4

Control Statements: Part 1

Let's all move one place on.
—Lewis Carroll

The wheel is come full circle.
—William Shakespeare

How many apples fell on Newton's head before he took the hint!
—Robert Frost

All the evolution we know of proceeds from the vague to the definite.
—Charles Sanders Peirce

OBJECTIVES

In this chapter you will learn:

- To use basic problem-solving techniques.
- To develop algorithms through the process of top-down, stepwise refinement using pseudocode.
- To use the `if` and `if...else` selection statements to choose among alternative actions.
- To use the `while` repetition statement to execute statements in a program repeatedly.
- To use counter-controlled repetition and sentinel-controlled repetition.
- To use the compound assignment, increment and decrement operators.
- The primitive data types.

Student Solution Exercises

4.11 Explain what happens when a Java program attempts to divide one integer by another. What happens to the fractional part of the calculation? How can a programmer avoid that outcome?

ANS: Dividing two integers results in integer division—any fractional part of the calculation is lost (i.e., truncated). For example, 7 ÷ 4, which yields 1.75 in conventional arithmetic, truncates to 1 in integer arithmetic, rather than rounding to 2. To obtain a floating-point result from dividing integer values, a programmer must temporarily treat these values as floating-point numbers in the calculation by using the unary cast operator `(double)`. As long as the `(double)` cast operator is applied to any variable in the calculation, the calculation will yield a `double` result which can be assigned to a `double` variable.

4.13 What type of repetition would be appropriate for calculating the sum of the first 100 positive integers? What type of repetition would be appropriate for calculating the sum of an arbitrary number of positive integers? Briefly describe how each of these tasks could be performed.

ANS: Counter-controlled repetition would be appropriate for calculating the sum of the first 100 positive integers because the number of repetitions is known in advance. The program that performs this task could use a `while` repetition statement with a counter variable that takes on the values 1 to 100. The program could then add the current counter value to the total variable in each repetition of the loop. Sentinel-controlled repetition would be appropriate for calculating the sum of an arbitrary number of positive integers. The program that performs this task could use a sentinel value of –1 to mark the end of data entry. The program could use a `while` repetition statement that totals positive integers from the user until the user enters the sentinel value.

4.15 Identify and correct the errors in each of the following pieces of code. [*Note:* There may be more than one error in each piece of code.]

a)
```
if ( age >= 65 );
   System.out.println( "Age greater than or equal to 65" );
else
   System.out.println( "Age is less than 65 )";
```

ANS: The semicolon at the end of the `if` condition should be removed. The closing double quote of the second `System.out.println` should be inside the closing parenthesis.

b)
```
int x = 1, total;
while ( x <= 10 )
{
   total += x;
   ++x;
}
```

ANS: The variable `total` should be initialized to zero.

c)
```
while ( x <= 100 )
   total += x;
   ++x;
```

ANS: The two statements should be enclosed in curly braces to properly group them into the body of the `while`; otherwise the loop will be an infinite loop.

d)
```
while ( y > 0 )
{
   System.out.println( y );
   ++y;
```

ANS: The ++ operator should be changed to --; otherwise the loop will be an infinite loop. The closing curly brace for the `while` loop is missing.

4.17 Drivers are concerned with the mileage their automobiles get. One driver has kept track of several tankfuls of gasoline by recording the miles driven and gallons used for each tankful. Develop a Java application that will input the miles driven and gallons used (both as integers) for each tankful. The program should calculate and display the miles per gallon obtained for each tankful and print the combined miles per gallon obtained for all tankfuls up to this point. All averaging calculations should produce floating-point results. Use class `Scanner` and sentinel-controlled repetition to obtain the data from the user.

ANS:

```
1  // Exercise 4.17 Solution: Gas.java
2  // Program calculates average mpg
3  import java.util.Scanner;
4
5  public class Gas
6  {
7     // perform miles-per-gallon calculations
8     public void calculateMPG()
9     {
10       Scanner input = new Scanner( System.in );
11
12       int miles; // miles for one tankful
13       int gallons; // gallons for one tankful
14       int totalMiles = 0; // total mailes for trip
15       int totalGallons = 0; // total gallons for trip
16
17       double milesPerGallon; // miles per gallon for tankful
18       double totalMilesPerGallon; // miles per gallon for trip
19
20       // prompt user for miles and obtain the input from user
21       System.out.print( "Enter miles (-1 to quit): " );
22       miles = input.nextInt();
23
24       // exit if the input is -1; otherwise, proceed with the program
25       while ( miles != -1 )
26       {
27          // prompt user for gallons and obtain the input from user
28          System.out.print( "Enter gallons:" );
29          gallons = input.nextInt();
30
31          // add gallons and miles for this tank to total
32          totalMiles += miles;
33          totalGallons += gallons;
34
35          // calculate miles per gallon for the current tank
36          if ( gallons != 0 )
37          {
38             milesPerGallon = (double) miles / gallons;
```

```
39             System.out.printf( "MPG this tankful: %.2f\n",
40                milesPerGallon );
41          } // end if statement
42
43          // calculate miles per gallon for the total trip
44          if ( totalGallons != 0 )
45          {
46             totalMilesPerGallon = (double) totalMiles / totalGallons;
47             System.out.printf( "Total MPG: %.2f\n",
48                totalMilesPerGallon );
49          } // end if statement
50
51          // prompt user for new value for miles
52          System.out.print( "Enter miles (-1 to quit): " );
53          miles = input.nextInt();
54       } // end while loop
55    } // end method calculateMPG
56 } // end class Gas
```

```
1  // Exercise 4.17 Solution: GasTest.java
2  // Test application for class Gas
3  public class GasTest
4  {
5     public static void main( String args[] )
6     {
7        Gas application = new Gas();
8        application.calculateMPG();
9     } // end main
10 } // end class GasTest
```

```
Enter miles (-1 to quit): 350
Enter gallons: 18
MPG this tankful: 19.44
Total MPG: 19.44
Enter miles (-1 to quit): 475
Enter gallons: 16
MPG this tankful: 29.69
Total MPG: 24.26
Enter miles (-1 to quit): 400
Enter gallons: 17
MPG this tankful: 23.53
Total MPG: 24.02
Enter miles (-1 to quit): -1
```

4.19 A large company pays its salespeople on a commission basis. The salespeople receive $200 per week plus 9% of their gross sales for that week. For example, a salesperson who sells $5000 worth of merchandise in a week receives $200 plus 9% of $5000, or a total of $650. You have been supplied with a list of the items sold by each salesperson. The values of these items are as follows:

```
Item    Value
1       239.99
2       129.75
```

```
3         99.95
4        350.89
```

Develop a Java application that inputs one salesperson's items sold for last week and calculates and displays that salesperson's earnings. There is no limit to the number of items that can be sold by a salesperson.

ANS:

```
1  // Exercise 4.19 Solution: Sales.java
2  // Program calculates commissions based on sales.
3  import java.util.Scanner;
4
5  public class Sales
6  {
7     // calculate sales for individual products
8     public void calculateSales()
9     {
10       Scanner input = new Scanner( System.in );
11
12       double gross = 0.0; // total gross sales
13       double earnings; // earnings made from sales
14       int product = 0; // the product number
15       int numberSold; // number sold of a given product
16
17       while ( product < 4 )
18       {
19          product++;
20
21          // prompt and read number of the product sold from user
22          System.out.printf( "Enter number sold of product #%d: ",
23             product );
24          numberSold = input.nextInt();
25
26          // determine gross of each individual product and add to total
27          if ( product == 1 )
28             gross += numberSold * 239.99;
29          else if ( product == 2 )
30             gross += numberSold * 129.75;
31          else if ( product == 3 )
32             gross += numberSold * 99.95;
33          else if ( product == 4 )
34             gross += numberSold * 350.89;
35       } // end while loop
36
37       earnings = 0.09 * gross + 200; // calculate earnings
38       System.out.printf( "Earnings this week: $%.2f\n", earnings );
39    } // end method calculateSales
40 } // end class Sales
```

```
1  // Exercise 4.19 Solution: SalesTest.java
2  // Test application for class Sales
3  public class SalesTest
4  {
```

```
5     public static void main( String args[] )
6     {
7        Sales application = new Sales();
8        application.calculateSales();
9     } // end main
10 } // end class SalesTest
```

```
Enter number sold of product #1: 100
Enter number sold of product #2: 65
Enter number sold of product #3: 854
Enter number sold of product #4: 193
Earnings this week: $16896.06
```

4.22 Write a Java application that uses looping to print the following table of values:

```
N       10*N    100*N   1000*N

1       10      100     1000
2       20      200     2000
3       30      300     3000
4       40      400     4000
5       50      500     5000
```

ANS:

```
1  // Exercise 4.22 Solution: Table.java
2  // Program prints a table of values using a while loop.
3
4  public class Table
5  {
6     public static void main( String args[] )
7     {
8        int n = 1;
9
10       System.out.println( "N\t10*N\t100*N\t1000*N\n" );
11
12       while ( n <= 5 )
13       {
14          System.out.printf( "%d\t%d\t%d\t%d\n",
15             n,  ( 10 * n ), ( 100 * n ), ( 1000 * n ) );
16          n++;
17       } // end while loop
18    } // end main
19 } // end class Table
```

```
N       10*N    100*N   1000*N

1       10      100     1000
2       20      200     2000
3       30      300     3000
4       40      400     4000
5       50      500     5000
```

4.24 Modify the program in Fig. 4.12 to validate its inputs. For any input, if the value entered is other than 1 or 2, keep looping until the user enters a correct value.

ANS:

```
1  // Exercise 4.24 Solution: Analysis.java
2  // Program performs analysis of examination results.
3  import java.util.Scanner; // program uses class Scanner
4
5  public class Analysis
6  {
7     // anaylze the results of 10 tests
8     public void processExamResults()
9     {
10       // create Scanner to obtain input from command window
11       Scanner input = new Scanner( System.in );
12
13       // initializing variables in declarations
14       int passes = 0; // number of passes
15       int failures = 0; // number of failures
16       int studentCounter = 1; // student counter
17       int result; // one exam result
18
19       // process 10 students using counter-controlled loop
20       while ( studentCounter <= 10 )
21       {
22          // prompt user for input and obtain value from user
23          System.out.print( "Enter result (1 = pass, 2 = fail): " );
24          result = input.nextInt();
25
26          // if...else nested in while
27          if ( result == 1 )
28          {
29             passes = passes + 1;
30             studentCounter = studentCounter + 1;
31          } // end if
32          else if ( result == 2 )
33          {
34             failures = failures + 1;
35             studentCounter = studentCounter + 1;
36          } // end else if
37          else
38             System.out.println( "Invalid Input" );
39       } // end while
40
41       // termination phase; prepare and display results
```

```
42         System.out.printf( "Passed: %d\nFailed: %d\n", passes, failures );
43
44         // determine whether more than 8 students passed
45         if ( passes > 8 )
46            System.out.println( "Raise Tuition" );
47      } // end method processExamResults
48   } // end class Analysis
```

```
1   // Exercise 4.24 Solution: AnalysisTest.java
2   // Test application for class Analysis
3   public class AnalysisTest
4   {
5      public static void main( String args[] )
6      {
7         Analysis application = new Analysis();
8         application.processExamResults();
9      } // end main
10  } // end class AnalysisTest
```

```
Enter result (1 = pass, 2 = fail): 3
Invalid Input
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Passed: 8
Failed: 2
```

4.25 What does the following program print?

```
1   public class Mystery2
2   {
3      public static void main( String args[] )
4      {
5         int count = 1;
6
7         while ( count <= 10 )
8         {
9            System.out.println( count % 2 == 1 ? "****" : "++++++++" );
10           ++count;
11        } // end while
12     } // end main
13
14  } // end class Mystery2
```

ANS:

```
****
++++++++
****
++++++++
****
++++++++
****
++++++++
****
++++++++
```

4.26 What does the following program print?

```
1  public class Mystery3
2  {
3     public static void main( String args[] )
4     {
5        int row = 10;
6        int column;
7
8        while ( row >= 1 )
9        {
10          column = 1;
11
12          while ( column <= 10 )
13          {
14             System.out.print( row % 2 == 1 ? "<" : ">" );
15             ++column;
16          } // end while
17
18          --row;
19          System.out.println();
20       } // end while
21    } // end main
22
23 } // end class Mystery3
```

ANS:

```
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
```

4.27 *(Dangling-`else` Problem)* Determine the output for each of the given sets of code when `x` is `9` and `y` is `11` and when `x` is `11` and `y` is `9`. Note that the compiler ignores the indentation in a Java program. Also, the Java compiler always associates an `else` with the immediately preceding `if` unless told to do otherwise by the placement of braces (`{}`). On first glance, the programmer may not be sure which `if` an `else` matches—this situation is referred to as the "dangling-`else` problem." We have eliminated the indentation from the following code to make the problem more challenging. [*Hint*: Apply the indentation conventions you have learned.]

a)
```
if ( x < 10 )
if ( y > 10 )
System.out.println( "*****" );
else
System.out.println( "#####" );
System.out.println( "$$$$$" );
```

ANS:

```
When:  x = 9, y = 11
*****
$$$$$

When:  x = 11, y = 9
$$$$$
```

4.29 Write an application that prompts the user to enter the size of the side of a square, then displays a hollow square of that size made of asterisks. Your program should work for squares of all side lengths between 1 and 20.

ANS:

```
1  // Exercise 4.29 Solution: Hollow.java
2  // Program prints a hollow square.
3  import java.util.Scanner;
4
5  public class Hollow
6  {
7     // draw a hollow box surrounded by stars
8     public void drawHollowBox()
9     {
10       Scanner input = new Scanner( System.in );
11
12       int stars; // number of stars on a side
13       int column; // the current column of the square being printed
14       int row = 1; // the current row of the square being printed
15
16       // prompt and read the length of the side from the user
17       System.out.print( "Enter length of side:" );
18       stars = input.nextInt();
19
20       if ( stars < 1 )
21       {
22          stars = 1;
23          System.out.println( "Invalid Input\nUsing default value 1" );
24       } // end if
```

```
25         else if ( stars > 20 )
26         {
27            stars = 20;
28            System.out.println( "Invalid Input\nUsing default value 20" );
29         } // end else if
30
31         // repeat for as many rows as the user entered
32         while ( row <= stars )
33         {
34            column = 1;
35
36            // and for as many columns as rows
37            while ( column <= stars )
38            {
39               if ( row == 1 )
40                  System.out.print( "*" );
41               else if ( row == stars )
42                  System.out.print( "*" );
43               else if ( column == 1 )
44                  System.out.print( "*" );
45               else if ( column == stars )
46                  System.out.print( "*" );
47               else
48                  System.out.print( " " );
49
50               column++;
51            } // end inner while loop
52
53            System.out.println();
54            row++;
55         } // end outer while loop
56      } // end method drawHollowBox
57   } // end class Hollow
```

```
1   // Exercise 4.29 Solution: HollowTest.java
2   // Test application for class Hollow
3   public class HollowTest
4   {
5      public static void main( String args[] )
6      {
7         Hollow application = new Hollow();
8         application.drawHollowBox();
9      } // end main
10  } // end class HollowTest
```

```
Enter length of side:15
***************
*             *
*             *
*             *
*             *
*             *
*             *
*             *
*             *
*             *
*             *
*             *
*             *
*             *
***************
```

4.32 Write an application that uses only the output statements

```
System.out.print( "* " );
System.out.print( " " );
System.out.println();
```

to display the checkerboard pattern that follows. Note that a `System.out.println` method call with no arguments causes the program to output a single newline character. [*Hint*: Repetition statements are required.]

```
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *
```

ANS:

```
1  // Exercise 4.32 Solution: Stars.java
2  // Program prints a checkerboard pattern.
3  public class Stars
4  {
5     public static void main( String args[] )
6     {
7        int row = 1;
8
9        while ( row <= 8 )
10       {
11          int column = 1;
12
13          if ( row % 2 == 0 )
14             System.out.print( " " );
```

```
15
16            while ( column <= 8 )
17            {
18               System.out.print( "* " );
19               column++;
20            } // end inner while loop
21
22            System.out.println();
23            row++;
24         } // end outer while loop
25      } // end main
26   } // end class Stars
```

```
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *
```

4.36 Write an application that reads three nonzero integers and determines and prints whether they could represent the sides of a right triangle.

ANS:

```
1  // Exercise 4.36 Solution: Triangle2.java
2  // Program takes three integers and determines if they
3  // form the sides of a right triangle.
4  import java.util.Scanner;
5
6  public class Triangle2
7  {
8     // checks if three sides can form a right triangle
9     public void checkSides()
10    {
11       Scanner input = new Scanner( System.in );
12
13       int side1; // length of side 1
14       int side2; // length of side 2
15       int side3; // length of side 3
16       boolean isRightTriangle; // whether the sides can form a triangle
17
18       // get values of three sides
19       System.out.print( "Enter side 1: " );
20       side1 = input.nextInt();
21
22       System.out.print( "Enter side 2: " );
23       side2 = input.nextInt();
24
25       System.out.print( "Enter side 3: " );
26       side3 = input.nextInt();
27
```

```
28         // square the sides
29         int side1Square = side1 * side1;
30         int side2Square = side2 * side2;
31         int side3Square = side3 * side3;
32
33         // test if these form a right triangle
34         isRightTriangle = false;
35
36         if ( ( side1Square + side2Square ) == side3Square )
37           isRightTriangle = true;
38         else if ( ( side1Square + side3Square ) == side2Square )
39            isRightTriangle = true;
40         else if ( ( side2Square + side3Square ) == side1Square )
41            isRightTriangle = true;
42
43         if ( isRightTriangle )
44            System.out.println( "These are the sides of a right triangle." );
45         else
46            System.out.println( "These do not form a right triangle." );
47      } // end method checkSides
48   } // end class Triangle2
```

```
1   // Exercise 4.36 Solution: Triangle2Test.java
2   // Test application for class Triangle2
3   public class Triangle2Test
4   {
5      public static void main( String args[] )
6      {
7         Triangle2 application = new Triangle2();
8         application.checkSides();
9      } // end main
10  } // end class Triangle2Test
```

```
Enter side 1: 5
Enter side 2: 4
Enter side 3: 3
These are the sides of a right triangle
```