Control Statements: Part I, Solutions

4

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

Objectives

In this chapter you'll learn:

- Basic problem-solving techniques.
- To develop algorithms through the process of top-down, stepwise refinement.
- 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 portability of primitive data types.

Self-Review Exercises

4.1 Fill in the blanks in each of the following statements:

a) All programs can be written in terms of three types of control structures: ________, ________ and ________.

ANS: sequence, selection, repetition.

b) The ________ statement is used to execute one action when a condition is true and another when that condition is false.

ANS: `if...else`.

c) Repeating a set of instructions a specific number of times is called ________ repetition.

ANS: counter-controlled (or definite).

d) When it's not known in advance how many times a set of statements will be repeated, a(n) ________ value can be used to terminate the repetition.

ANS: sentinel, signal, flag or dummy.

e) The ________ structure is built into Java—by default, statements execute in the order they appear.

ANS: sequence.

f) Instance variables of types `char`, `byte`, `short`, `int`, `long`, `float` and `double` are all given the value ________ by default.

ANS: `0` (zero).

g) Java is a(n) ________ language—it requires all variables to have a type.

ANS: strongly typed.

h) If the increment operator is ________ to a variable, first the variable is incremented by 1, then its new value is used in the expression.

ANS: prefixed.

4.2 State whether each of the following is *true* or *false*. If *false*, explain why.

a) An algorithm is a procedure for solving a problem in terms of the actions to execute and the order in which they execute.

ANS: True.

b) A set of statements contained within a pair of parentheses is called a block.

ANS: False. A set of statements contained within a pair of braces (`{` and `}`) is called a block.

c) A selection statement specifies that an action is to be repeated while some condition remains true.

ANS: False. A repetition statement specifies that an action is to be repeated while some condition remains true.

d) A nested control statement appears in the body of another control statement.

ANS: True.

e) Java provides the arithmetic compound assignment operators `+=`, `-=`, `*=`, `/=` and `%=` for abbreviating assignment expressions.

ANS: True.

f) The primitive types (`boolean`, `char`, `byte`, `short`, `int`, `long`, `float` and `double`) are portable across only Windows platforms.

ANS: False. The primitive types (`boolean`, `char`, `byte`, `short`, `int`, `long`, `float` and `double`) are portable across all computer platforms that support Java.

g) Specifying the order in which statements (actions) execute in a program is called program control.

ANS: True.

h) The unary cast operator `(double)` creates a temporary integer copy of its operand.

ANS: False. The unary cast operator `(double)` creates a temporary floating-point copy of its operand.

i) Instance variables of type `boolean` are given the value `true` by default.

ANS: False. Instance variables of type `boolean` are given the value `false` by default.

j) Pseudocode helps a programmer think out a program before attempting to write it in a programming language.

ANS: True.

4.3 Write four different Java statements that each add 1 to integer variable `x`.

ANS:
```
x = x + 1;
x += 1;
++x;
x++;
```

4.4 Write Java statements to accomplish each of the following tasks:

a) Use one statement to assign the sum of `x` and `y` to `z`, then increment `x` by 1.

ANS: `z = x++ + y;`

b) Test whether variable `count` is greater than 10. If it is, print `"Count is greater than 10"`.

ANS:
```
if ( count > 10 )
   System.out.println( "Count is greater than 10" );
```

c) Use one statement to decrement the variable `x` by 1, then subtract it from variable `total` and store the result in variable `total`.

ANS: `total -= --x;`

d) Calculate the remainder after `q` is divided by `divisor`, and assign the result to `q`. Write this statement in two different ways.

ANS:
```
q %= divisor;
q = q % divisor;
```

4.5 Write a Java statement to accomplish each of the following tasks:

a) Declare variables `sum` and `x` to be of type `int`.

ANS:
```
int sum;
int x;
```

b) Assign `1` to variable `x`.

ANS: `x = 1;`

c) Assign `0` to variable `sum`.

ANS: `sum = 0;`

d) Add variable `x` to variable `sum`, and assign the result to variable `sum`.

ANS: `sum += x;` or `sum = sum + x;`

e) Print `"The sum is: "`, followed by the value of variable `sum`.

ANS: `System.out.printf( "The sum is: %d\n", sum );`

4.6 Combine the statements that you wrote in Exercise 4.5 into a Java application that calculates and prints the sum of the integers from 1 to 10. Use a `while` statement to loop through the calculation and increment statements. The loop should terminate when the value of `x` becomes 11.

ANS: The program is as follows:

```
// Exercise 4.6: Calculate.java
// Calculate the sum of the integers from 1 to 10
public class Calculate
{
   public static void main( String[] args )
   {
      int sum;
      int x;

      x = 1;   // initialize x to 1 for counting
      sum = 0; // initialize sum to 0 for totaling

```

```
        while ( x <= 10 ) // while x is less than or equal to 10
        {
           sum += x; // add x to sum
           ++x; // increment x
        } // end while

        System.out.printf( "The sum is: %d\n", sum );
     } // end main
  } // end class Calculate
```

```
The sum is: 55
```

4.7 Determine the value of the variables in the statement `product *= x++;` after the calculation is performed. Assume that all variables are type `int` and initially have the value 5.

ANS: `product = 25, x = 6`

4.8 Identify and correct the errors in each of the following sets of code:

a)
```
while ( c <= 5 )
{
   product *= c;
   ++c;
```

ANS: Error: The closing right brace of the `while` statement's body is missing.
Correction: Add a closing right brace after the statement `++c;`.

b)
```
if ( gender == 1 )
   System.out.println( "Woman" );
else;
   System.out.println( "Man" );
```

ANS: Error: The semicolon after `else` results in a logic error. The second output statement will always be executed.
Correction: Remove the semicolon after `else`.

4.9 What is wrong with the following `while` statement?

```
while ( z >= 0 )
   sum += z;
```

ANS: The value of the variable `z` is never changed in the `while` statement. Therefore, if the loop-continuation condition `( z >= 0 )` is true, an infinite loop is created. To prevent an infinite loop from occurring, `z` must be decremented so that it eventually becomes less than 0.

Exercises

NOTE: Solutions to the programming exercises are located in the ***ch04solutions*** *folder. Each exercise has its own folder named* ***ex04_##*** *where ## is a two-digit number representing the exercise number. For example, exercise 4.17's solution is located in the folder* ***ex04_17***.

4.10 Compare and contrast the `if` single-selection statement and the `while` repetition statement. How are these two statements similar? How are they different?

ANS: The `if` single-selection statement and the `while` repetition statement both perform an action (or set of actions) based on whether a condition is true or false. However, if the condition is true, the `if` single-selection statement performs the action(s) once, whereas the `while` repetition statement repeatedly performs the action(s) until the condition becomes false.

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.

4.12 Describe the two ways in which control statements can be combined.

ANS: Control statements can be "attached" (that is, stacked) to one another by connecting the exit point of one to the entry point of the next. Control statements also may be nested by placing one inside another.

4.13 What type of repetition would be appropriate for calculating the sum of the first 100 positive integers? What type 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.

4.14 What is the difference between preincrementing and postincrementing a variable?

ANS: Preincrementing a variable causes it to be incremented by 1, and then the new value of the variable is used in the expression in which it appears. Postincrementing a variable causes the current value of the variable to be used in the expression in which it appears, and then the variable's value is incremented by 1. Preincrementing and postincrementing a variable have the same effect when the incrementing operation appears in a statement by itself.

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 is 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 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 if `y` starts with a value greater than 0. The closing brace for the `while` loop is missing.

4.16 What does the following program print?

```java
// Exercise 4.16: Mystery.java
public class Mystery
{
   public static void main( String[] args )
   {
      int y;
      int x = 1;
      int total = 0;

      while ( x <= 10 )
      {
         y = x * x;
         System.out.println( y );
         total += y;
         ++x;
      } // end while

      System.out.printf( "Total is %d\n", total );
   } // end main
} // end class Mystery
```

ANS:

```
1
4
9
16
25
36
49
64
81
100
Total is 385
```

4.25 What does the following program print?

```java
// Exercise 4.25: Mystery2.java
public class Mystery2
{
   public static void main( String[] args )
   {
      int count = 1;

      while ( count <= 10 )
      {
         System.out.println( count % 2 == 1 ? "****" : "++++++++" );
         ++count;
      } // end while
   } // end main
} // end class Mystery2
```

ANS:

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

4.26 What does the following program print?

```java
// Exercise 4.26: Mystery3.java
public class Mystery3
{
   public static void main( String[] args )
   {
      int row = 10;
      int column;

      while ( row >= 1 )
      {
         column = 1;

         while ( column <= 10 )
         {
            System.out.print( row % 2 == 1 ? "<" : ">" );
            ++column;
         } // end while

         --row;
         System.out.println();
      } // end while
   } // end main
} // 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` a particular `else` matches—this situation is referred to as the "dangling-`else` problem." We've eliminated the indentation from the following code to make the problem more challenging. [*Hint:* Apply the indentation conventions you've learned.]

a)
```java
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
$$$$$
```

b)
```java
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
#####
$$$$$
```

4.28 *(Another Dangling-`else` Problem)* Modify the given code to produce the output shown in each part of the problem. Use proper indentation techniques. Make no changes other than inserting braces and changing the indentation of the code. The compiler ignores indentation in a Java program. We've eliminated the indentation from the given code to make the problem more challenging. [*Note:* It's possible that no modification is necessary for some of the parts.]

```
if ( y == 8 )
if ( x == 5 )
System.out.println( "@@@@@" );
else
System.out.println( "#####" );
System.out.println( "$$$$$" );
System.out.println( "&&&&&" );
```

a) Assuming that `x = 5` and `y = 8`, the following output is produced:

```
@@@@@
$$$$$
&&&&&
```

ANS:

```
if ( y == 8 )
   if ( x == 5 )
      System.out.println( "@@@@@" );
   else
      System.out.println( "#####" );
System.out.println( "$$$$$" );
System.out.println( "&&&&&" );
```

b) Assuming that `x = 5` and `y = 8`, the following output is produced:

```
@@@@@
```

ANS:

```
if ( y == 8 )
   if ( x == 5 )
      System.out.println( "@@@@@" );
   else
   {
      System.out.println( "#####" );
      System.out.println( "$$$$$" );
      System.out.println( "&&&&&" );
   }
```

c) Assuming that `x = 5` and `y = 8`, the following output is produced:

```
@@@@@
&&&&&
```

ANS:

```
if ( y == 8 )
   if ( x == 5 )
      System.out.println( "@@@@@" );
   else
   {
      System.out.println( "#####" );
      System.out.println( "$$$$$" );
   }
System.out.println( "&&&&&" );
```

d) Assuming that `x = 5` and `y = 7`, the following output is produced. [*Note:* The last three output statements after the `else` are all part of a block.]

```
#####
$$$$$
&&&&&
```

ANS:

```
if ( y == 8 )
   if ( x == 5 )
      System.out.println( "@@@@@" );
else
{
   System.out.println( "#####" );
   System.out.println( "$$$$$" );
   System.out.println( "&&&&&" );
}
```

4.33 ***(Multiples of 2 with an Infinite Loop)*** Write an application that keeps displaying in the command window the multiples of the integer 2—namely, 2, 4, 8, 16, 32, 64, and so on. Your loop should not terminate (i.e., it should create an infinite loop). What happens when you run this program?

ANS: The program quickly produces values that are outside the range that can be stored in an int variable, then continues looping, but prints 0s. [The source code for this exercises is located in the `ch04solutions\ex04_33` folder.]

4.34 What is wrong with the following statement? Provide the correct statement to add one to the sum of `x` and `y`.

```
System.out.println( ++(x + y) );
```

ANS: ++ can be applied only to a variable—not to an expression with multiple terms. The correct statement is `System.out.println( x + y + 1 );` cd.