9.3 Multiple-Form Programs

A Visual Basic program can contain more than one form. Additional forms are added from the menu bar's *Project* menu by clicking on *Add Windows Form*, which brings up an Add New Item dialog box with "Windows Form" highlighted. To add the new form, optionally type in a name and press the *Add* button. The new form has a default name such as Form1 or Form2. The name of each form in the program appears in the Solution Explorer window. See Fig. 9.18. When you double-click on the name of a form, its Form Designer appears in the Document window. In practice, forms are given descriptive names. However, we will initially use the default names.

VideoNote
Multiple-form programs

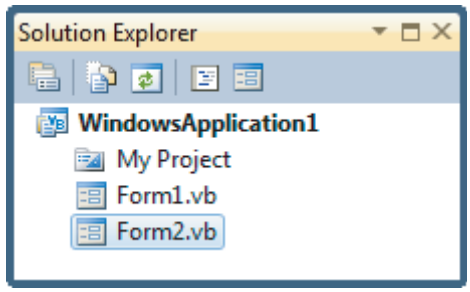


FIGURE 9.18 **Solution Explorer window after a second form is added.**

The most common use of an additional form is as a customized input dialog box (Fig. 9.19) or a customized message dialog box (Fig. 9.20). The form in Fig. 9.19 could appear to limit access to the rest of the program only to a user who enters a registered user name and password. In Fig. 9.20 the output of the Weekly Payroll case study from Chapter 5 is displayed in a second form instead of in a list box.

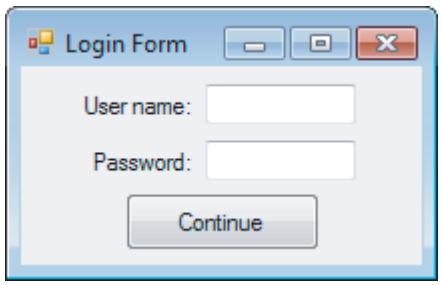


FIGURE 9.19 **Customized input dialog box.**

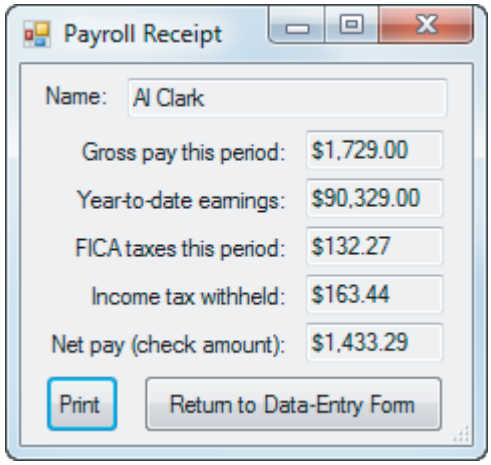


FIGURE 9.20 **Customized message dialog box.**

Startup Form

When a program starts running, only one form (called the **startup form**) will be loaded. By default, the first form created is the startup form. The following steps change the startup form.

1. Right-click on the name of the program at the top of the Solution Explorer window and click on *Properties* in the drop-down context menu. The program's Project Designer will appear.
2. Click the Application tab.

3. Select a form from the *Startup form* drop-down list.
4. Close the Project Designer by clicking the × symbol on its tab.

Scope of Variables, Constants, and Procedures

We have considered block-level, procedure-level (or local-level), and class-level (or module-level) scope. A variable or constant declared inside a block (such as a Do loop or If block) can no longer be referred to when execution passes out of the block. If you declare a variable or constant inside a procedure but outside any block within that procedure, you can think of the variable as having block-level scope, where the block is the entire procedure. You declare a class-level variable or constant for a form by placing its Dim or Const statement outside of any procedure. Class-level variables or constants can be referred to anywhere in the form's code.

If a program has more than one form, then you can extend the scope of a class-level variable to all the forms in the program by using the keyword Public in place of the keyword Dim in its declaration statement. The variable is then said to have **namespace-level scope**. Let's refer to the form in which the variable is declared as its *declaration form*. When such a variable is referred to in the code of another form, the declaration form's name (followed by a period) must precede the name of the variable. For instance, the variable *total* declared as a class- level variable in Form1 with the statement

```
Public total as Double
```

must be referred to as *Form1.total* when used in Form2.

The scope of a class-level constant is converted to namespace-level by preceding the keyword Const with the keyword Public. A general procedure and a Structure declaration have namespace-level scope by default. Preceding their header with the keyword Private will limit their access to their declaration form. Controls always have namespace-level scope. Just as with variables, the names of namespace-level constants, general procedures, and controls must be preceded by their declaration form's name (followed by a period) when referred to in another form's code.

Modality

A form can invoke another form as a modal or modeless form. A **modal** form must be closed before the user can continue working with the rest of the program. (Ordinary input and message dialog boxes are examples of modal forms.) With a **modeless** (or **nonmodal**) form, the user can shift the focus between the form and another form without having to first close the initial form. (Visual Basic's *Find* dialog box is an example of a modeless form.) In this book, new forms will always be invoked as modal forms.

Close and ShowDialog Methods

The statement `Me.Close()` closes the form whose code contains it. The Close method actually can be used to close any form in the program. The statement

```
frmOther.Close()
```

where *frmOther* is a form other than the form containing the statement, closes frmOther.

The statement

```
frmOther.ShowDialog()
```

displays the other form as a modal form and gives it the focus. (The statement `frmOther.Show()` displays the other form as a modeless form.)

The FormClosing Event Procedure

The Load event procedure occurs before a form is displayed for the first time or before it is displayed after having been closed. Analogous to the Load event is the FormClosing event that occurs before the form is closed. (A form is closed by the execution of a Close method, by the user's clicking on the form's Close button in the title bar, or by the user's pressing Alt + F4.)

Importing an Existing Form

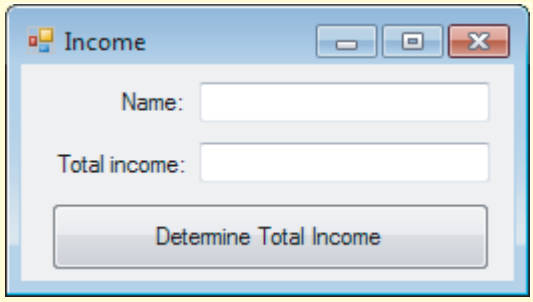
You can add a form created in another program to the current program with the following steps:

1. Click on *Add Existing Item* in the menu bar's *Project* menu.
2. Navigate to the program containing the form. The program will contain a file named *formName*.vb.
3. Double-click on *formName*.vb. That file will be copied into your program's Solution Explorer and you will have added its form to your program. ***Note:*** If the added form refers to a text file, the text file will have to be copied separately into your program's *bin\Debug* folder.

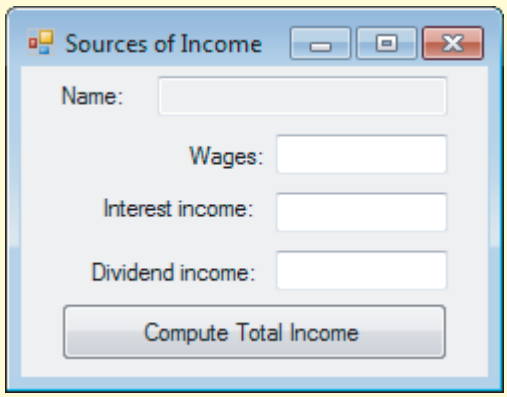
Deleting a Form from a Program

To remove a form from a program, right-click on its name in the Solution Explorer, and click on *Delete* in the drop-down context menu. (An input dialog box will ask you to confirm the deletion.) If the deleted form was the startup form, you will have to select a new startup form.

Example 1 The following program uses a second form as a dialog box to obtain and total the different sources of income. Initially, only frmIncome is visible. The user types in his or her name and then can either type in the total income or click on the button for assistance in totaling the different sources of income. Clicking on the button from frmIncome causes frmSources to appear and be active. The user fills in the three text boxes and then clicks on the button to have the amounts totaled and displayed in the "Total income" text box of frmIncome.



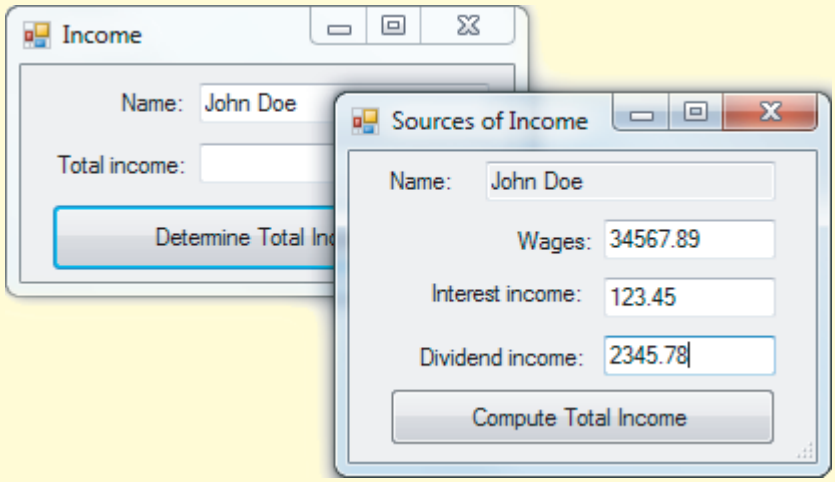
OBJECT	PROPERTY	SETTING
frmIncome	Text	Income
lblName	Text	Name:
txtName		
lblTotIncome	Text	Total income:
txtTotIncome		
btnDetermine	Text	Determine Total Income



OBJECT	PROPERTY	SETTING
frmSources	Text	Sources of Income
lblName	Text	Name:
txtName	ReadOnly	True
lblWages	Text	Wages:
txtWages		
lblIntIncome	Text	Interest income:
txtIntIncome		
lblDivIncome	Text	Dividend income:
txtDivIncome		
btnCompute	Text	Compute Total Income

```
'frmIncome's code   (startup form)

Private Sub btnDetermine_Click(...) Handles btnDetermine.Click
  frmSources.txtName.Text = txtName.Text
  frmSources.ShowDialog()     'Show the second form and wait until it closes.
  '                       Then execute the rest of the code in this procedure.
  txtTotIncome.Text = FormatCurrency(frmSources.sum)
End Sub

'frmSources's code

Public sum As Double    'holds the sum of the text boxes' values

Private Sub frmSources_Load(...) Handles MyBase.Load
  txtWages.Clear()
  txtIntIncome.Clear()
  txtDivIncome.Clear()
End Sub

Private Sub btnCompute_Click(...) Handles btnCompute.Click
  'Store the total into the namespace-level variable sum.
  sum = CDbl(txtWages.Text) + CDbl(txtIntIncome.Text) +
        CDbl(txtDivIncome.Text)
  Me.Close()   'Close the form since it is not needed any more
End Sub
```

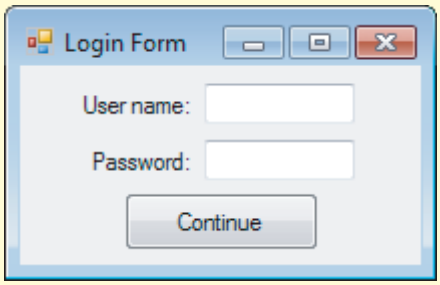
[Run, enter a name, click on the button, and fill in the sources of income.] ***Note:*** After the *Compute Total Income* button is pressed, frmSources will disappear and the sum of the three numbers will be displayed in the Total Income text box of frmIncome.



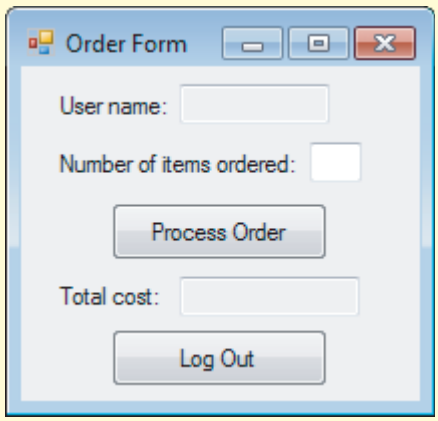
Example 2 The following program uses two forms. The startup form, frmOrder, processes an order after first requesting a user name and password with frmLogin. Even though frmOrder is the startup form, frmLogin is actually the first form seen by the user. It is invoked by frmOrder's Load event procedure.

The form frmLogin uses the text file MasterFile.txt to check for a registered user name and password. Each line of the text file consists of a user name concatenated with an underscore character and a password. The first three lines of the file contain the data dcook_idol08,

JQPublic_vbguy21, and shawnj_dance09. After checking that the text boxes have been filled in, the program uses a query to determine if the user name and password combination is in the text file. The user gets three chances to enter an acceptable response. Code in a FormClosing event procedure prevents the user from closing the login form without first giving a satisfactory user name and password.



OBJECT	PROPERTY	SETTING
frmLogin	Text	Login Form
lblUserName	Text	User name:
txtUserName		
lblPassword	Text	Password:
txtPassword		
btnContinue	Text	Continue

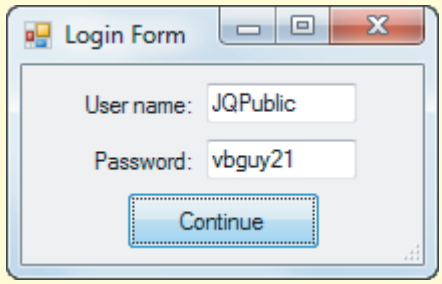


OBJECT	PROPERTY	SETTING
frmOrder	Text	Order Form
lblUserName	Text	User name:
txtUserName	ReadOnly	True
lblNumItems	Text	Number of items ordered:
txtNumItems		
btnProcess	Text	Process Order
lblTotalCost	Text	Total cost:
txtTotalCost	ReadOnly	True
btnLogOut	Text	Log Out

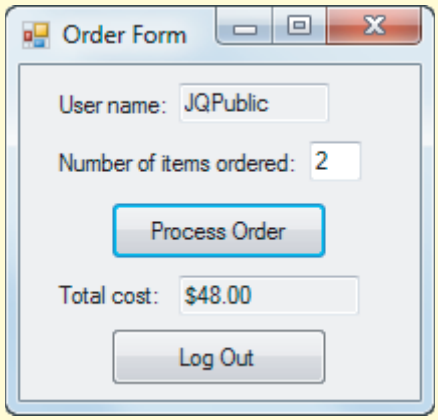
```
'frmOrder's code   (startup form)

Private Sub frmOrder_Load(...) Handles MyBase.Load
  frmLogin.ShowDialog()
  txtUserName.Text = frmLogin.userName
End Sub

Private Sub btnProcess_Click(...) Handles btnProcess.Click
  Dim numItems As Integer
  Dim totalCost As Double
  numItems = CInt(txtNumItems.Text)
  'cost per item: $20; shipping cost: $8
  totalCost = (numItems * 20) + 8
  txtTotalCost.Text = FormatCurrency(totalCost)
End Sub

Private Sub btnLogOut_Click(...) Handles btnLogOut.Click
  Me.Close()
End Sub

'frmLogin's code

Public userName As String
Dim numTries As Integer = 0
Dim idVerified As Boolean = False
```

```
Private Sub btnContinue_Click(...) Handles btnContinue.Click
  If (txtUserName.Text = "") Or (txtPassword.Text = "") Then
    MessageBox.Show("You must enter both a user name and a password.")
  Else
    If Confirm(txtUserName.Text, txtPassword.Text) Then
      idVerified = True
      userName = txtUserName.Text
      Me.Close()
    Else
      MessageBox.Show("Improper user name or password.")
      txtUserName.Clear()
      txtPassword.Clear()
    End If
  End If
  numTries += 1
  If (numTries = 3) And (Not idVerified) Then
    MessageBox.Show("This program is being terminated.")
    frmOrder.Close()
    Me.Close()
  End If
End Sub

Function Confirm(ByVal userName As String,
                 ByVal password As String) As Boolean
  Dim query = From line In IO.File.ReadAllLines("MasterFile.txt")
              Where line = userName & "_" & password
              Select line
  If query.Count = 1 Then
    Return True
  Else
    Return False
  End If
End Function

Private Sub frmLogin_FormClosing(...) Handles Me.FormClosing
  If Not idVerified Then
    MessageBox.Show("This program is being terminated.")
    frmOrder.Close()
  End If
End Sub
```

[Run, enter a user name and password into the form.]



[Click on the button in the Login form. Then enter a quantity into the text box of the Order form below and click on the *Process Order* button.]



Example 3 We can easily modify the Weekly Payroll case study from Chapter 5 so that instead of the output being displayed in a list box, it is displayed in the form shown in Fig. 9.20. The steps are as follows:

1. Start a new program with the name 9-3-3.
2. Delete Form1.vb from the Solution Explorer.
3. Add the existing form frmPayroll from the program 5-5 (Weekly Payroll) to the new program.
4. Change the startup form to frmPayroll.
5. Add a new form to the program and name it frmReceipt.
6. Design the form for frmReceipt as shown in Fig. 9.20 on page 425 with the settings in Fig. 9.21.

OBJECT	PROPERTY	SETTING
frmReceipt	Text	Payroll Receipt
lblGrossPay	Text	Gross pay this period:
txtGrossPay		
lblTotalPay	Text	Year-to-date-earnings:
txtTotalPay		
lblFicaTax	Text	FICA tax this period:
txtFicaTax		
lblFedTax	Text	Income tax withheld:
txtFedTax		
lblCheck	Text	Net pay (check amount):
txtCheck		
btnPrint	Text	Print
btnReturn	Text	Return to Data-Entry Form

FIGURE 9.21 Controls and settings for frmReceipt.

7. Double-click on the PrintForm control in the *Visual Basic PowerPacks* group of the Toolbox. The control will appear with the default name PrintForm1 in the component tray.

8. Add the code shown in Fig. 9.22 to frmReceipt. ***Note:*** The code inside the btnPrint event procedure prints the contents of the form on the printer.

```
Sub SetPayrollInfo(ByVal empName As String, ByVal pay As Double,
                   ByVal totalPay As Double, ByVal ficaTax As Double,
                   ByVal fedTax As Double, ByVal check As Double)
  txtName.Text = empName
  txtGrossPay.Text = FormatCurrency(pay)
  txtTotalPay.Text = FormatCurrency(totalPay)
  txtFicaTax.Text = FormatCurrency(ficaTax)
  txtFedTax.Text = FormatCurrency(fedTax)
  txtCheck.Text = FormatCurrency(check)
End Sub

Private Sub btnPrint_Click(...) Handles btnPrint.Click
  PrintForm1.PrintAction = Printing.PrintAction.PrintToPrinter
  PrintForm1.Print()
End Sub

Private Sub btnReturn_Click(...) Handles btnReturn.Click
  Me.Close()
End Sub
```

FIGURE 9.22 Code for frmReceipt.

9. In the btnDisplay_Click procedure of frmPayroll, replace the line

```
ShowPayroll(empName, pay, totalPay, ficaTax, fedTax, check) 'Task 6
```

with

```
frmReceipt.SetPayrollInfo(empName, pay, totalPay, ficaTax,
                          fedTax, check)          'Task 6
frmReceipt.ShowDialog()
```

Practice Problems 9.3

1. Rewrite the program in Example 2 without using the namespace-level variable *userName*.

EXERCISES 9.3

In Exercises 1 through 4, determine the output displayed when the button is clicked.

1.
```
'Form1's code    (startup form)
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Form2.ShowDialog()
  txtOutput.Text = FormatCurrency(Form2.totalCost)
End Sub

Function GetTotalCost(ByVal price As Double) As Double
  Return price + (Form2.SALES_TAX_RATE * price)
End Function
```

```
'Form2's code
Public Const SALES_TAX_RATE As Double = 0.06
Public totalCost As Double

Private Sub Form2_Load(...) Handles Me.Load
  Dim price = InputBox("What is the price?")
  totalCost = Form1.GetTotalCost(CDbl(price))
  Me.Close()
End Sub
```

(Assume that the response is 100.)

2.
```
'Form1's code    (startup form)
Private Sub Form1_Load(...) Handles MyBase.Load
  Form2.ShowDialog()
  Dim name As String = Form2.txtName.Text
  Dim dob As Date = CDate(Form2.txtDateOfBirth.Text)
  Dim parsedName() As String = name.Split(" "c)
  Dim firstName = parsedName.First
  Dim message As String
  If dob.AddYears(21) <= Today Then
    message = ", you are at least 21 years old."
  Else
    message = ", you are not yet 21 years old."
  End If
  txtOutput.Text = firstName & message
End Sub

'Form2's code
Private Sub Form2_Load(...) Handles MyBase.Load
  txtName.Text = "John Doe"
  txtDateOfBirth.Text = "2/3/1989"
End Sub

Private Sub btnRecord_Click(...) Handles btnRecord.Click
  Me.Close()
End Sub
```

3.
```
'Form1's code   (startup form)
Private Sub Form1_Load(...) Handles MyBase.Load
  Form2.ShowDialog()
  Dim name As String = Form2.fullName
  Dim lastName As String = Form2.GetLastName(name)
  txtOutput.Text = "Your last name begins with " &
                   lastName.Substring(0, 1) & "."
End Sub

'Form2's code
Public fullName As String

Private Sub btnDetermine_Click(...) Handles btnDetermine.Click
   fullName = "John Fitzgerald Kennedy"
```

```
    Me.Close()
End Sub

Function GetLastName(ByVal nom As String) As String
  Dim parsedName() As String = nom.Split(" "c)
  Return parsedName.Last
End Function
```

4.
```
'Form1's code    (startup form)
Public average As Double

Private Sub Form1_Load(...) Handles MyBase.Load
  Form2.ShowDialog()
End Sub

Private Sub btnComputeAverage_Click(...) Handles btnComputeAverage.Click
  Dim num As Double = 0
  Dim count As Integer = 0
  Dim sum As Double = 0
  num = CDbl(InputBox("Enter a number"))
  Do While num <> -1
    count += 1
    sum += num
    num = CDbl(InputBox("Enter a number"))
  Loop
  average = sum / count
  Form3.ShowDialog()
  Me.Close()
End Sub

'Form2's code
Private Sub Form2_Load(...) Handles MyBase.Load
  Dim message As String = "The purpose of this program is to" &
      " calculate the average of a set of nonnegative numbers" &
      " input by the user. Enter the numbers one at a time" &
      " and enter -1 to signal the end of data entry."
  MessageBox.Show(message, "Instructions")
  Me.Close()
End Sub

'Form3's code
Private Sub Form3_Load(...) Handles MyBase.Load
  txtAverage.Text = "The average is " & Form1.average & "."
End Sub
```

(Assume the responses are 80, 100, and −1.)

5. Consider Example 2 of Section 6.1. Alter the program so that a second form appears (instead of a message box) when the button is pressed. The second form should allow the user to make a selection by clicking on one of three radio buttons with the captions *Movie 1*, *Movie 2*, and *Movie 3*.

6. Consider the program in Example 9 from Section 7.1 that determines a person's first and last names. Alter the program so that the person's full name is typed into the startup form and a second form is used to display their first and last names.
7. Consider Example 4 of Section 6.1. Alter the program so that a second form showing the balance after each year appears when the button is clicked on. See Fig. 9.23.

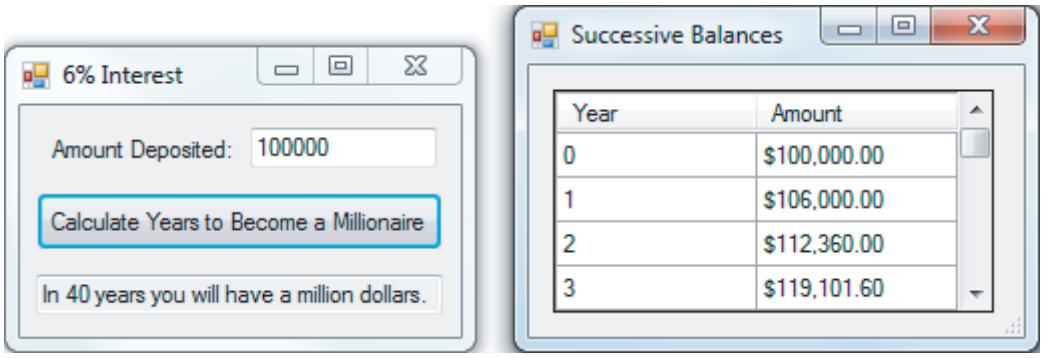


FIGURE 9.23 **Possible outcome of Exercise 7.**

8. Consider the Analyze-a-Loan case study from Chapter 7. Alter the program so that the amortization table and the interest-rate-change table are each displayed in a separate form when requested. The program should have three forms, and the startup form should not contain a DataGridView control.
9. Consider the Recording Checks and Deposits case study from Chapter 8. Alter the program so that the list of transactions is displayed in a second form when requested.
10. Write a program that allows student grades on three exams to be entered one student at a time in frmStudent and then displays each student's average and the class average in frmGrades. Initially, frmGrades (the startup form) should look like Fig. 9.24 with the text box and DataGridView controls empty. The form frmStudent should initially look like Fig. 9.25 with the four text boxes empty. The Load event procedure of frmGrades should invoke frmStudent.

    Each time a student's name and grades are recorded, the number in the title bar of frmStudent should increase by 1. The *Terminate* button should be clicked on after all students have been recorded. The data for the students should be stored in an array of structures with the structure having four members.

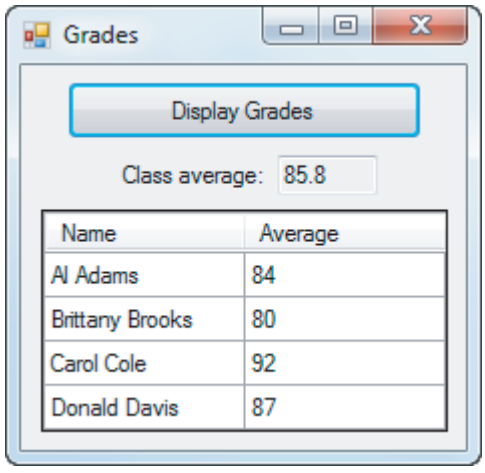


FIGURE 9.24 **frmGrades**

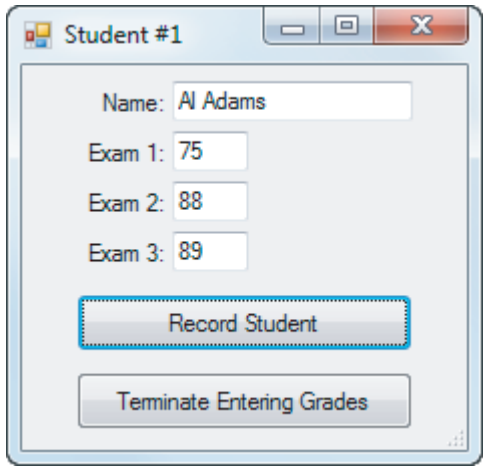


FIGURE 9.25 **frmStudent**

11. Write a program consisting of three forms that gathers customer billing information. The first form (the startup form) should initially look like Fig. 9.26, but with the text box and list box blank and no radio button selected. (The second and third forms should initially look like Figs. 9.27 and 9.29 with all text boxes blank.) After the user provides a name, selects a billing method, and clicks on the button in frmCustomer, either frmCustInfo or frmCardInfo should appear to obtain the necessary information.

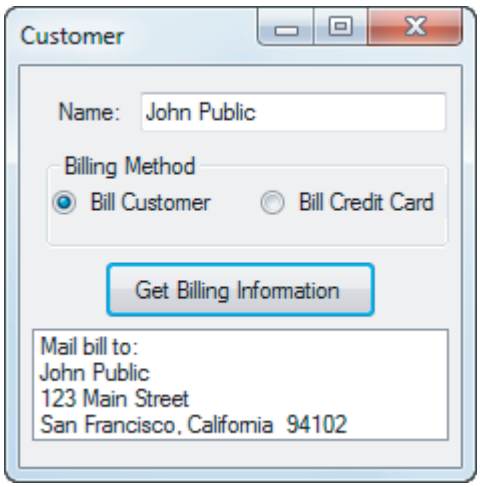


FIGURE 9.26 frmCustomer

FIGURE 9.27 frmCustInfo

Let's first consider frmCustInfo, that appears when the *Bill Customer* radio button is selected. The Name read-only text box should be filled automatically with the name that was entered in frmCustomer. The user enters information into the other text boxes, selects a state from the sorted drop-down-list combo box, and clicks on the button to display the mailing address in the list box of frmCustomer as shown in Fig. 9.26. ***Note:*** The names of the states can be obtained from the file States.txt.

The form frmCardInfo, which appears when the *Bill Credit Card* radio button in frmCustomer has been selected, contains two text boxes, one simple combo box (for type of credit card) and two DropDownList style combo boxes. The Name text box is initially automatically filled with the name that was entered in frmCustomer. However, the name can be

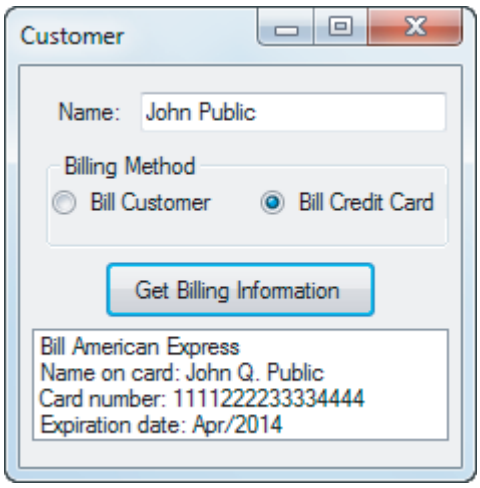


FIGURE 9.28 frmCustomer

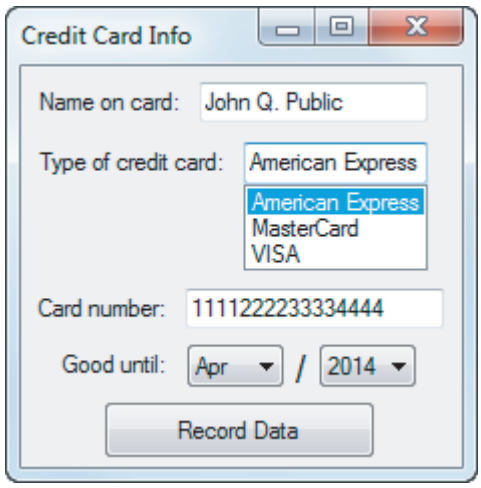


FIGURE 9.29 frmCardInfo

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.

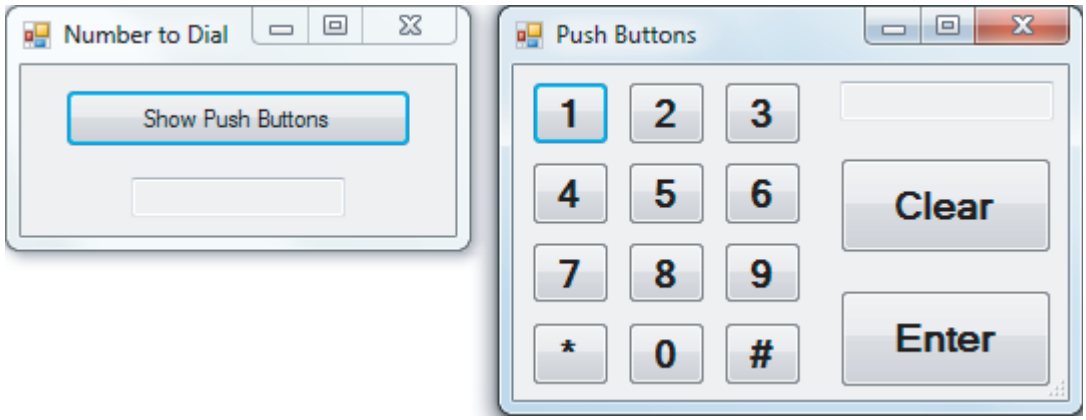


FIGURE 9.30 **Sample run of Exercise 12.**

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.