UNIT – 3

Windows Programming

Programming with C#

Code : CS-23

MessageBox class with all types of Show() method

- A MessageBox is a type of windows form which is used to display some message to the user while running an application.
- MessageBox is a class in C#, and Show is a method that displays a message in a small window in the center of the Form.
- MessageBox is used to provide confirmations of a task being done or to provide warnings before a task is done.
- You can also use MessageBox control to add additional options such as a **caption, an icon, or help buttons**.
- The simplest form of a MessageBox is a dialog with a **text and OK**
- button. When you click OK button, the box disappears.
- **Note: By default the OK Button will be shown.**
- We can create a message box by using **MessageBox.Show** method.
- For display a message box we use show method of the **MessageBox** class.
- MessageBox.Show() returns object of DialogResult.
- it has **21** overloaded methods.

- **Syntax :**

  MessageBox.Show(String, Title, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions);

- **Example :**

  MessageBox.Show("Message”, "Title", MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button2, MesageBoxOptions.RightAlign, true);

- **Types of MessageBox Buttons :**

1. **OK:** It is defined as MessageBoxButtons.OK
2. **OK and Cancel:** It is defined as MessageBoxButtons.OkCancel.
3. **Abort Retry and Ignore:** It is defined as MessageBoxButtons.AbortRetryIgnore.
4. **Yes No and Cancel:** It is defined as MessageBoxButtons.YesNoCancel.
5. **Yes and No:** It is defined as MessageBoxButtons.YesNo.
6. **Retry and Cancel:** It is defined as MessageBoxButtons.RetryCancel.

- **Types of MessageBox Icons :**

1. **None:** No icons are displayed in the Message box.
2. **Hand:** A hand icon is displayed. It is defined as MessageBoxIcon.Hand.
3. **Question:** A question mark is displayed. It is defined as MessageBoxIcon.Question.
4. **Exclamation:** An exclamation mark is displayed. It is defined as MessageBoxIcon.Exclamation.
5. **Asterisk:** An asterisk symbol is displayed. It is defined as MessageBoxIcon.Asterisk.
6. **Stop:** A stop icon is displayed. It is defined as MessageBoxIcon.Stop.
7. **Error:** An error icon is displayed. It is defined as MessageBoxIcon.Error.
8. **Warning:** A warning icon is displayed. It is defined as MessageBoxIcon.Warning.
9. **Information:** An info symbol is displayed. It is defined as MessageBoxIcon.Information.

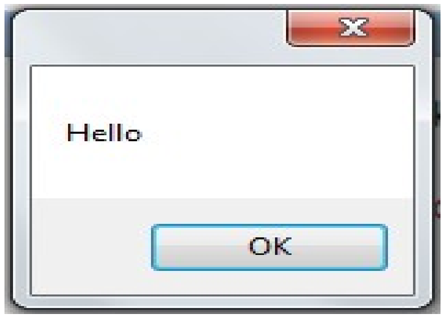
- **Types of MessageBoxDefaultButton Options**

1. **Button1 :** It makes the first button on the messagebox the default button.
2. **Button2 :** It makes the second button on the messagebox the default button.
3. **Button3 :** It makes the third button on the messagebox the default button.

- **Types of MessageBox Options**

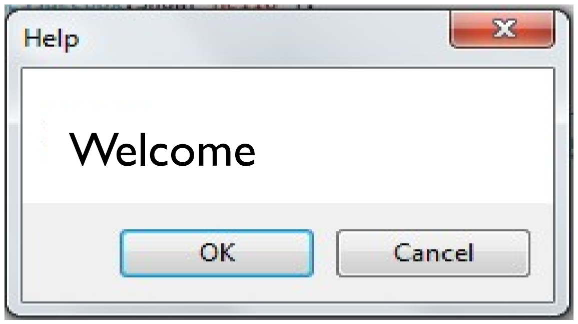
1. **ServiceNotification:** It is defined as MessageBoxOptions.ServiceNotification. This is used to display the message box on the current desktop which is active. The message box is displayed even when no user is logged on to the desktop.
2. **DefaultDesktopOnly:** It is defined as MessageBoxOptions.DefaultDesktopOnly. This also displays on the currently active desktop. The difference between this and service notification is that here the message is displayed on the interactive window.
3. **RightAlign**: It is defined as MessageBoxOptions.RightAlign. This is used to format the message in right alignment.
4. **RtlReading:** It is defined as MessageBoxOptions.RtlReading. This denotes that message is displayed from right to left order.

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



Hello

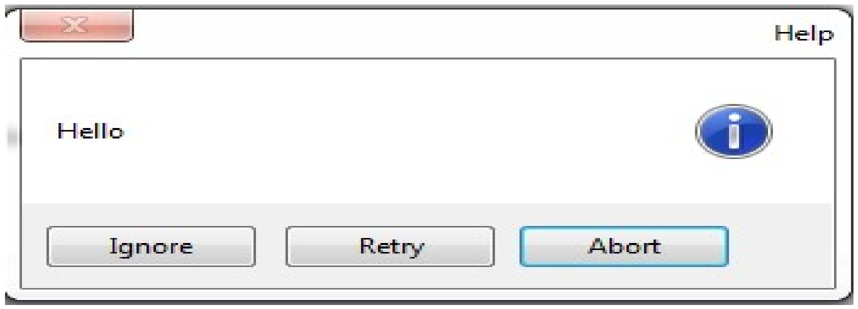
```
private void button3_Click(object sender, EventArgs e)
{
    string msg = "Welcome";
    string title = "Help";
    MessageBox.Show(msg, title, MessageBoxButtons.OKCancel);
}
```



Welcome

```
private void button3_Click(object sender, EventArgs e)
{

    string message = "Do you want to abort this operation?";
    string title = "Close Window";
    MessageBoxButtons buttons = MessageBoxButtons.AbortRetryIgnore;
    DialogResult result = MessageBox.Show(message, title, buttons, MessageBoxIcon.Warning);
    if (result == DialogResult.Abort)
    {
        this.Close();
    }
    else if (result == DialogResult.Retry)
    {
        MessageBox.Show("Retry plz");
    }
    else
    {
        MessageBox.Show("Ignore");
    }
}
```



Hello

Form and Properties :

- **Windows Form :**
- Desktop base application nothing but software, this software is made up of one or more forms, and the forms are what users see.
- Just like website contains Web Pages same desktop base application
- contains windows form.
- **Example:**
- What ATM machine is display is a Form, the main one form is called MDI Form and which forms we get from option of that MDI, and those are called Child form of MDI.
- **Key Facts of the Windows Form :**
- A from is a usually rectangular.
- The form is the primary platform for user interface.
- Forms can be Single Document Interface(SDI), Multiple Document Interface (MDI), dialog boxes etc.
- Form is container.
- Form has properties, events and methods.
- Using "this" keyword we access properties and method in coding window.

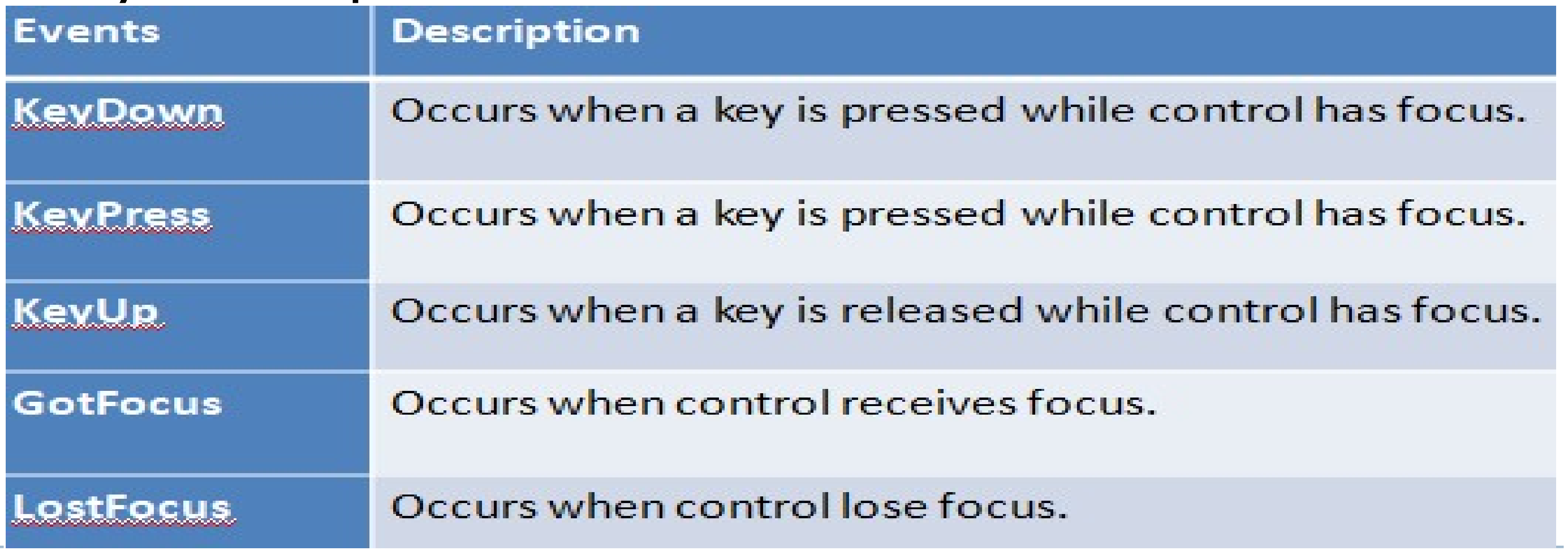
- **System.Windows.Forms** namespace contains various properties, methods, events etc. in the form class is System.Windows.Forms.Form.
- Before we go for discussion of specific classes, let us discuss three terms that are essential to understand the .NET Framework especially for Windows.Forms namespace.
- **There are three terms are component, container and control :**
- **1. component :** is an object that permits sharing between applications.
- The Component class encapsulates this notion, and is the basis for most of the members of the Windows Forms namespace.
- **2. container :** is an object that can hold zero or more components.
- A container is simply a grouping mechanism, and ensures that sets of components are encapsulated and manipulated in similar ways.
- **3. control :** is a component with a visual aspect.
- In the Windows Forms namespace, a control is a component that presents a graphical interface on the Windows desktop.
- It is worth noting that the System.Web.UI namespace defines a Control class as well to represent graphical objects that appear on web pages.
- Commonly used Properties of Form Control are as follows.

Properties of Control :

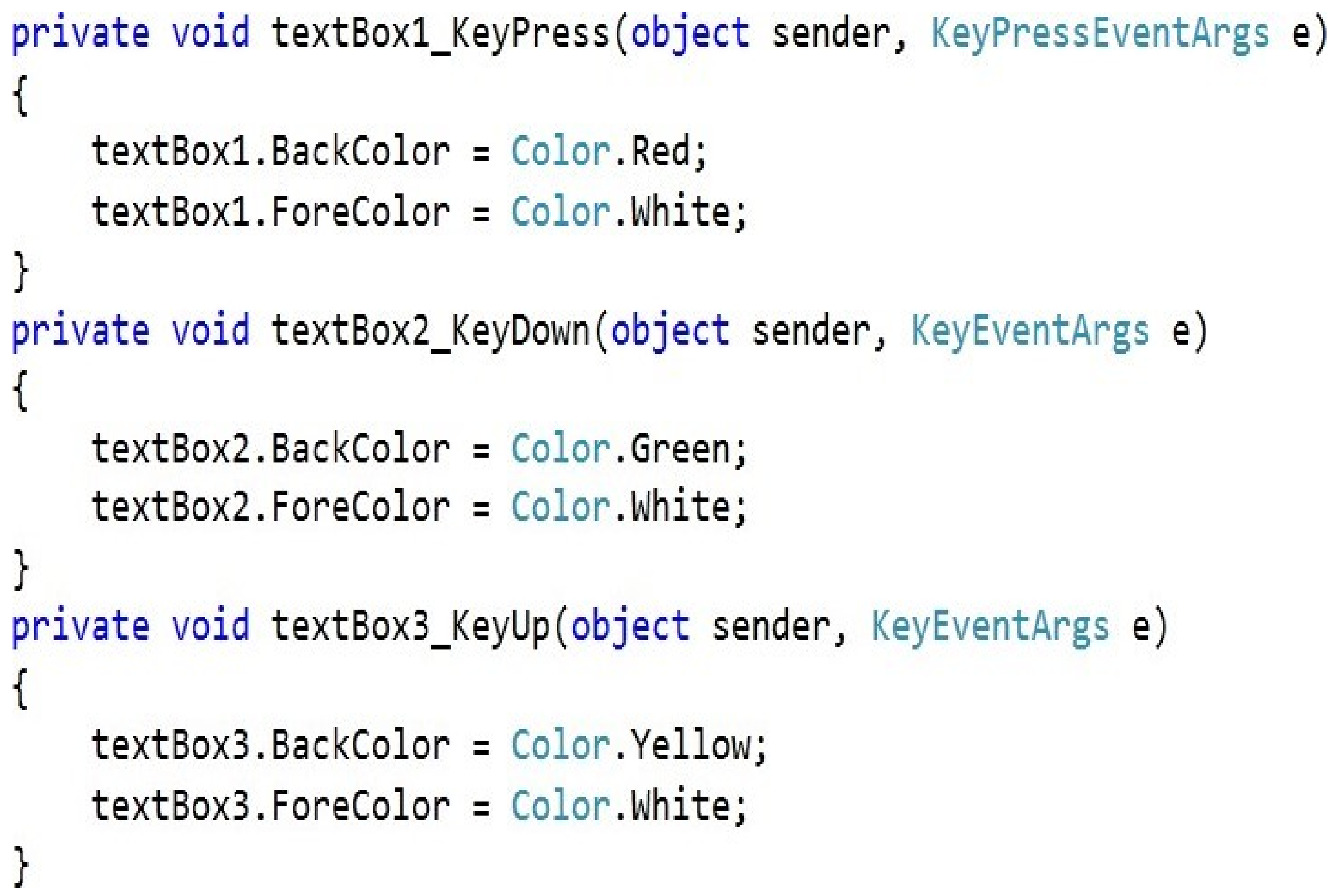
Property	Description
AcceptButton	Sets one button for Enter key.
AutoScroll	Its sets scrollbar to the form when its required.
CancleButton	Sets the one button for the ESC key.
Icon	Sets the icon for the form.
ControlBox	Sets max. mini. And cancle buttons. False will set all invisible
IsMdiContainer	Sets form as MDI container (child).
StartPosition	CenterParent, CenterScreen etc.
WindowState	Set starting position , Maximize, Minimize
ShowInTaskbar	Set true or false to show or hide on task bar.

Now let’s learns some EVENT’s :

- **There are Main two events Mouse Events & Key Events :**
- **1. Handling Key Events :**
- Most of every control handles its own keyboard input so you will need to handle keyboard input directly.
- For example you might want to perform an action when the user presses a specific button or releases the specific button.
- Most controls support three events that you can use to work directly with keyboard input. These are listed in table.



Events	Description
KeyDown	Occurs when a key is pressed while control has focus.
KeyPress	Occurs when a key is pressed while control has focus.
KeyUp	Occurs when a key is released while control has focus.
GotFocus	Occurs when control receives focus.
LostFocus	Occurs when control lose focus.

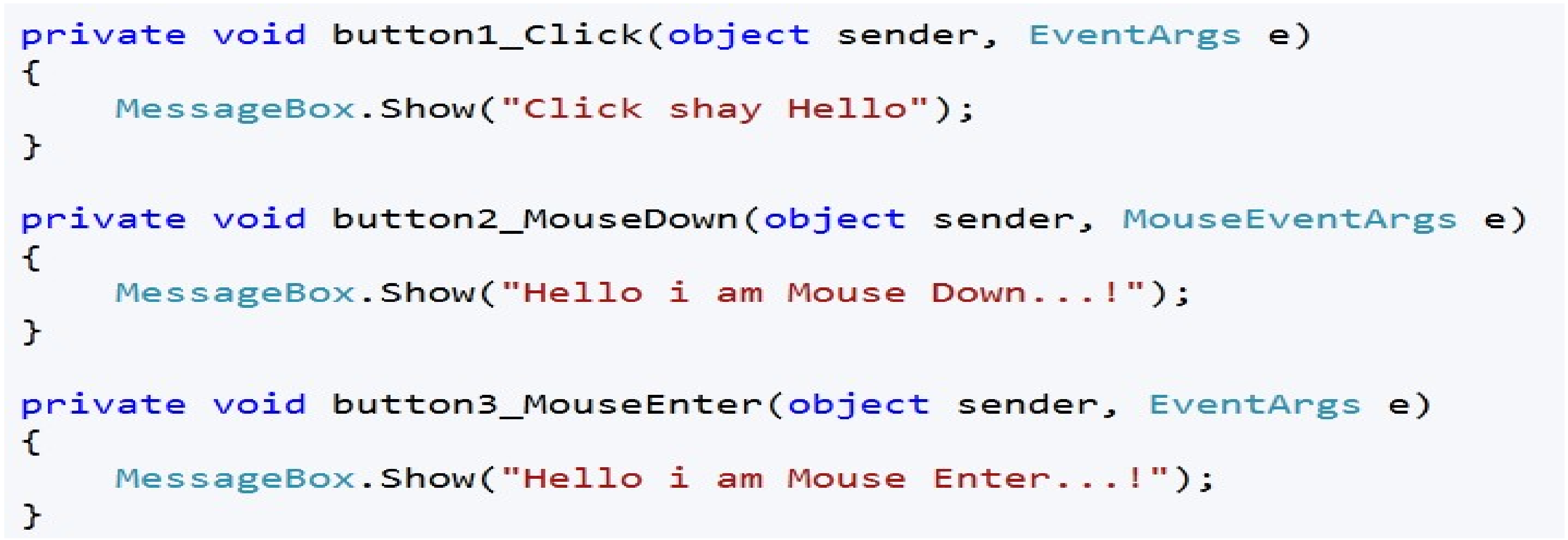


```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    textBox1.BackColor = Color.Red;
    textBox1.ForeColor = Color.White;
}
private void textBox2_KeyDown(object sender, KeyEventArgs e)
{
    textBox2.BackColor = Color.Green;
    textBox2.ForeColor = Color.White;
}
private void textBox3_KeyUp(object sender, KeyEventArgs e)
{
    textBox3.BackColor = Color.Yellow;
    textBox3.ForeColor = Color.White;
}
```

2. Handling Mouse Events :

- As with keyword input, most controls support mouse input natively, you don't have to write code to deal with mouse event.
- C# supports seven events that enable you to deal with mouse input directly. These events are listed in table, in the order in which they occur.

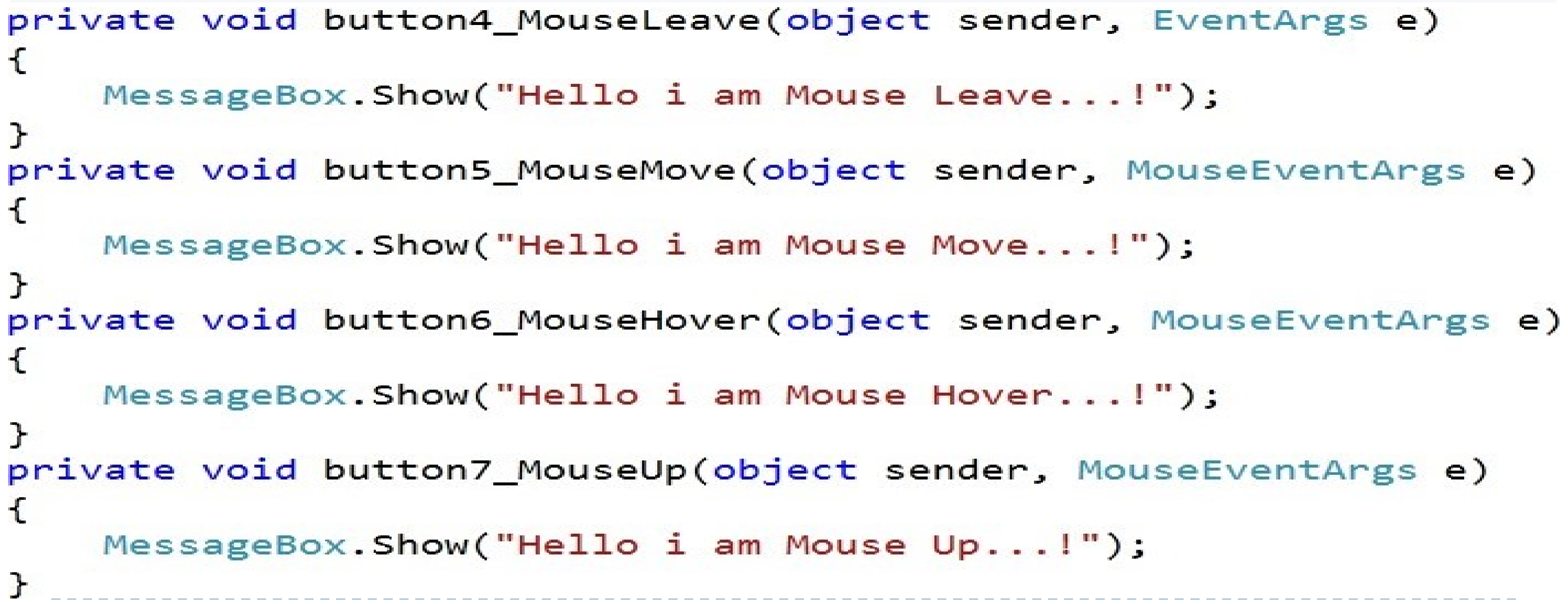
Events	Description
MouseClick	Occurs when Control is clicked by mouse
MouseEnter	Occurs When mouse pointer enters contol
MouseDown	Occurs When Mouse button is pressed over the control.
MouseLeave	Occurs When the mouse pointer leaves the control
MouseMove	Occurs When mouse pointer is moves over the control.
MouseUp	Occurs when mouse button is released over the control
MouseWheel	Occurs when mouse wheel moves while control has focus



```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Click shay Hello");
}

private void button2_MouseDown(object sender, MouseEventArgs e)
{
    MessageBox.Show("Hello i am Mouse Down...!");
}

private void button3_MouseEnter(object sender, EventArgs e)
{
    MessageBox.Show("Hello i am Mouse Enter...!");
}
```



```
private void button4_MouseLeave(object sender, EventArgs e)
{
    MessageBox.Show("Hello i am Mouse Leave...!");
}
private void button5_MouseMove(object sender, MouseEventArgs e)
{
    MessageBox.Show("Hello i am Mouse Move...!");
}
private void button6_MouseHover(object sender, MouseEventArgs e)
{
    MessageBox.Show("Hello i am Mouse Hover...!");
}
private void button7_MouseUp(object sender, MouseEventArgs e)
{
    MessageBox.Show("Hello i am Mouse Up...!");
}
```

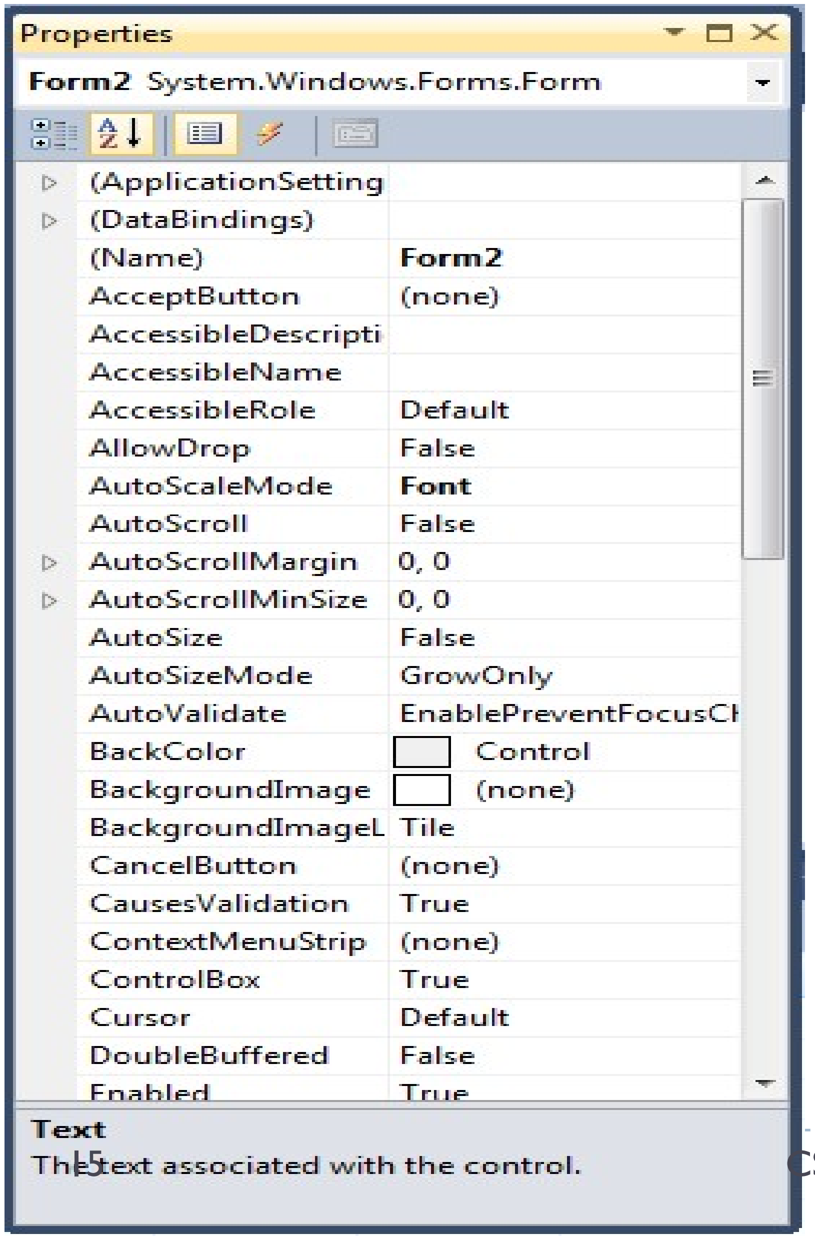
Windows Controls :

- In .NET so many controls are available which is used for create the windows application.
- Using that controls we can create the User Interface.
- All controls have a number of properties that are used to
- manipulate the behavior of the control.
- The base class of most controls, Control, has a number of properties that other controls either inherit directly or override to provide some kind of custom behavior.
- There are two ways to add controls to the form at design time and runtime.
- **Design Time :**
- There are two way in desingn time
- Drag and Drop
- Double click on toolbox control
- **Runtime only one way :**
- Create Object of any Colntrol.

```
TextBox1 t1=new TextBox();
T1.Text="Hello Word…!!!";
T1.BackColor=Color.Red;
This.Controls.Add(t1);
```

Properties:- All controls have a number of properties that are used to manipulate the behavior of the control.

Controls:-The ToolBox contains a selection of all the controls available to you as a .NET developer.

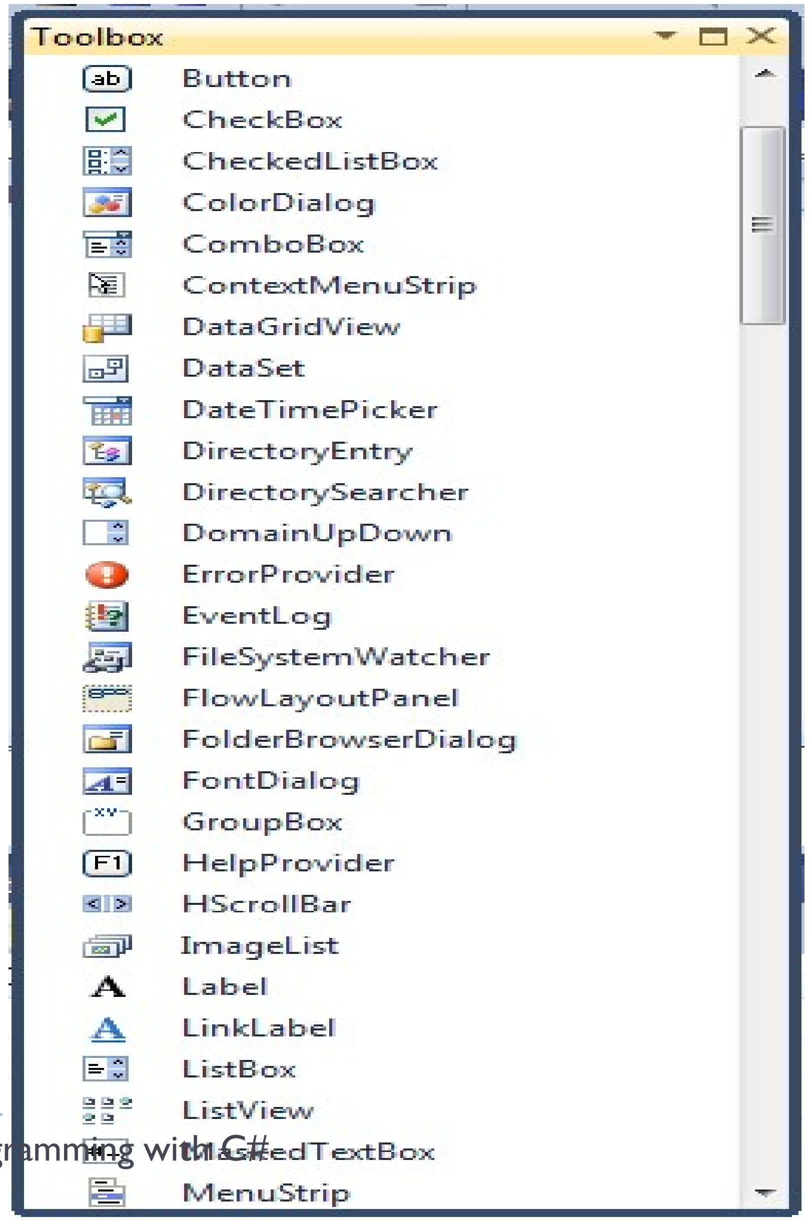


Properties

Form2 System.Windows.Forms.Form

▷ (ApplicationSetting	
▷ (DataBindings)	
(Name)	**Form2**
AcceptButton	(none)
AccessibleDescripti	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
AutoScaleMode	**Font**
AutoScroll	False
▷ AutoScrollMargin	0, 0
▷ AutoScrollMinSize	0, 0
AutoSize	False
AutoSizeMode	GrowOnly
AutoValidate	EnablePreventFocusCh
BackColor	Control
BackgroundImage	(none)
BackgroundImageL	Tile
CancelButton	(none)
CausesValidation	True
ContextMenuStrip	(none)
ControlBox	True
Cursor	Default
DoubleBuffered	False
Enabled	True

Text
The text associated with the control.



Toolbox

- Button
- CheckBox
- CheckedListBox
- ColorDialog
- ComboBox
- ContextMenuStrip
- DataGridView
- DataSet
- DateTimePicker
- DirectoryEntry
- DirectorySearcher
- DomainUpDown
- ErrorProvider
- EventLog
- FileSystemWatcher
- FlowLayoutPanel
- FolderBrowserDialog
- FontDialog
- GroupBox
- HelpProvider
- HScrollBar
- ImageList
- Label
- LinkLabel
- ListBox
- ListView
- MaskedTextBox
- MenuStrip

CommonProperty of all Controls :

Property	Description
BackColor	The background color of a control.
ForeColor	To set or get Foreground Color of Control.
Size	In windows forms, you are allowed to set the size of the button automatically using the **AutoSize Property** of the control.
TabIndex	The number the control has in the tab order of its container.
TabStop	Specifies whether the control can be accessed by the Tab key.
Text	Gets or sets the text associated with this control.
Visible	Specifies whether or not the control is visible at runtime.
Length	The length of the control.
Name	To give Name of Control.
Font	To decide size,style of the Text .
Modifiers	To set Access of any control Like , private,public,protected or internal

Button Control :

- A Button is an essential part of an application, or software, or webpage.
- It allows the user to interact with the application or software.
- For example, if a user wants to exit from the current application so, he/she click the exit button which closes the application.
- It can be used to perform many actions like to submit, upload, download, etc.
- according to the requirement of your program.
- It can be available with different shape, size, color, etc.
- **Examples :**
- **1. To create a Button control, you simply drag and drop a Button control from**
  - **Toolbox to Form in Visual Studio.**

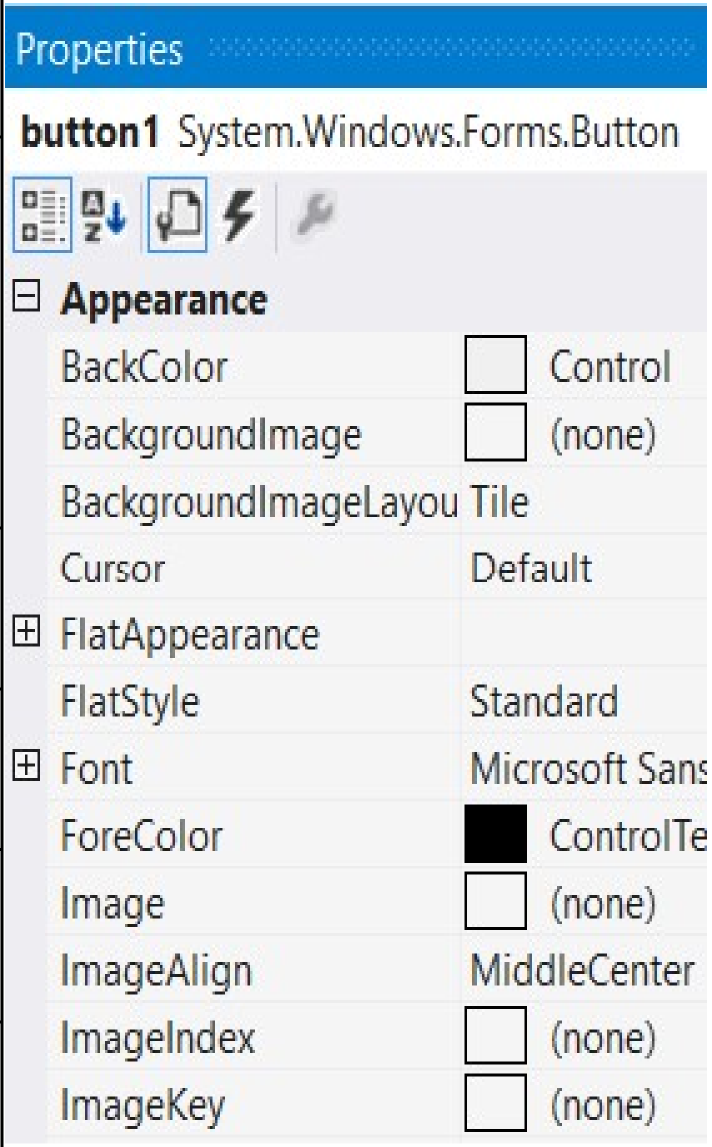
```
private void button1_Click(object sender, EventArgs e)
{
        MessageBox.Show("Hello SyBca");
}
```

- **2. Create a button using the Button() constructor is provided by the Button class.**

```
Button btn = new Button();          // Create Object of Button class
btn.Text = "This is Button";        //Add Some text on Button
this.Controls.Add(btn);             //Add button in Form
```

Properties of Button :

Properties	Description
DialogResult	Obtain or sets the value returned to the parent form when the button is clicked. Often used when you are creating dialogboxes.
FlatStyle	Obtain or sets a flat style appearance
Image	Obtain or sets an image displayed in a button
Image Allign	Obtain or sets the alignment of the image in a button.
TextAlign	Obtain or sets the alignment of the text in the button.


Properties

button1 System.Windows.Forms.Button

⊟ **Appearance**

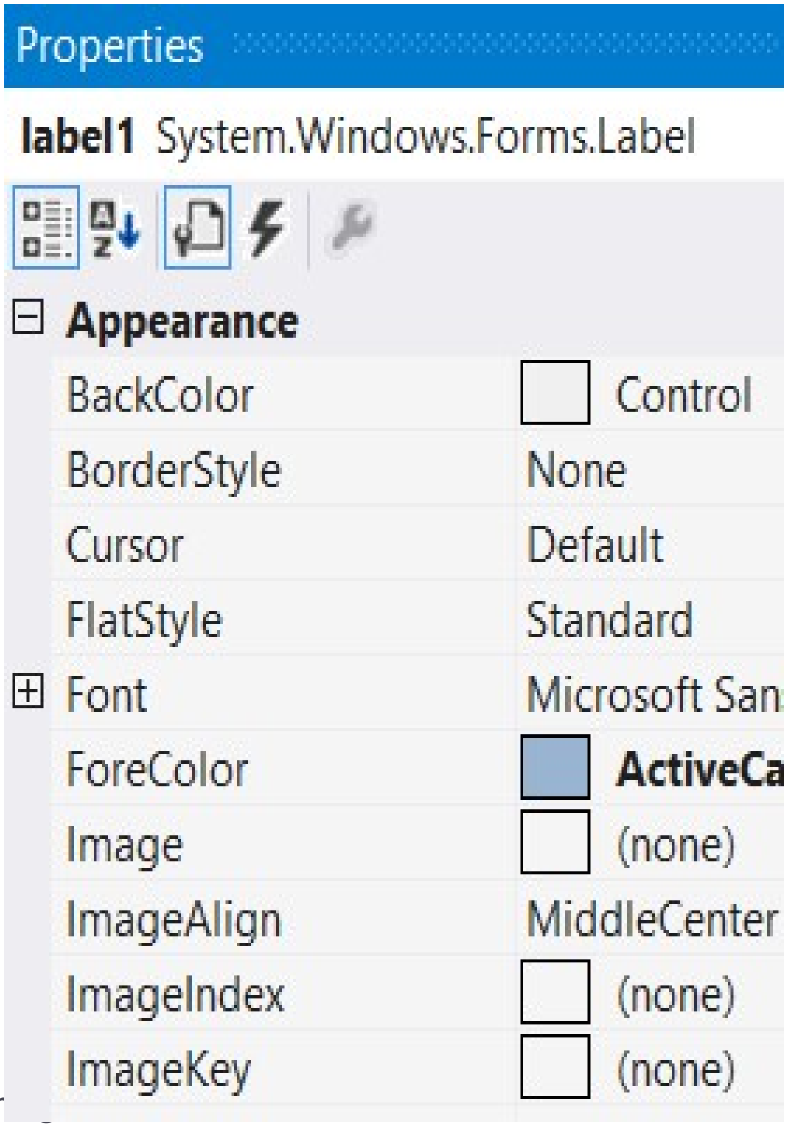
BackColor	⬜	Control
BackgroundImage	⬜	(none)
BackgroundImageLayou	Tile	
Cursor	Default	
⊞ FlatAppearance		
FlatStyle	Standard	
⊞ Font	Microsoft Sans	
ForeColor	⬛	ControlTe
Image	⬜	(none)
ImageAlign	MiddleCenter	
ImageIndex	⬜	(none)
ImageKey	⬜	(none)

Label Control :

- 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.
- The Label is a class and it is defined under System.Windows.Forms namespace.
- **Example :**

  label_nm.Text="This is a Label Control";

  label_nm.ForeColor=Color.Red;


Properties

label1 System.Windows.Forms.Label

Appearance	
BackColor	Control
BorderStyle	None
Cursor	Default
FlatStyle	Standard
Font	Microsoft San
ForeColor	**ActiveCa**
Image	(none)
ImageAlign	MiddleCenter
ImageIndex	(none)
ImageKey	(none)

TextBox Control :

- A TextBox control is used to display, or accept as input, a single line of text.
- This control has additional functionality that is not found in the standard Windows text box control, including multiline editing and password character masking.
- The TextBox control is a windows forms control that lets you enter text on a windows form at runtime.
- TextBox controls are mostly used when the user requires text area where one or few lines of text can be displayed.
- By default, a TextBox control accepts only a single line of text.

Events of TextBox :

Events	**Description**
TextChanged	Fires when the text property value changes
KeyDown	Fires when a key is pressed down while the control has focus
KeyPress	Fires when a key is pressed while the control has focus
KeyUp	Fires when a key is released while the control has focus.

Properties of TextBox Control :

Property	Description
BorderStyle	None, FixedSingle, Fixed3D
PasswordChar,MultiLine	True/False
MaxLength	Default is 32767
CharacterCasing	Upper, Lower
ReadOnly	true/false
ScrollBars	true / false
Text	Text contained in the control
TextAlign	left, right, center
Wordwrap	Wraps the word in the textbox.
Clear	It clear all text from the text box
ClearUndo	Clears information about the most recent operation of textbox.
Copy	It copies the selected text in the text box to the clipboard
Cut	It moved the selected text in the text box to the clipboard
Paste	It replaces the selected textmin the text box with the contents of the clipboard.
Select	It selects text in the text box
SelectAll	It selectes all text in the text box.
Undo	It undoes the last edit operation in the text box.

RadioButton Control :

- A radio button control is also called an option button.
- It is similar to CheckBox control but with two basic differences, first is its round shape and the second is that you can select only one.
- RadioButton control at a time in a group of radio buttons whereas CheckBox controls can be selected more than one at the same time.
- The radio button control's properties and events is same as checkbox control.
- **Example :**

```
private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
        MessageBox.Show("Your Selection is : "+radioButton3.Text);
}
```

CheckBox Control :

- The check box control is a small rectangle with a corresponding text displayed at the right side of it.
- This text works as a label for the check box
- you want the user to have multiple options to choose from the existing list you use the checkbox control.
- **Example :**

```
string ch = "";
 CheckBox[] chh = new CheckBox[4];
 chh[0] = checkBox2;
 chh[1] = checkBox3;
 chh[2] = checkBox4;
 for (int i = 0; i < chh.Length; i++)
 {
     if(chh[i].Checked==true)
     {
            ch += " "+chh[i].Text;
     }
  }
  MessageBox.Show(ch);
```

Properties of RadioButton & CheckBox :

Property	Description
Appearance	Controls the appearance of the check box.
AutoCheck	Causes the check box to automatically change state when click
AutoSize	Specifies whether a control will automatically size itself to fit its content.
CheckAlign	Determine the location of the check box inside the control.
Checked	Indicates whether the component is in the checked state.
CheckState	Indicate the checked of the component.
Text	Give the caption of the control.

Events of RadioButton & CheckBox :

Event	Description
CheckedChanged	It occurs when the Checked property has changed.
AppearanceChanged	It occurs when the Appearance property has changed.

ComboBox Control :

- ComboBox control is a .net control that is widely used for selecting an option from a list.
- The windows forms combobox control is used to display data in a drop-down list. When the user selects an item from the combo box the list contained in the ComboBox automatically collapse.
- A user can choose only a single item from the expandable list of item. You can add or remove an item from this list.
- Each item in a ComboBox control is recognized by its position in the list, which is known as an index.
- **Example :**

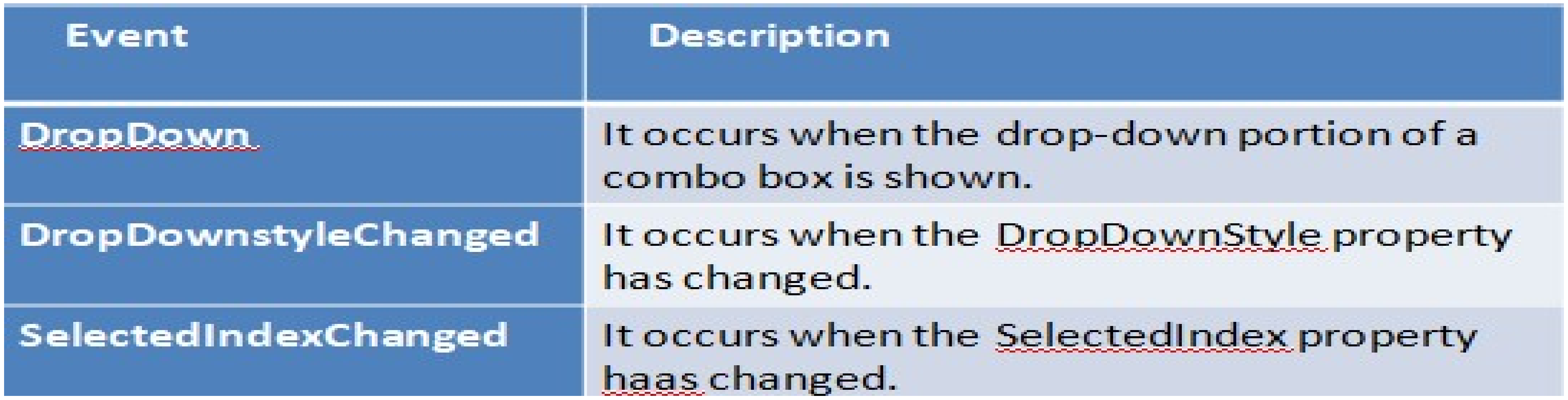
```
comboBox2.Items.Add("Fybca");
comboBox2.Items.Add("Sybca");
comboBox2.Items.Add("Tybca");

private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
{
        MessageBox.Show(comboBox2.Text);
}
```

Properties of ComboBox Control :

Property	Description
DropDownStyle	Obtain the style of the combo box.
Items	Obtains a collection of the items in this combo box.
SelectedItem	Obtain currently selected item in the combo box.
SelectedIndex	Obtain the index of the currently selected item
SelectedText	Obtain the selected text in the text box part of a combo box.
Sorted	Obtain if the items in the combo box are sorted.

Events of ComboBox Control :



Event	Description
DropDown	It occurs when the drop-down portion of a combo box is shown.
DropDownstyleChanged	It occurs when the DropDownStyle property has changed.
SelectedIndexChanged	It occurs when the SelectedIndex property haas changed.

Methods of ComboBox Control :

Method	Description
FindString	It finds the first item in the combo box that begins with the indicated string.
FindStringExact	It finds the item that matches the indicated string exactly.
GetItemText	Obtain an item’s text.
Select	It selects a range of text.
SelectAll	It selects all the text in the text box of the combo box.

Example of SelectedIndex :

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
        if (comboBox1.SelectedIndex == 0)
           this.BackColor = Color.Black;
        else if (comboBox1.SelectedIndex == 1)
           this.BackColor = Color.Blue;
        else
           this.BackColor = Color.White;
}
```

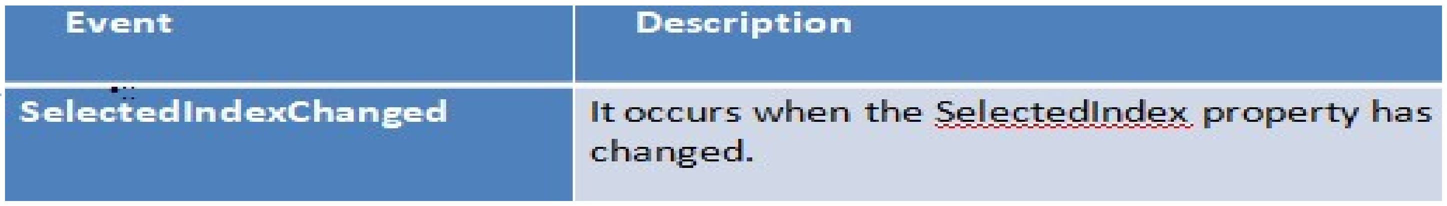
ListBox Control :

- List box control is a standard windows control that is used to display the text as a list. The text can be displayed as a sorted or an unsorted list.
- The list box control recognizes these texts as a collection of items. You can add the text as an item into this collection for displaying it on a ListBox control.
- Similarly you can remove an item for not displaying it on the listBox control. Each item in a listbox control is recognized by its position in the list, which is known as its index.
- **Example :**

```
listBox2.Items.Add("Fybca");
listBox2.Items.Add("Sybca");
listBox2.Items.Add("Tybca");
```

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    MessageBox.Show(listBox2.Text);
}
```

Events of ComboBox Control :



Event	Description
SelectedIndexChanged	It occurs when the SelectedIndex property has changed.

Properties of ComboBox Control :

Property	Description
MultiColumn	True/false
ScrollAlwaysVisible	True/False(vertical scroll bar is appeared or not)
Sorted	True/false
SelectedItem	It is read-only property. it returns selected first value.
selectedItems	It is read-only property. Returns more than one selected items
SelectedIndex	It is read-only property, it returns selected index.
SelectionMode	MultiSimple, none, one etc.

Events of ComboBox Control :

Method	Description
ClearSelected ()	Clears selected all the items in the ListBox
FindString()	It returns Boolean answer.
IndexOf()	it returns index of the occurrence of particular String. if not found return -1
GetSelected()	It checks whether given index of the item is selected or not. It returns Boolean result.
SetSelected()	It selects or deselects the item of the given index
Add()	Add the value.

CheckedListBox Control :

- Allows the user to select multiple items from a list of items.
- This control inherits from the ListBox control and therefore has the same properties, methods and events.
- You should change the CheckOnClick property to True to let the user check and uncheck elements with a single click.
- **Example :**

```
label1.Text="";
foreach (string item in checkedListBox1.CheckedItems)
{
            listBox1.Items.Add(item);
           label1.Text += item;
}
```

PictureBox Control :

- The picture box control is used to display an image.
- The image can be a BMP, JPEG, GIF, and PNG, metafile or icon.

Properties of PictureBox Control :

Property	Description
Border Style	It indicates that what type of border control box has. This property have 3 options none, FixedSingle and Fixed3D.
Image	The image displayed in the picture box.
InitialImage	Image to display while another image is loading.
Lock	The lock property determines if we can move or resize the control
SizeMode	It indicates how the picture box will handle image placement and control sizing. This property have 5 options Normal, StretchImage, AutoSize, CenterImage, Zoom.

- **Example :**

```
pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;

pictureBox1.Image =
   Image.FromFile("E:\\image_path\\image.png");
```

Or

```
pictureBox1.Image =
   Image.FromFile(@"E:\image_path\image.png");
```

ScrollBar Control :

- Scrollbar are a common element of the windows interface, so the scrollbar control is often used with controls that do not derived from the Scrollable class.
- The HScrollBar and VScrollBar controls are contains the same properties, event, and methods.
- ScrollBar controls are not the same as the built-in scrollbars that are attached to text boxes, list boxes, combo boxes or MDI forms.
- There are Two types of ScrollBar :

- Hscrollbar
- Vscrollbar
- **Example :**

```
private void hScrollBar1_Scroll(object sender, ScrollEventArgs e)
{
        this.BackColor = Color.FromArgb(hScrollBar1.Value,0,0);
}
```

Properties of ScrollBar Control :

Property	Description
Value	This property is an integer value corresponding to the position of the scroll box in the scroll bar. By default value is 0.
Maximum	This property is occurs uppermost range of the control in hscrollbar it indicate right-most and in vscroll bar is indicate bottom position.
Minimum	This property is occurs lowermost range of the control in hscrollbar it indicate left-most and in vscroll bar is indicate Top position.
LargeChange	It occurs when PAGE UP or PAGEDOWN or click in the scroll bar at time value property is change.
SmallChange	It occurs when user can presses one of the arrow keys or clicks one of the scrollbar buttons.

Events of ScrollBar Control :

Event	Description
Scroll	Occurs when the user moves the scroll box.

TreeView Control :

- It display a hierarchy of nodes both parent and child.
- Windows explorer and solution explorer contain TreeView control.
- TreeView control is collection of nodes.(like ListBox contains items
- collection.)
- The main starting node is called root node.
- Under the root, a real tree is made of branches and leaves.
- A node can have a node as child.
- We can display TreeView control with check boxes next to the nodes, if the TreeView's CheckBoxes property is set to true.
- Nodes Collection:
  - The items in the TreeView are stored in the nodes collection.
  - We can add items using two ways, design time and run time.

- **Example :**

```
treeView1.Nodes.Add(“fybca");   //abc has 0 index
treeView1.Nodes.Add(“sybca"); //abc has 1 index
treeView1.Nodes.Add(“tybca"); //abc has 2 index

treeView1.Nodes[0].Nodes.Add("a");
//a is child node of fybca

treeView1.Nodes[0].Nodes[0].Nodes.Add("xyz");
//xyz is a child node of a and sub child node of fybca
```

Method of TreeView Control :

Method	Description
Clear ()	It clears all the nodes from the TreeView
Contains()	It returns Boolean answer. It checks whether the given object is in the TreeView or not.
Remove()	It removes the item from the TreeView
RemoveAt()	It removes item from the TreeView by index.

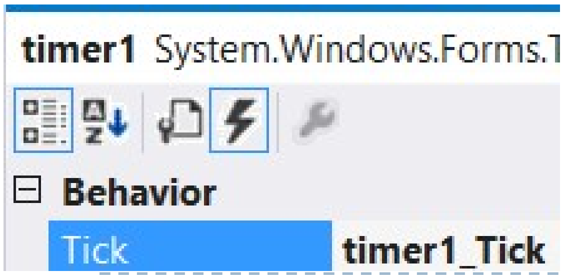
Properties of TreeView Control :

Property	Description
SelectedNode	Return selected node.
CheckBoxes	True/False (default value is false)
HotTracking	True/false (default false) Nodes get hyperlink effect when the mouse is moved over them.
ShowLines	True/false (default true). Lines are displayed between nodes or not
ShowPlusMinus	True/false (default true). Plus/minus are displayed between nodes or not
ShowRootLine	True/false (default true). RootLine are displayed between parent and child node or not
FullPath	Returns Full Path of the node

Timer Control :

- 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.
- 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 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.

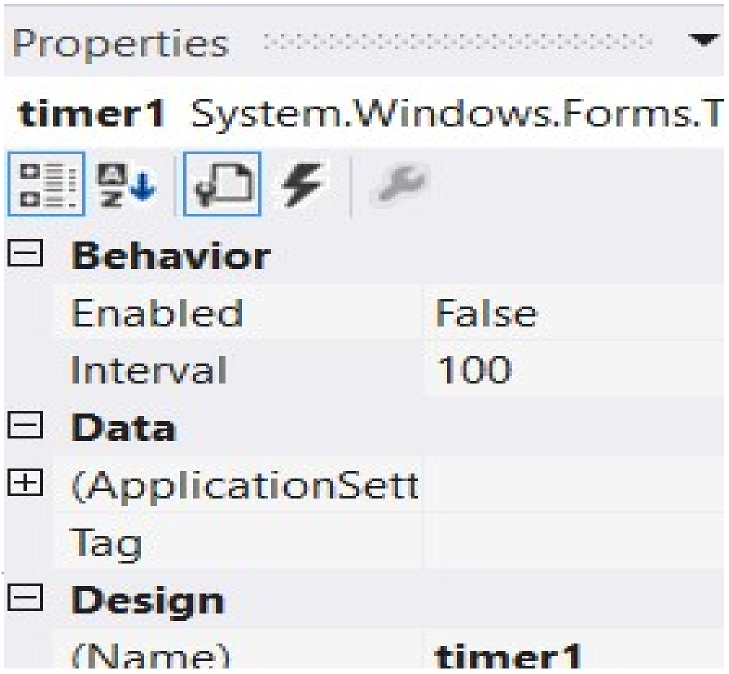
```
int v=0;
private void timer1_Tick(object sender, EventArgs e)
{
        txt_log.Text = (v++).ToString();
}
private void btn_start_Click(object sender, EventArgs e)
{
        timer1.Start();
}
private void  btn_stop_Click(object sender, EventArgs e)
{
        timer1.Stop();
}
```



```
timer1 System.Windows.Forms.T
Behavior
  Tick        timer1_Tick
```

← : Event of Timer

Properties of Timer :→



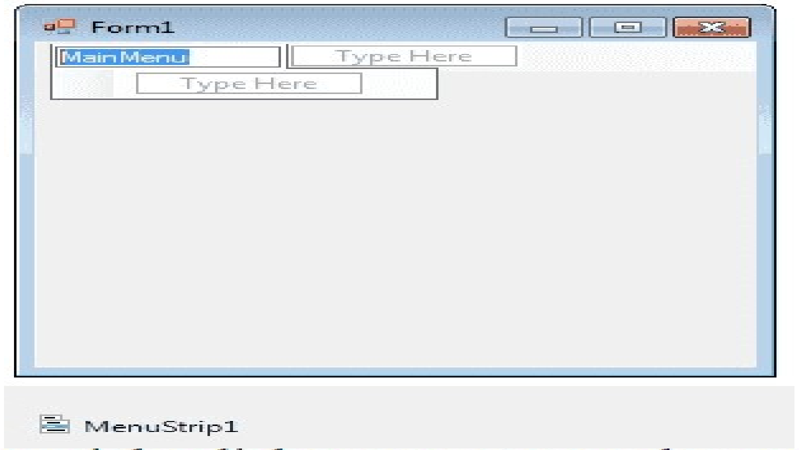
```
Properties
timer1 System.Windows.Forms.T
Behavior
  Enabled               False
  Interval              100
Data
  (ApplicationSett
  Tag
Design
  (Name)                timer1
```

Menu Control :

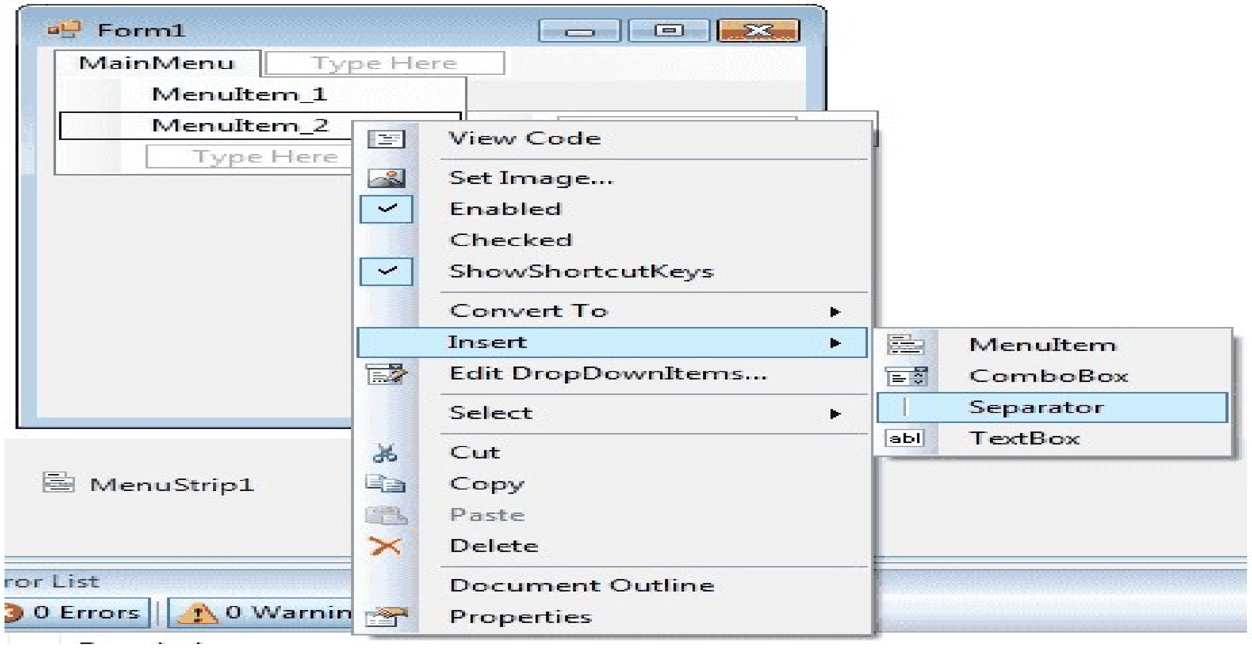
- A Menu on a Windows Form is created with a MainMenu object, which is a collection of MenuItem objects.
- MainMenu is the container for the Menu structure of the form and menus are made of MenuItem objects that represent individual parts of a menu.
- You can add menus to Windows Forms at design time by adding the MainMenu component and then appending menu items to it using the Menu Designer.
- Two types of Menu:
- 1. MenuStrip
- – The MenuStrip control represents the container for the menu structure. The MenuStrip control works as the top-level container for the menu structure.
- 2. ContextMenuStrip
- – The ContextMenuStrip control provides functionality of context menus, A context menu is also known as a popup menu. A context menu appears when you right click on a Form or on a control.

1. MenuStrip Control :

- After drag the Menustrip on your form you can directly create the menu items by type a value into the "Type Here" box on the menubar part of your form.
- From the following picture you can understand how to create each menu items on mainmenu Object.



If you need a seperator bar , right click on your menu then go to insert->Seperator.

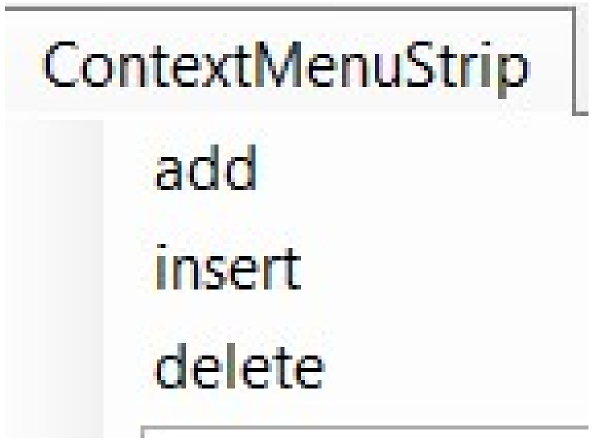


- **Example :**

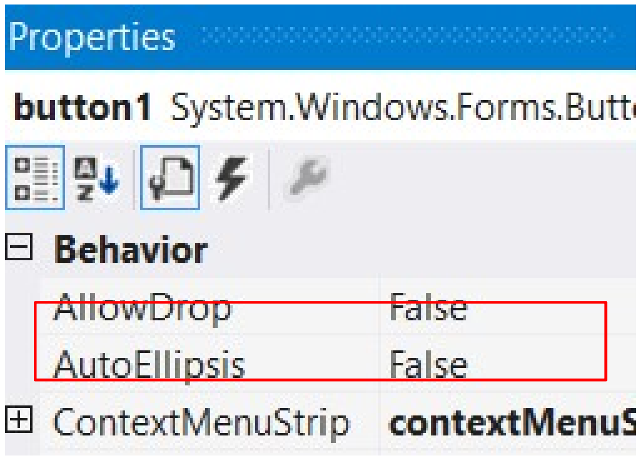
```
private void  Button1_Click(object sender, EventArgs e)
{
        menuStrip1.Items.Add(“new");
        menuStrip1.Items.Add(“open");
}
private void newToolStripMenuItem_Click_1(object sender,
  EventArgs e)
{
          this.Hide();
          other o = new other(this);
          o.Show();
}
```

2. ContextMenuStrip :

- Creating a Context Menu
- To create a ContextMenuStrip control at design- time, you simply drag and drop a ContextMenuStrip control from Toolbox onto a Form in Visual Studio.
- After you drag and drop a ContextMenuStrip on a Form, the ContextMenuStrip1 is added to the Form , Once a ContextMenuStrip is on the Form, you can add menu items and set its properties and events.



Add Menu



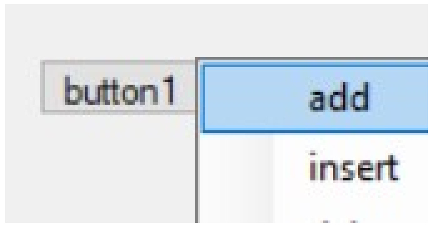
Properties

button1 System.Windows.Forms.Butt

Behavior

AllowDrop	False
AutoEllipsis	False
ContextMenuStrip	contextMenuS

Add Context menu on Button



After Right Click on Button

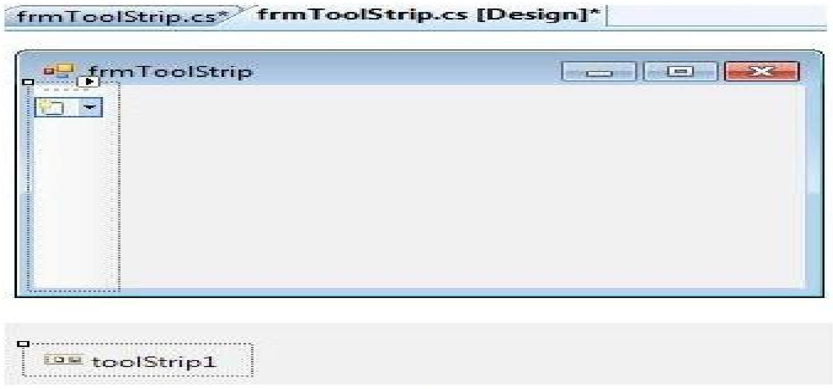
ToolStrip Control :

- ToolStrip is a container for ToolStripItem elements.
- Each individual element on the ToolStrip is a ToolStripItem that manages the layout and event model for the type it contains.
- The ToolStrip controls provide a common interface for Menus and Strips in Windows Forms.
- **ToolStrip Properties :**

Property	Description
Size	Set the height and width of the control.
Text	Set the text associated with this control.
RenderMode	Set a value that indicates which visual styles will be applied to the ToolStrip.
LayoutStyle	Set a value indicating how the ToolStrip lays out the items collection.

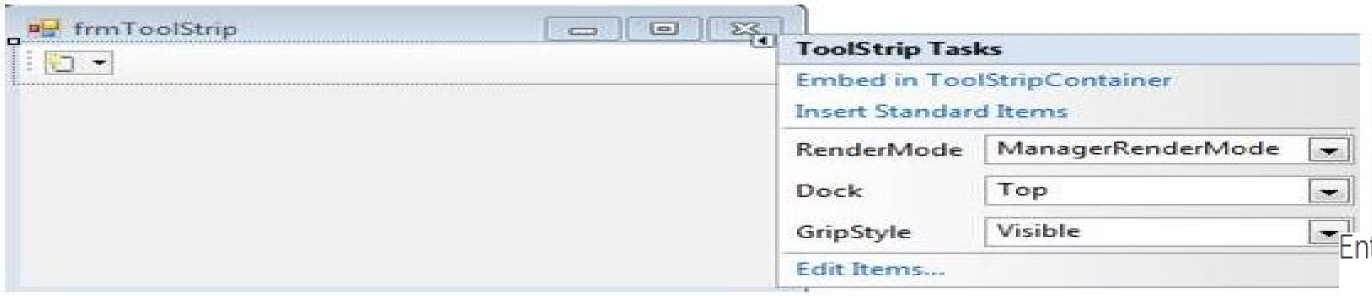
How to use ToolStrip Control

Drag and drop ToolStrip Control from toolbox on the window Form.



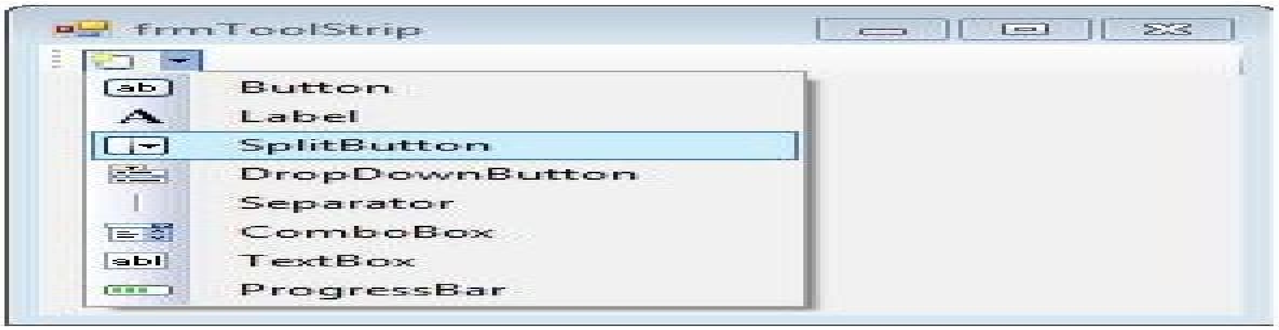
Add ToolStrip Item which you want to show. Add one of the items in your ToolStrip that derives from ToolStrip Item.

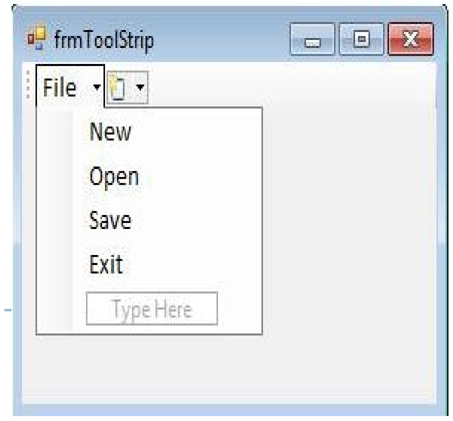
You can also add standard item through smart tag.



RenderMode: ManagerRenderMode
Dock: Top
GripStyle: Visible

Enter text which you want to show as Menu.





Here SplitButton is added

Panel Control :

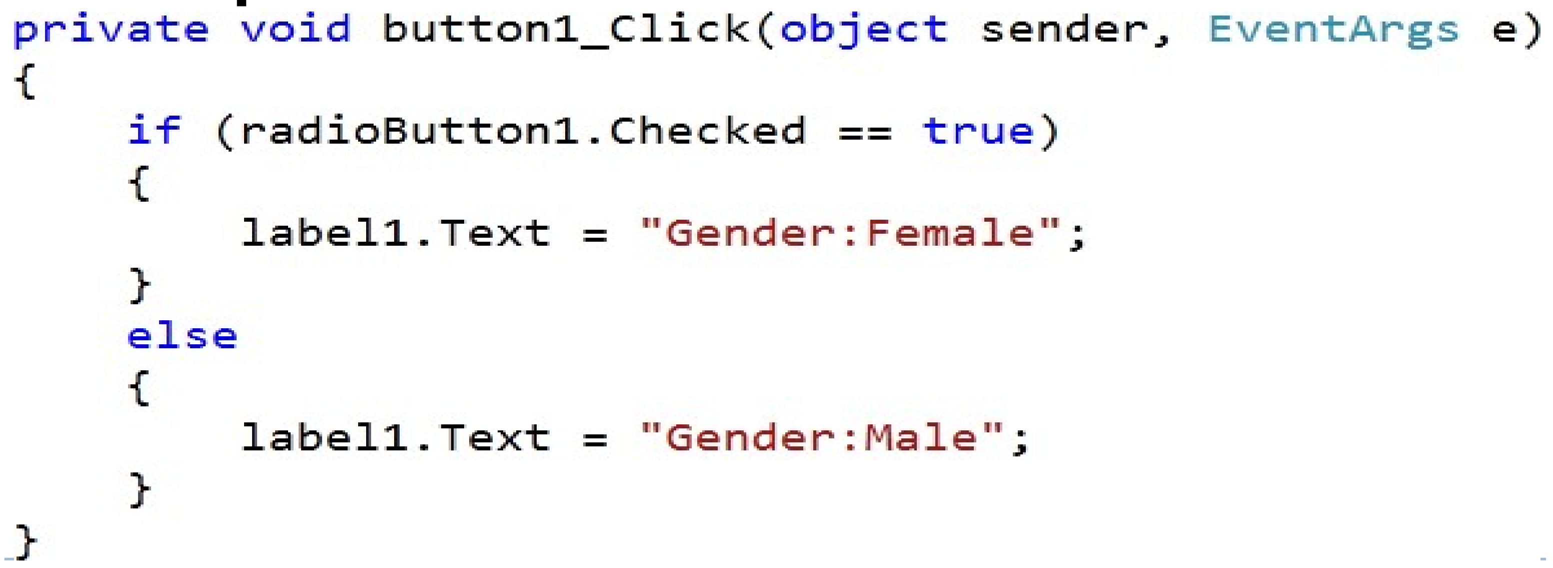
- A panel is simply a control that contains other controls. By grouping controls together and placing them in a panel, it is a little easier to manage the control.
- The panel control is similar to the groupbox control. The difference between the two is that only the panel control can have scroll bars and only the group box control displays a caption.
- At design time when you move a panel control, all its contained controls also move with it. The panel control works like a container for the controls which are contained inside it and is not displayed at runtime.
- Panel does not show a border by default, but by setting the
- BorderStyle property to panel the border will show you.

- Properties of Panel :

Property	Description
BorderStyle	Indicates the border style for the control.
DisplayRectangle	Gets the rectangle that represents the virtual display area of the control.
Enabled	Gets or sets a value indicating whether the control can respond to user interaction.
Visible	Gets or sets a value indicating whether the control and all its child controls are displayed.

GroupBox Control :

- In Windows form, GroupBox is a container which contains multiple controls on it and the controls are related to each other.
- Or in other words, GroupBox is a frame display around a group of controls with a suitable optional title.
- The main use of a group box is to hold a logical group of RadioButton controls.
- **Example :**



```
private void button1_Click(object sender, EventArgs e)
{
    if (radioButton1.Checked == true)
    {
        label1.Text = "Gender:Female";
    }
    else
    {
        label1.Text = "Gender:Male";
    }
}
```

GroupBox	Panel
It has the Text property.	It has not Text property.
We cannot display scrollbars on GroupBox.	We can display scrollbars on panel if the height/width of the controls exceeds that of the panel. For that set AutoScroll property to true.
It dose not have the click event.	It has the click event.
BorderStyle property is not there as Border or Frame is there by default .	To display Border, we have to use the BorderStyle property.
GroupBox don't.	Panel allows drop.

DiloagBoxes Controls :

- .NET contains built-in dialog boxes which allow us to create our own file open, file save and color dialog control like we see in all windows application.
- The common dialog controls are given below.

1. ColorDialog
2. FontDialog
3. OpenFileDialog
4. SaveFileDialog

- To use any common control just add that control to the form .
- When a common dialog control is added to a project, a new icon appears in the components
- tray of the form.
- They do not take place any specific position on the form like other controls.
- To display common dialog control at runtime we need to call.
- ShowDialog method of the dialog control.

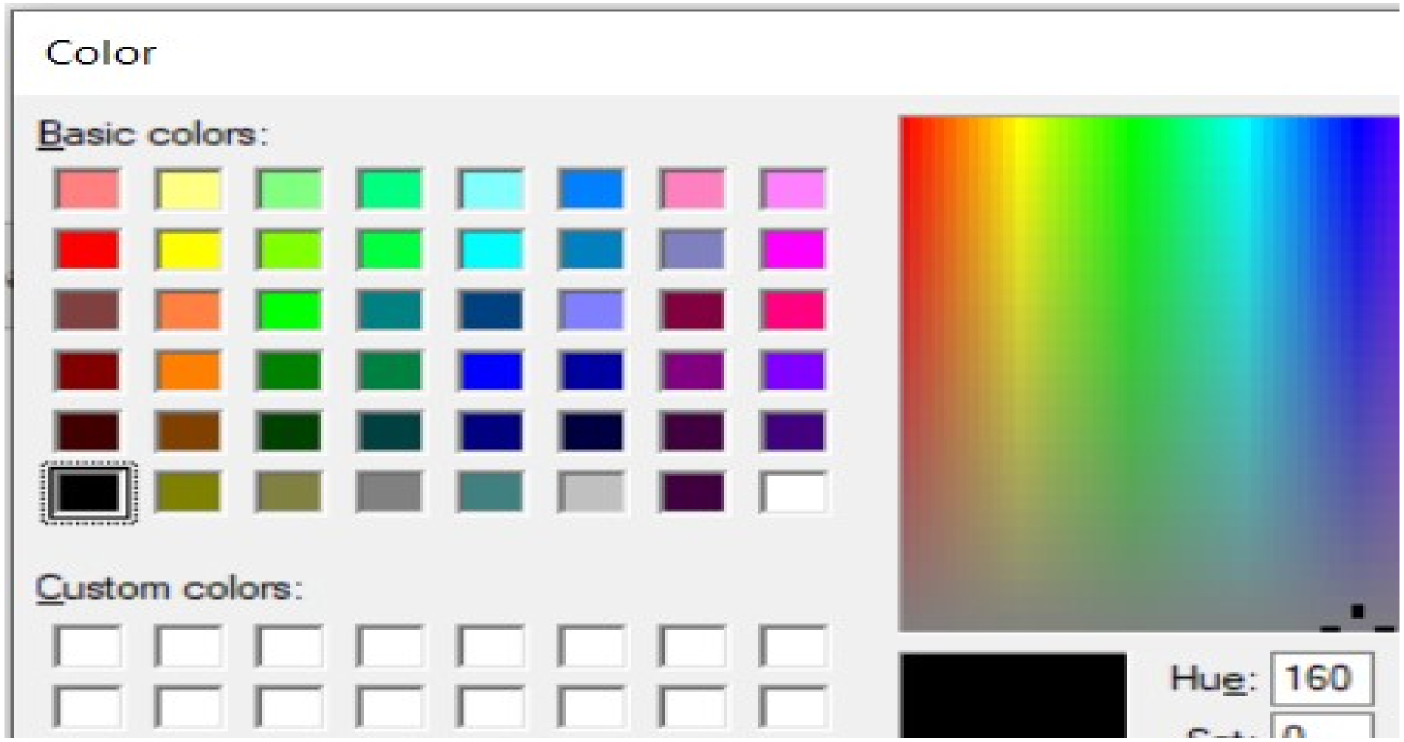
1) ColorDialog :

- ColorDialog allow us to select color.
- User can select easily color form it.
- We can get/set selected color value in Color property.
- As we shown in any other windows application the colorDialog allow as same manipulation with different colors.

Property	Description
AllowFullOpen	True/false. Default value is true.
Color	Get/Set the selected color value. Default color is black if we set this property before opening the dialogbox, the selected color will display on the Color dialogbox.
FullOpen	True/False. Default value is False. If we want to give color using custom color setting and the value is true then button is enable true otherwise enable false.

- **Example :**

```
private void button1_Click(object sender, EventArgs e)
{
             ColorDialog colorDialog1 = new ColorDialog();
             colorDialog1.FullOpen = true;
            if(colorDialog1.ShowDialog()==DialogResult.OK)
            {
               //  button1.BackColor = colorDialog1.Color;
                                              //or
               this.BackColor = colorDialog1.Color;
            }
 }
```

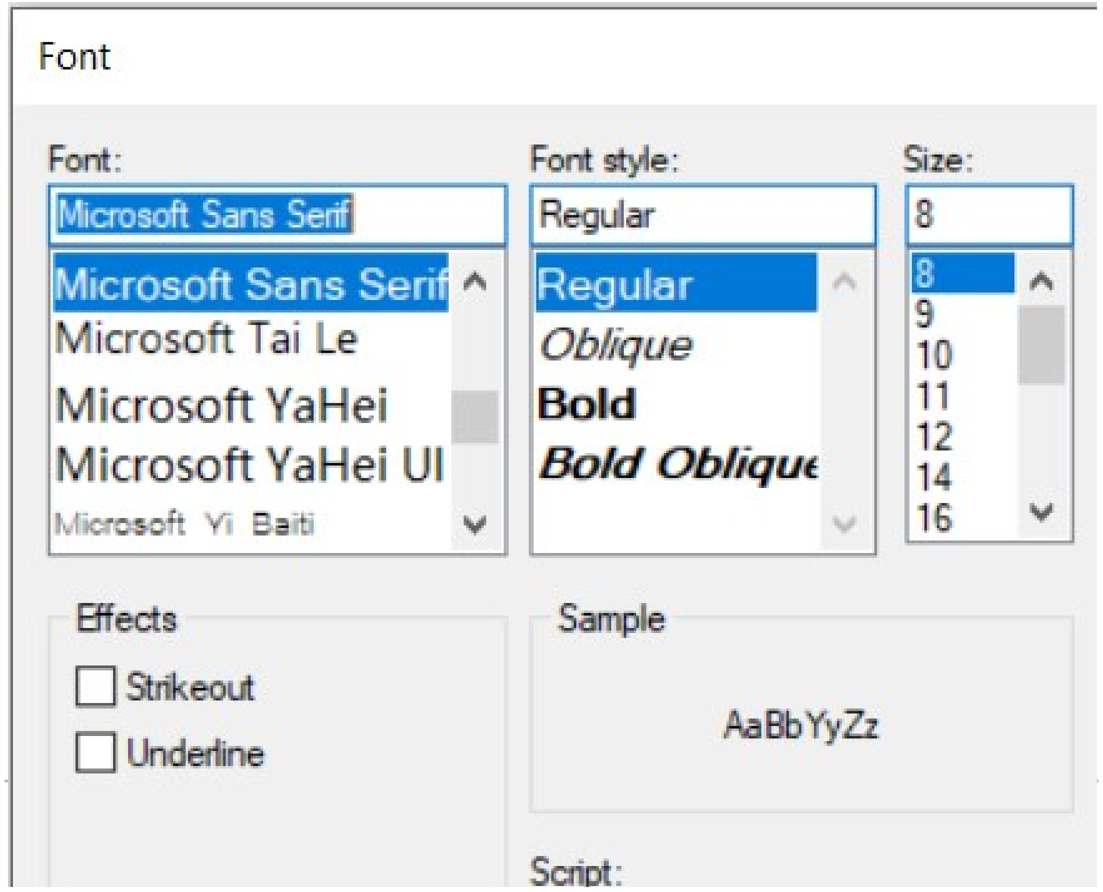


2) FontDialog :

- FontDialog allow us to select font, fontsize and fontstyle and color.
- We can get/set selected font value in font property.

Property	**Description**
AllowFullOpen	True/false. Default value is true.
AllowScriptChange	True/False. Default value is true. The user can change the character set specified in the script.
Color	True/false. Default value is true It returns color selected by user. Default color is black.
Font	Gets/sets font
MaxSize, MinSize	Starting and ending font size. Default value is 0 and no font size limits.
ShowApplay	True/False. Default value is false. If we want to display ShowApplay button then make it true.
ShowColor	True/false. Default value is false. Related with color dialog ComboBox on FontDialog control.
ShowEffects	Whether the dialog box contains controls that allow the user to specify strikethrough, underline and text color options.

```
private void button2_Click(object sender, EventArgs e)
    {
        FontDialog   fontDialog1 = new FontDialog();
        if(fontDialog1.ShowDialog()==DialogResult.OK)
        {
            //richTextBox1.Text = fontDialog1.Font;
            this.Font = fontDialog1.Font;
        }
    }
```


Font

Font: Microsoft Sans Serif

Font style: Regular

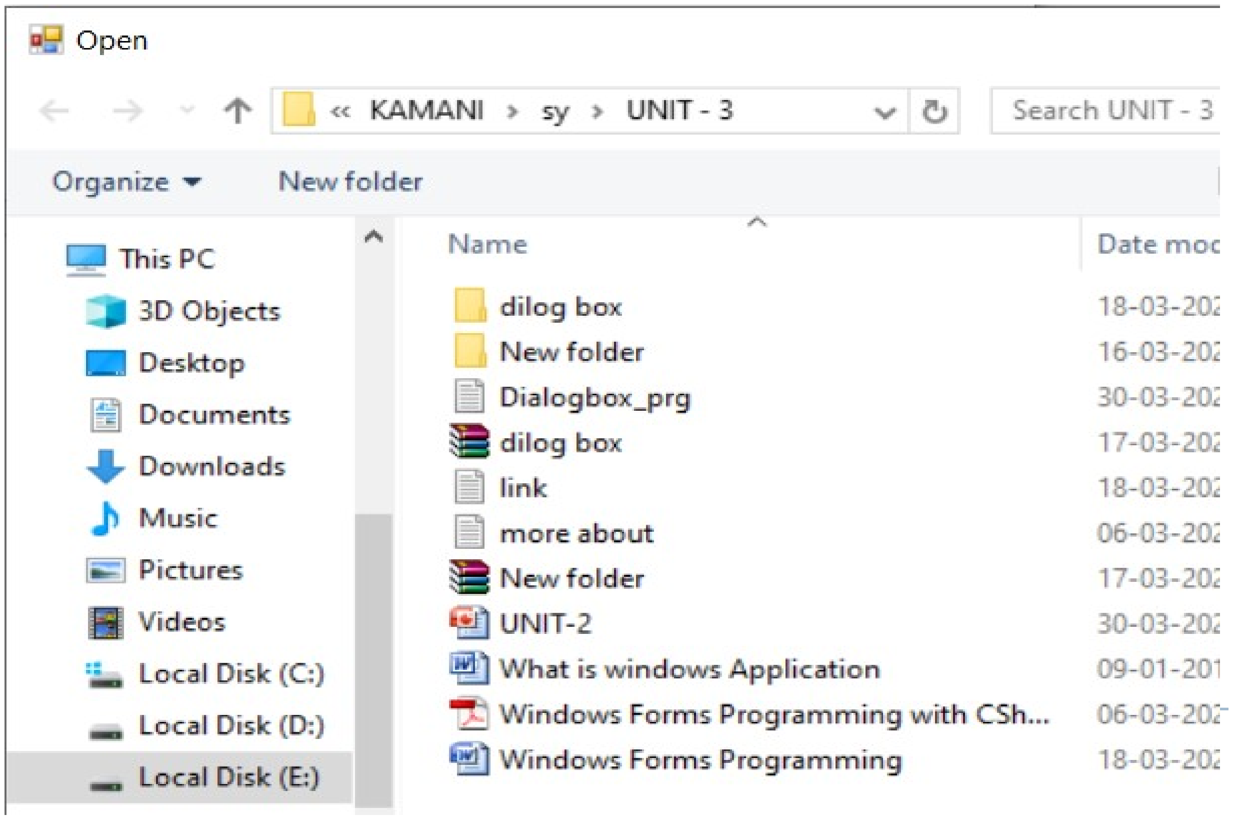
Size: 8

3) OpenFileDialog :

- It allow us to selected file to be open.
- Like Microsoft Word open any document file when we click on open option from file menu same way open file dialog allows the operation to be performed.
- Different properties of OpenFileDialog control are as follows.
- If you want to open any file then you must use StreamReader Class of **using System.IO;** Namespace

Property	**Description**
CheckFileExists	Boolean value. The default value is true. Provide warning message when we trying to open the name of the file which does not exists.
FileName	Returns a string that contains the complete path and filename of the selected file.
Filter	Related with "Files of type" drop down listnox. Ex. set "Text File|*.txt|Word File|*.doc"
Font	Gets/sets font
InitialDirectory	Gets/sets initial directory
Title	Gets/sets title of the OpenFileDialog Box

```
private void button4_Click(object sender, EventArgs e)
{
        OpenFileDialog openFileDialog1 = new OpenFileDialog();
         if(openFileDialog1.ShowDialog()==DialogResult.OK)
         {
                var fileStream = openFileDialog1.OpenFile();
                 using (StreamReader reader = new StreamReader(fileStream))
                 {
                            richTextBox1.Text = reader.ReadToEnd();
                  }
          }
}
```



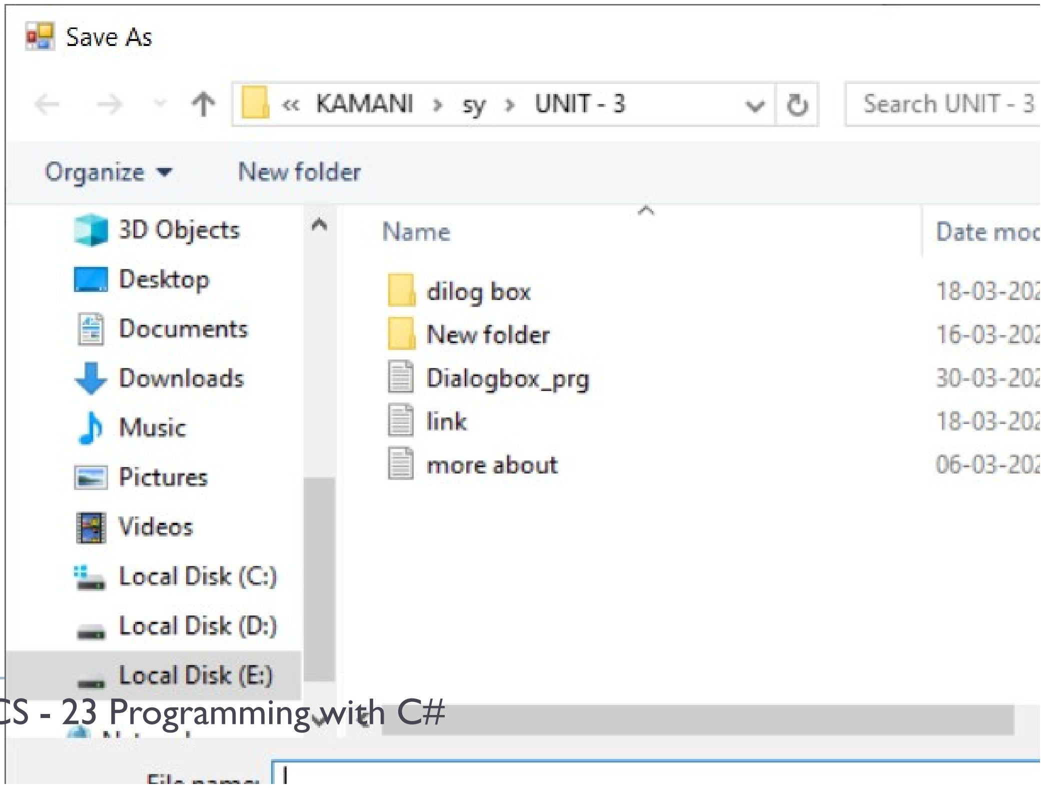
Name	Date moc
dilog box	18-03-202
New folder	16-03-202
Dialogbox_prg	30-03-202
dilog box	17-03-202
link	18-03-202
more about	06-03-202
New folder	17-03-202
UNIT-2	30-03-202
What is windows Application	09-01-201
Windows Forms Programming with CSh...	06-03-202
Windows Forms Programming	18-03-202

4) SaveFileDialog :

- Allow us to save the file in a specified location.
- Basically it is SaveAs file dialog.
- The default title of SaveFile dialog is Save As.

Property	**Description**
CheckFileExists	Boolean value. The default value is true. Provide warning message when we trying to open the name of the file which does not exists.
FileName	Returns a string that contains the complete path and filename of the selected file.
Filter	Related with "Files of type" drop down listnox. Ex. set "Text File|*.txt|Word File|*.doc"
DefaultExt	You can set Default Extention of any file
Font	Gets/sets font
InitialDirectory	Gets/sets initial directory
Title	Gets/sets title of the OpenFileDialog Box

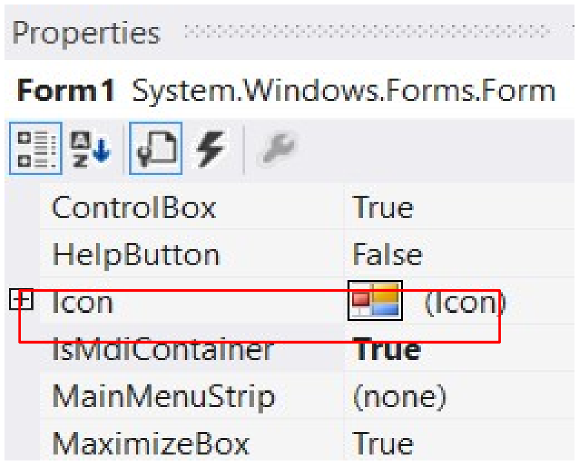
```
private void button3_Click(object sender, EventArgs e)
{
        saveFileDialog1.DefaultExt = "txt";
        saveFileDialog1.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
            using (Stream s = File.Open(saveFileDialog1.FileName,
   FileMode.CreateNew))
using (StreamWriter sw = new StreamWriter(s))
            {
                sw.Write(richTextBox1.Text);
            }
        }
}
```



MDI Form :

- The Multiple-Document Interface (MDI) is a specification that defines a user interface for applications that enable the user to work with more than one document at the same time under one parent form (window).
- Visualize the working style of an application in which you are allowed to open multiple forms in one parent container window, and all the open forms will get listed under the Windows menu.
- Whereas having an individual window for each instance of the same application is termed as single document interface (SDI);
- applications such as Notepad, Microsoft Paint, Calculator, and so on, are SDI applications. SDI applications get opened only in their own windows and can become difficult to manage, unlike when you have multiple documents or forms open inside one MDI interface.

```
private void Form1_Load(object sender, EventArgs e)
{
        IsMdiContainer = true;
}
```

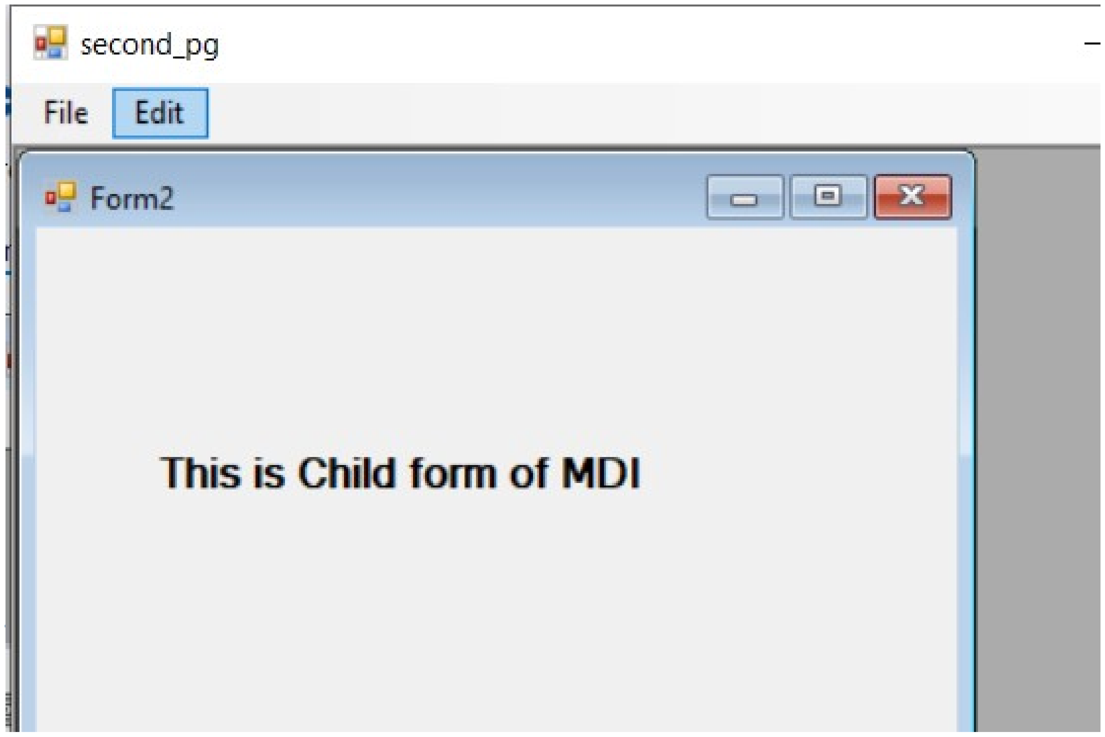


Properties

Form1 System.Windows.Forms.Form

ControlBox	True
HelpButton	False
Icon	(Icon)
IsMdiContainer	**True**
MainMenuStrip	(none)
MaximizeBox	True

```
private void newToolStripMenuItem_Click(object sender, EventArgs e)
{
        second_pg obj_sc = new second_pg();
        obj_sc.MdiParent = this;
        obj_sc.Show();
}
```

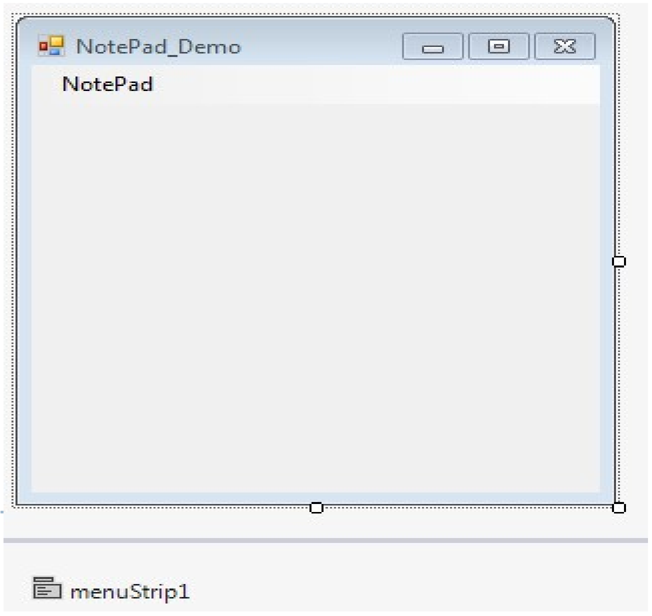


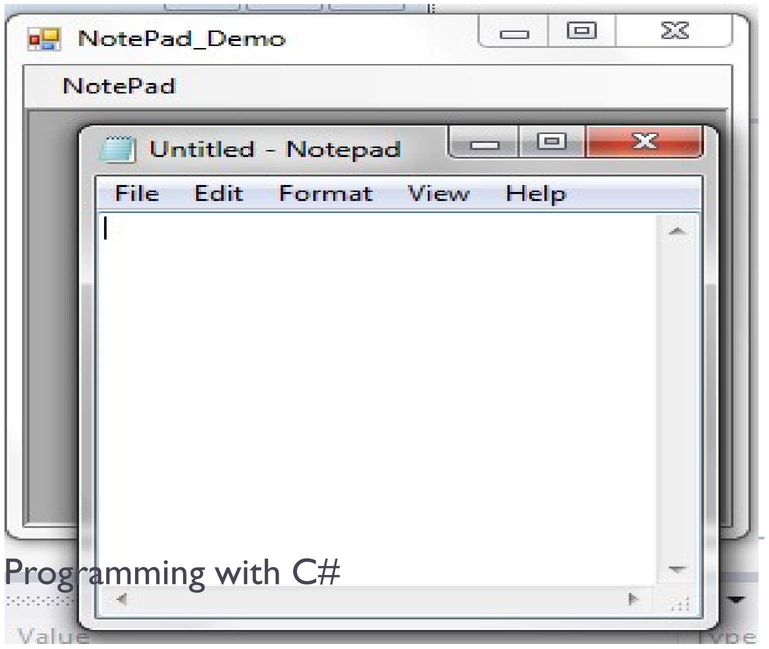
This is Child form of MDI

MDI With Notepad :

```
 using System.Diagnostics;
private void Form1_Load(object sender, EventArgs e)
{
    IsMdiContainer = true;
}

private void notepadToolStripMenuItem_Click(object sender, EventArgs e)
{
    System.Diagnostics.Process.Start("notepad.exe");
}
```





- More About MDI Form :
- https://www.c-sharpcorner.com/UploadFile/84c85b/building-mdi-winforms-application-using-C-Sharp/

Assignment Questions :

1. What is MessageBox class with all types of Show().
2. Explain TextBox V/S RitchTextBox
3. What is Event ? Explain any tree event.
4. Explain TreeView, Timer, ProgressBar, PictureBox Controls.
5. Difference Between MenuStrip V/S ContextMenuStrip.
6. What is DialogBox? Explain SaveFileDialog, ColorDialog with Example.
7. What is MDI? Explain in detail.