



Unit 2

Designing User Interface-1

Introduction

In previous lesson, we have learned how to write simple Visual Basic code. In this lesson, we will learn how to work with some common controls and write codes for them. Some of the commonly used controls are label, text box, button, list box and combo box. However, in this lesson, we shall only deal with the text box the label, and buttons we shall deal with other controls later.

Lesson 2.1-3

Adding Basic Controls



Outcomes

Upon completion of this unit you will be able to

- *Place* textbox control on the Form.
- *Place* label control on the Form.
- *Place* command button on the Form.

TextBox Controls

The TextBox is the standard control for accepting input from the user as well as to display the output. For this reason, they tend to be the most frequently used controls in the majority of Windows applications. It can handle string (text) and numeric data but not images or pictures. String in a TextBox can be converted to a numeric data by using the function Val (text).

In this section, we will discuss the most useful properties of TextBox controls. After you place a TextBox control on a form, you must set a few basic properties. The first thing I do as soon as I create a new TextBox control is clear its *Text* property. If this is a multiline field, I also set the *MultiLine* property to True.

You can set the *Alignment* property of TextBox controls to left align, right align, or center the contents of the control.



If you're dealing with a numeric field, you probably want to set a limit on the number of characters that the user can enter in the field. You can do it very easily using the *MaxLength* property. A 0 value (the default) means that you can enter any number of characters; any positive value *N* enforces a limit to the length of the field's contents to be *N* characters long.

If you're creating password fields, you should set the *PasswordChar* property to a character string, typically an asterisk. In this case, your program can read and modify the contents of this `TextBox` control as usual, but users see only a row of asterisks.



Tip

The `Caption` property is the most common property that displays text on a control such as a command button and a label. `Text Box` controls does not support the `Caption` property. The `Text` property holds text for `Text Box` controls.

Table 1. Common `TextBox` properties.

Property	Description
<code>Alignment</code>	Determines whether the text box's text appears left-justified, centered, or right-justified within the text box's boundaries.
<code>BackColor</code>	Specifies the text box's background color. Click the <code>BackColor</code> property's palette down arrow to see a list of colors and click <code>Categorized</code> to see a list of common Windows control colors.
<code>BorderStyle</code>	Determines whether a single-line border appears around the text box.
<code>Enabled</code>	Determines whether the text box is active. Often, you'll change the <code>Enabled</code> property at runtime with code when a text box is no longer needed.
<code>Font</code>	Produces a <code>Font</code> dialog box in which you can set the <code>Text</code> property's font name, style, and size.
<code>ForeColor</code>	Holds the color of the text box's text.
<code>Height</code>	Holds the height of the text box's outline in twips.



Left	Holds the number of twips from the text box's left edge to the Form window's left edge.
Locked	Determines whether the user can edit the text inside the text box that appears
MaxLength	This property specifies the maximum number of characters that the text box will accept. A value of 0 indicates that the user can enter a value of any length.
MousePointer	Determines the shape of the mouse cursor when the user moves the mouse over the text box.
MultiLine	Lets the text box hold multiple lines of text or sets the text box to hold only a single line of text. Add scrollbars if you wish to put text in a multiline text box so your users can scroll through the text.
PasswordChar	This property designates a character, such as an asterisk(*), that will appear in place of the characters the user types into the text box. In other words, if the user is entering a secret code, only asterisks will appear on the screen as the user types the code instead of the code's value so that nobody looking over the user's shoulder can read the code. Although the asterisks appear, the text box receives the actual value that the user types.
Text	This property specifies the initial text (the default value) that appears in the text box.

Label

The label is a very useful control for Visual Basic, as it is not only used to provide a descriptive caption and guides to the users, it can also be used to display outputs. It is different from text box because it can only display static text, which means the user cannot change the text. Using the syntax `Label.Text`, it can display text and numeric data. In most cases, you just place a Label control where you need it, set its *Caption* property to a suitable string. *Caption* is the default property for Label controls. Other useful properties are *BorderStyle* and *Alignment*.

Table 2. Common label properties.

Property	Description
Alignment	Determines how the text is aligned in the Label.
AutoSize	Enlarges the label's size properties, when True, Allows for automatic resizing of the Label.
BackColor	Specifies the label's background color. Click the BackColor's palette down arrow to see a list of colors and click Categorized to see a list of common Windows control colors.



BackStyle	Determines whether the background shows through the label or if the label covers up its background text, graphics, and color.
BorderStyle	Determines whether a single-line border appears around the label.
Caption	Holds the text that appears on the label.
Enabled	Determines whether the label is active. Often, you'll change the Enabled property at runtime with code when a label is no longer needed.
Font	Specifies the font name, style and size of the text displayed in the Label. Produces a Font dialog box in which you can set the caption's font name, style, and size.
ForeColor	Holds the color of the label's text.
Height	Holds the height of the label's outline in twips.
Left	Holds the number of twips from the label's left edge to the Form window's left edge.
MousePointer	Determines the shape of the mouse cursor when the user moves the mouse over the label.
TabIndex	Specifies the order of the label in the focus order. Although the label cannot receive the direct focus, the label can be part of the focus order.
ToolTip	Text Holds the text that appears as a tooltip at runtime.
Top	Holds the number of twips from the label's top edge to the Form window's top edge.
Visible	Determines whether the label appears or is hidden from the user.
Width	Holds the width of the label in twips.



Tip

Ensure that all Label controls are large enough to display their text.

Command Buttons

Command buttons appear in almost every window of every Windows application. Command buttons determine when the user wants to do something, such as exit the application or begin printing. In almost



every case, you will perform these tasks to add a command button to an application. In most cases, you just draw the control on the form's surface, set its *Caption* property to a suitable string. To make the button functional; you write code in its *Click* event procedure. The *CommandButton* control supports the usual set of keyboard and mouse events (*KeyDown*, *KeyPress*, *KeyUp*, *MouseDown*, *MouseMove*, *MouseUp*, but not the *DbClick* event) and also the *GotFocus* and *LostFocus* events, but you'll rarely have to write code in the corresponding event procedures.

Although the command button control supports 36 properties, you'll only set the *Name* and *Caption* properties in most cases. In addition, although command button controls support 15 events, you'll only write code for the *Click* event in most cases.

**Tip**

You can set some properties only at design time (such as a control's *Name* property), you can set some properties both at design time and at runtime inside event procedures and other module code (such as a *caption*), and you can set some properties only at runtime from within the program (such as a list box's entries). *ToolTip Text* holds the text that appears as a tooltip at runtime.

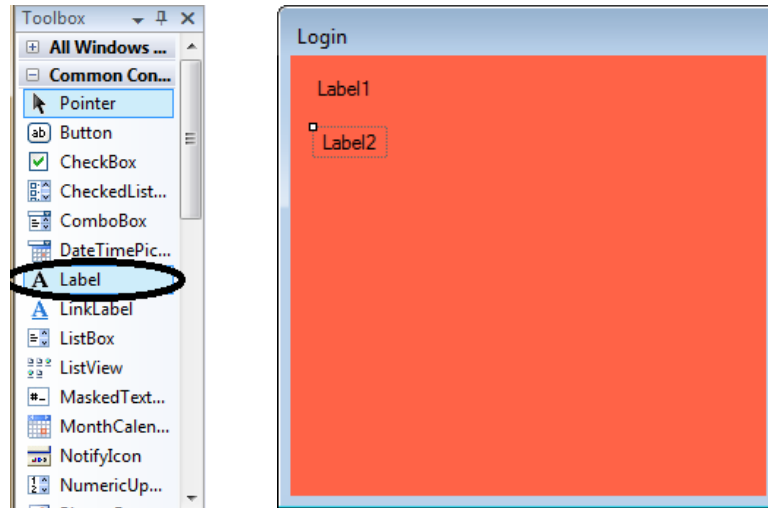
**Note it**

Command button resides on most forms just so the user can click the button to trigger some event that the user is ready to start.

In this section we will add two labels, two text box and two buttons as windows form controls, and then we will change the various property of this controls.

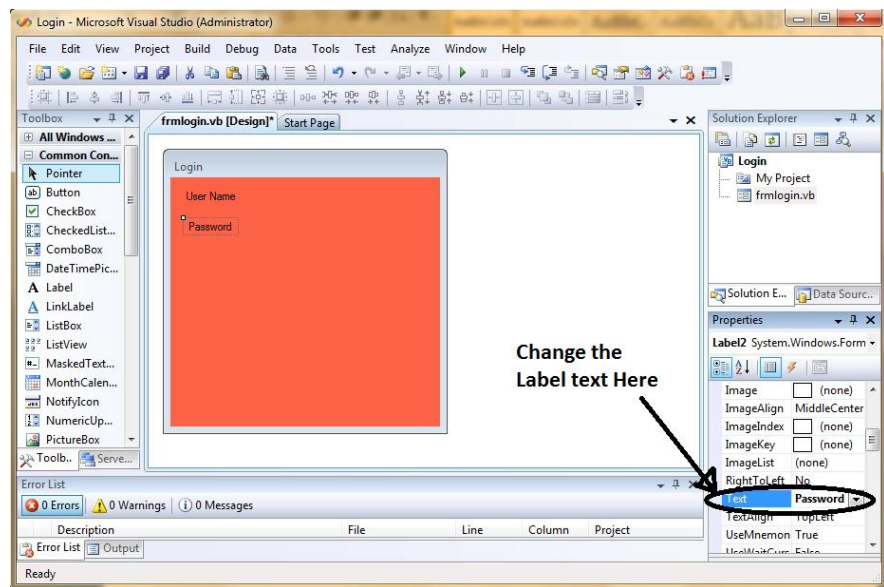
Step 1: Now consider the previous Login form, or open the Login project.

Step 2: Now take a Two Label control from left side Toolbox or Drag a Label control from Toolbox and drop on Login forms like the following:



Step 3: Now just single click on Label1 which is appeared on windows form, and then change the Text of Label1 from right side Labels properties window. Give the Text name is User name.

Step 4: Now just single click on Label2 which is appeared on windows form, and then change the Text of Label2 from right side Labels properties window. Give the Text name is Password. Now you will look like the following:

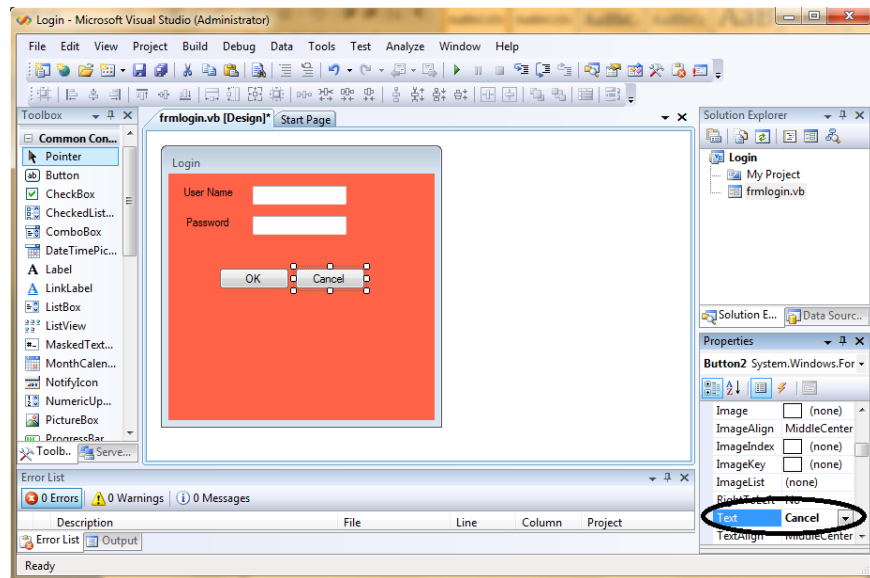




Step 5: Now drag two TextBox controls from the Toolbox and drop on Login form, which will look like the following:

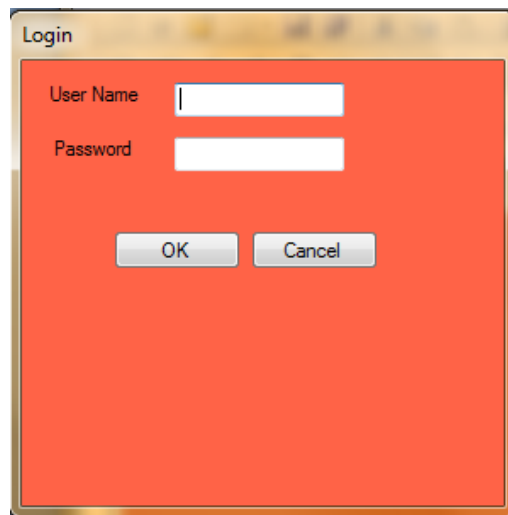


Step 6: Now drag Two Button controls from Toolbox and drop on windows form, which will look like following:

Step 7: Now change the text of buttons using button Text property window. So at first, just single click on Button1, and then change the text property of Button1 from the properties window, and put the name OK. Then just single click on Button2, and then change the text property of Button2 from the properties window, and put the name Cancel, which will look like the following:



Step 8: Now save the project by clicking on  icon from the toolbar and run the project by clicking on  from the toolbar, which will look like the following:



Example

Now we will see how to create a simple calculator that adds two numbers using the Textbox, labels and button. In this program, you insert two textboxes, two labels and one button. The two textboxes are for the users to enter two numbers, one label is to display the equal sign with label Sum. The last label is to display the answer of the Sum. For the first label, change the text property of the label by typing Sum= over the default text Label1. Further, change its font to bold and font size to 10. For the second label, delete the default text Label2 and change its font to bold and font size to 10.



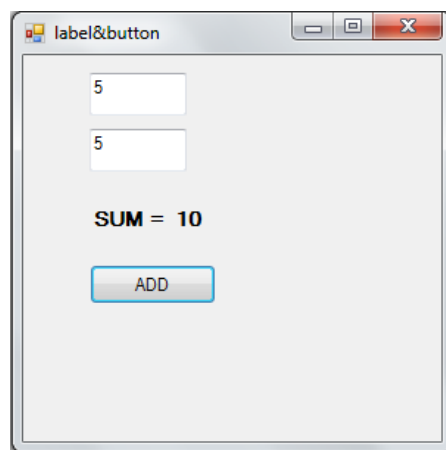
Code

```
Private Sub Button1_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
```

```
    Label2.Text = Val (TextBox1.Text) + Val (TextBox2.Text)
```

```
End Sub
```

*The function Val is to convert text to numeric value. Without using Val, you will see that two numbers are joined together without adding them.



Unit summary



Summary

In this unit you learned the three fundamental controls that appear on almost every application's Form window: command buttons, labels, and text boxes.



Assessment



Assessment

MCQ

1. A Label control displays the text specified by property
 - a) Caption
 - b)Data
 - c) Text
 - d)Name
2. A Button appears flat if its property is set to Flat.
 - a) BorderStyle
 - b)FlatStyle
 - c) Style
 - d)BackStyle

Exercise

1. Write a note on Text Boxes.
2. What is the purpose of the caption property of a command button? Briefly explain.
3. Which property must be set to center text in a label? What should be the value of the property?
4. Explain how to change a label's caption bold at design time and at run time.
5. What basic statements will clear the current contents of a textbox and a label?
6. You can clear the contents of a textbox and label by setting the property to an empty string. Use "" (no space between quotation marks). Write the code for this.
7. Explain the following properties of Textbox control
 - a) Locked
 - b) MaxLength
 - b) MultiLine
 - c) PasswordChar
 - d) ScrollBars
 - e) Text



Lesson 2.4-2.5

Check box and Radio button

Upon completion of this unit you will be able to:



- Use checkboxes to allow users to select options
- Use radioto allow users to select options.

Outcomes

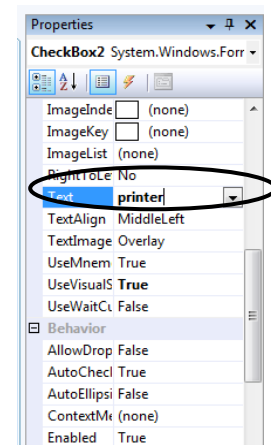
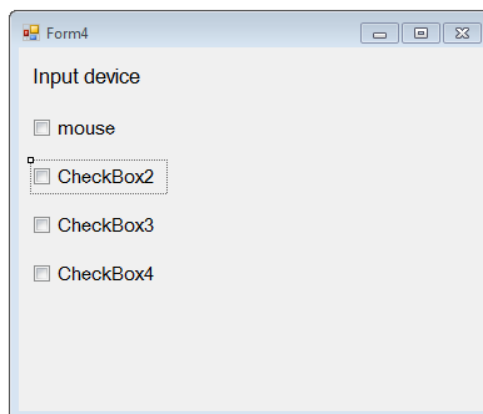
CheckBox

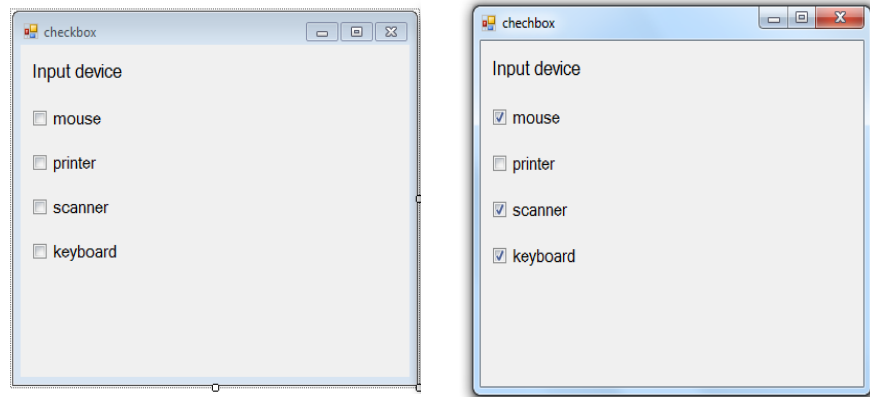
The Check box is a very useful control in Visual Basic 2008. It allows the user to select or deselect one or more items by checking the checkbox/checkboxes concerned. For example, in the Font dialog box of any Microsoft Text editor like FrontPage, there are many checkboxes under the Effects section. The user can choose underline, subscript, small caps, superscript, blink and etc. In Visual Basic, you may create a shopping cart where the user can click on checkboxes that correspond to the items they intend to buy, and the total payment can be computed at the same time as shown in Example 1.

The value property of a checkbox is set to 0 if unchecked and 1 if it is checked. Use the caption property of a checkbox for the text that you want to appear next to the box. The three letter prefix for naming a checkbox is “chk”

Example 1

In this example, the user can select what are the inputs devices are there

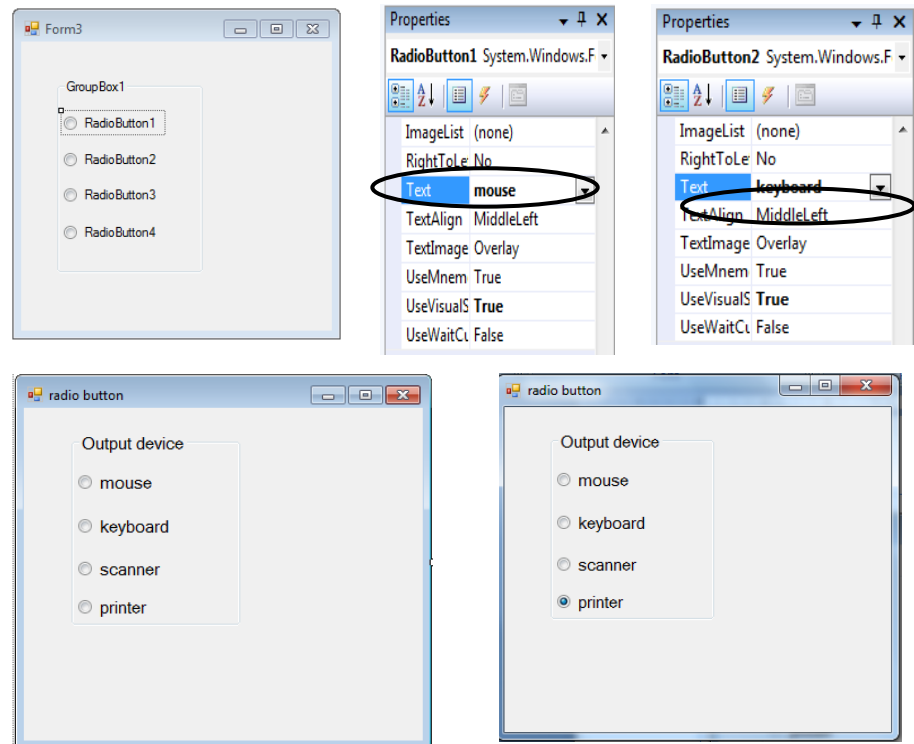




Using Radio Button

The radio button is also a very useful control in Visual Basic 2008. However, it operates differently from the check boxes. While the checkboxes work independently and allows the user to select one or more items, radio buttons are mutually exclusive, which means the user can only choose one item only out of a number of choices. Here is an example which allows the users to select one color only.

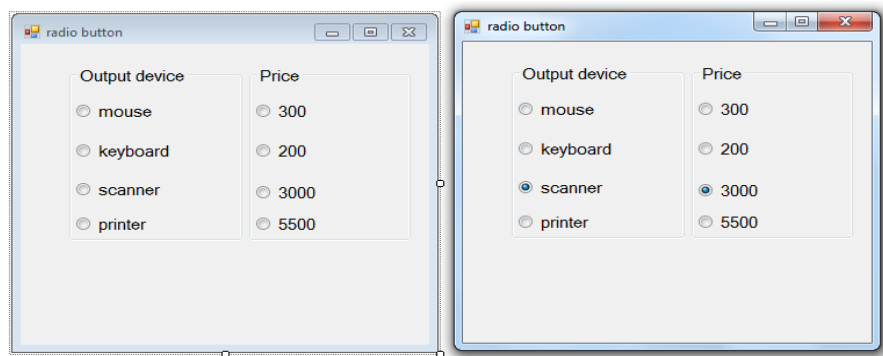
Example 1





Although the user may only select one item at a time, he may make more than one selection if those items belong to different categories. For example, the user wish to choose device name and price, he needs to select one device and one price, which means one selection in each category. This is easily achieved in VB2008 by using the Groupbox control under the containers categories. After inserting the Groupbox into the form, you can proceed to insert the radio buttons into the Groupbox. In Example 2, the users can select one device and price of the device.

Example 2





Assessment



Assessment

MCQ

1. Which property sets a CheckBox's label?
a) Text b) Value
c) Label d) Checked
2. Which property specifies whether a CheckBox is selected?
a) Selected b) Checked
c) Clicked d) Check
3. How many CheckBoxes in a GUI can be selected at once?
a) 0 b) 1
c) 4 d) any number
4. The text that appears alongside a CheckBox is referred to as the _____.
a) CheckBox label b) CheckBox name
c) CheckBox value d) CheckBox data
5. A CheckBox is selected when its Checked property is set to _____.
a) On b) True
c) Selected d) Checked.

Exercise

1. Write notes on CheckBox and RadioButton.



Lesson 2.6-2.7

ComboBox and ListBox

Upon completion of this unit you will be able to:



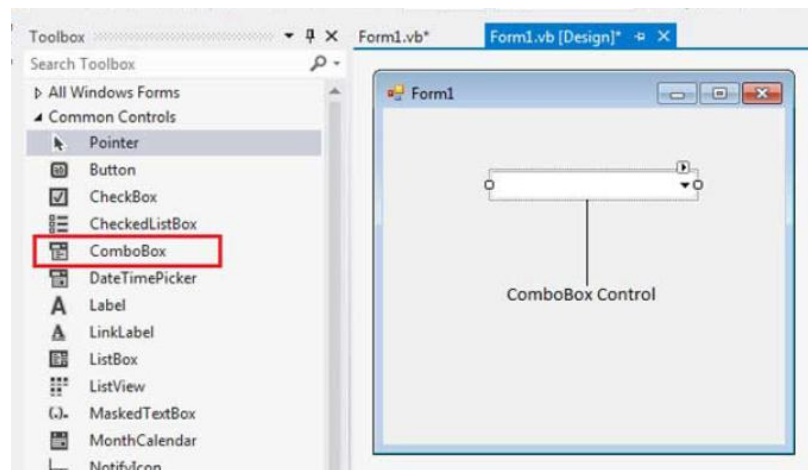
Outcomes

ComboBox

- Use ComboBox.
- Use ListBox.

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. However, the user needs to click on the small arrowhead on the right of the combo box to see the items which are presented in a drop-down list.

Let's create a combo box by dragging a ComboBox control from the Toolbox and dropping it on the form.



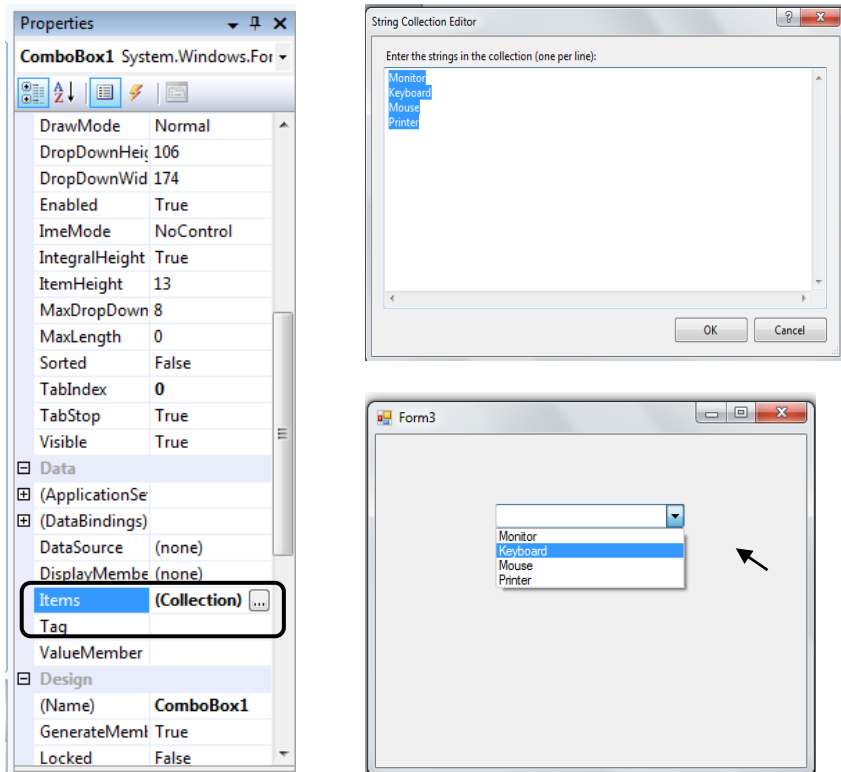
Adding items to a ComboBox

In order to add items to the list at design time, use the String Collection Editor under Items property. You will also need type an item under the text property in order to display the default item at runtime. At runtime, you can access and manipulate the items through the methods and properties of the Items collection, which are described shortly.

To demonstrate how to add items at design time, start a new project and insert a ComboBox on the form. Now Right-click on the ComboBox



to access the properties window. Next, click on collection of the Item property, you will be presented with String Collection Editor whereby you can enter the items one by one by typing the text and press the Enter key



Besides, you may add items using the **Add () method**. For example, if you wish to add a number of items to Combo box 1, you can key in the following statement

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    ComboBox1.Items.Add("Scanner")
End Sub
```

You can also allow the user to add new items using the **InputDialog** function, as follows:

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    Dim newitem
    newitem = InputBox("Enter new Item")
    ComboBox1.Items.Add(newitem)
End Sub
```




Removing items from a Combo Box

To remove items at design time, simply open the String Collection Editor and delete the items at line by line or all at once using the Delete key.

To remove the items at runtime, you can use the **Remove** method, as illustrated in the following example. In this example, add a second button and label it "Remove Items". Click on this button and enter the following code:

```
Private Sub Button2_Click(ByVal sender As System.Object,  
ByVal e As System.EventArgs) Handles Button2.Click  
  
    ComboBox1.Items.Remove("Printer")  
  
End Sub
```

The item Printer will be removed after running the program.

ListBox

The function of the List Box is to present a list of items where the user can click and select the items from the list. Items can be added at design time and at runtime. The items can also be removed at design time and also at runtime. The list box displays the items all at once in a text area whilst combo box displays only one item initially and the user needs to click on the handle of the combo box to view the items in a drop-down list.

Adding items to a List Box

To demonstrate how to add items at design time, start a new project and insert a list box on the form. Now Right-click on the list box to access the properties window. Next, click on collection of the Item property, you will be presented with String Collection Editor whereby you can enter the items one by one by typing the text and press the Enter key, as shown in Figure –

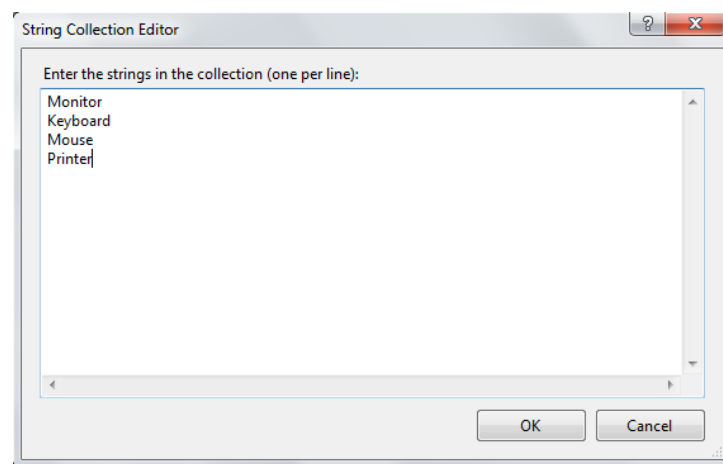
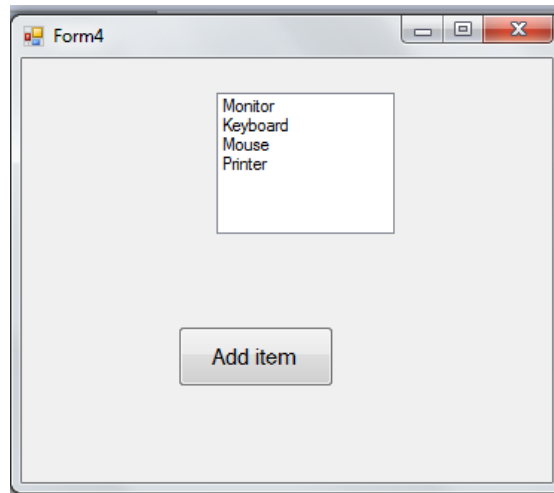


Figure 1: String Collection Editor

After clicking on the OK button, the items will be displayed in the text box, as shown in Figure 6.2



Items can also be added at runtime using the **Add () method**. For a list box, Item is an object subordinated to the object ListBox. Item comprises a method call Add () that is used to add items to the list box. To add an item to a list box, you can use the following syntax:

ListBox.Item.Add(“Text”)

For example, if you wish to add a new item to ListBox1 above, you can key-in the following statement

```
Private Sub Button1_Click(ByVal sender As System.Object,  
ByVal e As System.EventArgs) Handles Button1.Click  
    ListBox1.Items.Add("Scanner")  
End Sub
```

The item Scanner will be added to the end of the list, as shown in Figure 2

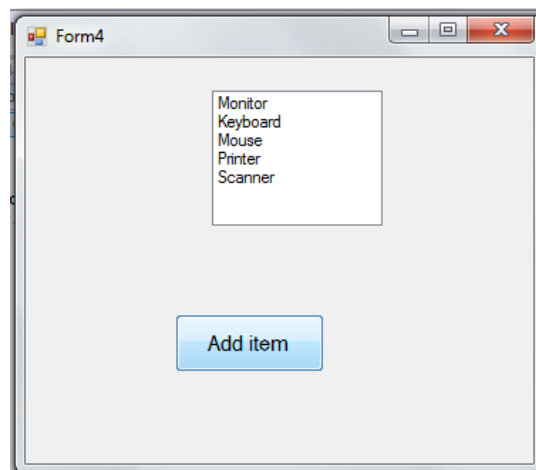


Figure 2

You can also allow the user to add items using the InputBox function, as follows:

```
Private Sub Button1_Click(ByVal sender As System.Object,
```



```
ByVal e As System.EventArgs) Handles Button1.Click
    Dim newItem
    newItem = InputBox("Enter new Item")
    ListBox1.Items.Add(newItem)
End Sub
```

Removing items from a List Box

To remove items at design time, simply open the String Collection Editor and delete the items at line by line or all at once using the Delete key. To remove the items at runtime, you can use the **Remove** method, as illustrated in the following example. In this example, add a second button and label it “Remove Items”. Click on this button and enter the following code:

```
Private Sub Button2_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles Button2.Click

    ListBox1.Items.Remove("Printer")

End Sub
```

The item printer will be removed after running the program. You can also let the user choose which item to delete.

To clear all the items at once, use the clear method, as illustrated in the following example. In this example, add a button and label it “Clear Items”

```
Private Sub Button3_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles Button3.Click

    ListBox1.Items.Clear()

End Sub
```



Assessment



Assessment

MCQ

1. The ComboBox at the top of the Properties window is the.
a) component object box b) control box
c) control object box d) component box

Exercise

1. Write the steps to add items in ListBox.