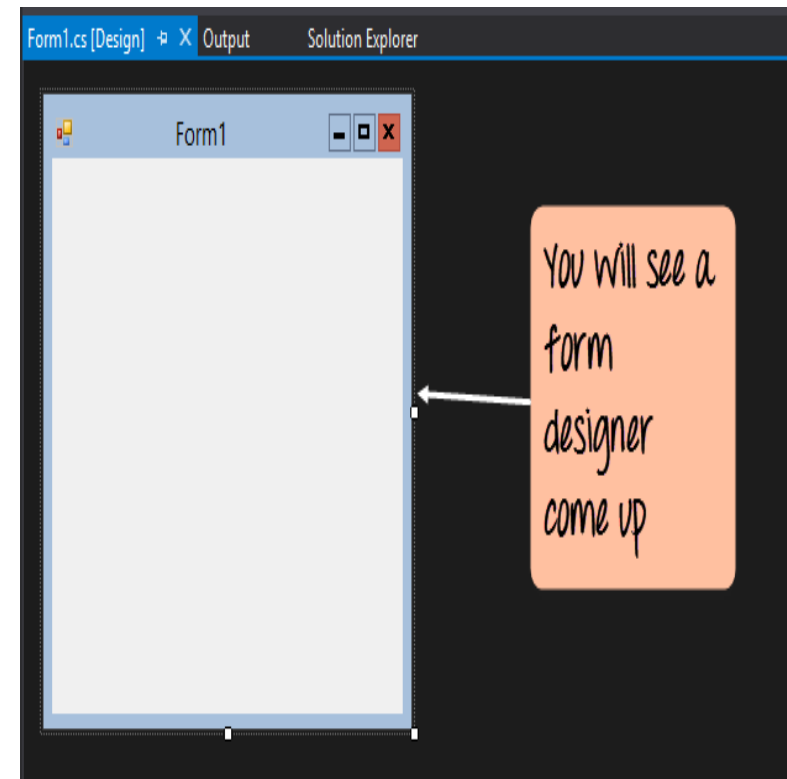
OK

Cancel

# Windows Forms Application

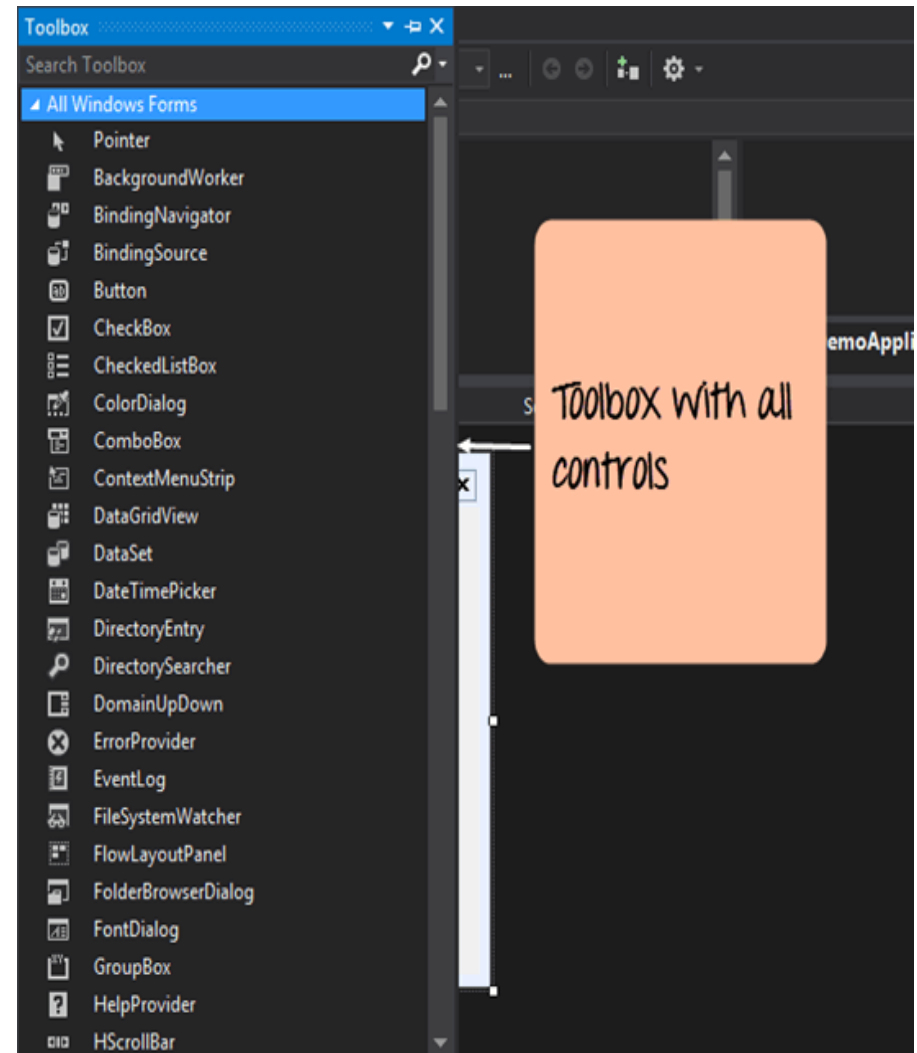
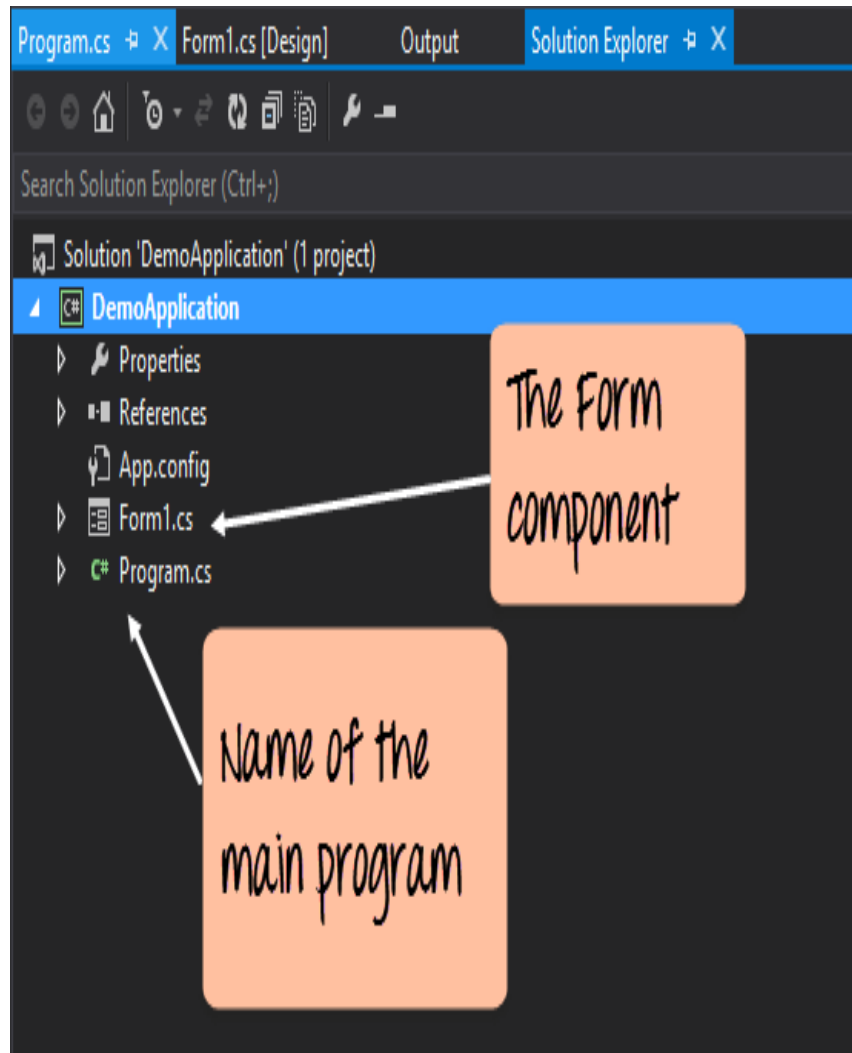


- In the project dialog box, we can see various options for creating different types of projects in Visual Studio. Click the Windows option on the left-hand side.
- When we click the Windows options in the previous step, we will be able to see an option for Windows Forms Application. Click this option.
- We then give a name for the application which in our case is DemoApplication. We also need to provide a location to store our application.
- Finally, we click the 'OK' button to let Visual Studio create our project.





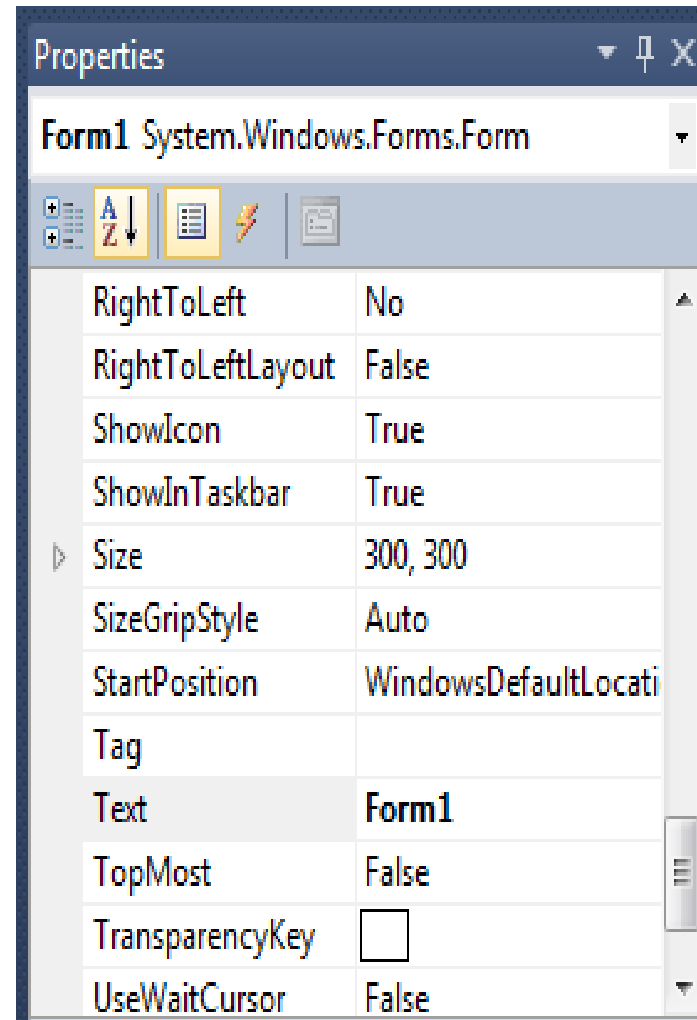
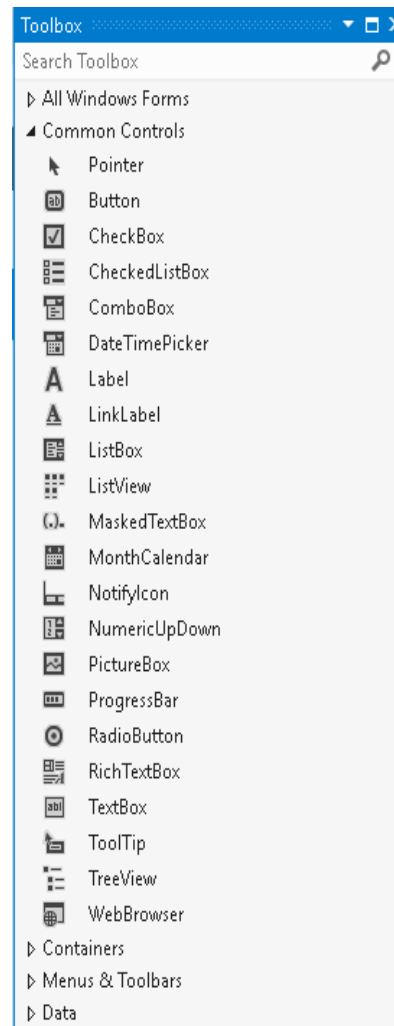
The toolbox contains all the controls which can be added to a Windows Forms. Controls like a textbox, label, button etc. Drag and drop the controls that you needed on created Form.





➤ In properties panel you can configure your controls like their size, color, name etc.

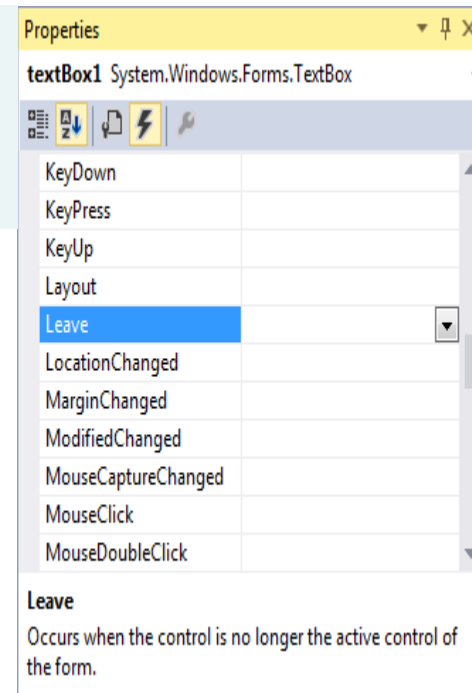
➤ In ToolBox panel you can find all the controllers for your project.



# Events

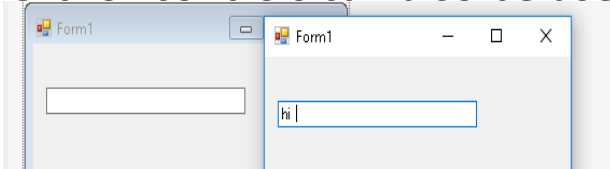
Events are user actions such as key press, clicks, mouse movements, etc., or some occurrence such as system generated notifications. Applications need to respond to events when they occur. For example, interrupts. Events are used for inter-process communication.

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    //
    // This changes the main window text when you type into the TextBox.
    //
    this.Text = textBox1.Text;
}
```

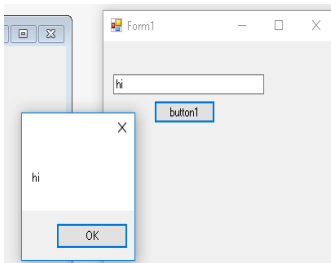
The image shows the Visual Studio Properties window for a control named 'textBox1' of type 'System.Windows.Forms.TextBox'. The window has a yellow title bar and a dropdown menu showing 'textBox1 System.Windows.Forms.TextBox'. Below the title bar, there are icons for 'Show All Properties', 'Show Default Properties', 'Show Inherited Properties', and 'Show Custom Properties'. The main area of the window is a list of properties. The 'Leave' property is currently selected and highlighted in blue. Below the list, there is a section titled 'Leave' with a description: 'Occurs when the control is no longer the active control of the form.'

# Control Details

**TextBox** : A **TextBox** control is used to display, or accept as input, a single line of text. TextBox controls can also be used to accept passwords and other sensitive information.



**Button** : A **button** accepts clicks. In Windows Forms we use a Button control that accepts click events and performs other actions in the user interface. This control provides a way to accept input—and invoke logic based on that input.

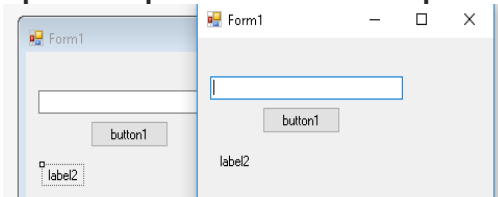


```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show(textBox1.Text);
}
```

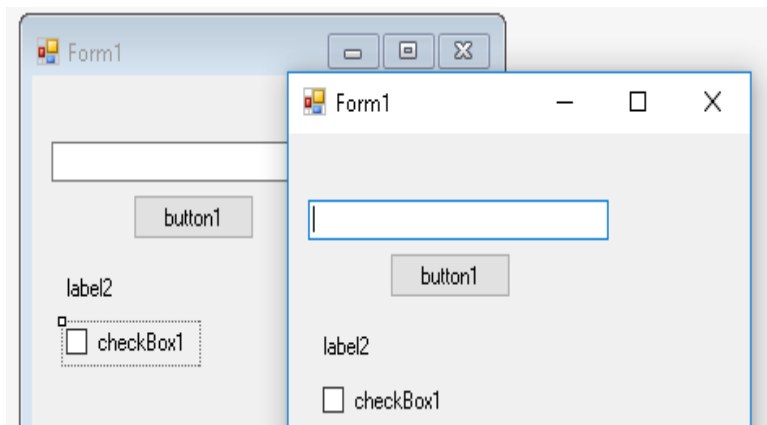
Button click  
event

# Control Details

**Label** : A **Label** control is used as a display medium for text on Forms. Label control does not participate in user input or capture mouse or keyboard events.

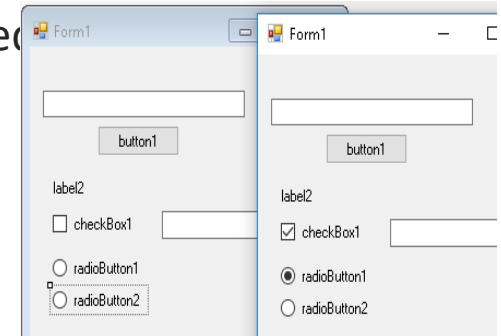


**CheckBox** : A **CheckBox** control allows users to select single or multiple items from a list of items.

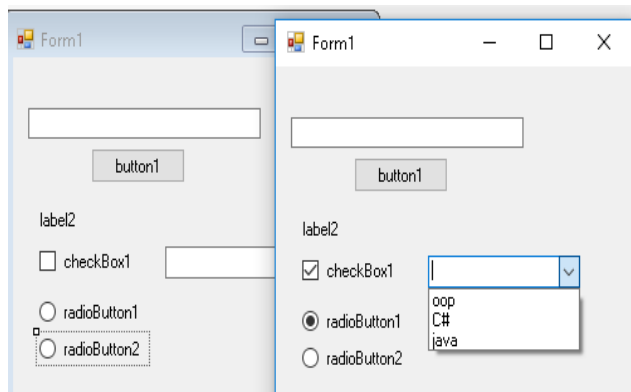


# Control Details

**RadioButton** : A **RadioButton** control provides a round interface to select one option from a number of options. Radio buttons are usually placed in a group on a container control, such as a Panel or a GroupBox, and one of them is selected.



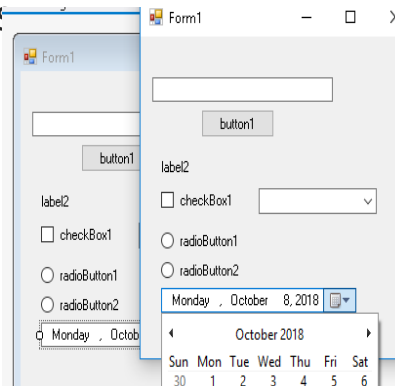
**ComboBox** : A **ComboBox** control provides combined functionality of a text box and a listbox in a single control. Only one list item is displayed at one time in a ComboBox and rest of the available items are loaded in a drop down list.



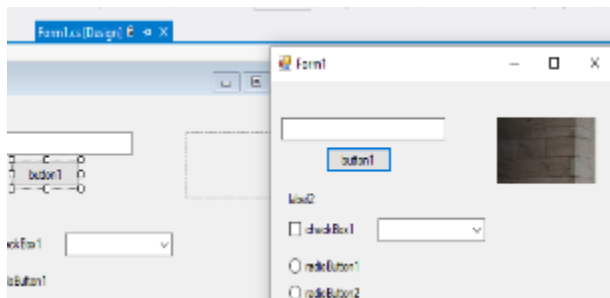
```
comboBox1.Items.Add("test1");  
comboBox1.Items.Add("test2");  
comboBox1.Items.Add("test3");  
comboBox1.SelectedItem = "test3";  
or  
comboBox1.SelectedIndex = comboBox1.FindStringExact("test3");
```

# Control Details

***DateTimePicker*** : A **DateTimePicker** control allows users to select a date and time in Windows Forms applications.

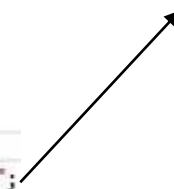


***PictureBox*** : **PictureBox** control is used to display images in Windows Forms.



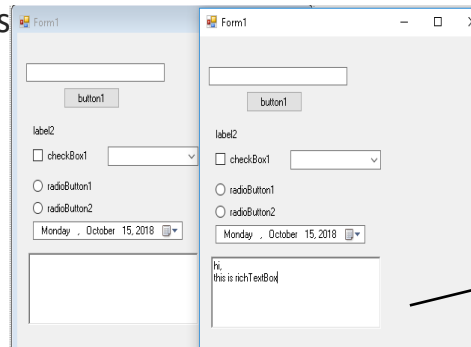
```
private void button1_Click(object sender, EventArgs e)
{
    pictureBox1.ImageLocation = @"C:\Users\ [redacted] \Downloads\centenary company history image IIHIH.jpg";
}
```

Image  
Location



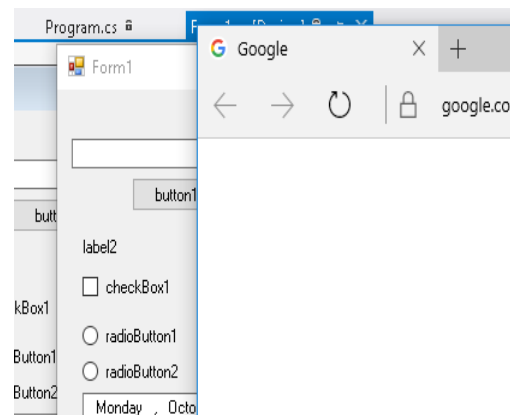
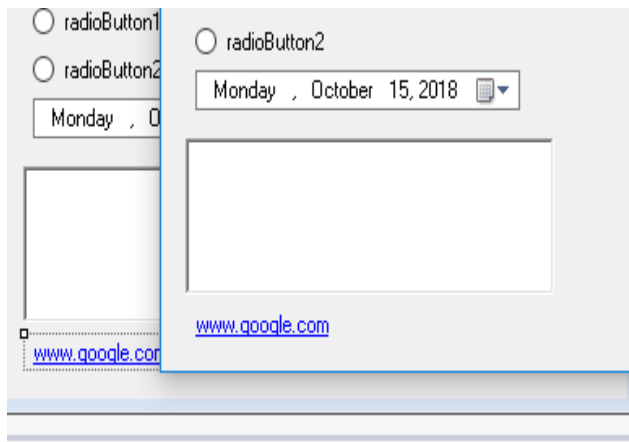
# Control Details

**RichTextBox** : **RichTextBox** has many formatting options. It applies colors, background colors, multiple fonts, and margins to the text. It adds more complexity than the regular TextBox. But the RichTextBox provides needed features to programs



richTextBox

**LinkLabel** : A **LinkLabel** control is a label control that can display a hyperlink. A LinkLabel control is inherited from the Label class so it has all the functionality provided by the Windows Forms Label control. LinkLabel control does not participate in user input or capture mouse or keyboard events.

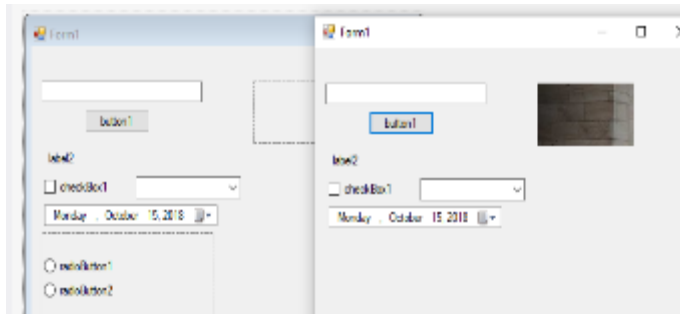


```
using System.Diagnostics;
```

```
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e) {  
    Process.Start("https://www.google.com");  
}
```

# Control Details

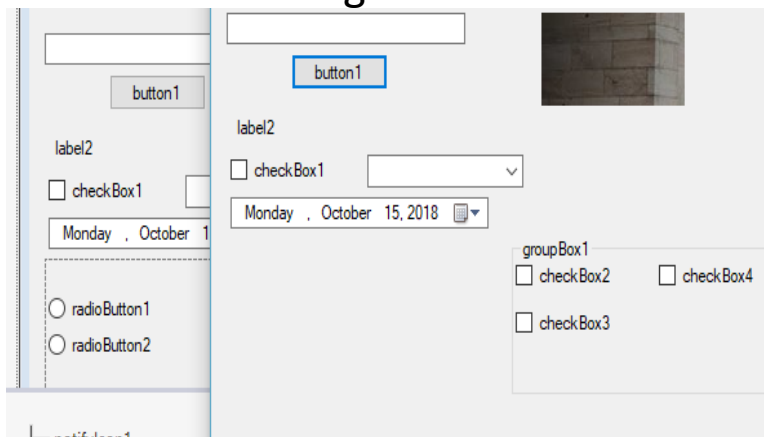
**Panel** : The **Panel** Control is a container control to host a group of similar child controls. Panel Control is useful when you need to show and hide a group of controls.



```
private void button1_Click(object sender, EventArgs e)
{
    panel1.Visible = false;
    pictureBox1.ImageLocation = @"C:\Users\Shaif Hasan\Downloads\centenary company history image IIHIH.jpg";
}
```

For hiding  
or showing  
controls

**GroupBox** : A **GroupBox** control is a container control that is used to place Windows Forms child controls in a group. The purpose of a GroupBox is to define user interfaces where we can categorize related controls in a group.







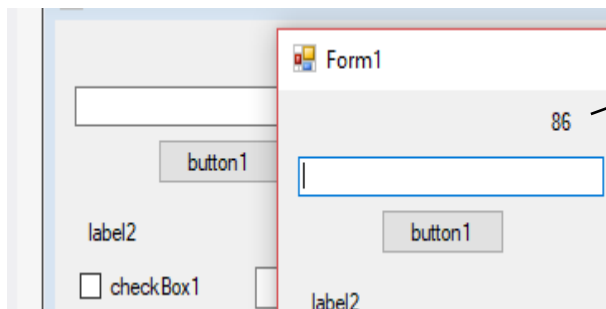
# Control Details

**DataGridView** : **DataGridView** displays data from SQL databases.

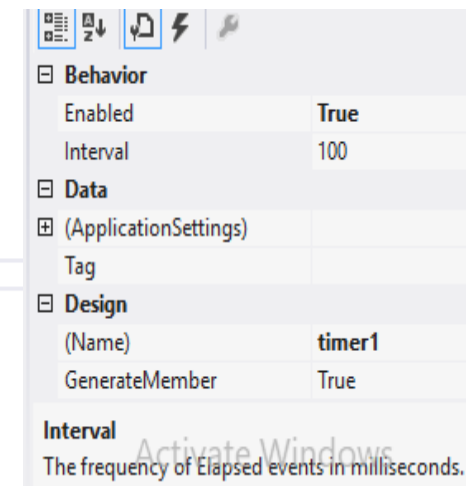
| Logged in ID : |      |            |             | Notice   | Edit Profile | Change PPA | Logout |
|----------------|------|------------|-------------|----------|--------------|------------|--------|
| LOGINSID       | ACCD | U_DATE     | TIME        | COURSEID | BATCHID      | P_STATUS   |        |
| 3              | 1    | 16-12-2017 | 11:36:05 PM | 101      | 1            | Examinee   |        |
| 2              | 1001 | 16-12-2017 | 11:34:10 PM | 1        | 1            | Admin      |        |
| 1              | 3    | 16-12-2017 | 11:37:49 PM | 101      | 1            | Examinee   |        |

```
private void AdminAccessLoginRecords_Load(object sender, EventArgs e)
{
    DataTable t = a.GetLoginRec();
    dataGridView1.DataSource = t;
}
```

**Timer** : The **Timer** control allows you to set a time interval to periodically execute an event at a specified interval. It is useful when you want to execute certain applications after a certain interval.

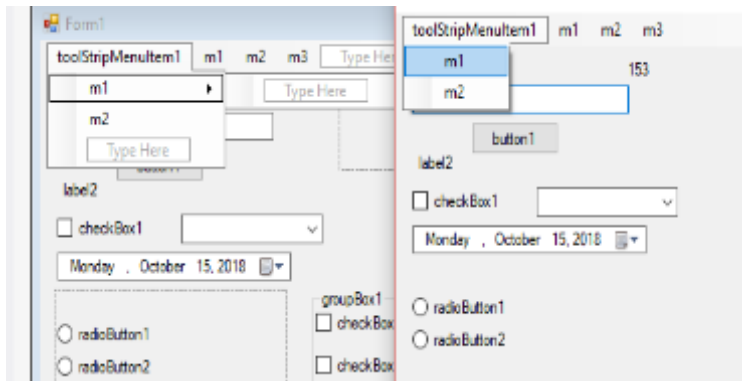


```
private void timer1_Tick(object sender, EventArgs e)
{
    count++;
    label3.Text = count.ToString();
}
```



# Control Details

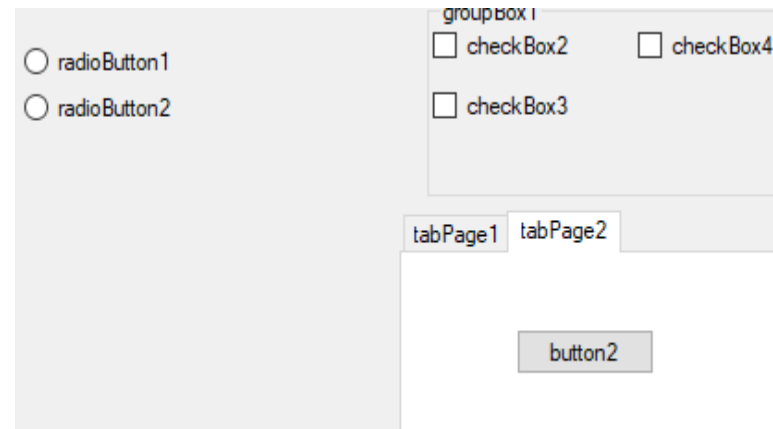
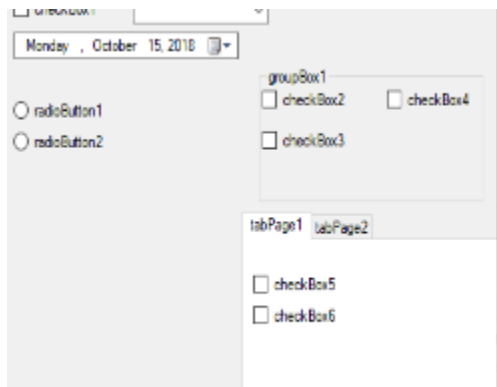
**MenuStrip** : New way to create **menus** in **C#** and WinForms is using **MenuStrip** control.



MenuStrip click event

```
private void m1ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    groupBox1.Visible = false;
}
```

**TabControl** : The TabControl manages tab pages where each page may host different child controls.



# Control Details

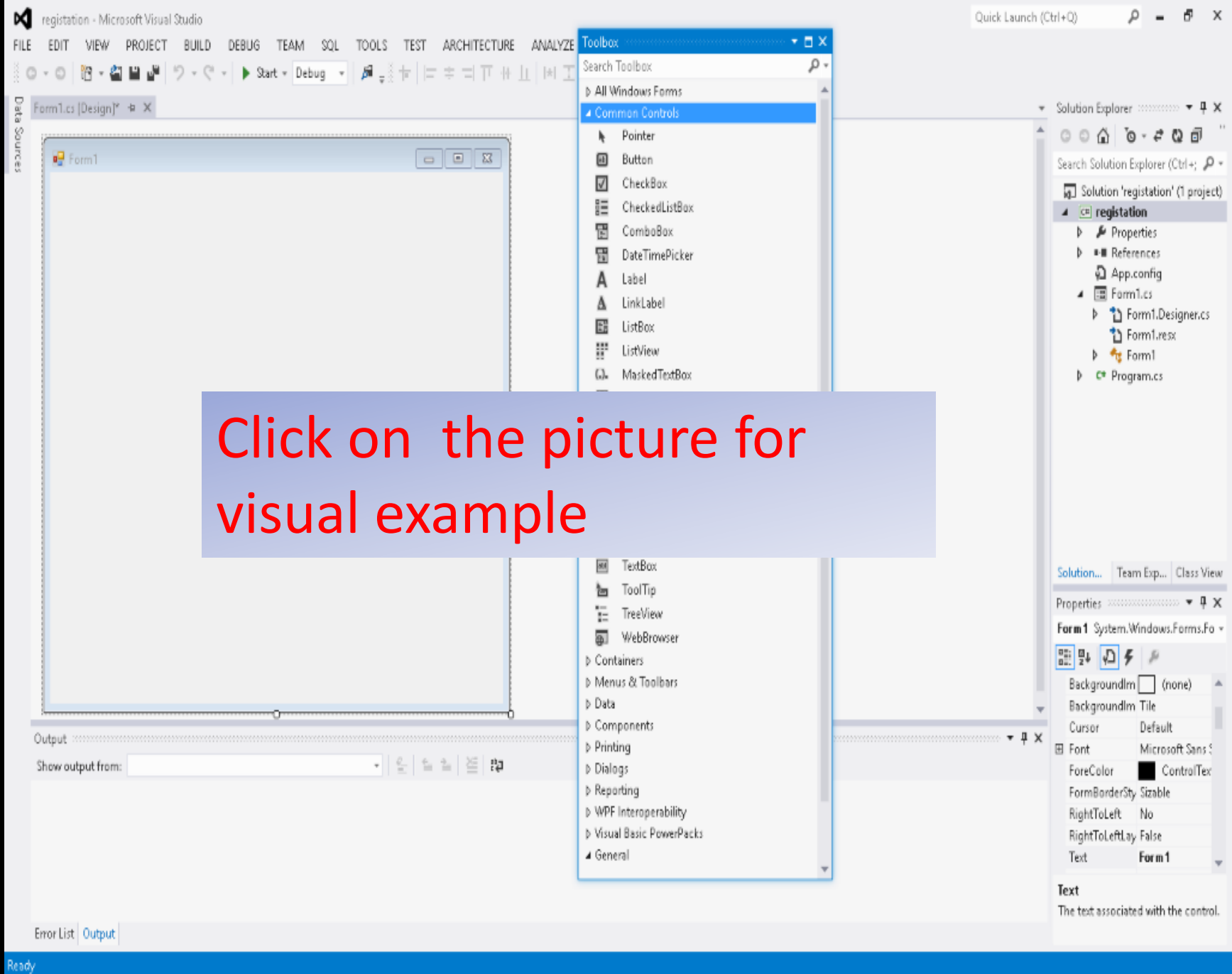
## User Define Events



In general terms, an event is something special that is going to happen. For example, Microsoft launches events for developers, to make them aware about the features of new or existing products. Microsoft notifies the developers about the event by email or other advertisement options. So in this case, Microsoft is a publisher who launches (raises) an **event** and **notifies** the developers about it and developers are the **subscribers** of the event and attend (**handle**) the event.

A custom class can also have an event to notify other subscriber classes about something that has happened or is going to happen.

Lets see an Example

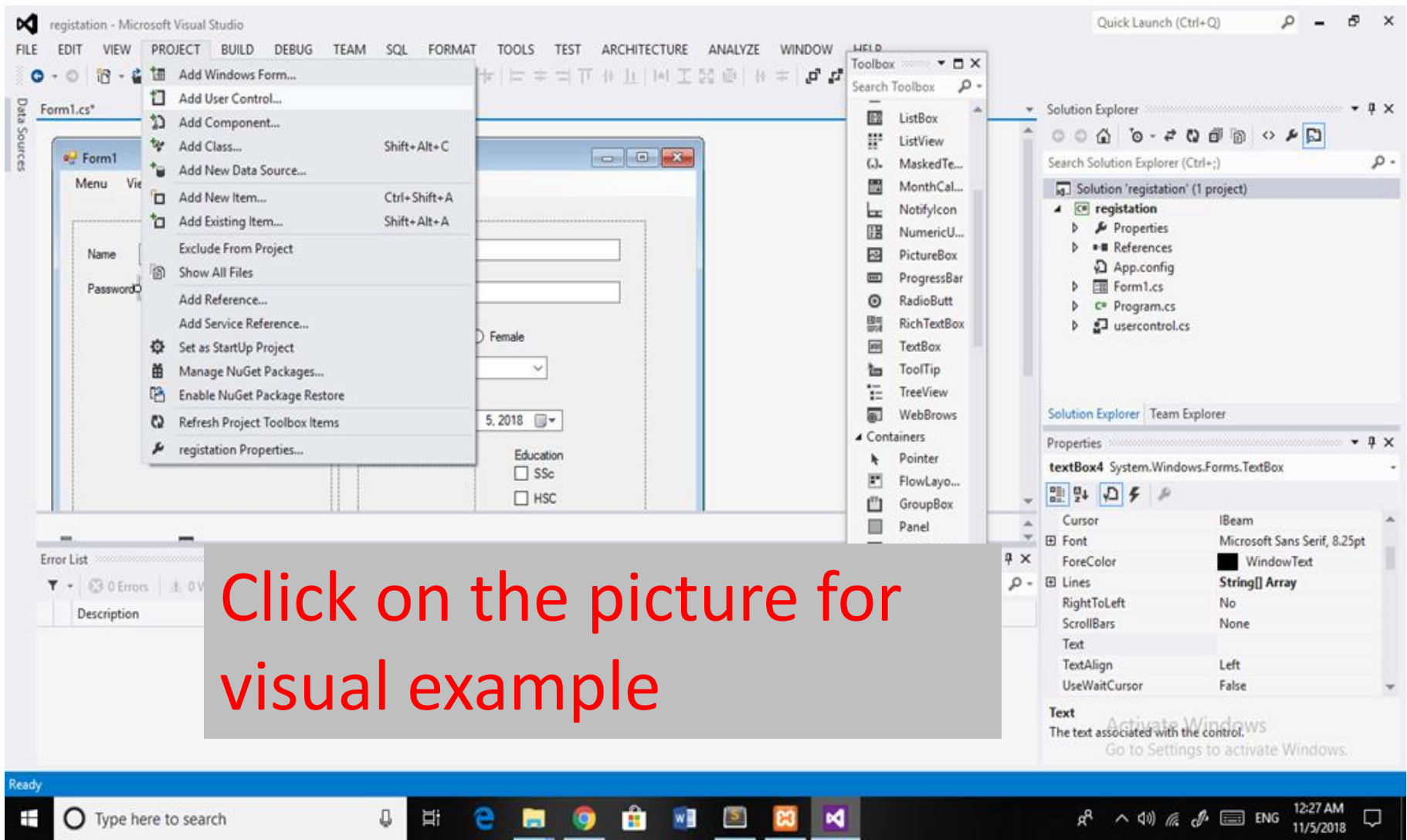


# User Controls

## Advantages



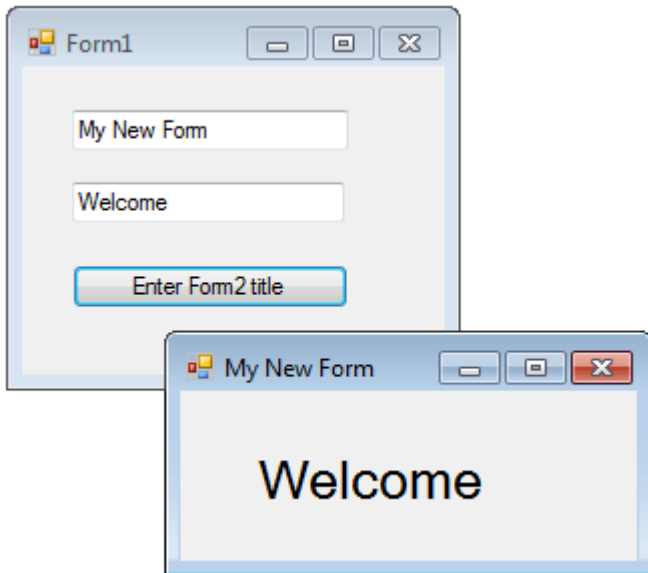
- Code re-usability.
- Time saving.
- Less effort.
- Easy to find Bug and fix it.
- Save memory also.



# Multiple Forms



In order to communicate between the forms, we need to use the Forms constructor to send these values. Constructors performs initialize the data members of the new object and it has the same name of the class.



```
public MyNewForm(string title,string txt)
{
    InitializeComponent();
    this.Text = title;
    label1.Text = txt;
}
```

# Conclusion



These are some common *Controls* that are shown in above slides. You can search for other controls and their details using the books and references.



# *Thank You*





# Books

- C# 4.0 The Complete Reference; Herbert Schildt; McGraw-Hill Osborne Media; 2010
- Head First C# by Andrew Stellman
- Fundamentals of Computer Programming with CSharp – Nakov v2013



# References

- <https://docs.microsoft.com/en-us/visualstudio/ide/step-1-create-a-windows-forms-application-project?view=vs-2019>
- <https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2019>
- <https://www.guru99.com/c-sharp-windows-forms-application.html>
- [https://en.wikipedia.org/wiki/Windows\\_Forms](https://en.wikipedia.org/wiki/Windows_Forms)
- <https://www.geeksforgeeks.org/introduction-to-c-sharp-windows-forms-applications/>
- [http://csharp.net-informations.com/gui/cs\\_forms.htm](http://csharp.net-informations.com/gui/cs_forms.htm)