GANPAT UNIVERSITY

U. V. PATEL COLLEGE OF ENGINEERING

2CEIT403

APPLICATION DEVELOPMENT TOOLS

UNIT 3

WINDOWS FORMS AND CONTROLS

Prepared by: Prof. Y. J. Prajapati (Asst. Prof in IT Dept. , UVPCE)

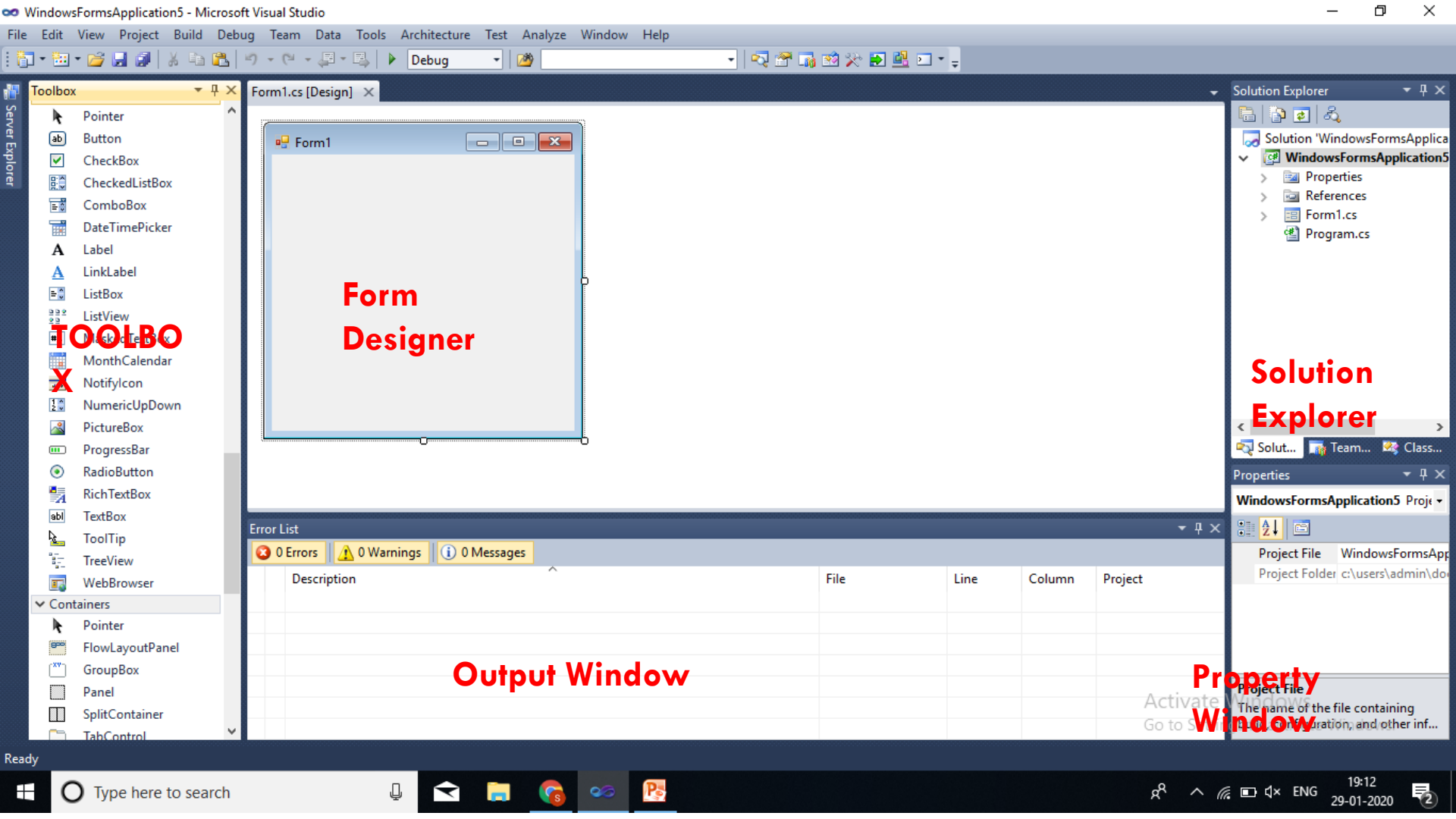
Outline

- Creating Windows Forms
- Windows Forms Properties and Events
- Controls (Properties and Events of Controls):
  - Button, Label, TextBox,
  - NumericUpDown ,Checkbox, RadioButton
  - DateTimePicker, GroupBox
  - ListBox, ListView, ComboBox,
  - TabControl, PictureBox, ProgressBar,
  - RichTextbox, Timer,DatagridView

Windows Forms Application

- A Windows Forms Application is a type of application that has a **graphical user interface.**
- **Controls** are **elements of** GUI application.
- **Use of forms** to build **user interfaces**. Each time you create a Windows application, Visual Studio will display a **default blank form**, onto **which you can drag the controls onto your applications** main form and adjust **their size and position.**
- The form can be viewed in two modes:

  Design View and **Code View**


WindowsFormsApplication5 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Debug
Toolbox
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
WebBrowser
Containers
Pointer
FlowLayoutPanel
GroupBox
Panel
SplitContainer
TabControl
TOOLBOX
Form1.cs [Design]
Form1
Form Designer
Solution Explorer
Solution 'WindowsFormsApplica
WindowsFormsApplication5
Properties
References
Form1.cs
Program.cs
Solut...
Team...
Class...
Solution Explorer
Properties
WindowsFormsApplication5 Proje
Project File WindowsFormsApp
Project Folder c:\users\admin\do
Project File
The name of the file containing
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
Output Window
Property Window
Ready
Type here to search
ENG
19:12
29-01-2020

Partial Classes

- It is **possible to split the definition** of a class or a **struct, an interface or a method** over **two or more source files.**
- Each source file contains a **section of code** or **method definition**, and **all parts are combined when the application is compiled.**
- Use the **partial keyword as modifier** to split a class definition.

Partial Classes

```
public partial class Employee
{
        public voidDoWork()
        {
        }
}
public partial class Employee
{
        public voidGoToLunch()
        {
        }
}
```

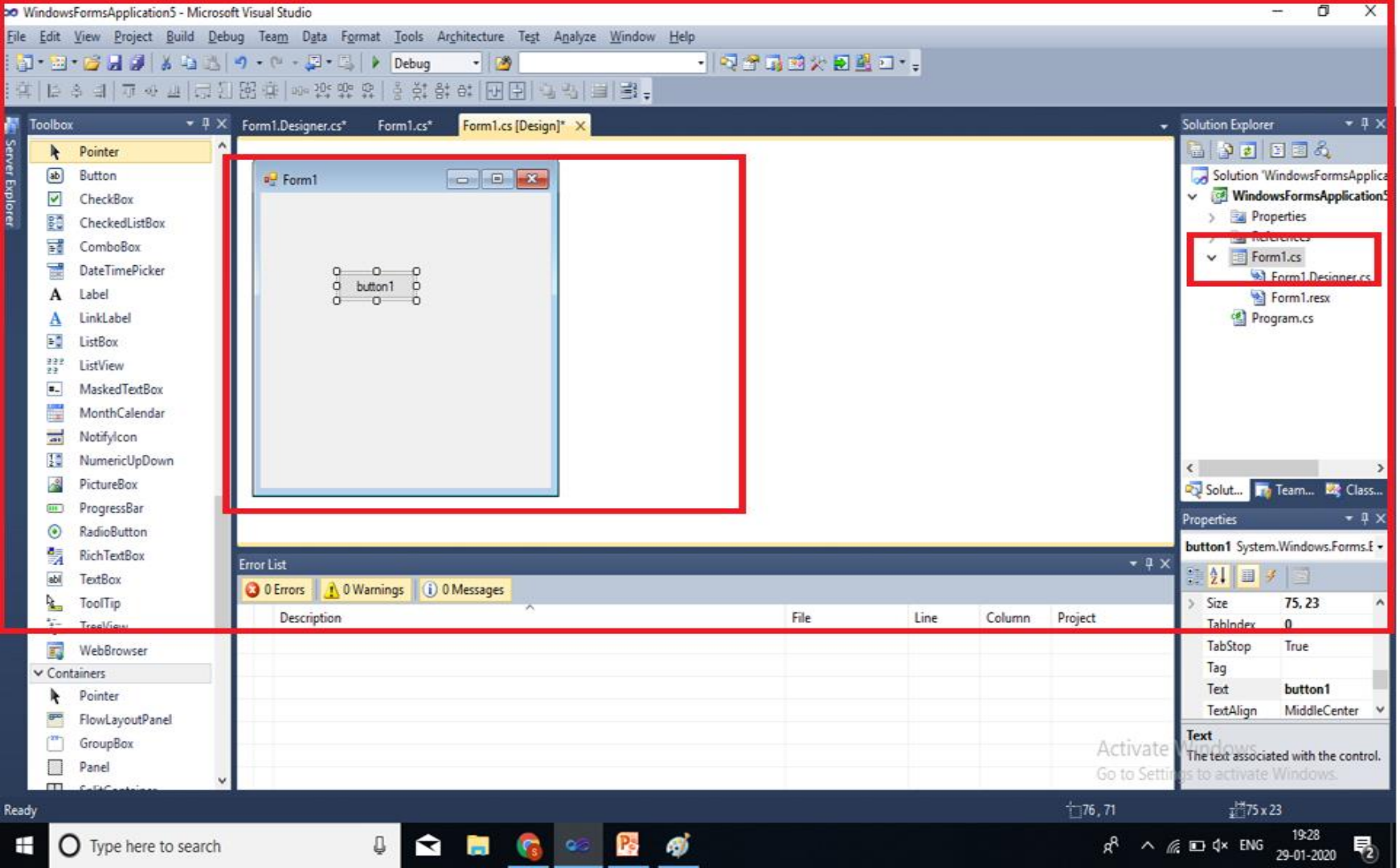
Partial Classes

- **When we create a new windows form, a class that inherits from System.Windows.Forms.Form is generated.**
- This class is separated in two files.

  FormName.cs

  FormName.Designer.cs

  where FormName is the name of the form.


WindowsFormsApplication5 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Format Tools Architecture Test Analyze Window Help
Debug
Toolbox
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
WebBrowser
Containers
Pointer
FlowLayoutPanel
GroupBox
Panel
Form1.Designer.cs*
Form1.cs*
Form1.cs [Design]*
Form1
button1
Solution Explorer
Solution 'WindowsFormsApplica
WindowsFormsApplication5
Properties
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Solut...
Team...
Class...
Properties
button1 System.Windows.Forms.E
Size 75, 23
TabIndex 0
TabStop True
Tag
Text button1
TextAlign MiddleCenter
Text
The text associated with the control.
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
Activate Windows
Go to Settings to activate Windows.
Ready
76 , 71
75 x 23
Type here to search
ENG
19:28
29-01-2020

Form1.Designer.cs* | Form1.cs* | Form1.cs [Design]*

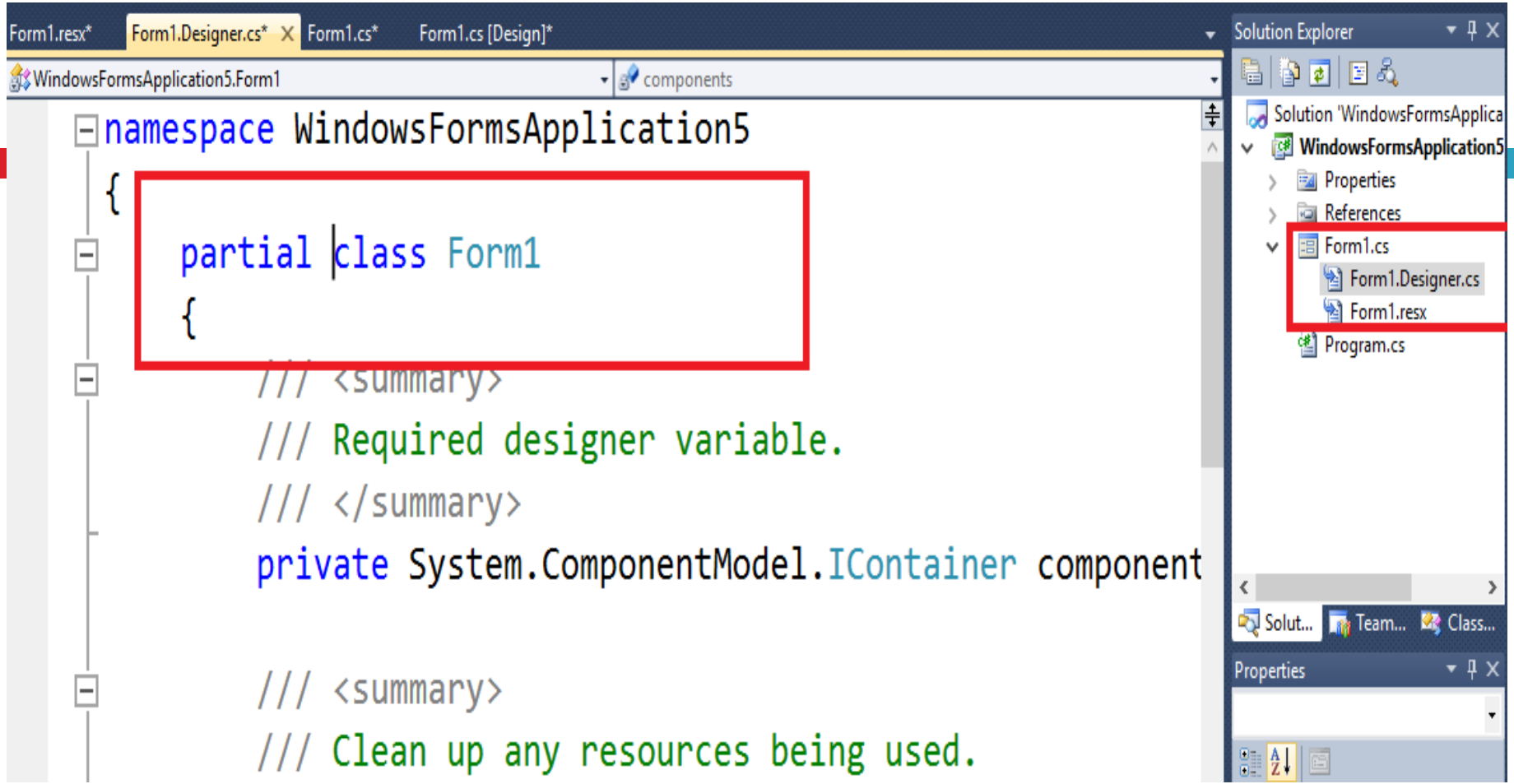
WindowsFormsApplication5.Form1 | button1_Click(object sender, EventArgs e)

```
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication5
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
```

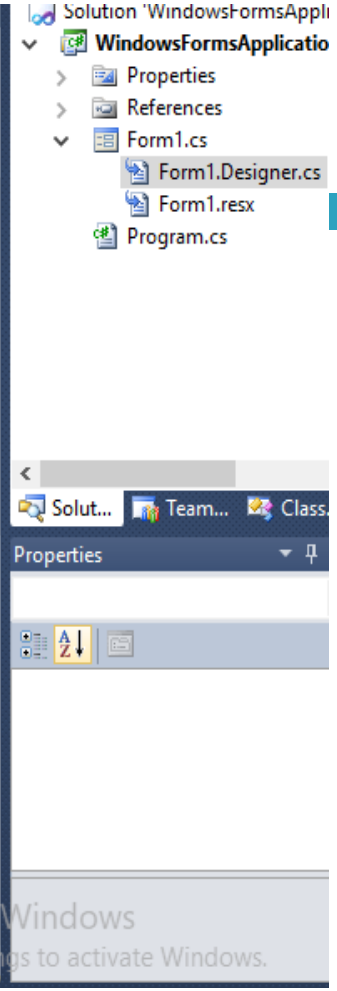

Form1.resx*
Form1.Designer.cs*
Form1.cs*
Form1.cs [Design]*
WindowsFormsApplication5.Form1
components
namespace WindowsFormsApplication5
{
partial class Form1
{
/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.IContainer component
/// <summary>
/// Clean up any resources being used.
Solution Explorer
Solution 'WindowsFormsApplica
WindowsFormsApplication5
Properties
References
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Solut...
Team...
Class...
Properties

```
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be dispose
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        Windows Form Designer generated code

        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.CheckBox checkBox1;
    }
}
```



Files of Windows Form Application

1. **form1.cs:** It is the **code-behind** file of the windows form. It is the class file of the windows form where the necessary **methods, functions also event driven methods** are written.

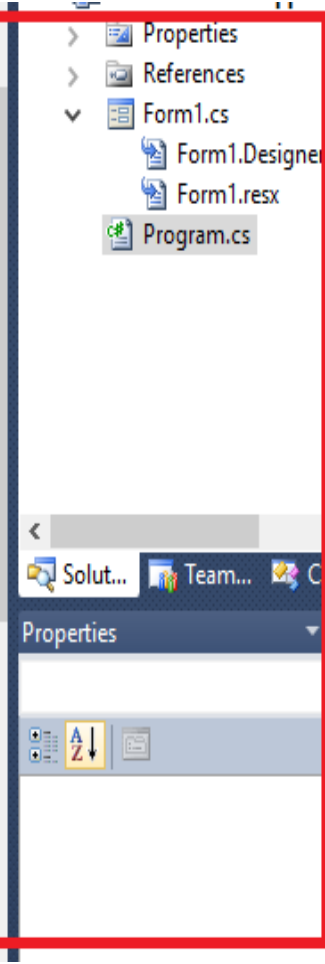
2. **form1.designer.cs:** It is the **designer file** where **form elements are initialized.** If any element is **dragged and dropped in the form window then** that element **will be automatically initialized in this class.**

3. **Form1.resx:** It is **useful for resources** of the form.

Files of Windows Form Application

4. **Program.cs:**
   - It contains **Main Method**.
   - The **Main Method is the entry point** for your program.
   - The code between the curly brackets of Main will get executed when the program first starts.

```
namespace WindowsFormsApplication5
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```



The Application Class

- It Provides **static methods** and **properties** to **manage an application,** such as **methods to start and stop an application.**
- This **class** cannot be **inherited.**
- The **Application class** provides the **overloaded Run()** method that can be **used to start** a program.
- One of the versions of this method **takes a form as argument**. This **form must be the first**, **main** or **primary form** of your **application**; it will be the **first to display** when the application comes up.

Adding Controls to the Form

- **All the controls are located** in **the Toolbox**.
- To add controls to the form, **choose a control in the Toolbox and double click it.**
- **Alternatively**, you can drag the control from the Toolbox to the form.
- **InitializeComponent() :**
  - **code for initializing the controls** and their **properties and events** is located inside a method called **InitializeComponent**.
  - **This method is called in the form's constructor** located at the main code file for **the form's class.**

Changing the Properties of Controls

- User can change **certain properties** of the **form and controls** using **properties window**.
- **Properties Window allows to view** and **change the value** of all the **available properties** of a **selected control** in the **Design View.**
- **Also user can change** or set properties of **controls** in **form load event**.

Event Handling

- Windows application uses **basic event-handling**.
- **An event is an action** recognized by a **form or control.**
- Events include both **user-generated actions** like mouse clicks and keystrokes and **system-generated events** such as program loading.
- GUI-based events: Mouse move, Mouse click, Mouse double-click, Key press, Button click, Menu selection, Change in focus, Window activation.

event-driven programming

- **event-driven programming** is a programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs/threads.
- Each form and control in Windows has a predefined set of events.

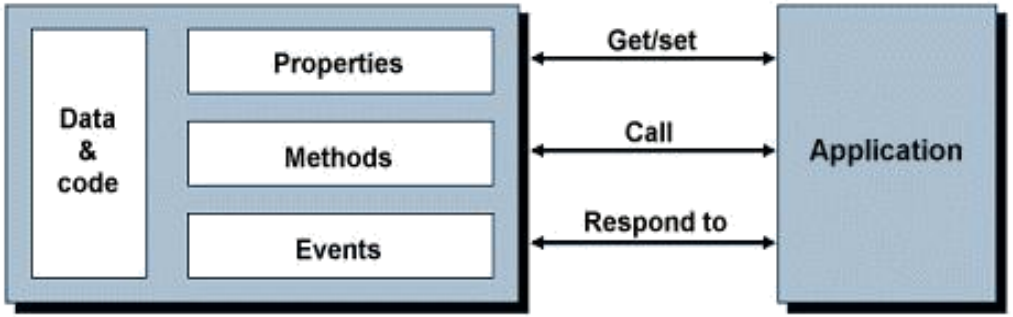
Sequence of events in an event-driven application:

1. The application starts and a form is loaded and displayed.

2. The form receives an event. The event might be caused by the user (for example, a keystroke), by the system (for example, a timer event), or indirectly by your code (for example, a Load event when your code loads a form).

3. If there is code in the corresponding event procedure, it executes.

4. The application waits for the next event.



The basic structure of an object and its relationship to the application

Parameters of Event

sender:

- It is an object which represents the control that sent by the event.
- It is used to access useful properties of control.
- Example:

```
private void button1_Click(object sender, EventArgs e)
{
   Button source = (Button)sender;
   MessageBox.Show("The message inside the button is " + source.Text);
}
```

Event Arguments:

- The second argument is an instance of the EventArgs class.
- Example:

```
private void button1_MouseClick(object sender, MouseEventArgs e)
    {
            int x1=e.X;
            int y1=e.Y;
            label1.Text = "Mouse Clicked at " + x1 + " , " + y1;
    }
```

- It contain data about the event that happened.
- The EventArgs is actually a base class and contains no useful members. Certain events will have event arguments that are derived from EventArgs and contain useful properties that the event handler can use.

Windows Form: Properties

Form properties:

AcceptButton	Gets or sets the button on the form that is clicked when the user presses the ENTER key.
AutoScroll	Gets or sets a value indicating whether the form enables autoscrolling.
BackColor	Getsorsetsthebackgroundcolorforthecontrol. (Overrides Control.BackColor.)
BackgroundImage	Gets or sets the background image displayed in the control. (Inherited from Control.)
Icon	Gets or sets the icon for the form.

Windows Form: Properties

Form properties:

CancelButton	Gets or sets the button control that is clicked when the user presses the ESC key.
Location	Gets or sets the Point that represents the upper-left corner of the Form in screen coordinates.
Size	Gets or sets the size of the form.
Text	Gets or sets the text associated with this control. (Overrides Control.Text.)
MinimizeBox	Gets or sets a value indicating whether the Minimize button is displayed in the caption bar of the form.
MinimumSize	Gets or sets the minimum size the form can be resized to. (Overrides Control.MinimumSize.)
Font	Gets or sets the font of the text displayed by the control. (Inherited from Control.)
ForeColor	Gets or sets the foreground color of the control.
Name	Gets or sets the name of the control.

Windows Form: Events

1. Load
   - This event occurs before a form is displayed for the first time.
   - You can set properties in the Load event handler.
   - This is also a good place to put some initialization code.
2. Shown
   - This event occurs whenever the form is displayed first time.
   - minimizing, maximizing, restoring, hiding, showing, or invalidating and repainting will not raise this event.

Windows Form: Events

3. Activated

- Occurs when the form is activated in code or by the user. The Activate event fires when the user switches to your form.
- If the user switches to a different program (or form), then switches back to your form, activate will fire again.
- When the application is active and has multiple forms, the active form is the form with the input focus.
- A form that is not visible cannot be the active form. The simplest way to activate a visible form is to click it or use an appropriate keyboard combination.

Windows Form: Events

4. Closing

- Occurs before the form is closed.
- In Form Closing, you can use logic to see if the program should stay open.
- Set the Cancel property on the FormClosingEventArgs to true and the Form will remain open.
- If you cancel this event, the form remains opened. To cancel the closure of a form, set the Cancel property of the FormClosingEventArgs passed to your event handler to **true**.

Windows Form: Events

5. Closed

- The Form Closed event occurs after the form has been closed by the user or by the Close method or the Exit method of the Application class.
- You can use this event to perform tasks such as freeing resources used by the form and to save information entered in the form or to update its parent form.

Windows Form: Methods

1. ShowDialog() and Show():

- These methods are used in windows applications to call one form from another.
- Eg. We have Form1 and we want to call Form2.

```
Private void button1_click(Object s, EventArgs e)
{
    Form2 f= new Form2();
    f.show();
}
Private void button1_click(Object s, EventArgs e)
{
    Form2 f= new Form2();
    f.ShowDialog();
}
```

- **Difference:**

Show() method does not make the target form (Form2) as model dialog box. ShowDialog() method will make Form2 as model dialog box. So, When we use ShowDialog() method, we cannot click anywhere on Form1 unless we close the instance of Form2. In case of Show() method, we can close Form1 even when Form2 is open.

Windows Form: Methods

2. Close():

- Closes the form.
- You can completely close a form and release the resources it consumes.
- You should close a form when it's no longer needed, so Windows can reclaim all resources used by the form.
- To do so, you invoke the Close method of the form like this: calls the Closing() and Closed() events

  Example: this.Close();

3. Hide():

- Conceals control from the user.

Windows Form
data Passing between forms

Method 1: Using constructor

- A method is invoked whenever user creates an object of second form class, this method is called a constructor.
- Code a constructor for form2 class with one string parameter.
- In the constructor, assign the text to the label's text property. Instantiate form2 class in form1's button click event handler using the constructor with one string parameter and pass the textbox's text to the constructor.

Form1	Form2

private void button1_Click(object sender,
System.EventArgs e)
{
Form2 frm=new Form2(textBox1.Text);
frm.Show();
}
	public Form2(string strTextBox)
	{
	InitializeComponent();
	label1.Text=strTextBox;
	}

Windows Form
data Passing between forms

Method 2: Using Properties

- Properties allow clients to access class state as if they were accessing member fields directly, while actually implementing that access through a class method.
- In this method, add one property to each form. In form1 use one property for retrieving value from the textbox and in form2, one property to set the label's text property. Then, in form1's button click event handler, instantiate form2 and use the form2's property to set the label's text.

Form1	Form2

public string _textBox1
{
get{return textBox1.Text;}
}
private void button1_Click(object sender, System.EventArgs e) {
Form2 frm=new Form2();
frm._textBox=_textBox1;
frm.Show();
}
	public string _textBox
	{
	set{ label1.Text=value;}
	}

Label

- Namespace : System.Windows.Forms.Label
- It is used to display read-only results.
- Label controls can also be used to add descriptive text to a Form to provide the user with helpful information.

Properties:

- **Text:** Gets or sets the text.

  e.g. label1.Text = "This is my first Label";
- **Font:** Gets or sets the font of the text displayed by the control.

  e.g. dynamicLabel.Font = new Font("Georgia", 16);
- **ForeColor:** Gets or sets the foreground color of the control.
  e.g.label1.ForeColor = Color.Blue;
- **Name:** Gets or sets the name of the control.

Buttons

- This control provides a way to accept input and invoke logic based on that input.

Properties:

Text: Gets or sets the display text of the Button.

E.g. button1.Text = "Click Here";

Enabled: It is used to make button enable(able to click) or disable(not able to click). E.g. button1.Enabled = false;

Visible button: It is used to change visibility of button.

E.g. button1.Visible = false;

BackColor: It is used to change the back color of button.

Name: Gets or sets the name of the control.

Events:

Click: It occurs when button is "pressed". This is the most widely using event.

```
private void button1_Click(object sender, EventArgs e)
{
}
```

TextBox

- A TextBox control accepts user input on a Form.
- A text box object is used to display text on a form or to get user input while a C# program is running.
- By default allows single line of Text.
- **Properties:**

Text	Gets or sets the current text in the TextBox. textBox1.Text = "I am TextBox";
Name	Gets or sets the name of the control.
Multiline	By default, a TextBox control accepts input in a single line only. To make it multi-line, you need to set Multiline property to true. By default, the Multiline property is false. textBox1.Multiline = true;
ReadOnly	You can make a TextBox control read-only (non-editable) by setting the ReadOnly property to true. dynamicTextBox.ReadOnly = true;

TextBox: Properties

MaxLength`	Gets or sets the maximum number of characters the user can type or paste into the text box control. **dynamicTextBox.MaxLength = 50;**
ScrollBars	• A Multiline TextBox control can have scrollbars. • The ScrollBars property of TextBox control is used to show scrollbars on a control. • The ScrollBars property is represented by a ScrollBars enumeration has four values **Both, Vertical, Horizontal and None.** **dynamicTextBox.ScrollBars = ScrollBars.Both;**
Password Character	• It is used to apply masking on text of TextBox. • Textbox will automatically act as a password textbox, meaning command such as Ctrl Copy are disabled. **dynamicTextBox.PasswordChar='#';**
UseSystemPasswordChar	• This property accepts bool values that is switched to true to make any textbox act as a password textbox. • The difference is that characters will be masked with a PasswordChar set by the system. **dynamicTextBox. UseSystemPasswordChar=true;**

TextBox: Events

1. **TextChanged**
   - The TextChanged event is only triggered when the text is changed by user.
2. **Keydown event**
   - It is raised when user presses a key and holds it on the keyboard.
   - Example: when the person presses a key (when the keyboard first detects a finger on a key, this happens when the key is pressed down. It is called before the key value actually is painted.)
   - The program will display an alert when the Enter key is pressed.

```
private void textBox1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        MessageBox.Show("You press Enter Key");

    }
    if (e.KeyCode == Keys.CapsLock)
    {
        MessageBox.Show("You press Caps Lock Key");
    }
```

TextBox: Events

3. KeyPress

- Occurs when a character, space or backspace key is pressed while the control has focus. The KeyPress event is not raised by non-character keys other than space and backspace; however, the non-character keys do raise the KeyDown and KeyUp events.

4. KeyUp

- KeyUp is raised after the user releases a key on the keyboard.
- Key events occur in the following order:
  - KeyDown
  - KeyPress
  - KeyUp

TextBox: Events

5. **Enter Event**

```
private void textBox1_Enter(object sender, System.EventArgs e)
{
// If the TextBox contains text, change its foreground and background
colors. if (textBox1.Text != String.Empty)
    {
        textBox1.ForeColor = Color.Red;
        textBox1.BackColor = Color.Black;
// Move the selection pointer to the end of the text of the control.
textBox1.Select(textBox1.Text.Length, 0);
    }
}
```

6. **Leave Event**

- Occurs when the input focus leaves the control.

```
private void textBox1_Leave(object sender, System.EventArgs e)
{
// Reset the colors and selection of the TextBox after focus is lost.
    textBox1.ForeColor = Color.Black;
    textBox1.BackColor = Color.White;    textBox1.Select(0,0);
}
```

ListBox

- ListBox stores and display several text items.
- Users can select one or more items from the list.
- The ListBox also provides features that enable you to efficiently add items to the ListBox.

ListBox

Q: How to add Items in ListBox?

1. Using **Add()** : Adds an item to the list of items for a ListBox.

```
private void Form2_Load(object sender, EventArgs e)
{
    this.listBox1.Items.Add("ADT");
    this.listBox1.Items.Add("OS");
}
```

2. Using **Insert(index,item name):** Inserts an item into the list box at the specified index.

```
private void Form2_Load(object sender, EventArgs e)
{
    this.listBox1.Items.Insert(2, "Maths");
}
```

ListBox

Q: How to set Multiple selection in listbox?

- **SelectionMode Property** : Gets or sets the method in which items are selected in the ListBox.
- Values of SelectionMode are following:

MultiExtended	Multiple items can be selected, and the user can use the SHIFT, CTRL, and arrow keys to make selections
MultiSimple	Multiple items can be selected.
None	No items can be selected.
One	Only one item can be selected.

- E.g.: listBox1.SelectionMode = SelectionMode.MultiSimple;

ListBox: Properties

Property	Description
Items	Gets or sets the items from ListBox.
Count	Gets the number of items in the collection. e.g. int count = listBox1.Items.Count;
Sorted	Gets or sets a value indicating whether the items in the ListBox are sorted alphabetically. e.g. listBox1.Sorted = true;
SelectionMode	Gets or sets the method in which items are selected in the ListBox.
SelectedItem	Gets the currently selected item in the ListBox. e.g. string a = listBox1.SelectedItem.ToString();
SelectedItems	Gets collection containing the currently selected items in the ListBox.

ListBox: Methods

Method	Description
Add()	Adds an item to the list of items for a ListBox.
Insert (index, item name)	Inserts an item into the list box at the specified index.
Remove(item)	Removes the specified object from the collection. listBox1.Items.Remove(listBox1.SelectedItem);
Clear()	Removes all items from the ListBox. listBox1.Items.Clear();
FindStringExact (string)	Finds the first item in the ListBox that exactly matches the specified string. Return Value is the zero-based index of the first item found; returns -1 if no match is found.
SetSelected(index,value)	Selects or clears the selection for the specified item in a ListBox. Value has boolean type to select or clear the specified item.
ClearSelected()	Unselects all items in the ListBox. listBox1.ClearSelected();
GetSelected (index)	Returns a value indicating whether the specified item is selected. bool b= listBox1.GetSelected(0);

ListBox: Examples

Q: How to get all items of listbox?

```
string a="“;
for (int i = 0; i < listBox1.Items.Count; i++)
{
a =a+ listBox1.Items[i].ToString()+"\n";
}
MessageBox.Show(a);
```

Q: How to get selected items of listbox?

```
string a = "“;
for (int i = 0; i < listBox1.SelectedItems.Count; i++)
{
a =a+ listBox1.SelectedItems[i].ToString()+"\n“;
}
MessageBox.Show(a);
```

ListBox: Examples

Q: How to remove multiple selected items?

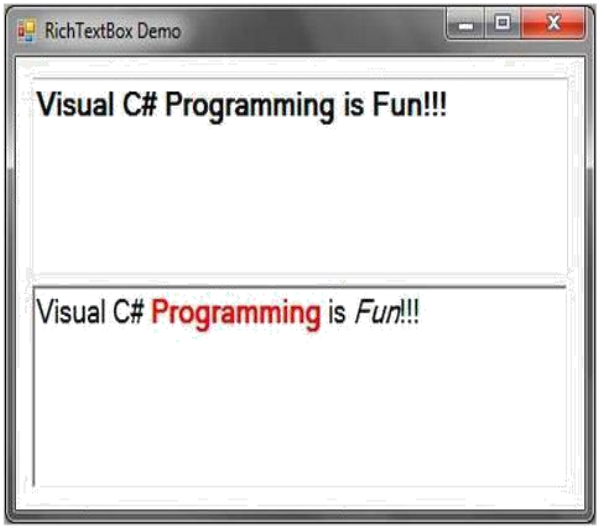
```
for (int i = listBox1.SelectedItems.Count-1; i >= 0; i--)
{
this.listBox1.Items.Remove(listBox1.SelectedItems[i]);
}
```

Q: How to search item in listbox?

```
string a = textBox1.Text;
int index=listBox1.FindStringExact(a);
if (index > -1)
{
MessageBox.Show("Found at index = "+ index.ToString());
listBox1.SetSelected(index, true);
}
 else
 {
 MessageBox.Show("Not found");
}
```

RichTextBox

- The RichTextBox control (System.Windows.Forms.RichTextBox) is similar to a TextBox control but it allows you to format different parts of the text inside it.
- You can also use a rich text box control to accept input from the user.
- RichTextBox control is used to show formatted text and save it in Rich Text Format (RTF).



RichTextBox: Properties

DetectUrls	Specifies whether to automatically detect and add the target link to urls inside the RichTextBox control
ReadOnly	Tells whether the RichTextBox control is read-only.
Rtf	Contains the text of the RichTextBox control including the rich text format (rtf) codes.
Scrollbars	Specifies the type of scrollbars to display.
SelectedText	Gets or sets the selected text within the RichTextBox.
SelectionBackColor	Specifies the background color of the selected text. richTextBox1.SelectionBackColor = Color.Yellow
SelectionBullet	Specifies whether the bullet style is applied to the current selection or insertion point.
SelectionColor	Specifies the color of the selected text. richTextBox1.SelectionColor = Color.Red;
Text	The plain text inside the RichTextBox control.
WordWrap	Specifies whether to wrap the text inside the RichTextBox control.

RichTextBox: Methods & Events

Event:

- LinkClicked(): Occurs when a link was clicked.

Method:

- Select(starting index, length): It is used to select texts.

  e.g.: richTextBox1.Select(10, 11);

FileStream

- The FileStream Class represents a File in the computer which is useful to read, write, open and close files on a file system.
- FileStream allows to move data to and from the stream as arrays of bytes.
- Namespace required : System.IO
- Syntax of Constructor:

  public FileStream (string path, FileMode mode);

FileMode: specifies the operation to perform on the file.

FileMode.Append: It opens an existing file and puts cursor at the end of file, or creates the file, if the file does not exist.

FileMode.Create: Create new file and open it for read and write, if the file exists overwrite it.

FileMode.CreateNew: Create a new File , if the file exist , it throws exception.

FileMode.Open: Open an existing file.

```
FileStream fs = new FileStream(@"c:\file.txt", FileMode.Open);
```

StreamReader & StreamWriter

- Namespace: System.IO namespace.
- Both classes are useful when user want to read or write character based data.
- Both of these classes deal with Unicode characters.
- StreamReader derives from the Abstract class "TextReader" and StreamWriter derives from "TextWriter".
- Implements a TextReader that reads characters from a byte stream in a particular encoding.

StreamWriter

- StreamWriter Implements a TextWriter for writing characters to a stream in a particular encoding.
- **Syntax:**

  public StreamWriter(Stream stream)
- This constructor takes the first argument the stream, the stream which writes data.
- Example:

```
FileStream fs = new FileStream(textBox1.Text, FileMode.Create);
StreamWriter sw = new StreamWriter(fs);
string s = richTextBox1.Text;
sw.Write(s.Replace("\n",Environment.NewLine));
```

Member	Description
Close()	Closes the current StreamWriter object and the underlying stream. This method is equivalent to Dispose(), that is used to release resources.
Write()	This method is used to write data to a text stream without a newline.
WriteLine()	This method is used to write data to a text stream with a newline.

StreamReader

- StreamReader reads text files.
- Reads all characters from the current position to the end of the stream.
- **Syntax:**

  public StreamReader (Stream stream)
- This constructor takes one argument the stream, the stream reads data.
- Example:

```
FileStream fs = new FileStream(@"c:\file.txt", FileMode.Open);
StreamReader sr = new StreamReader(fs);
string s= sr.ReadToEnd();
```

Member	Description
Close()	Closes the current StreamReader object and the underlying stream. This method is equivalent to Dispose(), that is used to release resources.
Read()	Reads the next character from the input stream.
ReadLine()	Reads a line of characters from the current stream and returns the data as a string.
ReadToEnd()	Reads the stream from the current position to the end of the stream.

Listview

- ListView = ListBox + View property
- This property allows you to specify a predefined way of displaying the items.
- The listbox control is used for displaying a list of items only, while the listview control can show items with icons in different views (large icon, small icon, list, details, report), show data with multiple columns and provide formatting options for the items.

Listview

Property	Description
View	Gets or sets how items are displayed in the control. listView1.View = View.Details; Values: Details, LargeIcon, SmallIcon, List, Tile
GridLines	Gets or sets a value indicating whether grid lines appear between the rows and columns containing the items and sub items in the control. listView1.GridLines = true;
FullRowSelect	It will indicating whether clicking an item selects all its sub items. listView1.FullRowSelect= true;
HideSelection	Gets or sets a value indicating whether the selected item in the control remains highlighted when the control loses focus. listView1.HideSelection = true;
MultiSelect	Gets or sets a value indicating whether multiple items can be selected. ListView1.MultiSelect = False

Listview

Property	Description
LabelEdit	Gets or sets a value indicating whether the user can edit the labels of items in the control. ListView1.LabelEdit = true;
CheckBoxes	Gets or sets a value indicating whether a check box appears next to each item in the control. ListView1. CheckBoxes=true;

ListView: **Events**

- **SelectedIndexChanged** : Occurs when the Selected Indices collection changes.

ComboBox

- A ComboBox control is a combination of TextBox and ListBox control.
- Only one list item is displayed at one time in a ComboBox and other available items are loaded in a drop down list.
- The ComboBox control is used to display a drop-down list of various items. It is a combination of a text box in which the user enters an item and a drop-down list from which the user selects an item.

ComboBox

Add item to combobox:

```
comboBox1.Items.Add("Sunday");
comboBox1.Items.Add("Monday");
comboBox1.Items.Add("Tuesday");
```

Remove value from ComboBox:

- remove item by the specified index or specified item by name.
- E.g. comboBox1.Items.RemoveAt(1);
- E.g. comboBox1.Items.Remove("Friday");

ComboBox

Property	Description
Enabled	It makes combobox in read only. e.g. comboBox1.Enabled = false.
DropDownStyle	It specifies whether the list is always displayed or whether the list is displayed in a drop-down. e.g. comboBox1.DropDownStyle = ComboBoxStyle.DropDown; Possible values: DropDown, DropDownList, Simple
SelectedItem	It will display selected item in a combobox. e.g. comboBox1.SelectedItem = "Monday";

ComboBox-Event

SelectedIndexChanged event

- It will fire when user change the selected item in a combo box.
- If user want to perform any operation when selection is changed, write the program on SelectedIndexChanged event.

Radio Button

- User can select a single option from a group of choices.

Properties:

- **Checked** : Gets or sets a value indicating whether the control is checked.
- **Text :** Gets or sets the text associated with this control.

Events:

- **CheckChanged :** The most useful event is probably CheckChanged. The CheckedChanged event occurs when the value of the Checked property changes.

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
MessageBox.Show(radioButton1.Text);
}
```

CheckBoxes

CheckBoxes allow the user to make multiple selections from a number of options.

Properties:

- **CheckAlign** : Gets or sets the horizontal and vertical alignment of the check mark on a CheckBox control.

  checkBox1.CheckAlign = ContentAlignment.TopRight;

- **Checked** : Gets or set a value indicating whether the CheckBox is in the checked state.

  checkBox1.Checked = false;

- **Text:** CheckBox comes with a caption, which you can set in the Text property. checkBox1.Text = "Net-informations.com";

Events:

- **CheckChanged :** The most useful event is probably CheckChanged. The CheckedChanged event occurs when the value of the Checked property changes.

GroupBox

Represents a Windows control that displays a frame around a group of controls with an optional caption.

Properties:

- **Text :** Gets or sets the text associated with this control.

- **Dock :** These options allow you to tie the size of the control to the size of an enclosing control or the window itself.

  Possible values of Dock:

  - Top, Bottom, Left, Right: These will push the control to the selected edge.
  - Center: This will expand the control to fill the entire window.

PictureBox

PictureBox control is used to display images in bitmap, GIF , icon , or JPEG formats.

Properties:

- **Image** : User can set the Image for display using Image property.

  e.g. pictureBox1.Image = Image.FromFile("c:\\testImage.jpg");

- **Width and Height:** sets its width and height.

```
imageControl.Width = 400;
imageControl.Height = 400;
```

PictureBox

- **SizeMode :** It controls how the pictureBox will handle the image placement and control sizing. Indicates how the image is displayed.
- There are five different SizeMode is available to PictureBox control.
  - **AutoSize:** The PictureBox is sized equal to the size of the image that it contains.
  - **CenterImage:** The image is displayed in the center if the PictureBox is larger than the image. If the image is larger than the PictureBox, the picture is placed in the center of the PictureBox and the outside edges are clipped.
  - **Normal:** The image is placed in the upper-left corner of the PictureBox. The image is clipped if it is larger than the PictureBox it is contained in.
  - **StretchImage:** The image within the PictureBox is stretched or shrunk to fit the size of the PictureBox.
  - **Zoom:** The size of the image is increased or decreased maintaining the size ratio.

Message Box

- A MessageBox is a predefined dialog box that displays application-related information to the user.
- Message boxes are also used to request information from the user.

Method

- The Show method of the MessageBox class returns a value that can be used to determine a choice made by the user.
- **Show(String)**: Displays a message box with specified text.
- **Show(String, String)**: Displays a message box with specified text and caption.
- **Show(String, String, MessageBoxButtons, MessageBoxIcon)**: Displays a message box with specified text, caption, buttons, and icon.

Message Box

Show(String, String, MessageBoxButtons, MessageBoxIcon):

Displays a message box with specified text, caption, buttons, and icon.

MessageBox Buttons:	**MessageBox Icons:**
OK, OK, Cancel Yes, No Yes, No, Cancel Retry, Cancel Abort, Retry, Ignore	 Error Information Question Warning

Message Box

- **Retrun the value of a messagebox:**
- If the user clicks a button, the program gets a DialogResult.
- Here are the different return values and the buttons for the values:
  - OK --- DialogResult.OK
  - Yes --- DialogResult.Yes
  - No --- DialogResult.No
  - Abort --- DialogResult.Abort
  - Retry --- DialogResult.Retry
  - Cancel --- DialogResult.Cancel
  - Ignore --- DialogResult.Ignore

Example: MessageBox.Show("text", "caption", "buttons", "icon")

Timer

- Timer Object is used when we want to set an interval between events, periodic checking, to start a process at a fixed time schedule, to increase or decrease the speed in an animation graphics with time schedule etc.

- Timer that raises an event at user-defined intervals.
- A Timer control raises an event at a given interval of time without using a secondary thread. If you need to execute some code after certain interval of time continuously, you can use a timer control.

Timer

Property:

- **Enabled** : Enabled property of timer represents if the timer is running. We can set this property to true to start a timer and false to stop a timer.
- **Interval** : Interval property represents time on in milliseconds, before the Tick event is raised relative to the last occurrence of the Tick event. One second equals to 1000 milliseconds. So if you want a timer event to be fired every 5 seconds, you need to set Interval property to 5000.
- A Timer control does not have a visual representation and works as a component in the background.

Timer

Event:

Tick Occurs when the specified timer interval has elapsed and the timer is enabled.

```
private void timer1_Tick(object sender, EventArgs e)
 {

}
```

Method:

- **Start** : Starts the timer.
- **Stop** : Stops the timer.

Timer

Here we run this program display current time and date.

```
private void timer1_Tick(object sender, EventArgs e)
 {
   label1.Text = DateTime.Now.ToString();
}
```


THANK YOU!
ANY QUESTIONS?