VB.NET FORMS AND USER INTERFACE

By Lecturer Suraj Pandey CCT College

Visual Studio IDE (Integrated Development Environment):

- Visual Studio is a powerful and customizable programming environment that contains all the tools we need to build programs quickly and efficiently. It offers a set of tools that help us write and modify the code for our programs, and also detect and correct errors in our programs. Visual Basic.NET IDE is built out of a collection of different windows. Some windows are used for writing code, some for designing interfaces, and others for getting a general overview of files or classes in our application. Visual Studio organizes our work in projects and solutions. A solution can contain more than one project, such as a DLL (Dynamic Link Libraries) and an executable that references that DLL.

- IDE Contents: The Visual Studio IDE consists of several sections, or tools, that the developer uses while programming. As we view the IDE for a new project we generally have three sections in view:
- The Toolbox on the left
- The Solution Explorer on the right
- The Code / Design view in the middle

- 1. Toolbox:
- The Toolbox is a palette of developer objects, or controls, that are placed on forms or web pages, and then code is added to allow the user to interact with them. An example would be TextBox, Button and ListBox controls. With these three controls added to a Windows Form object the developer could write code that would take text.

- 2. Solution Explorer: This is a section that is used to view and modify the contents of the project. A Visual Studio Windows Application Project will generally have a Form object with a code page, references to System components and possibly other modules with special code that is used by the application.
- 3. Properties Windows: The properties windows shows all the control (like TextBox) properties to be changed at design time. Most of these properties can be also changed with code at run time, but basically most properties change the way the control is displayed on your application.

- 4. Code / Design view: This is where the magic takes place. Forms are designed graphically. In other words, the developer has a form on the screen that can be sized and modified to look the way it will be displayed to the application users. Controls are added to the form from the Toolbox, the color and caption can be changed along with many other items.
- This center window of the IDE is also where developers write the code that makes everything in the application work. The code is written in modules, or files, that are either connected to an object (Forms) or called specifically when needed.

- 5. Object Browser: By pressing F2 or selecting it into the View menu, it's possible to explore all the available objects of the libraries (types, functions…)

Creating MDI (Multiple Document Interface) Application:

- A Multiple Document Interface (MDI) programs can display multiple child windows inside them. This is in contrast to single document interface (SDI) applications, which can manipulate only one document at a time. Visual Studio Environment is an example of Multiple Document Interface (MDI) and notepad is an example of an SDI application, opening a document closes any previously opened document. Any windows can become an MDI parent, if you set the IsMdiContainer property to True.
- IsMdiContainer = True

- The following vb.net program shows a MDI form with two child forms. Create a new VB.Net project, then you will get a default form Form1. Then add two more forms in the project (Form2, Form 3). Create a Menu on your form and call these two forms on menu click event. If we want the MDI parent to auto-size the child form we can code like this.

```
form.MdiParent = Me
form.Dock = DockStyle.Fill
form.Show()
```

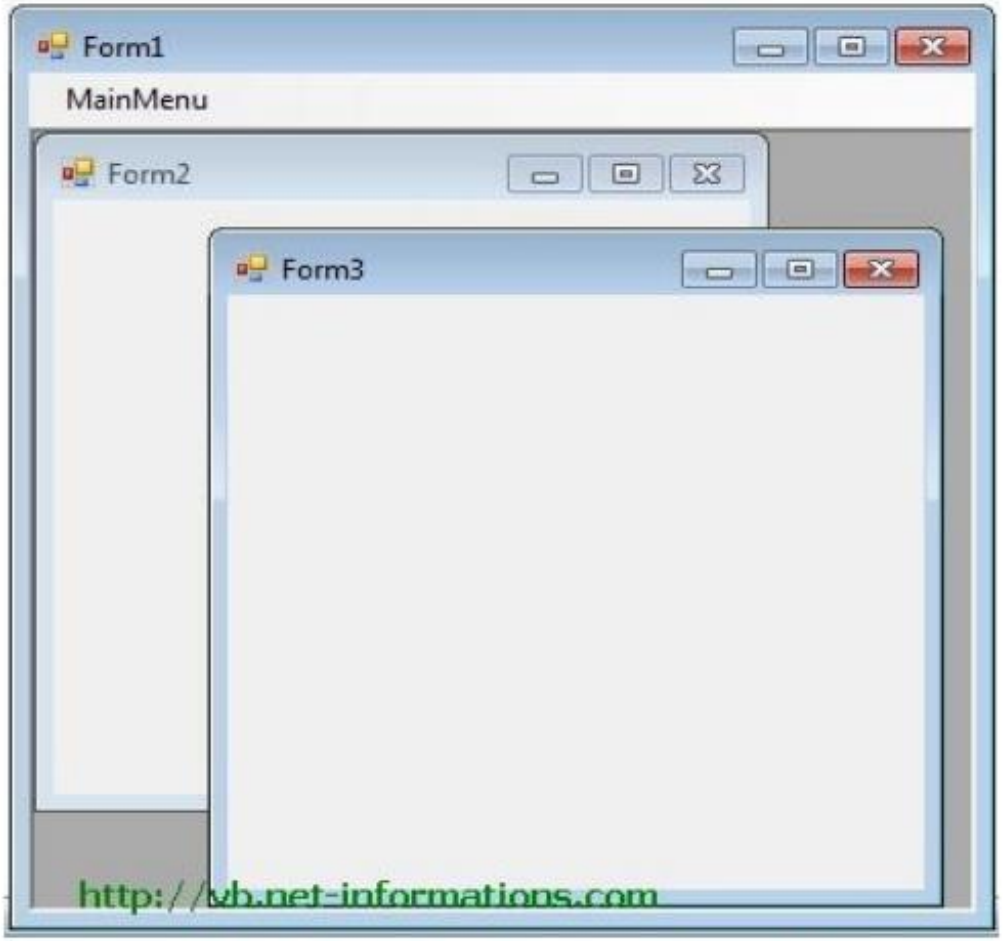


Fig: Multiple Document Interface

Example:

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
        IsMdiContainer = True
    End Sub

    Private Sub MenuItem1ToolStripMenuItem_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles MenuItem1ToolStripMenuItem.Click
            Dim frm2 As New Form2
            frm2.Show()
            frm2.MdiParent = Me
    End Sub

    Private Sub MenuItem2ToolStripMenuItem_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles MenuItem2ToolStripMenuItem.Click
            Dim frm3 As New Form3
            frm3.Show()
            frm3.MdiParent = Me
    End Sub
End Class
```

Showing and hiding forms

- Showing and hiding controls and forms is easy just use the control's or form's visible property.
- Setting this property to TRUE displays the control or form and if it is FALSE then it hides.
- For example
- Private sub button1_click(ByVal sender As System.Object._ByVal e As System.EventArgs)Handles Button1.click

```
Button1.Visible=false
End Sub
```

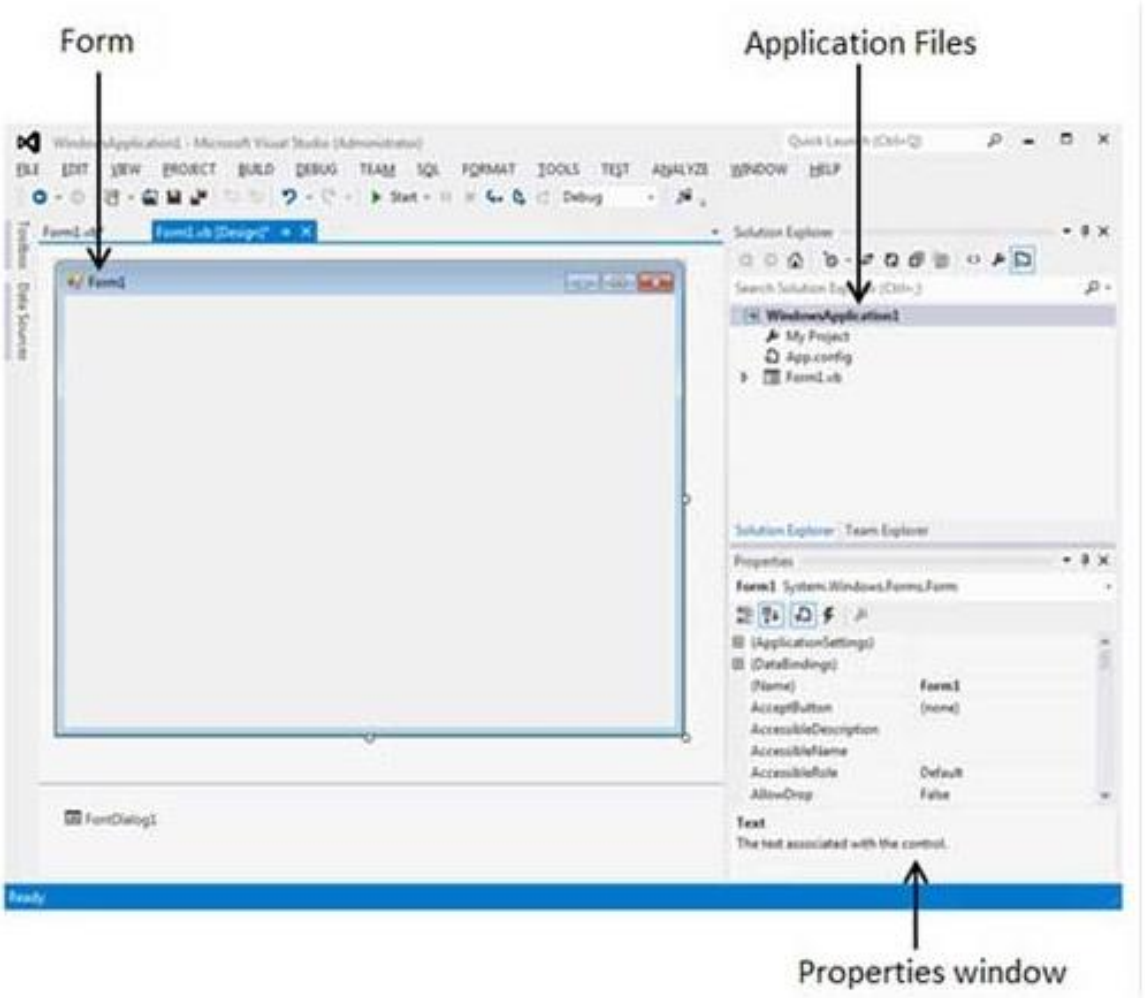
- We can also use the Show and Hide methods of controls and forms to show and hide them.
- For example, when the user clicks Button1, we can hide Form2 and show Form3 this way
- Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
- Form2.Hide()
- Form3.Show()
- End Sub

Basic Controls

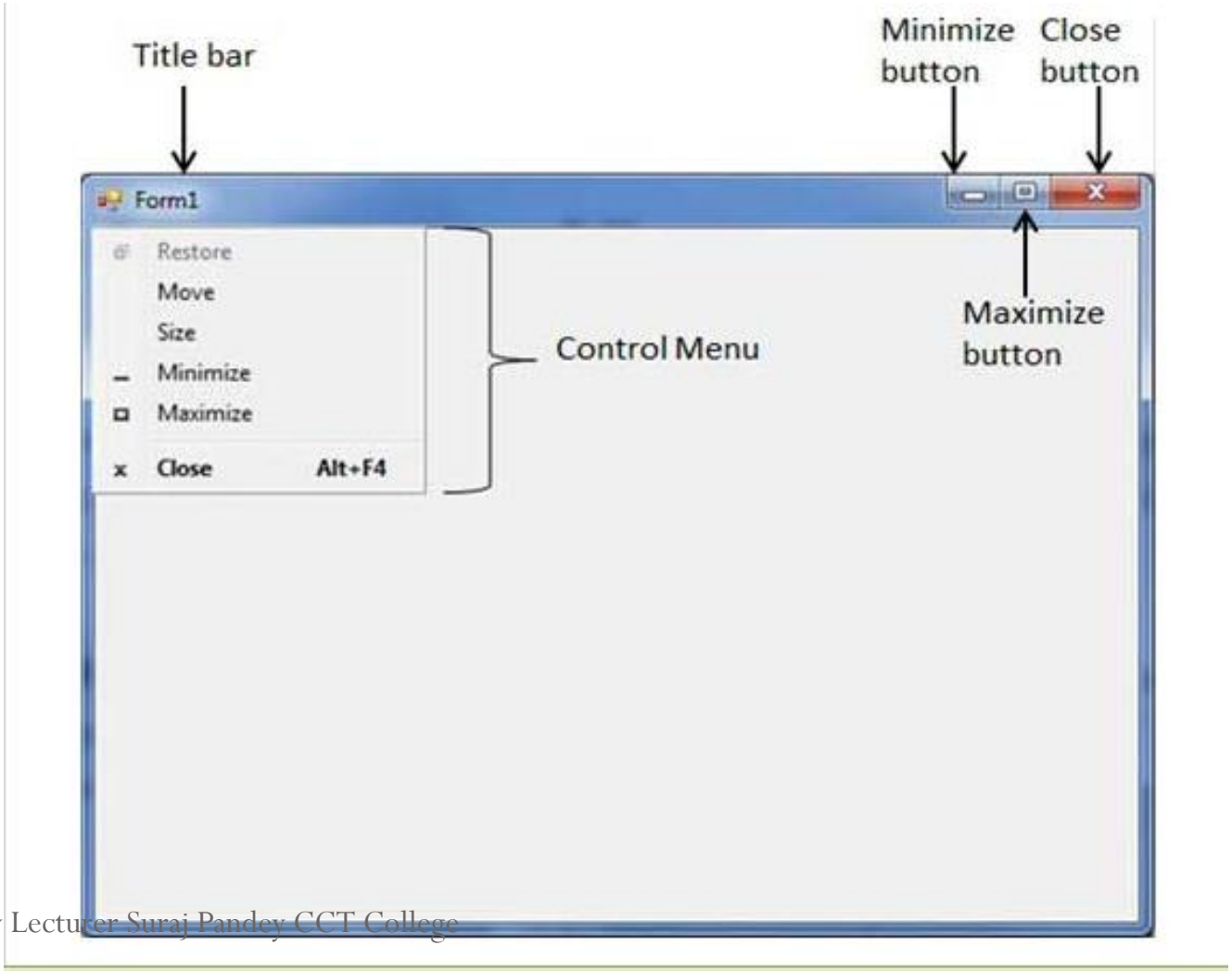
- VB.Net provides a huge variety of controls that help you to create rich user interface. Functionalities of all these controls are defined in the respective control classes. The control classes are defined in the **System.Windows.Forms** namespace.
- The following table lists some of the commonly used controls

S.N.	Widget & Description
1	Forms The container for all the controls that make up the user interface.
2	TextBox It represents a Windows text box control.
3	Label It represents a standard Windows label.
4	Button It represents a Windows button control.
5	ListBox It represents a Windows control to display a list of items.
6	ComboBox It represents a Windows combo box control.
7	RadioButton It enables the user to select a single option from a group of choices when paired with other RadioButton controls.
8	CheckBox It represents a Windows CheckBox.
9	PictureBox It represents a Windows picture box control for displaying an image.
10	ProgressBar It represents a Windows progress bar control.
11	ScrollBar It Implements the basic functionality of a scroll bar control.
12	DateTimePicker 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.
13	TreeView It displays a hierarchical collection of labeled items, each represented by a TreeNode.
14	ListView It represents a Windows list view control, which displays a collection of items that can be displayed using one of four different views.

- Let's start with creating a Window Forms Application by following the following steps in Microsoft Visual Studio: **File -> New Project -> Windows Forms Applications**
- Finally, select OK, Microsoft Visual Studio creates your project and displays following window Form with a name **Form1**.


Form
Application Files
Properties window

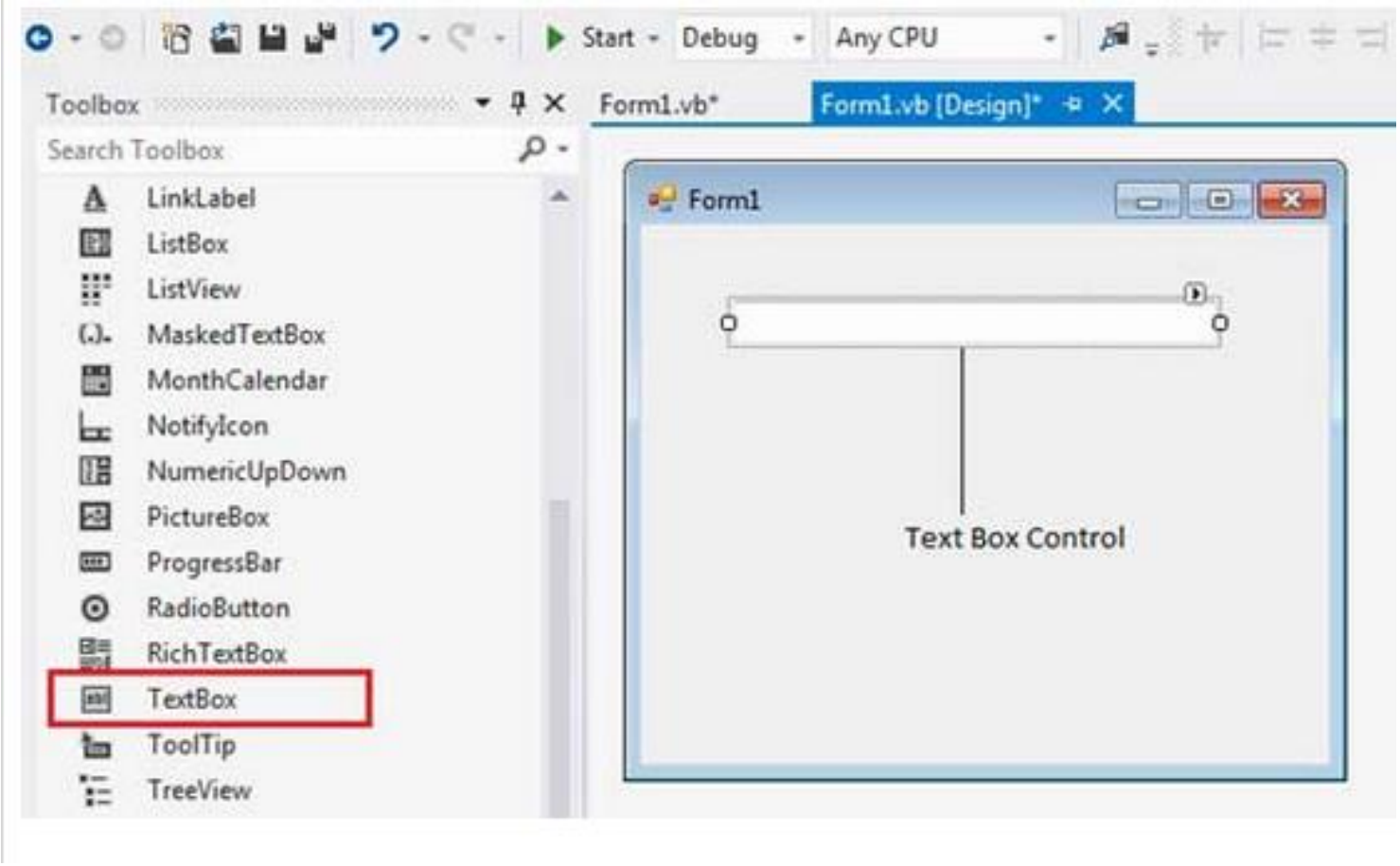
- Visual Basic Form is the container for all the controls that make up the user interface. Every window you see in a running visual basic application is a form, thus the terms form and window describe the same entity. Visual Studio creates a default form for you when you create a **Windows Forms Application**.
- Every form will have title bar on which the form's caption is displayed and there will be buttons to close, maximize and minimize the form shown below:


Title bar
Minimize button
Close button
Form1
Restore
Move
Size
Minimize
Maximize
Close Alt+F4
Control Menu
Maximize button

- If you click the icon on the top left corner, it opens the control menu, which contains the various commands to control the form like to move control from one place to another place, to maximize or minimize the form or to close the form.

VB.Net - TextBox Control

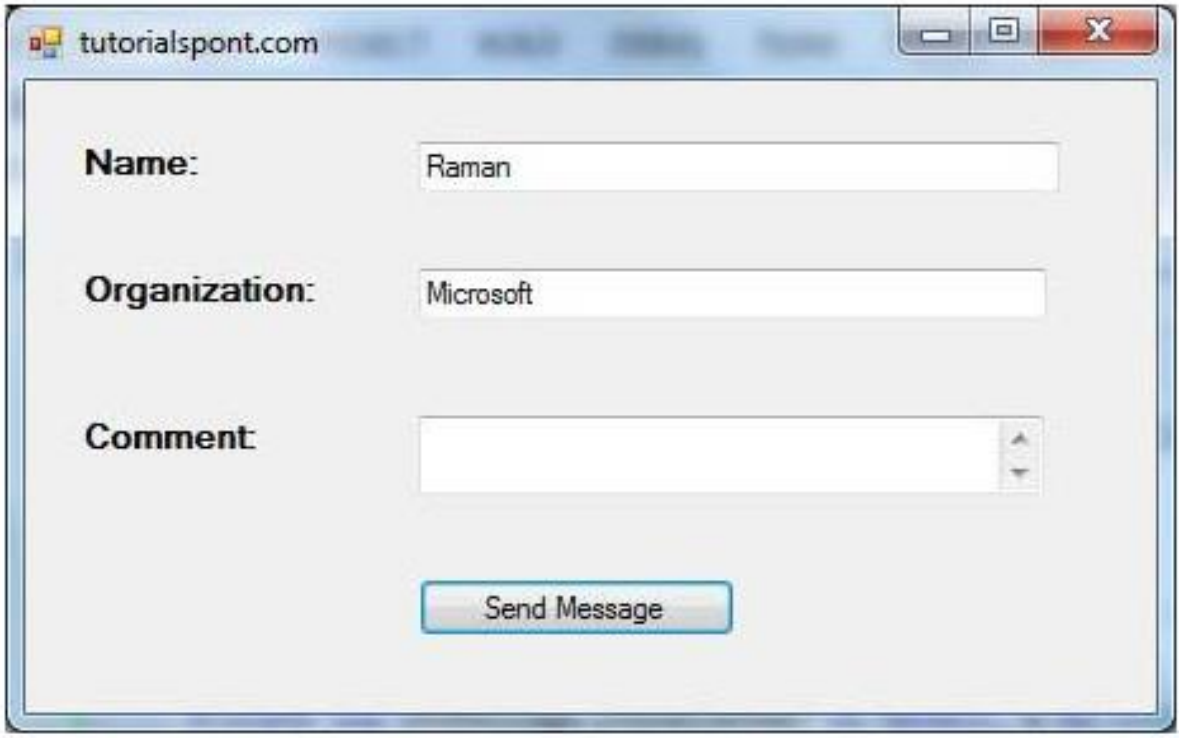
- Text box controls allow entering text on a form at runtime. By default, it takes a single line of text, however, you can make it accept multiple texts and even add scroll bars to it.
- Let's create a text box by dragging a Text Box control from the Toolbox and dropping it on the form.


Start
Debug
Any CPU
Toolbox
Search Toolbox
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
Form1.vb*
Form1.vb [Design]*
Form1
Text Box Control

- **Example**
- In this example, we create three text boxes and use the Click event of a button to display the entered text using a message box. Take the following steps:

1. Drag and drop three Label controls and three TextBox controls on the form.
2. Change the texts on the labels to: Name, Organization and Comments, respectively.
3. Change the names of the text boxes to txtName, txtOrg and txtComment, respectively.
4. Drag and drop a button control on the form. Set its name to btnMessage and its text property to 'Send Message'.
5. Click the button to add the Click event in the code window and add the following code.

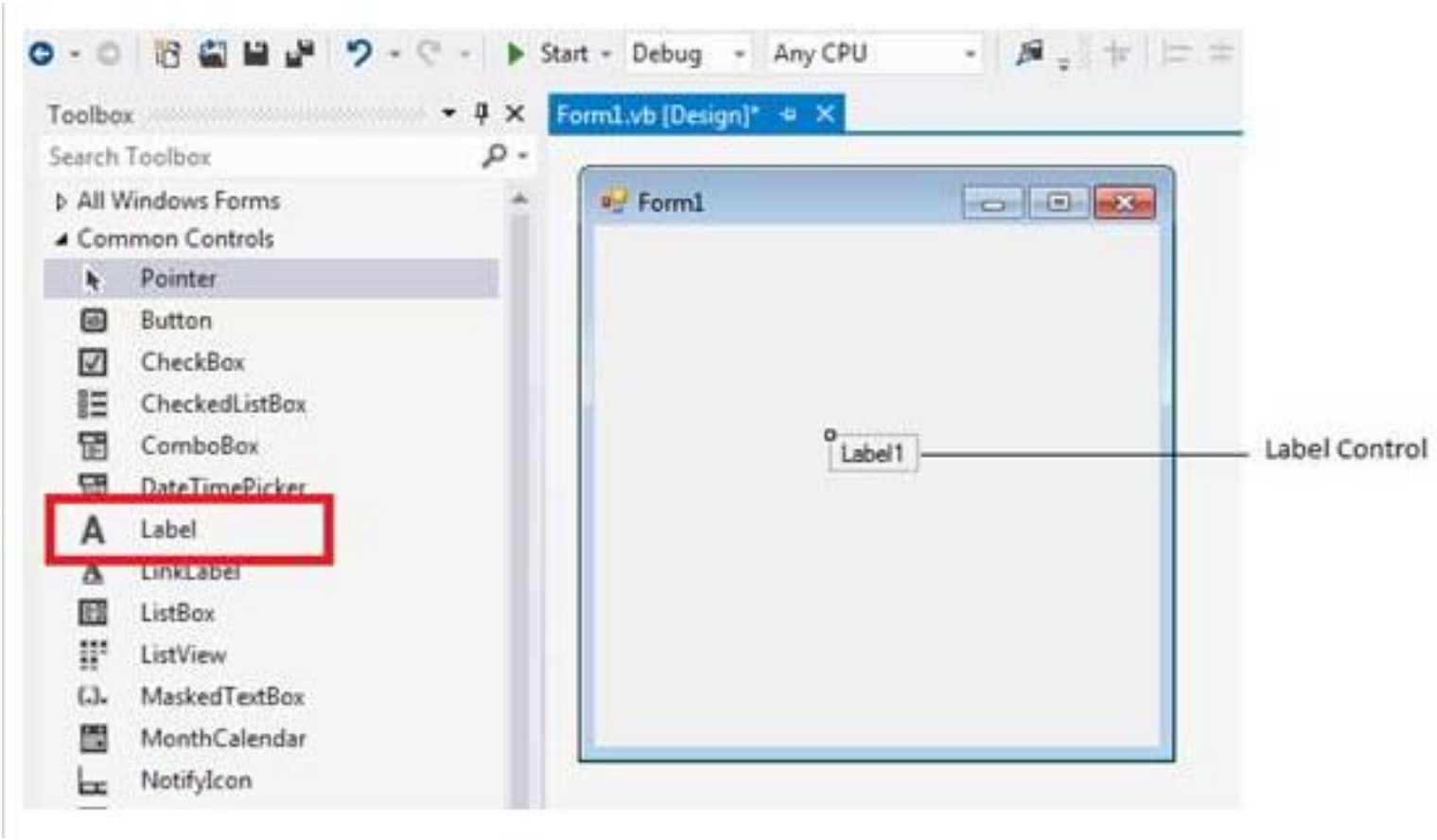
```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) _
Handles MyBase.Load
        ' Set the caption bar text of the form.
        Me.Text = "cct.com"
    End Sub
    Private Sub btnMessage_Click(sender As Object, e As EventArgs) _
Handles btnMessage.Click
        MessageBox.Show("Thank you " + txtName.Text + " from " +
txtOrg.Text)
    End Sub
End Class
```



the Send Message button would show the following message box:

VB.Net - Label Control

- The Label control represents a standard Windows label. It is generally used to display some informative text on the GUI which is not changed during runtime.
- Let's create a label by dragging a Label control from the Toolbox and dropping it on the form.


Start
Debug
Any CPU
Toolbox
Search Toolbox
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
Form1.vb [Design]*
Form1
Label1
Label Control

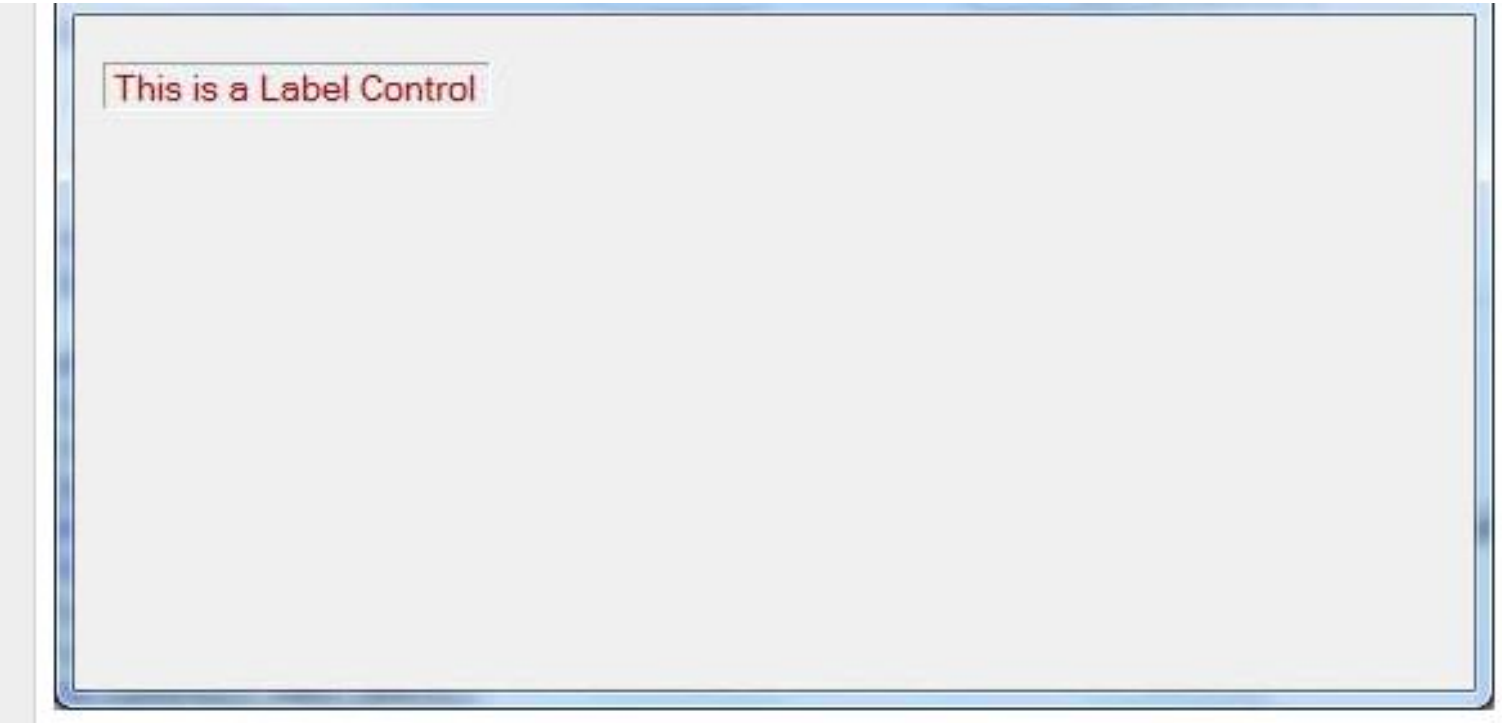
- **Example**
- Following is an example, which shows how we can create two labels. Let us create the first label from the designer view tab and set its properties from the properties window. We will use the Click and the DoubleClick events of the label to move the first label and change its text and create the second label and add it to the form, respectively.

  Take the following steps:

1. Drag and drop a Label control on the form.
2. Set the Text property to provide the caption "This is a Label Control".
3. Set the Font property from the properties window.
4. Click the label to add the Click event in the code window and add the following codes.

- Public Class Form1 Private Sub Form1_Load(sender As Object, e As EventArgs) _ Handles MyBase.Load

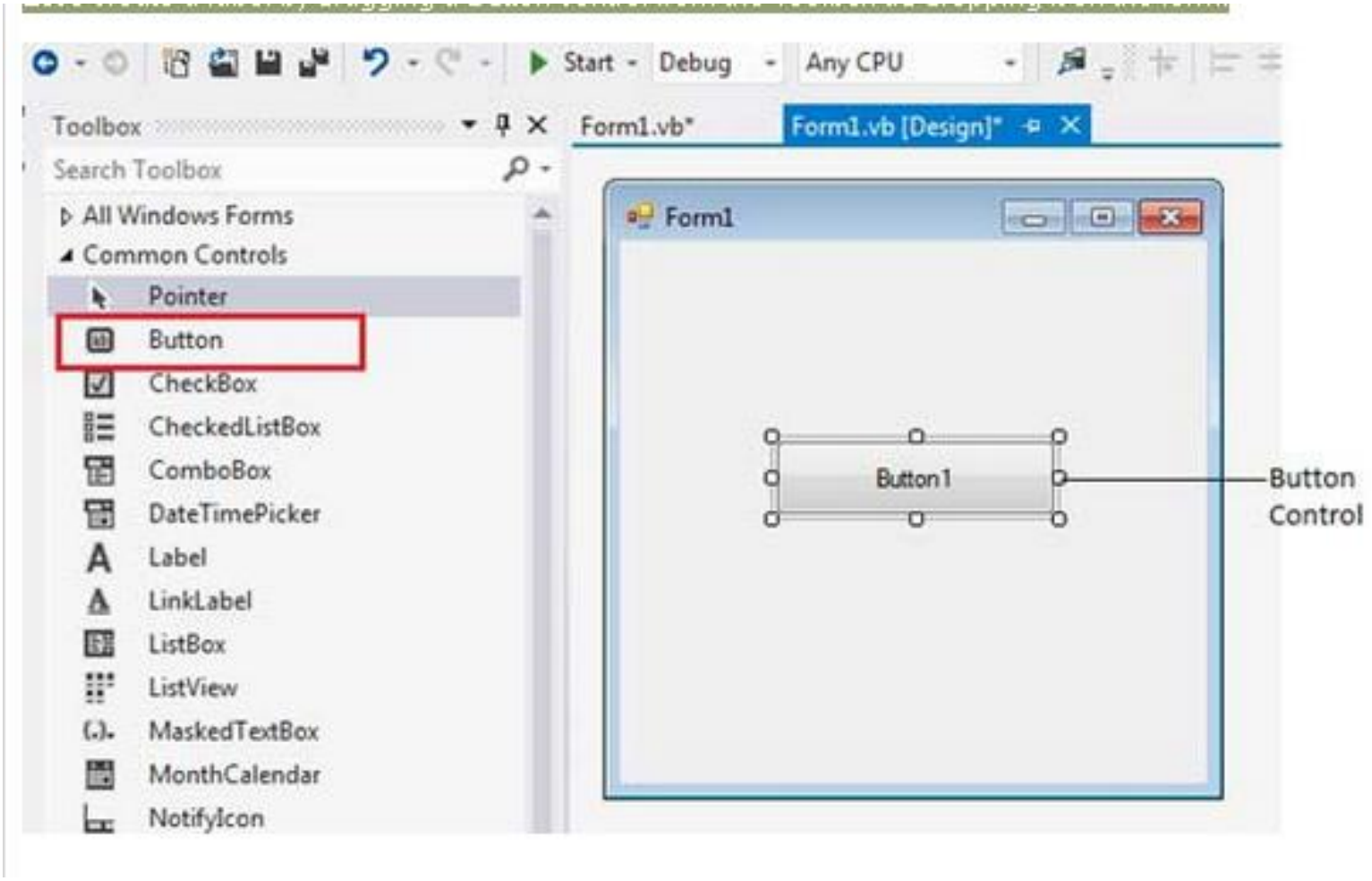
```
' Create two buttons to use as the accept and cancel buttons.
' Set window width and height
        Me.Height = 300
        Me.Width = 560
' Set the caption bar text of the form.
        Me.Text = “CCT"
' Display a help button on the form.
        Me.HelpButton = True
End Sub
Private Sub Label1_Click(sender As Object, e As EventArgs) _ Handles Label1.Click
        Label1.Location = New Point(50, 50)
        Label1.Text = "You have just moved the label"
End Sub
Private Sub Label1_DoubleClick(sender As Object, e As EventArgs) Handles Label1.DoubleClick
        Dim Label2 As New Label
        Label2.Text = "New Label"
        Label2.Location = New Point(Label1.Left, Label1.Height + _ Label1.Top + 25)
        Me.Controls.Add(Label2)
End Sub
End Class
```


This is a Label Control

Clicking and double clicking the label would produce the following effect:

VB.Net - Button Control

- The Button control represents a standard Windows button. It is generally used to generate a Click event by providing a handler for the Click event.
- Let's create a label by dragging a Button control from the Toolbox ad dropping it on the form.


Start
Debug
Any CPU
Toolbox
Search Toolbox
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
Form1.vb*
Form1.vb [Design]*
Form1
Button1
Button
Control

- **Example**

  In the following example, we will create three buttons:

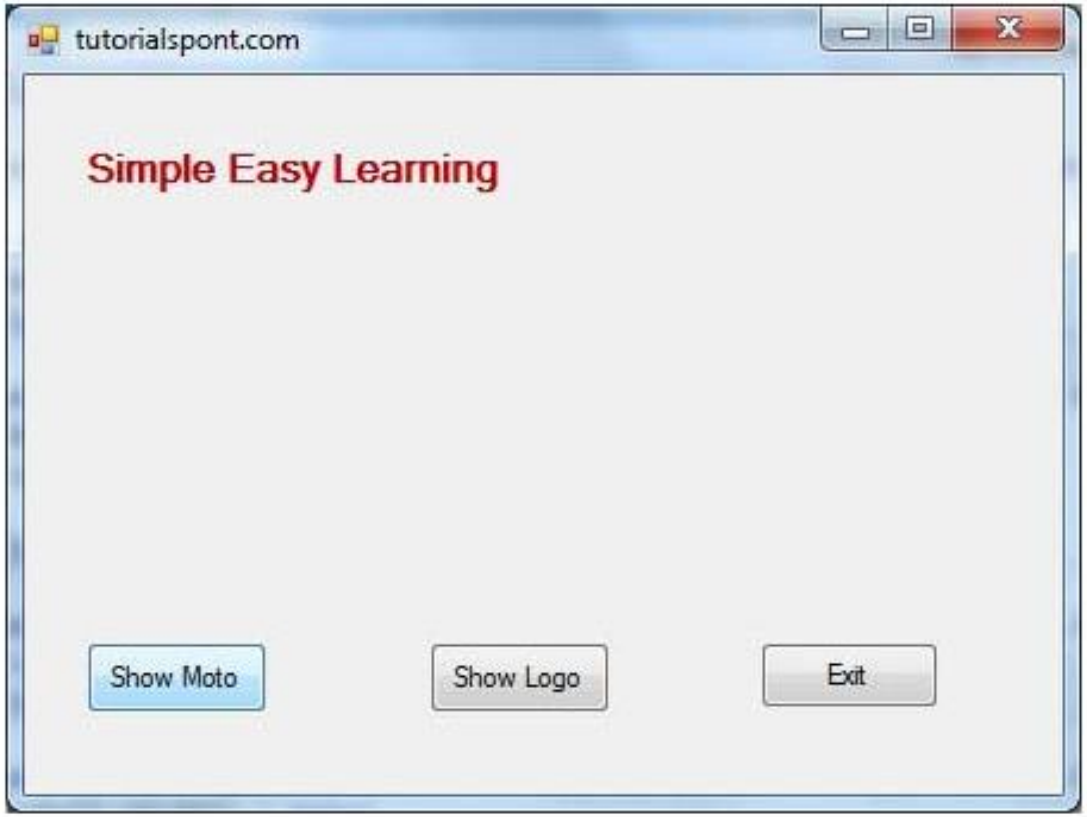
1. Set captions for the buttons
2. Set some image for the button
3. Handle the click events of each buttons
4. Take following steps:
5. Drag and drop a Label control on the form.
6. Set the Text property to provide the caption “cct".
7. Drag and drop three buttons on the form.
8. Using the properties window, change the Name properties of the buttons to btnMoto, btnLogo and btnExit respectively.
9. Using the properties window, change the Text properties of the buttons to Show Moto, Show Logo and Exit respectively.
10. Drag and Drop another button, using the properties window, set its Image property and name it btnImage.
11. At this stage, the form looks like:


TP
SIMPLY EASY LEARNING
Show Moto
Show Logo
Exit

- Public Class Form1 Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

```
' Set the caption bar text of the form.
        Me.Text = “cct“
        btnImage.Visible = False
End Sub
Private Sub btnMoto_Click(sender As Object, e As EventArgs) Handles btnMoto.Click
        btnImage.Visible = False
        Label1.Text = "Simple Easy Learning"
End Sub
Private Sub btnExit_Click(sender As Object, e As EventArgs) Handles btnExit.Click
        Application.Exit()
End Sub
Private Sub btnLogo_Click(sender As Object, e As EventArgs) Handles btnLogo.Click
        Label1.Visible = False
        btnImage.Visible =True
End Sub
End Class
```

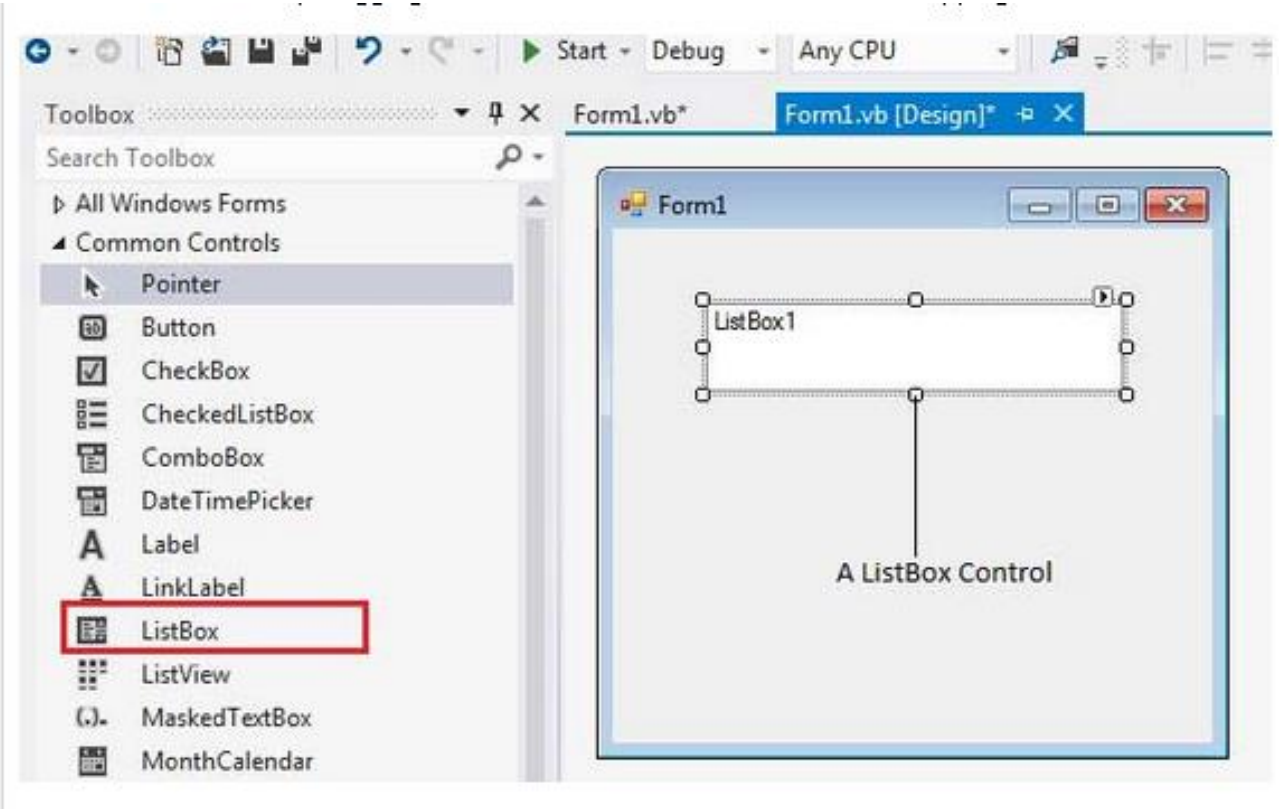
Clicking the first button, displays:



Clicking the second button displays:

VB.Net - ListBox Control

- The ListBox represents a Windows control to display a list of items to a user. A user can select an item from the list. It allows the programmer to add items at design time by using the properties window or at the runtime.
- Let's create a list box by dragging a ListBox control from the Toolbox and dropping it on the form.


Start
Debug
Any CPU
Toolbox
Search Toolbox
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
Form1.vb*
Form1.vb [Design]*
Form1
ListBox1
A ListBox Control

Example 1

In the following example, let us add a list box at design time and add items on it at runtime.

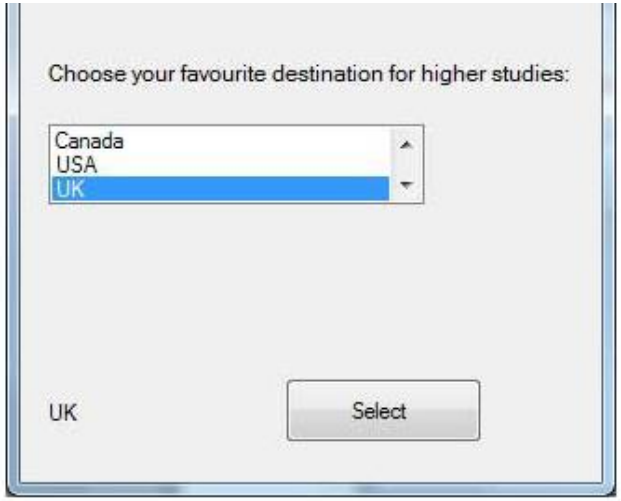
- Take the following steps:
- Drag and drop two labels, a button and a ListBox control on the form.
- Set the Text property of the first label to provide the caption "Choose your favourite destination for higher studies".
- Set the Text property of the second label to provide the caption "Destination". The text on this label will change at runtime when the user selects an item on the list.
- Click the listbox and the button controls to add the following codes in the code editor.

```
Public Class Form1
   Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
      ' Set the caption bar text of the form.
      Me.Text = "tutorialspont.com"
      ListBox1.Items.Add("Canada")
      ListBox1.Items.Add("USA")
      ListBox1.Items.Add("UK")
      ListBox1.Items.Add("Japan")
      ListBox1.Items.Add("Russia")
      ListBox1.Items.Add("China")
      ListBox1.Items.Add("India")
   End Sub

   Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
      MsgBox("You have selected " + ListBox1.SelectedItem.ToString())
   End Sub
   Private Sub ListBox1_SelectedIndexChanged(sender As Object, e As
EventArgs)
        Handles ListBox1.SelectedIndexChanged
      Label2.Text = ListBox1.SelectedItem.ToString()
   End Sub
End Class
```

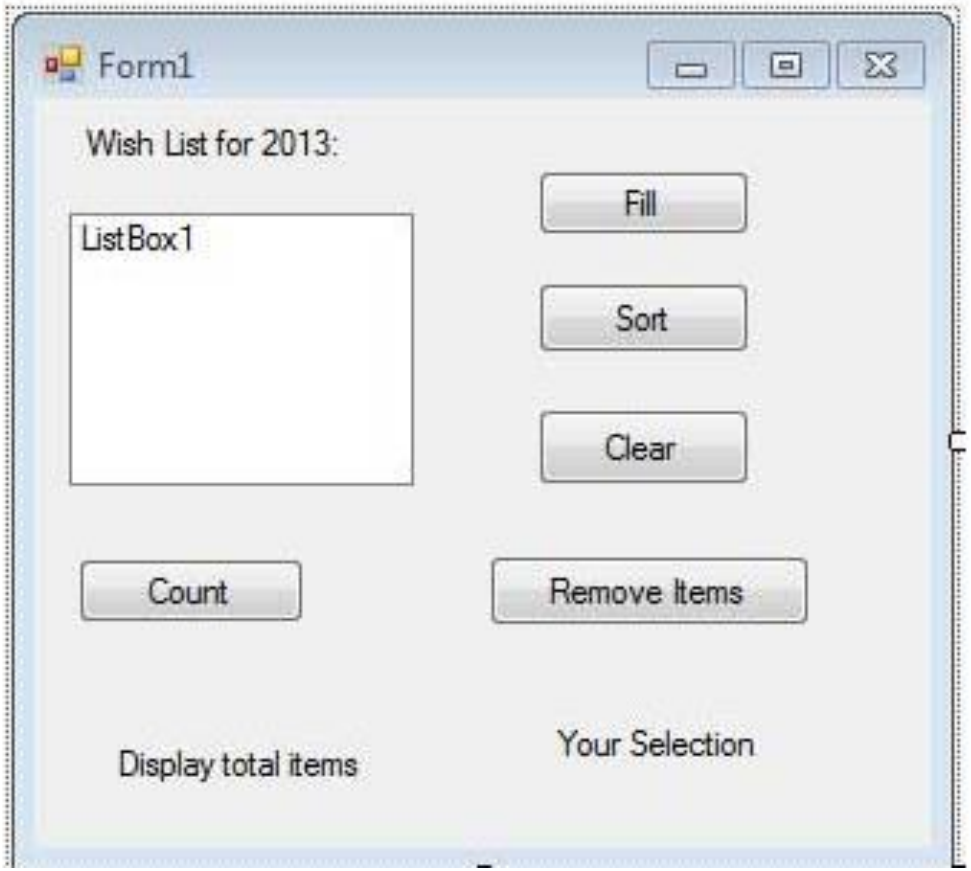
- When the above code is executed and run using Start button available at the Microsoft Visual Studio tool bar, it will show the following window:


Choose your favourite destination for higher studies:
Canada
USA
UK
Destination
Select



Clicking the Select button displays a message box with the user's choice:

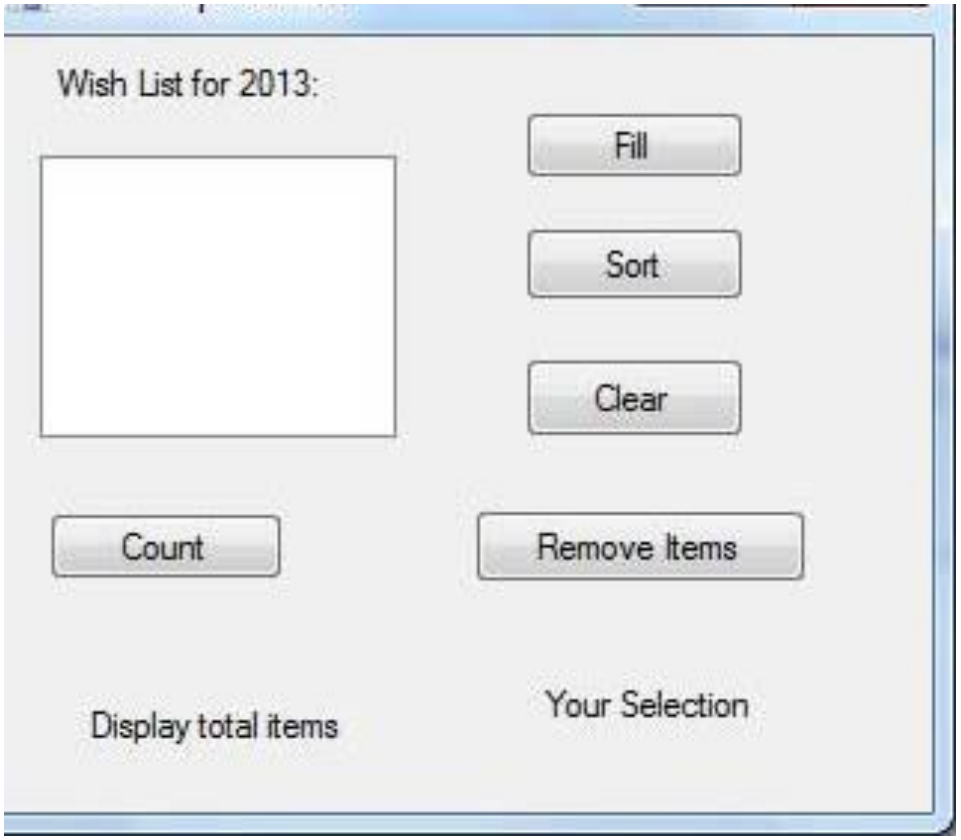
- **Example 2**
- In this example, we will fill up a list box with items, retrieve the total number of items in the list box, sort the list box, remove some items and clear the entire list box.
- Design the Form:


Form1
Wish List for 2013:
ListBox1
Fill
Sort
Clear
Count
Remove Items
Display total items
Your Selection

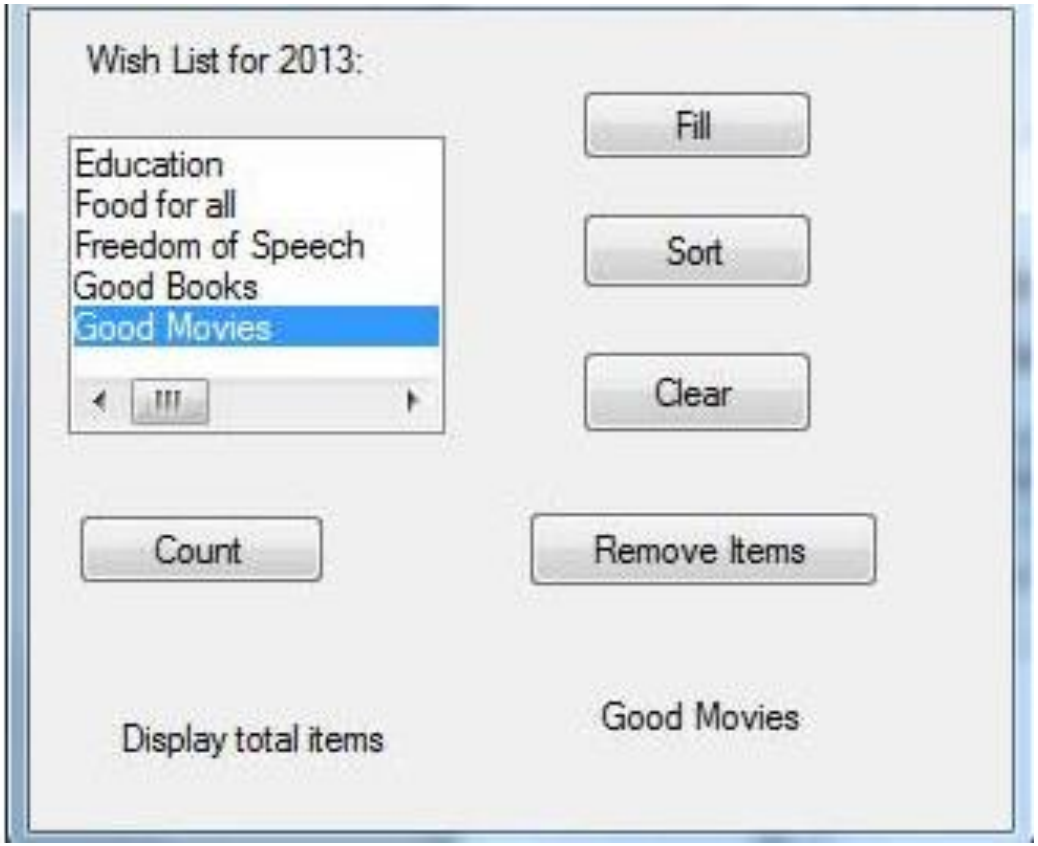
```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' Set the caption bar text of the form.
        Me.Text = "tutorialspont.com"
        ' creating multi-column and multiselect list box
        ListBox1.MultiColumn = True
        ListBox1.SelectionMode = SelectionMode.MultiExtended
    End Sub
    'populates the list
    Private Sub Button1_Click_1(sender As Object, e As EventArgs) _
            Handles Button1.Click
        ListBox1.Items.Add("Safety")
        ListBox1.Items.Add("Security")
        ListBox1.Items.Add("Governance")
        ListBox1.Items.Add("Good Music")
        ListBox1.Items.Add("Good Movies")
        ListBox1.Items.Add("Good Books")
        ListBox1.Items.Add("Education")
        ListBox1.Items.Add("Roads")
        ListBox1.Items.Add("Health")
        ListBox1.Items.Add("Food for all")
        ListBox1.Items.Add("Shelter for all")
        ListBox1.Items.Add("Industrialisation")
        ListBox1.Items.Add("Peace")
        ListBox1.Items.Add("Liberty")
        ListBox1.Items.Add("Freedom of Speech")
    End Sub
```

```
    'sorting the list
    Private Sub Button2_Click(sender As Object, e As EventArgs) _
            Handles Button2.Click
        ListBox1.Sorted = True
    End Sub
    'clears the list
    Private Sub Button3_Click(sender As Object, e As EventArgs) _
            Handles Button3.Click
        ListBox1.Items.Clear()
    End Sub
    'removing the selected item
    Private Sub Button4_Click(sender As Object, e As EventArgs) _
            Handles Button4.Click
        ListBox1.Items.Remove(ListBox1.SelectedItem.ToString)
    End Sub
    'counting the numer of items
    Private Sub Button5_Click(sender As Object, e As EventArgs) _
            Handles Button5.Click
        Label1.Text = ListBox1.Items.Count
    End Sub
    'displaying the selected item on the third label
    Private Sub ListBox1_SelectedIndexChanged(sender As Object, e As EventArgs)
             Handles ListBox1.SelectedIndexChanged
        Label3.Text = ListBox1.SelectedItem.ToString()
    End Sub
End Class
```

- When the above code is executed and run using Start button available at the Microsoft Visual Studio tool bar, it will show the following window:

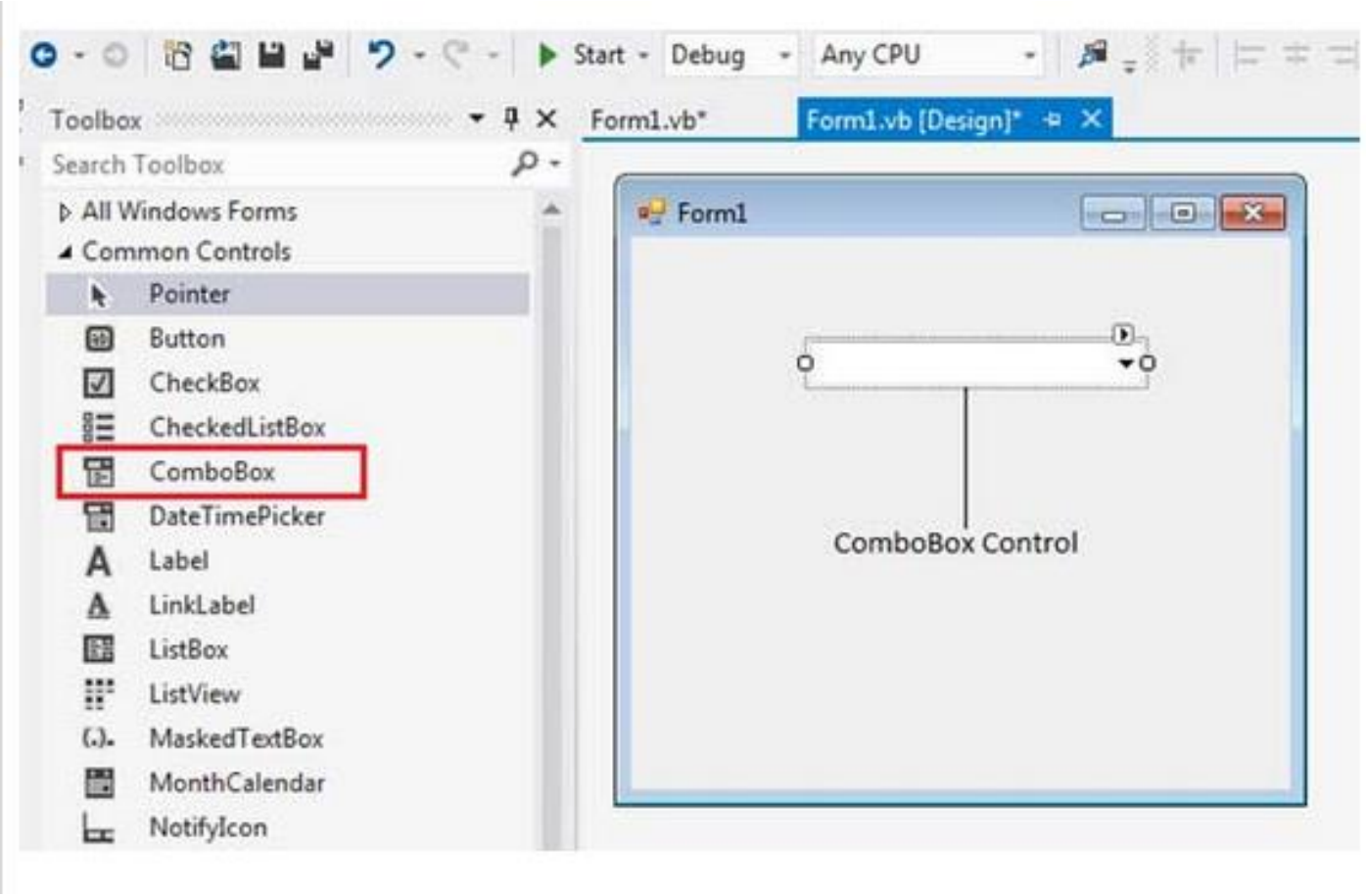

Wish List for 2013:
Fill
Sort
Clear
Count
Remove Items
Display total items
Your Selection

- Fill the list and check workings of other buttons:


Wish List for 2013:
Education
Food for all
Freedom of Speech
Good Books
Good Movies
Fill
Sort
Clear
Count
Remove Items
Display total items
Good Movies

ComboBox Control

- The ComboBox control is used to display a drop-down list of various items. It is a combination of a text box in which the user enters an item and a drop-down list from which the user selects an item.
- Let's create a combo box by dragging a ComboBox control from the Toolbox and dropping it on the form.

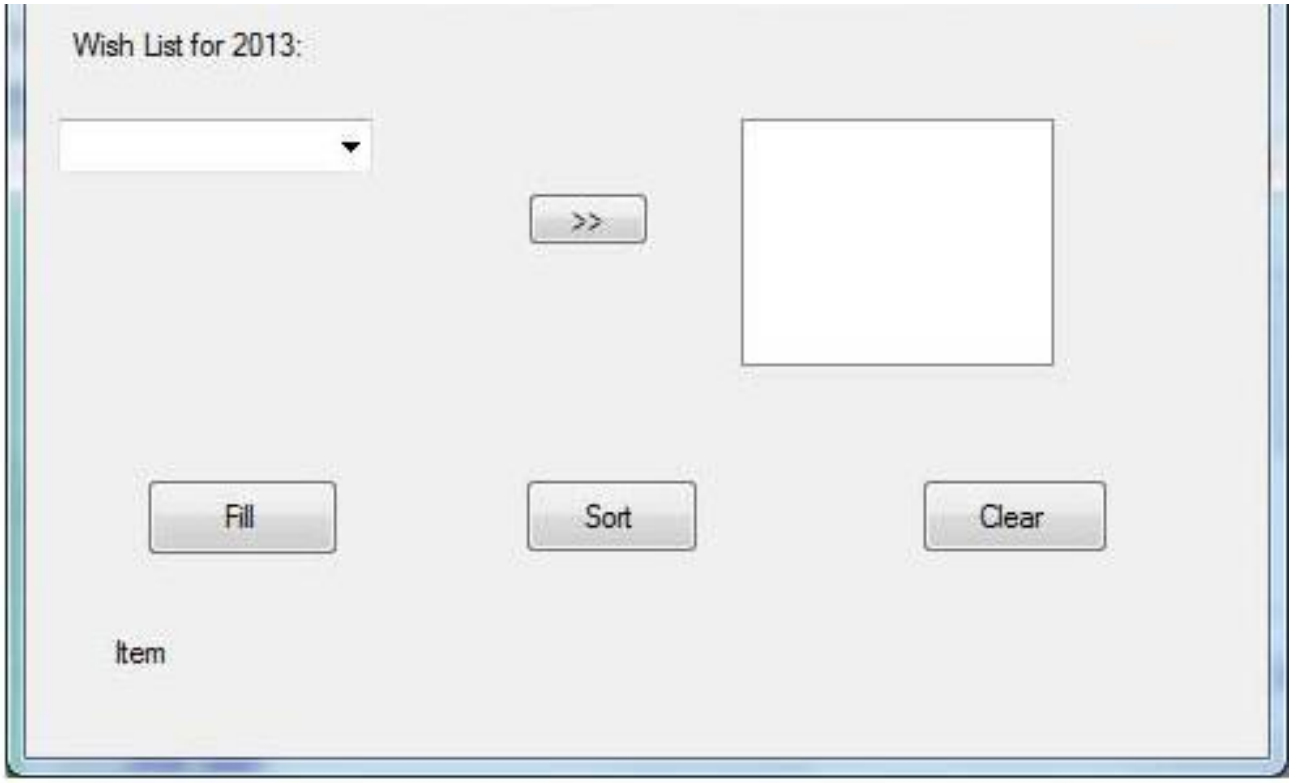

Start
Debug
Any CPU
Toolbox
Search Toolbox
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
Form1.vb*
Form1.vb [Design]*
Form1
ComboBox Control

- **Example**
- In this example, let us fill a combo box with various items, get the selected items in the combo box and show them in a list box and sort the items.
- Drag and drop a combo box to store the items, a list box to display the selected items, four button controls to add to the list box with selected items, to fill the combo box, to sort the items and to clear the combo box list, respectively.
- Add a label control that would display the selected item.

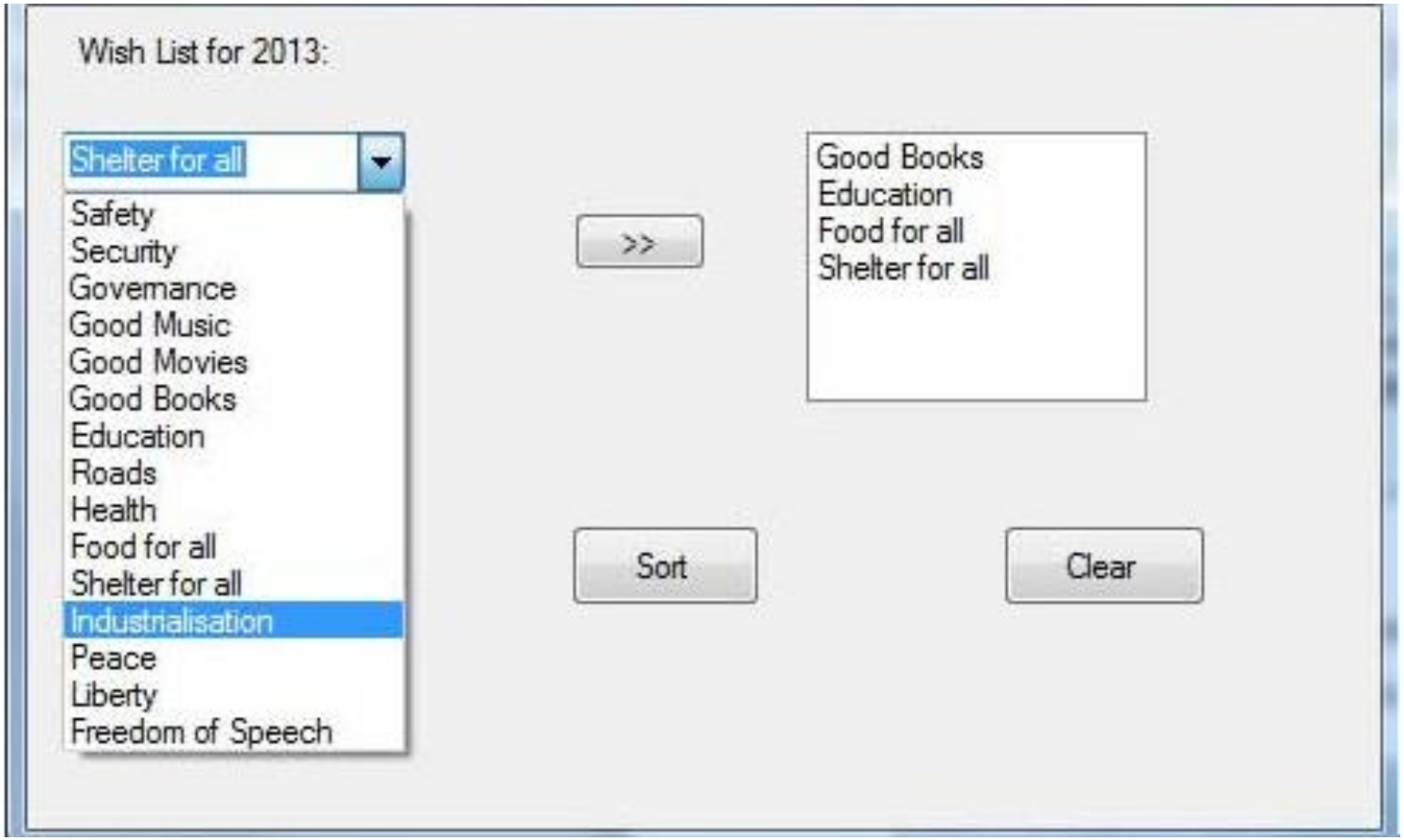
```
Public Class Form1
   Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
       ' Set the caption bar text of the form.
       Me.Text = "cct.com"
   End Sub
   'sends the selected items to the list box
   Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
       If ComboBox1.SelectedIndex > -1 Then
           Dim sindex As Integer
           sindex = ComboBox1.SelectedIndex
           Dim sitem As Object
           sitem = ComboBox1.SelectedItem
           ListBox1.Items.Add(sitem)
       End If
   End Sub
   'populates the list
   Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
       ComboBox1.Items.Clear()
       ComboBox1.Items.Add("Safety")
       ComboBox1.Items.Add("Security")
       ComboBox1.Items.Add("Governance")
       ComboBox1.Items.Add("Good Music")
       ComboBox1.Items.Add("Good Movies")
       ComboBox1.Items.Add("Good Books")
       ComboBox1.Items.Add("Education")
       ComboBox1.Items.Add("Roads")
       ComboBox1.Items.Add("Health")
       ComboBox1.Items.Add("Food for all")
       ComboBox1.Items.Add("Shelter for all")
       ComboBox1.Items.Add("Industrialisation")
       ComboBox1.Items.Add("Peace")
       ComboBox1.Items.Add("Liberty")
       ComboBox1.Items.Add("Freedom of Speech")
       ComboBox1.Text = "Select from..."
   End Sub
```

```
    'sorting the list
    Private Sub Button3_Click(sender As Object, e As EventArgs)
        ComboBox1.Sorted = True
    End Sub
    'clears the list
    Private Sub Button4_Click(sender As Object, e As EventArgs)
        ComboBox1.Items.Clear()
    End Sub
    'displaying the selected item on the label
    Private Sub ComboBox1_SelectedIndexChanged(sender As Object, e As
EventArgs) _
        Handles ListBox1.SelectedIndexChanged
        Label1.Text = ComboBox1.SelectedItem.ToString()
    End Sub
End Class
```

When the above code is executed and run using **Start** button available at the Microsoft Visual Studio tool bar, it will show the following window:

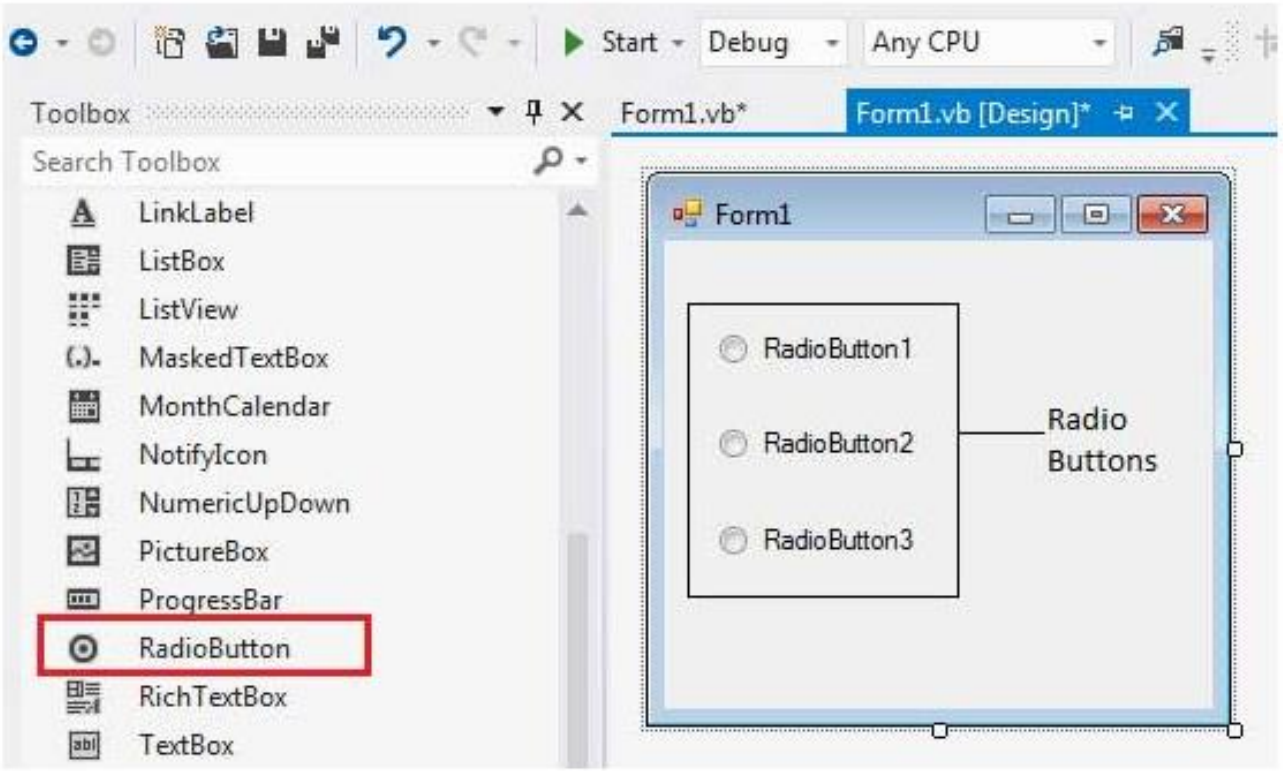


Click on various buttons to check the actions performed by each:

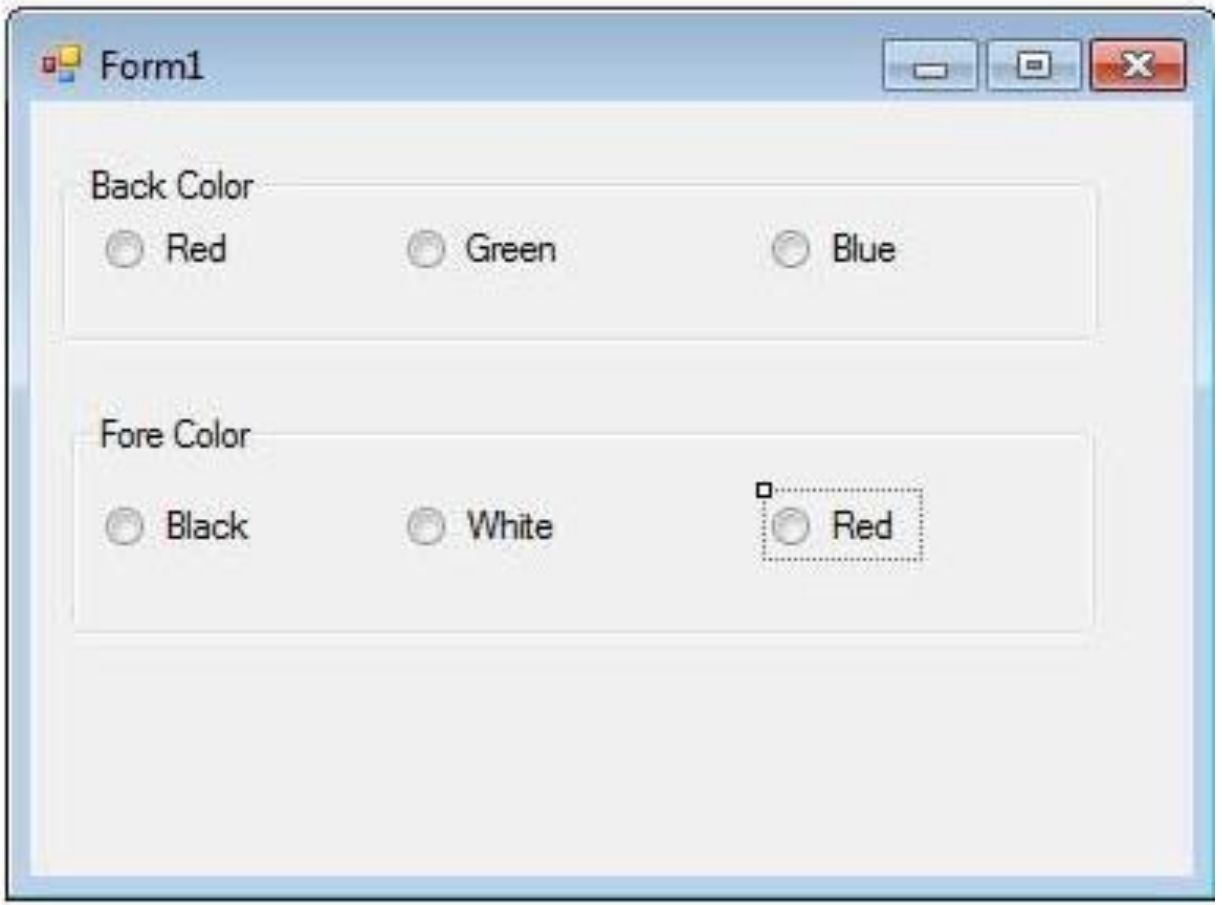


RadioButton Control

- The RadioButton control is used to provide a set of mutually exclusive options. The user can select one radio button in a group. If you need to place more than one group of radio buttons in the same form, you should place them in different container controls like a GroupBox control.
- Let's create three radio buttons by dragging RadioButton controls from the Toolbox and dropping on the form.


Start
Debug
Any CPU
Toolbox
Search Toolbox
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
Form1.vb*
Form1.vb [Design]*
Form1
RadioButton1
RadioButton2
RadioButton3
Radio Buttons

- **Example**
- In the following example, let us create two groups of radio buttons and use their CheckedChanged events for changing the BackColor and ForeColor property of the form.
- Let's double click on the radio buttons and put the follow code in the opened window.


Form1
Back Color
Red
Green
Blue
Fore Color
Black
White
Red

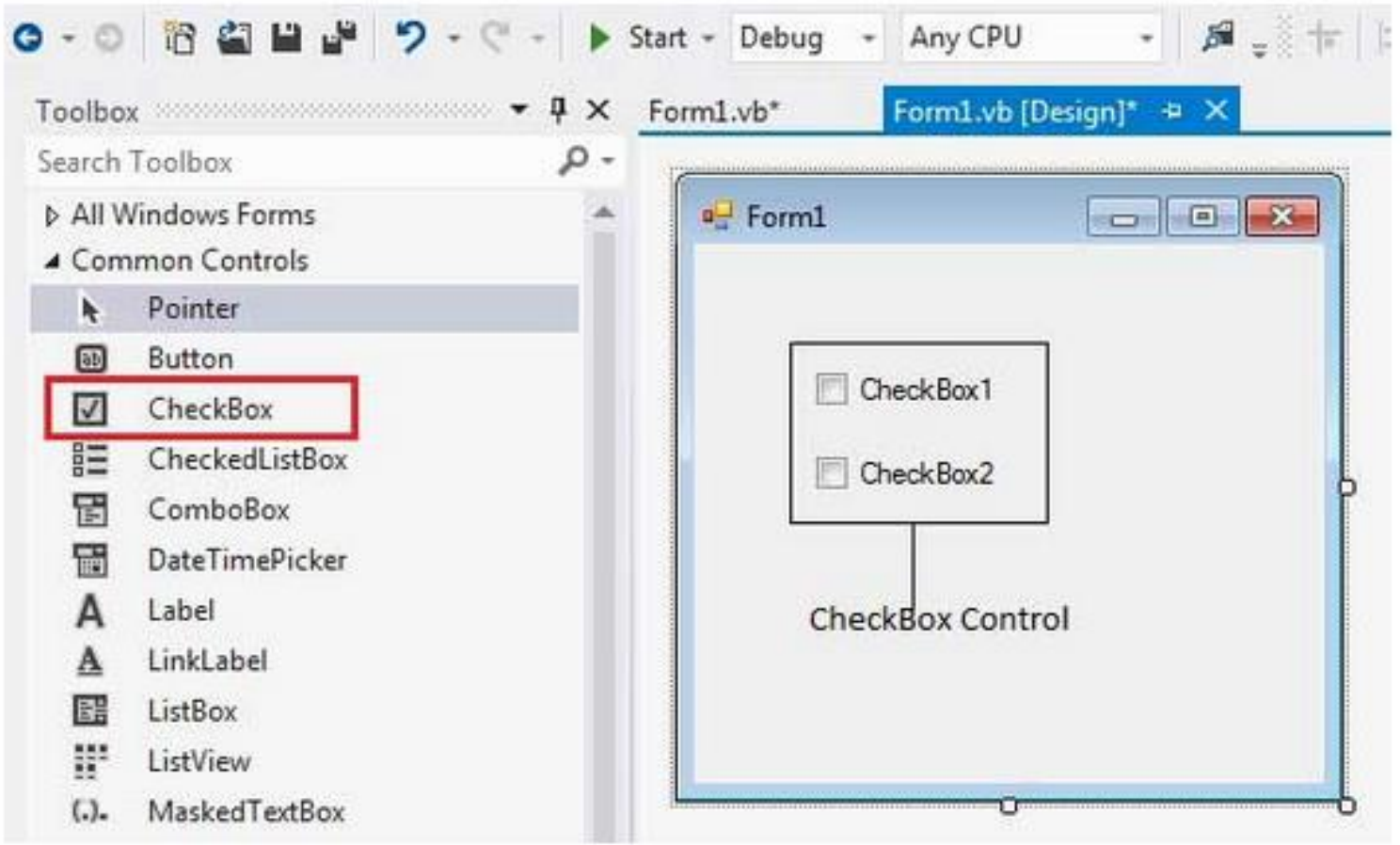
```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        ' Set the caption bar text of the form.
        Me.Text = "cct.com"
    End Sub

    Private Sub RadioButton1_CheckedChanged(sender As Object, _
        e As EventArgs) Handles RadioButton1.CheckedChanged
        Me.BackColor = Color.Red
    End Sub
    Private Sub RadioButton2_CheckedChanged(sender As Object, _
        e As EventArgs) Handles RadioButton2.CheckedChanged
        Me.BackColor = Color.Green
    End Sub
    Private Sub RadioButton3_CheckedChanged(sender As Object, _
        e As EventArgs) Handles RadioButton3.CheckedChanged
        Me.BackColor = Color.Blue
    End Sub

    Private Sub RadioButton4_CheckedChanged(sender As Object, _
        e As EventArgs) Handles RadioButton4.CheckedChanged
        Me.ForeColor = Color.Black
    End Sub
    Private Sub RadioButton5_CheckedChanged(sender As Object, _
        e As EventArgs) Handles RadioButton5.CheckedChanged
        Me.ForeColor = Color.White
    End Sub
    Private Sub RadioButton6_CheckedChanged(sender As Object, _
        e As EventArgs) Handles RadioButton6.CheckedChanged
        Me.ForeColor = Color.Red
    End Sub
End Class
```

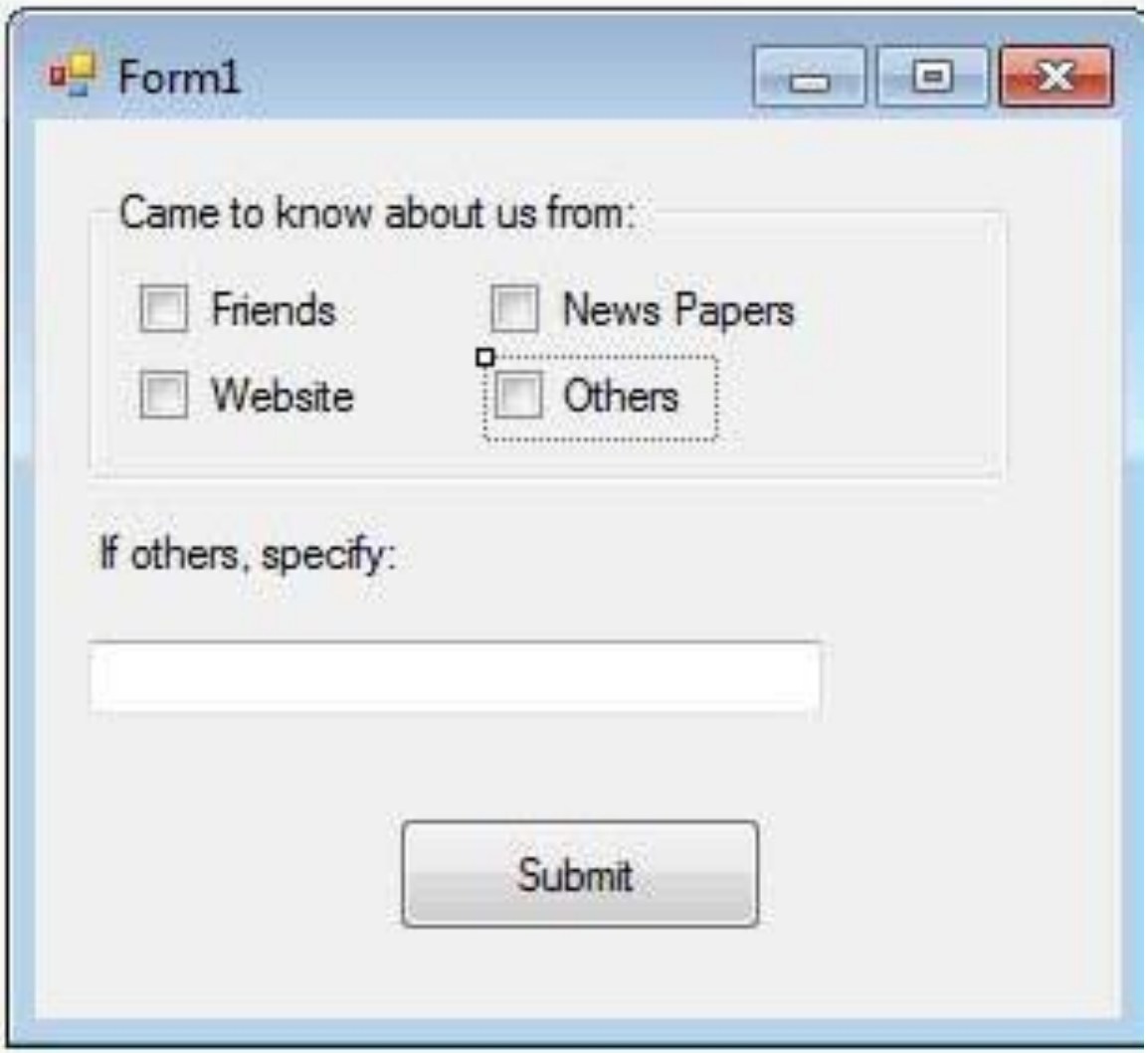
CheckBox Control

- The CheckBox control allows the user to set true/false or yes/no type options. The user can select or deselect it. When a check box is selected it has the value True, and when it is cleared, it holds the value False.
- Let's create two check boxes by dragging CheckBox controls from the Toolbox and dropping on the form.


Start
Debug
Any CPU
Toolbox
Search Toolbox
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
Form1.vb*
Form1.vb [Design]*
Form1
CheckBox1
CheckBox2
CheckBox Control

- The CheckBox control has three states, **checked**, **unchecked** and **indeterminate**. In the indeterminate state, the check box is grayed out. To enable the indeterminate state, the *ThreeState* property of the check box is set to be **True**.

- **Example**
- In this example, let us add four check boxes in a group box. The check boxes will allow the users to choose the source from which they came to know about the organization. If the user chooses the check box with text "others", then the user is asked to specify and a text box is provided to give input. When the user clicks the Submit button, he/she gets an appropriate message.
- The form in design view:


Form1
Came to know about us from:
Friends
News Papers
Website
Others
If others, specify:
Submit

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) _
        Handles MyBase.Load
        ' Set the caption bar text of the form.
        Me.Text = "cct.com"
        Label1.Visible = False
        TextBox1.Visible = False
        TextBox1.Multiline = True
    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) _
        Handles Button1.Click
        Dim str As String
        str = " "
        If CheckBox1.Checked = True Then
            str &= CheckBox1.Text
            str &= " "
        End If
        If CheckBox2.Checked = True Then
            str &= CheckBox2.Text
            str &= " "
        End If
        If CheckBox3.Checked = True Then
            str &= CheckBox3.Text
            str &= " "
        End If
        If CheckBox4.Checked = True Then
            str &= TextBox1.Text
            str &= " "
        End If
        If str <> Nothing Then
            MsgBox(str + vbLf + "Thank you")
        End If
    End Sub
    Private Sub CheckBox4_CheckedChanged(sender As Object, _
        e As EventArgs) Handles CheckBox4.CheckedChanged
        Label1.Visible = True
        TextBox1.Visible = True
    End Sub
End Class
```

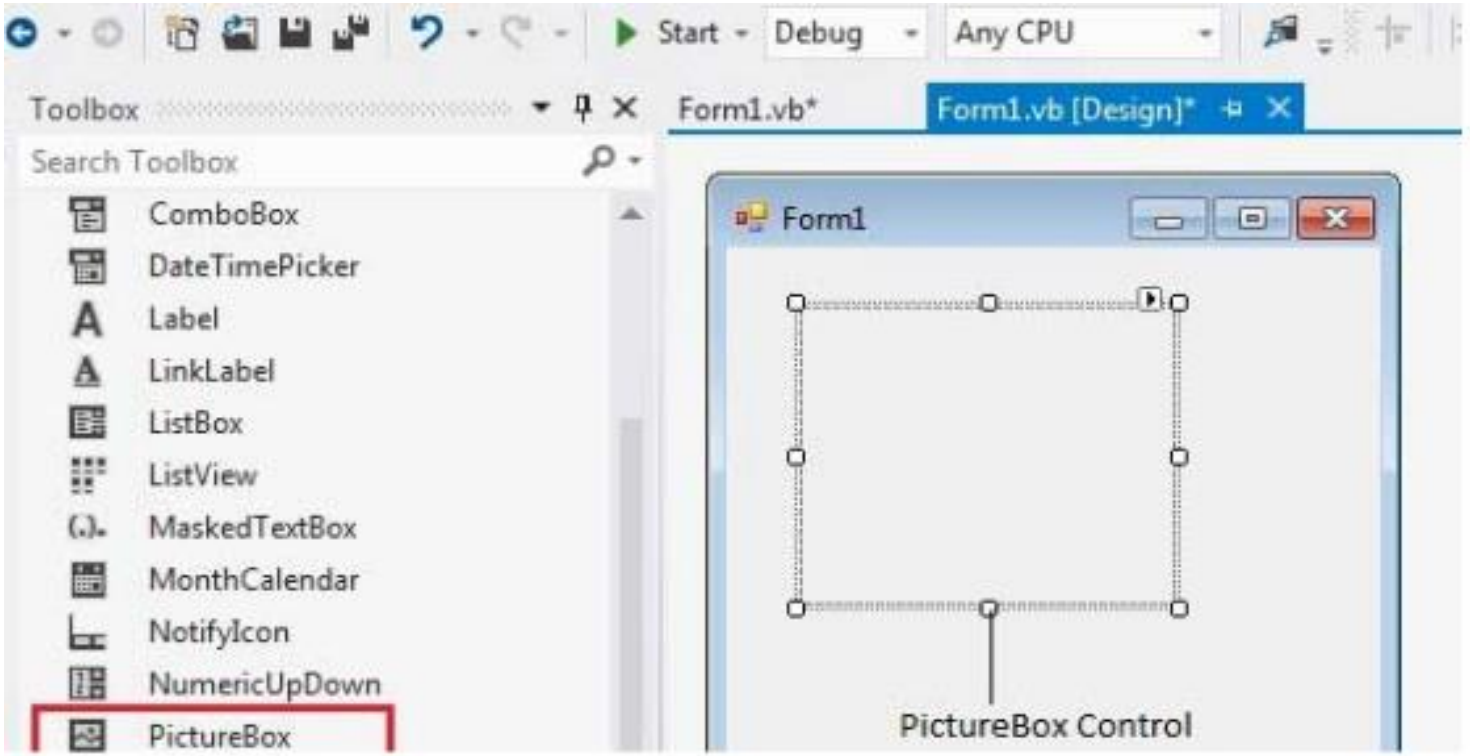
When the above code is executed and run using **Start** button available at the Microsoft Visual Studio tool bar, it will show the following window:

Checking all the boxes:

Clicking the Submit button:

PictureBox Control

- The PictureBox control is used for displaying images on the form. The Image property of the control allows you to set an image both at design time or at run time.
- Let's create a picture box by dragging a PictureBox control from the Toolbox and dropping it on the form.


Start
Debug
Any CPU
Toolbox
Search Toolbox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
Form1.vb*
Form1.vb [Design]*
Form1
PictureBox Control

- **Example**
- In this example, let us put a picture box and a button control on the form. We set the image property of the picture box to logo.png. The Click event of the button named Button1 is coded to stretch the image to a specified size:

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        ' Set the caption bar text of the form.
        Me.Text = "cct.com"
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
        PictureBox1.ClientSize = New Size(300, 300)
        PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
    End Sub
End Class
```

ProgressBar Control

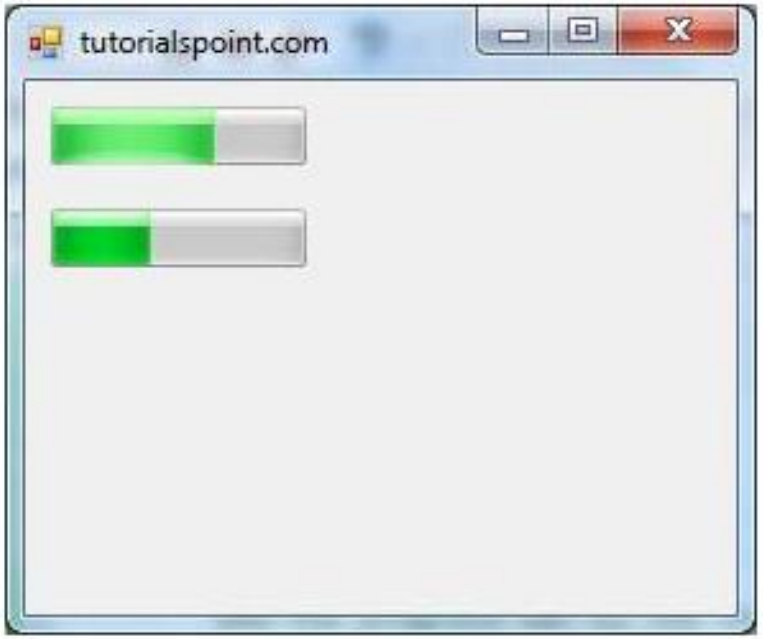
- It represents a Windows progress bar control. It is used to provide visual feedback to your users about the status of some task. It shows a bar that fills in from left to right as the operation progresses.
- Let's click on a ProgressBar control from the Toolbox and place it on the form.

- The main properties of a progress bar are *Value*, *Maximum* and *Minimum*. The Minimum and Maximum properties are used to set the minimum and maximum values that the progress bar can display. The Value property specifies the current position of the progress bar.
- The ProgressBar control is typically used when an application performs tasks such as copying files or printing documents. To a user the application might look unresponsive if there is no visual cue. In such cases, using the ProgressBar allows the programmer to provide a visual status of progress.

- **Example**
- In this example, let us create a progress bar at runtime. Let's double click on the Form and put the follow code in the opened window.

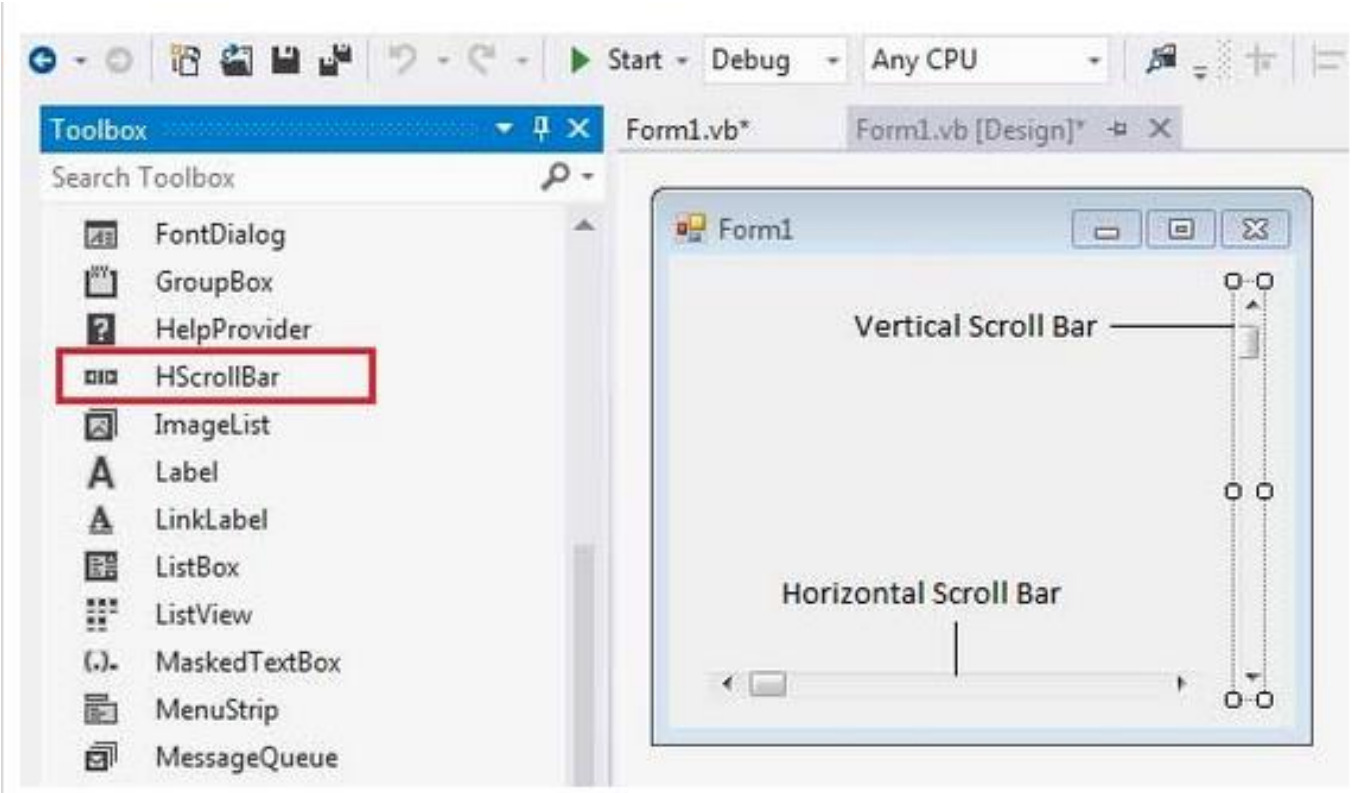
```
Public Class Form1
   Private Sub Form1_Load(sender As Object, e As EventArgs) _
      Handles MyBase.Load
      'create two progress bars
      Dim ProgressBar1 As ProgressBar
      Dim ProgressBar2 As ProgressBar
      ProgressBar1 = New ProgressBar()
      ProgressBar2 = New ProgressBar()
      'set position
      ProgressBar1.Location = New Point(10, 10)
      ProgressBar2.Location = New Point(10, 50)
      'set values
      ProgressBar1.Minimum = 0
      ProgressBar1.Maximum = 200
      ProgressBar1.Value = 130
      ProgressBar2.Minimum = 0
      ProgressBar2.Maximum = 100
      ProgressBar2.Value = 40
      'add the progress bar to the form
      Me.Controls.Add(ProgressBar1)
      Me.Controls.Add(ProgressBar2)
      ' Set the caption bar text of the form.
      Me.Text = "tutorialspoint.com"
   End Sub
End Class
```

When the above code is executed and run using **Start** button available at the Microsoft Visual Studio tool bar, it will show the following window:



ScrollBar Control

- The ScrollBar controls display vertical and horizontal scroll bars on the form. This is used for navigating through large amount of information. There are two types of scroll bar controls: **HScrollBar** for horizontal scroll bars and **VScrollBar** for vertical scroll bars. These are used independently from each other.
- Let's click on HScrollBar control and VScrollBar control from the Toolbox and place them on the form.


Start
Debug
Any CPU
Toolbox
Search Toolbox
FontDialog
GroupBox
HelpProvider
HScrollBar
ImageList
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MenuStrip
MessageQueue
Form1.vb*
Form1.vb [Design]*
Form1
Vertical Scroll Bar
Horizontal Scroll Bar

- **Example**
- In this example, let us create two scroll bars at runtime. Let's double click on the Form and put the follow code in the opened window.

```
Public Class Form1
   Private Sub Form1_Load(sender As Object, e As EventArgs) _
      Handles MyBase.Load
      'create two scroll bars
      Dim hs As HScrollBar
      Dim vs As VScrollBar
      hs = New HScrollBar()
      vs = New VScrollBar()
      'set properties
      hs.Location = New Point(10, 200)
      hs.Size = New Size(175, 15)
      hs.Value = 50
      vs.Location = New Point(200, 30)
      vs.Size = New Size(15, 175)
      hs.Value = 50
      'adding the scroll bars to the form
      Me.Controls.Add(hs)
      Me.Controls.Add(vs)
      ' Set the caption bar text of the form.
      Me.Text = "tutorialspoint.com"
   End Sub
End Class
```