INTERNATIONAL EDITION

Starting Out With Visual Basic® 2012

SIXTH EDITION

Tony Gaddis • Kip Irvine

ALWAYS LEARNING

PEARSON

This page intentionally left blank

4.8 The Select Case Statement

CONCEPT: **In a `Select Case` statement, one of several possible actions is taken, depending on the value of an expression.**

The `If...Then...ElseIf` statement allows your program to branch into one of several possible paths. It performs a series of tests and branches when one of these tests is true. The **`Select Case` statement**, which is a similar mechanism, tests the value of an expression only once, and then uses that value to determine which set of statements to branch to. Following is the general format of the `Select Case` statement. The items inside the brackets are optional.

```
Select Case TestExpression
  [Case ExpressionList
    [one or more statements]]
  [Case ExpressionList
    [one or more statements]]
  [Case Else
    [one or more statements]]
End Select
```

`Case` statements may be repeated as many times as necessary.

The first line starts with `Select Case` and is followed by a test expression. The test expression may be any numeric or string expression that you wish to test.

Starting on the next line is a sequence of one or more `Case` statements. Each `Case` statement follows this general form:

```
Case ExpressionList
  one or more statements
```

After the word `Case` is an expression list, so-called because it may hold one or more expressions. Beginning on the next line, one or more statements appear. These statements are executed if the value of the test expression matches any of the expressions in the `Case` statement's expression list.

A `Case Else` comes after all the `Case` statements. This branch is selected if none of the `Case` expression lists match the test expression. The entire `Select Case` construct is terminated with an `End Select` statement.

WARNING: The `Case Else` section is optional. If you leave it out, however, your program will have nowhere to branch to if the test expression doesn't match any of the expressions in the `Case` expression lists.

Here is an example of the `Select Case` statement:

```
Select Case CInt(txtInput.Text)
  Case 1
    MessageBox.Show("Day 1 is Monday.")
  Case 2
    MessageBox.Show("Day 2 is Tuesday.")
  Case 3
    MessageBox.Show("Day 3 is Wednesday.")
```

```
    Case 4
      MessageBox.Show("Day 4 is Thursday.")
    Case 5
      MessageBox.Show("Day 5 is Friday.")
    Case 6
      MessageBox.Show("Day 6 is Saturday.")
    Case 7
      MessageBox.Show("Day 7 is Sunday.")
    Case Else
      MessageBox.Show("That value is invalid.")
End Select
```

Let's look at this example more closely. The test expression is `CInt(txtInput.Text)`. The `Case` statements `Case 1`, `Case 2`, `Case 3`, `Case 4`, `Case 5`, `Case 6`, and `Case 7` mark where the program is to branch to if the test expression is equal to the values 1, 2, 3, 4, 5, 6, or 7. The `Case Else` section is branched to if the test expression is not equal to any of these values.

Suppose the user has entered 3 into the `txtInput` text box, so the expression `CInt(txtInput.Text)` is equal to 3. Visual Basic compares this value with the first `Case` statement's expression list:

```
Select Case CInt(txtInput.Text)
➡ Case 1
    MessageBox.Show("Day 1 is Monday.")
```

The only value in the expression list is 1, and this is not equal to 3, so Visual Basic goes to the next `Case`:

```
Select Case CInt(txtInput.Text)
  Case 1
    MessageBox.Show("Day 1 is Monday.")
➡ Case 2
    MessageBox.Show("Day 2 is Tuesday.")
```

Once again, the value in the expression list does not equal 3, so Visual Basic goes to the next `Case`:

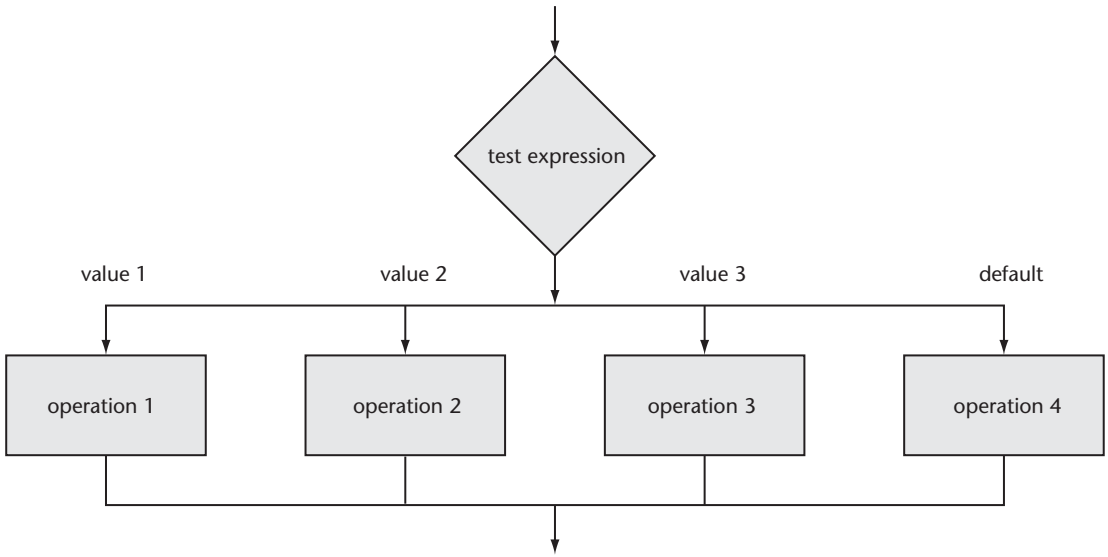
```
Select Case CInt(txtInput.Text)
  Case 1
    MessageBox.Show("Day 1 is Monday.")
  Case 2
    MessageBox.Show("Day 2 is Tuesday.")
➡ Case 3
    MessageBox.Show("Day 3 is Wednesday.")
```

This time, the value in the `Case`'s expression list matches the value of the test expression, so the `MessageBox.Show` statement on the next line executes. (If there had been multiple statements appearing between the `Case 3` and `Case 4` statements, all would have executed.) After the `MessageBox.Show` statement executes, the program jumps to the statement immediately following the `End Select` statement.

Flowcharting the `Select Case` Statement

The flowchart segment in Figure 4-20 shows the general form of a `Select Case` statement. The diamond represents the test expression, which is compared to a series of values. The path of execution follows the value matching the test expression. If none of the values matches a test expression, the default path is followed (`Case Else`).

Figure 4-20 General form of a `Select Case` statement



More about the Expression List

The `Case` statement's expression list can contain multiple expressions, separated by commas. For example, the first `Case` statement in the following code compares `intNumber` to 1, 3, 5, 7, and 9, and the second `Case` statement compares it to 2, 4, 6, 8, and 10. In the following code, assume that `strStatus` is a string variable:

```
Select Case intNumber
  Case 1, 3, 5, 7, 9
    strStatus = "Odd"
  Case 2, 4, 6, 8, 10
    strStatus = "Even"
  Case Else
    strStatus = "Out of Range"
End Select
```

The `Case` statement can also test string values. In the following code, assume that `strAnimal` is a string variable:

```
Select Case strAnimal
  Case "Dogs", "Cats"
    MessageBox.Show("House Pets")
  Case "Cows", "Pigs", "Goats"
    MessageBox.Show("Farm Animals")
  Case "Lions", "Tigers", "Bears"
    MessageBox.Show("Oh My!")
End Select
```

You can use relational operators in the `Case` statement, as shown by the following example. The `Is` keyword represents the test expression in the relational comparison.

```
Select Case dblTemperature
  Case Is <= 75
    blnTooCold = True
  Case Is >= 100
    blnTooHot = True
  Case Else
    blnJustRight = True
End Select
```

Finally, you can determine whether the test expression falls within a range of values. This requires the To keyword, as shown in the following code.

```
Select Case intScore
  Case Is >= 90
    strGrade = "A"
  Case 80 To 89
    strGrade = "B"
  Case 70 To 79
    strGrade = "C"
  Case 60 To 69
    strGrade = "D"
  Case 0 To 59
    strGrade = "F"
  Case Else
    MessageBox.Show("Invalid Score")
End Select
```

The numbers used on each side of the To keyword are included in the range. So, the statement `Case 80 To 89` matches the values 80, 89, or any number in between.

TIP: The To keyword works properly only when the smaller number appears on its left and the larger number appears on its right. You can write an expression such as `10 To 0`, but it will not function properly at runtime.

Tutorial 4-7 examines a sales commission calculator application.

Tutorial 4-7:

Examining *Crazy Al's Sales Commission Calculator* application

Crazy Al's Computer Emporium is a retail seller of personal computers. The sales staff at Crazy Al's works strictly on commission. At the end of the month, each salesperson's commission is calculated according to Table 4-8.

For example, a salesperson with $16,000 in monthly sales earns a 12% commission ($1,920.00). A salesperson with $20,000 in monthly sales earns a 14% commission ($2,800.00).

Table 4-8 Sales commission rates

Sales This Month	Commission Rate
Less than $10,000	5%
$10,000 – $14,999	10%
$15,000 – $17,999	12%
$18,000 – $21,999	14%
$22,000 or more	16%

Because the staff is paid once per month, Crazy Al's allows each employee to take up to $1,500 per month in advance pay. When sales commissions are calculated, the amount of each employee's advance pay is subtracted from the commission. If any salesperson's commission is less than the amount of the advance, he or she must reimburse Crazy Al's for the difference.

Here are two examples:

- Beverly's monthly sales were $21,400, so her commission is $2,996. She took $1,500 in advance pay. At the end of the month she gets a check for $1,496.
- John's monthly sales were $12,600, so his commission is $1,260. He took $1,500 in advance pay. At the end of the month he must pay back $240 to Crazy Al's.

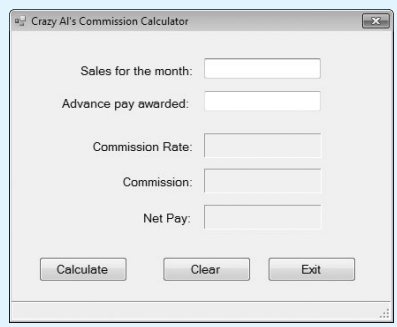
In this tutorial, you examine the *Crazy Al's Commission Calculator* application used to determine a salesperson's commission.

Step 1: Open the *Crazy Al* project from the student sample programs folder named *Chap4\Crazy Al*.

Step 2: Run the application. The form shown in Figure 4-21 appears.

Step 3: Enter **16000** as the amount of sales for this month (first text box).

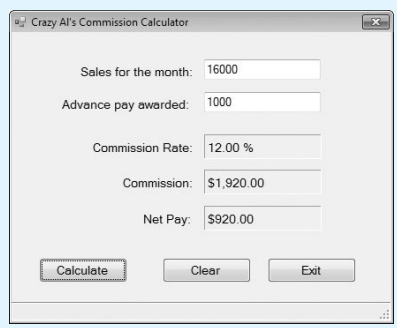
Figure 4-21 *Crazy Al's Commission Calculator* form



Step 4: Enter **1000** as the amount of advance pay taken (second text box). Click the *Calculate* button. You should see the commission rate, commission, and net pay information, as shown in Figure 4-22.

Step 5: Click the *Clear* button to reset the contents of the input and display fields.

Figure 4-22 Calculations filled in



Experiment with other values for sales and advance pay.