.Net

Ch. 03

Windows Programming

By : GOHIL MANOHARSINH A.

Syllabus : Ch.03

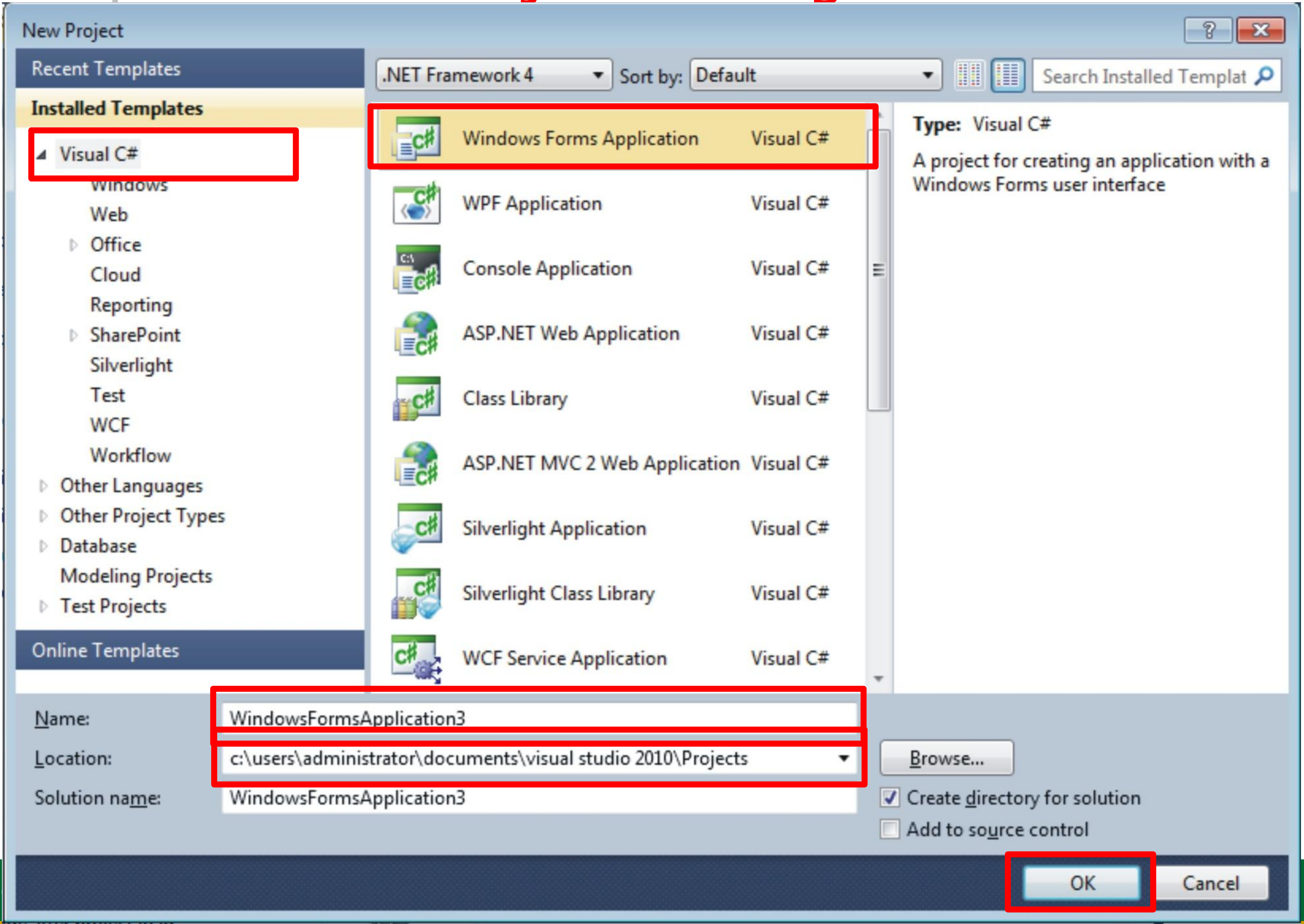
- Creating windows application
- MessageBox class with all types of Show() method
- Basic Introduction to Form and properties
- Concept of adding various Events with event parameters
- Different Windows Controls
  - Button, Label, TextBox, RadioButton, CheckBox, ComboBox, ListBox, PictureBox, ScrollBar, TreeView, Menu (MenuStrip, ContextMenuStrip), ToolStrip, Timer, Panel and GroupBox, Dialog Boxes (ColorDialog, FontDialog, SaveFileDialog and OpenFileDialog) MDI concept with MDI Notepad, Concept of Inheriting Form.

Creating a Windows FORM Application :

- Step : 1
  - Install Microsoft Visual Studio 2010 **IDE.**
- Step : 2
  - Goto File Menu
    - File → New → Project
      - It will display a Screen like,

Creating a Windows FORM Application :

- Step : 2 New Project Dialog BOX


New Project

Recent Templates

Installed Templates

- Visual C#
  - Windows
  - Web
  - Office
  - Cloud
  - Reporting
  - SharePoint
  - Silverlight
  - Test
  - WCF
  - Workflow
- Other Languages
- Other Project Types
- Database
- Modeling Projects
- Test Projects

Online Templates

.NET Framework 4 · Sort by: Default · Search Installed Templat

Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
ASP.NET Web Application	Visual C#
Class Library	Visual C#
ASP.NET MVC 2 Web Application	Visual C#
Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
WCF Service Application	Visual C#

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: WindowsFormsApplication3

Location: c:\users\administrator\documents\visual studio 2010\Projects

Solution name: WindowsFormsApplication3

Browse...

✓ Create directory for solution

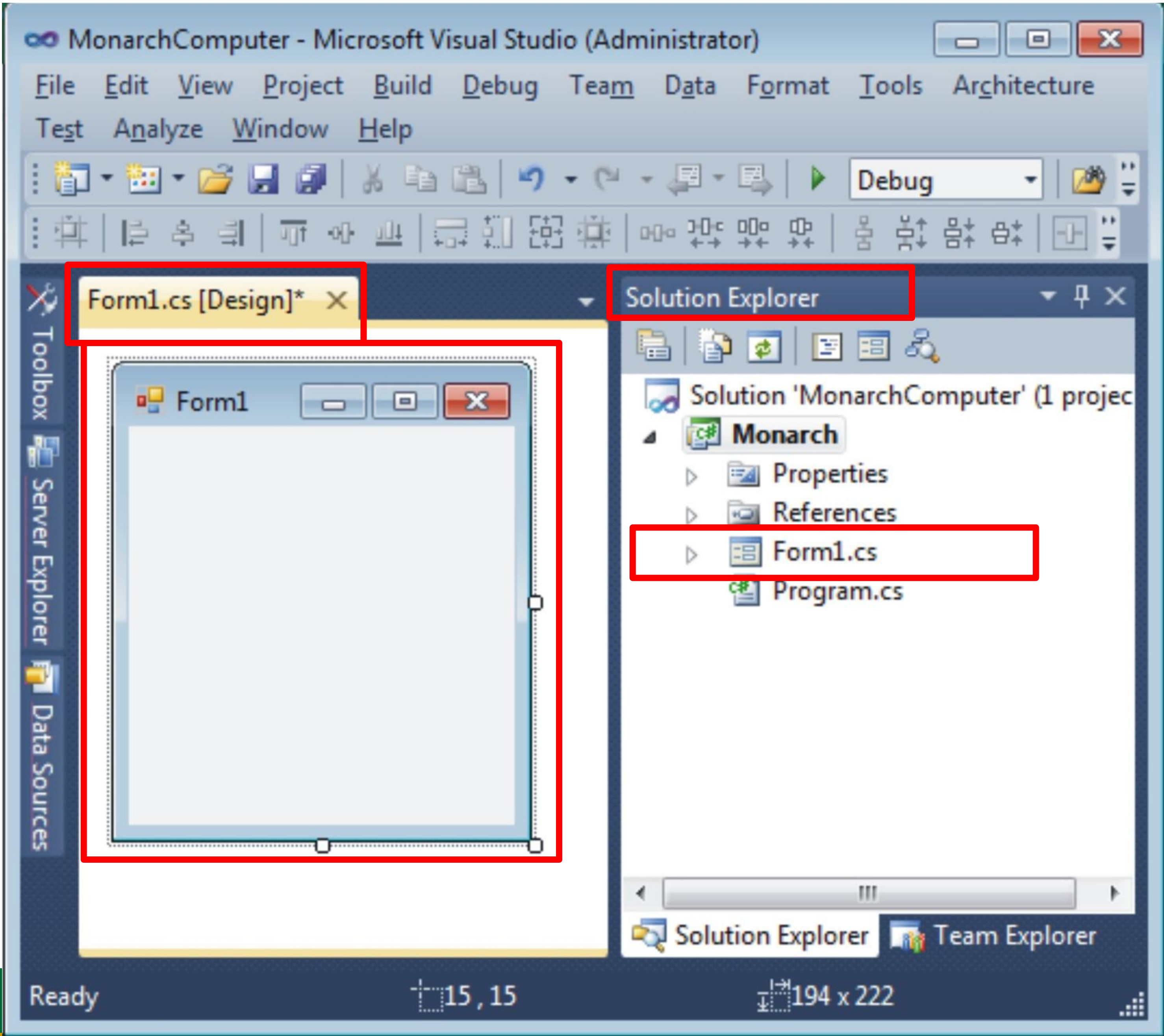
Add to source control

OK · Cancel

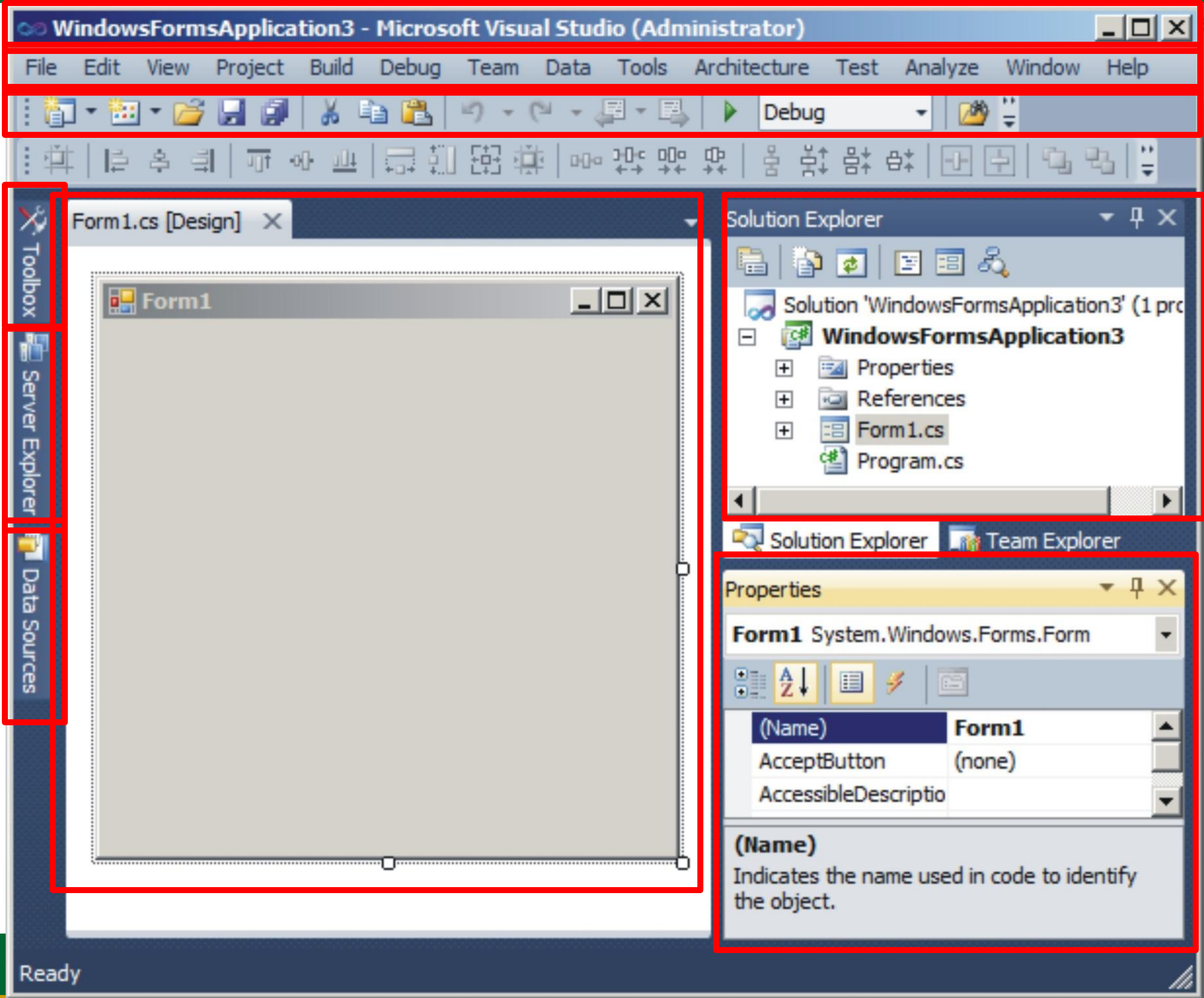
Creating a Windows FORM Application :

- Step : 3

First screen of a project.



Visual Studio 2010 IDE (Integrated Development Environment)



WindowsFormsApplication3 - Microsoft Visual Studio (Administrator)

Form1 System.Windows.Forms.Form

(Name)	Form1
AcceptButton	(none)
AccessibleDescriptio	

(Name)
Indicates the name used in code to identify the object.

Visual Studio 2010 IDE (Integrated Development Environment)

- Menubar :
  - This bar contains various menus such as **file** and **view** that help perform tasks such as creating a new project and displaying the ToolBox if it is hidden.
- Toolbar :
  - This contains command buttons that we can use to quickly perform tasks, which include **saving** a projects and **running** a Windows or Web-based application.

Visual Studio 2010 IDE (Integrated Development Environment)

- Toolbox :
  - This contains various controls such as **TextBox, Label, Timer, ListBox** that helps to create the user interface of Windows or Web Based Application.
- Solution Explorer :
  - This provides an organised view of files related to a project. In solution explorer, we can perform specific tasks such as adding an item, files or form to a project and viewing the C# code for a specific form.

Visual Studio 2010 IDE (Integrated Development Environment)

- Properties Window :
  - This window displays the properties of the windows or web forms and its controls, which have been added to the project.
  - We can add various methods for selected control using method list available in properties window.
- Sever Explorer :
  - It contains information and commands related to database and tables.

Introduction to Windows Forms :

- Windows form is a basic unit of our application.
- It provides user interface (UI) to our application.
- A form is ultimately a blank slate in that a developer enhance (વધારવું) with controls to create a user interface and with code.
- It will helpful to provide logical operations and manipulate data.

Introduction to Windows Forms :

- With windows forms we can develop smart client application.
- Smart clients are graphically rich application that are easy to deploy and update.
- Using windows forms we can design any application very easily.
- Windows forms always looks attractive and easy to use.

Introduction to Windows Forms :

- A form can represent any type of window including the **main window**, **a child window**, or a **dialog box**.
- When a new form is created, it is **empty**.
- To supply functionality, we have to add menus and controls, such as :
  - Pushbuttons, lists, and check boxes.
- We can also say that a form is a container for windows objects.

Introduction to Windows Forms :

- To build Windows-based applications with forms we have to use a namespace.
  - **System.Windows.Forms**
- To work with Windows Forms we must use three terms which are as given belongs :
  - Component
  - Container and
  - Control

Introduction to Windows Forms :

- **Component**
  - A component is an object that permits sharing between applications.
- **Container**
  - A container is an object that can hold zero or more components.
  - A container is simply a grouping mechanism.
  - Container are used throughout the windows forms namespace whenever a group of objects is required.

Introduction to Windows Forms :

- **Control**
  - A 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.

Label Control :

- Labels are generally used to provide descriptive text to the user.
- The text might be related to other controls or to provide messages.
- Usually a label is used together with a text box.
- This control is always read only that means user can not change it at runtime.

Property	Description
AutoSize	Gets or Sets whether the label should automatically resize to display its contents.
BorderStyle	Gets or Sets the border for the label. Taken from the BorderStyle enumeration. Default is None.

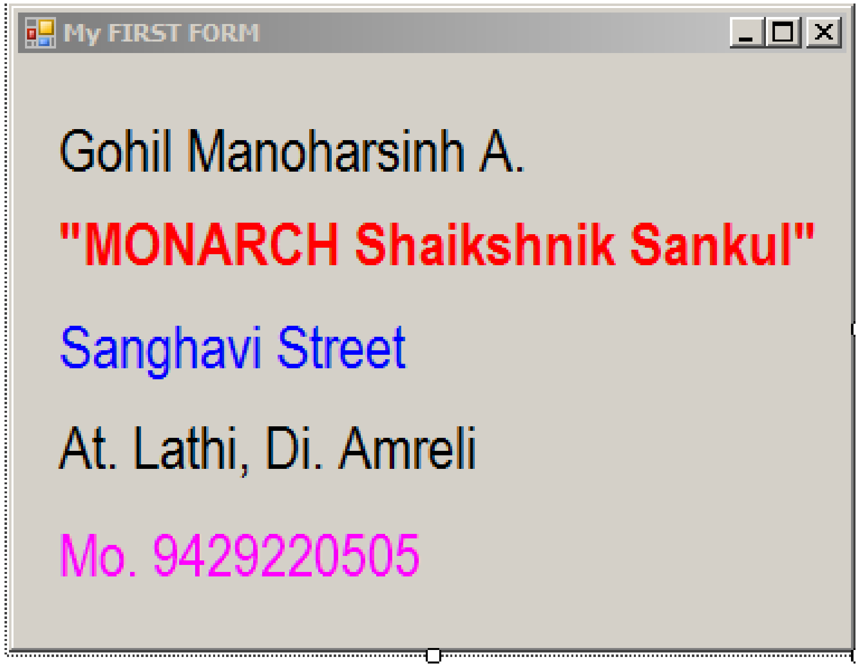
Label Control :

Property	Description
FlatStyle	Gets or Sets the flat style for the label, using the FlatStyle enumeration. Default is standard.
Image	Gets or Sets an image to appear on the label.
ImageList	Gets or Sets an ImageList object to associate with the label control.
TextAlign	Gets or Sets the alignment of text.
UseMnemonic	Gets or sets whether an ampresand (&) in the Text property is interpreted as an access key prefix character.

Def. Create a Form As Given and NOTEDOWN it's all required properties.

- Control : Form
  - Name : *frmFirst*
  - Text : My FIRST FORM
  - Size : **Width = 378**
    Height = 490
- Control : Label
  - Name : *lblName*
  - Text : Gohil Manoharsinh A.
- Control : Label
  - Name : *lblAddress1*
  - Text : "MANOHARCH Shaikshnik Sankul"
  - ForeColor : Red

(Etc.)


My FIRST FORM

Gohil Manoharsinh A.

"MONARCH Shaikshnik Sankul"

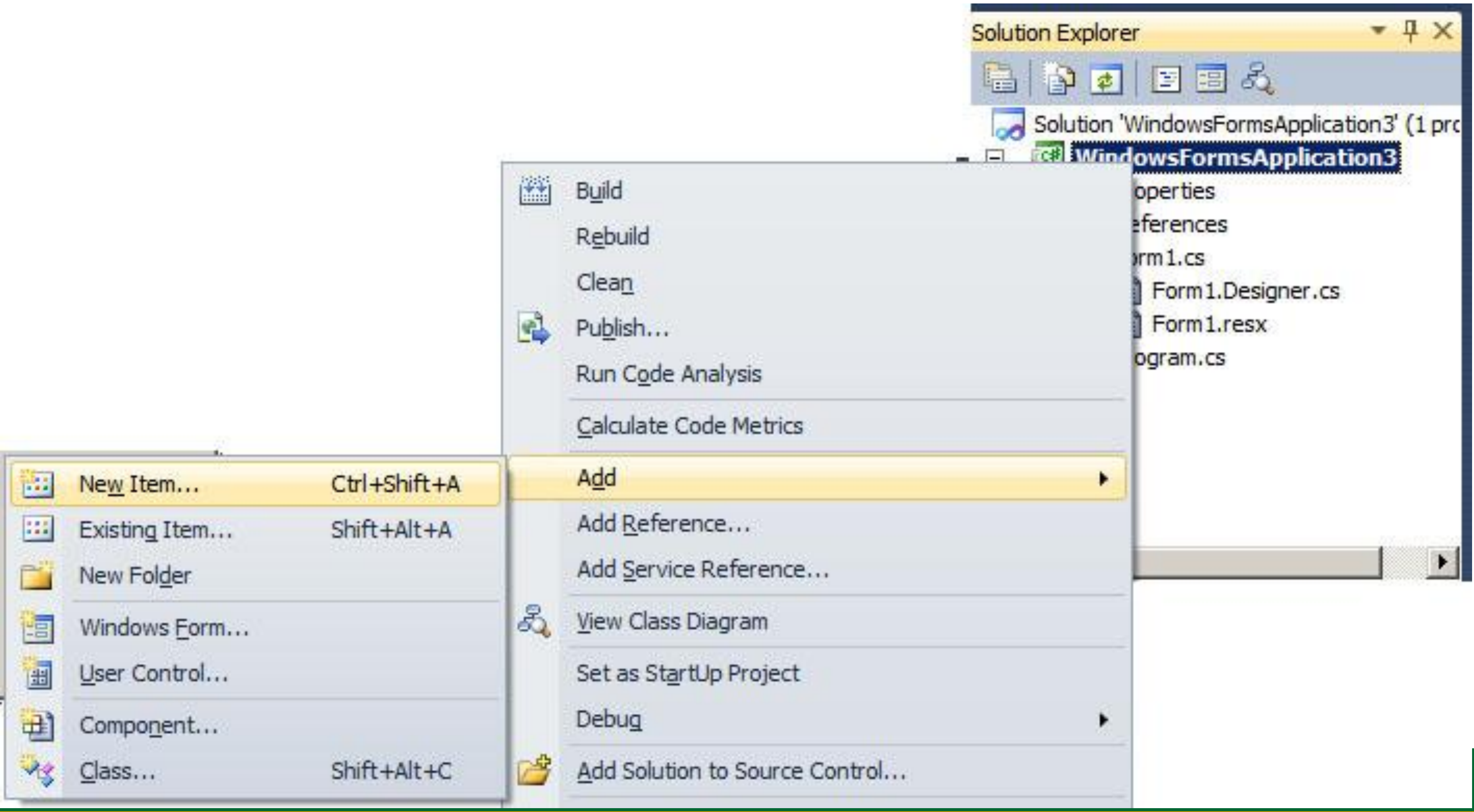
Sanghavi Street

At. Lathi, Di. Amreli

Mo. 9429220505

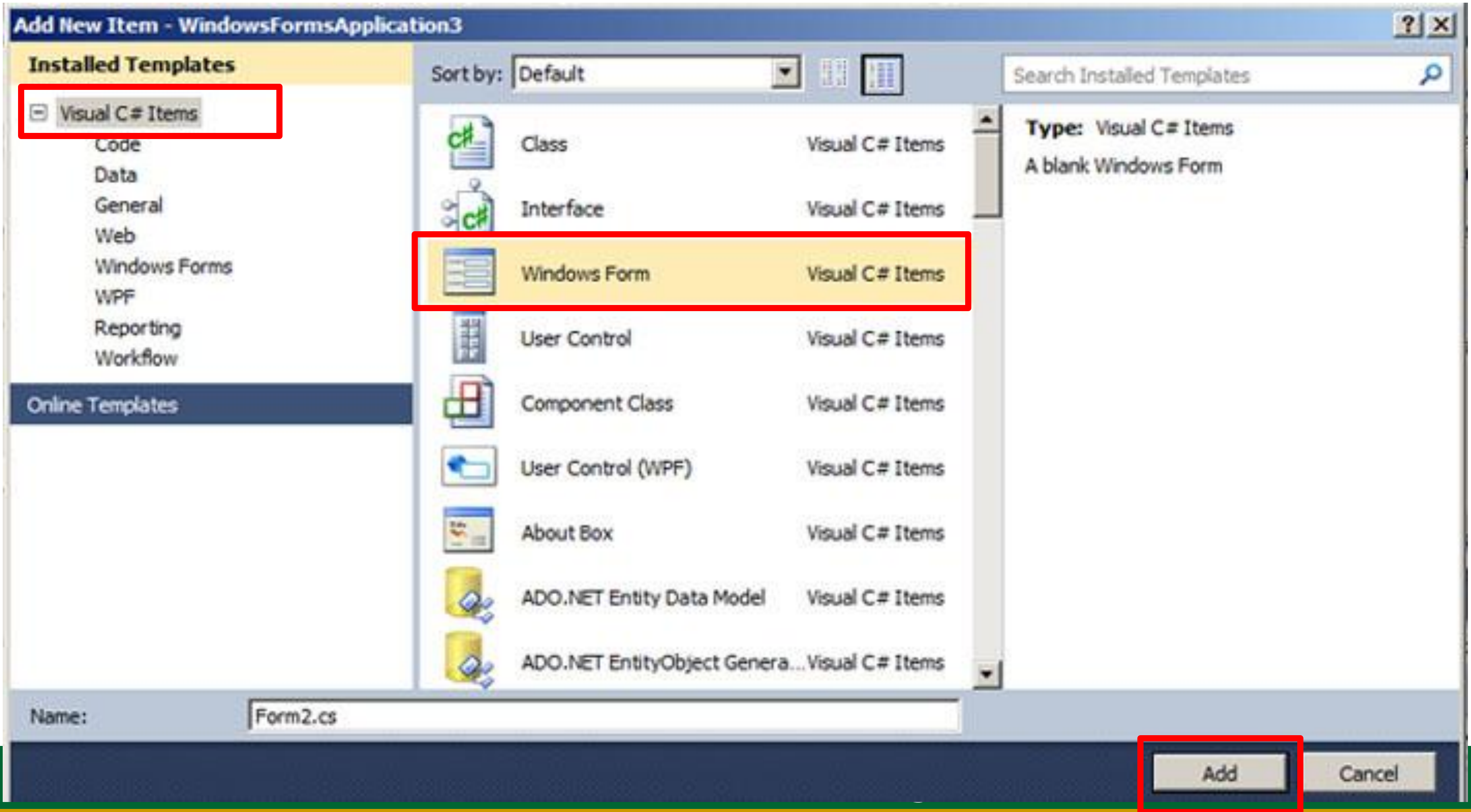
Add and Manage Multiple FORMS :

- To add a new FORM by Right CLICK :



Add and Manage Multiple FORMS :

- To add a new FORM by Right CLICK :
- Now we can see a dialog box as given.


Add New Item - WindowsFormsApplication3

Installed Templates
- Visual C# Items
  - Code
  - Data
  - General
  - Web
  - Windows Forms
  - WPF
  - Reporting
  - Workflow

Online Templates

Sort by: Default

Search Installed Templates

Class	Visual C# Items
Interface	Visual C# Items
Windows Form	Visual C# Items
User Control	Visual C# Items
Component Class	Visual C# Items
User Control (WPF)	Visual C# Items
About Box	Visual C# Items
ADO.NET Entity Data Model	Visual C# Items
ADO.NET EntityObject Genera...	Visual C# Items

Type: Visual C# Items

A blank Windows Form

Name: Form2.cs

Add and Manage Multiple FORMS :

- The above process will add a new form in the Project.
- In the form you have to design and put required forms. But when we run the form It will just display First FORM. **It will not display New Added Form**.
- To execute the New Added Form. We have to use **run method of application class.**

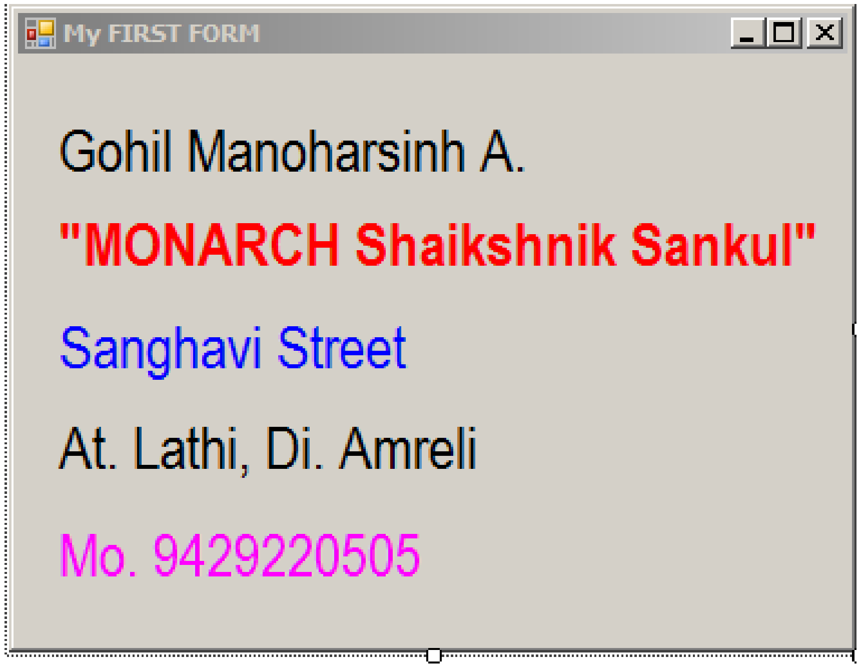
Add and Manage Multiple FORMS :

- To run an application we have to use following syntax.
- Syntax :

  Application.Run(new<**Name of FORM**>( ))
- Purpose :
  - To execute the form as requirement.
- **Where to put this run method.**
  - We have to put this method in "**Program.cs**" file which is available in **solution explorer.** It must be written in Main(). Like, **Application.Run(new Form1())**

Def. Add 2 Forms in a project. NOW set **second form** for first execution. And Notedown all the updated properties.

- Form1 Form2



My FIRST FORM

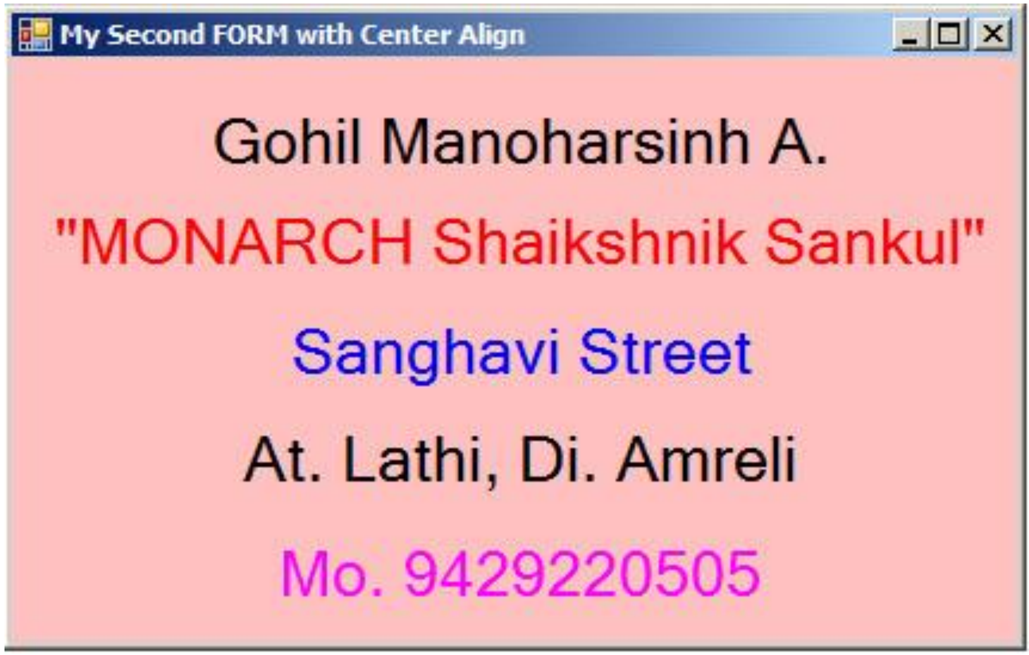
Gohil Manoharsinh A.

"MONARCH Shaikshnik Sankul"

Sanghavi Street

At. Lathi, Di. Amreli

Mo. 9429220505



My Second FORM with Center Align

Gohil Manoharsinh A.

"MONARCH Shaikshnik Sankul"

Sanghavi Street

At. Lathi, Di. Amreli

Mo. 9429220505

Add and Manage Multiple FORMS :

- How to call another Form from current form.
  - To call another form we have to use following method.
- Syntax (Define Object):

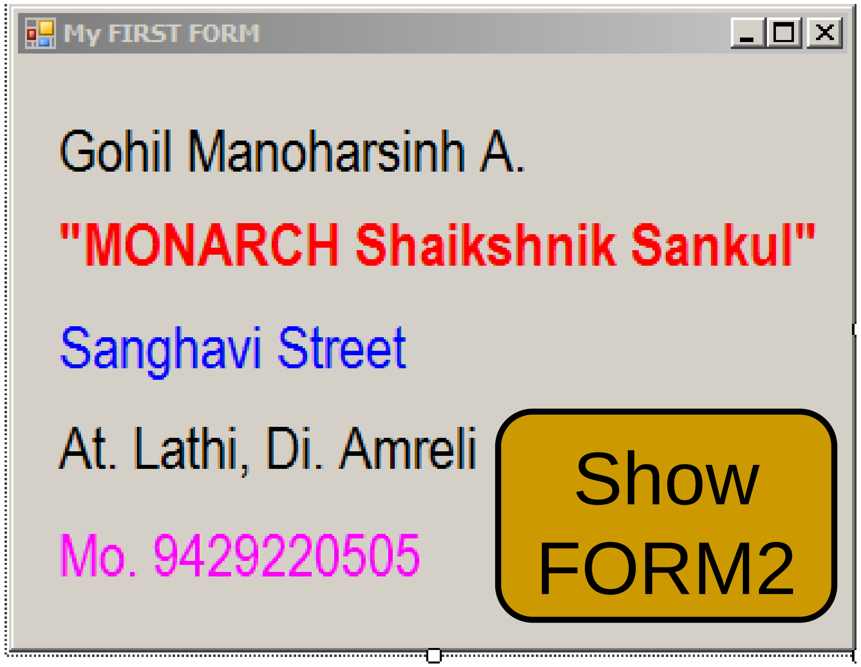
```
<FormName> NewObject=new <FormName>();
```

- Example :

```
Form1 f3 = new Form1();
    // Instantiate a Form3 object.
f3.Show();
    // Instantiate a Form3 object.
```

Def. Use to forms as given with Show button.

- Form1 Form2



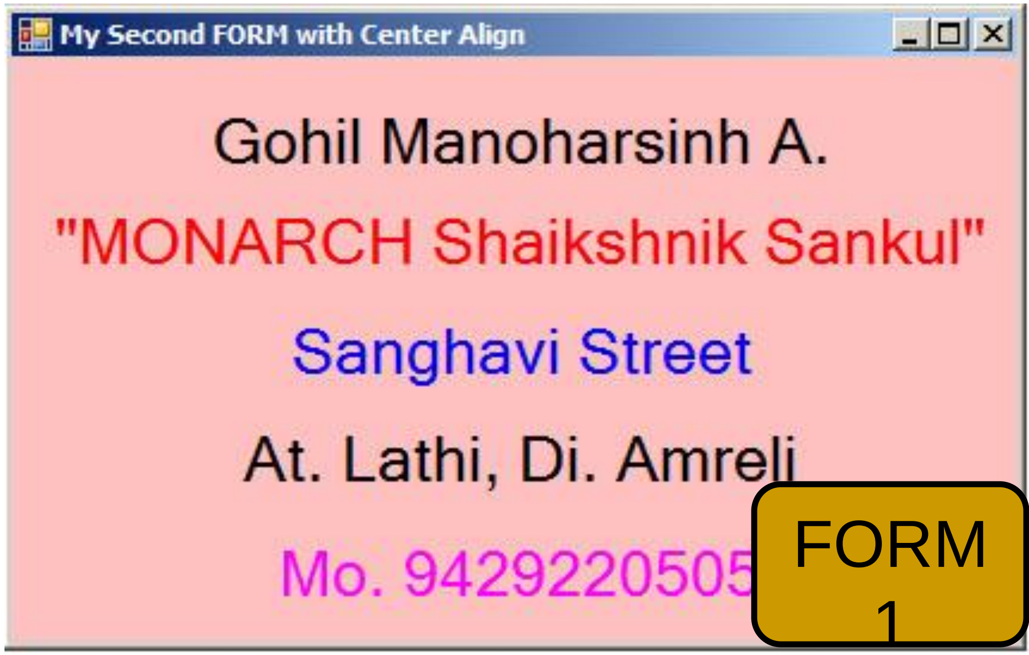
Gohil Manoharsinh A.

"MONARCH Shaikshnik Sankul"

Sanghavi Street

At. Lathi, Di. Amreli

Mo. 9429220505



Gohil Manoharsinh A.

"MONARCH Shaikshnik Sankul"

Sanghavi Street

At. Lathi, Di. Amreli

Mo. 9429220505

MessageBox

- MessageBox is a built-in feature of Windows
- A message box is a predefined window that lets you display a message.
- We can also obtain simple responses form the user, such as Yes, No or OK.
- In a window application, a message box is supported by the Message Box class.
- We cannot create an object of that class to display a message box.
- We have to call the static method **.Show()** to display message box.

Message Box

- MessageBox buttons is enumeration that defines the following values:

Ok	AbortRetryIgnore	OKCancel
YesNo	RetryCancel	YesNoCancel

- Return Value of show method :

Abort	DialogResult.Abort	Cancel	DialogResult.Cancel
Ignore	DialogResult.Ignore	No	DialogResult.No
OK	DialogResult.OK	Retry	DialogResult.Retry
Yes	DialogResult.Yes		

Message Box

- MessageBox Icon Type :

None	Asterisk	Error
Exclamation	Hand	Information
Question	Stop	Warning

Def. Demonstrate Use of MessageBox

```
 public partial class Form1 : Form
{ public Form1()
   {   InitializeComponent();  }
private void Form1_Load(object sender, EventArgs e)
{ DialogResult result = MessageBox.Show("Testing
                          of MESSAGEBOX", "Message",
                          MessageBoxButtons.OKCancel,
                          MessageBoxIcon.Information);
   if (result != DialogResult.OK)
        Application.Exit();
}
}
```

Event :

- Events occurs at time of user interaction (Run Time).
- Event-driven applications executes code in response to an event.
- Each form and control exposes a predefined set of events that you can program against.
- Some of the more common events are,
  - Click, DoubleClick, KeyDown, KeyPress, Validating, Paint, mouse events are depends upon the type of object.

Handling Mouse Event :

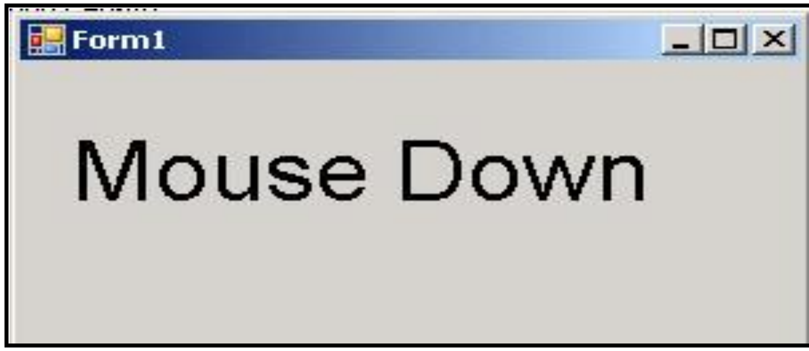
Events	Description
MouseDown	Occurs when a mouse button is pressed down while the pointer is over control.
MouseHandler	Occurs when the mouse pointer enters the control.
MouseHover	Occurs when a mouse button is remains, or hovers, over a control for a configurable amount of time.
MouseLeave	Occurs when the mouse pointer leaves the control.

Handling Mouse Event :

Event	Description
MouseMove	Occurs when the mouse pointer moves over the control.
MouseUp	Occurs when a mouse button is released while the pointer is over the control.
MouseWheel	Occurs when a mouse wheel moves while the control has focus.

- ***To add various events we can use EVENTS from Property Window…***

Def. Demonstrate use of mouse Events



Like…

```
private void Form1_MouseDown(object sender,
                                        MouseEventArgs e)
{
        label1.Text = "Mouse Down";
}
```

Handling Key Event :

- To handle keyboard events only at the form level and not enable other controls to receive keyboard events, set the KeyPressEventArgs Handled property in your form's KeyPress event-handling method to true.
- Key Event

Events	Description
KeyDown	Occurs when a key on the keyboard is pressed down.
KeyPress	Occurs when a character is pressed on the keyboard, and again each time the character is repeated while it continues to be pressed.
KeyUp	Occurs when a key on the keyboard is released.

Handling Key Event :

- ***Def. Demonstrate User of Key Down/Up event.***

- KeyPress Event:
  - The KeyPressEventArgs class is the event argument class associated with the KeyPress event.
  - This class represents the keyboard character pressed by the user.
  - It is part of the System.
  - **Windows.Forms** namespace, and inherits from the **System.EventArgs** class.

Handling Key Event :

- KeyPress Event:
  - **KeyEventArgs e**
  - E will store the KeyChar of given key…
- ***Def. Write a program to find KeyAscii for pressed key.***

```
private void textBox1_KeyPress
            (object sender, KeyPressEventArgs e)
{ label1.Text=((int)e.KeyChar).ToString();
}
```

Def. Validate data entry using KeyPress

- Coding to Enter Only **Numeric** & **BackSpace**

```
private void textBox1_KeyPress
               (object sender, KeyPressEventArgs e)
{ char Ch = e.KeyChar;
  if (!Char.IsDigit(Ch) && Ch != 8)
             e.Handled = true;
}
```

- Coding to Enter Only **Alpahbets** & **BackSpace**

```
private void textBox1_KeyPress ( )
{ char Ch = e.KeyChar;
  if (!Char.IsLetter(Ch) && Ch != 8)
             e.Handled = true;  }
```

Def. Validate data entry using KeyPress

- Enter Only **UPPER case Letter** & **BackSpace**

```
private void textBox1_KeyPress( )
{ char Ch = e.KeyChar;
  if (!Char.IsUpper(Ch) && Ch != 8)
               e.Handled = true;  }
```

- Enter Only **Lower case Letter** & **BackSpace**

```
private void textBox1_KeyPress( )
{ char Ch = e.KeyChar;
  if (!Char.IsUpper(Ch) && Ch != 8)
               e.Handled = true;  }
```

Basic Control Programming

- In general, the functionality of a window is expressed by two types of items: Controls and Menus.
- It is through these items that a user interactions with your program.
- Windows defines many different types of controls, including push buttons, checkboxes, radio buttons and list box etc.

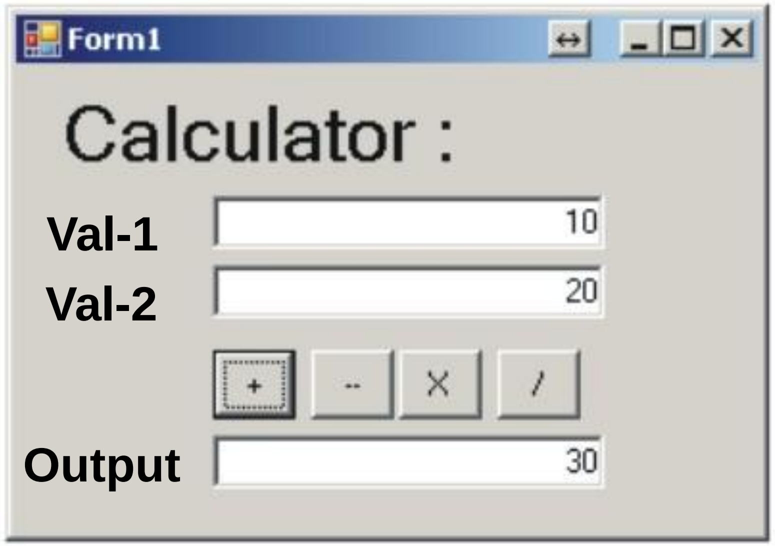
Button Control :

- The Button Control allows the user to click it to perform an action.
- When the button is clicked, it looks as if it is being pushed in and released.
- Whenever the user clicks a button, the Click event handler is invoked.
- The ButtonBase class represents a control that can be displayed as a button.

Button Control :

Property	Description
FlateStyle	Gets or Sets the flat style appearance of the button.
Image	Gets or Sets an image to display on the Button.
ImageAlign	Gets or Sets the alignment of an image on the button.
ImageIndex	Gets or Sets an image to display on the button as an index into the ImageList property.
ImageList	Gets or Sets an ImageList object to associate with the button control.
TextAlign	Gets or Sets the alignment of text.

Def. Design A Calculator Using Button



```
private void btnPlus_Click(object sender, EventArgs e)
{ textBox3.Text =
      (Int32.Parse(textBox1.Text) +
      Int32.Parse(textBox2.Text)).ToString();
}
```

TextBox Control

- The TextBox control is one of the most used controls from the basic controls.
- TextBox control is derived from TextBoxBase.
- Windows Forms text boxes are used to get input from the user or display text.
- TextBox may contain editable, read-only and also be multi line.

TextBox : Public Properties

AcceptsTab	Gets or Sets whether a multiline text box displays a Tab character or moves focus to the next control when the Tab key is pressed.
CanUndo	Gets or sets whether the user can undo the previous edit performed in the text box.
Lines	Gets or sets the array of strings representing the lines of text in the control.
MaxLength	Gets or sets the maximum number of characters the control will accept.
Multiline	Gets or sets whether this is a multiline text box.

TextBox : Public Properties

ReadOnly	Gets or sets whether the text is read-only.
SelectedText	Gets or sets the currently selected text in the control. The SelectedStart property indicates the location of the first selected character.
WordWrap	Gets or sets whether a multiline control automatically wraps to the next line as required.

PublicMethods of TextBoxBase class:

AppendText	Appends a string to the existing text in the control
Copy	Copies the current text into the Clipboard.

Paste	Replaces the current selection with the contents of the Clipboard.
ScrollToCaret	Ensures the current caret position is visible in a multiline text box.
Select All	Selects all text in the control. The Select method can be used to select a substring.
Undo	Undoes the last edit operation in the text box.

Public Events of TextBoxBase class:

AcceptsTab Changed	Occurs when the AccptsTab property changes.
MultilineChanged	Occurs when the Multiline property changes.

Publie Properties of TextBox class:

AcceptsReturn	Gest or sets whether the Enter key in a multiline text box adds a new line of text or activates the default button for the form.
CharacterCasing	Gets or sets how the control modifies the case of entered characters. This can be used to display all uppercase or lowercase letters in the text box.
PasswordChar	Gets or sets the character used to mask the display in the control. When this property is set, cutting or copying to the clipboard is disabled.
ScrollBars	Gets or sets which scrollbars should appear in a multiline textbox.
TextAlign	Gets or sets how displayed text is aligned within the control.

TextBox Events :

- Generally we can use with TextBox
  - GotFocus (**Enter**)
  - LostFocus (**Leave**)
  - KeyPress
  - KeyDown
  - KeyUp events.

- **Def.** ***Set Back Color Using Enter and Leave events.***

```
private void textBox1_Enter(object sender,
   EventArgs e)
{ textBox1.BackColor = Color.Violet;}
```

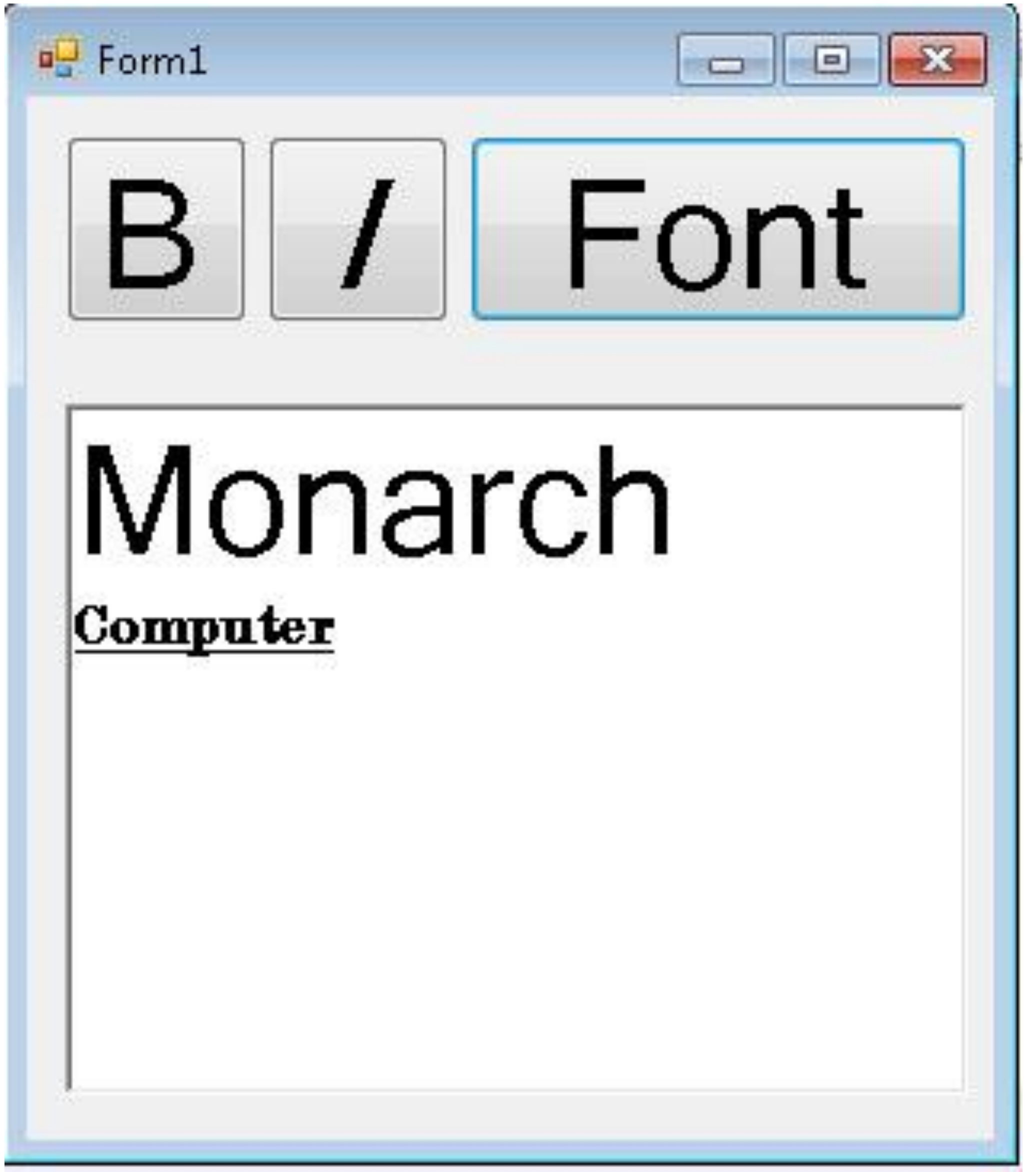
RithTextBox

- The RichTextBox is a text editing control that can handle special formatting features.
- The windows forms RichTextBox control is used for displaying, entering and manipulating text with formatting.
- RichTextBox is very similar to TextBox control but additionally it can display fonts, colors, and links; load text and embedded images from a file; and find specified characters.
- RichTextBox control is used to provide text manipulation and display features similar to word processing application.

RithTextBox

- As the name implies, the RichTextBox control uses Rich Text Format (RTF) to handle the special formatting.
  - We can make formatting changes by using the selection properties:
    - SelectionFont, SelectionColor, SelectionBullet, Paragraph formatting with selectionIndent, SelectionRightIndent, and SelectionHangingIndent.
  - To manipulate files, the LoadFile and SaveFile methods can display and write multiple file formats including plain text, Unicode plain text, and Rich Text Format.

Def. Desing a Demo for RichTextBox



Monarch

Computer

CODING….

```
 private void btnBold_Click(object sender, EventArgs e)
{ Font bfont = new Font(richTextBox1.Font,
                                        FontStyle.Bold);
  Font rfont = new Font(richTextBox1.Font,
                                        FontStyle.Regular);
  if(richTextBox1.SelectedText.Length == 0)
       return;
  if (richTextBox1.SelectionFont.Bold)
       richTextBox1.SelectionFont = rfont;
  else
       richTextBox1.SelectionFont = bfont;
}
```

RadioButton :

- RadioButton is also derived from ButtonBase.
- Radio buttons are generally used as a group.
- It is also known as option buttons and allows the user to choose one of several options.
- When you have multiple Radio Button controls in the same container, only one at a time may be selected.

RadioButton :

Property	Description
Appearance	Gets or Sets whether the control appears as a normal radio button or as a toggle button.
AutoCheck	Gets or Sets the behavior of related radio buttons when this button is clicked.
CheckAlign	Gets or Sets the alignment of the click box portion of the control.

Radio Button:

- Public Method Of RadioButton :
  - PerformClick
    - Stands a click event to the control.
- Public Events of RadioButton :
  - AppearanceChanged
    - Occurs when the Appearance property chages.
  - CheckedChanged
    - Occurs when the checked property change.

Def. Demonstrate Use of RadioButton

- Set Font Color to Red
- Set Font Color to Green
- Set Font Color to Blue

Coding :

```
private void rbtnBlue_CheckedChanged(object sender, EventArgs e)
{
     if (rbtnBlue.Checked == true)
       textBox1.ForeColor = Color.Blue;
}
```

CheckBox :

- The CheckBox control is also derived from ButtonBase.
- The ckeckbox control indicates whether a particular condition is on or off.
- This is mostly used to present True/False selection to the user.
- You can use check box controls in group to display multiple choices from which the user can select one or more.

CheckBox :

Property	Description
Appearance	Gets or Sets whether the control appears as a normal radio button or as a toggle button.
AutoCheck	Gets or Sets whether the control is checked automatically or manually. Default is true.
Checked	Gets or Sets whether the control is checked.
CheckState	Gets or Sets the state of a three-state check box as a CheckState enumeration value.

CheckBox :

Property	Description
ThreeState	Gets or sets whether the check box displays three states. The default is false.

- Public Events of CheckBox :
  - CheckedChanged
    - Occurs when the checked property changes.
  - CheckStateChanged
    - Occurs when the CheckState property changes.

Def. Demonstrate use of CheckBox

- Set Font Bold
- Set Font Italic
- Set Font UnderLine
- **Coding :**

```
private void checkBox1_CheckedChanged(object sender,
    EventArgs e)
{ if (checkBox1.Checked == true)
    TextBox1.Font=new Font(textBox1.Font, FontStyle.Bold);
    else
    TextBox1.Font=new Font(textBox1.Font, FontStyle.Regular);
}
```

ComboBox, ListBox and Checked List Box

- ComboBox, ListBox, and CheckedListBox are all derived from the **ListControl class**
- If there is a need to have multiple selections or if the user needs to be able to see several items in the list at any time, these all are best for that.
- **ListControl :**

Public Properties of ListControl Class:

Property	Description
DataSource	Gets or Sets the data source for this control. When set, the individual items cannot modified.
Display Member	Gets or Sets the property to use when displaying objects in the combo control.
SelectedIndex	Gets or Sets the zero based index of the object selected in the control.
SelectedValue	Gets or Sets the value of the object selected in the control.
ValueMember	Gets or Sets the item value in Combo control.

Public Method of ListControl Class:

Property	Description
GetItemText	Returns the text associated with a given item, based on the current DisplayMember property setting.

Public Event of ListControl Class:

Property	Description
DataSourceChaged	Occurs when the DisplaySource property changes.
DisplayMemberChanged	Occures when the DisplayMember property changes.

ListBox Control :

- A ListBox control displays a list from which the user can select one or more items.
- If the total number of items exceeds the number that can be displayed, a scroll bar is automatically added to the ListBox control.

Public Static Fields :

Property	Description
DefaultItemHeight	The default item height for an owner-drawn ListBox object.
No Matches	The value returned by ListBox methods when no matches are found during a search.

ListBox Control : Public Properties

Property	Description
DrawMode	Gets or Sets how this list box should be drawn.
ItemHeight	Gets or Sets the height of an item in the list box.
Items	Gets the collection of items to display.
MultiColum n	Gets or Sets whether this list box should support multiple columns.
SelectedIn dices	Gets a collection of zero based indices for the items selected in the list box.

ListBox Control : Public Properties

Property	Description
SelectedItem	Gets or Sets the currently selected object.
SelectedItems	Gets a collection of all items selected in the list.
SelectionMode	Gets or Sets how items are selected in the list box.
TopIndex	Gets the index of the first visible item in the list.

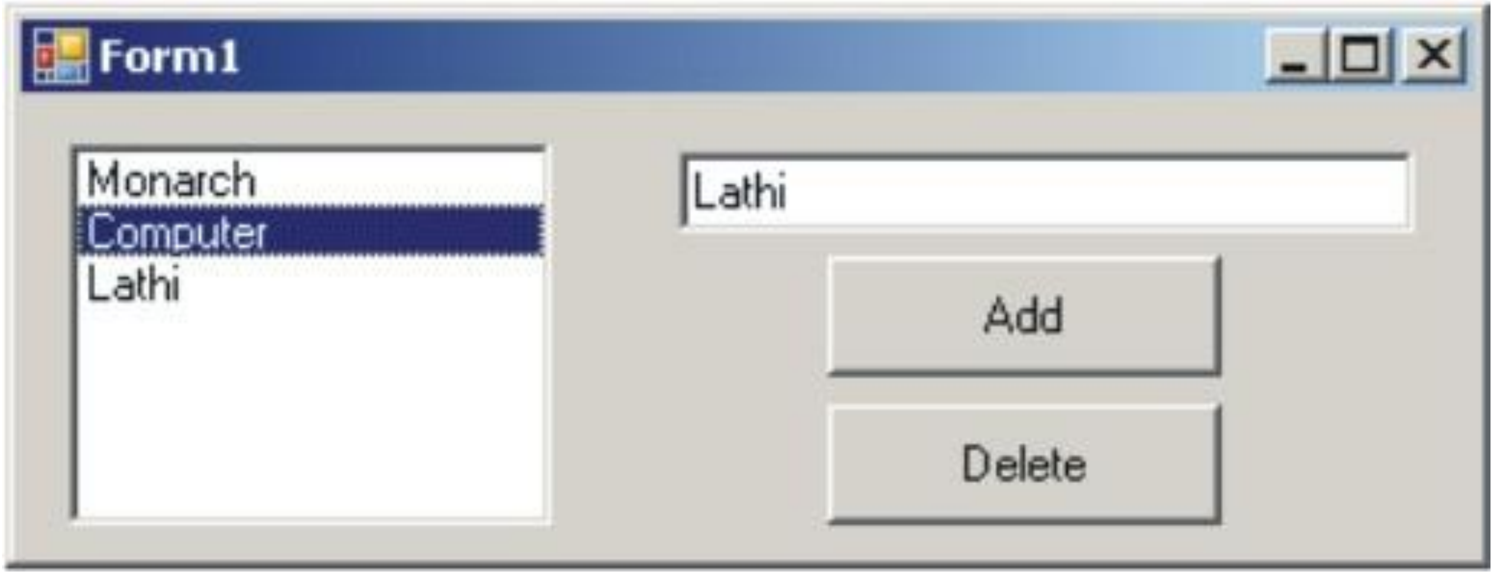
ListBox Control : Public Methods

Property	Description
BeginUpdate	Prevents the control from painting its contents while items are added to the list box.
ClearSelected	Deselects all selected items in the control.
FindString	Returns the index of the first item with a display value beginning with a given string.
GetSelected	Indicates whether a specified item is selected.
IndexFromPoint	Returns the index of the item located at the specified coordinates.
SetSelected	Selects or deselects a given item.

ListBox Control : Public Events

Property	Description
DrawItem	Occurs when an item in an owner-drawn list box requires painting.
MeasureItem	Occurs when the size of an item in an owner-drawn list box is required.
SelectedIndexChanged	Occurs whenever a new item is selected in the list box, for both single and multiple selection boxes.

Def. Demonstrate Use of ListBox



```
private void btnAdd_Click(object sender, EventArgs e)
{       listBox1.Items.Add(txtData.Text);          }
private void listBox1_Click(object sender, EventArgs e)
{       MessageBox.Show(listBox1.Text);          }
private void btnDel_Click(object sender, EventArgs e)
{    listBox1.Items.Remove(listBox1.SelectedItem);}
```

ComboBox Control :

- The **ComboBox** control is used to display data in a drop down combo box.
- By default, the Combobox control appears in two parts:
  - Top part is a text box
  - Second part is a list box that displays a list of items to select by user.

Public Properties :

Property	Description
DrawMode	Gets or sets how elements in the list are drown in a windows.

ComboBox Control :

Public Properties :

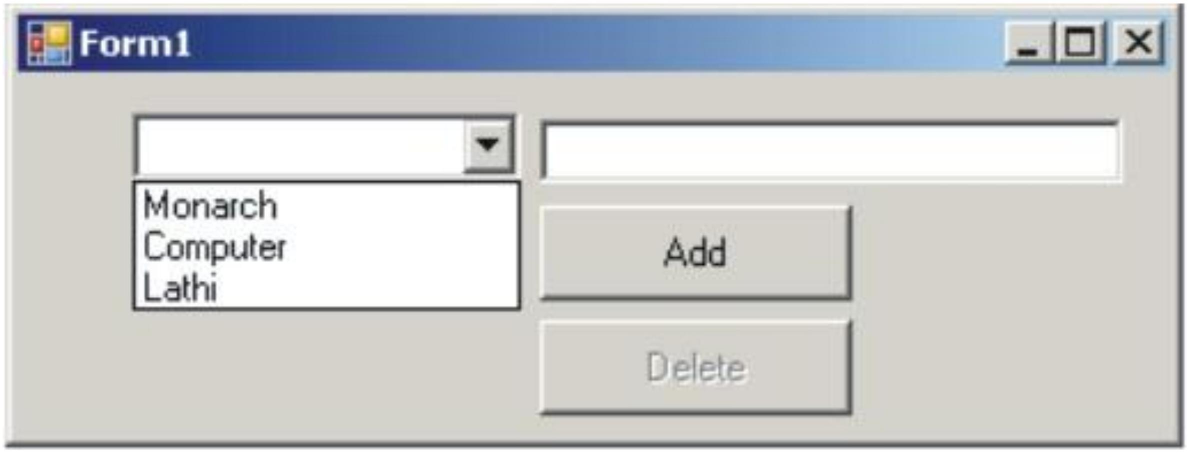
Property	Description
DropDownStyle	Gets or sets the style used to display the edit and list box controls in the combo box.
DropDownWidth	Gets or sets the width of the list box portion of the control.
DroppedDown	Gets or sets whether the combo box is currently displaying its list box portion.
Items	Gets or sets the collection of items contained by this combo box.

ComboBox Control :

Public Properties :

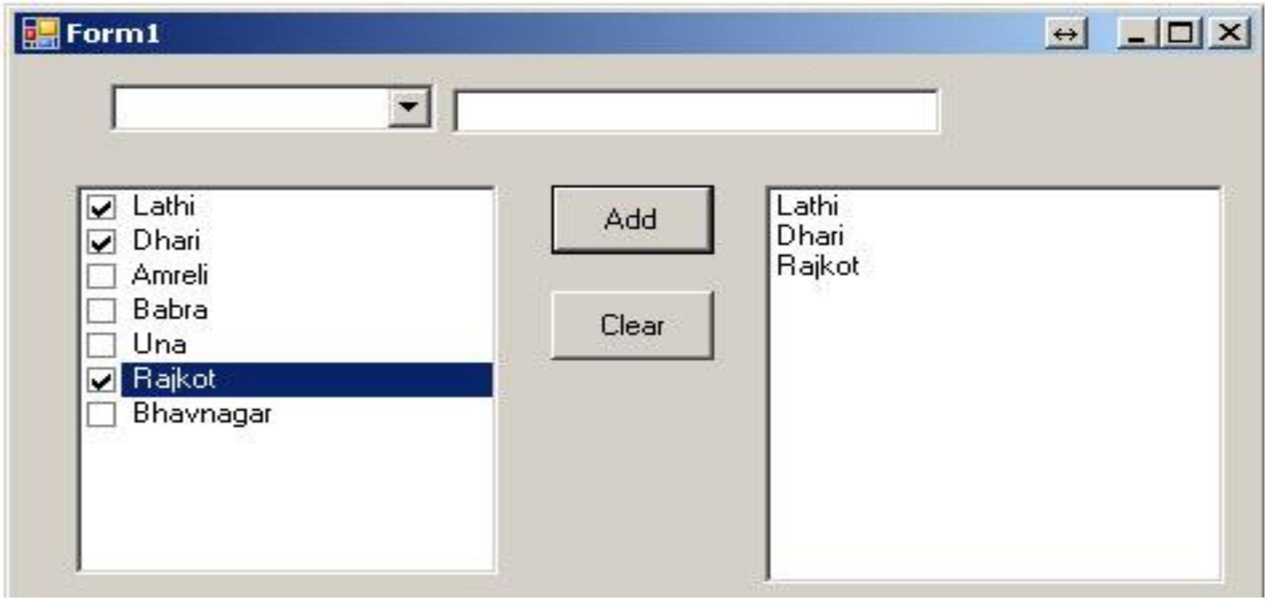
Property	Description
MaxDropDown Items	Gets or sets the maximum number of items permitted in the list box portion of the control.
MaxLength	Gets or sets the maximum number of characters permitted in the text box portion of the control.
SelectedItem	Gets or sets the currently selected item in the control.
SelectedText	Gets or set any text that is selected in the text box portion of the control.

Def. Demonstrate Use of ComboBox



```
private void btnAdd_Click(object sender, EventArgs e)
{     comboBox1.Items.Add(txtData.Text);        }
private void btnDel_Click(object sender, EventArgs e)
{
comboBox1.Items.Remove(comboBox1.SelectedItem);
}
```

Def. Demonstrate Use of CheckedList



- [✓] Lathi
- [✓] Dhari
- [ ] Amreli
- [ ] Babra
- [ ] Una
- [✓] Rajkot
- [ ] Bhavnagar

```
Lathi
Dhari
Rajkot
```

```
 private void btnAdd_Click(object sender,
   EventArgs e)
{ if (this.chkListBox1.CheckedItems.Count > 0)
   {   foreach(string item in
                           chkListBox1.CheckedItems)
             listBox1.Items.Add(item.ToString());
   }}
```

Def. Demonstrate Use of CheckedList

```
private void btnClearList_Click
                    (object sender, EventArgs e)
{ for (int i = 0; i < chkListBox1.Items.Count; i++)
      chkListBox1.SetItemChecked(i, false);
  listBox1.Items.Clear();
}
```

ListView

- The ListView control display a list of items with icons.
- You can use a list view to create a user interface like the right pane of Windows Explorer.
- The control has four view modes: LargeIcon, SmallIcon, List and Details.

ListView : Public Properties

Property	Description
Activation	Gets or Sets how an items is activated, and whether the font changes as the mouse passes over the item.
CheckBoxes	Gets or Sets whether a check box is displayed next to each item.
Columns	Gets the collection of ColumnHeader components associated with the control.
HeaderStyles	Gets or Sets the column header style for the control. Default is ColumnHeaderStyle.Clickable.

ListView : Public Properties

Property	Description
Items	Gets the collection of items in the list.
LabelEdit	Gets or Sets whether the user can edit item labels in the list.
MultiSelect	Gets or Sets whether multiple items in the list may be selected at the same time.
SelectedItems	Gets the collection of items selected in the list.
Sorting	Gets or Sets how items in the list are sorted, if at all.
View	Gets or Sets the current view enumeration value for the list.

ListView : Public Events

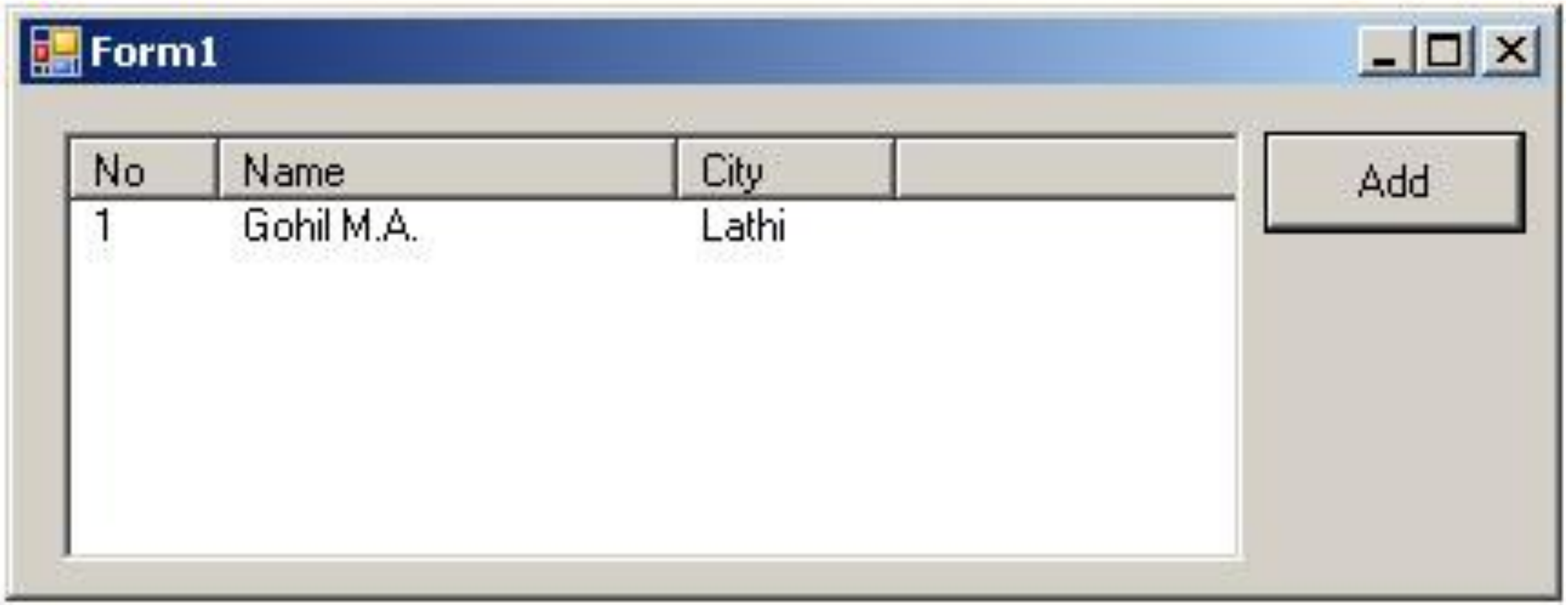
- Events

Events	Description
AfterLabelEdit	Occurs after an item label has been edited.
ColumnClick	Occurs when the user clicks a column header in the Details view.
ItemActive	Occurs when an item is activated.
ItemDrag	Occurs when a user begins dragging an item in the list.
SelectedIndexChanged	Occurs when the selection state of an item changes.

Basic ListView : Public methods

- Clear
  - Removes all items and columns for the list view control.
- EnsureVisible
  - Ensures a given item is visible, scrolling it into view if necessary.

Def. Add Your No, Name, City in ListView



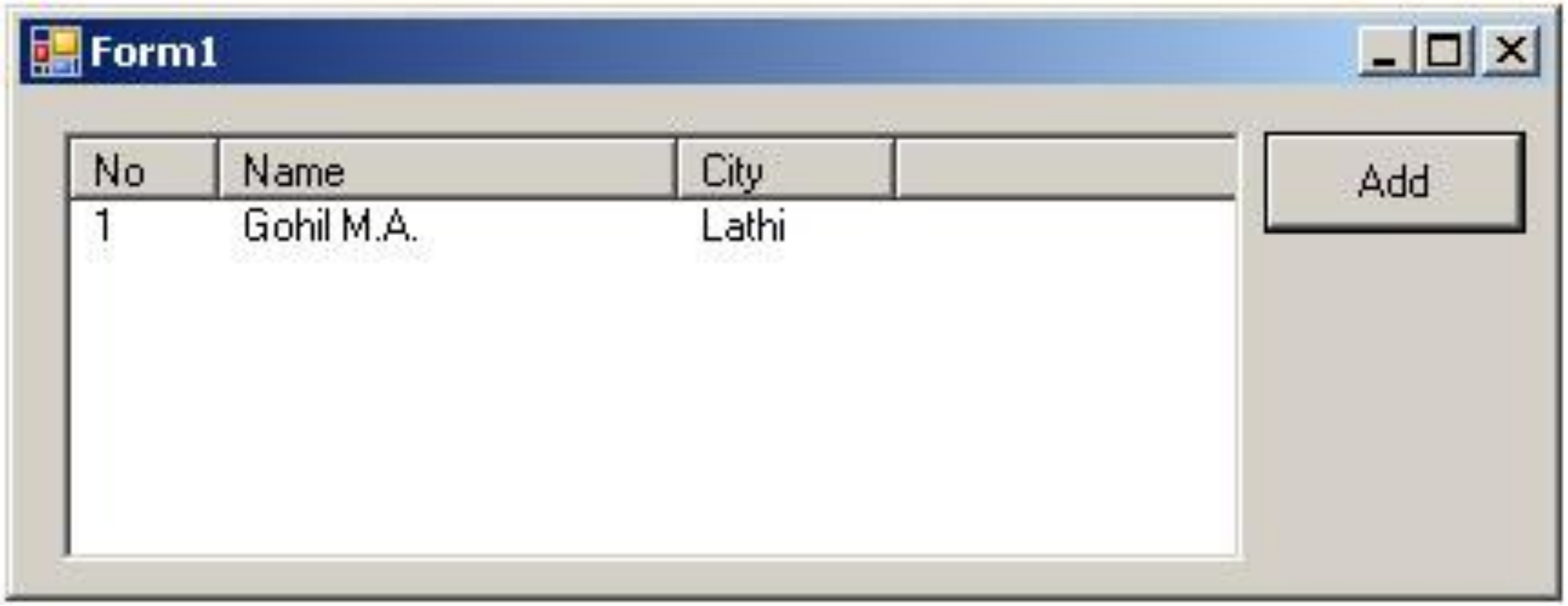
No	Name	City
1	Gohil M.A.	Lathi

//Properties of Listview

```
listView1.View = View.Details;
listView1.FullRowSelect = true;
```

//Add Column TITLES

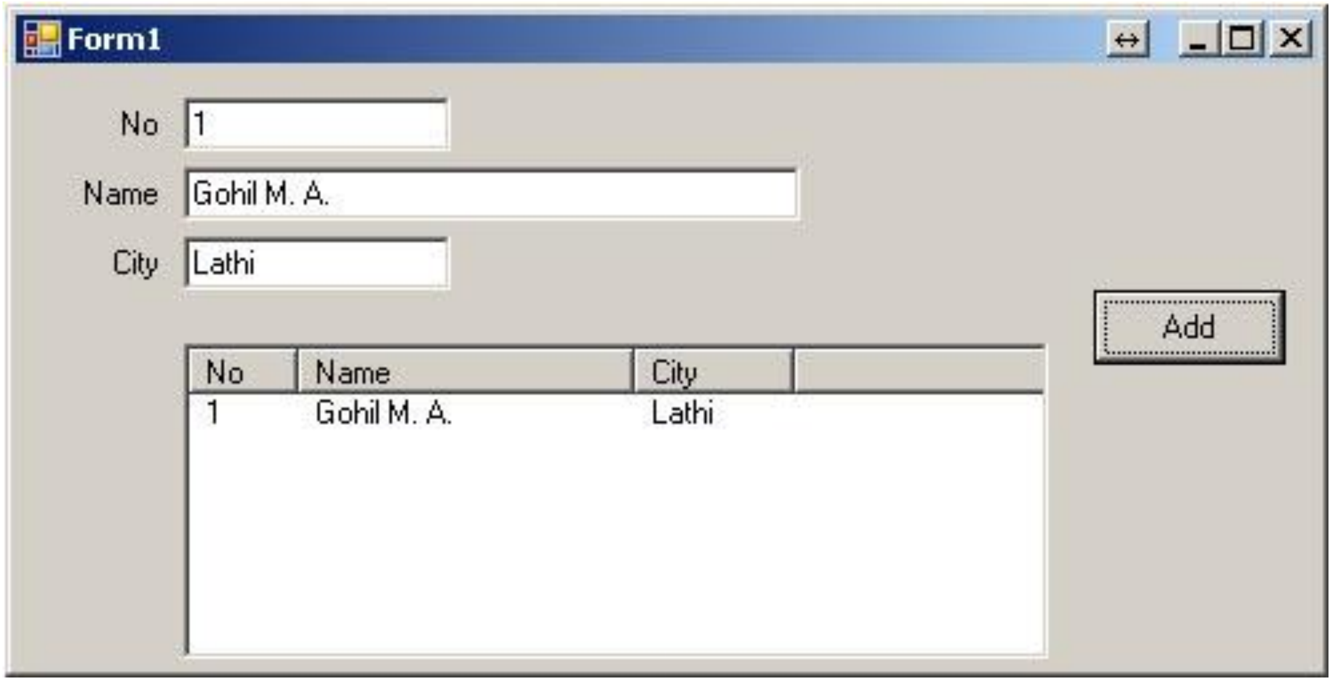
```
listView1.Columns.Add("No", 50);
listView1.Columns.Add("Name", 250);
listView1.Columns.Add("City", 250);
```



No	Name	City
1	Gohil M.A.	Lathi

```
private void Add_Click(object sender, EventArgs e)
{ ListViewItem lv = new ListViewItem("1");
  lv.SubItems.Add("Gohil M.A.");
  lv.SubItems.Add("Lathi");
  listView1.Items.Add(lv);
}
```

Def. Add Your No, Name, City in ListView using TEXTBOX



No: 1
Name: Gohil M. A.
City: Lathi

No	Name	City
1	Gohil M. A.	Lathi

```
private void Add_Click(object sender, EventArgs e)
{  ListViewItem lv = new ListViewItem(txtNo.text);
   lv.SubItems.Add(txtName.text);
   lv.SubItems.Add(txtCity.text);
   listView1.Items.Add(lv); }
```

Def. Add Your No, Name, City in ListView using TEXTBOX. Add Delete selected Item.

```
private void listView1_Click(object sender, EventArgs e)
{  txtNo.Text = listView1.SelectedItems[0].SubItems[0].Text;
   txtName.Text = listView1.SelectedItems[0].SubItems[1].Text;
   txtCity.Text = listView1.SelectedItems[0].SubItems[2].Text;
}
 private void btnDel_Click(object sender, EventArgs e)
{  if (listView1.SelectedItems.Count == 0) return;
        listView1.SelectedItems[0].Remove();   }
```

Note : To Set Focus we can use following syntax :

Syntax : **<objectName>.focus();**

TreeView

- The TreeView control used to display a hierarchy of nodes to users, like the way fields and folders are displayed in the left pane of windows Explorer.
- Each node in the tree view might contain other nodes, called child nodes.
- You can display parent nodes, or nodes that contain child nodes, as expanded or collapsed.

TreeView

Property	Description
CheckBoxes	Gets or Sets whether a check box is displayed next to each node in the tree.
HideSelection	Gets or sets whether a selected node remains highlighted even when the control does not have focus.
ImageIndex	Gets or Sets an index into the tree’s image list of the default image to display by a tree node.
LabelEdit	Gets or sets whether node labels can be edited.

TreeView

Property	Description
Nodes	Gets the collection of TreeNode objects assigned to the control.
SelectedNode	Gets or sets the selected tree node.
ShowPlusMinus	Gets or Sets whether to indicate the expansion state of parent tree nodes by drawing a plus '+' or minus '-' sign next to each node.
Sorted	Gets or sets whether the nodes are sorted alphabetically based on their label text.

TreeView

Events	Description
AfterExpand	Occurs after a tree node is expanded.
AfterLabelEdit	Occurs after a tree node label is edited.
BeforeCollapse	Occurs before a tree node is collapsed.
BeforeSelected	Occurs before a tree node is selected.

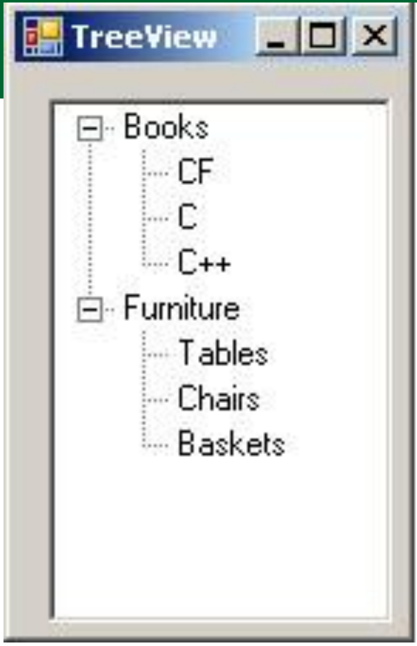
TreeView

Events	Description
AfterExpand	Occurs after a tree node is expanded.
AfterLabelEdit	Occurs after a tree node label is edited.
BeforeCollapse	Occurs before a tree node is collapsed.
BeforeSelected	Occurs before a tree node is selected.

TreeView

- Basic Methods
  - CollapseAll
  - GetNodeAt
  - GetNodeCount

Def. Add following root node and child node in given tree



```
treeView1.Nodes.Add("Books");
   treeView1.Nodes[0].Nodes.Add("CF");
   treeView1.Nodes[0].Nodes.Add("C");
   treeView1.Nodes[0].Nodes.Add("C++");

treeView1.Nodes.Add("Furniture");
   treeView1.Nodes[1].Nodes.Add("Tables");
   treeView1.Nodes[1].Nodes.Add("Chairs");
   treeView1.Nodes[1].Nodes.Add("Baskets");
```

ImageList :

- The ImageList is used to store images, which can then be displayed by controls.
- And Image List allows you to write code for a single, consistent catalog of Images.
- For example, you can rotate Images displayed by a button control simply by changing the button’s Image index or Image key property.

ImageList :

Property	Description
ColorDepth	Gets or Sets the color depth for images in the list.
Handle	Gets the Win32 handle for the image list.
HandleCreated	Gets whether the underlying Win32 handle has been created.
Images	Gets the collection of images for the image list. Use this collection to add, remove, and otherwise manage the list's images programmatically.
ImageSize	Gets or sets the size for images in the list.

ImageList :

- Basic Methods
  - Draw
- Basic Events
  - RecreateHandle

ImageList :

- To add Image List
  - Goto Tool Box :
    - Add Image List control to form.
    - Set Image List's **Images** properties and select images from your computer.
    - Add ImageList to Tree View or List View.

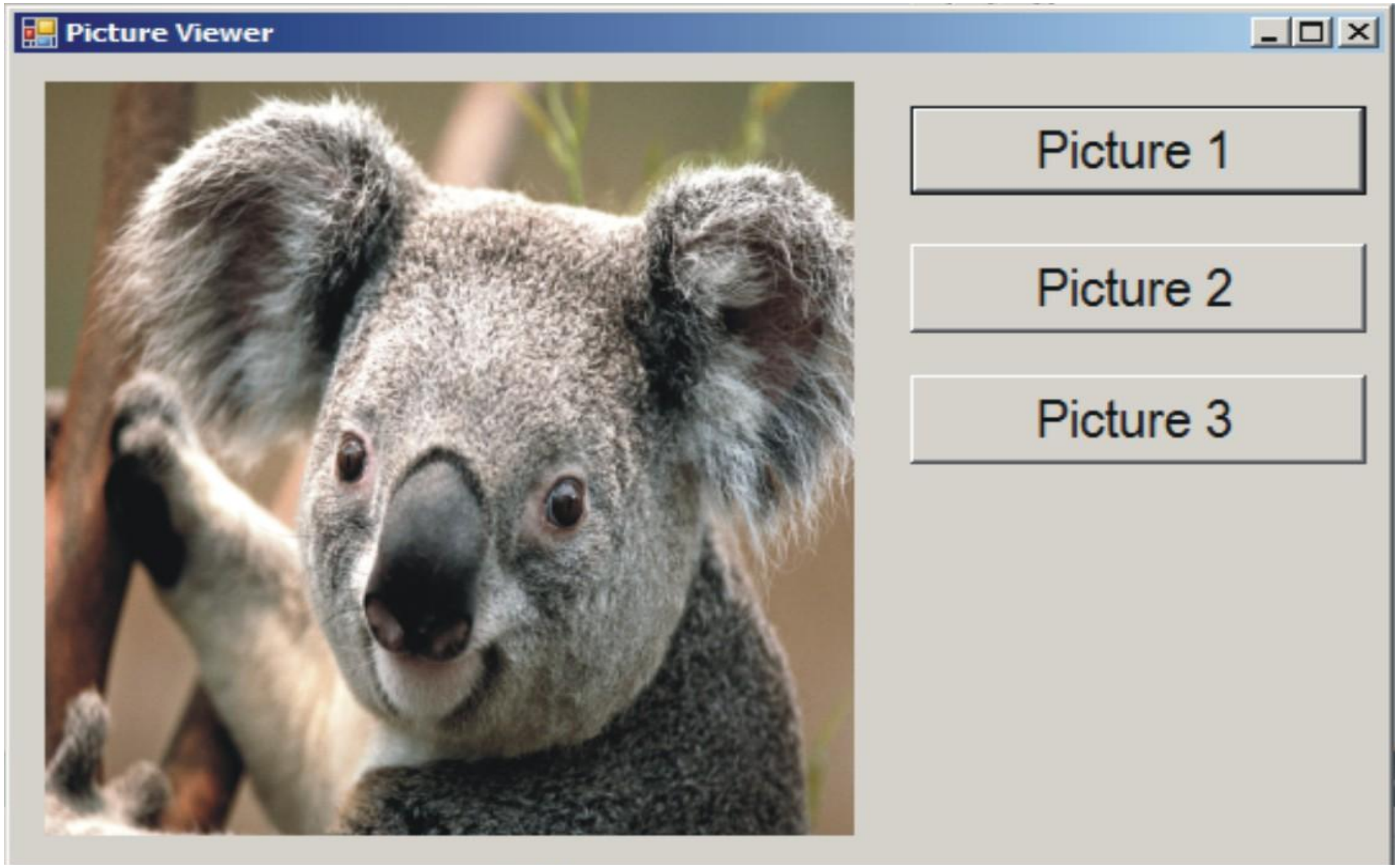
PictureBox :

- The PictureBox control is used to display graphics in bitmap, GIF, JPEG, metafile, or icon format.
- Scroll bars are not supported when the image is larger that the client area, so care must be taken to ensure that the image appears property within the control.
- You can set a picture at run time or design time using Image property.

PictureBox :

Property	Description
BorderStyle	Gets or Sets the style of border to display the control.
Image	Gets or sets the image to display in the picturebox.
SizeMode	Gets or sets the PictureBoxSizeMode enumeration value indicating how the image is displayed. Default is Normal.

Def. Display various pictures when we click on a button.



```
PictureBox.Image = Image.FromFile("K.JPG");
```

Note : Picture must be in the folder of Bin/Debug

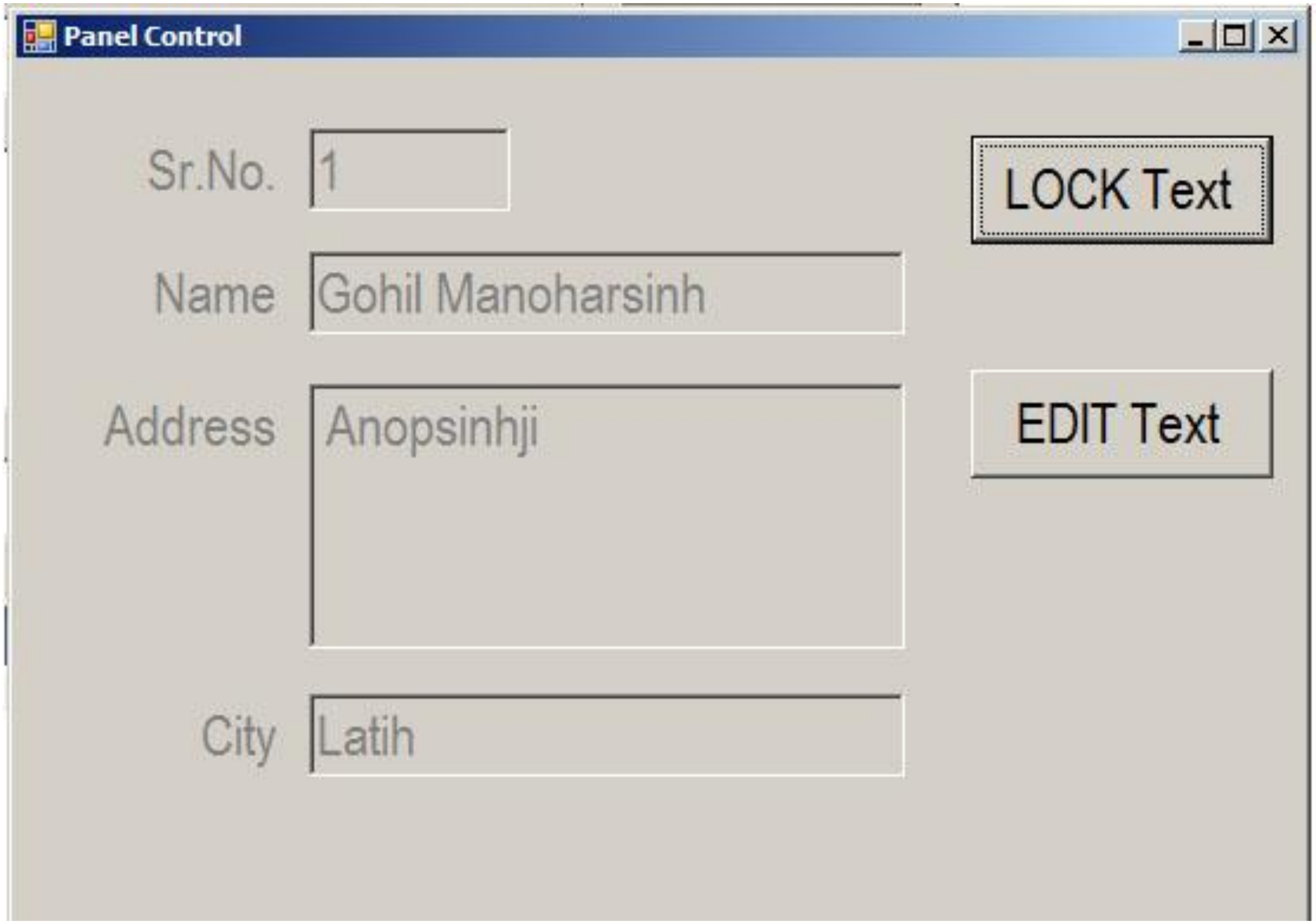
Panel :

- In the .NET Framewok, controls can act as containers for other controls.
- When you drag the container; the controls inside the container will drag with it.
- Two such containers in the System.Windows.Forms namespace are the GroupBox and panel class.

Panel :

Property	Description
BorderStyle	Gets or Sets the style of border to display around the control.
DisplayRectangle	Gets the display are for the control.
Enabled	Gets or sets whether the panel is enabled.
Visible	Gets or sets whether the panel is visible.

Def. Demonstrate use of PANEL control as a container. (Enable)


Panel Control

Sr.No. 1

Name Gohil Manoharsinh

Address Anopsinhji

City Latih

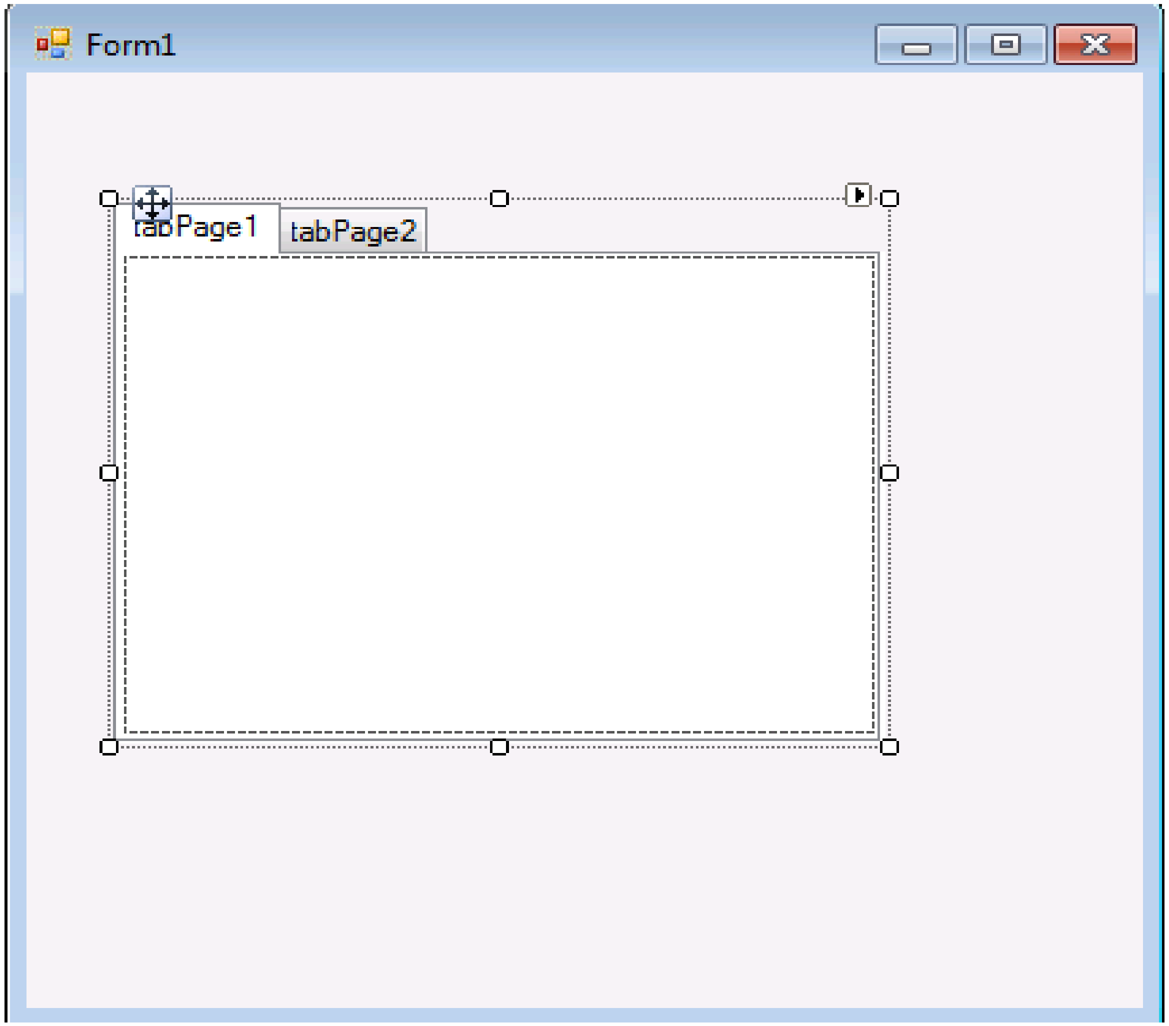
GroupBox

- The GroupBox class inherits directly from the Control class to provide a collection of control objects with no scrolling capabilities.
- A group box includes a simple border, and the Text property for this control displays an optional label as part of the border.
- In general, use a group box control to provide simple containment of controls

TabControl

- Tab controls are used to compact a large amount of data into a single form by segmenting the data into different screens, or tab pages.
- The TabControl displays multiple tabs, like divides in a diary.
- The tabs can contain pictures and other controls.
- You can use the tab control to produce the kind of multiple-page dialog box that appears many places in the Windows operating system.

TabControl



TabControl

Property	Description
Alignment	Gets or sets the are of the control where tabs are displayed, called the tab strip. Defaults to the top of the control.
Appearance	Gets or sets how the tabs are displayed, such as a normal tab or 3D button.
DrawMode	Gets or sets how the tabs are drawn in the control.
HotTrack	Gets or sets whether the tabs change their appearance when the mouse passes over them.

TabControl

Property	Description
ImageList	Gets or sets the list of images to use on the control’s tabs.
ItemSize	Gets or sets the default size of each tab.
Multiline	Gets or sets whether more than one line of tabs can be displayed.
RowCount	Gets the number of rows currently displayed on the control’s tab strip.
SelectedIndex	Get or sets the index of the currently selected tab page.

Def. Design a Form in three parts using tab Control as given…

- Student FORM Details Using TAB Control
  - Personal Details
    - Surname, Name, FatherName, Mother Name, Birth Date, Phone, Mobile, Email Address
  - Education Details
    - 10th Result, 12th Result, Batchelor Result and others
  - Family Details
    - Mother Name, Age, Education, Profession
    - Father Name, Age, Education, Profession
    - Brothers List (List Box), Sisters List (List Box)

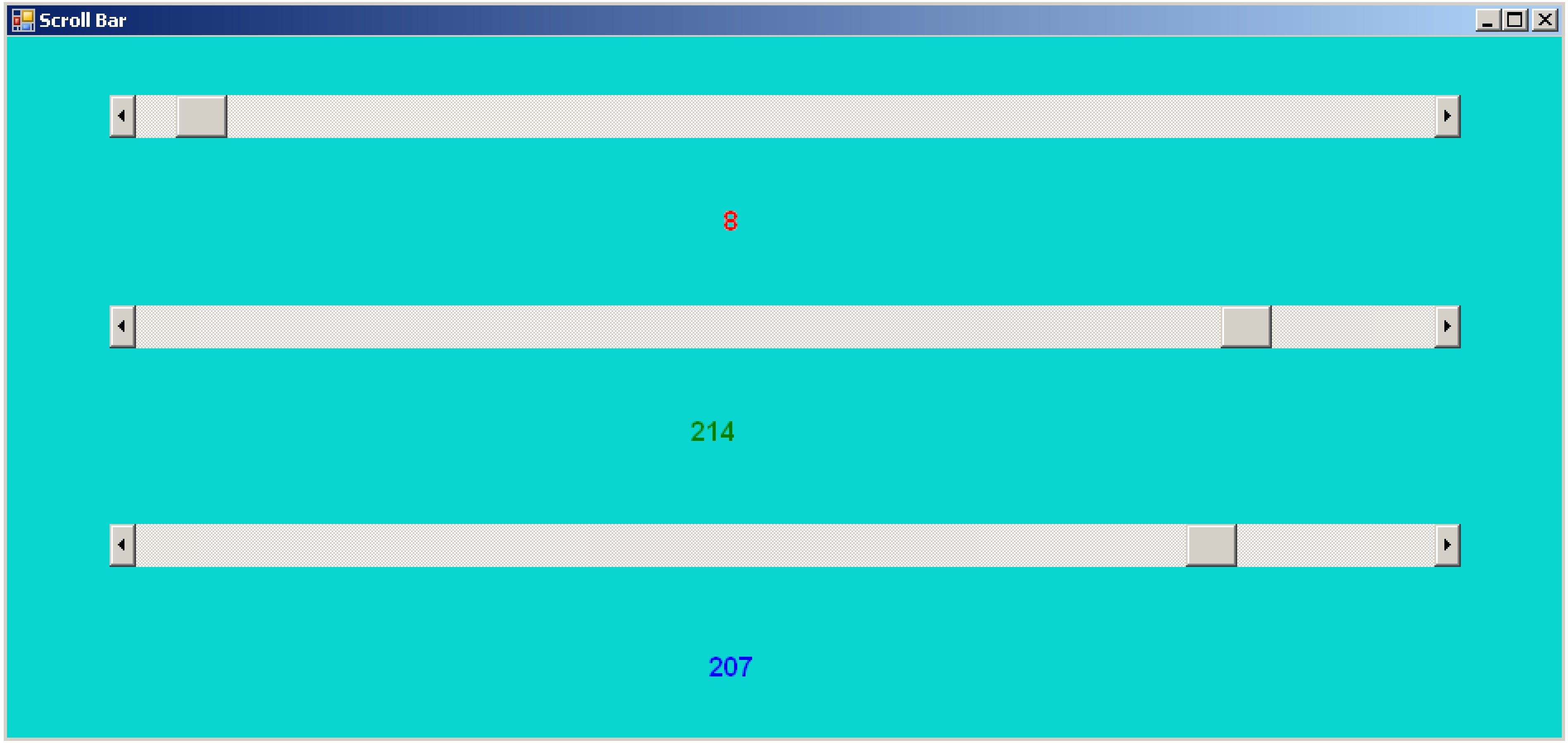
TabePages Class :

- Tab pages are the heart and soul of tab control.
- They define the tabs displayed to the user and the layout of controls that appear when each page is displayed.
- Tab pages are inherited from the Panel class.
- Tab page displays each tab as a simple text string, as specified by the Text property for each page inherited from the Control class.

Scrollbar

- Scrollbar controls are used to provide easy navigation through a long list of items or a large amount of information by scrolling either horizontally or vertically within an application or control.
- Scroll bars are a common element of the Windows interface, so the ScrollBar control is often used with controls that do not derive from the ScrollableControl class.
  - Control :→ **hScrollBar**

Def. Demonstrate use of Scrollbar



```
private void HsbBlue_ValueChanged(object sender, EventArgs e)
{
    this.BackColor = Color.FromArgb(HsbRed.Value, HsbGreen.Value, HsbBlue.Value);
    lblBlue.Text = HsbBlue.Value.ToString();
}
```

Tooltip

- The ToolTip class is a component that provides a small popup window for a control.
- Tool tips provide sort and quick explanations of the purpose of a control or other object.
- A number of classes provide their own tool tip mechanism through a ToolTipText property, in particular the StatusBarPanel, TabPage and ToolBarButton classes.

ToolTip

Property	Description
Active	Gets or sets whether the ToolTip is currently active.
AutomaticDelay	Gets or sets the default delay time in milliseconds.
AutoPopDelay	Gets or sets the time in milliseconds before a displayed tool tip will disappear.
InitialDelay	Gets or sets the time in milliseconds before a tool tip will appear when the mouse is stationary.

ToolTip

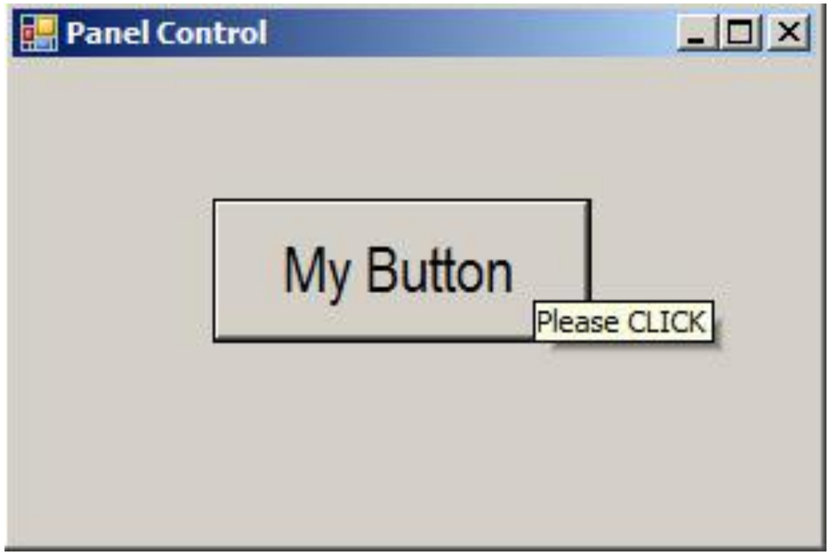
Property	Description
ReshowDelay	Gets or sets the time in milliseconds after the first tool tip is displayed before subsequent tool tips are displayed as the mouse moves from one assigned control to another. The default time is one fifthe(1/5).
ShowAllays	Gets or sets whether to display the tool tip for an inactive control. Default is false.

ToolTip

Public Method	Description
GetToolTip	Retrieves the tool tip string associated with a given control.
RemoveAll	Removes all tool tip strings defined in this component.
SetToolTip	Associates a tool tip string with a given control.

Def. Set tooltip for different Controls.

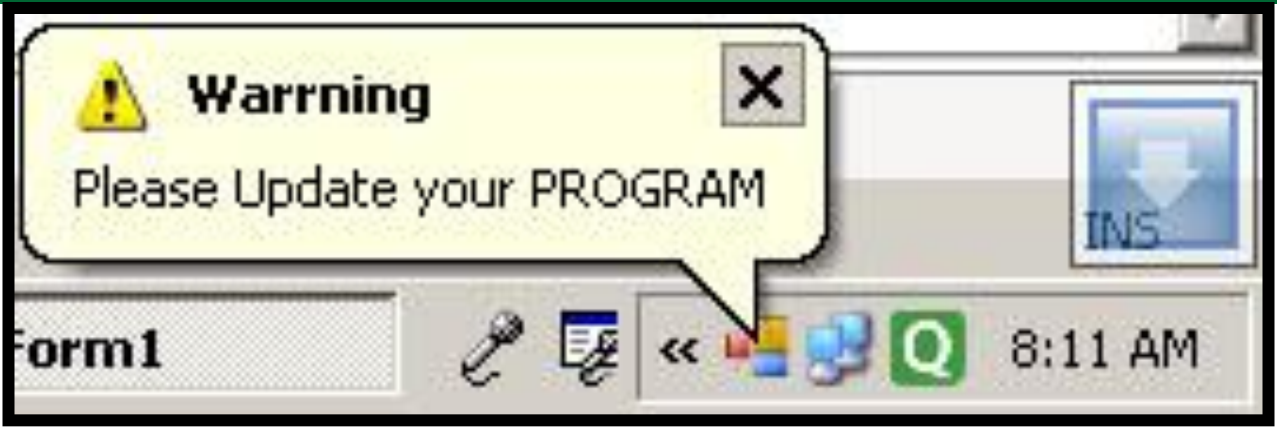
- Insert ToolTip Control
- It will add a new **ToolTip on toolTip1** Property to Selected control.
- Set required tooltip as your need.



NotifyIcon

- Icon in the notification area are shortcuts to processes that are running in the background of a computer, such as a My Computer or Recycle Bin.
- These processes do not come with their own interfaces.
- The NotifyIcon class provides a way to program in this functionality.

Def. Demonstrate use of NotifyIcon



```
public Form1(){
   InitializeComponent();
   this.notifyIcon1.Icon = this.Icon;
   this.notifyIcon1.BalloonTipIcon = ToolTipIcon.Warning;
   this.notifyIcon1.BalloonTipText = "Please Update your PROGRAM";
   this.notifyIcon1.BalloonTipTitle = "Warrning";
   this.notifyIcon1.ShowBalloonTip(12);
}
```

Timer :

- The Timer is a component that raises an event at regular intervals.
- This component is designed for a Windows Form environment.
- This timer object is normally associated and configured within a form.

Timer

Property	Description
Enabled	Gets or Sets whether the time is currently active.
Interval	Gets or sets the time in milliseconds between timer ticks.

Methods

Methods	Description
Start	Starts the timer.
Stop	Stops the timer.

Events

Events	Description
Tick	Occurs when the timer is enabled.

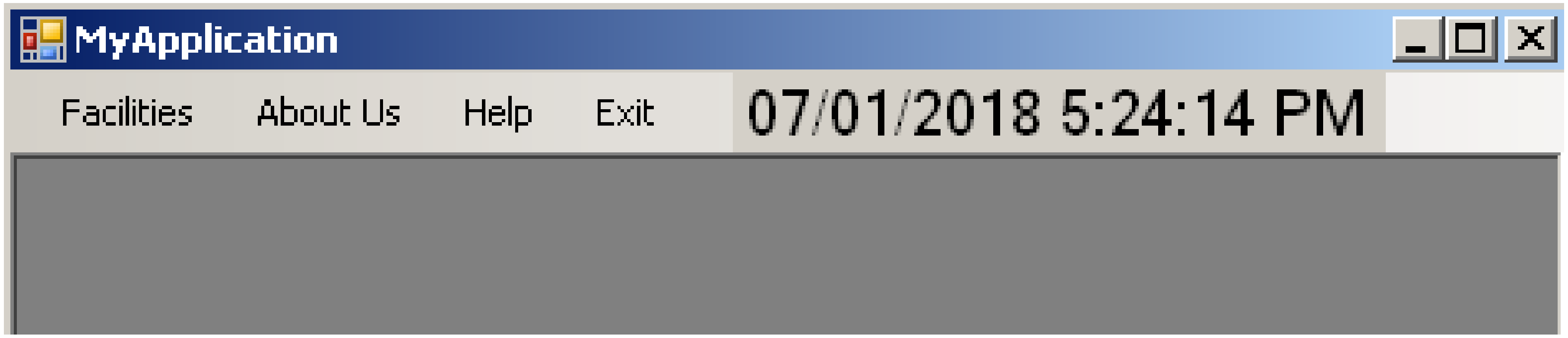
Timer

- Steps to USE Timer :
- Step-1
  - Add Timer Control
- Step-2
  - Set interval
- Step-3
  - Start timer control using any event like button…

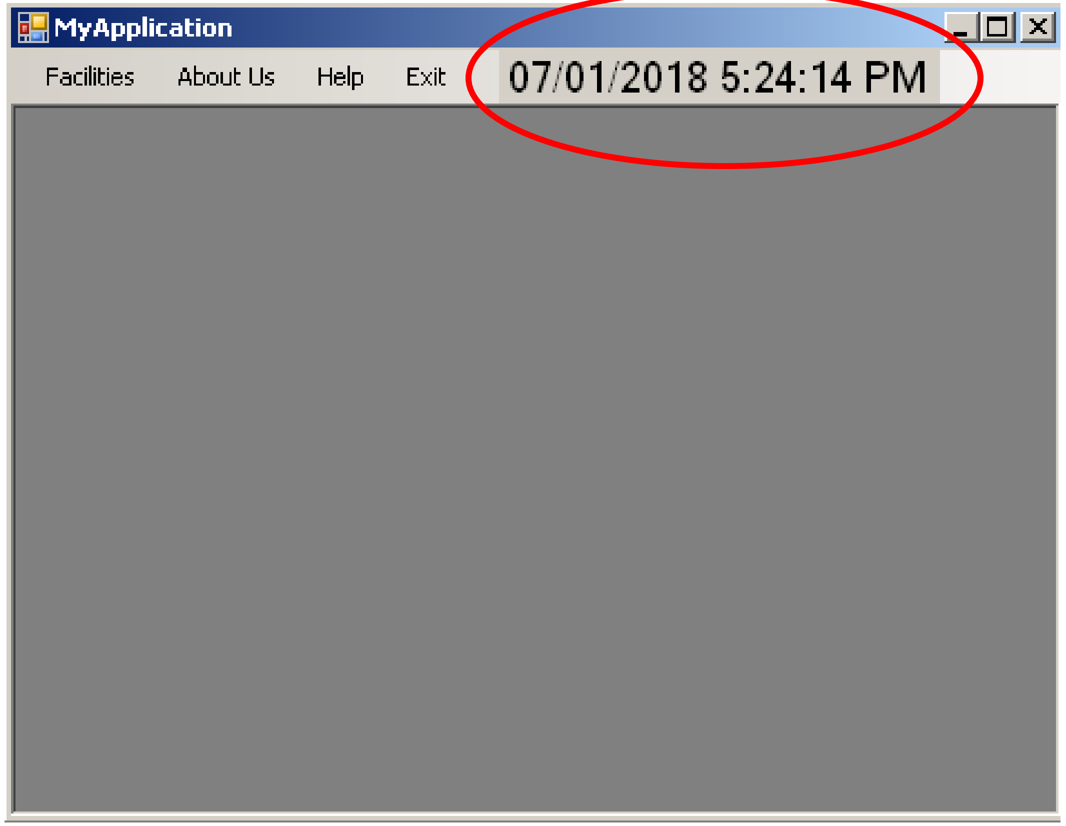
    timer1.Start();

Manus & Toolbar : MenuStrip

- To add a new menu in any form we can use MenuStrip Tool form Toolbox. This box is available in Manus & Toolbar group.
- We can use Manus and SubMenus using this strip.



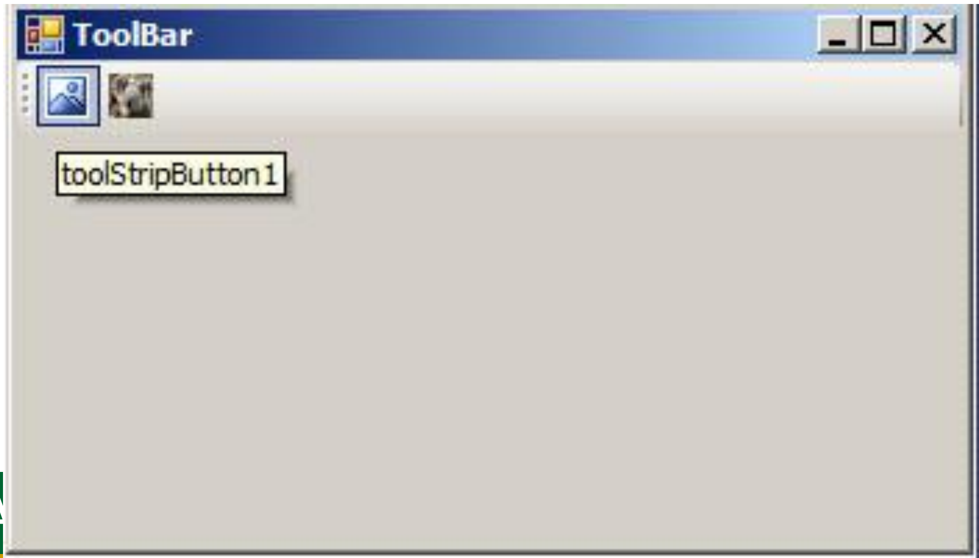
Def. Add current time in MAIN MENU :



```
private void timer1_Tick(  )
{
            lblTime.Text = DateTime.Now.ToString();
}
```

Manus & Toolbar : ToolBar

- To add a new tool in any form we can use ToolStrip Tool form Toolbox. This box is available in Manus & Toolbar group.
- We can use Manus and SubMenus using this strip.
- DEF : Design a toolbar to execute all of your definition from toolbar.



ProgressBar

- A ProgressBar control visually indicates the progress of a lengthy operation in one of three styles:
  - Segmented blocks that increase in steps from left to right.
  - A continuous bar that fills in form left to right.
  - A block that scrolls across a ProgressBar in a marquee fashion.

ProgressBar

Property	Description
Style	Determines the style of ProgressBar that is displayed.
Minimum	To get or set minimum value of range.
Maximum	To get or set maximum value of range.
Step	Gets or sets the amount by which a call to the PerformStep method.
Value	To get or set position within the range of the progress bar.

ProgressBar : Methods

Methods	Description
Increment	Enables you to increment the value of the progress bar by a specific amount.
PerformSetup	Increments the value of the progress bar by the amount specified by the step property.

Step-1

- Add progress bar

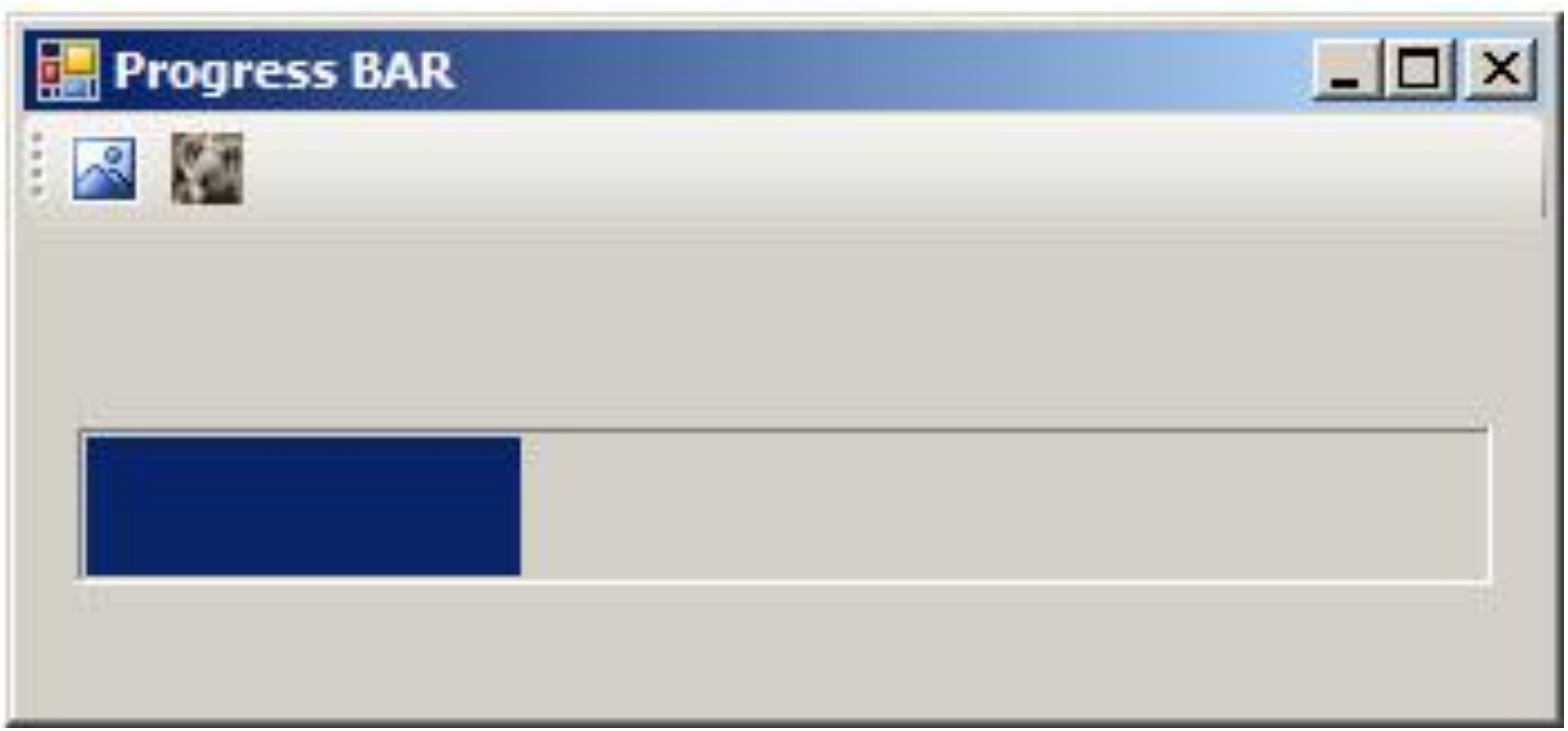
Step-2

- Add timer and start timer.

Step-3

- Add this.progressBar1.Increment(1); code in timer.

Demonstrate Use of ProgressBar



Introduction to MDI

- MDI stands for Multiple-document interface (MDI).
- MDI applications enable you to display multiple document at the same time, with each document displayed in its own window.
- MDI application often have a window menu item with submenus for switching multiple documents interfaces are useful.

Introduction to MDI

- To Creating parent WINDOW
  - Add new form to your application
  - To make this form as the parent form, set its **IsMDIContainer** property to true.
  - Add **Menu Strip** in current form.
    - Add required code in Menu Strip Command

Concept of Inheriting Forms :

- Inheritance is a powerful feature of OOP.
- Form inheritance, a new feature of .NET that lets you create a base form that becomes the basis for creating more advanced forms.
- The new “Derived” form **automatically inherit all the functionality** contained in the base form.
- When the base form is modified, the “derived” classes automatically follow suit and adopt the changes. The same concept applies to any type of object.

Dialog Box (User Defined)

- A dialog box is a form, defined with particular properties.
- Like a form, a dialog box is referred to as a container.
- Like a form, a dialog box is mostly used to host child controls, insuring the role of dialog between the user and the machine.
- A dialog box has following characteristics :
  - There is only close button.
  - It can’t minimized, maximized, or restored
  - It is usually modal, in which case the user is not allowed to continue any other operation on the same application until the doalogbox is dismissed

Dialog Box

- Uses of dialog box are :
  - Display specific information to users.
  - Gather information from users.
  - Both display and gather information.
- How to create and use dialog box
  - ***Ready Designed Dialog ::***
  - Add New AboutBox1.Cs from project menu
    - Add New Item (Ctrl+Shift+A)
      - Select **About Box** (It will create Aboutbox1.Cs file)
    - Now add required coding to code window

Def. Demonstrate use of DialogBox

```
private void button1_Click(object sender, EventArgs e)
{ var info = new Info(); //Info=FormName
   DialogResult result = info.ShowDialog();
   if (result == DialogResult.OK)
    {          MessageBox.Show("Dialog OK");
   }
```

}Note : *We can use NEW FORM as DialogBox too…*

Reusable Dialog Boxes (Common)

- Windows implements a variety of reusable dialog boxes that are common to all applications, including dialog boxes for **opening files, saving files, and printing.**
- Since the dialog boxes are implemented by the operating system, they can be shared among all the applications that run on the operation system, which helps user experience consistency; when users are familiar with the use of an operating system provided dialog box in one application.
- Because these dialog boxes are available to all the applications they are known as common dialog boxes.

Reusable Dialog Boxes (Common)

- Windows Presentation Foundation (WPF) encapsulates the open file, save file, and print-common dialog boxes and exposes them as managed classes for you to use in standalone application.
- Some of the common dialog boxes are as
  - Color dialog box
  - Font dialog box
  - OpenFile dialog box
  - SaveFile dialog box etc.

Reusable Dialog Boxes (Common)

- Color Dialog box
  - To call the color dialog we can use
    - colorDialog.ShowDialog();

Def. : Demonstrate use of Color Dialog Box

- **Note : Add colordialog control from Tools.**

```
private void button1_Click(object sender, EventArgs e)
{
   DialogResult result = colorDialog1.ShowDialog();
   if (result == DialogResult.OK)
   {  label1.BackColor = colorDialog1.Color;    }
}
```

Reusable Dialog Boxes (Common)

- **Font Dialog box**
  - To call the Font dialog we can use
    - FontDialog1.ShowDialog();

Def. : Demonstrate use of Font Dialog Box

- **Note : Add Fontdialog control first.**

```
private void button1_Click(object sender, EventArgs e)
{
   DialogResult result=fontDialog1.ShowDialog();
   if (result == DialogResult.OK)
   {       Font font = fontDialog1.Font;
           label1.Font = font; }
}
```

Reusable Dialog Boxes (Common)

- **OpenFile Dialog box**
  - To call the Open File dialog we can use
    - OpenFileDialog1.ShowDialog();

Def. : USE of OpenFile Dialog (txtFile)

- **Note : Add OpenFileDialog control first.**

```
private void button1_Click(object sender, EventArgs e)
{ DialogResult result = openFileDialog1.ShowDialog();
  if (result == DialogResult.OK)
   {  System.IO.StreamReader sr = new
              System.IO.StreamReader(openFileDialog1
              .FileName);
      textBox1.Text= sr.ReadToEnd();
      sr.Close(); }
}
```

Reusable Dialog Boxes (Common)

- **OpenFile Dialog box**
  - To call the Open File dialog we can use
    - OpenFileDialog1.ShowDialog();

Def. : USE of OpenFile Dialog (RTF File)

- **Note : Add OpenFileDialog control first.**

```
private void button1_Click(object sender, EventArgs e)
{ //Set Default FILE Type
  openFileDialog1.DefaultExt = "*.rtf";
  openFileDialog1.Filter = "RICH Text Files|*.rtf";
  DialogResult result = openFileDialog1.ShowDialog();
  if (result == DialogResult.OK)
  {
    richTextBox1.LoadFile(openFileDialog1.FileName);
  }
}
```

Reusable Dialog Boxes (Common)

- **SaveFile Dialog box**
  - To call the Save File dialog we can use
    - SaveFileDialog1.ShowDialog();

Def. : Demonstrate use of SaveFile Dialog

- **Note : Add SaveFileDialog control first.**

```
private void button1_Click(object sender, EventArgs e)
{ DialogResult result =saveFileDialog1.ShowDialog();
  if(result == DialogResult.OK)
  {   string name = saveFileDialog1.FileName;
      File.WriteAllText(name, textBox1.Text); }
}
```

Required -> using System;

using System.IO; using System.Text;

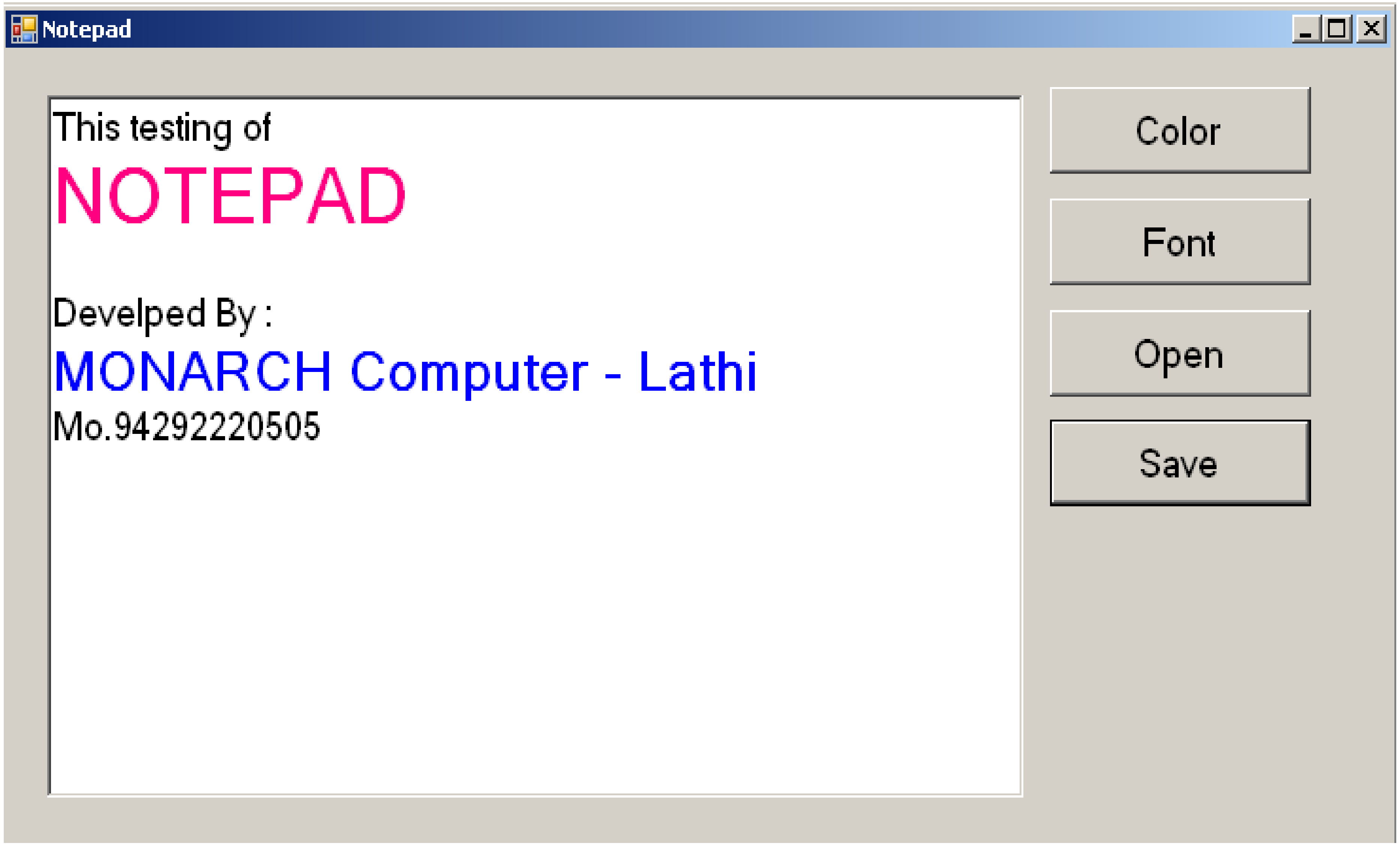
Reusable Dialog Boxes (Common)

- **SaveFile Dialog box with *RICH Text Box***
  - To call the Save File dialog we can use
    - SaveFileDialog1.ShowDialog();

Def. : Demonstrate use of SaveFile Dialog

```
private void button1_Click(object sender, EventArgs e)
{  //Set Default FILE Type
   openFileDialog1.DefaultExt = "*.rtf";
   openFileDialog1.Filter = "RICH Text Files|*.rtf";
   DialogResult result = openFileDialog1.ShowDialog();
   // Initialize the OpenFileDialog to look for RTF files.
   if (result == DialogResult.OK)
   {
      richTextBox1.LoadFile(openFileDialog1.FileName);
   }
}
```

Def. Design a full featured notepad



This testing of

NOTEPAD

Develped By :

MONARCH Computer - Lathi

Mo.94292220505