Unit -3. Introduction to Windows Controls

3.1. Working with Tool Box Controls
- 3.1.1. **Common controls** - Label, Text Box, Button, Check Box, Radio Button, Date Time Picker, List Box, Combo box, Picture Box, Rich Text Box, Tree View, Tool Tip, Progress bar, Masked Text box, Notify Icon, Link Label, Checked List box
- 3.1.2. Container Controls
- 3.1.3. Data - Data Set, Data Grid (Discuss in Unit:5)
- 3.1.4. Component Controls
  Image list, error provider, Help provider, Timer

3.2. Working with Menus and Dialogue Boxes

3.3. Exception Handling
- 3.3.1. Structured Error Handling
- 3.3.2. Unstructured Error Handling

❖ VB.NET Form

- A **Form** is used in VB.NET to create a form-based or window-based application.
- Using the form, we can build a attractive user interface.
- It is like a container for holding different control that allows the user to interact with an application.
- The controls are an object in a form such as buttons, Textboxes, Textarea, labels, etc. to perform some action. However, we can add any control to the runtime by creating an instance of it.
- A Form uses a **System.Windows.Form** namespace, and it has a wide family of controls that add both forms and functions in a Window-based user interface.

• VB.NET Form Properties

- The following are the most important list of properties related to a form. And these properties can be set or read while the application is being executed.

Properties	Description
BackColor	It is used to set the background color for the form.
BackgroundImage	It is used to set the background image of the form.
Cursor	It is used to set the cursor image when it hovers over the form.
Font	It is used to get or set the font used in a form.
Locked	It determines whether the form is locked or not.
FormBorderStyle	It is used to set or get border style in a form.
Text	It is used to set the title for a form window.
IsMDIChild	It is used to authenticate whether a form is a container of a Multiple Document Interface (MDI) child form.
Autoscroll	It allows whether to enable auto-scrolling in a form.
AcceptButton	It is used to set the form button if the enter key is pressed.
Name	It is used to define the name of the form.
Enabled	It uses the True or False value to enable mouse or keyboard events in the form.
TopMost	It uses a Boolean value that represents whether you want to put the window form on top of the other form. By default, it is False.

- ## Form Events

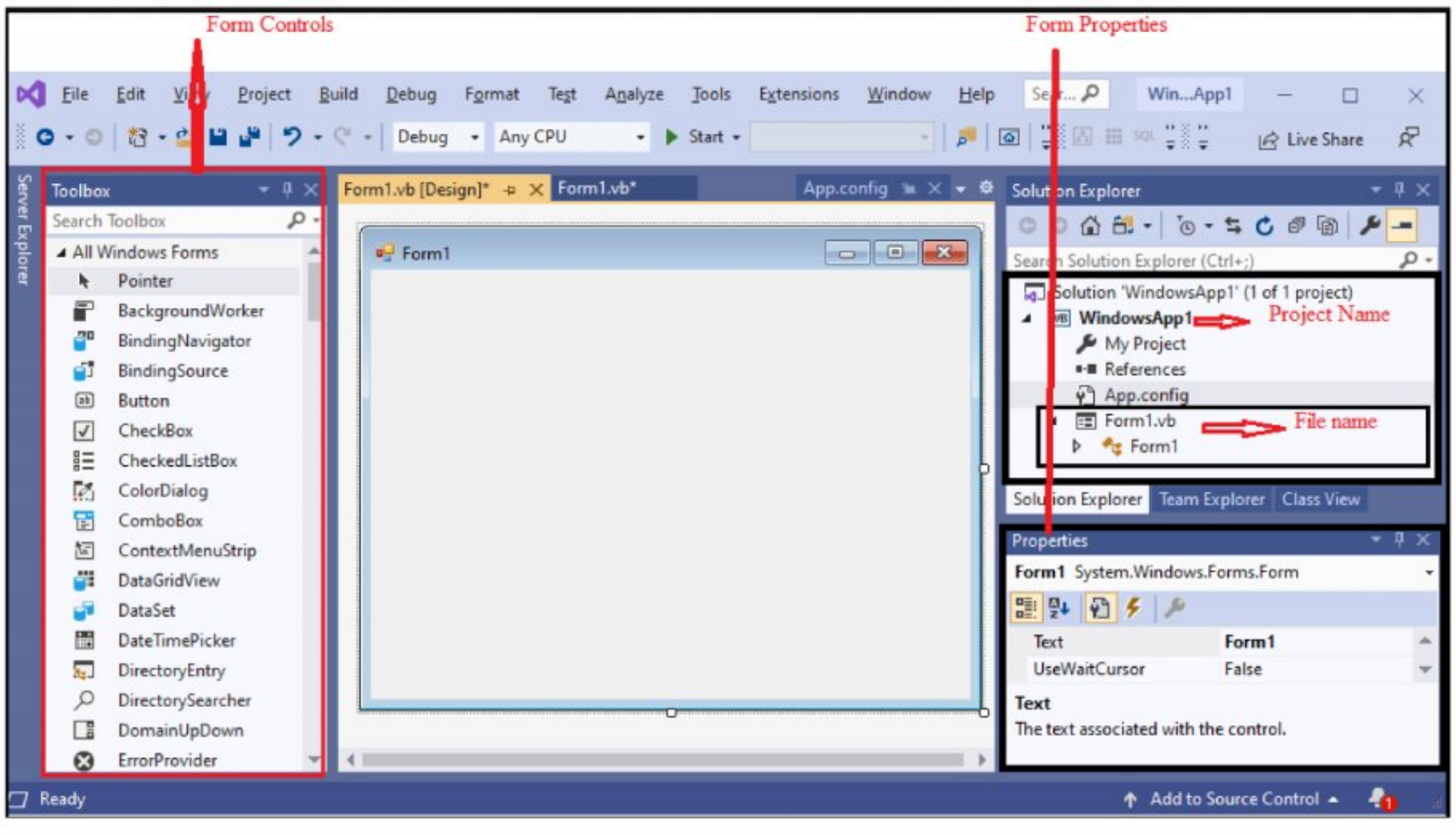
The following are the most important list of events related to a form.

Events	Description
Activated	An activated event is found when the user or program activates the form.
Click	A click event is active when the form is clicked.
Closed	A closed event is found before closing the form.
Closing	It exists when a form is closing.
DoubleClick	DoubleClick The DoubleClick event is activated when a user double clicks on the form.
DragDrop	A DragDrop event is activated when a drag and drop operation is performed.
MouseDown	A MouseDown event is activated when the mouse pointer is on the form, and the mouse button is pressed.
GotFocus	A GotFocus event is activated when the form control receives a focus.
HelpButtonClicked	It is activated when a user clicked on the help button.
KeyDown	A KeyDown event is activated when a key is pressed while focussing on the form.
KeyUp	A KeyUp event is activated when a key is released while focusing on the form.
Load	The load event is used to load a form before it is first displayed.
LostFocus	It is activated when the form loses focus.
MouseEnter	A MouseEnter event is activated when the mouse pointer enters the form.
MouseHover	A MouseHover event is activated when the mouse pointer put on the form.
MouseLeave	A MouseLeave event is activated when the mouse pointer leaves the form surface.
Shown	It is activated whenever the form is displayed for the first time.
Scroll	A Scroll event is activated when a form is scrolled through a user or code.
Resize	A Resize event is activated when a form is resized.
Move	A Move event is activated when a form is moved.

For creating a Windows Forms application in VB.NET, we need to follow the following steps in Microsoft Visual Studio.

1. GoTo File Menu.
2. Click on New Project.
3. Click on Windows Forms App or Application

And finally, click on the 'Create' button to create your project, and then, it displays the following window form with a name Form1.



3.1. Working with Tool Box Controls

- **What is Control?**
  - A control is an object that can be drawn on to the Form.
  - Mainly controls are visible object.
  - The controls are to enable or enhance user Interaction with the application.
  - Examples of these controls are TextBox, Button, Checkbox,Label,Timer, TreeView, etc.
  - All controls are unique with through its features.
  - All the controls have properties, methods, and events.
  - **Control class** is the base class of all the windows controls.
  - We can work with controls in two ways:

1. At design time
2. At runtime.

- We can add controls from the Toolbox at design time as given below:

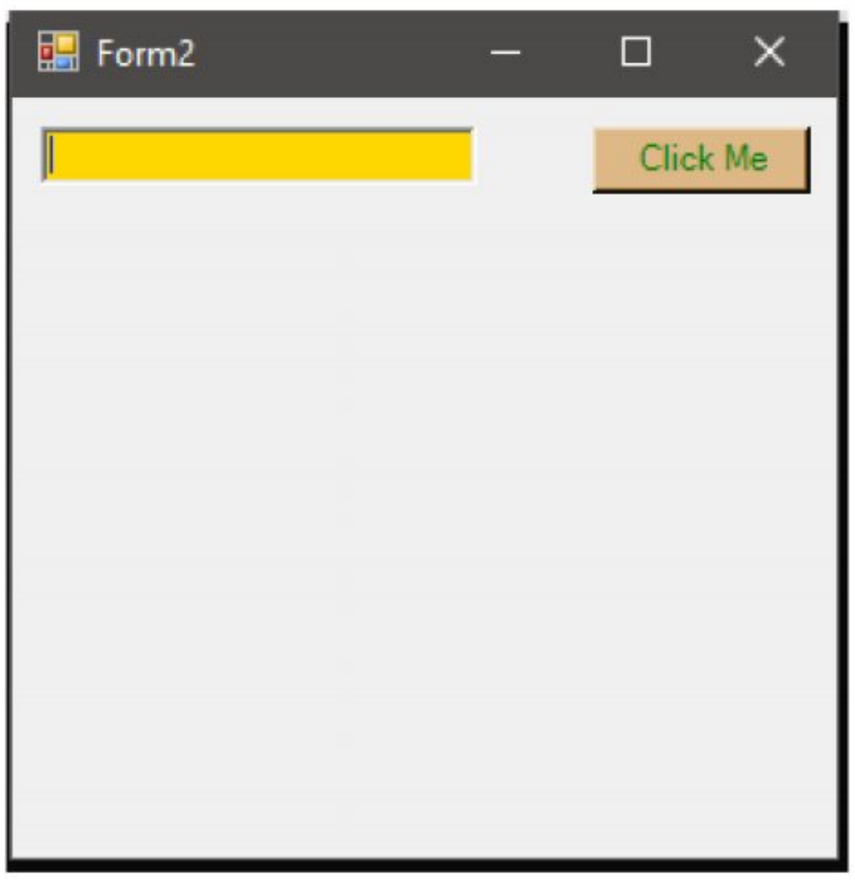
1. Dragging and dropping
2. Double click on the control

- We can set their properties in the properties window
- **We can add controls at runtime as given steps below:**

1. Create an object of Control.
2. Set the properties of it.
3. Add that controls to the form using Me.Controls.Add(Object name)

Example:

```
Private Sub Form2_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    Dim txt As New TextBox
    Dim btn As New Button
    With txt
        .Text = ""
        .ForeColor = Color.Red
        .BackColor = Color.Gold
        .Name = "txtno"
        .Top = 10
        .Left = 10
        .Width = 150
    End With
    With btn
        .Text = "Click Me"
        .ForeColor = Color.Green
        .BackColor = Color.BurlyWood
        .Name = "txtbtn"
        .Top = 10
        .Left = 200
    End With
    Me.Controls.Add(txt)
    Me.Controls.Add(btn)
End Sub
```



- **With.....End With Statement**
  - In VB.NET, the **With End** statement is not the same as a loop structure.
  - It is used to access and execute statements on a specified object without specifying the name of the objects with each statement.
  - Within a **With** statement block, you can specify a member of an object that begins with a period **(.)** to define multiple statements.

Syntax:

```
With objExpression
   [ Statements to be Executed]
End With
```

objExpression: It defines the data type of **objExpression**. It may be any class or structure type or basic
Statement: It defines one or more executed statements within the **With** block
Example:
As Above

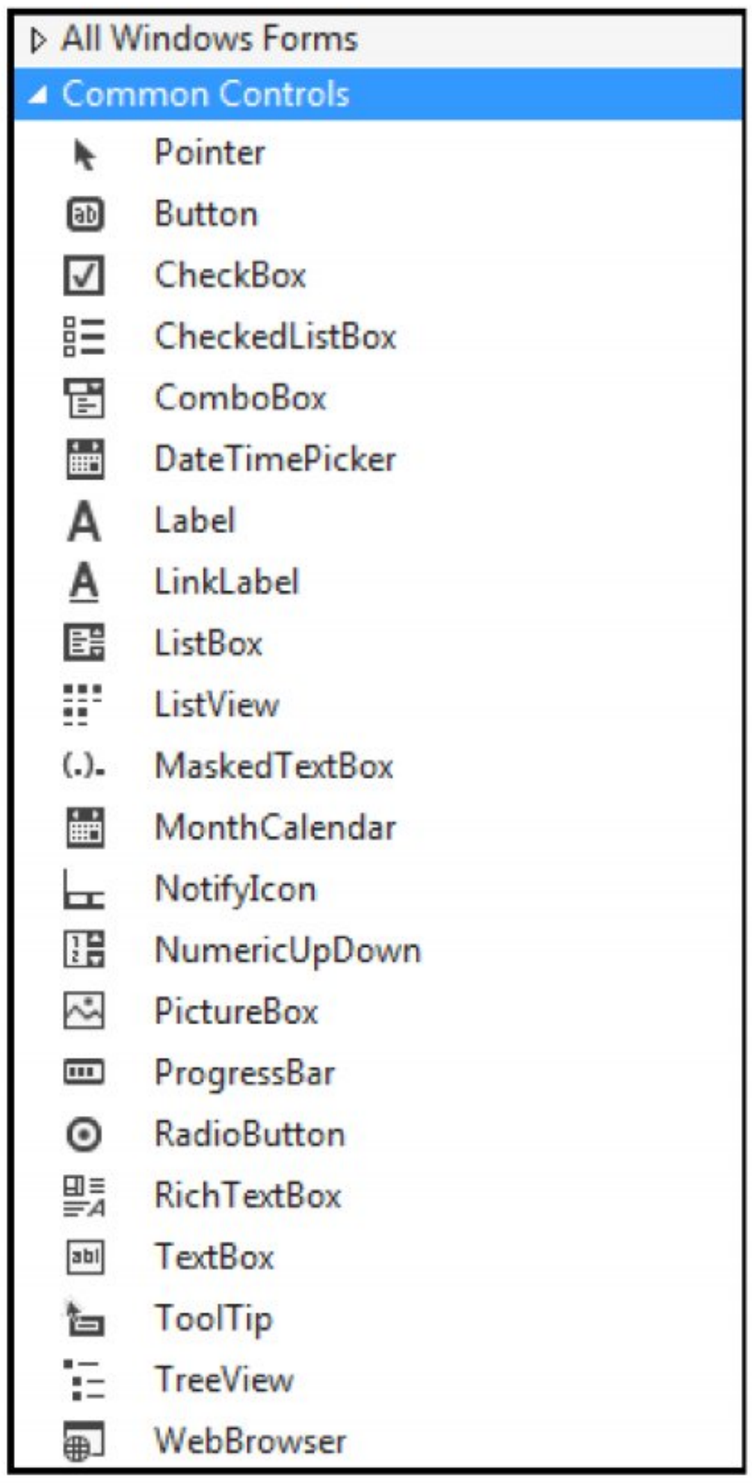
- **Some common properties of the controls are as given below:**

Property	Description
BackColor	Gets/Sets the background color
BackgroundImage	Gets/Sets the background image
ContextMenu	Gets/Sets the shortcut menu for the control
Cursor	Gets/Sets the cursor to be displayed when the user moves the mouse over the form
Enabled	Gets/Sets a value indicating if the control is enabled
Font	Gets/Sets the font for the control
ForeColor	Gets/Sets the font color (foreground) of the control
Height	Gets/Sets the height of the control
Left	Gets/Sets the x-coordinates of a control's left edge in pixels
Location	Gets/Sets the co-ordinates of the upper-left corner of the control
Locked	Gets/Sets the resize or move the control at design time.
Name	Gets/Sets name for the control
Right	Returns the distance between the right edge of the control and the left edge of its container
Size	Gets/Sets size of the control in pixels
TabIndex	Gets/Sets the tab order of this control in its container
TabStop	Gets/Sets a value specifying if the user can Tab or shift+Tab to this control with the tab key
Text	Gets/Sets the text for this control
Top	Gets/Sets the top coordinates of the control
Visible	Gets/Sets a value for visibility of the control
Width	Gets/Sets the width of the control

- **Some common events of the controls are as given below:**

Event	Description
Click	Occurs when the control is clicked.
GotFocus	Occurs when the control receives focus.
KeyDown	Occurs when a key is pressed while the control has focus.
KeyPress	Occurs when a key is pressed while the control has focus.
Key Up	Occurs when a key is released while the control has focus.
LostFocus	Occurs when the control loses focus
MouseClick	Occurs when the control is clicked by the mouse.
MouseDown	Occurs when the mouse pointer is over the control and a mouse button is pressed
MouseEnter	Occurs when the mouse pointer enters the control.
MouseHover	Occurs when the mouse pointer rests on the control.
MouseLeave	Occurs when the mouse pointer leaves the control
MouseMove	Occurs when the mouse pointer is moved over the control.
Mouse Up	Occurs when the mouse pointer is over the control and a mouse button is released

3.1.1 Common Controls



- **Label**
  - In VB.NET, a label control is used to display descriptive text.
  - It does not participate in user input or keyboard or mouse events.
  - Also, we cannot rename label at runtime.
  - The labels are defined in the class System.Windows.Forms namespace.

VB.NET Label Properties

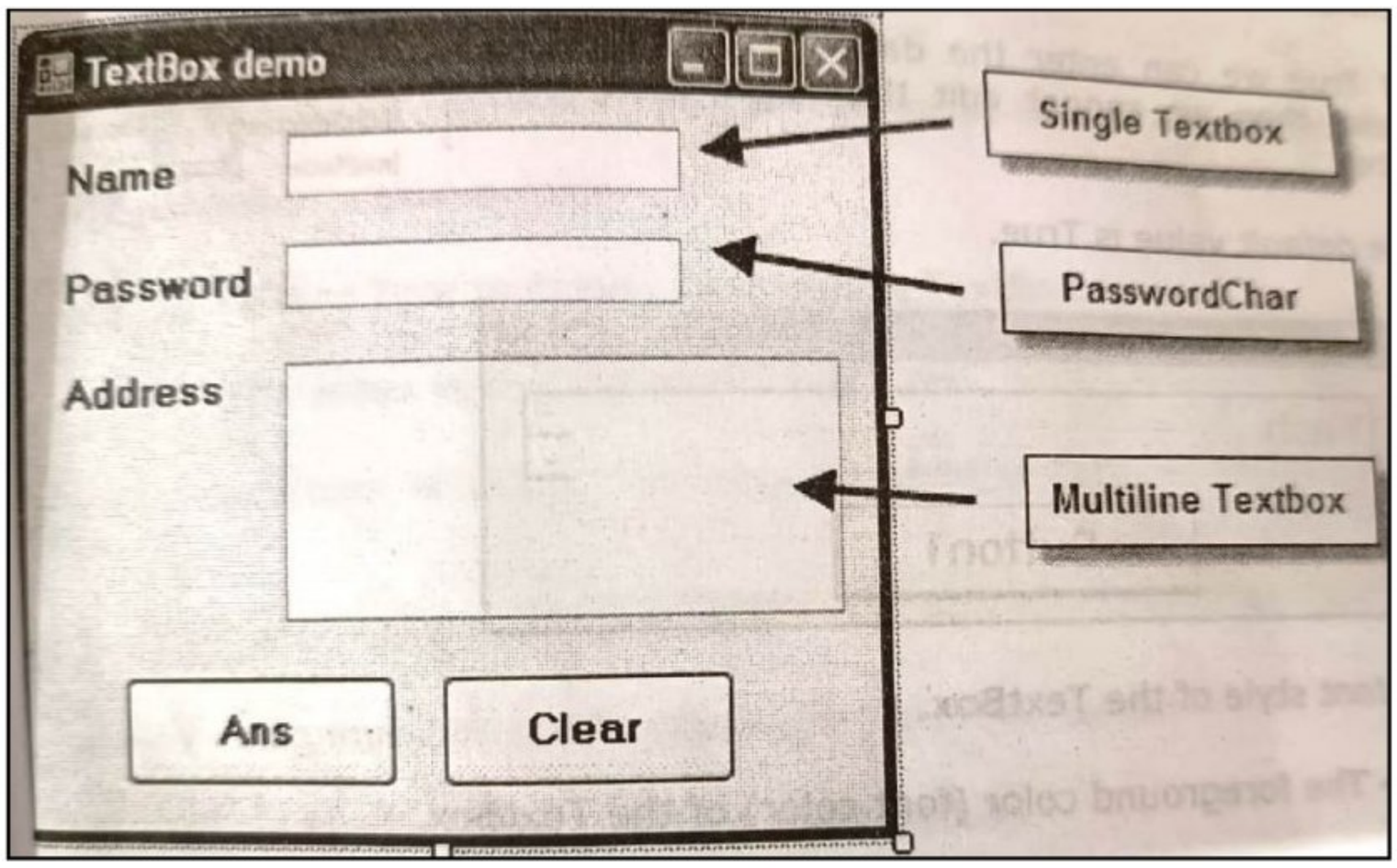
Properties	Description
AutoSize	As the name defines, an AutoSize property of label control is used to set or get a value if it is automatically resized to display all its contents.
Border Style	It is used to set the style of the border in the Windows form.
PreferredWidth	It is used to set or get the preferred width for the Label control.
Font	It is used to get or set the font of the text displayed on a Windows form.
PreferredHeight	It is used to set the height for the Label Control.
TextAlign	It is used to set the alignment of text such as centre, bottom, top, left, or right.
ForeColor	It is used to set the color of the text.
Text	It is used to set the name of a label in the Windows Form.
ContextMenu	It is used to get or sets the shortcut menu associated with the Label control.
DefaultSize	It is used to get the default size of the Label control.
Image	It is used to set the image to a label in Windows Form.
ImageIndex	It is used to set the index value to a label control displayed on the Windows form.

VB.NET Label Events

Events	Description
AutoSizeChanged	An AutoSizeChanged event occurs in the Label control when the value of AutoSize property is changed.
Click	Click event is occurring in the Label Control to perform a click.
DoubleClick	When a user performs a double-clicked in the Label control, the DoubleClick event occurs.
GotFocus	It occurs when the Label Control receives focus on the Window Form.
Leave	The Leave event is found when the input focus leaves the Label Control.
TabIndexChanged	It occurs when the value of Tabindex property is changed in the Label control.
ControlRemoved	When the control is removed from the Control.ControlCollection, a ControlRemoved event, occurs.
TabStopChanged	It occurs when the property of TabStop is changed in the Label Control.
BackColorChanged	A BackColorChanged event occurs in the Label control when the value of the BackColor property is changed.
ControlAdded	When a new control is added to the Control.ControlCollection, a ControlAdded event occurs.
DragDrop	A DragDrop event occurs in the Label control when a drag and drop operation is completed.

❖ Text Box

- TextBoxes are used to accept input from the user or used to display text.
- They are sometimes called an edit field or edit control.
- Typically, a TextBox control is used to display, or accept as input, a single line of text.
- TextBox can display multiple lines.
- TextBox control can work as a Password input control.
- By default, a TextBox holds up to 32767 characters, but when displaying multiple lines, a text box holds up to 2GB of text.
- TextBox can be read-only,
- The prefix of the TextBox is "txt".



Properties	Description
AutoCompleteMode	It is used to get or set a value that indicates how the automatic completion works for the textbox control.
Font	It is used to set the font style of the text displayed on a Windows form.
Lines	It is used to set the number of lines in a TextBox control.
CharacterCasing	It is used to get or set a value representing whether the TextBox control can modify the character's case as they typed.
Multiline	It is used to enter more than one line in a TextBox control, by changing the Multiline property value from False to True.
AcceptsReturn	It is used to get or set a value that indicates whether pressing the enter button in a multiline textbox; it creates a new line of text in control.
PasswordChar	It is used to set the password character that can be a mask in a single line of a TextBox control.
PreferredHeight	It is used to set the preferred height of the textbox control in the window form.
ScrollBars	It is used to display a scrollbar on a multiline textbox by setting a value for a Textbox control.
Text	It is used to get or set the text associated with the textbox control.
Visible	The Visible property sets a value that indicates whether the textbox should be displayed on a Windows Form.
WordWrap	The WordWrap properties validate whether the multiline Textbox control automatically wraps words to the beginning of the next line when necessary.

VB.NET TextBox Events

Events	Description
Click	When a textbox is clicked, a click event is called in the textbox control.
CausesValidationChanged	It occurs in the TextBox Control when the value of CauseValidation property is changed.
AcceptTabsChanged	It is found in the TextBox control when the property value of the AcceptTab is changed.
BackColorChanged	It is found in the TextBox Control when the property value of the BackColor is changed.
BorderStyleChanged	It is found in the TextBox Control when the value of the BorderStyle is changed.
ControlAdded	It is found when the new control is added to the Control.ControlCollection.
CursorChanged	It is found in TextBox, when the textbox control is removed from the Control.ControlCollection.
FontChanged	It occurs when the property of the Font is changed.
GetFocus	It is found in TextBox control to get the focus.
MouseClick	A MouseClick event occurs when the mouse clicks the control.
MultilineChanged	It is found in a textbox control when the value of multiline changes.

❖ Button

- Button control is used to perform a click event in Windows Forms, and it can be clicked by a mouse or by pressing Enter keys.
- It is used to submit all queries of the form by clicking the submit button or transfer control to the next form. However, we can set the buttons on the form by using drag and drop operation.

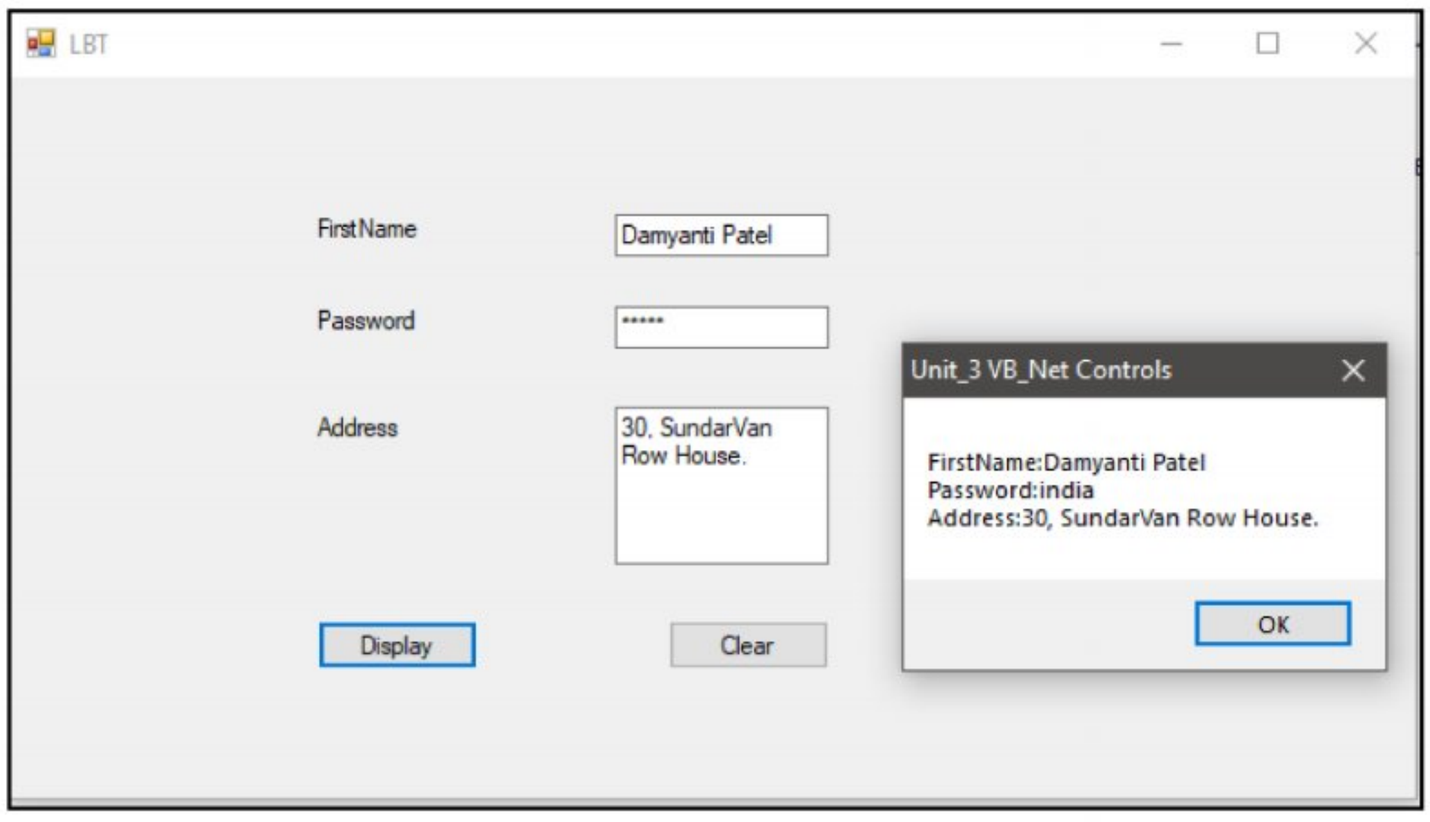
VB.NET Button Properties

Properties	Description
AutoSizeMode	It is used to get or set the auto mode value through which the button can automatically resize in the Windows form.
BackColor	It is used to set the background color of the Button in the Windows form.
BackgroundImage	It is used to set the background image of the button control.
ForeColor	It is used to set or get the foreground color of the button control.
Image	It is used to set or gets the image on the button control that is displayed.
Location	It is used to set the upper-left of the button control's coordinates relative to the upper-left corner in the windows form.
Text	It is used to set the name of the button control in the windows form.
AllowDrop	It is used to set or get a value representing whether the button control can accept data that can be dragged by the user on the form.
TabIndex	It is used to set or get the tab order of the button control within the form.

VB.NET Button Events

BackColorChanged	A BackColorChaged event is found in button control when the Background property is changed.
BackgroundImageChanged	A BackgoundImageChanged event is found in button control when the value of the BackgoundImage property is changed.
Click	A Click event is found in the button control when the control is clicked.
ContextManuChanged	It is found in button control when the value of the ContextMenu property is changed.
ControlAdded	A ControlAdded event is found in button control when a new control is added to the Control.ControlCollection.
CursorChanged	A CursorChanged event is found in button control when the value of the control is changed.
DoubleClick	When the user makes a double click on the button, a double click event is found in the button control.
TextChanged	It is found in the button control when the value of the text property is changed.
DragDrop	The DragDrop event is found in the button control when the drag and drop operation is completed in the Form.

Example:



Code:

Display button click event code:

```
MsgBox("FirstName:" & TextBox1.Text & vbCrLf & "Password:" & TextBox2.Text &
vbCrLf & "Address:" & TextBox3.Text)
```

Clear Button Click Event code:

```
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
```

❖ Check Box

- A checkbox is clicked to select and clicked again to deselect some option.
- When a checkbox is checked a check (a tick mark) appears indicating a selection.
- We can use CheckBox controls in groups to display multiple choices from which the user can select one or more.
- The difference between CheckBox and RadioButton is: Any number of CheckBox controls on a form can be selected and only one RadioButton in a group can be selected.

Here are some properties of the VB.NET CheckBox control.

Property	Description
Default	It is used to get the default size of the checkbox.
AutoCheck	The AutoCheck property is used to check whether the checked value or appearance of control can be automatically changed when the user clicked on the CheckBox control.
CheckAlign	It is used to set the checkmark's alignment, such as horizontal or vertical on the checkbox.
Appearance	The Appearance property is used to display the appearance of a checkbox control by setting a value.
CheckState	The CheckState property is used to verify whether the checkbox status is checked in the window form.
ThreeState	The ThreeState property is used to check whether the control allows one to set three check positions instead of two by setting values.
FlatStyle	It is used to obtain or set the flat appearance of a checkbox.

CheckBox Methods

There are some Methods of the VB.NET CheckBox control.

Method	Description
OnClick	The OnClick method is used to fetch the Click event in the CheckBox control.
OnCheckStateChanged	It is used to call the CheckStateChanged event in the CheckBox control.
ToString	The ToString method is used to return the current string of the CheckBox control.
OnCheckedChanged	When the Checked property is changed in the CheckBox control, the OnCheckedChanged events occur.
OnMouseUp	It is used when it receives the OnMouseUp event in the CheckBox control.

CheckBox Events

There are some Events of the VB.NET CheckBox control.

Event	Description
CheckedChanged	The CheckedChanged event is found when the value of the checked property is changed to CheckBox.
DoubleClick	It occurs when the user performs a double click on the CheckBox control.
CheckStateChanged	It occurs when the value of the CheckState property changes to the CheckBox control.
AppearanceChanged	It occurs when the property of the Appearance is changed to the CheckBox control.

Example:

Code:

```
Dim a As String
Button1.Click()
        If chkc.Checked = True Then
            a = a + chkc.Text + "  "
        End If
        If chkcplus.Checked = True Then
           a = a + chkcplus.Text + "  "
        End If
        If chkjava.Checked = True Then
            a += chkjava.Text + "  "
        End If

        If chkvb.Checked = True Then
            a += chkvb.Text + "  "
        End If

        If chkpython.Checked = True Then
            l += chkpython.Text + "  "
        End If
        Label1.Text = Label1.Text + a
        MsgBox(a)
    End Sub
```

❖ Radio Button

- It is also known as an option button. We can Checked or Unchecked the radiobutton.
- **We can take RadioButton on the form when multiple options are there and selection is only one.**
- Like gender, stream, initial name, etc.
- When the user checks one radio button within a group, the others get unchecked.
- RadioButton can make its own group when kept in container.
- To create multiple groups on one form, place each group in its own container, such as a GroupBox or Panel control.
- Most useful property of the RadioButton is **Checked** - Default value is False, set it to True if you want the RadioButton to be displayed as checked.
- The CheckChanged event is raised when the value of the Checked.

There are following properties of the VB.NET RadioButton control.

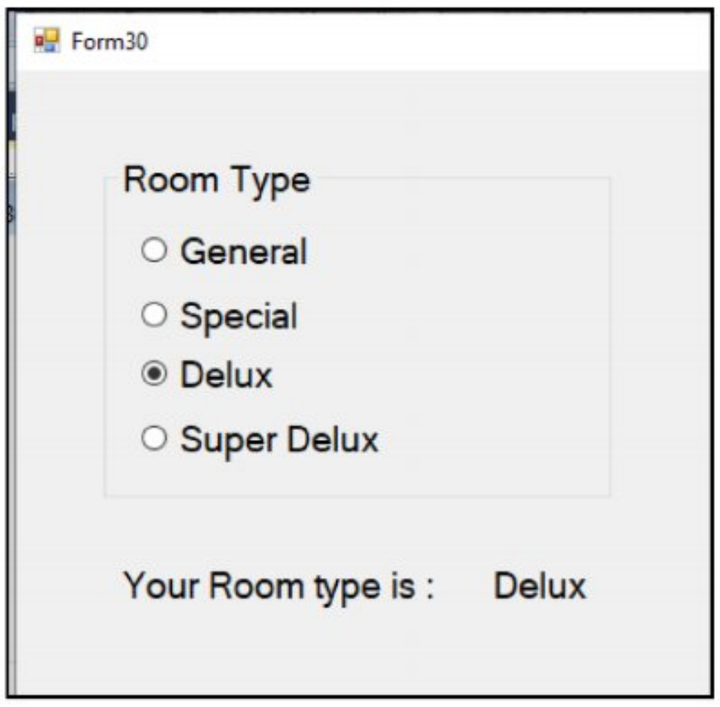
Property	Description
AllowDrop	It is used to set or get a value representing whether the RadioButton allows the user to drag on the form.
Appearance	It is used to get or set a value that represents the appearance of the RadioButton.
AutoScrollOffset	It is used to get or set the radio control in ScrollControlIntoView(Control).
AutoCheck	The AutoCheck property is used to check whether the checked value or appearance of control can be automatically changed when the user clicked on the RadioButton control.
AutoSize	The AutoSize property is used to check whether the radio control can be automatically resized by setting a value in the RadioButton control.
CanSelect	A CanSelect property is used to validate whether a radio control can be selected by setting a value in the RadioButton control.
CheckAlign	It is used to obtain or set a value that indicates the location of the check portion in the radioButton control.
Text	The Text property is used to set the name of the RadioButton control.

RadioButton Methods

Method Name	Description
Contains(Control)	The Contains() method is used to check if the defined control is available in the RadioButton control.
DefWndProc(Message)	It is used to send the specified message to the Window procedure.
DestroHandle()	It is used to destroy the handle associated with the RadioButton Control.
Focus()	The Focus() method is used to set the input focus to the window form's RadioButton control.
GetAutoSizeMode()	It is used to return a value that represents how the control will operate when the AutoSize property is enabled in the RadioButton control of the Window form.
ResetText()	As the name suggests, a ResetText() method is used to reset the property of text to its default value or empty.
Update()	It is used to reroute an invalid field, which causes control in the client region.

Example:

```
RadioButton1.CheckedChanged()
    If RadioButton1.Checked Then
        Label2.Text = RadioButton1.Text
    End If
RadioButton2.CheckedChanged
    If RadioButton2.Checked Then
        Label2.Text = RadioButton2.Text
    End If
RadioButton3.CheckedChanged
    If RadioButton3.Checked Then
        Label2.Text = RadioButton3.Text
    End If
```



❖ Date Time Picker

- It represents a Windows control that allows the user to select a date and a time and to display the date and time with a specified format .
- It makes easy to work with dates and times because it handles a lot of the data validation automatically.

Properties of the DateTimePicker Control

There are some properties of the VB.NET DateTimePicker control.

Property	Description
BackgroundImage	It is used to set the background image for the DateTimePicker control.
CalendarFont	It is used to set the font style for the calendar in the DateTimePicker control.
CustomFormat	The CustomFormat property is used to set the custom date and time format string in the DateTimePicker control.
Controls	It is used to obtain the collection of controls that are stored within the DateTimePicker control.
Checked	A checked property is used to check whether the value property is checked with a valid date and time in the DateTimePicker control.
Format	The Format property of the DateTimePicker is used to set the format for the Date and time displayed in the Windows Form.
MaxDate	The MaxDate property of the DateTimePicker is used to set the max data and time in control selected by the user.
Name	The Name property of the DateTimePicker control allows the user to set the name of the control.
MinimumDateTime	It is used to set the minimum date value that can be allowed by control.

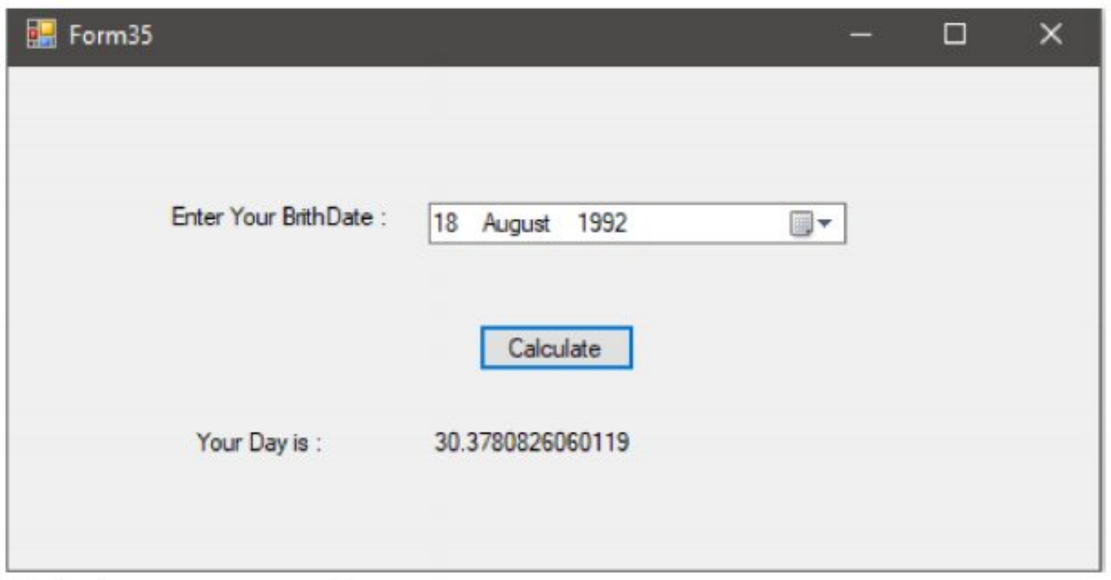
Methods of the DateTimePicker Control

There are some Methods of the VB.NET DateTimePicker control.

Method	Description
Contains(Control)	It is used to validate whether the specified control is a child of the DateTimePicker control or not.
CreateControl()	It is used to force the creation of visible control to handle the creation and any visible child controls.
GetAutoSizeMode()	The GetAutoSizeMode() method is used to check the behavior of the DateTimePicker control when the AutoAize property is enabled.
ResetBackColor()	It is used to reset the back color of the DateTimePicker control.
Select()	The Select() method is used to start or activate the DateTimePicker control.
Show()	The Show() method is used to display the control to the user.
ToString()	The ToString() method is used to return a string that represents the current DateTimePicker control.

Example:

```
Dim a, b As DateTime
Dim c, d As TimeSpan
Button1.Click()
    a = DateTimePicker1.Value
    b = Date.Now
    c = b.Subtract(a)
    Label3.Text = Val(c.TotalDays) / 365
```



❖ List Box

- A ListBox control displays a list of items from which the user can select one or more.
- List boxes are best used for displaying large number of choices.
- CheckBox control occupies space in the form. For example, If we give 50 choices to the user and if we use checkboxes for it then our form will be filled with checkboxes only moreover, it occupies more space. Instead we give 50 choices in the ListBox then it will look better and neat.
- A scroll bar automatically appears when many items in the ListBox.
- By default we can select only single item from the ListBox.

ListBox Properties

Properties Name	Description
AllowSelection	It takes a value that defines whether the list box allows the user to select the item from the list.
ColumnWidth	It is used to get or set the width of the columns in a multicolumn Listbox.
Container	As the name defines, a container gets the IContainer that stores the component of ListBox control.
Controls	It is used to get the collection of controls contained within the control.
Created	It takes a value that determines whether the control is created or not.
Width	It is used to set the width of the ListBox control.
Visible	It takes a value that determines whether the ListBox control and all its child are displayed on the Windows Form.
SelectionMode	It is used to get or set the method that determines which items are selected in the ListBox.
MultiColumn	It allows multiple columns of the item to be displayed by setting the True value in the Listbox.
Items	It allows the collection of items in listbox.

ListBox Methods

Method Name	Description
Add()	The Add() method is used to add items to an item collection.
Remove()	It is used to remove an item from an item collection.We can remove items using the item name.

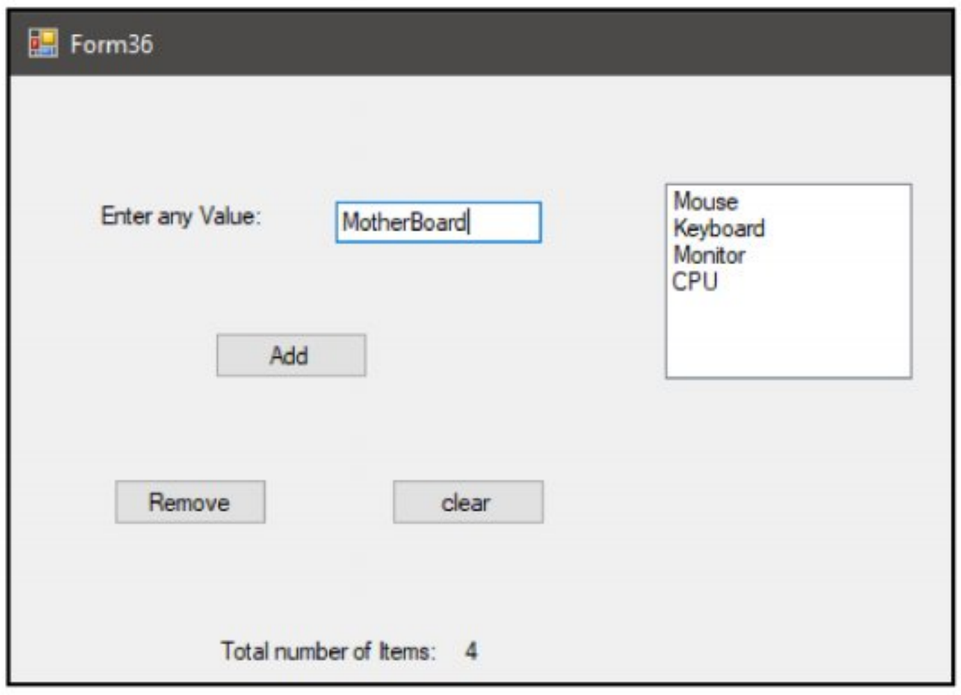
Clear()	It is used to remove all items from the item collection at the same time.
Contains()	It is used to check whether the particular item exists in the ListBox or not.
Show()	It is used to display the control to the user.
Sort()	As the name suggests, a Sort() method is used to arrange or sort the elements in the ListBox.
ResetText()	A ResetText() method is used to reset ListBox's text property and set the default value.
ResetBackColor()	It is used to reset the backColor property of the ListBox and set the default value.

Example:

```
Add_Button.Click()
      ListBox1.Items.Add(TextBox1.Text)
      Label3.Text = ListBox1.Items.Count()
      TextBox1.Text = ""
      TextBox1.Focus()

Remove_Button.Click()
     ListBox1.Items.RemoveAt(ListBox1.SelectedIndex())
     Label3.Text = ListBox1.Items.Count()

Clear_Button.Click()
      ListBox1.Items.Clear()
      Label3.Text = ListBox1.Items.Count()
```



❖ Combo box

- ComboBox is a combination of a TextBox and a ListBox.
- The ComboBox displays an editing field (TextBox) combined with a ListBox allowing us to select from the list or to enter new text.
- Runtime user can enter new data to the combo box control but new data store in it temporary.
- User can select only one item from the given list.
- It takes little space on the form and store long selection data.
- By default, the ComboBox control appears in two parts: Top part + List part
- The top part is a text box that allows the user to type a list item.
- The list part displays a list of items from which the user can select one.
- For example ,use ComboBox for list of Cities, list of Countries, etc.
- The text portion is editable. The list portion is always visible.
- We can add items, remove items and check items, etc are same as ListBox control.

Property	Description
AllowSelection	The AllowSelection property takes the value that indicates whether the list allows selecting the list item.
AutoCompleteMode	It takes a value that represents how automatic completion work for the ComboBox.
Created	It takes a value that determines whether the control is created or not.
DataBinding	It is used to bind the data with a ComboBox Control.
BackColor	The BackColor property is used to set the background color of the combo box control.
DataSource	It is used to get or set the data source for a ComboBox Control.
FlatStyle	It is used to set the style or appearance for the ComboBox Control.
MaxDropDownItems	The MaxDropDownItems property is used in the combo box control to display the maximum number of items by setting a value.
MaxLength	It is used by the user to enter maximum characters in the editable area of the combo box.
SelectedItem	It is used to set or get the selected item in the ComboBox Control.
Sorted	The Sorted property is used to sort all the items in the ComboBox by setting the value.

- **ComboBox Events**

Events	Description
FontChanged	It occurs when the property of the font value is changed.
Format	When the data is bound with a combo box control, a format event is called.
SelectIndexChanged	It occurs when the property value of SelectIndexChanged is changed.
HelpRequested	When the user requests for help in control, the HelpRequested event is called.
Leave	It occurs when the user leaves the focus on the ComboBox Control.

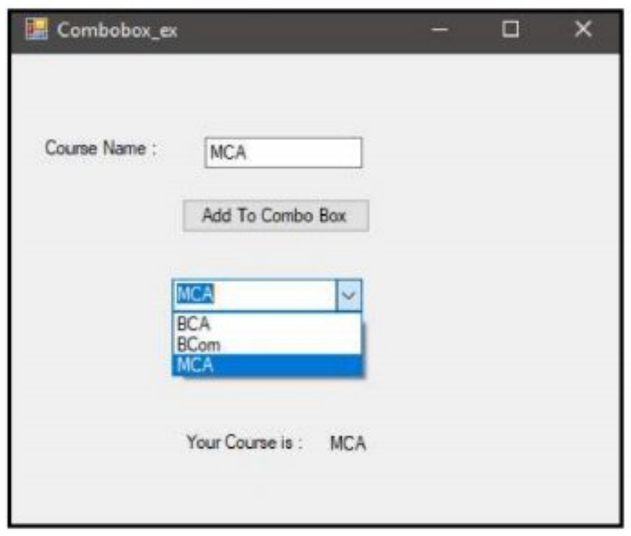
Example:

Button2.Click

```
ComboBox1.Items.Add(TextBox1.Text)
```

ComboBox1.SelectedIndexChanged

```
Label3.Text = ComboBox1.SelectedItem()
```



❖ Picture Box

- **Picture Box** control is used to display the images on Windows Form.
- The Picture Box control has an image property that allows the user to set the image at runtime or design time.

Properties of the Picture Box

There are following properties of the VB.NET Picture Box control.

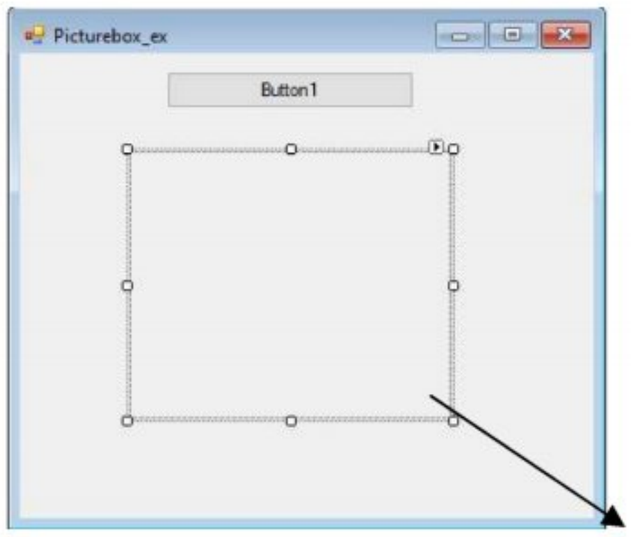
Property	Description
BackColor	It is used to set the background color for the Picture Box in the window form.
BackgroundImage	It is used to set the background image of a window form by setting or getting value in the picture box.
ErrorImage	The ErrorImage property is used to display an image if an error occurs while loading an image on a window form.
InitialImage	The initial image is used to display an image on the Picture Box when the main image is loaded onto a window form by setting a value in the Picture Box control.
SizeMode	It represents how the Picture Box will handle image placement and control sizing.
Text	It is used to set text for the picture box controls in the window form.
Image	The image property is used to display the image on the Picture Box of a Windows form.
BorderStyle	It is used to set the border style for the picture box in the windows form.
ImageLocation	It is used to set or get the path or URL of the image displayed on the picture box of the window form.

Events of the PictureBox Control

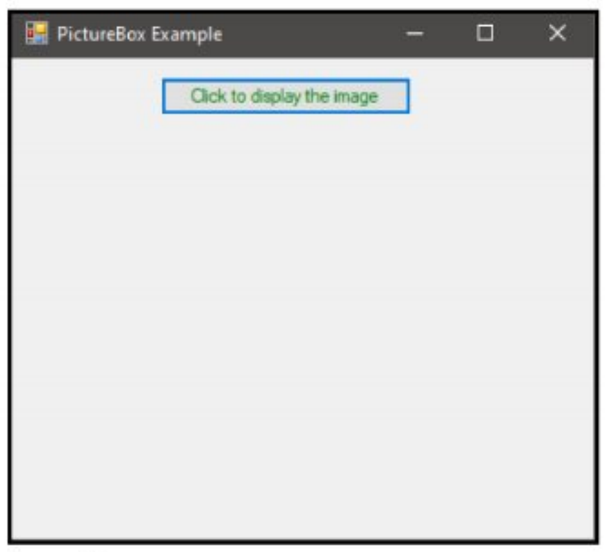
There are some Events of the VB.NET PictureBox control.

Events	Description
BackColorChanged	It occurs when the property of the backcolor is changed in the PictureBox control.
BackgroundImageLayoutChanged	It occurs when the property value of the BackgroundImage is changed in the PictureBox control.
ContextMenuChanged	It occurs when the property of the ContextMenu is changed in the PictureBox control.
Resize	The resize event occurs when the picture box control is changed.

Example:



PictureBox



Form_Load()

```
Me.Text = "PictureBox Example"              'Set the title name for the form
Button1.Text = "Click to display the image"
Button1.ForeColor = Color.Green
```

Button1_Click()

```
PictureBox1.Image = Image.FromFile("F:\VBLOGO.png")
'PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
PictureBox1.Height = 250
PictureBox1.Width = 400
Button1.Visible = False
```

❖ Rich Text Box

- It's an advance version of TextBox.
- Difference between TextBox and RichTextBox is like Notepad and WordPad,
- It provides more advanced text formatting option.
- It can load RTF,TXT format files for reading or editing.
- The wordpad save it's file in RTF format. The full form of RTF is Rich Text Format.
- RichTextBox allows formatting the text, say adding colors, displaying particular font types, selected text effect, bullets, alignment, indents and so on.
- By default RichTextBox is multiline. By default maxlength of it 2147483647.
- Some properties of RichTextBox are BulletIndent, Selection Font, Selection Color, Alignment, etc.
- Best example of RichTextBox is WordPad.

Properties:

Properties	Description
AutoSize	Property gets or sets the value specifying to change the rich text box automatically as the font changes.
BackColor	Property used to Get or set background color for the control.
AutoWordSelection	Property used to set or get a value specifying automatic word selection.
CanRedo	Property to used to indicate that any actions can be reapplied.
CanUndo	Property to used to undo any previous actions.
HideSelection	Property used to set or get value specify a text should stay highlighted when control loses focus.
MaxLength	Property used to set or get the maximum number of a line a user can type into a rich text box.
Multiline	Property used to set or get a value specifying multiline input for the control.
ScrollBars	Property used to set or get the kind of scroll bar to be used.
SelectedText	Property used to set or get the selected text within the control.
SelectionColor	Property used to set or get color for the selected text.
SelectionLength	Property used to set or get the number of characters selected in the control.
SelectionFont	Property used to set or get the font for the selected text.
Text	Property used to set or get the current text in the control.

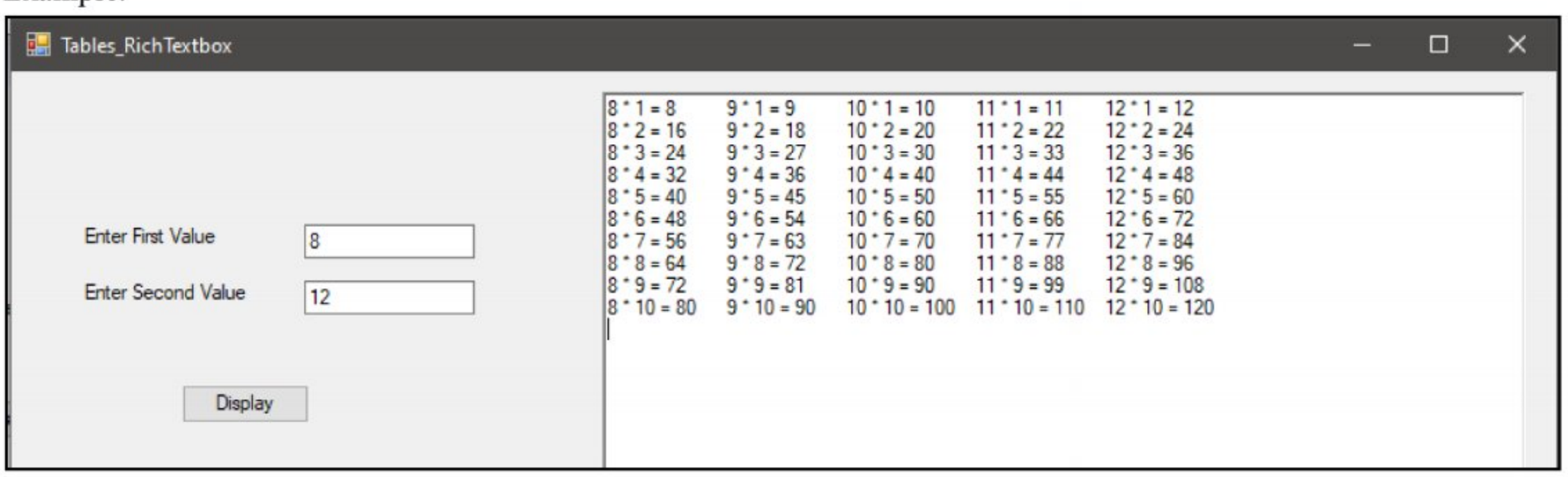
Methods:

Method	Description
Appends	Method used to append text to current text of the control.
CanPaste	Method determines if the infromation can be pasted from a clipboard.
Clear	Method used to clear text from the control.
Find	Method used to search a text inside the control.
GetLineFromCharIndex	Method used to get the line number from the specified character position.
GetPositionFromCharIndex	Method used to get the location within the control at the specified character index.
LoadFile	Methods used to load the contents of a file into the control.
Redo	Method to reapply the last operation.
Select	Methods used to select the text within the control.
Undo	Method to undo the last edit operation.

Events:

Events	Description
Click	Triggered when the control is clicked.
LinkClicked	Triggered when the user clicks on the link within the text.
ModifiedChanged	Triggered when the value of the Modified property is changed.
ReadOnlyChanged	Triggered when the value of the ReadOnly property is changed.
SelectionChanged	Triggered when the value of the Selection property is changed.
VScroll	Triggered when the vertical scroll bars are clicked.

Example:



Code:

```
Dim a, b As Integer
Dim str As String

Button_Click()
 a = Val(TextBox1.Text)
 b = Val(TextBox2.Text)
 For i = 1 To 10
   For j = a To b
 RichTextBox1.AppendText(String.Format("{0} * {1} = {2}", j, i, (j * i).ToString().PadRight(10)))
   Next
      RichTextBox1.AppendText(vbCrLf)
 Next
```

❖ Tree View

- The TreeView control is used to display a hierarchical representation of the same data in a tree structure.
- Windows Explorer and Solution Explorer contain TreeView control.
- TreeView control is collection of nodes. 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 a child.
- We can expand and collapse these nodes by clicking on the plus sign (+) button.
- We can display TreeView control with check boxes next to the nodes, if the TreeView's CheckBoxes property is set to true.
- It is also useful to provide the full path of the root node to the child node.

Properties of the TreeView Control

There are following properties of the TreeView control.

Properties	Description
Nodes	The nodes property of the tree view control is used to gather all the nodes used in the tree.
SelectedNode	It is used to obtain or set the tree node that is selected in the tree view control.
ShowRootLines	It gets or sets a value that represents whether you want to draw lines between the tree nodes connected with the root of the tree view.
Path Separator	The Path Separator property of the Tree View Control is used to set a delimiter string between the tree node paths.
ShowPlusMinus	It is used to get or set a value representing whether you want to display the plus (+) or minus sign button next to tree nodes containing the child nodes.
ShowLines	It takes a value representing whether you want to draw lines between the tree nodes of the tree view control.
TopNode	It is used to get or set full visible tree nodes on top of other nodes in the tree view control.
VisibleCount	It is used to obtain the fully visible tree node in the tree view control.
ItemHeight	The ItemHeight property is used to set the height of each tree node in control.
Scrollable	The Scrollable property is used in the tree-view to display the scroll bar by setting the value in control.

Methods of the TreeView Control

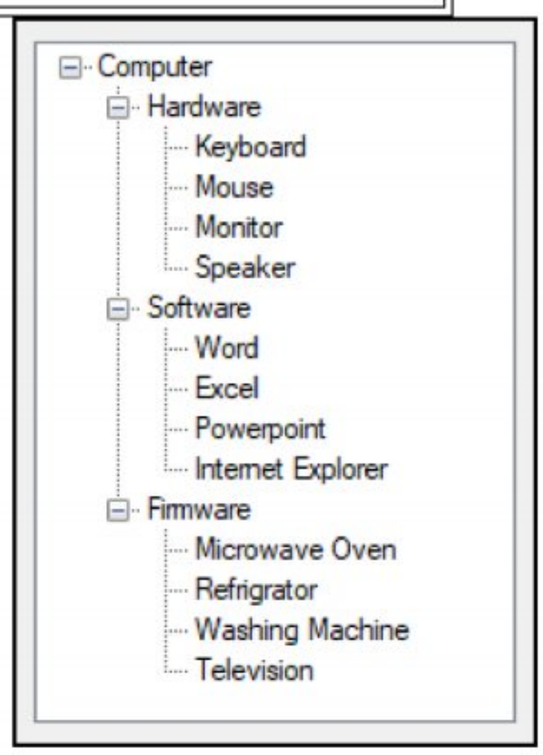
Method	Description
GetNodeAt	A GetNodeAt() method is used to get a node at the specified location of the tree view control.
Sort()	A Sort method is used to sort the tree nodes that are available in the tree view control.
ExpandAll()	As the name suggests, an ExpandAll method is used to expand all the tree nodes.
GetNodeCount	It is used to count the number of nodes that are available in the tree view control.
CollapseAll	It is used to collapse all tree nodes, including all child nodes in the tree view control.

Example:

Load()

```
TreeView2.Nodes.Add("Computer")
TreeView2.Nodes(0).Nodes.Add("Hardware")
TreeView2.Nodes(0).Nodes(0).Nodes.Add("Mouse")
TreeView2.Nodes(0).Nodes(0).Nodes.Add("Keyboard")
TreeView2.Nodes(0).Nodes(0).Nodes.Add("Monitor")
TreeView2.Nodes(0).Nodes(0).Nodes.Add("Speaker")

TreeView2.Nodes(0).Nodes.Add("Software")
TreeView2.Nodes(0).Nodes(1).Nodes.Add("Word")
TreeView2.Nodes(0).Nodes(1).Nodes.Add("Excel")
TreeView2.Nodes(0).Nodes(1).Nodes.Add("PowerPoint")
TreeView2.Nodes(0).Nodes(1).Nodes.Add("Internet Explorer")
```



```
TreeView2.Nodes(0).Nodes.Add("Firmware")
TreeView2.Nodes(0).Nodes(2).Nodes.Add("Microwave Oven")
TreeView2.Nodes(0).Nodes(2).Nodes.Add("Washing Machine")
TreeView2.Nodes(0).Nodes(2).Nodes.Add("Refrigrator")
TreeView2.Nodes(0).Nodes(2).Nodes.Add("Television")
```

❖ Tool Tip

- It represents a small rectangular pop-up window that displays a brief description of a control.
- It is used to provide hints to a user when the user places the pointer on a control.
- It is typically used to alert users to the intended use of a control.

Properties:

Properties	Description
BackColor	Property used to Get or set background color for the control.
AutomaticDelay	Property used to set or get the automatic delay for the tool tip.
AutoPopUpDelay	Property used to set a delay for automatic pop up of the control.
InitialDelay	Property used to set an initial delay for the control.
ToolTipIcon	Property used to choose the type of tool tip icon.
ShowAlways	Property used to get or set whether the tool tip should be appear when its parent control is inactive.

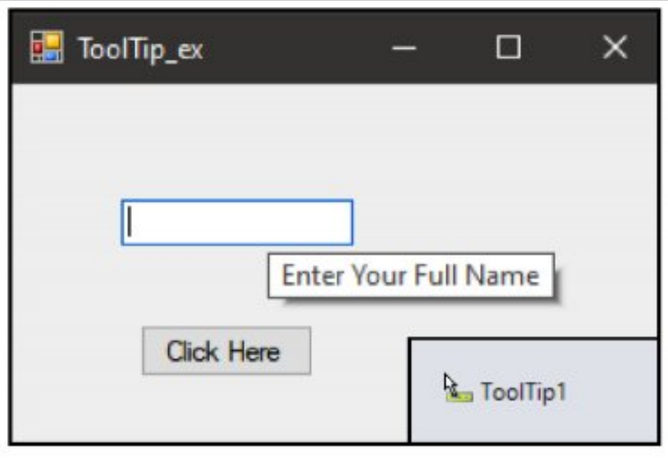
Events:

Events	Description
GetToolTip	Returns a tool tip text.
SetToolTip	Connects tool tip text with the tool tip.

Example :

```
TextBox1_MouseHover ()
    ToolTip1.SetToolTip(TextBox1, "Enter Your Full Name")

Button_click()
    MsgBox("Hello " & TextBox1.Text)
```



❖ Progress Bar

- The Window Progress Bar control is used by the user to acknowledge the progress status of some defined tasks, such as downloading a large file from the web, copying files, installing software, calculating complex results, and more.

Properties of the ProgressBar Control

There are following properties of the VB.NET ProgressBar control.

Property	Description
MarqueeAnimationSpeed	It is used to determine the progress status for a progress bar in milliseconds.
Padding	The padding property is used to create a space between the edges of the progress bar by setting the value in the progress bar control.
Step	It is used to get or set a value in control that calls the PerformStep method to increase the current state of the progress bar by adding a defined step.
Maximum	It is used to set the maximum length of the progress bar control in the windows form.
Minimum	It is used to set or get the minimum value of the progress bar control in the windows form.
AllowDrop	It obtains a value representing whether the progress bar control enables the user to be dragged onto the form.
Style	It is used to set a value that represents how types the progress bar should be displayed on the Windows Form.

Methods of the ProgressBar Control

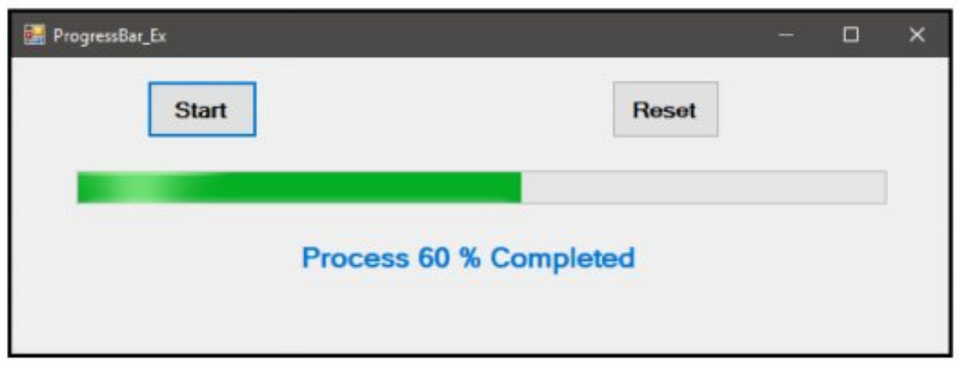
Event	Description
ForeColor	The ForeColor method is used to reset the forecolor to its default value.
ToString	The ToString method is used to display the progress bar control by returning the string.
Increment	It is used to increase the current state of the progress bar control by defining the specified time.
PerformStep	It is used to increase the progress bar by setting the step specified in the ProgressBar property.

Events of the ProgressBar Control

Events	Description
Leave	The Leave event occurs when the focus leaves the progress bar control.
MouseClick	A MouseClick event occurred when the user clicked on the progress bar control by the mouse.
TextChanged	It occurs when the property of the text is changed in the progress bar control.
PaddingChanged	It occurs when the padding property is changed in the progress bar control.

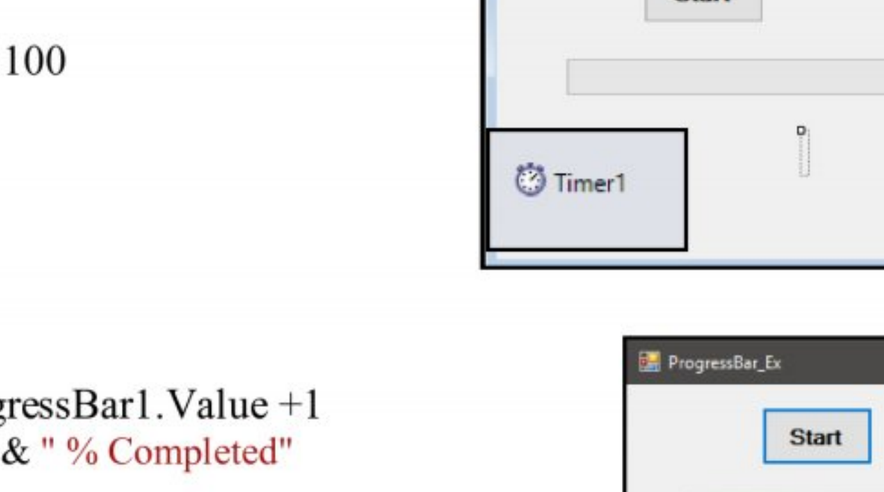
Example:

```
Start_Button.Click()
    Timer1.Enabled = True
    ProgressBar1.Maximum = 100
    a = 1
Reset_Button.Click()
    a = 1
    ProgressBar1.Value = 0
    Label1.Text = ""
Timer1_Tick()
    ProgressBar1.Value = ProgressBar1.Value +1
    Label1.Text = "Process " & a & " % Completed"
        a += 1
        If a > 100 Then
            Timer1.Enabled = False
        End If
```



❖ Masked Text box

- The .NET MaskedTextBox control provides a mechanism to validate user input on a Form.
- For example, if you need a TextBox that should accept dates in a certain format, you should set the masking in the MaskedTextBox.
- The MaskedTextBox uses the **MaskedTextProvider** mask syntax.
- The mask is used to distinguish between proper and improper user input.



Properties:

Properties	Description
TextAlign	Text alignment is set using this property.
Mask	This property is used to set the predefined mask description.
HidePromptOnLeave	Property used to hide the mask characters, literals when the focus is lost from the control.
AllowPromptAsInput	Property used to set the input character as per the specified by moving character.
TextMaskFormat	Property used to specify the formatting for the text.
RejectInputOnFirstFailure	Property used to gets or sets a value indicating whether to stop user input after the first invalid character is entered.

❖ Notify Icon

- It specifies a component that creates an icon in the notification area.
- Icons in the notification area are shortcuts of an application that are running in the background of a computer, such as a virus protection program or a volume control.
- These processes do not come with their own user interfaces.

Properties:

Properties	Description
ContextMenu	Property gets or sets the context menu for the tray icon.
Icon	Property used to Get or sets the current icon.
Text	Property used to Get or set the tooltip text when mouse hovers the icon.
Visible	Property to used to make the icon visible or invisible.

Methods:

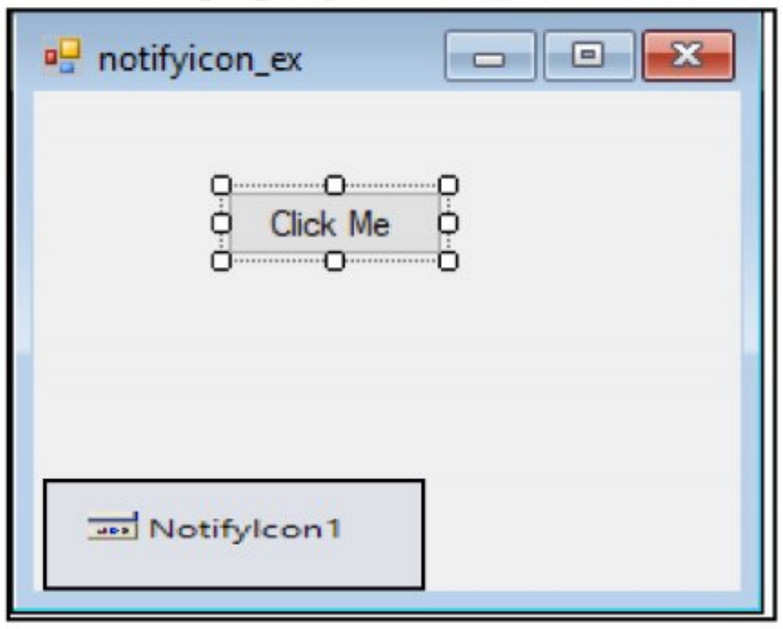
Method	Description
Dispose	Method used to release all the resources used by the component.
Finalize	Method used to release all unmanaged resources, do cleanup operation before garbage collection.
GetService	Method used to return the object that represents the service provided by that component.
GetType	Method used to get the type of current Instance.
ShowBalloonTips	Method used to display a balloon tip for the specified time period.

Events:

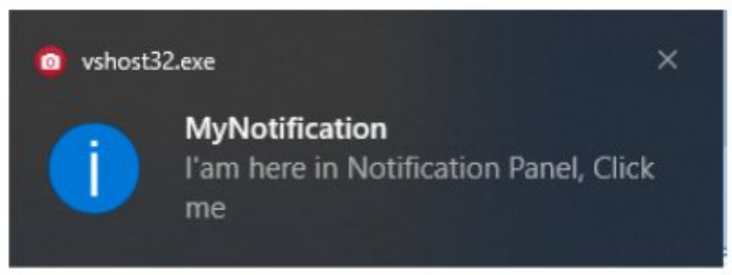
Events	Description
Click	Triggered when the icon the in the system tray is clicked.
DoubleClick	Triggered when the icon the in the system tray is double clicked.
MouseDown	Triggered when the user presses the mouse button on the icon.
MouseMove	Triggered when the user moves the mouse over the icon.
MouseUp	Triggered when the user releases the mouse button on the icon.

Example:

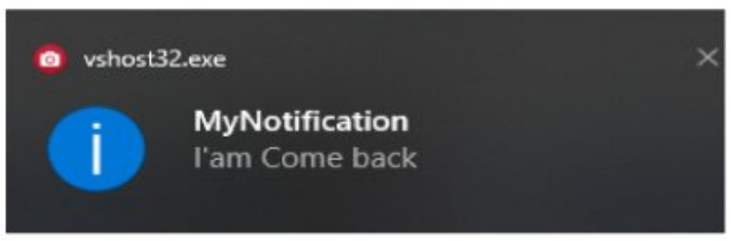
- Place one button and NotifyIcon in the form as follow:
- Set **Icon** property of NotifyIcon control and change **Text** property of Button control.



After Click on Button at Runtime see at Notification Panel:



After Click on Icon from Notification Panel:



Button1_Click()

```
Me.Hide()
NotifyIcon_APP.ShowBalloonTip(30, "CLICK ME", "I' am here in Notification Panel, Click me..", ToolTipIcon.Info)
```

NotifyIcon_APP_MouseClick()

```
Me.Show()
NotifyIcon_APP.ShowBalloonTip(30, "CLICK ME", "Hello... I am Back.", ToolTipIcon.Info)
```

❖ Link Label

- It is similar to a Label control, but it can display a hyperlink .
- Multiple hyperlinks can be specified in the text of the control.
- Each hyperlink can perform a different task within an application.
- Each hyperlink displayed in the LinkLabel control is an instance of **the LinkLabel.Link class**.
- The LinkLabel.Link class defines display information, state, and location of the hyperlink.

Properties:

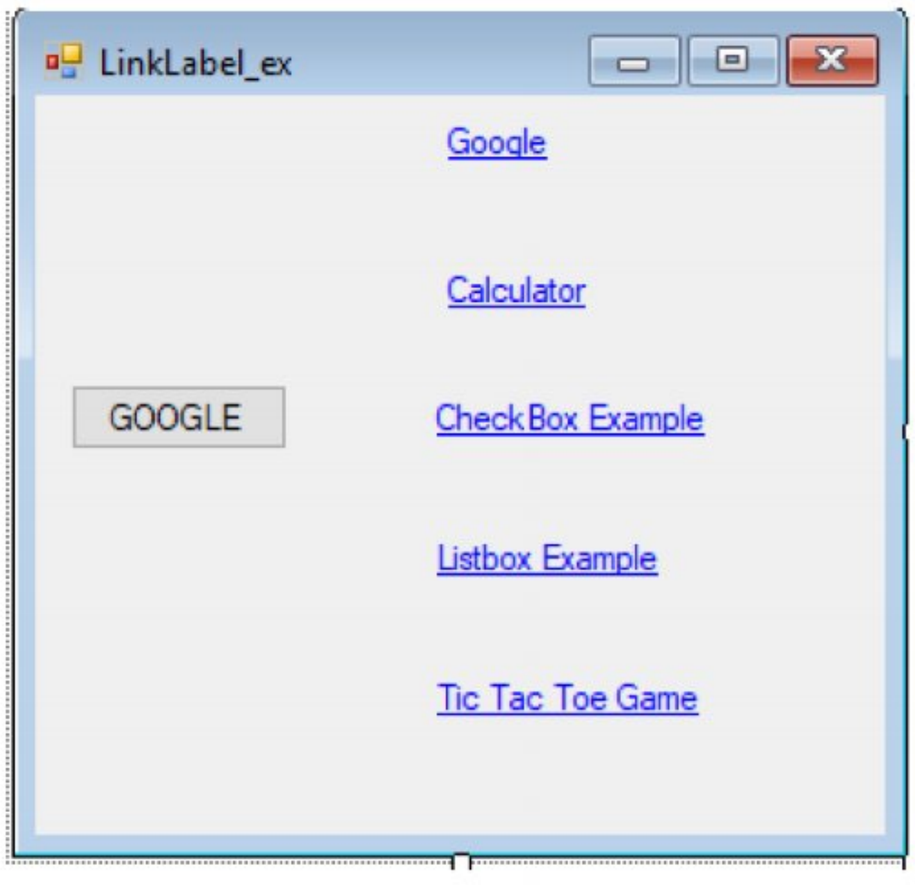
Properties	**Description**
TextAlign	Text alignment is set using this property either to right or left by giving values 0 or 1
Text	Property used for the label text.
Height	This property is used to specify height of the control.
Width	This property is used to specify the width of the control.
BackStyle	Property used to make a label transparent or opaque.
Autosize	Property is used to adjust the label text horizontally.
WordWrap	Property is used to adjust the label vertically.
ActiveLinkColor	Property used to specify the color of the active link.
DisabledLinkColor	Property used to specify the color of the disabled link.
LinkColor	Property is used to specify the color for a normal link.
VisitedLinkColor	Property is used to specify the color for a visited link.

Events:

Events	**Description**
LinkClicked	Triggered when the link is clicked.

Example:

```
Button1_Click()
    Process.Start("www.google.com")
LinkLabel1_LinkClicked()
    Process.Start("www.google.com")
LinkLabel2.LinkClicked()
    Calculator.Show()
LinkLabel5.LinkClicked()
    checkbox_ex.Show()
LinkLabel4.LinkClicked()
    Listbox_ex1.Show()
LinkLabel3.LinkClicked()
    Tic_Tac_Toe_Game.Show()
```



❖ Checked List box

- The **CheckedListBox** is similar to **Listbox** except that it displays all items in the list with a checkbox that allows users to check or uncheck single or multiple items.

CheckedListBox Properties

Properties	Description
AccessibilityObject	It obtains a value that determines whether the AccessibilityObject is assigned to the CheckedListBox control.
AccessibleName	It gets or sets a value that tells if the accessible client application has used the name of the checkedlistbox control.
AllowSelection	It gets a value that indicates whether the ListBox allows for the item to be selected from the list.
AllowScollOffset	It gets or sets a value representing whether the CheckedListBox control is scrolled in ScrollControlIntoView(Control).
BorderStyle	It is used to set the type of border around the CheckedListBox by getting or setting a value.
CheckedItems	It is used to store a collection of checked items in the CheckedListBox.
ScrollAlwaysVisible	It is used to set or get a value that indicates if the vertical scroll bar appears in Windows Forms at all times.
SelectedItems	It is used to get all selected items from CheckedListBox.
SelectionMode	It is used to get or set a value representing the items' selection mode in the CheckedListBox.
TopIndex	It is used to set the first visible item at the top of the index in the CheckedListBox.

CheckedListBox Methods

Methods	Description
ClearSelected()	It is used to unselect all the selected items in the CheckedListBox.
CreateAccessibilityInstance()	It is used to create new accessibility of the object in the CheckedListBox Control.
CreateItemCollection()	It is used to create a new instance for the collected item in the list box.
DestroyHandle()	It is used to destroy the handle associated with CheckedListBox.
Equals(Object)	It is used to validate whether the specified object is equal to the current object in the CheckedListBox or not.
FindForm()	It is used to obtain the form in which CheckedListBox has control.
GetItemText(Object)	It is used to get the text of the specified item in the CheckedListBox.
GetType()	It is used to get the current item type in the CheckedListBox.
Show()	A Show() method is used to display the CheckedListBox Control to the user.
Sort()	A Sort() method is used to sort or organize all the items available in CheckedListBox.

Example:

Add_Button_Click()

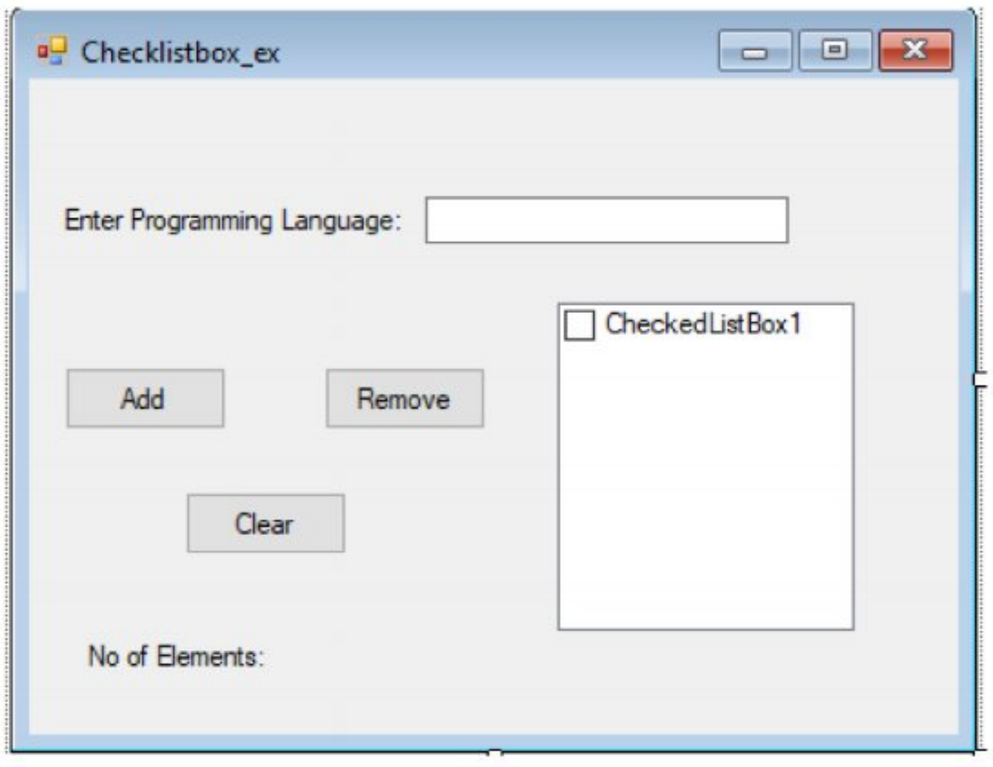
```
CheckedListBox1.Items.Add(TextBox1.Text)
Label3.Text = CheckedListBox1.Items.Count()
```

Remove_Button_Click()

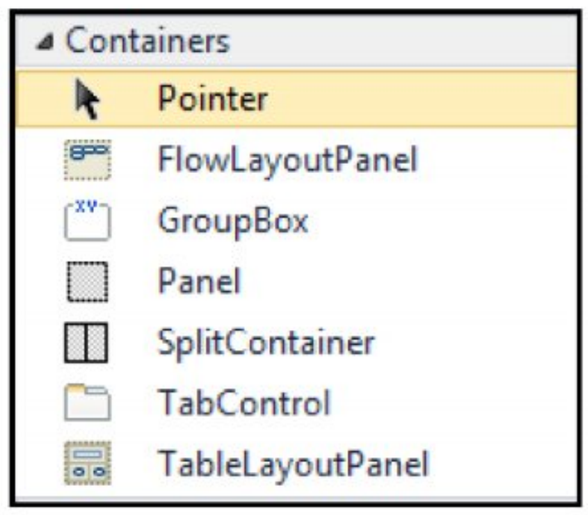
```
Dim a = (From i In CheckedListBox1.CheckedItems).ToList
  For Each i In a
   ' MsgBox(i)
   CheckedListBox1.Items.Remove(i.ToString)
  Next
  Label3.Text = CheckedListBox1.Items.Count()
```

Clear_Button_Click()

```
CheckedListBox1.Items.Clear()
Label3.Text = CheckedListBox1.Items.Count()
```



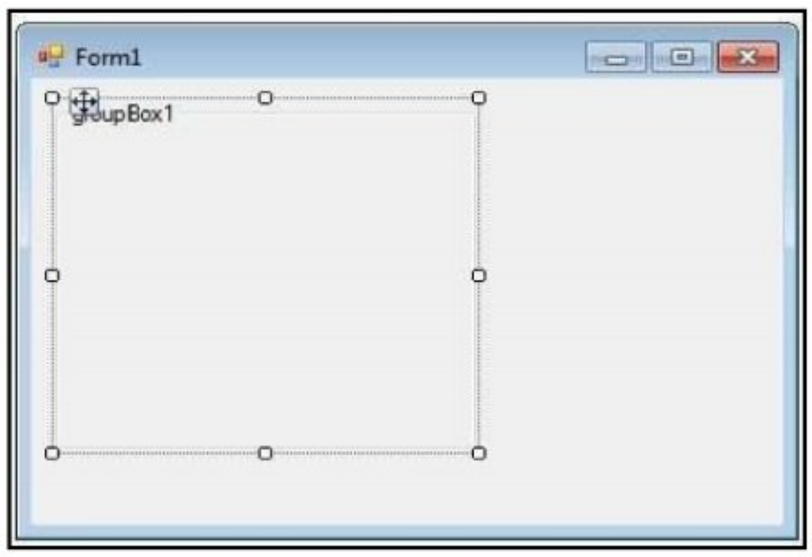
3.1.2. Container Controls

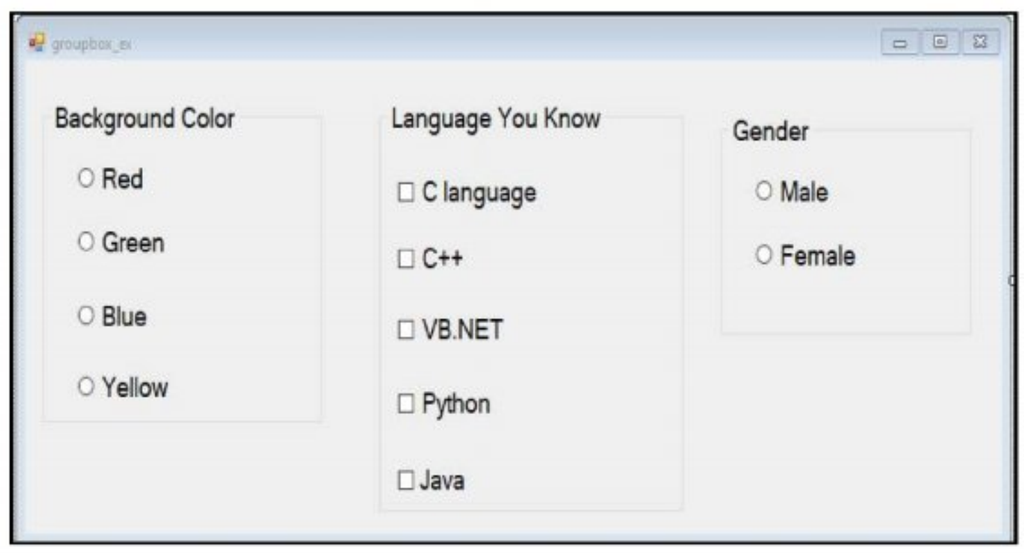


Container controls, as their name implies, are controls that can **host other controls inside them**. The Form is the perfect example here, as you put all your controls on the form. The host is known as the **parent**, and the controls inside the host are known as the **children**. Container controls can obviously host other Container controls as well; then you're looking at a **grandparent**, parent and child relationship.

(1) GroupBox

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





Property:

Text: It is used to speify the caption to be displayed for Groupbox control.

Event:

Enter: Occurs when the mouse is entered.

Methods:

OnEnter: It raise Enter event.

(2) Panel

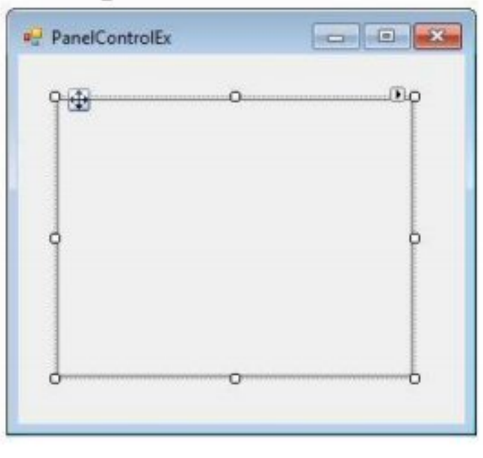
- It is used to group collections of controls.
- As GroupBox, it is a container for other objects.
- As with other container controls, if enabled property of the Panel control is set
- to false, the controls contained within the Panel will also be disabled.
- The Panel control is displayed by default without any borders.
- It does not have caption.

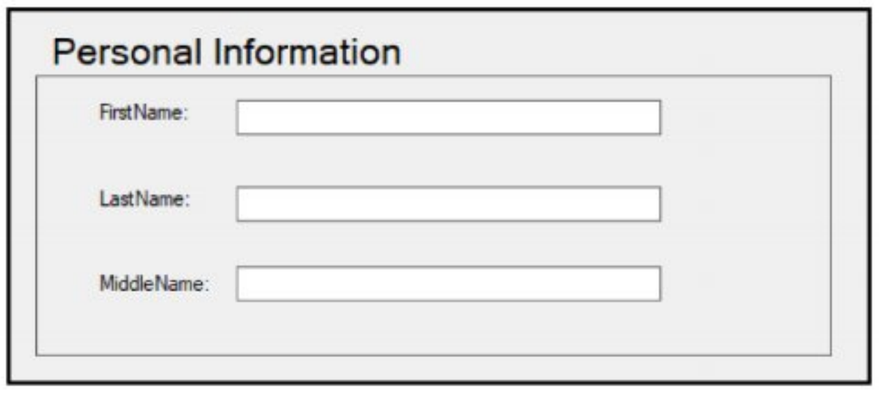
Panel Control Properties

Property	Description
BorderStyle	It indicates the border style for the control.
Enabled	It gets or sets a value indicating whether the control can respond to user interaction.
HorizontalScroll	It gets the characteristics associated with the horizontal scroll bar.
HScroll	It gets or sets a value indicating whether the horizontal scroll bar is visible.
VerticalScroll	It gets the characteristics associated with the vertical scroll bar.
VScroll	It gets or sets a value indicating whether the vertical scroll bar is visible.

Default Event: Paint()

Example:





(3) TabControl

- It contains tab pages, which are represented by TabPage objects.
- The order of tab pages in this collection reflects the order of the tabs appear in the control.
- The user can change the current TabPage by clicking one of the tabs in the control.
- It manages a related set of tab pages.

TabControl Properties

Property	Description
Alignment	It gets or sets the area of the control where the tabs are aligned.
Appearance	It gets or sets the visual appearance of selected tab of the control.
Image List	It gets or sets the images to display on the tabs.
ItemSize	It gets or sets the size of the tabs.
MultiLine	It gets or sets a value indicating whether more than one row of tabs can be displayed.
RowCount	It gets the number of rows that are currently being displayed in the selected tab of the control.
SelectedIndex	It gets or sets the index of the currently selected tab page.
SelectedTab	It gets or sets the currently selected tab page.
Show ToolTips	It gets or sets a value indicating whether a tab's ToolTip is shown when the mouse passes over the tab.
TabCount	It gets the number of tabs in the tab strip.
TabPages	It gets the collection of tab pages in tab control.

TabControl Methods

Method	Description
Deselect Tab	It unselects the tab of the control.
Remove All	It removes all the tab pages and additional controls from this tab control.
SelectTab	It makes the tab with the specified index as current tab.

✓ Default Event: Click()

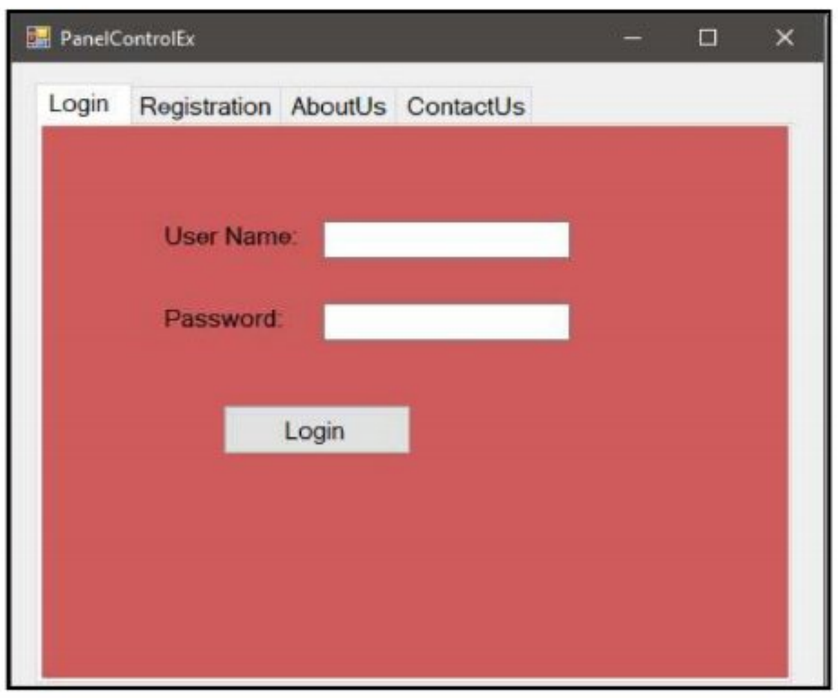
TabPage object

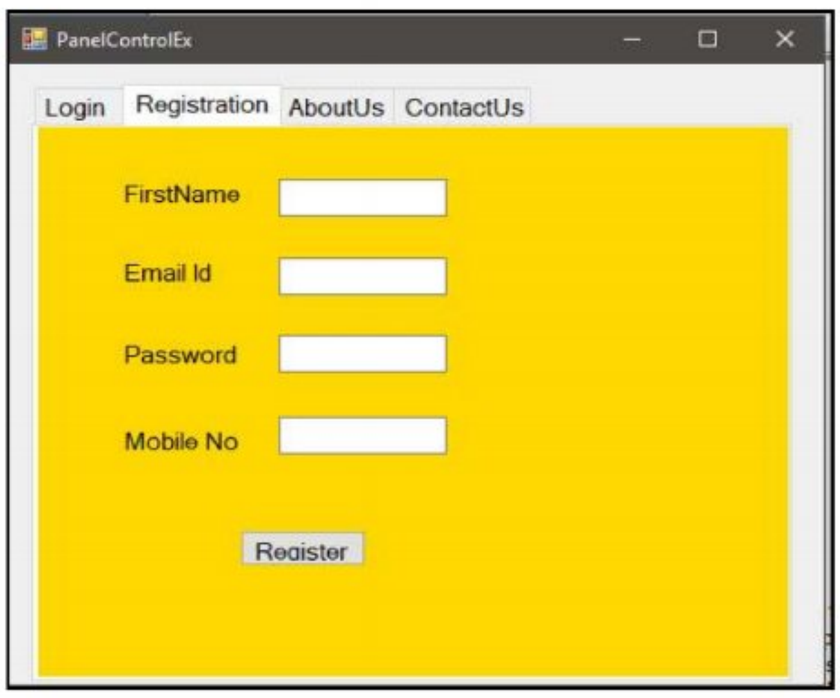
It represents single tab page of the TabControl.

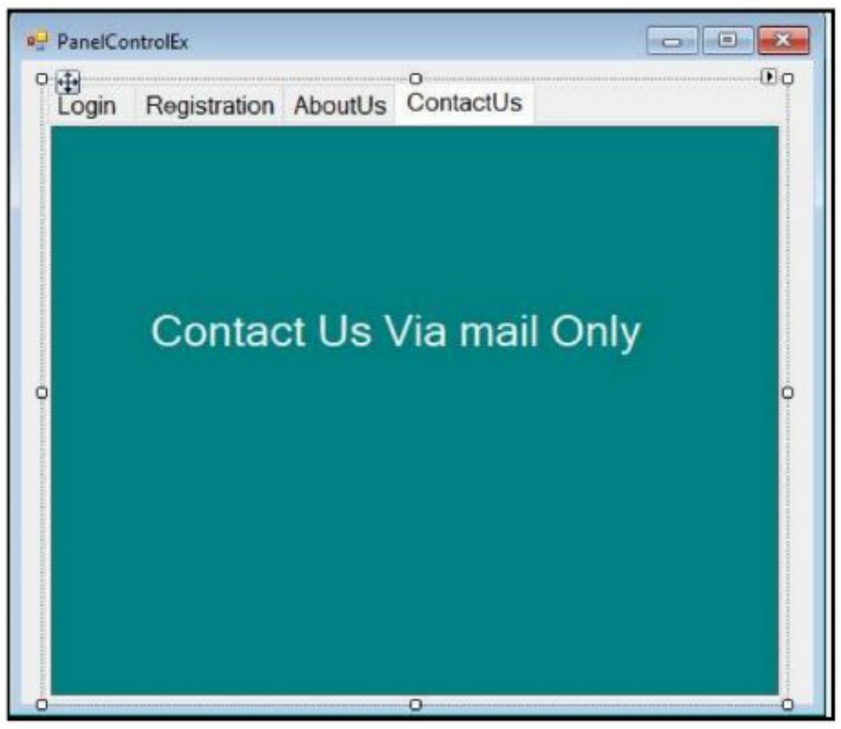
TabPage Properies

Property	Description
Image	Index It gets or sets the index to the image displayed on tab.
Text	It gets or sets the text to display on the tab.
ToolTipText	It gets or sets the ToolTip text for this tab.

Example:







(4) SplitContainer

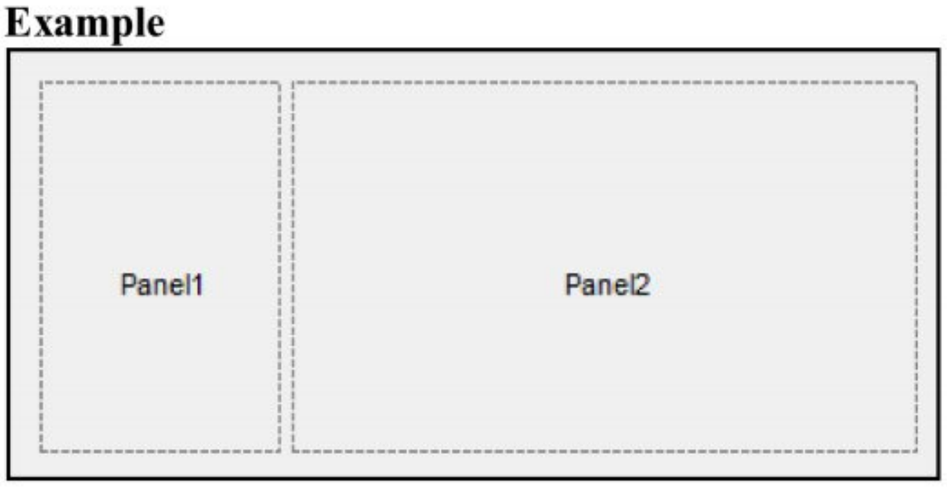
- It represents a control consisting of a movable bar that divides a display area of a container into two resizable panels. You can add controls to the two resizable panels, and you can add other SplitContainer controls to existing SplitContainer panels to create many resizable display areas.
- When the user passes the mouse pointer over the splitter, the cursor changes, which indicates that the controls inside the SplitContainer control can be resized.

SplitContainer Properties

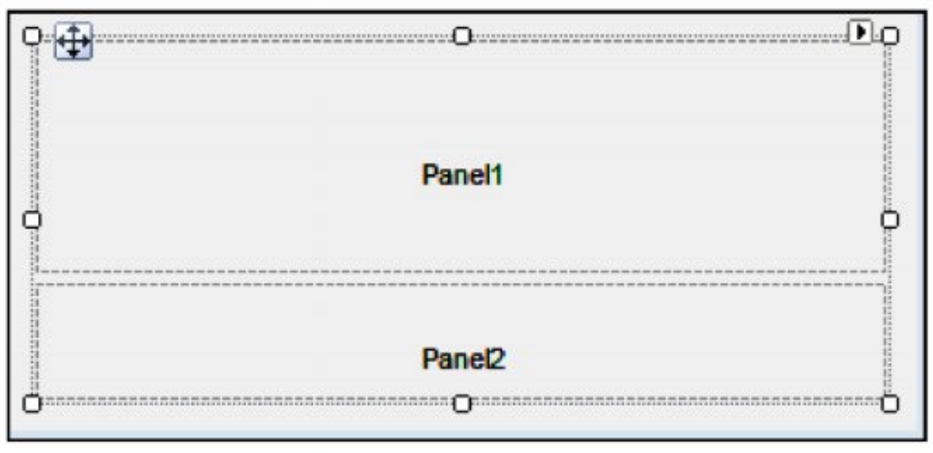
Property	Description
Active Control	It gets or sets the active control on the container control.
Orientation	It gets or sets a value indicating the horizontal or vertical orientation of the SplitContainer panels.
Panell	It gets the left or top panel of the Split Container, depending on Orientation.
Panell Collapsed	It gets or sets a value determining whether Panell is collapsed or expanded.
Panell Min Size	It gets or sets the minimum distance in pixels of the splitter from the right or bottom edge of Panell.
Panel2	It gets the right or bottom panel of the SplitContainer, depending on Orientation.
Panel2Collapsed	It gets or sets a value determining whether Panel2 is collapsed or expanded.
Panel2MinSize	It gets or sets the minimum distance in pixels of the splitter from the right or bottom edge of Panel2
SplitterDistance	It gets sets the location of the splitter, in pixels, from the left or top edge of the SplitContainer.
SplitterIncrement	It gets or sets a value representing the increment of splitter movement in pixels.
Splitter Width	It gets or sets the width of the splitter in pixels.

✓ Default Event: Paint()

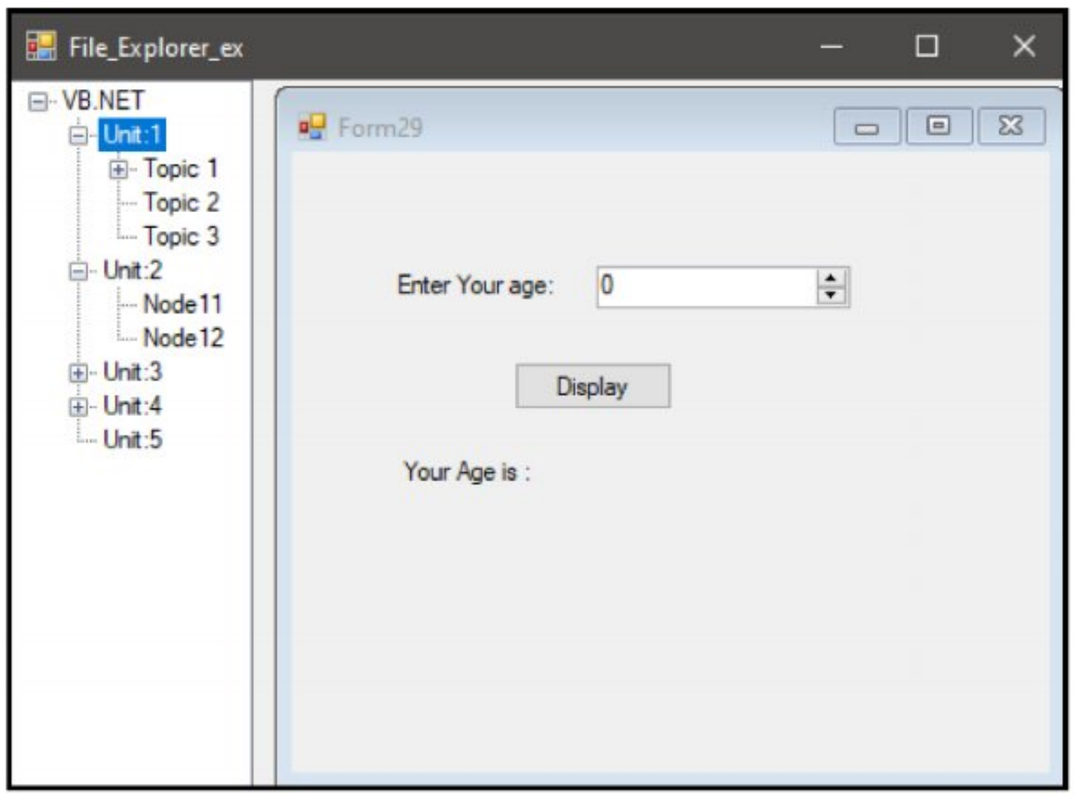
Example



Orientation:Vertical



Orientation:Horizontal



(5) FlowLayout Panel

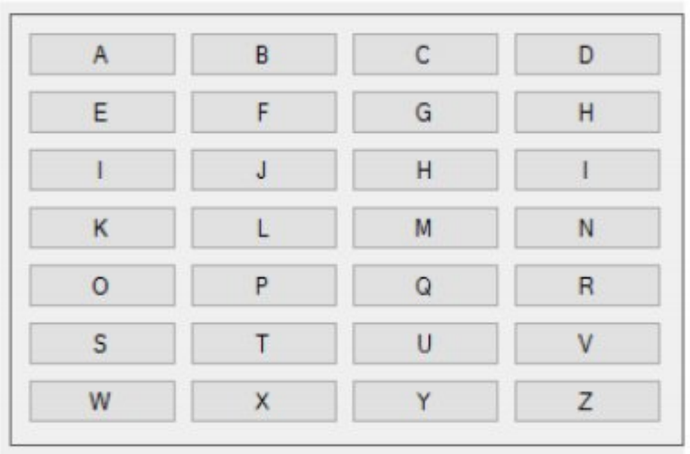
- It represents a panel that dynamically lays out its contents horizontally or vertically .
- Its contents can be wrapped from one row to the next or from one column to the next.
- Alternatively, its contents can be clipped instead of wrapped.

FlowLayout Panel Properties

Property	Description
DefaultPadding	It gets the internal spacing, in pixels, of the contents of a control.
FlowDirection	It gets or sets a value indicating the flow direction of the FlowLayout Panel control.
Margin	It gets or sets the space between controls.
Padding	It gets or sets padding within the control.
RightToLeft	It gets or sets a value indicating whether control's elements are aligned to support locals using right-to-left fonts.
WrapContents	It gets or sets a value indicating whether the control should wrap its contents or let the contents be clipped.

✓ Default Event: Paint()

Example:



(6) TableLayout Panel

- It represents a panel that dynamically lays out its contents in a grid composed of and columns .
- Margin, DefaultPadding, Padding properties of TableLayoutPanel is similar to FlowLayoutPanel Control.

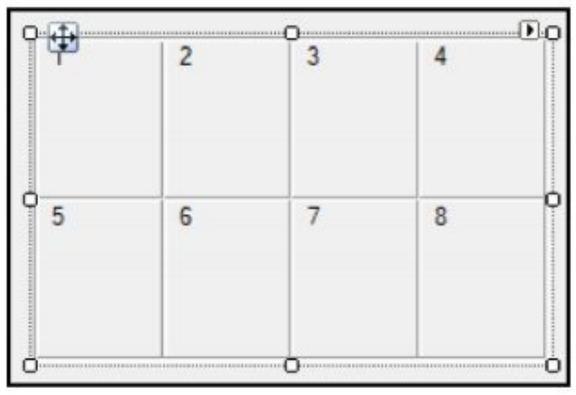
TableLayout Panel Properties

Property	Description
Cell BorderStyle	It gets or sets the style of the cell borders.
ColumnCount	It gets or sets the number of columns in the table.
ColumnStyles	It gets a collection of column styles for the Table LayoutPanel
RowCount	It gets or sets the number of rows in the table.
Row Styles	It gets a collection of row styles for the TableLayout Panel.

TableLayout Panel Methods

Method	Description
GetCellPosition	It gets Cell Position of the TableLayout Panel that represents the row and the column of the cell.
GetColumn	It returns the column position of the specified child control.
GetColumnSpan	It returns the number of columns spanned by the specified child control.
GetColumnWidths	It returns an array representing the widths, in pixels, of the columns in the TableLayout Panel.
GetRow	It returns the row position of the specified child control.
GetRowSpan	It returns the number of rows spanned by the specified child control.
GetRow Heights	It returns an array representing the heights, in pixels, of the rows in the TableLayoutPanel.
SetCellPosition	It sets Cell Position of the TableLayoutPanel that represents the row and the column of the cell.
SetColumnSpan	It sets the number of columns spanned by the child control.
SetColumn	It sets the column position of the specified child control.
SetRow	It sets the row position of the specified child control.
SetRowSpan	It sets the number of rows spanned by the child control.

Example:



3.1.4 Component Controls

(1) Image list
(2) Error provider
(3) Help provider
(4) Timer

(1) Image List

- It is used to store images.
- It is one kind of image repository.
- Controls having Imagelist and ImageIndex properties uses Imagelist control.
- Such controls are list view, tree view, toolbar, tab control, checkboxes, buttons, radio buttons and labels.

Properties:

Properties	Description
ColorDepth	Property gets the color depth for this image list.
Images	Property gets an ImageCollection object for this image list.
ImageSize	Property used to get or set the image size for the images in the list.
TransparentColor	Property used to get or set the transparent color for this list.

Methods:

Method	Description
Draw	Method used to draw the given image.

(2) Timer

- The timer control is a looping control used to repeat any task in a given time interval.
- It is an important control used in Client-side and Server-side programming
- Furthermore, if we want to execute an application after a specific amount of time, we can use the **Timer Control**.
- Once the timer is enabled, it generates a tick event handler to perform any defined task in its time interval property.
- It starts when the start() method of timer control is called, and it repeats the defined task continuously until the timer stops.

Properties:

Properties	Description
Name	The Name property is used to set the name of the control.
Enabled	The Enables property is used to enable or disable the timer control. By default, it is True.
Interval	An Interval property is used to set or obtain the iteration interval in milliseconds to raise the timer control's elapsed event. According to the interval, a timer repeats the task.

Events of Timer Control

Events	Description
Tick	A tick event is used to repeat the task according to the time set in the Interval property. It is the default event of a timer control that repeats the task between the Start() and Stop() methods.

Methods

Methods	Description
Start()	The Start() method is used to begin the Timer control's elapsed event by setting the Enabled property to true.
Stop()	The Stop() method is used to stop the timer control's elapsed event by setting Enabled property to false.

Example:

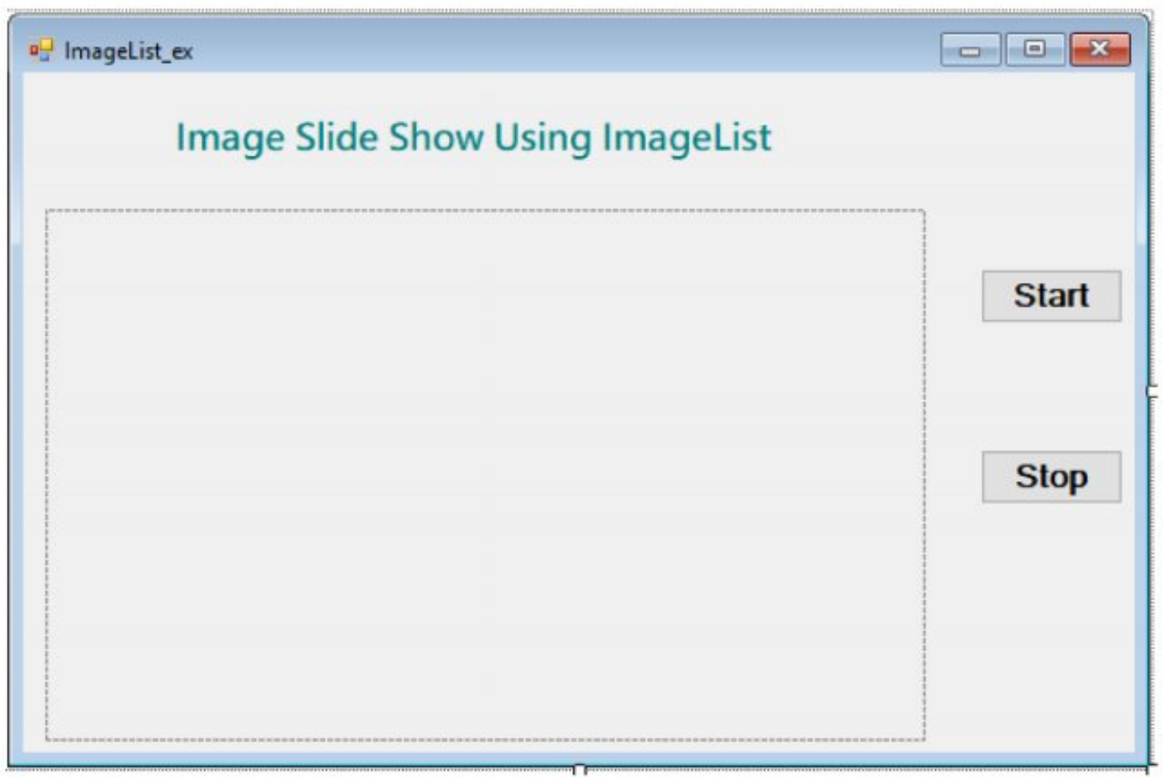
- **Set Images and ImageSize property of Imagelist Control.**

```
Dim n As Integer = 0

Start_Button.Click()
        Timer1.Start()
        Timer1.Interval = 500

Stop_Button.Click()
          Timer1.Stop()

Timer1.Tick():
      PictureBox1.Image = ImageList1.Images(n)
      If n = ImageList1.Images.Count() - 1 Then
        n = 0
      Else
        n += 1
      End If
```



(3) Error provider

- ErrorProvider allows us to set an error message for any control on the form when the input is not valid.
- When an error message is set, an icon indicating the error will appear next to the control and the error message is displayed as Tool Tip when the mouse is over the control.

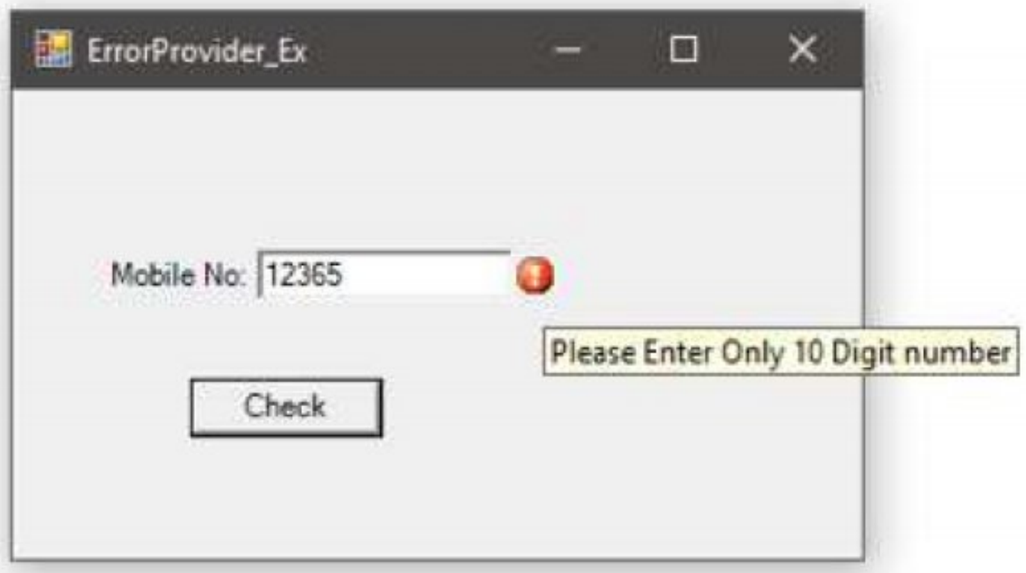
Properties:

Properties	Description
Icon	Gets or sets the Icon that is displayed next to a control when an error occurred.
BlinkRate	which allows setting the rate in milliseconds at which the icon blinks.
BlinkStyle	Gets or sets a value indicating when the error icon flashes.

Methods:

Methods	Description
GetIconAlignment	It gets a value indicating where the error icon should be placed in relation to the control.
SetIconAlignment	It gets a value indicating where the error icon should be placed in relation to the control.
SetError	It sets the error description string for the specified control.
GetError	It returns the current error description string for the specified control.

Example:



Check_Button()

```
If Not IsNumeric(TextBox1.Text) Or Not TextBox1.Text.Length = 10 Then
 ErrorProvider1.SetError(TextBox1, "Please Enter Only 10 Digit number")
Else
 ErrorProvider1.SetError(TextBox1, "")
 MsgBox(TextBox1.Text)
End If
```

(4) Help provider

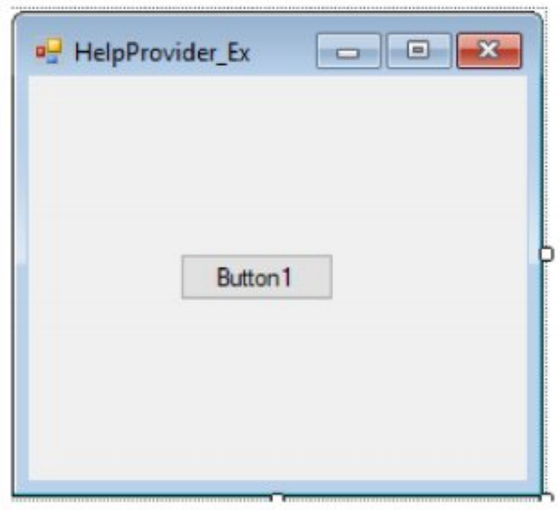
- This control provides pop-up or on line help for controls.
- HelpProvider control Provides help with your application is very useful as it allows your users to understand it more easily.
- Support for help in Visual Basic exists and you can display HTML files that can contain a set of linked topics.

- **Help class:**
  - The Help class allows us to display HTML help to users.
  - The Help class provides two methods: ShowHelp and ShowHelpIndex.

  ShowHelp - ShowHelp is used to display a help file for a particular control and requires that control to be displayed along with the help file. The URL you specify for the help file can be in the form of F:\myHelp (local machine) or http:\\www.vbheaven.com\help.htm (Web).

  ShowHelpIndex -ShowHelpIndex method is used to display the index of a specified help file. You call the ShowHelpIndex method just like you call the ShowHelp method.

- **HelpProvider Component:**
  - The HelpProvider component provides these additional properties to each control on the form. These properties are:
    HelpString - Determines the help string associated with a control.
    ShowHelp - Determines if help should be displayed for this control.

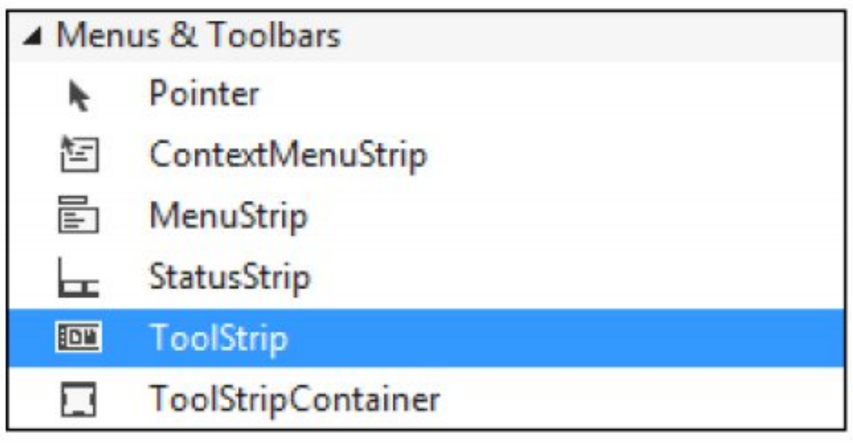


- Using the property display a help on the button when we press F1.
- Firstly set ShowHelp property of the The HelpProvider component(Button) ->True
- The following code displays a help string when Button2 has the focus and F1 key is pressed.

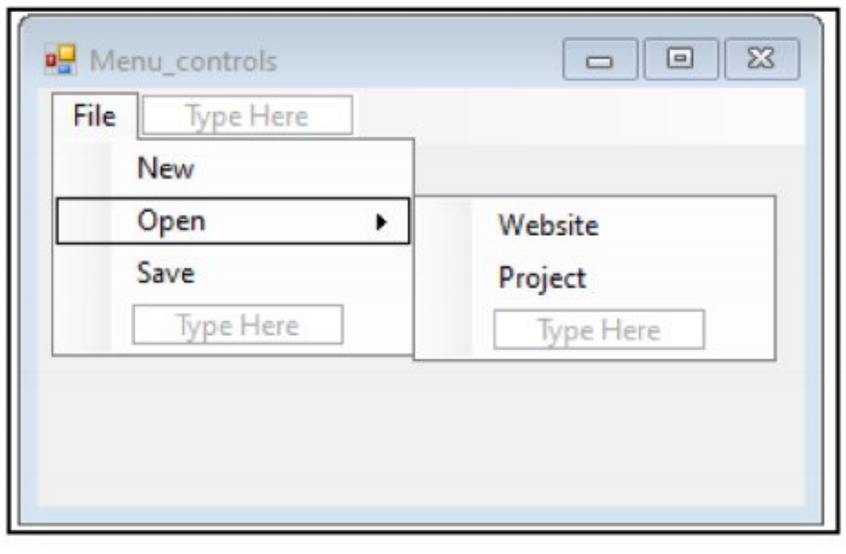
```
Form1_Load()
    HelpProvider1.SetHelpString(Button1, "Please Help me")
Button1_Click()
    Help.ShowHelp(Button1, "F:/help_file.html")
```

3.2. Working with Menus and Dialogue Boxes

❖ Menus & Toolbars



(1) MenuStrip



- A menu is used as a menu bar in the Windows form that contains a list of related commands, and it is implemented through MenuStrip Control.
- The Menu control is also known as the VB.NET **MenuStrip** Control.
- It provide a quick and easy way to add menus, menu items and submenu items to your application.
- It also provides a built-in editor that allows you to add, edit and delete menu items.
- The menus can contains access keys, shortcut keys, check marks or radio buttons.
  Access Key: An access key also known as accelerator key allows users to navigate the menus using the Alt key and a letter that is underlined in the menu item.

  Shortcut key: shortcut keys allow you to invoke the menu item without displaying the menus at all.

  Check marks and radiochecks: Either a checkmark symbol next to the menu item, or in the case of a radiocheck, a large dot next to the menu item. Both of these marks are indicators that the menu item is active.

Properties of the MenuStrip Control

Property	Description
Allow Item Reorder	It gets or sets a value indicating whether drag and drop item reordering are handled.
Allow Merge	It gets or sets a value indicating whether multiple MenuStrip can be combined.
Display Items	It gets the subset of items that are currently selected.
Enabled	It gets or sets a value indicating whether the control can respond to user interaction.
ImageList	It gets or sets the image list that contains the image to be displayed.
Items	It gets all the items.
Orientation	It gets the orientation of the ToolStripPanel.
ShowItem ToolTips	It gets or sets a value indicating whether ToolTips are shown for the MenuStrip
Stretch	It gets or sets a value indicating whether the MenuStrip stretches from end to end in its container.
Text	It gets or sets the text associated with this control.
TextDirection	It gets or sets the direction in which to draw text on a ToolStrip.

Methods of the MenuStrip Control

Method	Description
GetNextItem	It retrieves the next Item from the specified reference point and moving in the specified direction.
SetDisplayed Items	It resets the collection of displayed and overflow items after a layout is done.
SetItemLocation	It anchors an Item to a particular place.

Events of the MenuStrip Control

Events	Description
MenuActivate	When a user uses a menu bar control with a mouse or keyboard, a MenuActivate event occurs.
MenuDeactivate	The MenuDeactivate event occurs when the MenuStrip control is deactivated in the Windows form.

Default Event:

Click()

MenuItem

- It represents an individual item that is displayed within a MainMenu or ContextMenu.
- It is retained for both backward compatibility and future use.

Property	Description
BarBreak	It gets or sets a value indicating whether the MenuItem is placed on a new line or in a new column.
Checked	It gets or sets a value indicating whether a check mark appears next to the text of the menu item.
DefaultItem	It gets or sets a value indicating whether the menu item is the default menu item.
IsParent	It gets a value indicating whether the menu item contains child menu items.
MenuID	It gets a value indicating the Windows identifier for this menu item.
MenuItems	It gets a value indicating the collection of MenuItem objects associated with the menu.
MergeOrder	It gets or sets a value indicating the relative position of the menu item when it is merged with another.
Merge Type	It gets or sets a value indicating the behavior of this menu item when its menu is merged with another.
ShortCut	It gets or sets a value indicating the shortcut key associated with the menu item.
ShowShortCut	It gets or sets a value indicating whether the shortcut key that is associated with the menu item is displayed next to the menu item caption.
Text	It gets or sets a value indicating the caption of the menu item.

Methods:

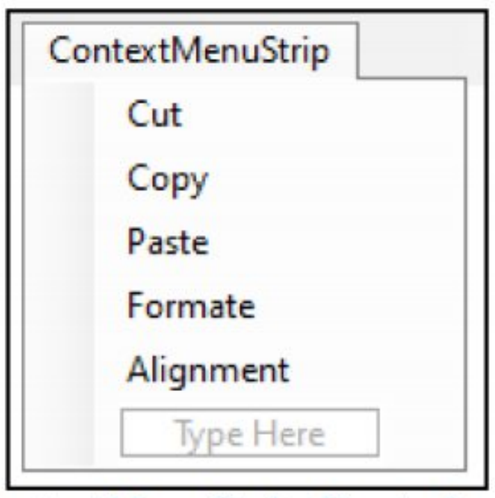
Method	Description
FindMenuItem	It gets the Menaltem that contains the value specified.
FindMergePosition	It returns the position at which a menu item should be inserted into the menu.
GetContextMenu	It gets the ContextMana that contains this menu
GetMainMenu	It gets the MainMenu that contains this menu
Marge Menu	It merges this Menultem with another MenuItem and returns the resulting merged MenuItem.

Events:

Event	Description
PerformClick	It raises the Click event for menu item.
PerformSelect	It raises the Select event for menu item.

(2) ContextMenuStrip

- It represents a shortcut menu.
- Context menu is a menu popup over controls, usually when user right clicks on them. They are called context menus because they appear over specific controls.
- Allowltem Reorder, AllowMerge, ImageList, Items, MaxItemSize, Orientation,
- OverflowButton, ShowItemToolTips, Stretch, Text, and TextDirection properties are similar to MenuStrip control.



Properties of the ContextMenuStrip Control

Property	Description
Allow Transparency	It gets or sets a value indicating whether the Opacity of the form can be adjusted
AutoClose	It gets or sets a value indicating whether the Drop Down control should automatically close when it has lost activation
Opacity	It determines the opacity of the form
Ownerltem	It gets or sets the Item that is the owner of this DropDown.
SourceControl	It gets the last control that caused this ContexMenuStrip to be displayed.
TopMost	It gets or sets a value indicating whether the form should be displayed as a topmost form.
ShowCheckMargin	It gets or sets a value indicating whether space for a check mark is shown on the left edge of the Menu Item
ShowImageMargin	It gets or sets a value indicating whether space for an image is shown on the left edge of the Menultem.

It supports all methods of MenuStrip control below table represents the methods and its meaning.

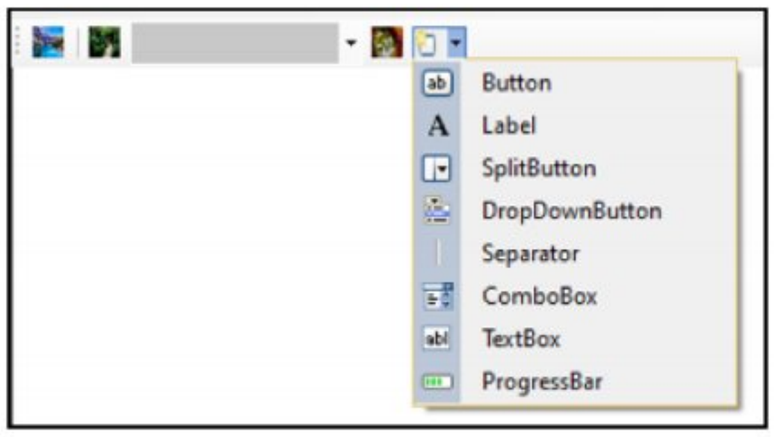
Methods

Method	Description
Close	It closes the Control.
Show	It displays the control on its default position.

✓ **Default Event: Click()**

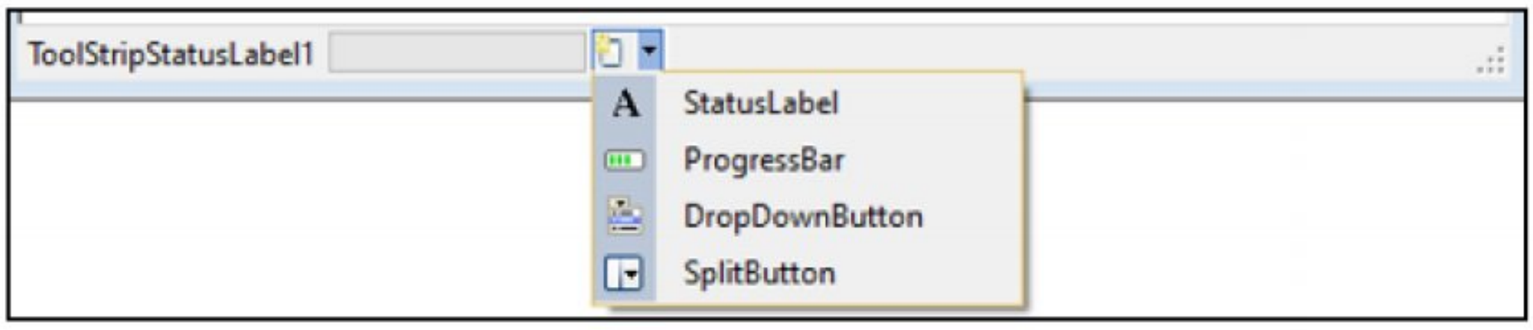
(3) ToolStrip

- It provides a container for Windows toolbar objects .
- ToolStrip is the base class for MenuStrip, ContextMenuStrip, and StatusStrip.
- It can contain Button, Separator, Label, DropDownButton, SplitButton, TextBox, and ComboBox.
- It supports AllowItem Reorder, AllowMerge, DisplayedItems, ImageList, Items, MaxItemSize, Orientation, OverflowButton, ShowItem ToolTips, Stretch, Text, and TextDirection properties, which are similar to MenuStrip control as mention as above table.
- Methods of ToolStrip control are similar to MenuStrip control as above table.



(4) StatusStrip

- It represents a windows Status bar.
- It can contain Status Label, DropDownButton, SplitButton, and ProgressBar.
- It displays information about an object being viewed on a Form, or information that relates to that operation of the object within your application.
- It supports AllowItemReorder, AllowMerge, DisplayedItems, Image List, Items, MaxItemSize, Orientation, OverflowButton, ShowItem ToolTips, Stretch, Text, and TextDirection properties, which are similar to MenuStrip control as mention in Menustrip topic.

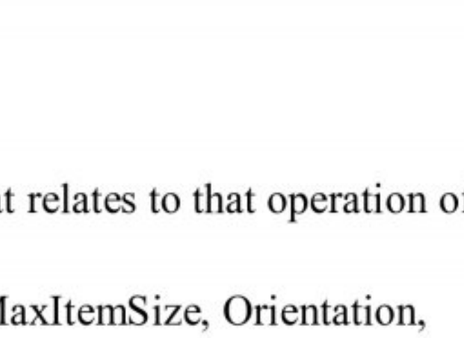


Methods of **StatuStrip** control are similar to MenuStrip control as mention in MenuStrip topic.

✓ **Default Event: Click()**

(5) ToolStrip Container

- It provides panels on each side of the form and a central panel that can hold one or more controls.
- It is similar to SplitContainer.
- It uses four docked side panels one central panel to create a typical arrangement.



Properties

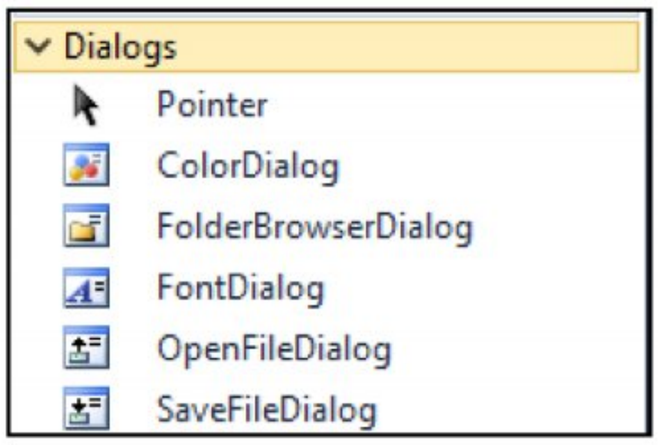
Property	Description
ActiveControl	It gets or sets the active control on the container control.
AutoScaleDimensions	It gets or sets the dimensions that the control was designed to.
AutoScaleFactor	It gets the scaling factor between the current and design-time automatic scaling dimensions.
AutoScale Mode	It gets or sets the automatic scaling mode of the control.
AutoScrollPosition	It gets or sets the location of the auto-scroll position.
ContentPanel	It gets the center panel of the ToolStripContainer.

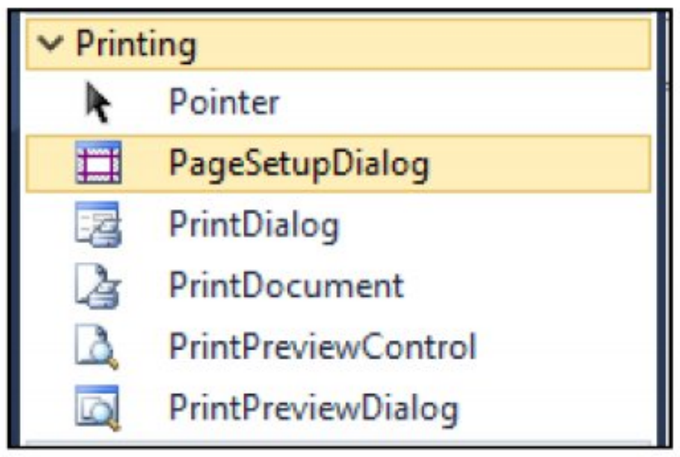
Methods of ToolStripContainer control are similar to **Panel control** as mention in Panel control topic.

✓ **Default Event: Click()**

❖ Dialoge Boxes

- Visual Basic .NET has built-in dialog boxes which allow user to create own Open, Save, Font, Color dialogs that are similar to other windows applications.
- They are available in Dialogs and Printing Tab of ToolBox.
- The Dialog Boxes are as follows:
  (1) OpenFileDialog
  (2) SaveFileDialog
  (3) FontDialog
  (4) ColorDialog
  (5) PrintDialog
  (6) PrintPreviewDialog
  (7) PageSetupDialog
  (8) FolderBrowserDialog.

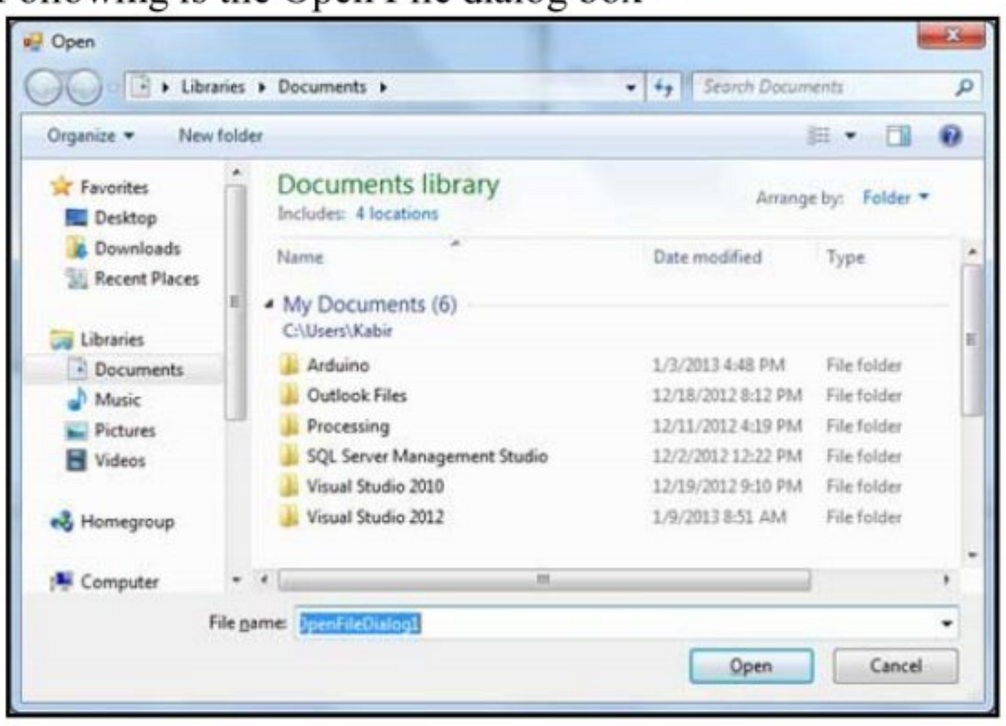




(1) OpenFileDialog

- The **OpenFileDialog** control prompts the user to open a file and allows the user to select a file to open.
- The user can check if the file exists and then open it.
- The OpenFileDialog control class inherits from the abstract class **FileDialog**.

Following is the Open File dialog box −



Properties of the OpenFileDialog Control

Property	Description
AddExtension	Gets or sets a value indicating whether the dialog box automatically adds an extension to a file name if the user omits the extension.
CheckFileExists	Gets or sets a value indicating whether the dialog box displays a warning if the user specifies a file name that does not exist.
DefaultExt	Gets or sets the default file name extension.
FileName	Gets or sets a string containing the file name selected in the file dialog box.
FileNames	Gets the file names of all selected files in the dialog box.
InitialDirectory	Gets or sets the initial directory displayed by the file dialog box.
Multiselect	Gets or sets a value indicating whether the dialog box allows multiple files to be selected.
ReadOnlyChecked	Gets or sets a value indicating whether the read-only check box is selected.
ShowHelp	Gets or sets a value indicating whether the Help button is displayed in the file dialog box
ShowReadOnly	Gets or sets a value indicating whether the dialog box contains a read-only check box.
Title	Gets or sets the file dialog box title.
ValidateNames	Gets or sets a value indicating whether the dialog box accepts only valid Win32 file names.

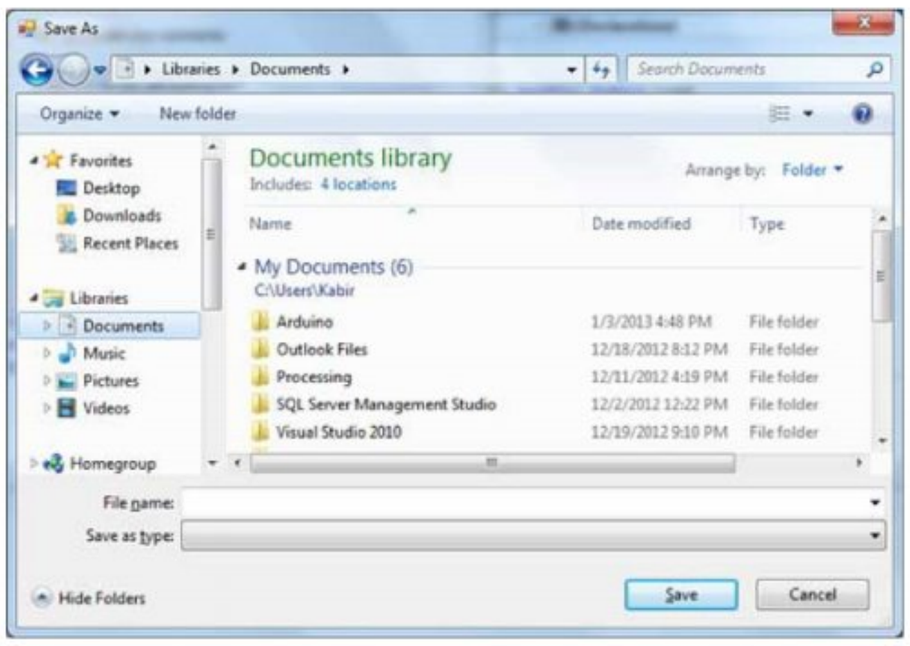
Methods of the OpenFileDialog Control

Method Name	Description
OpenFile	Opens the file selected by the user, with read-only permission. The file is specified by the FileName property.
Reset	Resets all options to their default value.
OnFileOk	It raises the FileOk event.
ShowDialog	It runs a common dialog box with a default owner.

- ✓ **Default Event:** FileOk()

(2) SaveFileDialog

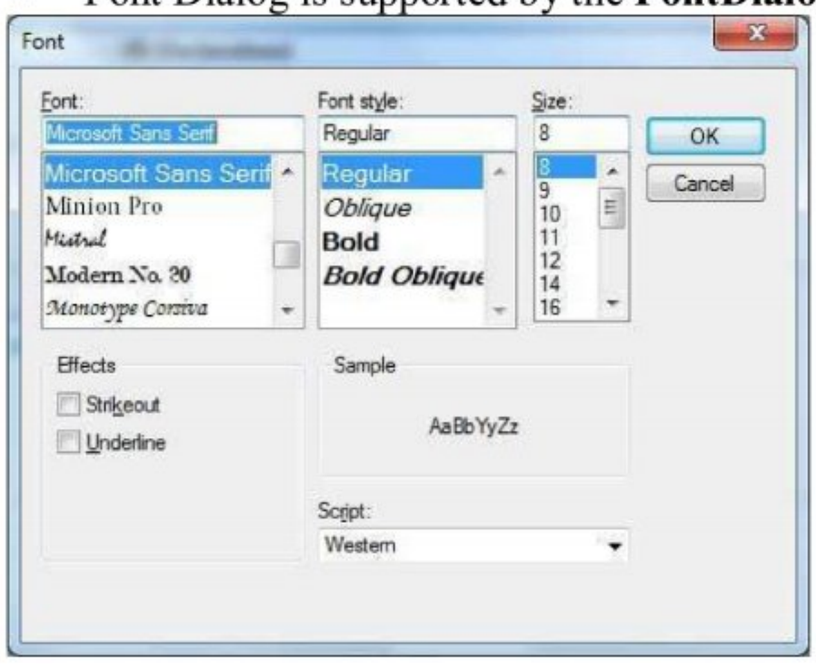
- The **SaveFileDialog** control prompts the user to select a location for saving a file and allows the user to specify the name of the file to save data.
- The SaveFileDialog control class inherits from the abstract class **FileDialog**.
- Following is the Save File dialog box –



- ✓ **Properties and methods of the Save File Dialog are the same as that of the Open File Dialog.**
- ✓ **Default Event:** FileOk()

(3) FontDialog

- Font Dialog is supported by the **FontDialog** class. It allow user to select a font size, face, style, etc.



By default, the Color ComboBox is not shown on the Font dialog box. You should set the **ShowColor** property of the FontDialog control to be **True**.

Properties of the FontDialog Control

Property	Description
AllowVectorFonts	Gets or sets a value indicating whether the dialog box allows vector font selections.
AllowVerticalFonts	Gets or sets a value indicating whether the dialog box displays both vertical and horizontal fonts, or only horizontal fonts.
Color	Gets or sets the selected font color.
Font	Gets or sets the selected font.
MaxSize	Gets or sets the maximum point size a user can select.
MinSize	Gets or sets the minimum point size a user can select.
ShowApply	Gets or sets a value indicating whether the dialog box contains an **Apply** button.
ShowColor	Gets or sets a value indicating whether the dialog box displays the color choice.
ShowEffects	Gets or sets a value indicating whether the dialog box contains controls that allow the user to specify strikethrough, underline, and text color options.
ShowHelp	Gets or sets a value indicating whether the dialog box displays a Help button.

Methods of the FontDialog Control

Method Name	Description
Reset	Resets all options to their default values.
RunDialog	When overridden in a derived class, specifies a common dialog box.
ShowDialog	Runs a common dialog box with a default owner.

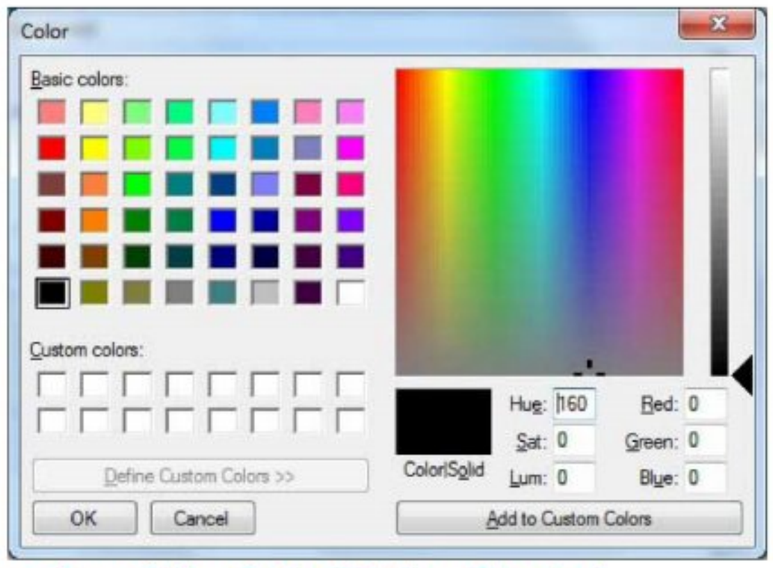
Events of the FontDialog Control

Event	Description
Apply	Occurs when the Apply button on the font dialog box is clicked.

(4) Color Dialogbox

- Color Dialog is supported by the **FontDialog** class and it allow user to select a color.
- The ColorDialog control class represents a common dialog box that displays available colors along with controls that enable the user to define custom colors.
- The main property of the ColorDialog control is *Color*, which returns a **Color** object.

Following is the Color dialog box −



Properties of the ColorDialog Control

Property	Description
AllowFullOpen	Gets or sets a value indicating whether the user can use the dialog box to define custom colors.
AnyColor	Gets or sets a value indicating whether the dialog box displays all available colors in the set of basic colors.
Color	Gets or sets the color selected by the user.
CustomColors	Gets or sets the set of custom colors shown in the dialog box.
ShowHelp	Gets or sets a value indicating whether a Help button appears in the color dialog box.
SolidColorOnly	Gets or sets a value indicating whether the dialog box will restrict users to selecting solid colors only.

Methods of the ColorDialog Control

Method Name	Description
Reset	Resets all options to their default values, the last selected color to black, and the custom colors to their default values.
RunDialog	When overridden in a derived class, specifies a common dialog box.
ShowDialog	Runs a common dialog box with a default owner.
OnHelpRequest	It raises the HelpRequest event.

Events of the ColorDialog Control

Event	& Description
HelpRequest	Occurs when the user clicks the Help button on a common dialog box.

(5) FolderBrowserDialog

- FolderBrowserDialog is supported by the **FolderBrowserDialog** class and it prompts the user to select a folder.

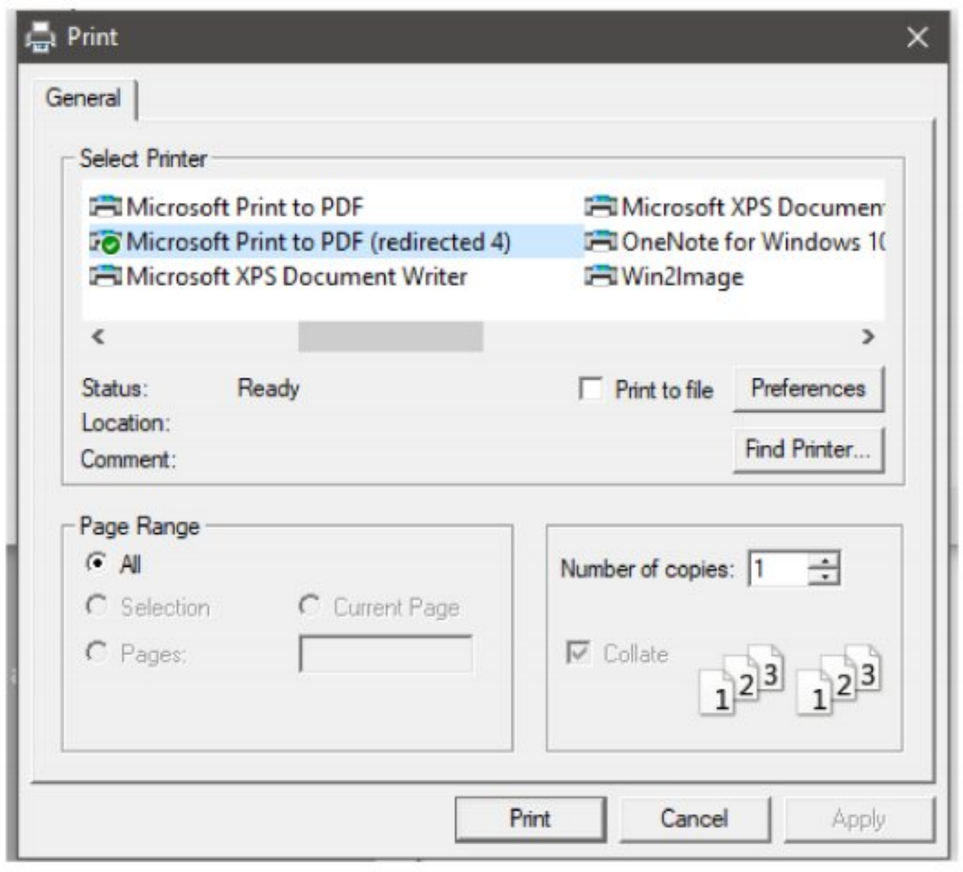
Properties of the FolderBrowserDialog Control

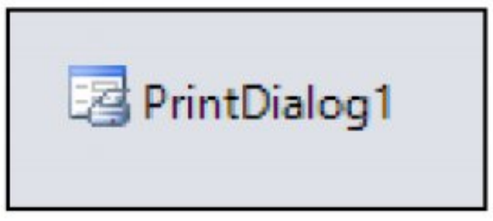
Property	Description
RootFolder	It gets or sets the root folder where the browsing starts from.
SelectedPath	It gets or sets the path selected by the user.
ShowNewFolderButton	It gets or sets a value indicating whether the new folder button appears in the folder browser dialog box.

- ✓ It also has OnHelpRequest, Reset and ShowDialog methods same as ColorDialog and FontDialog, OpenFileDialog, SaveFileDialog.
- ✓ Default Event:HelpRequest()

(6) PrintDialog

- Printing is the one of the common tasks during application development.
- The .NET Framework provides excellent support for printing documents.
- Print Dialog is supported by the PrintDialog class and it allow user to print document.
- It lets user to select a printer and choose which sections of the document to print from a Windows Forms application.
- Following is the Print dialog box –





Properties of the PrintDialog Control

Property	Description
AllowCurrentPage	Gets or sets a value indicating whether the **Current Page** option button is displayed.
AllowPrintToFile	Gets or sets a value indicating whether the **Print to file** check box is enabled.
AllowSelection	Gets or sets a value indicating whether the **Selection** option button is enabled.
AllowSomePages	Gets or sets a value indicating whether the **Pages** option button is enabled.
Document	Gets or sets a value indicating the PrintDocument used to obtain PrinterSettings.
PrinterSettings	Gets or sets the printer settings the dialog box modifies.
PrintToFile	Gets or sets a value indicating whether the **Print to file** check box is selected.
ShowHelp	Gets or sets a value indicating whether the **Help** button is displayed.
ShowNetwork	Gets or sets a value indicating whether the **Network** button is displayed.

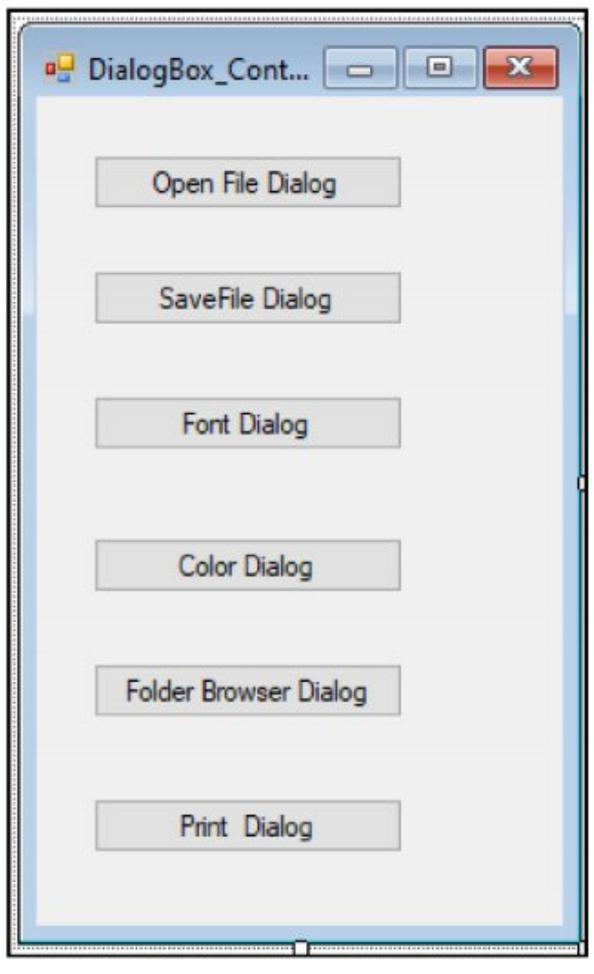
Methods of the PrintDialog Control

Method Name	Description
Reset	Resets all options to their default values.
RunDialog	When overridden in a derived class, specifies a common dialog box.
ShowDialog	Runs a common dialog box with a default owner.

There are various other controls related to printing of documents. Let us have a brief look at these controls and their purpose. These other controls are −

- The **PrintDocument** control − it provides support for actual events and operations of printing in Visual Basic and sets the properties for printing.
- The **PrinterSettings** control − it is used to configure how a document is printed by specifying the printer.
- The **PageSetUpDialog** control − it allows the user to specify page-related print settings including page orientation, paper size and margin size.
- The **PrintPreviewControl** control − it represents the raw preview part of print previewing from a Windows Forms application, without any dialog boxes or buttons.
- The **PrintPreviewDialog** control − it represents a dialog box form that contains a PrintPreviewControl for printing from a Windows Forms application.

Example:



Open File Dialog_Button_Click()

OpenFileDialog1.ShowDialog()

Save File Dialog_Button_Click()

SaveFileDialog1.ShowDialog()

Font Dialog_Button_Click()

FontDialog1.ShowDialog()

Color Dialog_Button_Click()

ColorDialog1.ShowDialog()

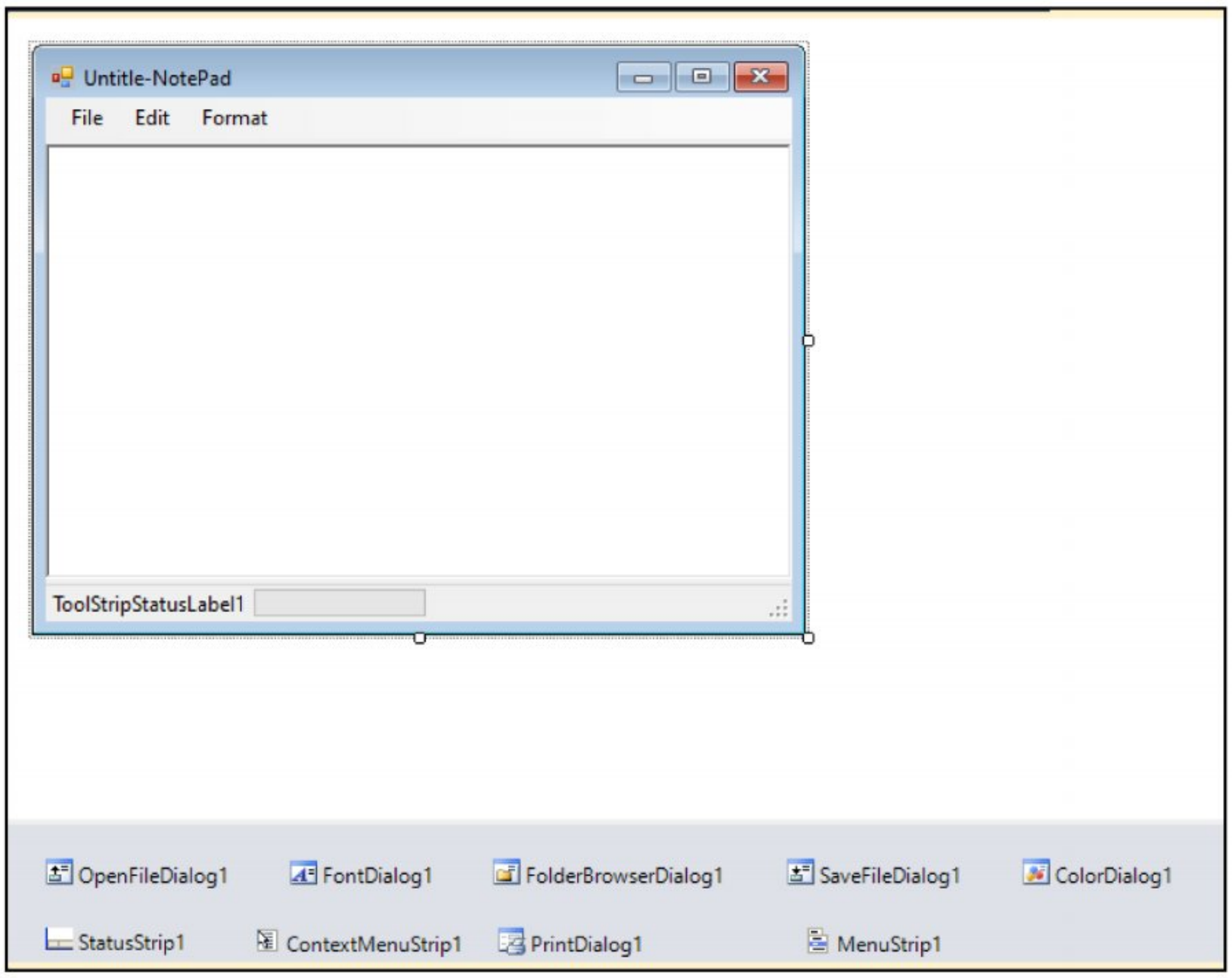
Folder Browser Dialog_Button_Click()

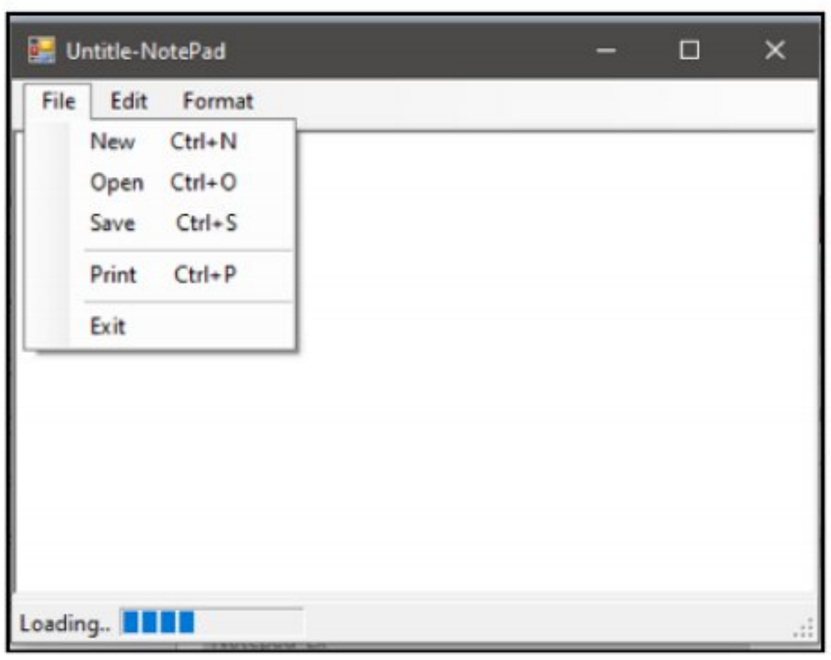
FolderBrowserDialog1.ShowDialog()

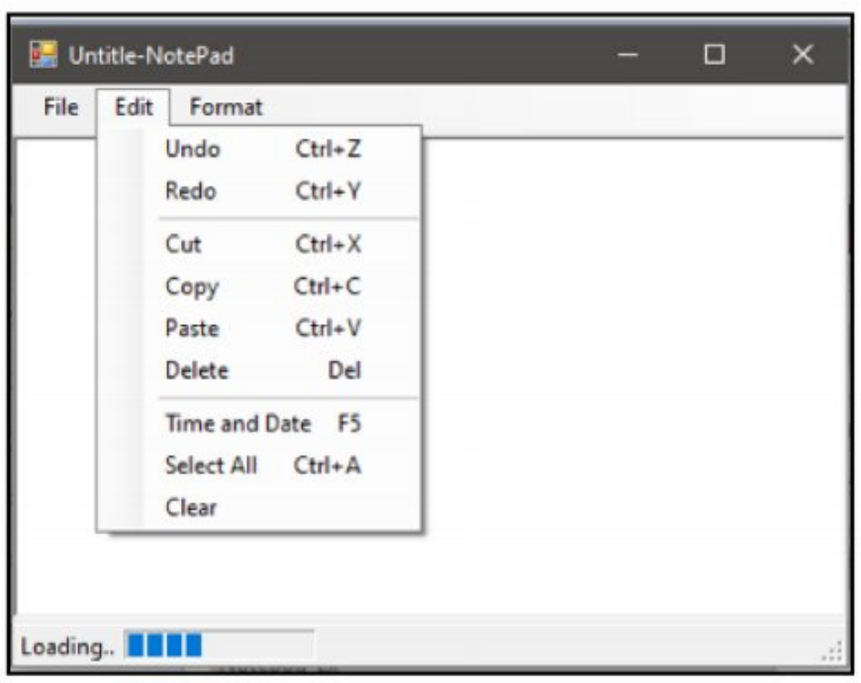
Print Dialog_Button_Click()

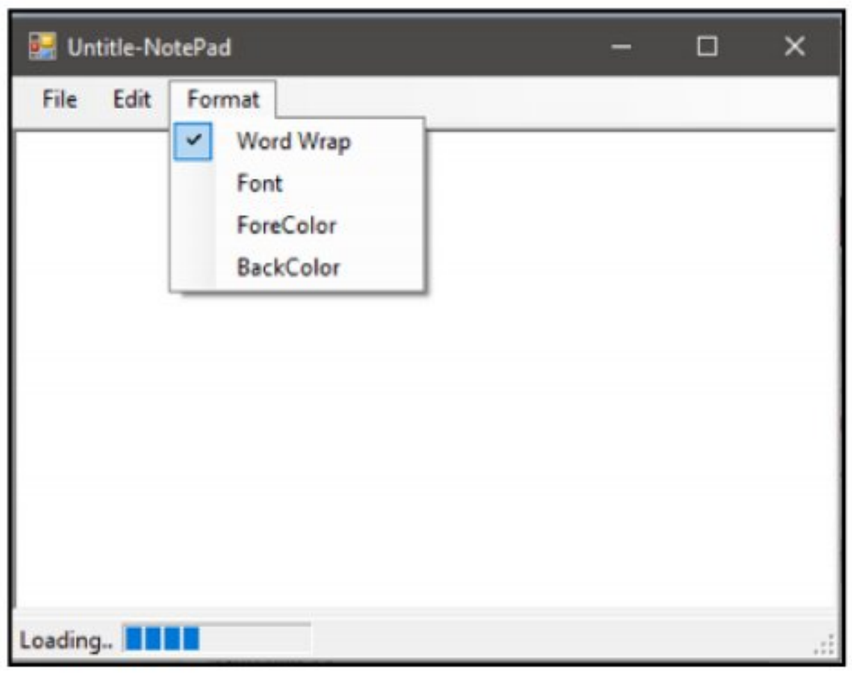
PrintDialog1.ShowDialog()

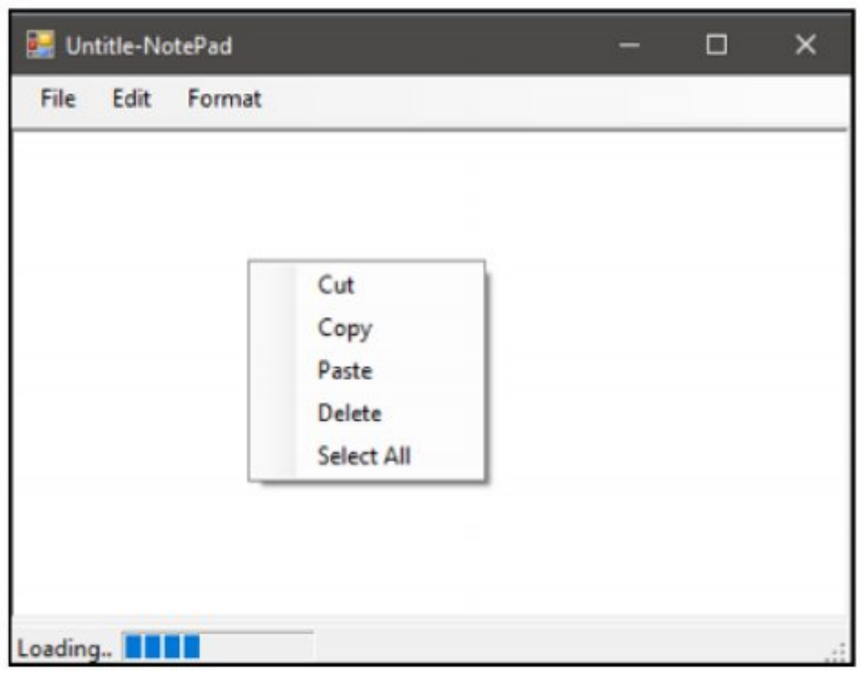
Example: (Notepad Simulation)











Code:

Notepad_Load()

```
ToolStripStatusLabel1.Text = "Loading.."
ToolStripProgressBar1.Value = 40
```

New:

```
If RichTextBox1.Text.Length <> 0 Then
    If MsgBox(" Save File ?", MsgBoxStyle.Information + MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then
        If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
            RichTextBox1.SaveFile(SaveFileDialog1.FileName)
        End If
    Else
      RichTextBox1.Clear()
      Me.Text = "Untitled-Notepad"
    End If
 End If
```

Open:

```
If OpenFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
        RichTextBox1.LoadFile(OpenFileDialog1.FileName)
End If
```

Save:

```
If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
        RichTextBox1.SaveFile(SaveFileDialog1.FileName)
    End If
```

Exit:

```
If RichTextBox1.Text.Length <> 0 Then
  If MsgBox("Do you Really want Exit ?", MsgBoxStyle.Information + MsgBoxStyle.YesNo) = MsgBoxResult.Yes
Then
            If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
                RichTextBox1.SaveFile(SaveFileDialog1.FileName)
            End If
        End If
    Else
        Me.Close()
    End If
```

Undo:

```
RichTextBox1.Undo()
```

Cut:

```
RichTextBox1.Cut()
```

Paste:

```
RichTextBox1.Paste()
```

Copy:

```
RichTextBox1.Copy()
```

Delete:

```
RichTextBox1.SelectedText = " "
```

SelectAll:

```
RichTextBox1.SelectAll()
```

Clear:

```
RichTextBox1.Clear()
```

Time & Date:

```
RichTextBox1.SelectedText = Date.Now
```

Font:

```
If FontDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
   RichTextBox1.SelectionFont = FontDialog1.Font
End If
```

WordWrap:

```
If RichTextBox1.WordWrap = True Then
        RichTextBox1.WordWrap = False
    Else
        RichTextBox1.WordWrap = True
    End If
```

Forecolor:

```
If ColorDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
        RichTextBox1.ForeColor = ColorDialog1.Color
    End If
```

BackColor:

```
If ColorDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
        RichTextBox1.BackColor = ColorDialog1.Color
    End If
```

Print:

```
PrintDialog1.ShowDialog()
```

3.3. Exception Handling in VB.NET

- **Error:**

Error means bug that means when you compile or run a program, abnormal situation may raise which is known as error.

There are different types of Error:

1. **Syntax Error**: The error raised by misspelled keywords or variables is called Syntax Error.
   Compiler will normally catch these errors while you are coding or after the compilation of program.
2. **Logic Error**: Due to the mistake present in the sequence of statements, code does not act as expected.
   This type of error is called Logic Error.
3. **Runtime Error**: The error which occurs when the code is executing, is called Runtime Error.

- **Exception**

Exceptions are run-time errors, which are raised when program is running. There are two ways to handle errors that occur at runtime:

(1) Unstructured exception handling
(2) Structured exception handling

- **Structured Exception Handling:**
  - It is implemented with Try...Catch... Finally statement, which is divided into a Try block, optional Catch blocks and an optional Finally block.
  - The Try block contains code where exceptions can occur; the Catch block contains code to handle the exceptions that occur.
  - If an exception occurs in the Try block the code throws the exception, so it can be caught and handled by the appropriate Catch statement.
  - After the rest of the statement finishes, execution is always passed to the Finally block.

The general syntax of Try…Catch...Finally block is as follow:

```
Try
    [ tryStatements]
    [Catch [ exception1  [ As type 1 ] ] [ When expression 1]
        CatchStatement1
    [Exit Try ]

    [Catch [ exception2  [ As type 2 ] ] [ When expression2]
        CatchStatement2
    [Exit Try ]
    ……..
    [Catch [exception N  [ As type N] ] [ When expression N]
        CatchStatement N
    [Exit Try ]
     [ Finally
        [ finallyStatements ]  ]
End Try
```

- Here typel, type2, ...,typeN in the Catch block are the Exception type which can be name of the Exception class that you want to handle.
- Exceptions are based on the Exception classes. The base class in the hierarchy of the Exception classes is System.Exception.
- It has two derived classes System.ApplicationException and System.SystemException.

- **Try**
  - A Try block identifies a block of code for which particular exceptions will be activated.
  - It's followed by one or more Catch blocks.
- **Catch**
  - A program catches an exception with an exception handler at the place in a program where you want to handle the problem.
  - The Catch keyword indicates the catching of an exception.
- **Finally**
  - The Finally block is used to execute a given set of statements, whether an exception is thrown or not thrown.
  - It will necessary when you used external resources in your program. Its is your program responsibility to release those resources when program no more needed.
  - For example, if you open a file, it must be closed whether an exception is raised or not.
- **Exit Try**
  - It is Optional keyword that breaks out of the Try… Catch…Finally structure.

The following table provides some of the predefined exception classes derived from the **System. SystemException** class:

Class Name	Description
System.IO. IOException	Handles I/O errors.
System. IndexOutOfRange Exception	Handles errors generated when a method refers to an array index out of range.
System. ArrayTypeMismatchException	Handles errors generated when type is mismatched with the array type.
System. NullReferenceException	Handles errors generated from differencing a null object.
System. DivideByZeroException	Handles errors generate from dividing a divided with zero.
System. InvalidCastException	Handles errors generated during typecasting.
System. OutOfMemory Exception	Handles errors generated from insufficient free memory.

Example:

```
Dim n1, n2, c As Integer
Button1_Click()
Try
   n1 = TextBox1.Text
   n2 = TextBox2.Text
   c = n1 \ n2
   MsgBox(c)
Catch ex As DivideByZeroException
  'MsgBox(ex.ToString)
  'MsgBox(ex.Message)
   MsgBox("Sorry!!! You Can't Devided any value with Zero")
Finally
   MsgBox("Execution Completed... Always Executed")
End Try
```

Example(When clause):

```
Dim n1, n2, c As Integer
Button1_Click()
 Try
   n1 = TextBox1.Text
   n2 = TextBox2.Text
   c = n1 \ n2
   MsgBox(c)
 Catch When Err.Number = 11
   MsgBox("Sorry!!! You Can't Devided any value with Zero")
Finally
        MsgBox("Execution Completed... Always Executed")
 End Try
```

Example(Nested Try Block):

```
Dim n1, n2, c As Integer
Sub Button1_Click()
  Try
    Try
      n1 = TextBox1.Text
      n2 = TextBox2.Text
       c = n1 \ n2
       MsgBox(c)
     Catch ex As DivideByZeroException
       MsgBox("Sorry!!! You Can't Devided any value with Zero")
     End Try
       c = n1 / n2
   Catch ex As OverflowException
      MsgBox("Overflow Exception handled")
   End Try
End Sub
```

Throwing an Exception

You can throw an exception using the **throw** statement and you can also rethrow a caught exception using the **Throw** statement.

Example:

```
Try
        Throw New OverflowException
Catch ex As OverflowException
        MsgBox("Overflow Exception handled")
        'MsgBox(ex.Message)
End Try
```

Throwing an Custom Exception

You can customize the exceptions by creating a new exception object based on the **ApplicationException** object.

Example:

```
Try
  If TextBox1.Text = "" Then
    Throw New ApplicationException("Exception:Please Enter a Number")
  Else
    MsgBox("You Entered: " & TextBox1.Text)
  End If
Catch ex As Exception
    MsgBox(ex.Message)
End Try
```

❖ **Unstructured Exception Handling:**

- Unstructured Exception Handling is implemented with On Error Go To statement, which is placed at the beginning of a code block to handle all possible exceptions that occurs during the execution of the code.

The general syntax of On Error GoTo statement is as follow.

Syntax:

On Error (GoTo [Line | Label | 0 | -1]| Resume Next | Line}

Go To label | line - The argument is any line label or line number.
If an exception occur program exception goes to the given location.

GoTo 0 - Disables enabled exception handler in the current procedure and reset it nothing. It clears the Error object.

Resume Next - It specifies that when an exception occurs, execution skips over the statement that cause the problem and goes to the statement immediately following a continues from that point.

Resume Line |Label - Line is the line number or label that specifies where to resume execution.

Err Object:

- When an error occurs, the **Err** object contains information about the error, which helps you to determine whether you can attempts to fix the error or ignore the error.
- The **Err** object also has method that allows you to raise(Raise()) errors or clear(Clear()) the state of the **Err** object.

The following table describe **Err** Object properties and description of each.

Property	Description
Description	Text message providing a short description of the error.
HelpContext	Integer containing the context ID for a topic in a Help file.
HelpFile	String expression containing the fully qualified path to a help file.
Number	Numeric value specifying an error.
Source	String expression representing the object or application that generated the error.

On Error Go To Line|Label

- If an exception occur program exception goes to the given location.
- The line argument is any line label or line number.

Example:

```
Dim n1, n2, c As Integer
   Button1_Click()
      On Error GoTo err_msg
      n1 = TextBox1.Text
      n2 = TextBox2.Text
      c = n1 / n2
      MsgBox("Division:" & c)
      Exit Sub
err_msg:
      MsgBox("Sorry!!! You Can't Devided any value with Zero")
      MsgBox(Err.Number & "   " & Err.Description)
```

Here, if Exception occurs then the execution is continued from the label specified with the On Error GoTo Statement.
Note: You must put Exit statement before line handler otherwise line handler code will always executed regardless of exception is generated or not.

2. On Error GoTo Resume Next

• One of the most useful aspects of unstructured exception handling is the **Resume** statement, which lets you resume program execution to the next line even after an exception has occurred.

Example:

```
Dim n1, n2, c As Integer
Sub Button1_Click()
    On Error GoTo err_msg
    n1 = TextBox1.Text
    n2 = TextBox2.Text
    c = n1 / n2
    MsgBox("Division:" & c)
    If n2 = 0 Then
       c = 0
    End If
    Exit Sub
 err_msg:
    'MsgBox(Err.Number & "    " & Err.Description)
    Resume Next
```

3. On Error GoTo 0

The On Error GoTo 0 statement disables any error handler in the current procedure.
If you do not include an On Error GoTo 0 statement, the error handler is still disabled when the procedure containing the exception handler ends.
The following example illustrate On Error GoTo 0

Example:

```
 Dim n1, n2, c As Integer
 Button1_Click()
    On Error GoTo err_msg1
    n1 = TextBox1.Text
    n2 = TextBox2.Text
    c = n1 / n2
    MsgBox("Division:" & c)
    If  n2 = 0 Then
       c = 0
       End If
       Exit Sub
err_msg2:
       On Error GoTo 0
       MsgBox("Program Completed")
       Exit Sub
err_msg1:
       MsgBox(Err.Number & "    " & Err.Description)
       Resume  err_msg2
        End Sub
```