Chapter 5

Common Controls

Learning Objectives

After completing this chapter, you will be able to create:

- *TextBox and MaskedTextBox controls.*
- *Label controls.*
- *LinkLabel controls.*
- *Button and ComboBox controls.*
- *ListBox, CheckedListBox, and PictureBox controls.*
- *NotifyIcon and NumericUpDown controls.*
- *Picker and RadioButton controls.*
- *TreeView and ListView controls.*
- *ProgressBar and Tooltip controls.*

INTRODUCTION

In this chapter, you will learn about various controls such as **TextBox** control, **RichTextBox** control, **ListBox** control, **CheckBox** control, and so on.

TextBox CONTROL

A **TextBox** control is the most commonly used control in the windows user interface. It is a small edit box that can be used to insert, select, and scroll the text if it does not fit into the area of the control. To add a **TextBox** control to the form, double-click on the control in the toolbox. Alternatively, you can drag and drop it directly on the form. When you select the **TextBox** control on the form, a small forward arrow will be displayed on the top-right corner of the control, as shown in Figure 5-1. If you click on the forward arrow, the **Multiline** check box will be displayed, as shown in Figure 5-2. If you select the **Multiline** check box, its **Multiline** property will be set to **True**. Alternatively, this property can be set to **True** in the **Properties** window, where the properties of the **TextBox** control are displayed.

*Figure 5-1 The **TextBox** control*

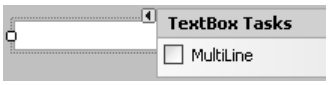

*Figure 5-2 The **TextBox** control with the **Mutiline** property*

Properties of the TextBox Controls

Some common properties of the **TextBox** control are listed in Table 5-1.

Properties	Functions
BackColor	Sets the background color of the control
BorderStyle	Sets the border style of the control
Cursor	Sets the appearance of the cursor when it is placed over a control
Font	Determines the font and size used to display the text in the control
ForeColor	Sets the foreground color of the display text and graphics in the control
Lines	Sets the lines of text in a multiline edit box
ScrollBars	Indicates the type of scroll bar that should appear in a multiline **TextBox** control
Text	Sets the current text in a **TextBox** control

Properties	Functions
TextAlign	Sets the alignment of the text in a **TextBox** control
MaxLength	Sets the maximum number of characters that can be entered in a **TextBox** control
PasswordChar	Displays the password characters
ReadOnly	Determines whether the text in a **TextBox** control can be changed
WordWrap	Determines whether the lines are automatically wrapped for a multiline **TextBox** control
Name	Sets the name used in the code to identify the control
Anchor	Determines the edge to which a control should be bound
Dock	Determines the borders to be docked to the container

Table 5-1 *Properties of the* ***TextBox*** *control*

Adding ScrollBars to the TextBox Controls

There are four ways to add scrollbars to a **TextBox** control, which are given below:

1. None
2. Horizontal
3. Vertical
4. Both

The following steps are required to add scrollbars to a **TextBox** control:

Step 1
Change the value of the **Multiline** property of the **TextBox** control to **True**.

Step 2
Change the default value of the **ScrollBars** property to the desired value.

Aligning the Text in the TextBox Controls

You can align the text in a **TextBox** control with the help of the **TextAlign** property. This property has the following three members:

1. Left
2. Right
3. Center

You can align the text in a **TextBox** control as per your requirement by assigning any one of the given values to the **TextAlign** property of the **TextBox** control.

Read-only Property of the TextBox Controls

Sometimes you do not want the user to change the value of the **Text** property of the **TextBox** control. In such cases, the value **True** should be assigned to the **ReadOnly** property.

Creating a TextBox Control through Source Code

The **TextBox** controls can be created through coding. For example, if you want to create a **TextBox** control of size 100, 30 with the value **TextBox1** assigned to its **Text** property, then enter the following source code for the form load event:

```
Dim TextBox1 As New TextBox                          1
TextBox1.Size = New Size(100, 30)                    2
TextBox1.Text = "TextBox1"                           3
Me.Controls.Add(TextBox1)                            4
End Sub                                              5
```

Label CONTROL

A **Label** control is used to display the text on a form. This text cannot be changed by the user at runtime.

Formatting Text in Label Controls

You can format the text in a **Label** control such that the size of the control matches with the size of the text in it. To do so, the value **True** should be assigned to the **AutoSize** property of the **Label** control.

Aligning Text in Label Controls

The **TextAlign** property of the **Label** control has nine members, which are listed below:

1. TopRight
2. TopLeft
3. TopCenter
4. MiddleRight
5. MiddleLeft
6. MiddleCenter
7. BottomRight
8. BottomLeft
9. BottomCenter

You can assign any one of the above mentioned members to the **TextAlign** property of the **Label** control. These values will determine the position of the text in a **Label** control.

LinkLabel CONTROL

These controls are like **Label** controls and they support web-style hyperlinks to other windows forms and internet.

Properties of LinkLabel Controls

Some common properties of the **LinkLabel** control are given in Table 5-2.

Properties	Functions
ActiveLinkColor	Sets the color of the hyperlink when the user clicks on the link
DisabledLinkColor	Sets the color of the disabled link
LinkColor	Determines the color of the default state of the hyperlink
LinkArea	Sets a portion of the text as a hyperlink

Table 5-2 *Properties of the* ***LinkLabel*** *control*

Creating a LinkLabel Control

The following steps are required to create a **LinkLabel** control:

Step 1
Double-click on the **LinkLabel** control in the toolbox; it will be added to the form.

Step 2
Enter the following text in the **Text** property of the **LinkLabel** control:

Click here to see the profile.

Step 3
Select the **LinkArea** property of the **LinkLabel** control. Next, choose the button on the right of the **LinkArea** property; the **LinkArea Editor** dialog box will be displayed, as shown in Figure 5-3. Click on the word **here** which is set as a hyperlink and choose the **OK** button in the **LinkArea Editor** dialog box.

Figure 5-3 *The* ***LinkArea Editor*** *dialog box*

Step 4
Select the **ActiveLinkColor**, **LinkColor**, and **VisitedLinkColor** properties one-by-one from the **Properties** window. Set their values according to your requirement.

Step 5
Double-click on the **LinkLabel** control; the code window will be displayed. Enter the following source code for the hyperlink:

```
Private Sub LinkLabel1_LinkClicked(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles
LinkLabel1.LinkVisited = True                                          1
System.Diagnostics.Process.Start("www.cadcim.com")                     2
End Sub                                                                3
```

Explanation

In the above program, whenever a user chooses the hyperlink portion of the text in the **LinkLabel** control, the **System.Diagnostics.Process.Start** method will to start the navigation.

Step 6
Execute the application; the **controls_demo** form will be displayed, as shown in Figure 5-4.

Figure 5-4 *The* ***controls_demo*** *form*

Step 7
Click on the word **here**, which is set as a hyperlink. It will start navigating the site *www.cadcim.com*.

Creating a LinkLabel Control through Source Code

The **LinkLabel** control can also be created through source code. For example, you may need

to create a **LinkLabel** control having the value **Click here to see the profile** in its **Text** property. Also, you need to set the word **here** as a hyperlink and navigate to the site *www.cadcim.com*. To do so, follow the procedure given next.

Enter the following source code for the **Load** event of the form:

```
Dim LinkLabel1 As New LinkLabel                                    1
LinkLabel1.Text = "Click here to see the profile"                  2
LinkLabel1.AutoSize = True                                         3
LinkLabel1.Links.Add(6, 4)                                         4
LinkLabel1.LinkVisited = True                                      5
LinkLabel1.Left = 70                                               6
LinkLabel1.Top = 100                                               7
AddHandler LinkLabel1.LinkClicked, AddressOf Me.LinkLabel1_Click   8
Me.Controls.Add(LinkLabel1)                                        9
```

Enter the following source code for the **Click** event of the **LinkLabel** control:

```
System.Diagnostics.Process.Start("www.cadcim.com")                 10
```

Explanation

The line-by-line explanation of the above given source code is as follows:

Line 4
LinkLabel1.Links.Add(6, 4)
In this lines, **Links** is a collection of multiple links where the hyperlink for the **LinkLabel** control is stored. The **Add** method is used to add the new hyperlink.

Line 8
AddHandler LinkLabel1.LinkClicked, AddressOf Me.LinkLabel1_Click
In this line, an event handler **LinkLabel1.LinkClick** is connected to the **Click** event of the **LinkLabel** control using the **AddHandler** method and the **AddressOf** operator.

Line 9
Me.Controls.Add(LinkLabel1)
This line is used to add a **LinkLabel** control to the form.

The output of the above source code is shown in Figure 5-5. When you click on the **here** link on the form, the www.cadcim.com website will be displayed, as shown in Figure 5-6.

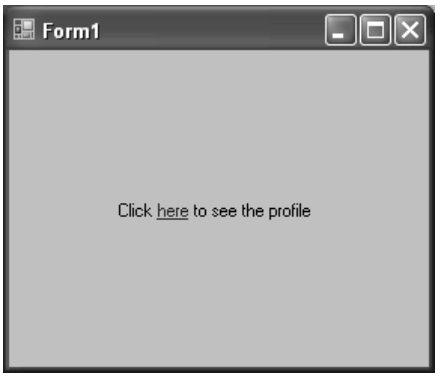

Figure 5-5 *Form displaying the* ***LinkLabel*** *control*

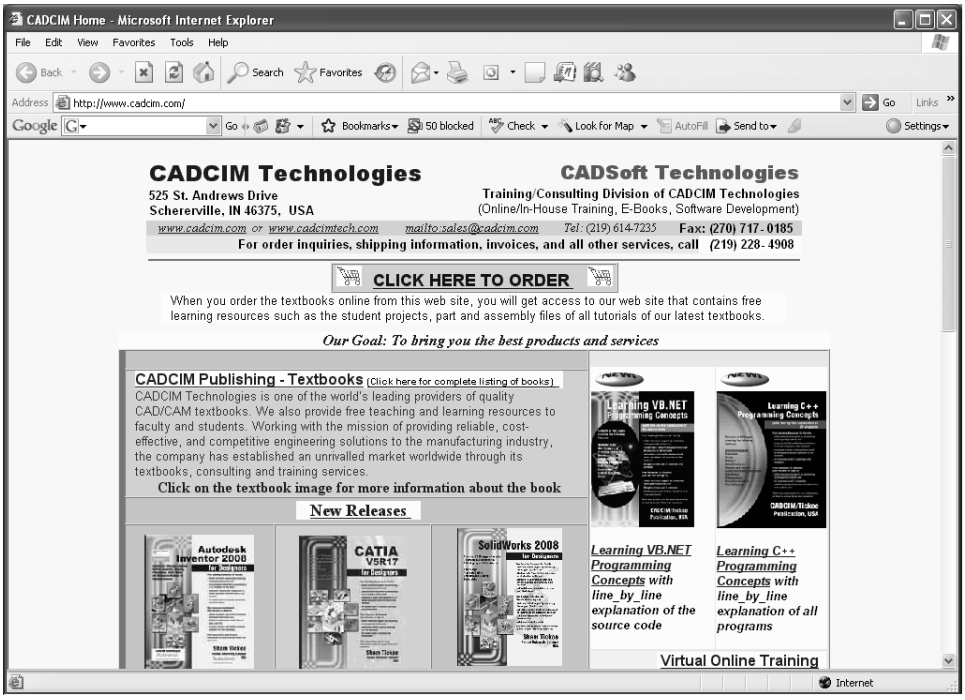

Figure 5-6 *Web page displayed using the* ***LinkLabel*** *control*

ListBox CONTROL

A **ListBox** control is used to allow the user to select a single or a set of multiple items from a list of options at runtime.

Creating a ListBox Control

To add a **ListBox** control to a form, double-click on it in the toolbox. Alternatively, drag the **ListBox** control from the toolbox and drop it on the required location of the form. If you select the **ListBox** control on the form, a small forward arrow will be displayed on top right corner of the control, as shown in Figure 5-7. When you click on the forward arrow, two properties, **Use data bound items** and **Edit Items** will be displayed, as shown in Figure 5-8.

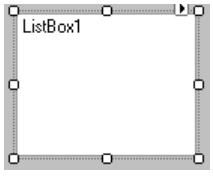

Figure 5-7 *The* ***ListBox*** *control*

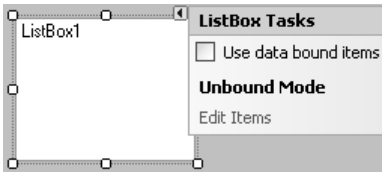

Figure 5-8 *The* ***ListBox*** *control with its properties*

If you select the **Use data bound items** check box, the data binding properties will be displayed, as shown in Figure 5-9. These properties will be discussed in detail in the later chapters.

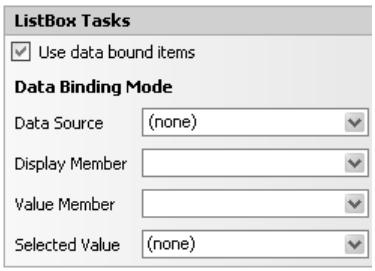

Figure 5-9 *The* ***Use data bound items*** *property*

You can use the **Edit Items** link to add the entries in the **ListBox** control. To add the entries, click on the **Edit Items** link; the **String Collection Editor** dialog box will be displayed, as shown in Figure 5-10.

Figure 5-10 *The* ***String Collection Editor*** *dialog box*

Properties of the ListBox Controls

Some common properties of a **ListBox** control are listed in Table 5-3.

Properties	Functions
ColumnWidth	Sets the width of each column in a multicolumn **ListBox** control
DrawMode	Sets the drawing mode of a **ListBox** control
HorizontalExtent	Sets the width of the **ListBox** control so that it can scroll horizontally. It can be set only if the **HorizontalScrollbar** property is set to **True**

Properties	Functions
HorizontalScrollbar	Determines whether a horizontal scroll bar will be displayed in a **ListBox** control
MultiColumn	Determines whether more than one column can be selected at a time in a **ListBox** control
ScrollAlwaysVisible	Determines whether the scroll bars are always visible in a **ListBox** control or not
SelectionMode	Sets the mode of selection for the items in a **ListBox** control. It can be multi-selection, single-selection, or none
Sorted	Determines that the items in a **ListBox** control are sorted alphabetically
Items	Stores the list of items in a **ListBox** control

Table 5-3 *Properties of the* ***ListBox*** *control*

Creating Multicolumn ListBox Controls

You can create a **ListBox** control with multicolumn using source code. The following source code is required to create a multicolumn **ListBox** control:

```
Private Sub Form1_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load                    1
Dim ListBox1 As ListBox                                             2
ListBox1 = New ListBox                                              3
ListBox1.Size = New System.Drawing.Size(600, 95)                    4
ListBox1.Location = New System.Drawing.Point(104, 24)               5
Me.Controls.Add(ListBox1)                                           6
ListBox1.MultiColumn = True                                         7
Dim list As Integer                                                 8
For list= 1 To 40                                                   9
ListBox1.Items.Add("Items" & list)                                  10
Next list                                                           11
End Sub                                                             12
```

Explanation

The line-by-line explanation of the above mentioned source code is as follows:

Lines 5 and 6
ListBox1.Size = New System.Drawing.Size(600, 95)
ListBox1.Location = New System.Drawing.Point(104, 24)
In these lines, the values 600, 95 are assigned to the **Size** property and the values 104, 24 are assigned to the **Location** property of the **ListBox1** with the help of the namespace **Drawing**.

The output of the above source code is shown in Figure 5-11.

Figure 5-11 *The **listbox_control** form with the output*

Note
*To view all the items given in Figure 5-11, you need to expand the **listbox_control** form.*

CheckedListBox CONTROL

A **CheckedListBox** control is similar to a **ListBox** control except that it contains a check box for each item. You can create a **CheckedListBox** control in the same way as you did in the **ListBox** control. To add a **CheckedListBox** control to a form, double-click on it or simply drag it from the toolbox and drop it on the form. When the **CheckedListBox** control is selected, a small forward arrow will be displayed on the control, as shown in Figure 5-12. When you click on the arrow, the **Edit Items** property will be displayed, as shown in Figure 5-13.

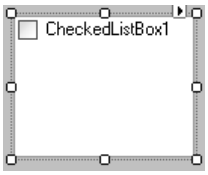

Figure 5-12 *The **CheckedListBox** control*

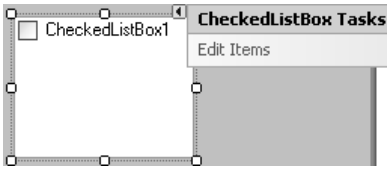

Figure 5-13 *The **CheckedListBox** control with the **Edit Items** property*

When you click on the **Edit Items** property, the **String Collection Editor** dialog box will be displayed, as shown in Figure 5-14. You will add the items that can be displayed in the **CheckedListBox** control using the **String Collection Editor** dialog box.

Figure 5-14 *The* ***String Collection Editor*** *dialog box*

Adding Items to CheckedListBox Controls

The steps required to add items to a **CheckedListBox** control are given below:

Step 1
Select the **Items** property of the **CheckedListBox** control from the **Properties** window; an ellipses button will be displayed on its right. Choose the button; the **String Collection Editor** dialog box will be displayed.

Step 2
Add the items to the **String Collection Editor** dialog box and choose the **OK** button; the items will be added to the **CheckedListBox** control.

During execution, the items in the **CheckedListBox** control appear as they did in the **ListBox** control, except that in this case, they have a check box in front of them. You can select the check boxes by double-clicking on them. But if the value **True** is assigned to the **CheckOnClick** property of the **CheckedListBox** control, then you can select the check boxes with a single-click.

Creating Multicolumn CheckedListBox Control through Source Code

You can create the **CheckedListBox** control with multicolumn using source code. The following source code is required to create a multicolumn **CheckedListBox** control:

```
Dim CheckedListBox1 As CheckedListBox
CheckedListBox1 = New CheckedListBox
CheckedListBox1.Size = New System.Drawing.Size(100, 95)
CheckedListBox1.Location= New System.Drawing.Point(60, 24)
Me.Controls.Add(CheckedListBox1)
Dim list As Integer
For list = 1 To 40
CheckedListBox1.Items.Add("Items" & list, True)
Next list
```

The output of the above source code is the same as that of the previous source code.

*Figure 5-15 The **listbox_ control** form with the new output*

After execution, the **CheckedListBox** control will be displayed, as shown in Figure 5-15.

PictureBox CONTROL

A **PictureBox** control helps you to display the image files such as JPEG, GIF, ico, and so on. To add a **PictureBox** control to a form, double-click on the control or drag and drop it from the toolbox. When you select the **PictureBox** control, a small forward arrow will be displayed on the top right corner of the control. When you click on the arrow, three important properties will be displayed, **Choose Image**, **Size Mode**, and **Dock in parent container**, as shown in Figure 5-16.

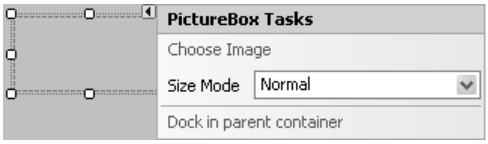

*Figure 5-16 The **PictureBox** control with three properties*

These properties are discussed next.

Choose Image

When you choose the **Choose Image** link; the **Select Resource** dialog box will be displayed, as shown in Figure 5-17. Alternatively, you can select the **Image** property from the **Properties** window. In the **Resource context** area of the **Select Resource** dialog box, there are two radio buttons, **Local resource** and **Project resource file**. The **Project resource file** radio button is selected by default. If you select the **Local resource** radio button, the **Import** button associated with the **Local resource** radio button will get enabled. Choose the **Import** button if you want to import an image from any location to your form. To remove the image, choose the **Clear** button. Note that this option is used to add an image only to the form and not to the project.

If you select the **Project resource file** radio button in this dialog box, the **Import** button associated with the **Project resource file** radio button will get enabled. Choose the **Import** button to add an image to the form and also to your project. You can view the entry of the project in the **Resources** folder of the **Solution Explorer** window.

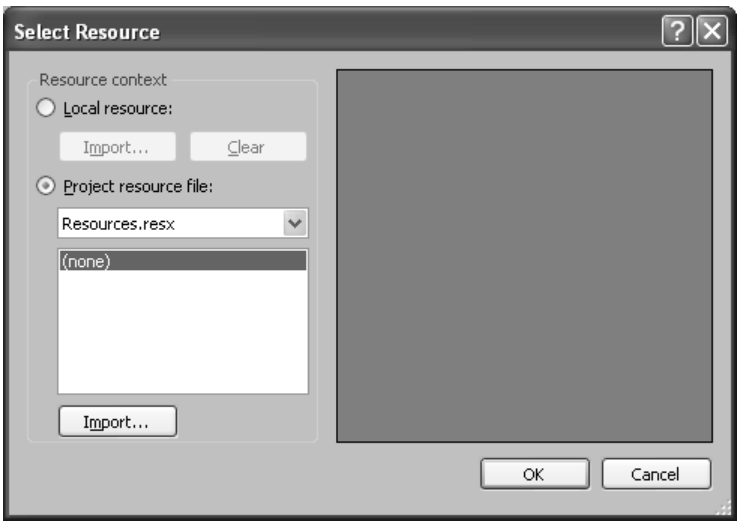

Figure 5-17 *The* ***Select Resource*** *dialog box*

Size Mode

The image can be positioned in a **PictureBox** control with the help of the **Size Mode** property. It has following four members:

Normal

It is the default value of the **Size Mode** property.

StretchImage

This value fits the image to the size of the **Picture Box** control. It means the whole image will be displayed in the **Picture Box** control.

AutoSize

This value displays an image in its original size.

CenterImage

This value of the **Size Mode** property displays the image at the center of the **Picture Box** control. Note that if the size of the **Picture Box** control is small in comparison to the image, then only the center part of the image will be displayed, as shown in Figure 5-18.

Zoom

This value adjusts the size of the image as per the size of the **Picture Box** control.

Figure 5-18 *Image displayed using the* ***Size Mode*** *property of the* ***PictureBox*** *control*

Adding an Image to the PictureBox Control through Source Code

You can add an image to a **PictureBox** control with the help of the source code. You can also set the **Size Mode** property of the **PictureBox** control through the source code. The following source code is required to add an image and set the **Size Mode** property of the **PictureBox** control:

```
Private Sub Form1_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load                    1
PictureBox1.Image = Image.FromFile("C:\Documents and Settings _
\Cadcim\My Documents\My Pictures\cadcimlogo.jpg")                   2
PictureBox1.SizeMode = PictureBoxSizeMode.CenterImage               3
End Sub                                                             4
```

Explanation

The line-by-line explanation of the above given source code is as follows:

Line 2
PictureBox1.Image = Image.FromFile("C:\Documents and Settings\Cadcim \My Documents\My Pictures\cadcimlogo.jpg")
In this line, the path of the *.jpg* file **cadcimlogo** is assigned to the **Image** property of the **PictureBox1**.

Line 3
PictureBox1.SizeMode = PictureBoxSizeMode.CenterImage
In this line, the value **CenterImage** is assigned to the **SizeMode** property of the **PictureBox1**.

PICKERS

VB.NET provides two types of pickers to display a calendar, **DateTimePicker** and **MonthCalendar**.

DateTimePicker Control

This control is used to set the date and time. It has a drop-down list, which displays the date and time. On choosing the down arrow button of this drop-down list, a calendar control will be displayed, as shown in Figure 5-19.

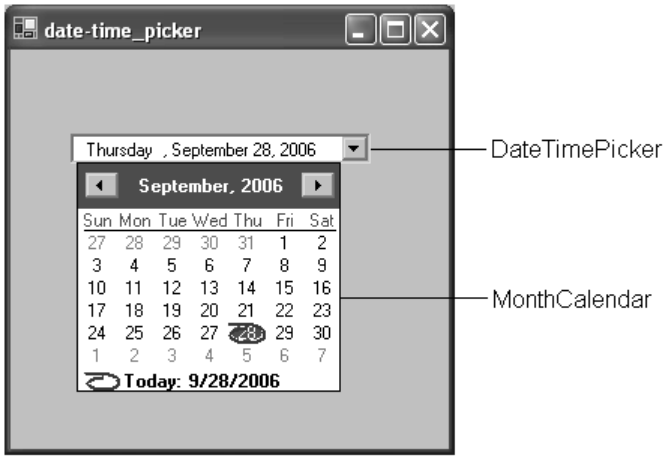

Figure 5-19 *The* ***DateTimePicker*** *control*

The date, month, and year selected by you will become the current value of the drop-down list.

Properties of the DateTimePicker Control

Some important properties of the **DateTimePicker** control are listed as below:

Properties	Functions
CalendarFont	Determines the font used for the calendar
CalendarForeColor	Determines the color of the font used within a month
CalendarMonth Background	Determines the background color within a month

Properties	Functions
CalendarTitle BackColor	Determines the background color of the calendar title
CalendarTitle ForeColor	Determines the foreground color of the calendar title
CalendarTrailing ForeColor	Determines the color of the header and trailing dates
Cursor	Determines the appearance of the cursor when it moves over the control
DropDownAlign	Determines the alignment of the drop-down calendar to the **DateTimePicker** control
Format	Determines the format of the date and time
ShowCheckBox	Determines whether the check box should appear in front of the selected date and time
ShowUpDown	Determines whether the arrows should be used to adjust the date-time values
MaxDate	Determines the maximum selectable date and time
MinDate	Determines the minimum selectable date and time
Value	Determines the current date-time value of the **DateTimePicker** control

Table 5-4 *Properties of the* ***DateTimePicker*** *control*

MonthCalendar Control

This control is used to select a date in a month, refer to Figure 5-19.

Properties of the MonthCalendar Control

Some important properties of the **MonthCalendar** control are listed in Table 5-5.

Properties	Functions
CalendarDimensions	Determines the number of rows and columns in months
FirstDayOfWeek	Sets the first day of the week
MaxSelectionCount	Determines the maximum number of days that can be selected for a month
ScrollChange	Determines the scroll rate
SelectionRange	Determines the selected range of dates of a month calendar control
ShowToday	Determines whether today's date should be displayed at the bottom of the **MonthCalendar** control
ShowTodayCircle	Determines whether today's date should be encircled
ShowWeekNumbers	Determines whether the week numbers should be displayed in front of rows of each day
AnnuallyBoldedDates	Determines the day to be made bold
BoldedDates	Determines the date to be made bold
MonthlyBoldedDates	Determines the monthly date to be made bold

Table 5-5 *Properties of the* ***MonthCalendar*** *control*

TreeView CONTROL

This control is used to create a tree view for displaying child nodes. Figure 5-20 shows a **TreeView** control.

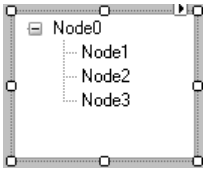

Figure 5-20 *The* ***TreeView*** *control*

Properties of TreeView Control

Some important properties of a **TreeView** control are listed Table 5-6.

Properties	Functions
HotTracking	Determines the appearance of the node when the cursor is placed over it
Indent	Sets the distance between the nodes
ItemHeight	Sets the height of the tree node
LabelEdit	Determines whether the user can edit the nodes
Nodes	Used to get the collection of the tree nodes
Scrollable	Determines whether the scroll bars should be displayed, if the **TreeView** control contains more nodes to fit in the visible area of the control
ShowLines	Determines whether the lines should be displayed between the nodes
ShowPlusMinus	Determines whether the plus and minus signs should be displayed in front of the tree nodes
ShowRootLines	Determines whether the root lines should be displayed between the child nodes

Table 5-6 *Properties of the* ***TreeView*** *control*

Adding a TreeView Control to a Form

You can add a **TreeView** control to a form in the same way as you added the other controls. Double-click on the **TreeView** control; it will be added to the form. To add nodes, select the **Node** property of the control. Next, choose the button on the right of the **Node** property; the **TreeNode Editor** dialog box will be displayed, as shown in Figure 5-21. Choose the **Add Root** button to add the root node and choose the **Add Child** button to add the child nodes. You can remove the nodes using the **Label** edit box of the **TreeNodeEditor** dialog box.

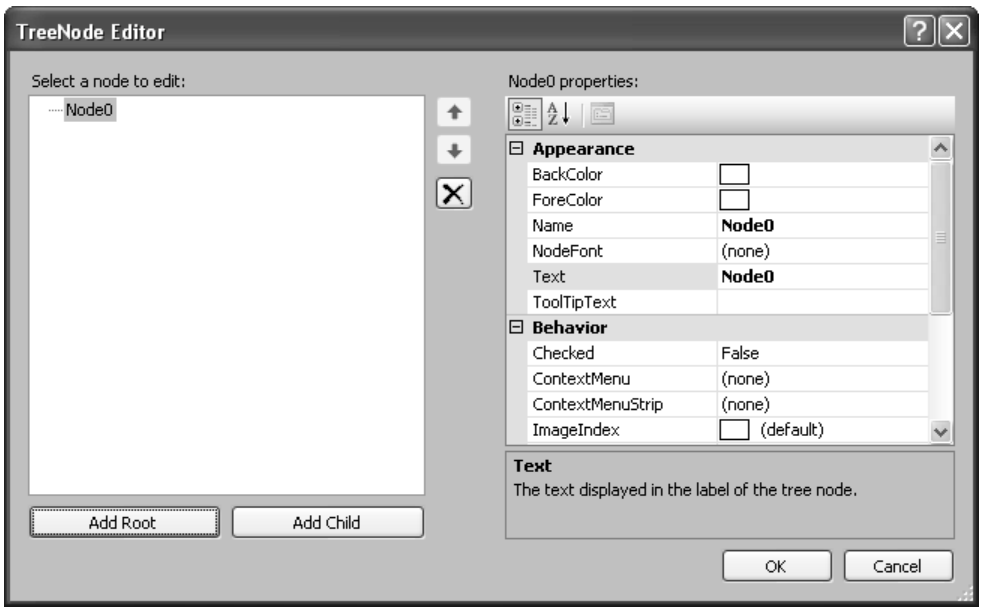

Figure 5-21 *The* ***TreeNode Editor*** *dialog box*

ListView CONTROL

This control is used to display a list of items on the form.

Adding a ListView Control to a Form

The following steps are required to add the **ListView** control to a form:

Step 1
Double-click on the **ListView** control on the toolbox; it will be added to the form. Adjust it according to your requirement.

Step 2
Change the value of its **View** property to **Details** in the **Properties** window.

Step 3
Select the **Columns** property from the **Properties** window and choose the button on its right; the **ColumnHeader Collection Editor** dialog box will be displayed, as shown in Figure 5-22.

Step 4
To add the columns, choose the **Add** button. If you want to edit the **Text** property of these buttons, you can do so by entering the value in the **Text** property of the **ColumnHeader Collection Editor** dialog box and then choosing the **OK** button.

Step 5
To add the items to these columns, select the **Items** property of the **ListView** control.

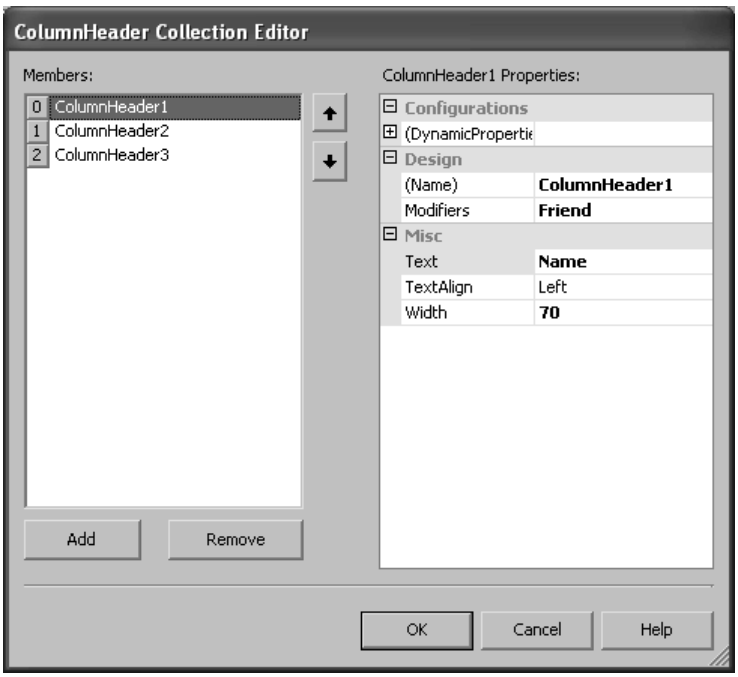

Figure 5-22 *The* ***ColumnHeader Collection Editor*** *dialog box*

Step 6
Choose the button on its right side; the **ListViewItem Collection Editor** dialog box will be displayed, as shown in Figure 5-23.

Step 7
Choose the **Add** button. Enter the value **John** in its **Text** property and then choose the **OK** button; the value **John** will be added to the **ListView** control.

Step 8
To add the sub items, select the **SubItems** property of the **ListViewItem** in the **ListViewItem Collection Editor** dialog box. Choose the button on its right; the **ListViewSubItem Collection Editor** dialog box will be displayed, as shown in Figure 5-24.

Step 9
Choose the **Add** button to add subitems to the **Name** field. Enter the value **Doctor** (for Designation) in the **Text** property. Again, choose the **Add** button and enter the value **876567** (for Contact Number) in the **Text** property; a list will be created, refer to Figure 5-24.

Step 10
To add more items and subitems, repeat the steps from 5 to 9.

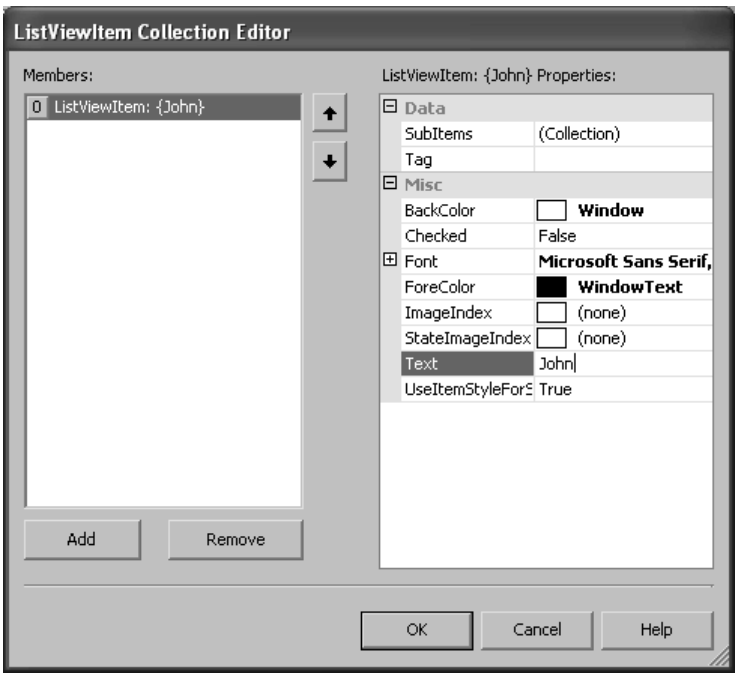

Figure 5-23 *The* ***ListViewItem Collection Editor*** *dialog box*

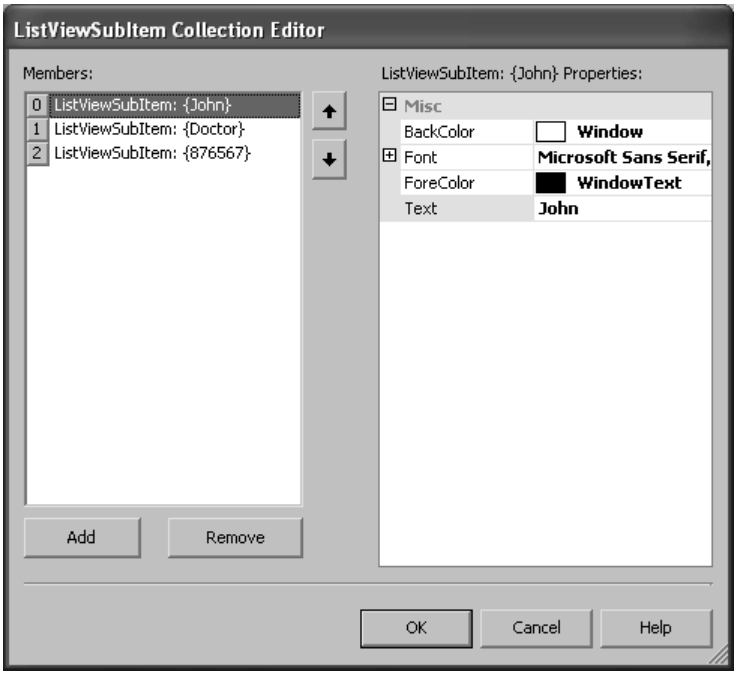

Figure 5-24 *The* ***ListViewSubItem Collection Editor*** *dialog box*

RichTextBox CONTROL

A **RichTextBox** control is similar to a **TextBox** control, except that the scroll bars can be added to it. It can also display fonts, colors, and links. Figure 5-25 shows the **RichTextBox** control.

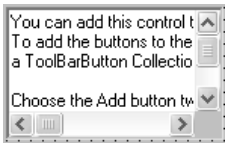

Figure 5-25 *The* ***RichTextBox*** *control*

Adding a RichTextBox Control to a Form

The following steps are required to add a **RichTextBox** control to a form:

Step 1
Double-click on the **RichTextBox** control in the toolbox; it will be added to the form. Alternatively, you can drag it from the toolbox and then drop it on the form.

Step 2
To add multiple lines of text to the **RichTextBox** control, select the **Lines** property from the **Properties** window. Choose the button on the right of the **Lines** property; the **String Collection Editor** dialog box will be displayed, as shown in Figure 5-26. Enter some lines of text in the **String Collection Editor** dialog box and choose the **OK** button; the lines of the text will be added to the **RichTextBox** control. Make sure the value of the **Multiline** property of the **RichTextBox** control is **True**.

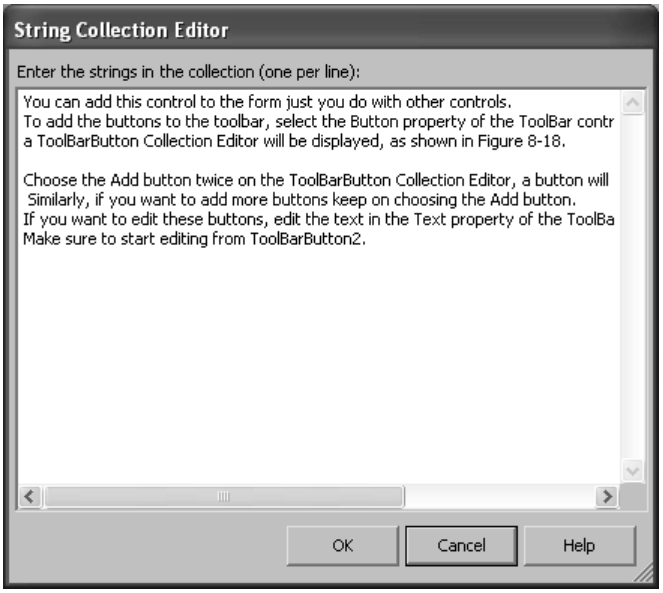

Figure 5-26 *The* ***String Collection Editor*** *dialog box*

Adding Scroll Bars to the RichTextBox Control

If the lines of the text exceed the visible area of the **RichTextBox** control, a vertical scroll bar will be added automatically to it. If you want to add a horizontal scroll bar, then change the value of the **WordWrap** property of the **RichTextBox** control to **False**. To add both scroll bars simultaneously, change the value of the **ScrollBars** property to **Both**.

Creating RichTextBox Control through Source Code

You can create a **RichTextBox** control with the help of source code. Add a **Button** control to the form and double-click on it; the code window will be displayed. Enter the following source code for the **Load** event of the form:

```
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load                          1
Dim RichTextBox1 As New RichTextBox                                       2
RichTextBox1.Size = New Size(144, 88)                                     3
RichTextBox1.Text = "If the lines of the text exceed the visible area of
the TextBox control, a vertical scroll bar will be added automatically to it"  4
RichTextBox1.Location = New Point(28, 32)                                 5
Me.Controls.Add(RichTextBox1)                                             6
End Sub                                                                   7
```

Execute the application; the output of the form will be the same as shown in Figure 5-25.

Button CONTROL

A **Button** control is a small rectangular shaped box that is used to handle events in coding. The default text of the **Button** control is **Button1** and the default event is the **Click** event. It means if you click on the **Button** control, some action will be performed.

To add a **Button** control to a form, double-click on it from the toolbox. Alternatively, you can drag the **Button** control and drop it on the form.

Properties of Button Controls

Some of the important properties of the **Button** control are as follows:

Properties	Functions
BackColor	Sets the background color of the control
Forecolor	Sets the color of the display text in the control
Font	Determines the font used to display the text in the control
FlatStyle	Determines the appearance of the control
Text	Sets the current text in the control
TextAlign	Sets the alignment of the text in the control
Dock	Determines the location of border to be docked to the form

Table 5-7 *Properties of the* ***Button*** *control*

Creating a Button Control through Source Code

The **Button** control can be created through coding. For example, if you want to create a **Button** control of size 100, 30 with the value **Button1** assigned to its **Text** property, then enter the following source code in the code window:

```
Private Sub Form2_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load                1
Dim Button1 As New Button                                       2
Button1.Size = New Size(100, 30)                                3
Button1.Text = "Button1"                                        4
Me.Controls.Add(Button1)                                        5
End Sub                                                         6
```

Explanation

The line-by-line explanation of the above source code is as follows:

Line 2
Dim Button1 As New Button
In the above line, the variable **Button1** is declared as a new instance of the **Button** class.

Line 3
Button1.Size = New Size(100, 30)
In the above line, values 100 and 30 are passed as an argument to the **Size** property of the **Button** control. It means the values for the length and width of the **Button** control are 100 and 30, respectively.

Line 4
Button1.Text = "Button1"
In the above line, the value **Button1** is assigned to the **Text** property of the **Button** control.

Line 5
Me.Controls.Add(Button1)
In the above line, the value **Button1** is passed as an argument to the **Controls.Add** property of the form. Here, **Me** defines the current form. So, it means that a new **Button** control named as **Button1** will be added to the current form.

The **Button** control created through this method is shown in Figure 5-27.

Figure 5-27 *The* ***Button*** *control created through source code*

CheckBox CONTROL

The **CheckBox** control looks similar to a **Label** control with a small square box associated with it. This control is used to select or clear the options from a list of given options. When an option is selected, a tick mark appears in the corresponding small square box, as shown in Figure 5-28.

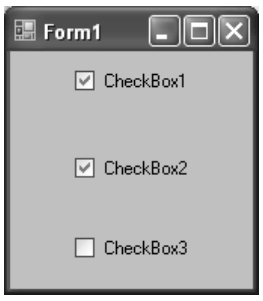

Figure 5-28 *The* ***CheckBox*** *control*

To add this control to a form, double-click on it in the toolbox. Alternatively, you can drag it from the toolbox and drop it on the form.

Properties of the CheckBox Controls

Some of the important properties of the **CheckBox** control are listed in Table 5-8.

Properties	Functions
Appearance	Determines the appearance of the control
CheckAlign	Determines the position of the box inside the control
CheckState	Determines the display of check mark in the box
FlatStyle	Determines the style of appearance of the control

Table 5-8 *Properties of the* ***CheckBox*** *control*

Creating a CheckBox Control through Source Code

The **CheckBox** control can be created using the following source code is as follows:

```
Dim CheckBox1 As New CheckBox                                    1
CheckBox1.Size = New Size(100, 30)                               2
CheckBox1.Text = "CheckBox1"                                     3
CheckBox1.CheckAlign = ContentAlignment.MiddleLeft               4
CheckBox1.FlatStyle = FlatStyle.Standard                         5
CheckBox1.Appearance = Appearance.Normal                         6
CheckBox1.CheckState = CheckState.Unchecked                      7
Me.Controls.Add(CheckBox1)                                       8
```

Explanation

The line-by-line explanation of the above source code is as follows:

Line 1
Dim CheckBox1 As New CheckBox
In the above line, the variable **CheckBox1** is declared as **CheckBox**.

Line 2
CheckBox1.Size = New Size(100, 30)
In this line, values 100 and 30 are assigned to the **Size** property of the **CheckBox** control.

Line 3
CheckBox1.Text = "CheckBox1"
In the above line, the value **CheckBox1** is assigned to the **Text** property of the **CheckBox** control.

Line 4
CheckBox1.CheckAlign = ContentAlignment.MiddleLeft
In the above line, the value **ContentAlignment.MiddleLeft** is assigned to the **CheckAlign** property of the **CheckBox** control, which means the box will appear in the middle-left of the control.

Line 5
CheckBox1.FlatStyle = FlatStyle.Standard
In the above line, the value **Standard** is assigned to the **FlatStyle** property of the **CheckBox** control.

Line 6
CheckBox1.Appearance = Appearance.Normal
In the above line, the value **Normal** is assigned to the **Appearance** property of the **CheckBox** control.

Line 7
CheckBox1.CheckState = CheckState.Unchecked
In the above line, the value **Unchecked** is assigned to the **CheckState** property of the **CheckBox** control. It means the **CheckBox** control will appear unchecked.

Line 8
Me.Controls.Add(CheckBox1)
In the above line, the value **CheckBox1** is passed as an argument to the **Controls.Add** property of the form. Here, **Me** defines the current form. It indicates that a new **CheckBox** control named as **CheckBox1** will be added to the current form.

ComboBox CONTROL

A **ComboBox** control comprises a **TextBox** control associated with a drop-down **ListBox** control. From the drop-down **ListBox** control, you can select any option, and it will be displayed in the edit box. There is a small forward arrow located at the upper right corner of the **ComboBox** control, as shown in Figure 5-29.

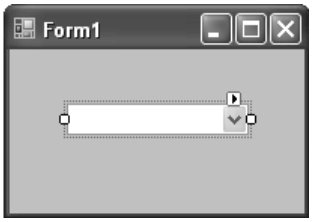

Figure 5-29 *The* ***ComboBox*** *control*

When you click on the arrow, two properties will be displayed, **Use data bound items** and **Edit Items**. When you choose the **Edit Items** property; the **String Collection Editor** dialog box will be displayed. In the **Enter the strings in collection (one per line)** area, you can add the required values for the **ComboBox** control. The **Use data bound items** property will be discussed in later the chapters.

Creating a ComboBox Control through Source Code

You can create a **ComboBox** control through coding. Enter the following source code for the **Load** event of the form:

```
Dim ComboBox1 As New ComboBox
ComboBox1.Size = New Size(100, 30)
ComboBox1.Text = “”
Me.Controls.Add(ComboBox1)
```

The explanation of this coding is same as that of the previous source code.

MaskedTextBox CONTROL

This control is just like a simple **TextBox** control with a specific property. With the help of this property, you can set any format such as **Numeric**, **Phone Number**, **Zip Code**, and so on. You can also add this control to your form as you added the other controls. When you click on the small forward arrow on the upper right corner of the **Masked TextBox** control, the **Set Mask** property will be displayed. If you choose this property; the **Input Mask** dialog box will be displayed, as shown in Figure 5-30.

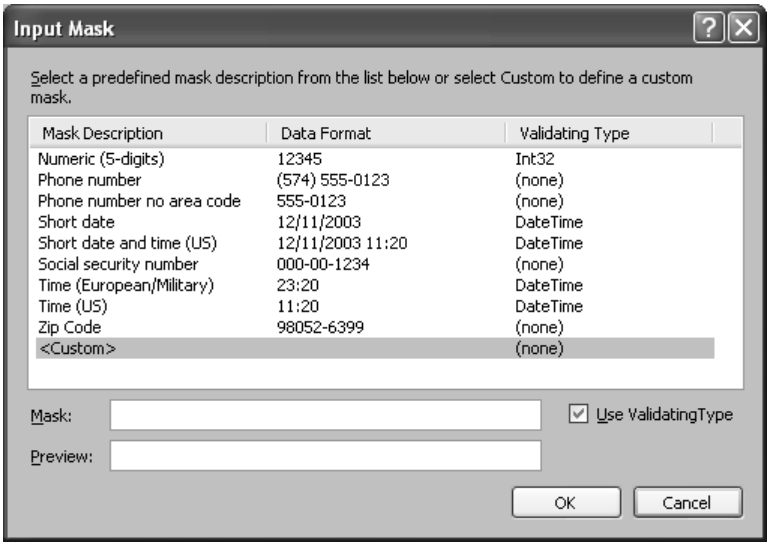

Figure 5-30 *The* ***Input Mask*** *dialog box*

You can set any format for the **Masked TextBox** control using this dialog box according to your requirement. For example, if you want to display phone number in your

MaskedTextBox control, select the **Phone number** option from the dialog box; it will get added to the **Mask** edit box. Also, you can preview its format in the **Preview** edit box, as shown in Figure 5-31. Similarly, you can set other formats also.

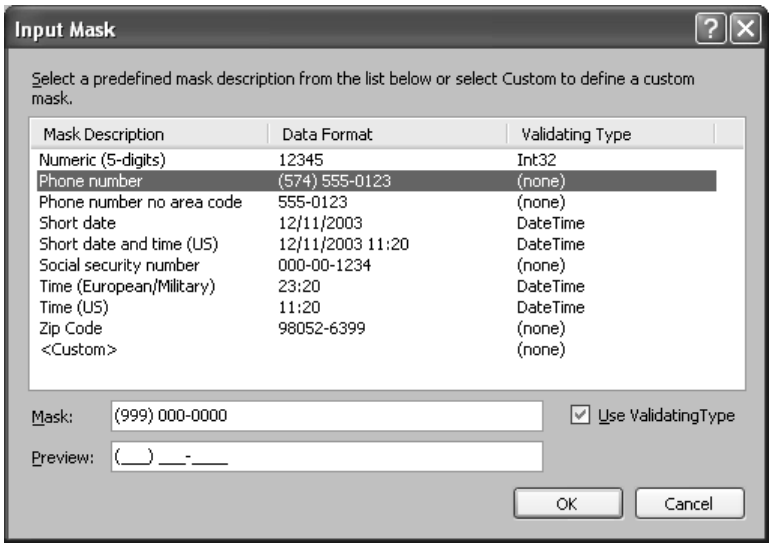

Figure 5-31 *The **Input Mask** dialog box with the **Phone number** option*

NotifyIcon CONTROL

The **NotifyIcon** control is located on the left of the taskbar of your system, as shown in Figure 5-32. This control is used to activate any application to a visible state from its hidden state.

Figure 5-32 *Taskbar with notify icons*

A **NotifyIcon** control can handle both the **Click** and the **Double-click** events. It means, the icon will be activated either by a single-click or by double-click. For example, if you double-click on the **Windows Live Messenger** icon in the taskbar, the **Windows Live Messenger** window will be activated.

To add the **NotifyIcon** control to the form, double-click on it in the toolbar; the **NotifyIcon** control will be added to the component tray.

You can add an icon of your choice in the **Icon** property of the **NotifyIcon** control. To add an icon, select the **Icon** property of the **NotifyIcon** control; an ellipse button will be displayed in front of the property. Choose the button; the **Open** dialog box will be displayed, as shown in Figure 5-33. Choose any of the icons and then choose the **Open** button; the icon will be added to the **Icon** property. When you execute this application, you will notice the icon is added to the taskbar.

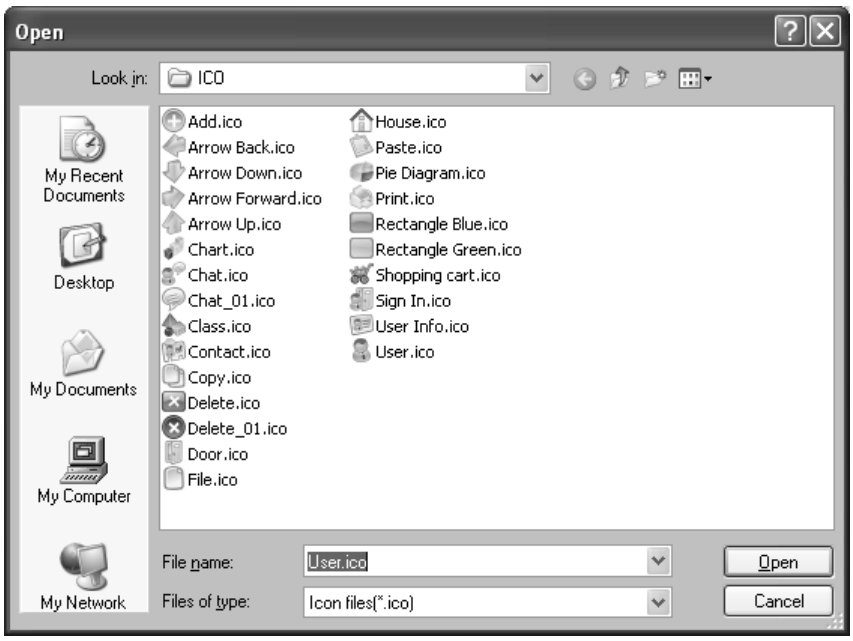

Figure 5-33 *The* ***Open*** *dialog box*

Creating a NotifyIcon

You can create your own **NotifyIcon** with the help of the icon designer. The following steps are required to create a **NotifyIcon**:

1. Choose **Project > Add New Item** from the menu bar; the **Add New Item** dialog box will be displayed. Select the **General** subnode under the **Common Items** node in the **Categories** area.

2. Select the **Icon File** option from the **Templates** area. Next, choose the **Add** button; the icon designer window will be displayed, as shown in Figure 5-34.

3. Now, you can design the icon using the tools displayed in the toolbar that is just above the icon. These tools can be used to draw different shapes, color them, edit them, and so on.

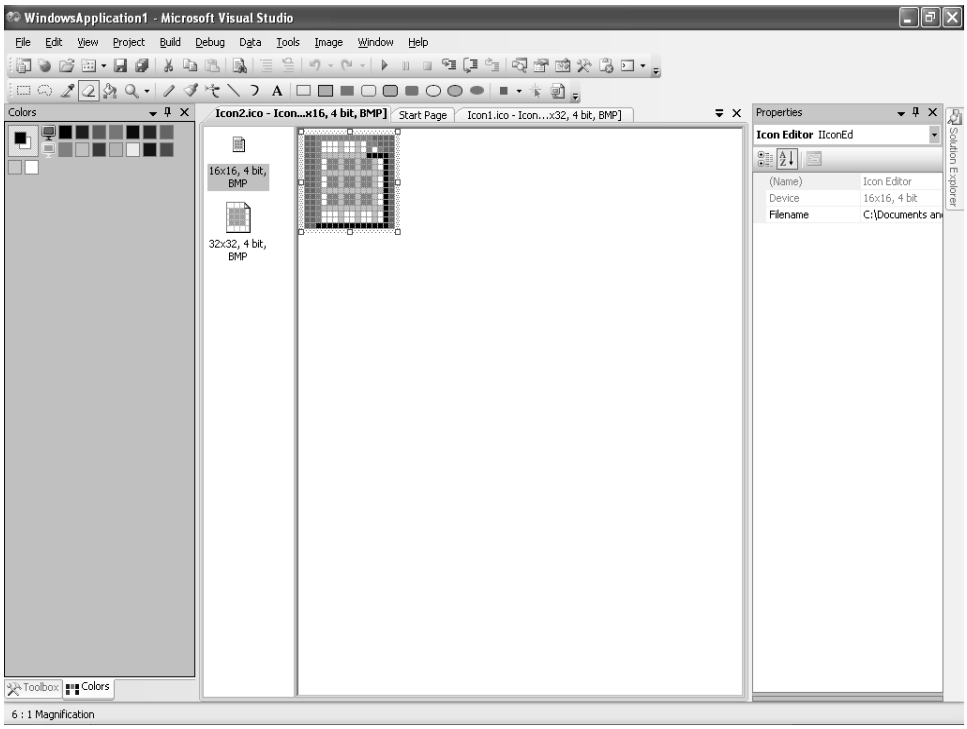

Figure 5-34 *The icon designer window*

NumericUpDown CONTROL

A **NumericUpDown** control is formed by the combination of the **TextBox** control and two arrows, as shown in Figure 5-35. The **TextBox** control is used to display a numeric value and the arrows are used to increase or decrease the value. You can use the **UP** arrow to increase the value and the **DOWN** arrow to decrease the value.

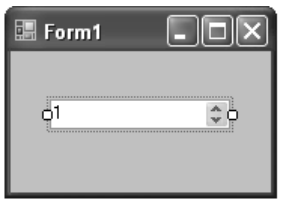

Figure 5-35 *The* ***NumericUpDown*** *control*

The **NumericUpDowm** control has two important properties, **Maximum** and **Minimum**. The default value for the **Minimum** property is 0 and for the **Maximum** property is 100. It means you can specify the values from 0 to 100. You can change the default values if needed.

ProgressBar CONTROL

A **ProgressBar** control is a horizontal bar that is used to indicate the completion of an action by displaying small rectangles in the bar. When the horizontal bar is completely filled with rectangles, it means the action is complete. These small rectangles provide an idea to the user about the progress of the current action. To add a **ProgressBar** control to the form, you need to double-click on it.

Properties of ProgressBar Controls

The important properties of the **ProgressBar** control are listed in Table 5-9.

Properties	Functions
Maximum	Determines the upper range of the **ProgressBar** control within the application
Minimum	Determines the lower range of the **ProgressBar** control within the application
Style	Sets the style of the **ProgressBar** control
Value	Determines the current value of the **ProgressBar** control in the range specified in the **Maximum** and **Minimum** properties
Step	It is an integer value used to increment the value of the **ProgressBar** control

Table 5-9 *Properties of the* ***ProgressBar*** *control*

RadioButton CONTROL

A **RadioButton** control is used to select only one option from the possible multiple options. It is similar to the **CheckBox** control. The only difference is that the **CheckBox** control can be used to select multiple options at a time, whereas the **RadioButton** control can be used to select only one option from the possible multiple options. The **RadioButton** control is shown in Figure 5-36.

Figure 5-36 *The* ***RadioButton*** *control*

Some of the important properties of the **RadioButton** control are given in Table 5-10.

Properties	Functions
Appearance	Sets the appearance of the **RadioButton** control
Checked	Determines whether the **RadioButton** control is selected or not
CheckAlign	Determines the position of the check box within the control
TextAlign	Determines the alignment of the text
AutoClick	Determines the state of the **RadioButton** control on a click

Table 5-10 *Properties of the* ***RadioButton*** *control*

ToolTip CONTROL

Whenever you place the mouse cursor over a control, a **ToolTip** control is displayed. A tooltip is a small rectangular box that gives a short description about the working of the control. Tooltips are used to provide quick help to the user. Whenever you double-click on this control, it is added to the component tray of the form but is not visible at runtime.

Some of the important properties of the **ToolTip** component are given in Table 5-11.

Properties	Functions
AutoPopDelay	Determines the duration for which a tool tip will be displayed on the screen
InitialDelay	Determines the time taken to display the tooltip when you place the cursor over a control
ReshowDelay	Determines the duration of time between the display of different tooltip windows as you move the mouse pointer from one control to another
AutomaticDelay	Sets the values of all the above mentioned properties of the **ToolTip** control to some appropriate values

Table 5-11 *Properties of the* ***ToolTip*** *control*

WebBrowser CONTROL

The **WebBrowser** control provides a wrapper that allows you to display web pages in your windows forms application.

Some important properties of the **WebBrowser** control are listed in Table 5-12.

Properties	Functions
Url	Specifies the **Url** that the **WebBrowser** control has to navigate
AllowNavigation	Specifies whether the **WebBrowser** control can browse to another web page once it is initially loaded

Table 5-10 *Properties of the* ***RadioButton*** *control*

The following application illustrates the use of the **Label** control, **LinkLabel** control, **TextBox** control, and **ListBox** control:

Application 1

Create an application that will prompt the user to enter his name and the book name that he wants to order.

In this application, the user will be prompted to enter his name and the book name that he wants to order. The application will then display the entire details in a message box.

The following steps are required to create this application:

Step 1
Start a new project and save it as **c05_VB_NET_2008_01**.

Step 2
Add three **Label** controls, two **TextBox** controls, a **LinkLabel** control, and a **ListBox** control to the form. The form will be displayed, as shown in Figure 5-37.

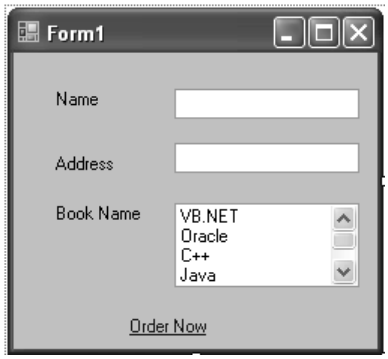

Figure 5-37 *The form after adding the controls*

Step 3
Change the **Text** property of the controls as follows:

Label1 to **Name**
Label2 to **Address**
Label3 to **Book Name**
LinkLabel to **Order Now**

Step 4
Select the **Items** property of the **ListBox** control from the **Properties** window. Choose the ellipse button; the **String Collection Editor** dialog box will be displayed, as shown in

Figure 5-38 *The* ***String Collection Editor*** *dialog box*

Figure 5-38. Add the following items to the editor:

VB.NET	Java
Oracle	Dreamweaver
C++	Excel

Step 5
Double-click on the **LinkLabel** control; the code window will be displayed. Enter the following source code for the **Click** event of the **LinkLabel** control in the code window:

```
MsgBox(" Mr. "  & TextBox1.Text & " has place the " & Chr(13) & "order for the "  &
ListBox1.SelectedItem & " book. "  & Chr(13) & "His mailing address is:" & Chr(13) &
TextBox2.Text, MsgBoxStyle.YesNo)
```

Note that the entire above mentioned code will be entered in a single line.

Explanation

In the above source code, the concatenation operator **&** is used to concatenate the text values of the **TextBox** control and the **ListBox** control. **(13)** is the ASCII value of the ENTER key. The result of this application will be displayed in a message box.

Step 6
Execute the application. To do so, press the F5 key. Alternatively, choose **Debug > Start Debugging** from the menu bar. The final form will be displayed, as shown in Figure 5-39.

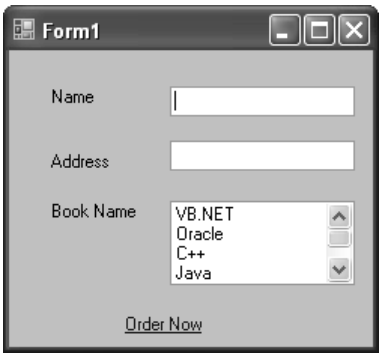

Figure 5-39 *The final form*

Step 7
Enter the required values in the **Name** and **Address** textboxes and then select any one option from the **Book Name** listbox.

Step 8
Next, choose the **Order Now** link; a message box with the complete details of the order will be displayed, as shown in Figure 5-40.

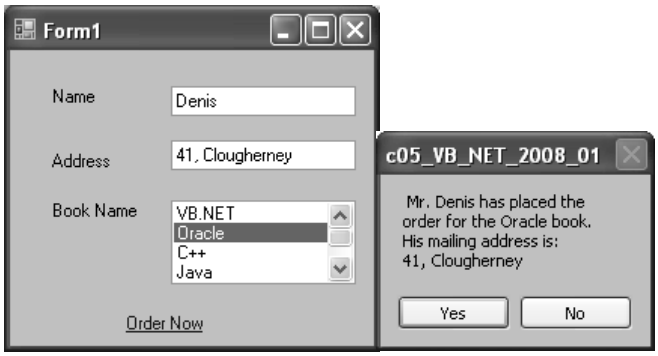

Figure 5-40 *Message box displaying the details of an order*

The following application illustrates the use of the **PictureBox** control, **Button** control, and **CheckedListBox** control:

Application 2

Create an application that will prompt the user to select an option from the existing three options to display a picture.

In the following application, the user will be prompted to select any one option from the multiple options to display a picture.

Step 1
Start a new project and save it as **c05_VB_NET_2008_02**.

Step 2
Add a **Label** control, a **CheckedListBox** control, a **PictureBox** control, and a **Button** control to the form.

Step 3
Change the **Text** property of the controls as follows:
Label1 to **Select the picture to display**
Button1 to **Show Picture**

The resultant form will be shown in Figure 5-41.

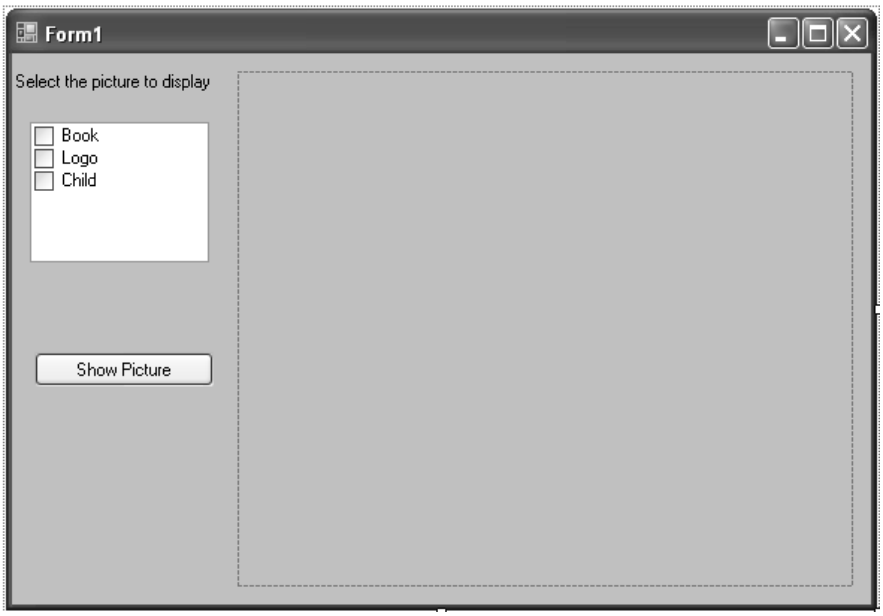

Figure 5-41 *Form after adding controls*

Step 4
Double-click on the **Show Picture** button; the code window will be displayed. Enter the following code for the **Click** event of the **Button** control:

```
Dim FileName As String                                    1
Dim I_Ctr As Integer                                      2
PictureBox1.ImageLocation = FileName                      3
PictureBox1.Show()                                        4
```

Step 5
Double-click on the **CheckedListBox** control and enter the following source code:

```
If CheckedListBox1.SelectedIndex = 0 Then                 5
FileName = "C:\cadcim\small_front_cover_page_vb_net.gif"  6
ElseIf CheckedListBox1.SelectedIndex = 1 Then             7
FileName = "C:\logo-cadcim\logo-5.bmp"                    8
Else                                                      9
FileName = "C:\Documents and Settings\CADCIM\My Documents\
          My Pictures\untitled3.bmp"                      10
End If                                                    11
```

Note

You need to specify the paths for the images in the above source code according to location of the files on your system.

Explanation

The line-by-line explanation of the above source code is as follows:

Line 1
Dim FileName As String
In the above line, the variable **FileName** is declared as **String**.

Line 2
Dim I_Ctr As Integer
In the above line, **I_Ctr** is declared as an **Integer**.

Line 3
PictureBox1.ImageLocation = FileName
In the above line, the value of the variable **FileName** is assigned to the **ImageLocation** property of the **PictureBox** control.

Line 4
PictureBox1.Show()
In the above line, **Show()** is the method of the **PictureBox** control. The **Show()** method is used to call the image files to be displayed in the **PictureBox** control.

Line 5
If CheckedListBox1.SelectedIndex = 0 Then
In the above line, if the value of the **SelectedIndex** property of the **CheckedListBox** control is 0, then the control will be transferred to the next line (line 6) and the image mentioned in that line will be displayed.

Line 6
FileName = "C:\cadcim\small_front_cover_page_vb_net.gif"
In the above line, the path **C:\cadcim\small_front_cover_page_vb_net.gif** is assigned to the variable **FileName**.

Line 7
ElseIf CheckedListBox1.SelectedIndex = 1 Then
If the statement in the line 5 is not evaluated to **True**, then the control will be transferred to this line. In this line, if the value of the **SelectedIndex** of the **CheckedListBox** control is 1, then the control will be transferred to the next line (line 8) and the image mentioned in the line will be displayed.

Line 8
FileName = "C:\logo-cadcim\logo-5.bmp"
In the above line, the path **C:\logo-cadcim\logo-5.bmp** is assigned to the variable **FileName**.

Line 9
Else
The **Else** statement in this line is used for the **If** statement in line 5. If both the conditions within the body of the **If** and the **ElseIf** statement are evaluated to false, then the control will be transferred to this line. It means the statement within the **Else** block will be executed.

Line 10
FileName = "C:\Documents and Settings\CADCIM\My Documents
My Pictures\untitled3.bmp"
In the above line, the path **C:\Documents and Settings\CADCIM\My Documents\My Pictures\untitled3.bmp** of the image is assigned to the variable **FileName**.

Line 11
End If
The above line indicates the end of the **If** statement.

Step 6
Execute the application. To do so, press the F5 key. Alternatively, choose **Start > Start Debugging** from the menu bar. If you select the **Child** check box and then choose the **Show Picture** button, the image of the child will be displayed, as shown in Figure 5-42.

Figure 5-42 *The output form of Application 2*

The following application illustrates the use of the **DateTimePicker** control, **RadioButton** control, **ComboBox** control, and **RichTextBox** control:

Application 3

Create an application that will prompt the user to enter his personal details.

In this application, the user will be prompted to enter his/her personal details using different controls.

The following steps are required to create this application:

Step 1
Start a new project and save it as **c05_VB_NET_2008_03**.

Step 2
Add seven **Label** controls, two **TextBox** controls, a **RichTextBox** control, two **RadioButton** controls, two **CheckBox** controls, and a **DateTimePicker** control to the form. Arrange the controls, as shown in Figure 5-43.

Step 3
Change the **Text** property of the controls as follows:

Label1 to **Name**
Label2 to **Date of Birth**
Label3 to **Designation**
Label4 to **Gender**
Label5 to **Phone No**
Label6 to **Tell about yourself**
Label7 to **About 500 characters**
Button1 to **Submit**
RadioButton1 to **Male**
RadioButton2 to **Female**
CheckBox1 to **Residential**
CheckBox2 to **Office**

Figure 5-43 *Design mode of Application 3*

Step 4
Change the **Name** property of the controls as follows:

TextBox1 to **TxtName**
TextBox2 to **TxtPhone**
ComboBox1 to **CmbRole**
RadioButton1 to **RbMale**
RadioButton2 to **Rbfemale**
Button1 to **Brn_Submit**

Step 5
Double-click on the **DateTimePicker** control; the code window will be displayed. Enter the following source code:

TextBox1.Text = DateTimePicker1.Value
In the above source code, the value of the **DateTime Picker** control is assigned to the **Text** property of the associated **TextBox** control.

Step 6
Double-click on the **Submit** button and enter the following source code in the code window:

MsgBox("Thanks for submitting details to CADCIM Technologies, USA", , "CADCIM Technologies")
In the above source code, a message box with the message **"Thanks for submitting details to CADCIM Technologies, USA",** , **"CADCIM Technologies** will be displayed whenever the user chooses the button.

Step 7
Execute the application; the output form will be displayed, as shown in Figure 5-44. Enter the details as follows:

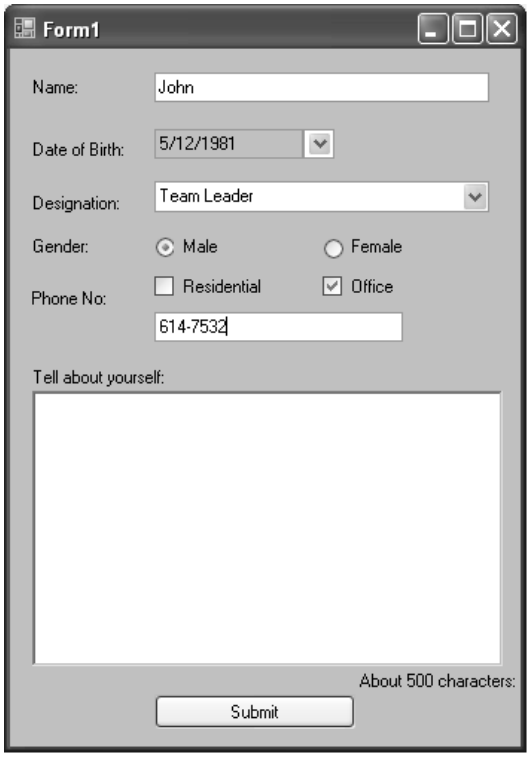

Figure 5-44 *The output form of Application 3*

Name: John
Date of Birth: 05/12/81
Designation: Team Leader
Gender: Male
Phone No: 614-7532

Next, choose the **Submit** button; a message box will be displayed, as shown in Figure 5-45.

Figure 5-45 *Message box displayed*

Self-Evaluation Test

Answer the following questions and then compare them to those given at the end of this chapter:

1. Sometimes you do not want the user to change the value of the **Text** property of the **TextBox** control. To do so, the value of the _____ property should be set to **True**.

2. You can align the text in the **TextBox** control using the _____ property.

3. The _____ property is used to set a portion of the text as hyperlink.

4. The _____ property is used to set the color of the hyperlink when the user clicks on it.

5. The _____ property is used to set the drawing mode of a **ListBox** control.

Review Questions

Answer the following questions:

1. The image can be positioned in the **PictureBox** control with the help of the _____ property.

2. _____ is the default value of the **Size Mode** property of a **PictureBox** control.

3. A _____ is displayed in a small rectangular box that gives a short description about the working of the control.

4. The _____ control is used to set the date and time.

5. The _____ property determines whether today's date should be displayed at the bottom of the **MonthCalendar** control.

Exercise

Exercise 1

Create an application that will prompt the user to enter his username and password in a form. When the user chooses the **OK** button on the form, the application should verify the username and password as **Sam** and **computer**, respectively. If the username and password entered by the user are correct, a registration form, similar to Figure 5-46, will be displayed. After entering the details, when the user chooses the **OK** button on the registration form, the application will display a message box with the message **Form submitted**.

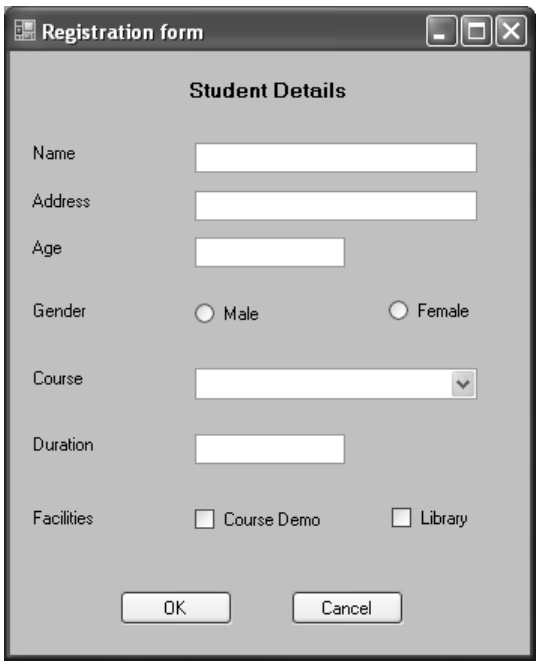

Figure 5-46 *Output form of Exercise 1*

Answers to Self-Evaluation Test
1. ReadOnly, 2. TextAlign, 3. LinkArea, 4. ActiveLinkColor, 5. DrawMode