26. Write a program that calculates the average of up to 50 numbers input by the user and stored in an array. See Fig. 11.2. The program should use a class named Statistics and have an AddNumber method that stores numbers into an array one at a time. The class should have a Count property that keeps track of the number of numbers stored and a method called Average that returns the average of the numbers.

27. Write a program that calculates an employee's FICA tax, with all computations performed by an instance of a class FICA. The FICA tax has two components: the social security benefits tax, which in 2009 is 6.2% of the first $106,800 of earnings for the year, and the Medicare tax, which is 1.45% of earnings.

28. Write a program that adds two fractions and displays their sum in reduced form. The program should use a Fraction class that stores the numerator and denominator of a fraction and has a Reduce method that divides each of the numerator and denominator by their greatest common divisor. Exercise 29 of Section 6.1 contains an algorithm for calculating the greatest common divisor of two numbers.

Solutions to Practice Problems 11.1

1. (d) A programmer is not a template for creating a program.
2.
```
Public Sub New(ByVal ssn As String)
  'Assign the value of ssn to the member variable m_ssn.
  m_ssn = ssn
End Sub

Public ReadOnly Property SocSecNum() As String
  Get
    Return m_ssn
  End Get
End Property
```

Note: Since a student's social security number never changes, there is no need to have a Set property procedure for SocSecNum.

11.2 Working with Objects

"An object without an event is like a telephone without a ringer."

Anonymous

Arrays of Objects

The elements of an array can have any data type—including a class. The program in Example 1 uses an array of type Student.

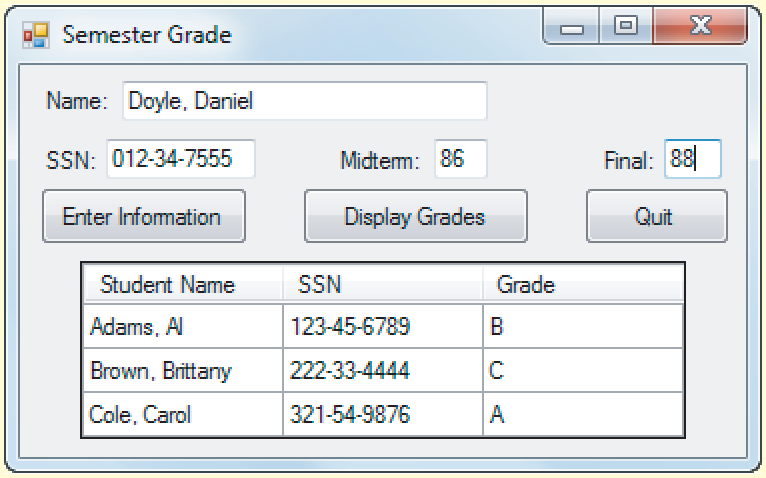
VideoNote Arrays of objects

Example 1 In the following program, which uses the same form design as Example 1 of the previous section, the user enters four pieces of data about a student into text boxes. When the *Enter Information* button is clicked on, the data are used to create and initialize an appropriate object and the object is added to an array. When the *Display Grades* button is clicked on, the name, social security number, and semester grade for each student in the array are displayed in the grid.

```
Public Class frmGrades
  Dim students(50) As Student
  Dim lastStudentAdded As Integer = -1   'Position in array of student
  '                                       most recently added
  Private Sub btnEnter_Click(...) Handles btnEnter.Click
    lastStudentAdded += 1
    students(lastStudentAdded) = New Student
    students(lastStudentAdded).Name = txtName.Text
    students(lastStudentAdded).SocSecNum = mtbSSN.Text
    students(lastStudentAdded).Midterm = CDbl(txtMidterm.Text)
    students(lastStudentAdded).Final = CDbl(txtFinal.Text)
    'Clear text boxes
    txtName.Clear()
    mtbSSN.Clear()
    txtMidterm.Clear()
    txtFinal.Clear()
    txtName.Focus()
    MessageBox.Show("Student Recorded.")
  End Sub

  Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    ReDim Preserve students(lastStudentAdded)
    Dim query = From pupil In students
                Select pupil.Name, pupil.SocSecNum, pupil.CalcSemGrade
    dgvGrades.DataSource = query.ToList
    dgvGrades.CurrentCell = Nothing
    dgvGrades.Columns("Name").HeaderText = "Student Name"
    dgvGrades.Columns("SocSecNum").HeaderText = "SSN"
    dgvGrades.Columns("CalcSemGrade").HeaderText = "Grade"
    ReDim Preserve students(50)
    txtName.Focus()
  End Sub

  Private Sub btnQuit_Click(...) Handles btnQuit.Click
    Me.Close()
  End Sub
End Class     'frmGrades

Class Student
  Private m_midterm As Double
  Private m_final As Double

  Public Property Name() As String

  Public Property SocSecNum() As String

  Public WriteOnly Property Midterm() As Double
    Set(ByVal value As Double)
      m_midterm = value
    End Set
  End Property

  Public WriteOnly Property Final() As Double
    Set(ByVal value As Double)
      m_final = value
    End Set
  End Property
```

```
    Function CalcSemGrade() As String
      Dim grade As Double
      grade = (m_midterm + m_final) / 2
      grade = Math.Round(grade)   'Round the grade.
      Select Case grade
        Case Is >= 90
          Return "A"
        Case Is >= 80
          Return "B"
        Case Is >= 70
          Return "C"
        Case Is >= 60
          Return "D"
        Case Else
          Return "F"
      End Select
    End Function
  End Class       'Student
```

[Run, type in data for Al Adams, click on the *Enter Information* button, repeat the process for Brittany Brown and Carol Cole, click on the *Display Grades* button, and then enter data for Daniel Doyle.]



Name: Doyle, Daniel

SSN: 012-34-7555 Midterm: 86 Final: 88

Student Name	SSN	Grade
Adams, Al	123-45-6789	B
Brown, Brittany	222-33-4444	C
Cole, Carol	321-54-9876	A

Events

In the previous section, we drew a parallel between classes and controls and showed how to define properties and methods for classes. Events can be defined by the programmer to communicate changes of properties, errors, and the progress of lengthy operations. Such events are called **user-defined events**. The statement for raising an event is located in the class block, and the event is dealt with in the form's code. Suppose that the event is named UserDefinedEvent and has the parameters *par1*, *par2*, and so on. In the class block, the statement

```
Public Event UserDefinedEvent(ByVal par1 As DataType1,
                              ByVal par2 As DataType2, ...)
```

should be placed in the Declarations section, and the statement

```
RaiseEvent UserDefinedEvent(arg1, arg2, ...)
```

should be placed at the locations in the class block code at which the event should be raised. In the form's code, an instance of the class, call it *object1*, must be declared with a statement of the type

```
Dim WithEvents object1 As ClassName
```

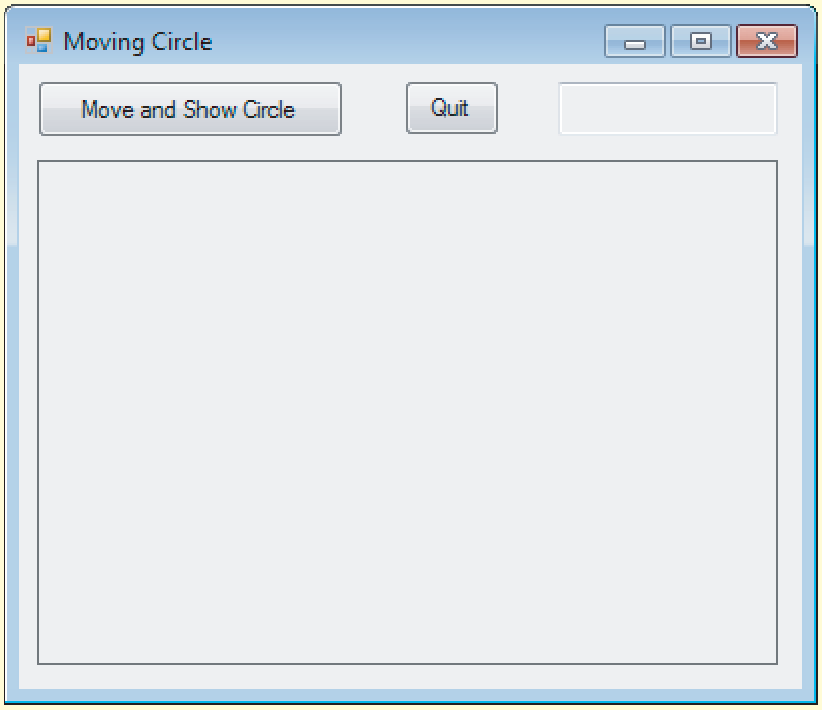
or the type

```
Dim WithEvents object1 As New ClassName
```

in order to be able to respond to the event. That is, the keyword WithEvents must be inserted into the declaration statement. The header of an event procedure for *object1* will be

```
Private Sub object1_UserDefinedEvent(ByVal par1 As DataType1,
                                     ByVal par2 As DataType2,...) _
                                     Handles object1.UserDefinedEvent
```

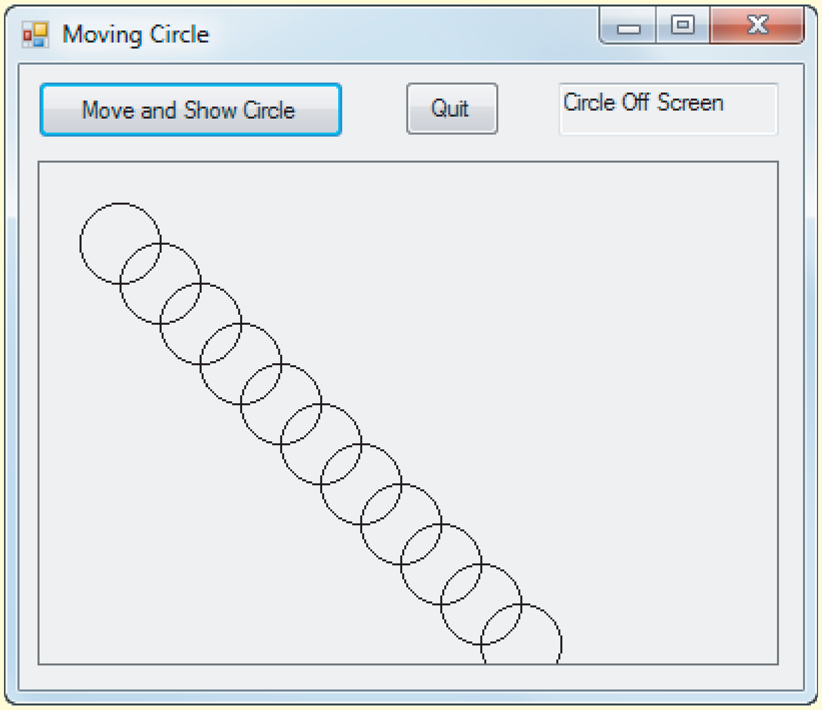
Example 2 Consider the Circle class defined in Example 3 of Section 11.1. In the following program, we add an event that is raised whenever the location of a circle changes. The event has parameters to pass the location and diameter of the circle. The form's code uses the event to determine if part (or all) of the drawn circle will fall outside the picture box. If so, the event procedure displays the message "Circle Off Screen" in a text box. Let's call the event PositionChanged.



OBJECT	PROPERTY	SETTING
frmCircle	Text	Moving Circle
btnMove	Text	Move and Show Circle
btnQuit	Text	Quit
txtCaution	ReadOnly	True
picCircle		

```
Public Class frmCircle
  Dim WithEvents round As New Circle()

  Private Sub btnMove_Click(...) Handles btnMove.Click
    round.Move(20)
    round.Show(picCircle.CreateGraphics)
  End Sub

  Private Sub btnQuit_Click(...) Handles btnQuit.Click
    Me.Close()
  End Sub

  Private Sub round_PositionChanged(ByVal x As Integer,
    ByVal y As Integer, ByVal d As Integer) Handles round.PositionChanged
    'This event is raised when the location of the circle changes.
    'The code determines if part of the circle is off the screen.
    If (x + d > picCircle.Width) Or
       (y + d > picCircle.Height) Then
      txtCaution.Text = "Circle Off Screen"
    End If
  End Sub
End Class      'frmCircle

Class Circle
  Public Event PositionChanged(ByVal x As Integer,
                               ByVal y As Integer, ByVal d As Integer)
  'Event is raised when the circle moves.

  Public Sub New()
    'Set the initial location of the circle to the upper-left
    'corner of the picture box, and set its diameter to 40.
    Xcoord = 0
    Ycoord = 0
    Diameter = 40
  End Sub

  Public Property Xcoord() As Integer

  Public Property Ycoord() As Integer

  Public Property Diameter() As Integer

  Sub Show(ByVal gr As Graphics)
    'Draw a circle with the given graphics context.
    gr.DrawEllipse(Pens.Black, Xcoord, Ycoord, Diameter, Diameter)
  End Sub

  Sub Move(ByVal distance As Integer)
    Xcoord += distance
    Ycoord += distance
    RaiseEvent PositionChanged(Xcoord, Ycoord, Diameter)
  End Sub
End Class      'Circle
```

[Run, and click on the *Move and Show Circle* button eleven times.]



Note: As the last circle appears, the words "Circle Off Screen" are displayed in the text box.

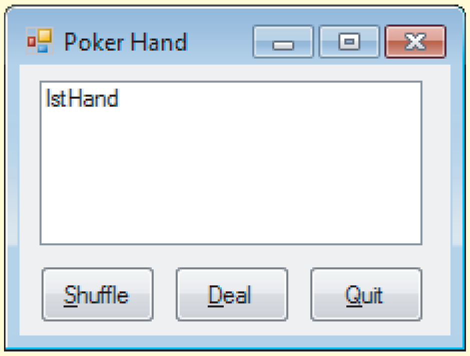
Containment

We say that class A **contains** class B when a member variable of class A makes use of an object of type class B. In Example 3, the class DeckOfCards contains the class Card.

Example 3 The following program deals a five-card poker hand. The program has a DeckOfCards object containing an array of 52 Card objects. The Card object has two properties, Denomination and Suit, and one method, IdentifyCard. The IdentifyCard method returns a string such as "Ace of Spades". In the DeckOfCards object, the New event procedure assigns denominations and suits to the 52 cards. The method ReadCard(n) returns the string identifying the *n*th card of the deck. The method ShuffleDeck uses the Random class to mix up the cards while making 2000 passes through the deck. The event

```
Shuffling(n As Integer, nMax As Integer)
```

is raised during each shuffling pass through the deck, and its parameters communicate the number of the pass and the total number of passes, so that the program that uses it can keep track of the progress.

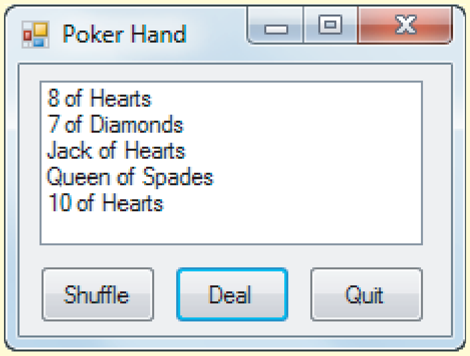


OBJECT	PROPERTY	SETTING
frmPoker	Text	Poker Hand
lstHand		
btnShuffle	Text	&Shuffle
btnDeal	Text	&Deal
btnQuit	Text	&Quit

```
Public Class frmPoker
  Dim WithEvents cards As New DeckOfCards()

  Private Sub btnShuffle_Click(...) Handles btnShuffle.Click
    cards.ShuffleDeck
  End Sub

  Private Sub btnDeal_Click(...) Handles btnDeal.Click
    Dim str As String
    lstHand.Items.Clear()
    For i As Integer = 0 To 4
      str = cards.ReadCard(i)
      lstHand.Items.Add(str)
    Next
  End Sub

  Private Sub btnQuit_Click(...) Handles btnQuit.Click
    Me.Close()
  End Sub

  Private Sub cards_Shuffling(ByVal n As Integer,
                      ByVal nMax As Integer) Handles cards.Shuffling
    'n is the number of the specific pass through the deck (1, 2, 3...).
    'nMax is the total number of passes when the deck is shuffled.
    lstHand.Items.Clear()
    lstHand.Items.Add("Shuffling Pass: " & n & " out of " & nMax)
    For i As Integer = 1 To 1000000     'Slow down the shuffle.
    Next
    lstHand.Update()  'Refresh contents of list box
  End Sub
End Class      'frmPoker

Class Card
  Private m_denomination As Integer  'A number from 0 through 12
  Private m_suit As String           'Hearts, Clubs, Diamonds, Spades

  Public Property Denomination() As Integer
    Get
      Return m_denomination
    End Get
    Set(ByVal value As Integer)
      'Only store valid values.
      If (value >= 0) And (value <= 12) Then
        m_denomination = value
      End If
    End Set
  End Property

  Public Property Suit() As String
    Get
      Return m_suit
    End Get
    Set(ByVal value As String)
      'Only store valid values.
      If (value = "Hearts") Or (value = "Clubs") Or
         (value = "Diamonds") Or (value = "Spades") Then
```

```
          m_suit = value
        End If
      End Set
    End Property

    Function IdentifyCard() As String
      Dim denom As String = ""
      Select Case Denomination + 1
        Case 1
          denom = "Ace"
        Case Is <= 10
          denom = CStr(Denomination + 1)
        Case 11
          denom = "Jack"
        Case 12
          denom = "Queen"
        Case 13
          denom = "King"
      End Select
      Return denom & " of " & m_suit
    End Function
  End Class      'Card

  Class DeckOfCards
    Private m_deck(51) As Card 'Class DeckOfCards contains class Card
    Public Event Shuffling(ByVal n As Integer, ByVal nMax As Integer)

    Public Sub New()
      'Make the first thirteen cards hearts, the
      'next thirteen cards diamonds, and so on.
      Dim suits() As String = {"Hearts", "Clubs", "Diamonds", "Spades"}
      For i As Integer = 0 To 3
        'Each pass corresponds to one of the four suits.
        For j As Integer = 0 To 12
          'Assign numbers from 0 through 12 to the
          'cards of each suit.
          m_deck(i * 13 + j) = New Card()
          m_deck(i * 13 + j).Suit = suits(i)
          m_deck(i * 13 + j).Denomination = j
        Next
      Next
    End Sub

    Function ReadCard(ByVal cardNum As Integer) As String
      Return m_deck(cardNum).IdentifyCard()
    End Function

    Sub Swap(ByVal i As Integer, ByVal j As Integer)
      'Swap the ith and jth cards in the deck.
      Dim tempCard As Card
      tempCard = m_deck(i)
      m_deck(i) = m_deck(j)
      m_deck(j) = tempCard
    End Sub
    Sub ShuffleDeck()
      'Do 2000 passes through the deck.  On each pass,
      'swap each card with a randomly selected card.
      Dim index As Integer
      Dim randomNum As New Random()
      For i As Integer = 1 To 2000
        For k As Integer = 0 To 51
          index = randomNum.Next(0, 52)  'Randomly select a number
          'from 0 through 51 inclusive.
          Swap(k, index)
        Next
        RaiseEvent Shuffling(i, 2000)
      Next
    End Sub
  End Class      'DeckOfCards
```

[Run, click on the *Shuffle* button, and click on the *Deal* button after the shuffling is complete.]



```
8 of Hearts
7 of Diamonds
Jack of Hearts
Queen of Spades
10 of Hearts
```

Practice Problems 11.2

Consider the program in Example 1 of Section 11.1, and suppose that mtbSSN is an ordinary (rather than a masked) text box.

1. Alter the Set SocSecNum property procedure to raise the event ImproperSSN when the social security number does not have 11 characters. The event should pass the length of the social security number and the student's name to the form's code.
2. What statement must be placed in the Declarations section of the Student class?
3. Write an event procedure to handle the event ImproperSSN.
4. What statement in the form's code must be altered?

EXERCISES 11.2

1. In Example 1 of this section, modify the event procedure btnDisplay_Click so that only the students who receive a grade of A are displayed.

The file UnitedStates.txt provides data on the 50 states. (This file is used in Exercises 2 through 5.) Each record contains five pieces of information about a single state: name, abbreviation, date it entered the union, land area (in square miles), and population in

the year 2000. The records are ordered by the date of entry into the union. The first three lines of the file are

```
Delaware,DE,12/7/1787,1954,759000
Pennsylvania,PA,12/12/1787,44817,12296000
New Jersey,NJ,12/18/1787,7417,8135000
```

2. Create a class State with five properties to hold the information about a single state and a method that calculates the density (people per square mile) of the state.
3. Write a program that requests a state's name in an input dialog box and displays the state's abbreviation, density, and date of entrance into the union. The program should use an array of State objects.
4. Write a program that displays the names of the states and their densities in a DataGridView ordered by density. The program should use an array of State objects.
5. Write a program that reads the data from the file one line at a time into an array of State objects and raises an event whenever the population of a state exceeds ten million. States with a large population should have their names and populations displayed in a list box by the corresponding event procedure. (***Hint***: Create a class called UnitedStates that contains the array and defines a method Add that adds a new state to the array. The Add method should raise the event.)
6. Consider the class Square from Exercise 19 of Section 11.1. Add the event IllegalNumber that is raised when any of the properties is set to a negative number. Show the new class block and write an event procedure for the event that displays an error message.
7. Consider the CashRegister class in Exercise 25 of Section 11.1. Add the event AttemptToOverdraw that is raised when the user tries to subtract more money than is in the cash register.
8. Consider the class PairOfDice discussed in Exercise 21 of Section 11.1. Add the event SnakeEyes that is raised whenever two ones appear during a roll of the dice. Write a program that uses this event.
9. Consider the Fraction class in Exercise 29 of Section 11.1. Add the event ZeroDenominator that is raised whenever a denominator is set to 0. Write a program that uses the event.
10. Write a program for the fraction calculator shown in Fig. 11.3. After the numerators and denominators of the two fractions to the left of the equals sign are placed in the four text boxes, one of four operations buttons should be clicked on. The result appears to the right of the equals sign. The program should use a Calculator class, which contains three members of the type Fraction discussed in Exercise 28 of Section 11.1. ***Note:*** In Fig. 11.3, the fraction bars are very short list boxes.

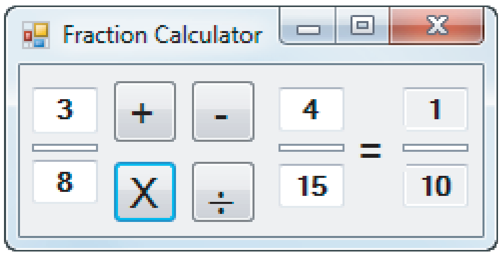


FIGURE 11.3 Sample Output for Exercise 10.

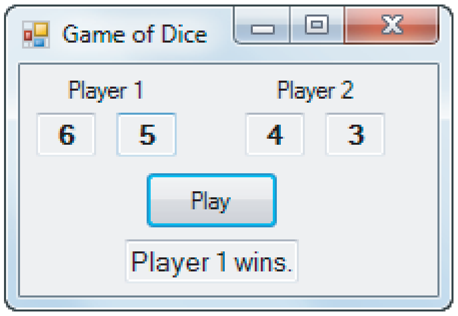
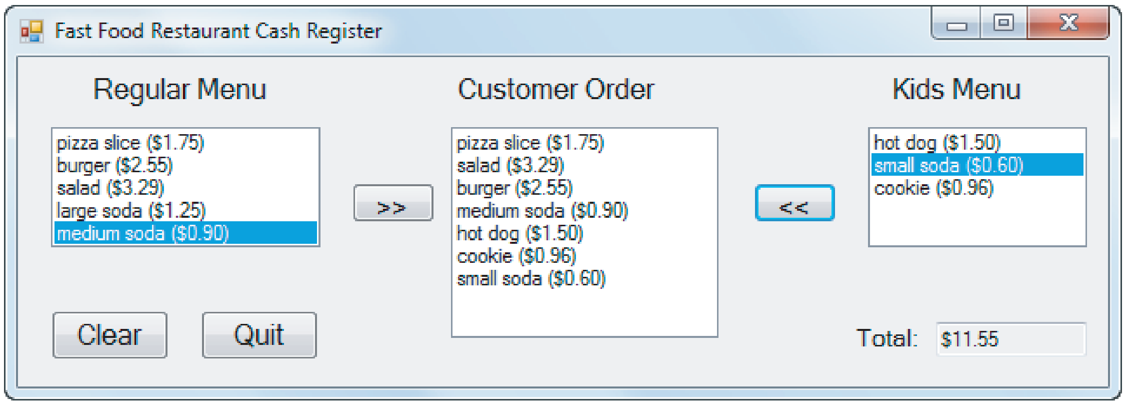


FIGURE 11.4 Sample Output for Exercise 11.

11. Write a program for a simple game in which each of two players rolls a pair of dice. The person with the highest tally wins. See Fig. 11.4. The program should use a class called HighRoller having two member variables of the type PairOfDice discussed in Exercise 21 of Section 11.1.

12. Write a program that takes orders at a fast food restaurant. See Fig. 11.5. The restaurant has two menus—a regular menu and a kids menu. An item is ordered by highlighting it in one of the list boxes and then clicking on the >> or << button to place it in the order list box in the center of the form. As each item is ordered, a running total is displayed in the text box at the lower right part of the form. The program should use a Choices class, which contains a Food class. (The contents of each of the three list boxes should be treated as Choices objects.) Each Food object should hold the name and price of a single food item. The Choices class should have a ChoiceChanged event that can be used by the form code to update the cost of the order.



Regular Menu:

```
pizza slice ($1.75)
burger ($2.55)
salad ($3.29)
large soda ($1.25)
medium soda ($0.90)
```

Customer Order:

```
pizza slice ($1.75)
salad ($3.29)
burger ($2.55)
medium soda ($0.90)
hot dog ($1.50)
cookie ($0.96)
small soda ($0.60)
```

Kids Menu:

```
hot dog ($1.50)
small soda ($0.60)
cookie ($0.96)
```

Total: $11.55

FIGURE 11.5 Sample output for Exercise 12.

13. Write a program to produce an employee's weekly paycheck receipt. The receipt should contain the employee's name, amount earned for the week, total amount earned for the year, FICA tax deduction, withholding tax deduction, and take-home amount. The program should use an Employee class and a Tax class. The Tax class must have properties for the amount earned for the week, the prior total amount earned for the year, the number of withholding allowances, and marital status. It should have methods for computing FICA and withholding taxes. The Employee class should store the employee's name, number of withholding allowances, marital status, hours worked this week, hourly salary, and previous amount earned for the year. The Employee class should use the Tax class to calculate the taxes to be deducted. The formula for calculating the FICA tax is given in Exercise 27 of Section 11.1. To compute the withholding tax, multiply the number of withholding allowances by $70.19, subtract the product from the amount earned, and use Table 11.1 or Table 11.2.

TABLE 11.1 2009 Federal income tax withheld for a single person paid weekly.

Adjusted Weekly Income	Income Tax Withheld
$0 to $138	$0
Over $138 to $200	10% of amount over $138
Over $200 to $696	$6.20 + 15% of amount over $200
Over $696 to $1,279	$80.60 + 25% of amount over $696
Over $1,279 to $3,338	$226.35 + 28% of amount over $1,279
Over $3,338 to $7,212	$802.87 + 33% of amount over $3,338
Over $7,212	$2,081.29 + 35% of amount over $7,212

TABLE 11.2 2009 Federal income tax withheld for a married person paid weekly.

Adjusted Weekly Income	Income Tax Withheld
$0 to $303	$0
Over $303 to $ 470	10% of amount over $303
Over $470 to $1,455	$16.70 + 15% of amount over $470
Over $1,455 to $2,272	$164.45 + 25% of amount over $1,455
Over $2,272 to $4,165	$368.70 + 28% of amount over $2,272
Over $4,165 to $7,321	$898.74 + 33% of amount over $4,165
Over $7,321	$1,940.22 + 35% of amount over $7,321

Solutions to Practice Problems 11.2

1.
```
Public Property SocSecNum() As String
  Get
    Return m_ssn
  End Get
  Set(ByVal value As String)
    If value.Length = 11 Then
      m_ssn = value
    Else
      RaiseEvent ImproperSSN(value.Length, m_name)
    End If
  End Set
End Property
```

2.
```
Public Event ImproperSSN(ByVal length As Integer,
                         ByVal studentName As String)
```

3.
```
Private Sub pupil_ImproperSSN(ByVal length As Integer,
             ByVal studentName As string) Handles pupil.ImproperSSN
  MessageBox.Show("The social security number entered for " &
        studentName & " consisted of " & length &
        " characters. Reenter the data for " & studentName & ".")
End Sub
```

4. The statement

```
Dim pupil As Student
```

must be changed to

```
Dim WithEvents pupil As Student
```

11.3 Inheritance

VideoNote Inheritance

The three relationships between classes are “use,” “containment,” and “inheritance.” One class **uses** another class if it manipulates objects of that class. We say that class A **contains** class B when a member variable of class A makes use of an object of type class B. Section 11.2 presents examples of use and containment.

Inheritance is a process by which one class (the **child** or **derived** class) inherits the properties, methods, and events of another class (the **parent** or **base** class). The child has access to all of its parent’s properties, methods and events as well as to all of its own. If the parent is itself a