18

Recursion: Solutions

We must learn to explore all the options and possibilities that confront us in a complex and rapidly changing world.
—James William Fulbright

O! thou hast damnable iteration, and art indeed able to corrupt a saint.
—William Shakespeare

It's a poor sort of memory that only works backwards.
—Lewis Carroll

Life can only be understood backwards; but it must be lived forwards.
—Soren Kierkegaard

Objectives

In this chapter you'll learn:

- The concept of recursion.
- How to write and use recursive methods.
- How to determine the base case and recursion step in a recursive algorithm.
- How recursive method calls are handled by the system.
- The differences between recursion and iteration, and when to use each.
- What the geometric shapes called fractals are and how to draw them using recursion.
- What recursive backtracking is and why it's an effective problem-solving technique.

Self-Review Exercises

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

a) A method that calls itself indirectly is not an example of recursion.

ANS: False. A method that calls itself in this manner is an example of indirect recursion.

b) Recursion can be efficient in computation because of reduced memory-space usage.

ANS: False. Recursion can be inefficient in computation because of multiple method calls and memory-space usage.

c) When a recursive method is called to solve a problem, it actually is capable of solving only the simplest case(s), or base case(s).

ANS: True.

d) To make recursion feasible, the recursion step in a recursive solution must resemble the original problem, but be a slightly larger version of it.

ANS: False. To make recursion feasible, the recursion step in a recursive solution must resemble the original problem, but be a slightly *smaller* version of it.

18.2 A ________ is needed to terminate recursion.

a) recursion step
b) `break` statement
c) `void` return type
d) base case

ANS: d) base case

18.3 The first call to invoke a recursive method is ________.

a) not recursive
b) recursive
c) the recursion step
d) none of the above

ANS: a) not recursive

18.4 Each time a fractal's pattern is applied, the fractal is said to be at a new ________.

a) width
b) height
c) level
d) volume

ANS: c) level

18.5 Iteration and recursion each involve a ________.

a) repetition statement
b) termination test
c) counter variable
d) none of the above

ANS: b) termination test

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

a) The ratio of successive Fibonacci numbers converges on a constant value of 1.618…, a number that has been called the ________ or the ________.

ANS: golden ratio, golden mean.

b) Iteration normally uses a repetition statement, whereas recursion normally uses a(n) ________ statement.

ANS: selection.

c) Fractals have a(n) ________ property—when subdivided into parts, each is a reduced-size copy of the whole.

ANS: self-similar.

Exercises

NOTE: Solutions to the programming exercises are located in the **ch18solutions** *folder. Each exercise has its own folder named* **ex18_##** *where ## is a two-digit number representing the exercise number. For example, exercise 18.17's solution is located in the folder* **ex18_17***.*

18.7 What does the following code do?

```java
public int mystery( int a, int b )
{
   if ( b == 1 )
      return a;
   else
      return a + mystery( a, b - 1 );
} // end method mystery
```

ANS: The method adds `a` to itself `b` times, which in essence multiplies the values `a` and `b`, recursively.

18.8 Find the error(s) in the following recursive method, and explain how to correct it (them). This method should find the sum of the values from 0 to `n`.

```java
1  public int sum( int n )
2  {
3     if ( n == 0 )
4        return 0;
5     else
6        return n + sum( n );
7  } // end method sum
```

ANS: The code above will result in infinite recursion, unless the value initially passed to the method is `0` (the base case). There is no code to make the recursive call on line 6 simpler than the previous call. The call on line 6 should decrease `n` by 1.

18.12 What does the following program do?

```java
// Exercise 18.12 Solution: MysteryClass.java
public class MysteryClass
{
   public static int mystery( int[] array2, int size )
   {
      if ( size == 1 )
         return array2[ 0 ];
      else
         return array2[ size - 1 ] + mystery( array2, size - 1 );
   } // end method mystery

   public static void main( String[] args )
   {
      int[] array = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

      int result = mystery( array, array.length );
      System.out.printf( "Result is: %d\n", result );
   } // end method main
} // end class MysteryClass
```

ANS: This code totals the values in an array.

18.13 What does the following program do?

```java
// Exercise 18.13 Solution: SomeClass.java
public class SomeClass
{
   public static String someMethod( int[] array2, int x )
   {
      if ( x < array2.length )
         return String.format(
            "%s%d ", someMethod( array2, x + 1 ), array2[ x ] );
      else
         return "";
   } // end method someMethod

   public static void main( String[] args )
   {
      int[] array = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
      String results = someMethod( array, 0 );
      System.out.println( results );
   } // end main
} // end class SomeClass
```

ANS: This code displays the values in an array backwards.