# 9

# Additional Controls and Objects


**9.1 List Boxes and Combo Boxes 402**

- A Review of List Box Features
- Some Additional Features of List Boxes
- The Combo Box Control
- A Helpful Feature of Combo Boxes

**9.2 Eight Additional Controls and Objects 410**

- The Timer Control
- The Random Class
- The ToolTip Control
- The Clipboard
- The Picture Box Control
- The MenuStrip Control
- The Horizontal and Vertical Scroll Bar Controls

**9.3 Multiple-Form Programs 425**

- Startup Form
- Scope of Variables, Constants, and Procedures
- Modality
- Close and ShowDialog Methods
- The FormClosing Event Procedure
- Importing an Existing Form
- Deleting a Form from a Program

**9.4 Graphics 437**

- Graphics Objects
- Lines, Rectangles, Circles, and Sectors
- Pie Charts
- Bar Charts
- Animation
- Printing Graphics

**Summary 450**

**Programming Projects 451**

## 9.1 List Boxes and Combo Boxes

The **list box** and **combo box** controls allow the user to make selections by clicking on an item. Certain styles of combo boxes also allow for input by assisted typing.

### A Review of List Box Features

A list box can be populated at design time with the String Collection Editor. Items are typed directly into the editor or copied (with Ctrl + C) from another application like Excel, Word, or Notepad and pasted (with Ctrl + V) into the editor. A list box can be populated at run time with the Items.Add method or by setting its DataSource property to an array or a query (converted to a list).

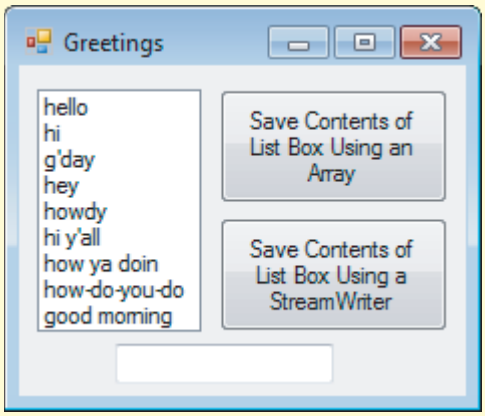
The items in a list box are indexed with zero-based numbering. That is, the items in a list box are identified as lstBox.Items(0), lstBox.Items(1), and so on. Table 9.1 summarizes the properties, methods, and events for list boxes that were presented in earlier chapters.

**TABLE 9.1 Previously discussed properties, methods, and events.**

| | |
|---|---|
| lstBox.Items.Add(*value*) | Method: Insert the value into the list box. |
| lstBox.Items.Clear() | Method: Remove all items from the list box. |
| lstBox.Text | Property: The selected item as a string. |
| lstBox.Items.Count | Property: The number of items in the list box. |
| lstBox.Sorted | Property: If set to True, items will be displayed in ascending ANSI order. |
| lstBox.SelectedIndex | Property: The index of the selected item. If no item is selected, the value is −1. |
| lstBox.SelectedItem | Property: The currently selected item. It must be converted to a string before being displayed in a text or message box. |
| lstBox.Items(*n*) | Property: The item having index *n*. It must be converted to a string before being displayed in a text or message box. |
| lstBox.DataSource | Property: The source of data to populate the list box. |
| lstBox.SelectedIndexChanged | Event: Occurs when the value of the SelectedIndex property changes. It is the default event procedure. |
| lstBox.Click | Event: Occurs when the user clicks on the list box. |
| lstBox.DoubleClick | Event: Occurs when the user double-clicks on the list box. |

***Note:*** You can programmatically change the selected (that is highlighted) item by changing the SelectedIndex value in code: the corresponding item in the list box will appear highlighted.

**Example 1** The following program shows two ways to copy the contents of a list box into a text file. When the top button is clicked, the items in the list box populate an array that is then used to create a text file. When the bottom button is clicked, a StreamWriter is used to copy the contents of the list box directly into a text file. Notice that the StreamWriter does not have to convert the items in the list box to strings before copying them into the file.




| OBJECT | PROPERTY | SETTING |
|---|---|---|
| frmGreetings | Text | Greetings |
| lstBox | Items | (shown in screen capture) |
| btnArray | Text | Save Contents of List Box Using an Array |
| btnSW | Text | Save Contents of List Box Using a StreamWriter |

```
Private Sub btnArray_Click(...) Handles btnArray.Click
  Dim ub As Integer = lstBox.Items.Count - 1   'upper bound of array
  Dim a(ub) As String
  For i As Integer = 0 To ub
    a(i) = CStr(lstBox.Items(i))
  Next
  IO.File.WriteAllLines("Greetings1.txt", a)
End Sub

Private Sub btnSW_Click(...) Handles btnSW.Click
  Dim sw As IO.StreamWriter = IO.File.CreateText("Greetings2.txt")
  For i As Integer = 0 To lstBox.Items.Count - 1
    sw.WriteLine(lstBox.Items(i))
  Next
  sw.Close()
End Sub
```

[Run, and click on each of the two buttons. Then end the program, click on the *Refresh* button in the Solution Explorer window, click on the *View All Files* button, and look at the new text files in the *bin\Debug* subfolder. Each text file will contain the contents of the list box.]

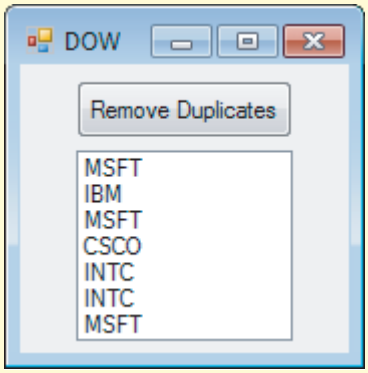
## Some Additional Features of List Boxes

Table 9.2 shows some additional useful methods for list boxes.

**TABLE 9.2 Some additional list box methods.**

| | |
|---|---|
| lstBox.Items.IndexOf(*value*) | Method: Index of the first item to have the value. |
| lstBox.Items.RemoveAt(*n*) | Method: Delete item having index *n*. |
| lstBox.Items.Remove(*strValue*) | Method: Delete first occurrence of the string value. |
| lstBox.Items.Insert(*n*, *value*) | Method: Insert the value as the item of index *n*. |

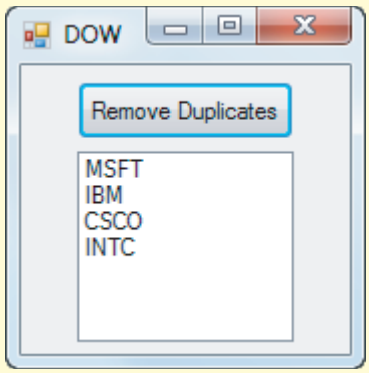
**Example 2** The following program removes all duplicates from a list box. The program looks at the first item in the list box and removes every matching item that follows it. The program then repeats the process with the next item remaining in the list box, and so on.




| OBJECT | PROPERTY | SETTING |
|---|---|---|
| frmDOW | Text | DOW |
| btnRemove | Text | Remove Duplicates |
| lstDOW | Items | (shown in screen capture) |

```
Private Sub btnRemove_Click(...) Handles btnRemove.Click
  Dim i As Integer = 0, j As Integer = 0
  Do While i < (lstDOW.Items.Count - 1)
    j = i + 1
    Do While j < lstDOW.Items.Count
      If CStr(lstDOW.Items(j)) = CStr(lstDOW.Items(i)) Then
        lstDOW.Items.RemoveAt(j)
      Else
        j += 1
      End If
    Loop
    i += 1
  Loop
End Sub
```

[Run, and then click on the button.]




**VideoNote**
List boxes and combo boxes

## The Combo Box Control

A combo box control can be thought of as a text box with a list box attached to it. Combo boxes have all the properties, methods, and events that list boxes have, plus a few more. The three different styles of combo box are shown in Fig. 9.1. (Each combo box's String Collection Editor was used to populate the combo box with the names of the seven continents.) The style of a combo box control is specified by setting its DropDownStyle property to Simple, DropDown, or DropDownList. In a Simple style combo box the list box is always visible. With the other two styles, the list box appears only at run time when the user clicks on the down-arrow button. See Fig. 9.2. The standard prefix for the name of a combo box is *cbo*.

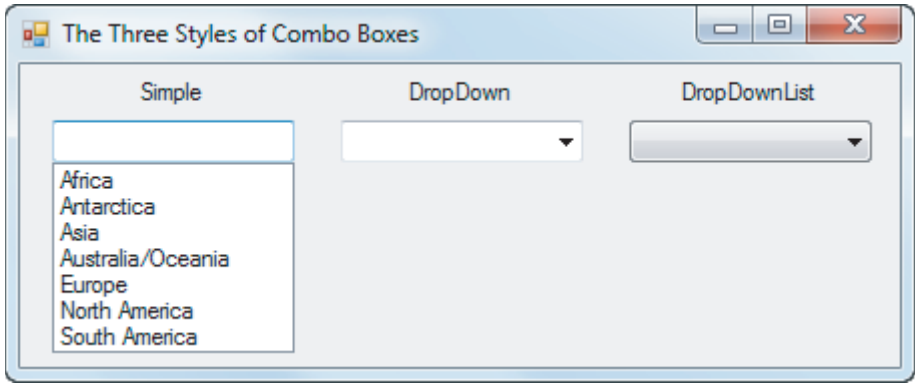



**FIGURE 9.1** **The three settings for the DropDownStyle property.**

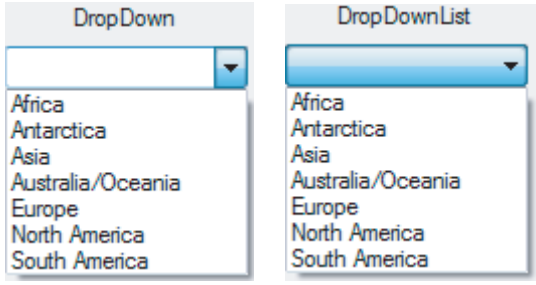


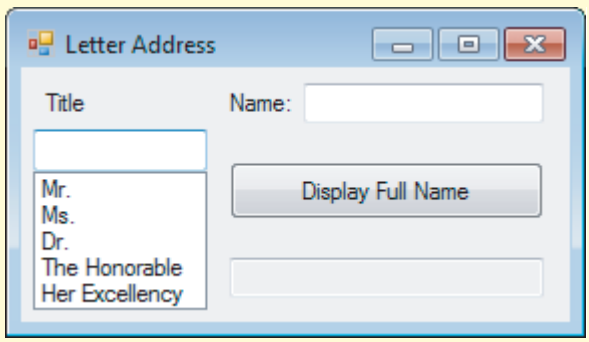

**FIGURE 9.2** **Combo boxes at run time after their down-arrows are clicked.**

With a Simple or DropDown combo box, the user can fill the text box either by typing directly into it or by selecting an item from the list. With a DropDownList style combo box, the user can fill the text box only by selecting an item from the list. (DropDown is the default setting of a combo box's DropDownStyle property.) A list that has dropped down disappears when the user clicks on an item or presses the Enter key. With any of the three styles, the value of

```
cboBox.Text
```

is the contents of the text box at the top of the combo box. Just like list boxes, combo boxes can be populated with their String Collection Editor, the Items.Add method, and the DataSource property.

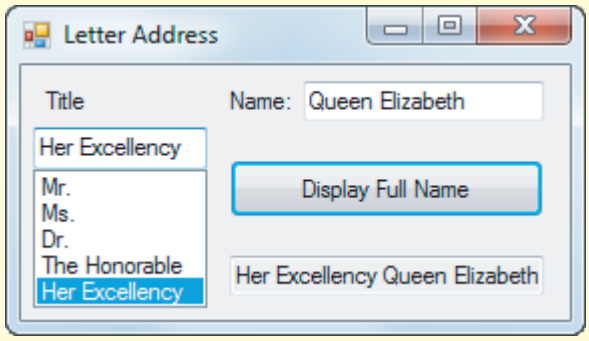
**Example 3** The following program uses a Simple combo box to obtain a person's title for the first line of the address of a letter.




| OBJECT | PROPERTY | SETTING |
|---|---|---|
| frmAddress | Text | Letter Address |
| lblTitle | Text | Title |
| cboTitle | Items | (shown in screen capture) |
| | DropDownStyle | Simple |
| lblName | Text | Name: |
| txtName | | |
| btnDisplay | Text | Display Full Name |
| txtDisplay | ReadOnly | True |

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  txtDisplay.Text = cboTitle.Text & " " & txtName.Text
End Sub
```

[Run, select an item from the combo box, type a name into the Name text box, and click on the button.]




The same program with a DropDown style combo box produces the form shown in Fig. 9.3. The form will look about the same when a DropDownList style combo box is used.

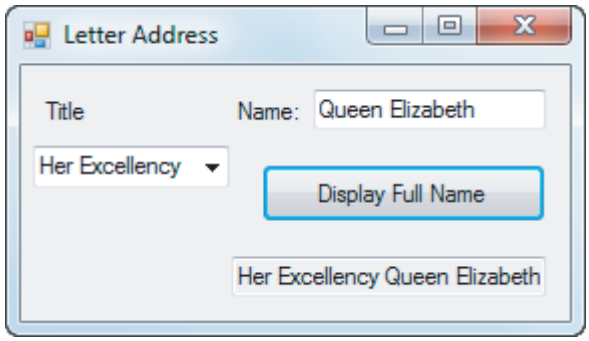



**FIGURE 9.3** **Example 3 with a DropDown style combo box.**

**Example 4** The following variation of Example 3 uses the same form design and settings as Example 3 except that the combo box is not filled at design time.

```
Private Sub frmAddress_Load(...) Handles MyBase.Load
  Dim titles() As String = {"Mr.", "Ms.", "Dr.",
                            "The Honorable", "Her Excellency"}
  'Fill combo box with elements from array
  cboTitle.DataSource = titles
End Sub

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  txtDisplay.Text = cboTitle.Text & " " & txtName.Text
End Sub
```

## A Helpful Feature of Combo Boxes

The file Nations.txt contains the names of the 192 members of the United Nations. Suppose a Simple style combo box has been filled with the nations in alphabetical order. Figure 9.4(a) shows the combo box when the program starts. Each time a letter is typed into the text box at the top of the combo box the list scrolls in a helpful way. For instance, after the letter *N* is typed, the combo box appears as shown in Fig. 9.4(b). The combo box has scrolled down so that the first nation beginning with the letter *N* is at the top of the list. Similarly, typing additional letters causes the list to scroll down so that the first nation beginning with the typed letters is at the top of the list. See Figs. 9.4(c) and 9.4(d). (***Notes:*** When the scroll box reaches the bottom of the scroll bar, no further scrolling will take place. Also, if the typed letters do not correspond to any nation, the list will return to the state in Fig. 9.4(a). Analogous results apply to DropDown style combo boxes.)

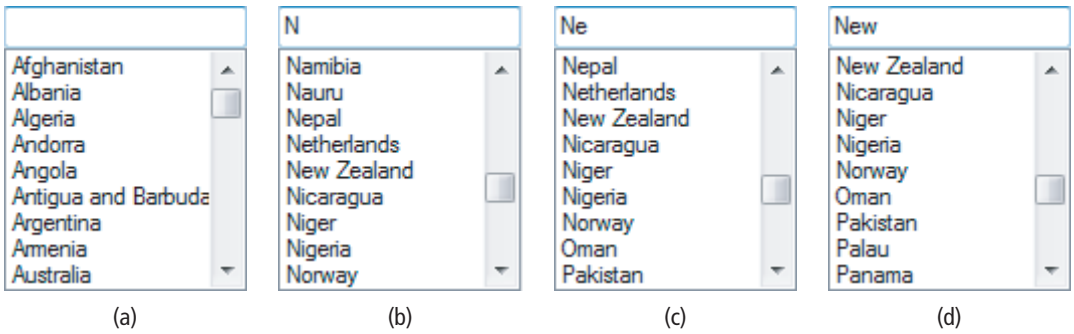



**FIGURE 9.4** Successive combo box displays.

## Practice Problems 9.1

1. Will the following two statements always have the same effect?

```
lstOxys.Items.RemoveAt(lstOxys.SelectedIndex)
lstOxys.Items.Remove(lstOxys.Text)
```

## EXERCISES 9.1

**In Exercises 1 through 8, determine the effect of the code on the list box lstBox shown below. (Assume that the Sorted property is set to True.)**


1. `lstBox.Items.Remove("Chopin")`
2. `lstBox.Items.RemoveAt(0)`
3. `lstBox.Items.RemoveAt(lstBox.SelectedIndex)`
4. `lstBox.Items.RemoveAt(lstBox.Items.Count - 1)`
5. `lstBox.Items.Add("Hayden")`

6.
```
Dim total As Integer = 0
For i As Integer = 0 To lstBox.Items.Count - 1
  If CStr(lstBox.Items(i)).Length = 6 Then
    total += 1
  End If
Next
txtOutput.Text = CStr(total)
```

7.
```
Dim highestIndex As Integer = lstBox.Items.Count - 1
Dim composers(highestIndex) As String
For i As Integer = 0 To highestIndex
  composers(i) = CStr(lstBox.Items(i))
Next
lstBox.Items.Clear()
lstBox.Sorted = False
For i As Integer = highestIndex To 0 Step -1
  lstBox.Items.Add(composers(i))
Next
```

8.
```
Dim highestIndex As Integer = lstBox.Items.Count - 1
Dim composers(highestIndex) As String
For i As Integer = 0 To highestIndex
  composers(i) = CStr(lstBox.Items(highestIndex - i))
Next
lstBox.Sorted = False
lstBox.DataSource = composers
```

**In Exercises 9 through 16, assume that cboBox has DropDownStyle set to Simple, appears as shown below, and has its Sorted property set to True. Give a statement or statements that will carry out the stated task. (The statements should do the job even if additional items have been added to the list.)**


9. Highlight the name Dante.
10. Highlight the third item of the list.
11. Delete the name Shakespeare.
12. Delete the name Goethe.
13. Delete the last name in the list.
14. Display every other item of the list in another list box.
15. Delete every item beginning with the letter M.
16. Determine if Cervantes is in the list.
17. The file PopularNames.txt contains the 20 most popular names given to newborns in a recent year. Write a program that uses a list box to sort the names into alphabetical order and then places the alphabetized list into a new ordered text file.
18. Rework the program in Example 2 using LINQ.

**19.** Suppose all the items in lstBox are numbers. Write a program to display them in lstBox in increasing numerical order. ***Note:*** Setting `lstBox.Sorted = True` will not do the job.

**20.** Suppose all the items in lstBox are words. Write a program to display them in lstBox in decreasing alphabetical order.

**21.** Write a program that contains a list box (with Sorted = False), a label, and two buttons captioned *Add an Item* and *Delete an Item*. When the *Add an Item* button is clicked, the program should request an item with an input dialog box and then insert the item above the currently highlighted item. When the *Delete an Item* button is clicked, the program should remove the highlighted item from the list. At all times, the label should display the number of items in the list.

**22.** Consider the Length Converter in Fig. 9.5. Write a program to carry out the conversion. (See the first programming project in Chapter 7 for a table of equivalent lengths.)

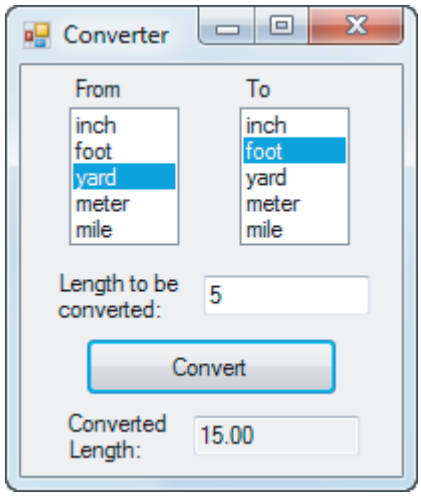



**FIGURE 9.5** **Possible output for Exercise 22.**

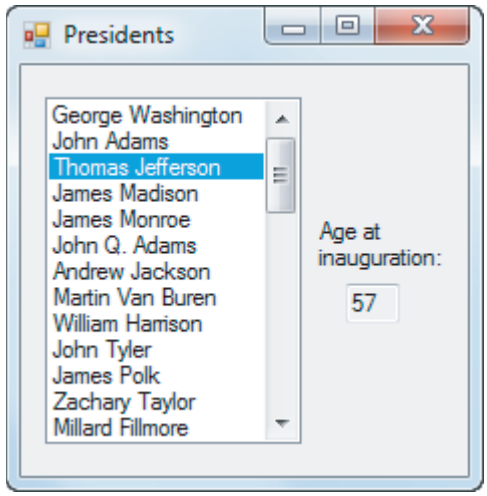



**FIGURE 9.6** **Possible output for Exercise 23.**

**23.** The file AgesAtInauguaral.txt gives the ages at inauguration of the first 44 U.S. presidents. The first four lines of the file contain the data 57, 61, 57, 57; the ages of Washington, Adams, Jefferson, and Madison at their inaugurations. The file USPres.txt contains the names of the first 44 United States presidents in the order they were inaugurated. Write a program that places the names of the presidents into an unsorted list box and the ages at inauguration into an array. When the user clicks on the name of a president, his age at inauguration should be displayed in a text box. (***Hint:*** A president's index number in the list box will be the same as the index number of his age at inauguration in the array.) See Fig. 9.6.

**24.** Write a program to ask a person which Monopoly® space he or she has landed on and then display the result in a text box. The response should be obtained with a combo box listing the squares most commonly landed on: Park Place, Illinois Avenue, Go, B&O Railroad, and Free Parking. (One possible outcome to be displayed in the text box is "You have landed on Park Place.")

**25.** Write a program to question a person about his or her computer and then display a descriptive sentence in a text box. The form should contain combo boxes for brand, amount of memory, and screen size. The lists should contain the most common responses for each category. Some common computers are Compaq, Dell, Hewlett Packard, Lenovo, and Apple. The most common amounts of memory are 1 GB, 2 GB, and 4 GB. The most common

screen sizes are 17, 19, 20, and 24 inches. (One possible outcome to be displayed in the text box is "You have a Dell computer with 2 GB of memory and a 17-inch monitor.")

**Solutions to Practice Problems 9.1**

1. Yes, if all the items in the list box are distinct. However, if an item is repeated and the second occurrence is selected, then the first statement will delete that item, whereas the second statement will delete the earlier occurrence of the item.

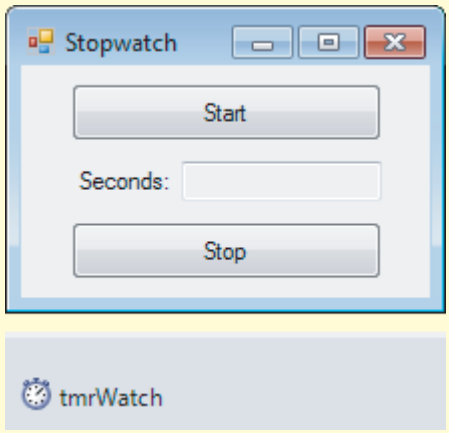
## 9.2 Eight Additional Controls and Objects

### The Timer Control

VideoNote
Additional controls

The **timer** control, which is not visible on the form during run time, raises an event after a specified amount of time has passed. (The timer control is found only in the *All Windows Forms* group of the Toolbox. When you double-click on the timer control in the Toolbox, it appears in the **component tray**, at the bottom of the Form Designer.) The length of time, measured in milliseconds, is set with the Interval property to be any integer from 1 to 2,147,483,647 (about 596 hours). The event raised each time Timer1.Interval milliseconds elapses is called Timer1.Tick. In order to begin timing, a timer must first be turned on by setting its Enabled property to True. A timer is turned off by setting its Enabled property to False. The standard prefix for the name of a timer control is *tmr*.

**Example 1** The following program creates a stopwatch that updates the time every tenth of a second.




| OBJECT | PROPERTY | SETTING |
|---|---|---|
| frmStopwatch | Text | Stopwatch |
| btnStart | Text | Start |
| lblSeconds | Text | Seconds: |
| txtSeconds | ReadOnly | True |
| btnStop | Text | Stop |
| tmrWatch | Interval | 100 |

```
Private Sub btnStart_Click(...) Handles btnStart.Click
  txtSeconds.Text = "0"     'Reset watch
  tmrWatch.Enabled = True
End Sub

Private Sub btnStop_Click(...) Handles btnStop.Click
  tmrWatch.Enabled = False
End Sub

Private Sub tmrWatch_Tick(...) Handles tmrWatch.Tick
  'Next line displays the time rounded to one decimal place
  txtSeconds.Text = CStr((CDbl(txtSeconds.Text) + 0.1))
End Sub
```