5

Control Statements: Part 2

Not everything that can be counted counts, and not every thing that counts can be counted.
—Albert Einstein

Who can control his fate?
—William Shakespeare

The used key is always bright.
—Benjamin Franklin

Intelligence … is the faculty of making artificial objects, especially tools to make tools.
—Henri Bergson

Every advantage in the past is judged in the light of the final issue.
—Demosthenes

OBJECTIVES

In this chapter you will learn:

- The essentials of counter-controlled repetition.
- To use the `for` and `do...while` repetition statements to execute statements in a program repeatedly.
- To understand multiple selection using the `switch` selection statement.
- To use the `break` and `continue` program control statements to alter the flow of control.
- To use the logical operators to form complex conditional expressions in control statements.

Student Solution Exercises

5.7 Describe the four basic elements of counter-controlled repetition.

ANS: Counter-controlled repetition requires a control variable (or loop counter), an initial value of the control variable, an increment (or decrement) by which the control variable is modified each time through the loop, and a loop-continuation condition that determines whether looping should continue.

5.9 Find and correct the error(s) in each of the following segments of code:

a)
```
For ( i = 100, i >= 1, i++ )
   System.out.println( i );
```

ANS: The F in `for` should be lowercase. Semicolons should be used in the `for` header instead of commas. `++` should be `--`.

```
1  // Exercise 5.9a: PartA.java
2  public class PartA
3  {
4     public static void main( String args[] )
5     {
6        int i;
7
8        For ( i = 100, i >= 1, i++ )
9           System.out.println( i );
10    } // end main
11 } // end class PartA
```

```
PartA.java:9: ';' expected
         System.out.println( i );
         ^
1 error
```

```
1  // Exercise 5.9a Solution: PartACorrected.java
2  public class PartACorrected
3  {
4     public static void main( String args[] )
5     {
6        int i;
7
8        for ( i = 100; i >= 1; i-- )
9           System.out.println( i );
10    } // end main
11 } // end class PartACorrected
```

```
100
99
98
.
.
.
3
2
1
```

b) The following code should print whether integer `value` is odd or even:

```
switch ( value % 2 )
{
   case 0:
      System.out.println( "Even integer" );

   case 1:
      System.out.println( "Odd integer" );
}
```

ANS: A `break` statement should be placed in `case 0:`.

```
1  // Exercise 5.9b: PartB.java
2  public class PartB
3  {
4     public static void main( String args[] )
5     {
6        int value = 8;
7
8        switch ( value % 2 )
9        {
10          case 0:
11             System.out.println( "Even integer" );
12
13          case 1:
14             System.out.println( "Odd integer" );
15
16    } // end main
17 } // end class PartB
```

```
Even integer
Odd integer
```

```
1  // Exercise 5.9b Solution: PartBCorrected.java
2  public class PartBCorrected
3  {
4     public static void main( String args[] )
5     {
6        int value = 8;
7
8        switch ( value % 2 )
9        {
10          case 0:
11             System.out.println( "Even integer" );
12             break;
13          case 1:
14             System.out.println( "Odd integer" );
15       }
16    } // end main
17 } // end class PartBCorrected
```

```
Even integer
```

c) The following code should output the odd integers from 19 to 1:

```
for ( i = 19; i >= 1; i += 2 )
   System.out.println( i );
```

ANS: The += operator in the for header should be -=.

```
1  // Exercise 5.9c: PartC.java
2  public class PartC
3  {
4     public static void main( String args[] )
5     {
6        int i;
7
8        for ( i = 19; i >= 1; i += 2 )
9           System.out.println( i );
10    } // end main
11 } // end class PartC
```

```
19
21
23
25
27
29
.
.
.
```

```
1  // Exercise 5.9c Solution: PartCCorrected.java
2  public class PartCCorrected
3  {
4     public static void main( String args[] )
5     {
6        int i;
7
8        for ( i = 19; i >= 1; i -= 2 )
9           System.out.println( i );
10    } // end main
11 } // end class PartCCorrected
```

```
19
17
15
13
11
9
7
5
3
1
```

d) The following code should output the even integers from 2 to 100:

```
counter = 2;

do
{
   System.out.println( counter );
   counter += 2;
} While ( counter < 100 );
```

ANS: The `W` in `while` should be lowercase. < should be <=.

```
1  // Exercise 5.9d: PartD.java
2  public class PartD
3  {
4     public static void main( String args[] )
5     {
6        int counter;
7
8        counter = 2;
9
10       do
11       {
12          System.out.println( counter );
13          counter += 2;
14       } While ( counter < 100 );
15    } // end main
16 } // end class PartD
```

```
PartD.java:14: while expected
      } While ( counter < 100 );
        ^
PartD.java:14: '(' expected
      } While ( counter < 100 );
                          ^
2 errors
```

```
1  // Exercise 5.9d Solution: PartDCorrected.java
2  public class PartDCorrected
3  {
4     public static void main( String args[] )
5     {
6        int counter;
7
8        counter = 2;
9
10       do
11       {
12          System.out.println( counter );
13          counter += 2;
14       } while ( counter <= 100 );
15    } // end main
16 } // end class PartDCorrected
```

```
2
4
6
.
.
.
96
98
100
```

5.10 What does the following program do?

```
1  public class Printing
2  {
3     public static void main( String args[] )
4     {
5        for ( int i = 1; i <= 10; i++ )
6        {
7           for ( int j = 1; j <= 5; j++ )
8              System.out.print( '@' );
9
10          System.out.println();
11       } // end outer for
12    } // end main
13 } // end class Printing
```

ANS:

```
@@@@@
@@@@@
@@@@@
@@@@@
@@@@@
@@@@@
@@@@@
@@@@@
@@@@@
@@@@@
```

5.11 Write an application that finds the smallest of several integers. Assume that the first value read specifies the number of values to input from the user.

ANS:

```
1  // Exercise 5.11 Solution: Small.java
2  // Program finds the smallest of several integers.
3  import java.util.Scanner;
4
```

```
5  public class Small
6  {
7     // finds the smallest integer
8     public void findSmallest()
9     {
10       Scanner input = new Scanner( System.in );
11
12       int smallest = 0; // smallest number
13       int number = 0; // number entered by user
14       int integers; // number of integers
15
16       System.out.print( "Enter number of integers: " );
17       integers = input.nextInt();
18
19       for ( int counter = 1; counter <= integers; counter++ )
20       {
21          System.out.print( "Enter integer: " );
22          number = input.nextInt();
23
24          if ( counter == 1 )
25             smallest = number;
26          else if ( number < smallest )
27             smallest = number;
28       } // end for loop
29
30       System.out.printf( "Smallest Integer is: %d\n", smallest );
31    } // end method findSmallest
32 } // end of class Small
```

```
1  // Exercise 5.11 Solution: SmallTest.java
2  // Test application for class Small
3  public class SmallTest
4  {
5     public static void main(String args[])
6     {
7        Small application = new Small();
8        application.findSmallest();
9     } // end main
10 } // end class SmallTest
```

```
Enter number of integers: 3
Enter integer: -5
Enter integer: 46
Enter integer: 0
Smallest Integer is: -5
```

5.13 *Factorials* are used frequently in probability problems. The factorial of a positive integer *n* (written *n*! and pronounced "*n* factorial") is equal to the product of the positive integers from 1 to *n*. Write an application that evaluates the factorials of the integers from 1 to 5. Display the results in tabular format. What difficulty might prevent you from calculating the factorial of 20?

ANS: Calculating the factorial of 20 might be difficult because the value of 20! exceeds the maximum value that can be stored in an `int`.

```
1  // Exercise 5.13 Solution: Factorial.java
2  // Program calculates factorials.
3  public class Factorial
4  {
5     public static void main( String args[] )
6     {
7        System.out.println( "n\tn!\n" );
8
9        for ( int number = 1; number <= 5; number++ )
10       {
11          int factorial = 1;
12
13          for ( int smaller = 1; smaller <= number; smaller++ )
14             factorial *= smaller;
15
16          System.out.printf( "%d\t%d\n", number, factorial );
17       } // end for loop
18    } // end main
19 } // end class Factorial
```

```
n       n!

1       1
2       2
3       6
4       24
5       120
```

5.16 One interesting application of computers is to display graphs and bar charts. Write an application that reads five numbers between 1 and 30. For each number that is read, your program should display the same number of adjacent asterisks. For example, if your program reads the number 7, it should display *******.

ANS:

```
1  // Exercise 5.16 Solution: Graphs.java
2  // Program prints 5 histograms with lengths determined by user.
3  import java.util.Scanner;
4
5  public class Graphs
6  {
7     // draws 5 histograms
8     public void drawHistograms()
9     {
10       Scanner input = new Scanner( System.in );
11
```

```
12          int number1 = 0; // first number
13          int number2 = 0; // second number
14          int number3 = 0; // third number
15          int number4 = 0; // fourth number
16          int number5 = 0; // fifth number
17
18          int inputNumber; // number entered by user
19          int value = 0; // number of stars to print
20          int counter = 1; // counter for current number
21
22          while ( counter <= 5 )
23          {
24             System.out.print( "Enter number: " );
25             inputNumber = input.nextInt();
26
27             // define appropriate num if input is between 1-30
28             if ( inputNumber >= 1 && inputNumber <= 30 )
29             {
30                switch ( counter )
31                {
32                   case 1:
33                      number1 = inputNumber;
34                      break; // done processing case
35
36                   case 2:
37                      number2 = inputNumber;
38                      break; // done processing case
39
40                   case 3:
41                      number3 = inputNumber;
42                      break; // done processing case
43
44                   case 4:
45                      number4 = inputNumber;
46                      break; // done processing case
47
48                   case 5:
49                      number5 = inputNumber;
50                      break; // done processing case
51                } // end switch
52
53                counter++;
54             } // end if
55             else
56                System.out.println(
57                   "Invalid Input\nNumber should be between 1 and 30" );
58          } // end while
59
60          // print histograms
61          for ( counter = 1; counter <= 5; counter++ )
62          {
63             switch ( counter )
64             {
65                case 1:
66                   value = number1;
```

```
                break; // done processing case

             case 2:
                value = number2;
                break; // done processing case

             case 3:
                value = number3;
                break; // done processing case

             case 4:
                value = number4;
                break; // done processing case

             case 5:
                value = number5;
                break; // done processing case
          }

          for ( int j = 1; j <= value; j++ )
             System.out.print( "*" );

          System.out.println();
       } // end for loop
    } // end method drawHistograms
} // end class Graphs
```

```
// Exercise 5.16 Solution: GraphsTest.java
// Test application for class Graphs
public class GraphsTest
{
   public static void main( String args[] )
   {
      Graphs application = new Graphs();
      application.drawHistograms();
   } // end main
} // end class GraphsTest
```

```
Enter number: 20
Enter number: 6
Enter number: 15
Enter number: 28
Enter number: 5
********************
******
***************
****************************
*****
```

5.19 Assume that `i = 1`, `j = 2`, `k = 3` and `m = 2`. What does each of the following statements print?

a) `System.out.println( i == 1 );`

ANS: true.

b) `System.out.println( j == 3 );`

ANS: false.

c) `System.out.println( ( i >= 1 ) && ( j < 4 ) );`

ANS: true.

d) `System.out.println( ( m <= 99 ) & ( k < m ) );`

ANS: false.

e) `System.out.println( ( j >= i ) || ( k == m ) );`

ANS: true.

f) `System.out.println( ( k + m < j ) | ( 3 - j >= k ) );`

ANS: false.

g) `System.out.println( !( k > m ) );`

ANS: false.

```
1  // Exercise 5.19 Solution: Mystery.java
2  // Printing conditional expressions outputs 'true' or 'false'.
3  public class Mystery
4  {
5     public static void main( String args[] )
6     {
7        int i = 1;
8        int j = 2;
9        int k = 3;
10       int m = 2;
11
12       // part a
13       System.out.println( i == 1 );
14
15       // part b
16       System.out.println( j == 3 );
17
18       // part c
19       System.out.println( ( i >= 1 ) && ( j < 4 ) );
20
21       // part d
22       System.out.println( ( m <= 99 ) & ( k < m ) );
23
24       // part e
25       System.out.println( ( j >= i ) || ( k == m ) );
26
27       // part f
28       System.out.println( ( k + m < j ) | ( 3 - j >= k ) );
29
30       // part g
31       System.out.println( !( k > m ) );
32    } // end main
33 } // end class Mystery
```

```
true
false
true
false
true
false
false
```

5.20 Calculate the value of π from the infinite series

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \cdots$$

Print a table that shows the value of π approximated by computing one term of this series, by two terms, by three terms, and so on. How many terms of this series do you have to use before you first get 3.14? 3.141? 3.1415? 3.14159?

ANS: [*Note:* The program starts to converge around 3.14 at 119, but does not really approach 3.141 until 1688, well beyond the program's range.]

```
1  // Exercise 5.20 Solution: Pi.java
2  // Program calculates Pi from the infinite series.
3
4  public class Pi
5  {
6     public static void main( String args[] )
7     {
8        double piValue = 0; // current approximation of pi
9        double number = 4.0; // numerator of fraction
10       double denominator = 1.0; // denominator
11       int accuracy = 400; // maximum number of terms in series
12
13       System.out.printf( "Accuracy: %d\n\n", accuracy );
14       System.out.println( "Term\t\tPi" );
15
16       for ( int term = 1; term <= accuracy; term++ )
17       {
18          if ( term % 2 != 0 )
19             piValue += number / denominator;
20          else
21             piValue -= number / denominator;
22
23          System.out.printf( "%d\t\t%.16f\n", term, piValue);
24          denominator += 2.0;
25       } // end for loop
26    } // end main
27 } // end class Pi
```

```
Accuracy: 400

Term            Pi
1               4.0000000000000000
2               2.6666666666666670
3               3.4666666666666670
4               2.8952380952380956
5               3.3396825396825403

.
.
.

395             3.1441242951029738
396             3.1390674050903313
397             3.1441115412820086
398             3.1390800947411280
399             3.1440989153182923
400             3.1390926574960143
```

5.23 (*De Morgan's Laws*) In this chapter, we have discussed the logical operators &&, &, ||, |, ^ and !. De Morgan's Laws can sometimes make it more convenient for us to express a logical expression. These laws state that the expression !(*condition1* && *condition2*) is logically equivalent to the expression (!*condition1* || !*condition2*). Also, the expression !(*condition1* || *condition2*) is logically equivalent to the expression (!*condition1* && !*condition2*). Use De Morgan's Laws to write equivalent expressions for each of the following, then write an application to show that both the original expression and the new expression in each case produce the same value:

a) `!( x < 5 ) && !( y >= 7 )`
b) `!( a == b ) || !( g != 5 )`
c) `!( ( x <= 8 ) && ( y > 4 ) )`
d) `!( ( i > 4 ) || ( j <= 6 ) )`

ANS:

```
1  // Exercise 5.23 Solution: DeMorgan.java
2  // Program tests DeMorgan's laws.
3  public class DeMorgan
4  {
5     public static void main( String args[] )
6     {
7        int x = 6;
8        int y = 0;
9
10       // part a
11       if ( !( x < 5 ) && !( y >= 7 ) )
12          System.out.println( "!( x < 5 ) && !( y >= 7 )" );
13
14       if ( !( ( x < 5 ) || ( y >= 7 ) ) )
15          System.out.println( "!( ( x < 5 ) || ( y >= 7 )" );
16
17       int a = 8;
18       int b = 22;
19       int g = 88;
```

```
20
21         // part b
22         if ( !( a == b ) || !( g != 5 ) )
23            System.out.println( "!( a == b ) || !( g != 5 )" );
24
25         if ( !( ( a == b ) && ( g != 5 ) ) )
26            System.out.println( "!( ( a == b ) && ( g != 5 ) )" );
27
28         x = 8;
29         y = 2;
30
31         // part c
32         if ( !( ( x <= 8 ) && ( y > 4 ) ) )
33            System.out.println( "!( ( x <= 8 ) && ( y > 4 ) )" );
34
35         if ( !( x <= 8 ) || !( y > 4 ) )
36            System.out.println( "!( x <= 8 ) || !( y > 4 )" );
37
38         int i = 0;
39         int j = 7;
40
41         // part d
42         if ( !( ( i > 4 ) || ( j <= 6 ) ) )
43            System.out.println( "!( ( i > 4 ) || ( j <= 6 ) )" );
44
45         if ( !( i > 4 ) && !( j <= 6 ) )
46            System.out.println( "!( i > 4 ) && !( j <= 6 )" );
47      } // end main
48   } // end class DeMorgan
```

```
!( x < 5 ) && !( y >= 7 )
!( ( x < 5 ) || ( y >= 7 ) )
!( a == b ) || !( g != 5 )
!( ( a == b ) && ( g != 5 ) )
!( ( x <= 8 ) && ( y > 4 ) )
!( x <= 8 ) || !( y > 4 )
!( ( i > 4 ) || ( j <= 6 ) )
!( i > 4 ) && !( j <= 6 )
```

5.27 What does the following program segment do?

```
for ( i = 1; i <= 5; i++ )
{
   for ( j = 1; j <= 3; j++ )
   {
      for ( k = 1; k <= 4; k++ )
         System.out.print( '*' );

      System.out.println();
   } // end inner for
```

```
      System.out.println();
   } // end outer for
```

ANS:

```
1  // Exercise 5.27 Solution: Mystery.java
2  // Prints 5 groups of 3 lines, each containing 4 asterisks.
3  public class Mystery
4  {
5     public static void main( String args[] )
6     {
7        int i;
8        int j;
9        int k;
10
11       for ( i = 1; i <= 5; i++ )
12       {
13          for ( j = 1; j <= 3; j++ )
14          {
15             for ( k = 1; k <= 4; k++ )
16                System.out.print( '*' );
17
18             System.out.println();
19          } // end inner for
20
21          System.out.println();
22       } // end outer for
23    } // end main
24 } // end class Mystery
```

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

****
****
****

****
****
****

****
****
****

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

5.29 *("The Twelve Days of Christmas" Song)* Write an application that uses repetition and `switch` statements to print the song "The Twelve Days of Christmas." One `switch` statement should be used to print the day ("first," "second," and so on). A separate `switch` statement should be used to print the remainder of each verse. Visit the website `en.wikipedia.org/wiki/Twelvetide` for the lyrics of the song.

ANS:

```
1  // Exercise 5.29 Solution: Twelve.java
2  // Program prints the 12 days of Christmas song.
3  public class Twelve
4  {
5     // print the 12 days of Christmas song
6     public void printSong()
7     {
8        for ( int day = 1; day <= 12; day++ )
9        {
10          System.out.print( "On the " );
11
12          // add correct day to String
13          switch ( day )
14          {
15             case 1:
16                System.out.print( "first" );
17                break;
18
19             case 2:
20                System.out.print( "second" );
21                break;
22
23             case 3:
24                System.out.print( "third" );
25                break;
26
27             case 4:
28                System.out.print( "fourth" );
29                break;
30
31             case 5:
32                System.out.print( "fifth" );
33                break;
34
35             case 6:
36                System.out.print( "sixth" );
37                break;
38
39             case 7:
40                System.out.print( "seventh" );
41                break;
42
43             case 8:
44                System.out.print( "eighth" );
45                break;
46
```

```
47                case 9:
48                   System.out.print( "ninth" );
49                   break;
50
51                case 10:
52                   System.out.print( "tenth" );
53                   break;
54
55                case 11:
56                   System.out.print( "eleventh" );
57                   break;
58
59                case 12:
60                   System.out.print( "twelfth" );
61                   break;
62             } // end switch
63
64             System.out.println(
65                " day of Christmas, my true love gave to me:" );
66
67             // add remainder of verse to String
68             switch ( day )
69             {
70                case 12:
71                   System.out.println( "Twelve lords-a-leaping," );
72
73                case 11:
74                   System.out.println( "Eleven pipers piping," );
75
76                case 10:
77                   System.out.println( "Ten drummers drumming," );
78
79                case 9:
80                   System.out.println( "Nine ladies dancing," );
81
82                case 8:
83                   System.out.println( "Eight maids-a-milking," );
84
85                case 7:
86                   System.out.println( "Seven swans-a-swimming," );
87
88                case 6:
89                   System.out.println( "Six geese-a-laying," );
90
91                case 5:
92                   System.out.println( "Five golden rings." );
93
94                case 4:
95                   System.out.println( "Four calling birds," );
96
97                case 3:
98                   System.out.println( "Three French hens," );
99
100               case 2:
```

```
                System.out.println( "Two turtle doves, and" );

            case 1:
               System.out.println( "a Partridge in a pear tree." );
            } // end switch

            System.out.println();
         } // end for
      } // end method printSong
   } // end class Twelve
```

```
// Exercise 5.29 Solution: TwelveTest.java
// Test application for class Twelve
public class TwelveTest
{
   public static void main( String args[] )
   {
      Twelve application = new Twelve();
      application.printSong();
   } // end main
} // end class TwelveTest
```

```
On the first day of Christmas, my true love gave to me:
a Partridge in a pear tree.

On the second day of Christmas, my true love gave to me:
Two turtle doves, and
a Partridge in a pear tree.

On the third day of Christmas, my true love gave to me:
Three French hens,
Two turtle doves, and
a Partridge in a pear tree.

On the fourth day of Christmas, my true love gave to me:
Four calling birds,
Three French hens,
Two turtle doves, and
a Partridge in a pear tree.

On the fifth day of Christmas, my true love gave to me:
Five golden rings.
Four calling birds,
Three French hens,
Two turtle doves, and
a Partridge in a pear tree.

On the sixth day of Christmas, my true love gave to me:
Six geese-a-laying,
Five golden rings.
Four calling birds,
Three French hens,
Two turtle doves, and
a Partridge in a pear tree.
```

```
On the seventh day of Christmas, my true love gave to me:
Seven swans-a-swimming,
Six geese-a-laying,
Five golden rings.
Four calling birds,
Three French hens,
Two turtle doves, and
a Partridge in a pear tree.

On the eighth day of Christmas, my true love gave to me:
Eight maids-a-milking,
Seven swans-a-swimming,
Six geese-a-laying,
Five golden rings.
Four calling birds,
Three French hens,
Two turtle doves, and
a Partridge in a pear tree.

On the ninth day of Christmas, my true love gave to me:
Nine ladies dancing,
Eight maids-a-milking,
Seven swans-a-swimming,
Six geese-a-laying,
Five golden rings.
Four calling birds,
Three French hens,
Two turtle doves, and
a Partridge in a pear tree.

On the tenth day of Christmas, my true love gave to me:
Ten drummers drumming,
Nine ladies dancing,
Eight maids-a-milking,
Seven swans-a-swimming,
Six geese-a-laying,
Five golden rings.
Four calling birds,
Three French hens,
Two turtle doves, and
a Partridge in a pear tree.

On the eleventh day of Christmas, my true love gave to me:
Eleven pipers piping,
Ten drummers drumming,
Nine ladies dancing,
Eight maids-a-milking,
Seven swans-a-swimming,
Six geese-a-laying,
Five golden rings.
Four calling birds,
Three French hens,
Two turtle doves, and
a Partridge in a pear tree.
```

```
On the twelfth day of Christmas, my true love gave to me:
Twelve lords-a-leaping,
Eleven pipers piping,
Ten drummers drumming,
Nine ladies dancing,
Eight maids-a-milking,
Seven swans-a-swimming,
Six geese-a-laying,
Five golden rings.
Four calling birds,
Three French hens,
Two turtle doves, and
a Partridge in a pear tree.
```