altered, if necessary, to look exactly like the name printed on the credit card. The list for the Year combo box should be filled by the Load event procedure and should contain the current year followed by the next five years. (***Note:*** The current year is given by Today.Year.) After the user provides the requested data, the information is displayed in the list box of frmCustomer as shown in Fig. 9.28.

12. Write a program containing the two forms shown in Fig. 9.30. Initially, the Number to Dial form appears. When the *Show Push Buttons* button is clicked, the Push Buttons form appears. The user enters a number by clicking on successive push buttons and then clicking on *Enter* to have the number transferred to the read-only text box at the bottom of the first form.

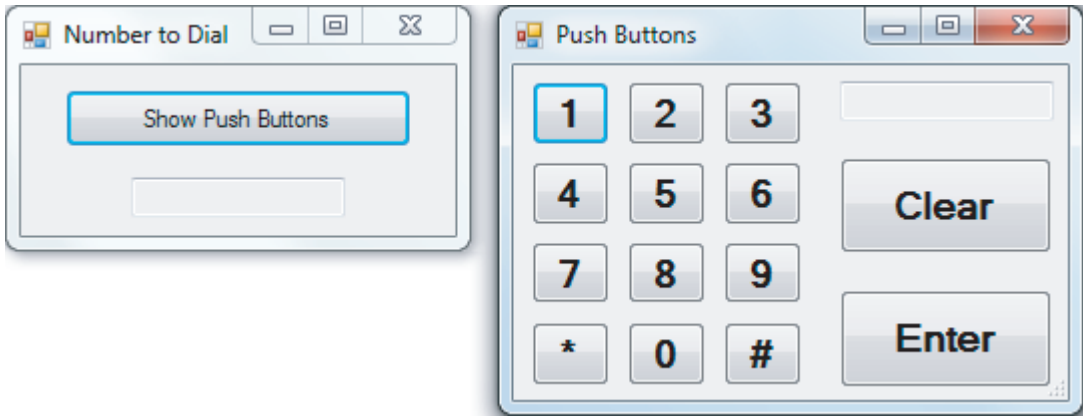


FIGURE 9.30 **Sample run of Exercise 12.**

Solutions to Practice Problems 9.3

1. In frmLogin's code, delete the two lines

```
Public userName As String
```

and

```
userName = txtUserName.Text
```

In frmOrder's code, change the line

```
txtUserName.Text = frmLogin.userName
```

to

```
txtUserName.Text = frmLogin.txtUserName.Text
```

9.4 Graphics

In this section, we draw bar charts and pie charts in a picture box, and illustrate one method for creating animation on a form.

Caution: Since the programs in this section mix text and graphics, what you see on the monitor will vary with the monitor's DPI setting. To guarantee the intended outcomes, you should check that your monitor is set to display 96 DPI (Dots Per Inch). For details, see the first item under "Configuring the Windows Environment" in Appendix B.

VideoNote
Graphics

Graphics Objects

A statement of the form

```
Dim gr As Graphics = picBox.CreateGraphics
```

declares *gr* to be a Graphics object for the picture box picBox.

The unit of measurement used in graphics methods is the **pixel**. To get a feel for how big a pixel is, the title bar of a form is 30 pixels high, and the border of a form is four pixels thick. The setting for the Size property of a picture box is two numbers separated by a comma. The two numbers give the width and height of the picture box in pixels. You can alter these numbers to specify a precise size. Each point of a picture box is identified by a pair of coordinates

```
(x, y)
```

where *x* (between 0 and picBox.Width) is its distance in pixels from the left side of the picture box, and *y* (between 0 and picBox.Height) is its distance in pixels from the top of the picture box.

Text is placed in a picture box with a statement of the form

```
gr.DrawString(string, Me.Font, Brushes.Color, x, y)
```

where *string* is either a string variable or literal, Me.Font specifies that the Form's font be used to display the text, and the upper-left corner of the first character of the text has coordinates (*x*, *y*). The color of the text is determined by *Color*. IntelliSense will provide a list of about 140 possible colors after `"Brushes."` is typed. As an example, the statements

```
Dim gr As Graphics = picBox.CreateGraphics
Dim strVar As String = "Hello"
gr.DrawString(strVar, Me.Font, Brushes.Blue, 4, 30)
gr.DrawString("World", Me.Font, Brushes.Red, 35, 50)
```

produce the output shown in Fig. 9.31.

FIGURE 9.31 **DrawString method.**

Lines, Rectangles, Circles, and Sectors

Let *gr* be a Graphics object for picBox. Then the statement

```
gr.DrawLine(Pens.Color, x1, y1, x2, y2)
```

draws a straight line segment from the point with coordinates (*x*1, *y*1) to the point with coordinates (*x*2, *y*2). The color of the line is determined by *Color*. IntelliSense will provide an extensive list of possible colors after `"Pens."` is typed. For instance, the statement

```
gr.DrawLine(Pens.Blue, 50, 20, 120, 75)
```

draws a blue line from the point with the coordinates (50, 20) to the point with the coordinates (120, 75). See Fig. 9.32.

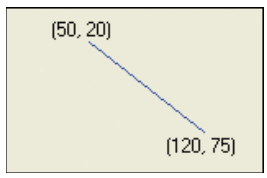


FIGURE 9.32 **DrawLine method.**

The statement

```
gr.FillRectangle(Brushes.Color, x, y, w, h)
```

draws a solid rectangle of width w and height h in the color specified and having the point with coordinates (x, y) as its upper-left vertex. The left side of the rectangle will be x pixels from the left side of the picture box and the top side of the rectangle will be y pixels from the top of the picture box. For instance, the statement

```
gr.FillRectangle(Brushes.Blue, 50, 20, 70, 55)
```

draws the rectangle shown in Fig. 9.33.

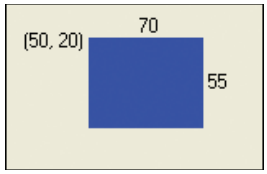


FIGURE 9.33 **FillRectangle method.**

The FillEllipse method draws a solid ellipse of a specified color, given the specifications of a circumscribed rectangle. The rectangle is specified by the coordinates of its upper-left point, its width, and its height. This method produces a circle when the width and height of the rectangle are the same. In particular, the statement

```
gr.FillEllipse(Brushes.Color, a - r, b - r, 2 * r, 2 * r)
```

draws a solid circle of the specified color with center (a, b) and radius r. For instance, the statement

```
gr.FillEllipse(Brushes.Blue, 80 - 40, 50 - 40, 2 * 40, 2 * 40)
```

draws a solid blue circle with center (80, 50) and radius 40. ***Note:*** If a rectangle were circumscribed about the circle, the rectangle would be a square with its upper-left vertex at (40, 10) and each side of length 80.

The FillPie method draws a solid sector of an ellipse in a color. The ellipse is specified by giving the coordinates, width, and height for the circumscribing rectangle, as in the FillEllipse method. The sector is determined by a radius line and the angle swept out by the radius line. We are interested solely in the case where the ellipse is a circle. The shaded region in Fig. 9.34 is a typical sector (or pie-shaped region) of a circle. The sector is determined by the two angles θ_1 and θ_2. The start angle, θ_1, is the angle through which the horizontal radius line must be rotated clockwise to reach the starting radius line of the sector. Angle θ_2 is the number of degrees through which the starting radius line must sweep (clockwise) to reach the ending radius line of the sector. The angles θ_1 and θ_2 are referred to as the **start angle** and the **sweep angle**, respectively. Figure 9.35 on the next page shows the start and sweep angles for three sectors of a circle.

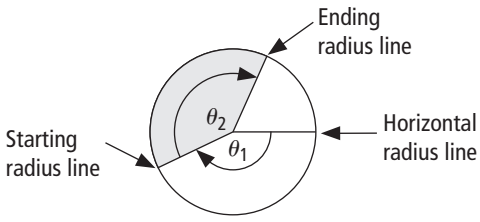


FIGURE 9.34 **A typical sector of a circle.**

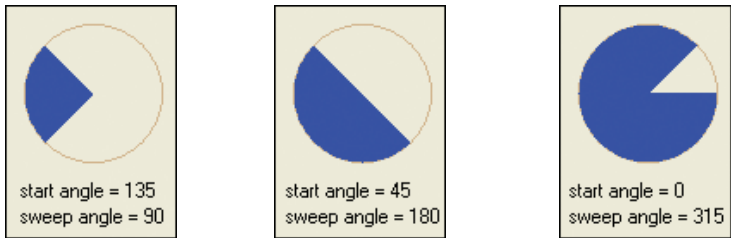


FIGURE 9.35 **FillPie method.**

In general, a statement of the form

```
gr.FillPie(Brushes.Color, a – r, b – r, 2 * r, 2 * r,
           startAngle, sweepAngle)
```

draws a sector of a circle of the specified color with center (*a*, *b*), radius *r*, and the given start and sweep angles. For instance, the middle image of Fig. 9.35 can be drawn with a statement such as

```
gr.FillPie(Brushes.Blue, 80 – 40, 80 – 40, 2 * 40, 2 * 40, 45, 180)
```

The Brushes, Pens, and Fonts appearing in the drawing statements so far are literals of objects. Variables also can be used to provide these values. For instance, the statement gr.FillRectangle(Brushes.Blue, 50, 20, 70, 55) can be replaced by the pair of statements

```
Dim br As Brush = Brushes.Blue
gr.FillRectangle(br, 50, 20, 70, 55)
```

The first statement declares *br* to be a variable of type Brush and assigns it the value Brushes.Blue.

Numeric variables used in the Draw and Fill statements discussed in this section must be of type Integer or Single. The **Single data type** is similar to the Double data type but has a smaller range. A variable of type Single can hold whole numbers, fractions, or mixed numbers between about $-3.4 \cdot 10^{38}$ and $3.4 \cdot 10^{38}$. The **CSng** function converts other data types to the Single data type.

Pie Charts

Consider the three pieces of data in Table 9.3. A pie chart can be used to graphically display the relative sizes of these numbers. The first step in creating a pie chart is to convert the numbers to percents. Since the total expenditures are \$419 billion, the federal outlay is 33/419 ≈ .08 or 8%. Similarly, the state and local expenditures are 49% and 43%. See Table 9.4. Our goal is to write a program to display the information in the pie chart of Fig. 9.36.

TABLE 9.3 **Financing for public schools (in billions).**

Federal	\$33
State	\$206
Local	\$180

TABLE 9.4 **Financing for public schools.**

Federal	.08 or 8%
State	.49 or 49%
Local	.43 or 43%

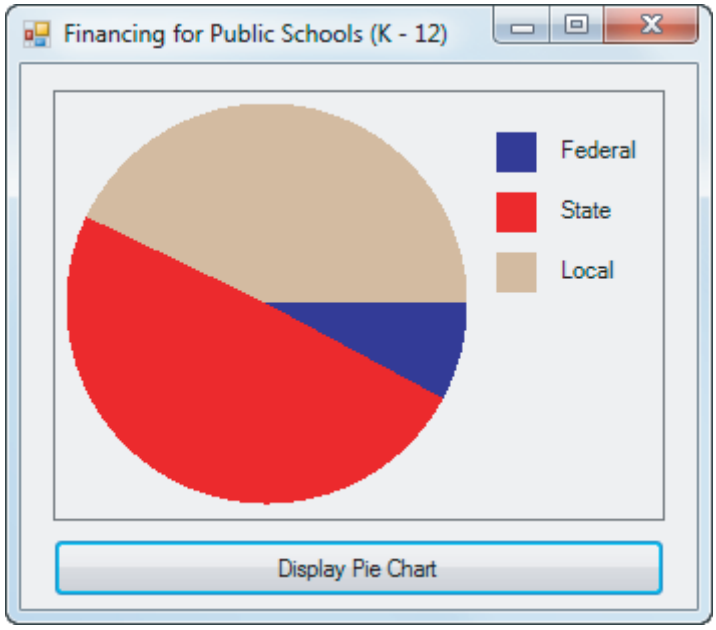


FIGURE 9.36 **Pie chart for Example 1.**

The blue sector in Fig. 9.36 has start angle 0 degrees and sweep angle .08 * 360 degrees . The red sector has start angle .08 * 360 and sweep angle .49 * 360. The tan sector has start angle .08 * 360 + .49 * 360 [or (.08 + .49) * 360 degrees] and sweep angle .43 * 360° degrees. Notice that each start angle is (sum of previous percentages) * 360. The sweep angle for each sector is the corresponding percentage times 360.

Example 1 The following program creates the pie chart (see Fig. 9.36) for the financing of public schools. The program is written so that it can be easily converted to handle a pie chart with up to six sectors. All that is required is to change the first two Dim statements and the Me.Text statement. The "Dim br() As Brush" line, which creates an array of brushes, has six brushes in order to accommodate additional sectors.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim legend() As String = {"Federal", "State", "Local"}
  Dim quantity() As Single = {33, 206, 180}
  Dim percent(quantity.Count – 1) As Single
  Dim sumOfQuantities As Single = 0
  Dim sumOfSweepAngles As Single = 0
  Dim br() As Brush = {Brushes.Blue, Brushes.Red, Brushes.Tan,
            Brushes.Green, Brushes.Orange, Brushes.Gray}
  Dim gr As Graphics = picOutput.CreateGraphics
  'The picture box has width 312 and height 215
  Dim r As Integer = 100     'Radius of circle
  Dim c As Integer = 105     'Center of circle has coordinates (c, c)
  Me.Text = "Financing for Public Schools (K – 12)"
  'Sum the numbers for the quantities
  For i As Integer = 0 To quantity.Count – 1
    sumOfQuantities += quantity(i)
  Next
```

```
    'Convert the quantities to percents
    For i As Integer = 0 To quantity.Count - 1
      percent(i) = quantity(i) / sumOfQuantities
    Next
    'Display the pie chart and the legends
    For i As Integer = 0 To quantity.Count - 1
      gr.FillPie(br(i), c - r, c - r, 2 * r, 2 * r,
                 sumOfSweepAngles, percent(i) * 360)
      sumOfSweepAngles += percent(i) * 360
      'Display small colored square and legend
      gr.FillRectangle(br(i), 220, 20 + 30 * i, 20, 20)
      gr.DrawString(legend(i), Me.Font, Brushes.Black, 250, 22 + 30 * i)
    Next
End Sub
```

Bar Charts

Our goal here is to produce the bar chart of Fig. 9.37. The picture box for the chart has a width of 210 and height of 150 pixels. (Here, the BorderStyle property is set to FixedSingle for instructional reasons. In general, the bar chart will look better with the BorderStyle property of the picture box left at its default setting: None.)

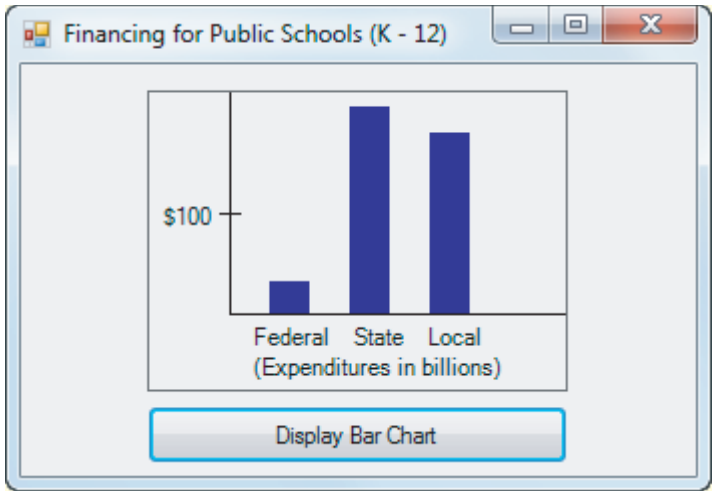


FIGURE 9.37 **Bar chart for Example 2.**

The three magnitudes for the graph are 33, 206, and 180. If we let a pixel correspond to one unit, then the largest rectangle will be 206 pixels high—a bit too large. With a pixel corresponding to 2 units, the largest rectangle will be 206/2 or 103 pixels high—a reasonable size. By setting the *x*-axis 110 pixels from the top of the picture box, the largest rectangle is accommodated comfortably. The top of the largest rectangle is 110 − 103 [that is, 110 − (206/2)] pixels from the top of the picture box. In general, a rectangle corresponding to the quantity q will be 110 − (q/2) pixels from the top of the picture box, and the height will be q/2 pixels.

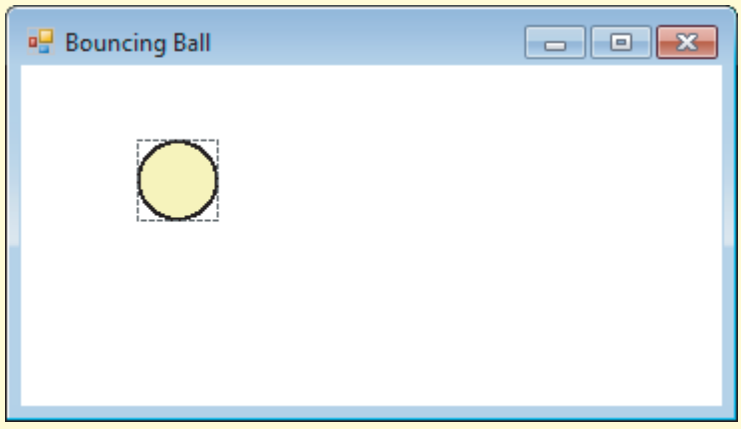
Example 2 The following program produces the bar chart of Fig. 9.37. Each rectangle is 20 pixels wide, and there are 20 pixels between rectangles.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim quantity() As Single = {33, 206, 180}
  Dim gr As Graphics = picOutput.CreateGraphics
  'The picture box has width 210 and height 150
  gr.DrawLine(Pens.Black, 40, 110, 210, 110) 'x-axis
  gr.DrawLine(Pens.Black, 40, 110, 40, 0)    'y-axis
  gr.DrawLine(Pens.Black, 35, 60, 45, 60)    'tick mark; 60 = 110 - (100/2)
  gr.DrawString("$100", Me.Font, Brushes.Black, 5, 55)
  Me.Text = "Financing for Public Schools (K – 12)"
  For i As Integer = 0 To quantity.Count – 1
    gr.FillRectangle(Brushes.Blue, 60 + i * 40,
                     (110 – quantity(i) / 2), 20, quantity(i) / 2)
  Next
  gr.DrawString("Federal  State     Local", Me.Font,
                Brushes.Black, 50, 115)
  gr.DrawString("(Expenditures in billions)", Me.Font,
                 Brushes.Black, 50, 130)
End Sub
```

Animation

One way to produce animation on a form is to place an image into a picture box and then move the picture by steadily changing the location of the picture box. Figure 9.38 shows a ball placed inside a small picture box.

Example 3 In the following program, the ball in Fig. 9.38 will initially move diagonally in a southeast direction and then bounce off any side of the form it hits. The **client area** of a form is the gray area within the title bar and borders of the form. The values of Me.ClientSize.Height and Me.ClientSize.Width are the height and width of the white area. The values of picBox.Top and picBox.Left are the distances of the picture box from the top and left sides of the client area.



OBJECT	PROPERTY	SETTING
frmBall	Text	Bouncing Ball
	BackColor	White
picBall	Image	Moon5.bmp
Timer1	Interval	10

FIGURE 9.38 The form for Example 3.

The speed at which the ball moves is determined by the setting for the Interval property of Timer1. At each tick, the ball will move x pixels horizontally, where $x = 1$ or -1. When $x = 1$ the ball moves to the right, and when $x = -1$ the ball moves to the left. The value of x reverses when the ball strikes the right or left side of the form. The value of y determines the vertical motion of the ball in a similar manner,

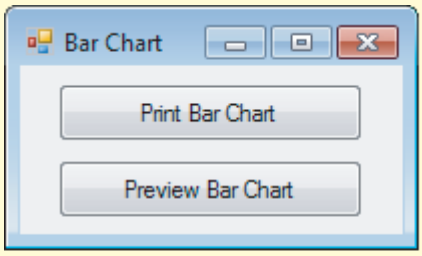
```
Dim x As Integer = 1
Dim y As Integer = 1

Private Sub frmBall_Load(...) Handles MyBase.Load
  Timer1.Enabled = True
End Sub

Private Sub Timer1_Tick(...) Handles Timer1.Tick
  If picBall.Left <= 0 Or
        picBall.Left >= (Me.ClientSize.Width - picBall.Width) Then
    x = -x
  End If
  picBall.Left += x
  If picBall.Top <= 0 Or
       picBall.Top >= (Me.ClientSize.Height - picBall.Height) Then
    y = -y
  End If
  picBall.Top += y
End Sub
```

Printing Graphics

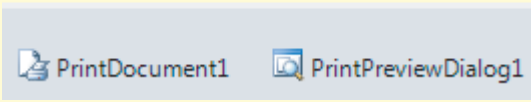
Graphics can be printed with a PrintDocument control in the same way that text was printed in Section 3.3.

Example 4 The following program produces the same output as Example 2. However, the output is printed instead of being displayed in a picture box. The changes from Example 2 are as follows:

1. The code was moved from the btnDisplay_Click event procedure to the procedure PrintDocument1_PrintPage.
2. The source of the graphics object *gr* was changed from picOutput.CreateGraphics to e.Graphics.
3. The Me.Text statement was replaced with a DrawString statement.
4. The values of the x-coordinates were increased by 300 to approximately center the graph horizontally, and the values of the y-coordinates were increased by 200 to lower the graph from the top edge of the page.



OBJECT	PROPERTY	SETTING
frmBarChart	Text	Bar Chart
btnPrint	Text	Print Bar Chart
btnPreview	Text	Preview Bar Chart
PrintDocument1		
PrintPreviewDialog1		




```
Private Sub btnPrint_Click(...) Handles btnPrint.Click
  PrintDocument1.Print()
End Sub

Private Sub PrintDocument1_PrintPage(...) Handles PrintDocument1.PrintPage
  Dim quantity() As Single = {33, 207, 180}
  Dim gr As Graphics = e.Graphics
  gr.DrawLine(Pens.Black, 340, 310, 510, 310)    'x-axis
  gr.DrawLine(Pens.Black, 340, 310, 340, 200)    'y-axis
  gr.DrawLine(Pens.Black, 335, 260, 345, 260)    'tick mark
  gr.DrawString("$100", Me.Font, Brushes.Black, 305, 255)
  gr.DrawString("Financing for Public Schools (K – 12)", Me.Font,
                Brushes.Black, 300, 175)
  For i As Integer = 0 To quantity.Count – 1
    gr.FillRectangle(Brushes.Blue, 360 + i * 40,
                     (310 – quantity(i) / 2), 20, quantity(i) / 2)
  Next
  gr.DrawString("Federal    State    Local", Me.Font,
                Brushes.Black, 350, 315)
  gr.DrawString("(Expenditures in billions)", Me.Font,
                Brushes.Black, 350, 330)
End Sub

Private Sub btnPreview_Click(...) Handles btnPreview.Click
  PrintPreviewDialog1.Document = PrintDocument1
  PrintPreviewDialog1.ShowDialog()
End Sub
```

Comments

1. A statement of the form

   ```
   Dim pn As Pen = Pens.Color
   ```

   declares *pn* to be a variable of type Pen and assigns it the value Pens.*Color*.

2. A statement of the form

   ```
   Dim fnt As Font = New Font(fontName, size)
   ```

   declares *fnt* to be a variable of type Font and assigns it the specified font and size. For instance, the statements

```
Dim gr As Graphics = picBox.CreateGraphics
Dim fnt As Font = New Font("Courier New", 10)
gr.DrawString("Hello", fnt, Brushes.Blue, 4, 30)
```

display the word Hello in 10-point Courier New font.

3. The statement

```
picBox.Refresh()
```

clears all graphics and text from the picture box.

Practice Problems 9.4

1. (True or False) The Draw and Fill methods discussed in this section use colored Brushes.
2. Write lines of code that place a smiling face in the upper left corner of the form. ***Note:*** The letter *J* corresponds to a smiling face in the Wingdings font.

EXERCISES 9.4

In Exercises 1 through 4, write a program to draw the given figures in a picture box.

1. Draw a circle whose center is located at the center of a picture box.
2. Draw a circle whose leftmost point is at the center of a picture box.
3. Use the FillEllipse method to create an unfilled red circle of radius 20.
4. Draw a triangle with two sides of the same length.

In Exercises 5 through 18 display the graphics in a picture box.

In Exercises 5 through 8, write a program to create the flag of the designated country. Refer to Fig. 9.39. *Note*: The Swiss flag is square. For the other three flags, the width is 1.5 times the height.

Italy Switzerland Niger Greenland

FIGURE 9.39 Flags of four countries.

5. Italy 6. Switzerland 7. Niger 8. Greenland

9. Write a program to draw displays such as the one in Fig. 9.40. Let the user specify the maximum number (in this display, 8).

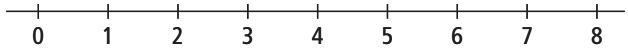


FIGURE 9.40 Drawing for Exercise 9.

10. Write a program to draw displays such as the one in Fig. 9.41. Let the user specify the number of lines (in this display, 3).

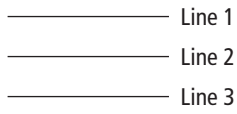


FIGURE 9.41 **Drawing for Exercise 10.**

11. Use the data in Table 9.5 to create a pie chart.

TABLE 9.5 **United States recreational beverage consumption.**

Soft Drinks	52.9%
Beer	14.7%
Bottled Water	11.1%
Other	21.3%

12. Use the data in Table 9.6 to create a bar chart.

TABLE 9.6 **United States minimum wage.**

1959	1.00
1968	1.15
1978	2.65
1988	3.35
1998	5.15
2009	7.25

13. Write a program to create the line chart in Fig. 9.42. Use the data in Table 9.7.

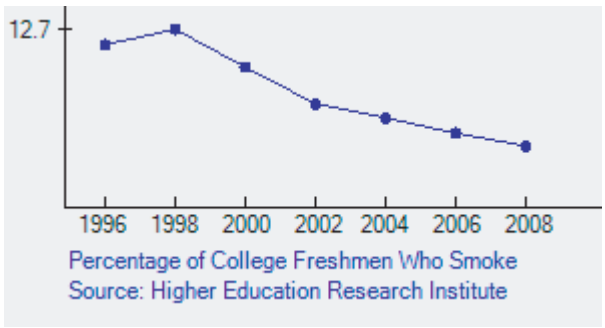


FIGURE 9.42 **Line chart for Exercise 13.**

TABLE 9.7 **Percentage of College Freshmen Who Smoke.**

	1996	1998	2000	2002	2004	2006	2008
Percent	11.6	12.7	10.0	7.4	6.4	5.3	4.4

Source: Higher Education Research Institute.

14. Write a program to create the line chart in Fig. 9.43. Use the data in Table 9.8

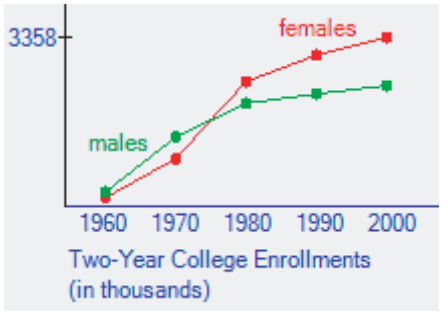


FIGURE 9.43 **Line chart for Exercise 14.**

TABLE 9.8 **Two-year college enrollments (in thousands).**

	1960	1970	1980	1990	2000
Male	283	1375	2047	2233	2398
Female	170	945	2479	3007	3358

15. Write a program to create the bar chart in Fig. 9.44.

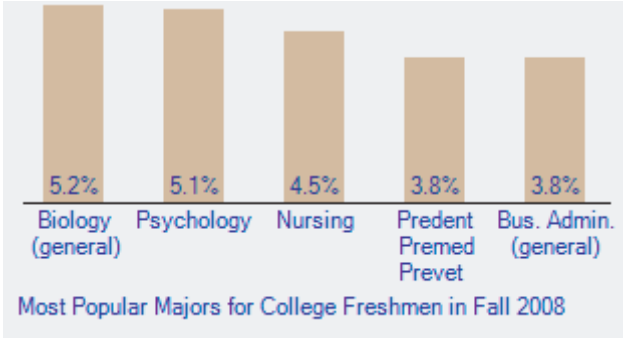


FIGURE 9.44 **Bar chart for Exercise 15.**

16. Write a program to create the bar chart in Fig. 9.45. Use the data in Table 9.9.

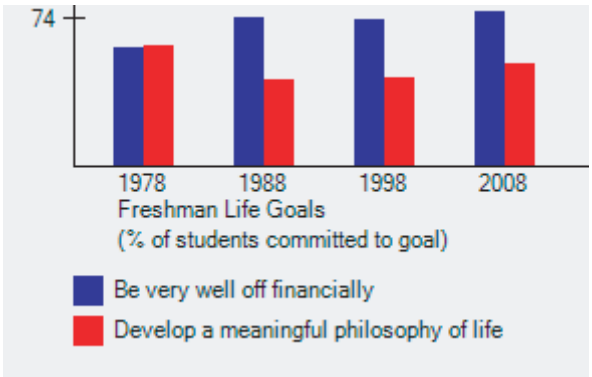


FIGURE 9.45 **Bar chart for Exercise 16.**

TABLE 9.9 Freshman Life Goals (% of students committed to goal).

	1978	1988	1998	2008
Be very well off financially	59	74	73	77
Develop a meaningful philosophy of life	60	43	44	51

Source: Higher Education Research Institute.

17. Write a program to create the bar chart in Figure 9.46. Use the data in Table 9.10. ***Note:*** Mandarin and Wu are spoken primarily in China.

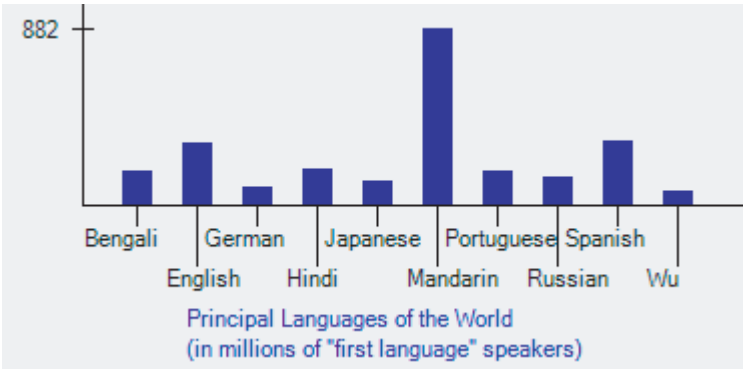


FIGURE 9.46 Bar chart for Exercise 17.

TABLE 9.10 Principal languages of the world.

Bengali	173
English	311
German	96
Hindi	182
Japanese	128
Mandarin	882
Portuguese	179
Russian	146
Spanish	326
Wu	78

18. Write a program that allows the user to display a budget as a pie chart. See Fig. 9.47 on the next page. After the user enters numbers into the four text boxes and click on the button, the pie chart should be displayed.

19. Write a program in which an airplane flies horizontally to the right across a form. After it flies off the form, the airplane should reappear on the left and fly horizontally across the screen again. ***Note:*** Use the image Airplane.bmp found in the folder Ch09\Pictures.

20. Rewrite the program in Example 1 so that the pie chart is printed instead of being displayed in a picture box. ***Note:*** You will most likely want to replace the variable *c* with the pair of variables *cx* and *cy*. Then the center of the circle will have coordinates (*cx*, *cy*).

21. Refer to Exercise 5. Write a program to print the flag of Italy.

22. Refer to Exercise 7. Write a program to print the flag of Niger.

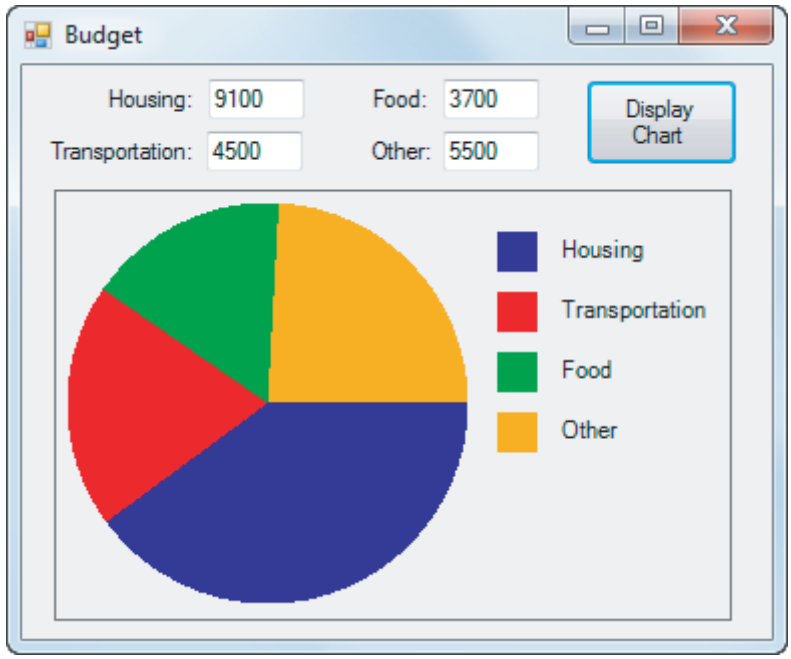


FIGURE 9.47 Form for Exercise 18.

Solutions to Practice Problems 9.4

1. False. Only the Fill methods and the DrawString method use colored Brushes. The DrawLine method uses colored Pens.
2.
```
Dim gr As Graphics = Me.CreateGraphics
Dim fnt As Font = New Font("Wingdings", 20)
gr.DrawString("J", fnt, Brushes.Red, 0, 0)
```

CHAPTER 9 SUMMARY

1. *List boxes* provide easy access to lists of data. Items() holds the items stored in the list box. Each item is identified by an index number. The lists can be automatically sorted (Sorted property = True) and altered (Items.AddItem, Items.RemoveAt, and Items.Remove methods), the currently highlighted item identified (Text property), and the number of items determined (Items.Count property).
2. Simple and DropDown style *combo boxes* are enhanced text boxes. They allow the user to fill the text box by selecting an item from a list or by typing the item directly into the text box. The contents of the text box are assigned to the combo box's Text property. A DropDownList style combo box is essentially a list box that drops down instead of being permanently displayed.
3. The *timer control* raises an event repeatedly after a specified time interval.
4. An object of type *Random* can generate a randomly selected integer from a specified range.
5. The *ToolTip control* allows the program to display guidance when the mouse hovers over a control.
6. The *Clipboard* is filled with the SetText method or by pressing Ctrl + C, and its contents are copied with the GetText method or by pressing Ctrl + V.