6 Methods: A Deeper Look, Solutions

E pluribus unum.
(One composed of many.)
—Virgil

O! call back yesterday, bid time return.
—William Shakespeare

Call me Ishmael.
—Herman Melville

Answer me in one word.
—William Shakespeare

There is a point at which methods devour themselves.
—Frantz Fanon

Objectives

In this chapter you'll learn:

- How `static` methods and fields are associated with classea rather than objects.
- How the method call/return mechanism is supported by the method-call stack.
- How packages group related classes.
- How to use random-number generation to implement game-playing applications.
- How the visibility of declarations is limited to specific regions of programs.
- What method overloading is and how to create overloaded methods.

Self-Review Exercises

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

a) A method is invoked with a(n) _____.

ANS: method call.

b) A variable known only within the method in which it's declared is called a(n) _____.

ANS: local variable.

c) The _____ statement in a called method can be used to pass the value of an expression back to the calling method.

ANS: `return`.

d) The keyword _____ indicates that a method does not return a value.

ANS: `void`.

e) Data can be added or removed only from the _____ of a stack.

ANS: top.

f) Stacks are known as _____ data structures—the last item pushed (inserted) on the stack is the first item popped (removed) from the stack.

ANS: last-in, first-out (LIFO).

g) The three ways to return control from a called method to a caller are _____, _____ and _____.

ANS: `return;` or `return` *expression*`;` or encountering the closing right brace of a method.

h) An object of class _____ produces random numbers.

ANS: `Random`.

i) The program-execution stack contains the memory for local variables on each invocation of a method during a program's execution. This data, stored as a portion of the program-execution stack, is known as the _____ or _____ of the method call.

ANS: activation record, stack frame.

j) If there are more method calls than can be stored on the program-execution stack, an error known as a(n) _____ occurs.

ANS: stack overflow.

k) The _____ of a declaration is the portion of a program that can refer to the entity in the declaration by name.

ANS: scope.

l) It's possible to have several methods with the same name that each operate on different types or numbers of arguments. This feature is called method _____.

ANS: method overloading.

m) The program-execution stack is also referred to as the _____ stack.

ANS: method call.

6.2 For the class `Craps` in Fig. 6.9, state the scope of each of the following entities:

ANS: class body.

a) the variable `randomNumbers`.

b) the variable `die1`.

ANS: block that defines method `rollDice`'s body.

c) the method `rollDice`.

ANS: class body.

d) the method `play`.

ANS: class body.

e) the variable `sumOfDice`.

ANS: block that defines method `play`'s body.

6.3 Write an application that tests whether the examples of the `Math` class method calls shown in Fig. 6.2 actually produce the indicated results.

ANS: The following solution demonstrates the `Math` class methods in Fig. 6.2:

```
1  // Exercise 6.3: MathTest.java
2  // Testing the Math class methods.
3
4  public class MathTest
5  {
6     public static void main( String[] args )
7     {
8        System.out.printf( "Math.abs( 23.7 ) = %f\n", Math.abs( 23.7 ) );
9        System.out.printf( "Math.abs( 0.0 ) = %f\n", Math.abs( 0.0 ) );
10       System.out.printf( "Math.abs( -23.7 ) = %f\n", Math.abs( -23.7 ) );
11       System.out.printf( "Math.ceil( 9.2 ) = %f\n", Math.ceil( 9.2 ) );
12       System.out.printf( "Math.ceil( -9.8 ) = %f\n", Math.ceil( -9.8 ) );
13       System.out.printf( "Math.cos( 0.0 ) = %f\n", Math.cos( 0.0 ) );
14       System.out.printf( "Math.exp( 1.0 ) = %f\n", Math.exp( 1.0 ) );
15       System.out.printf( "Math.exp( 2.0 ) = %f\n", Math.exp( 2.0 ) );
16       System.out.printf( "Math.floor( 9.2 ) = %f\n", Math.floor( 9.2 ) );
17       System.out.printf( "Math.floor( -9.8 ) = %f\n",
18          Math.floor( -9.8 ) );
19       System.out.printf( "Math.log( Math.E ) = %f\n",
20          Math.log( Math.E ) );
21       System.out.printf( "Math.log( Math.E * Math.E ) = %f\n",
22          Math.log( Math.E * Math.E ) );
23       System.out.printf( "Math.max( 2.3, 12.7 ) = %f\n",
24          Math.max( 2.3, 12.7 ) );
25       System.out.printf( "Math.max( -2.3, -12.7 ) = %f\n",
26          Math.max( -2.3, -12.7 ) );
27       System.out.printf( "Math.min( 2.3, 12.7 ) = %f\n",
28          Math.min( 2.3, 12.7 ) );
29       System.out.printf( "Math.min( -2.3, -12.7 ) = %f\n",
30          Math.min( -2.3, -12.7 ) );
31       System.out.printf( "Math.pow( 2.0, 7.0 ) = %f\n",
32          Math.pow( 2.0, 7.0 ) );
33       System.out.printf( "Math.pow( 9.0, 0.5 ) = %f\n",
34          Math.pow( 9.0, 0.5 ) );
35       System.out.printf( "Math.sin( 0.0 ) = %f\n", Math.sin( 0.0 ) );
36       System.out.printf( "Math.sqrt( 900.0 ) = %f\n",
37          Math.sqrt( 900.0 ) );
38       System.out.printf( "Math.tan( 0.0 ) = %f\n", Math.tan( 0.0 ) );
39    } // end main
40 } // end class MathTest
```

```
Math.abs( 23.7 ) = 23.700000
Math.abs( 0.0 ) = 0.000000
Math.abs( -23.7 ) = 23.700000
Math.ceil( 9.2 ) = 10.000000
Math.ceil( -9.8 ) = -9.000000
Math.cos( 0.0 ) = 1.000000
Math.exp( 1.0 ) = 2.718282
Math.exp( 2.0 ) = 7.389056
Math.floor( 9.2 ) = 9.000000
Math.floor( -9.8 ) = -10.000000
Math.log( Math.E ) = 1.000000
Math.log( Math.E * Math.E ) = 2.000000
Math.max( 2.3, 12.7 ) = 12.700000
Math.max( -2.3, -12.7 ) = -2.300000
Math.min( 2.3, 12.7 ) = 2.300000
Math.min( -2.3, -12.7 ) = -12.700000
Math.pow( 2.0, 7.0 ) = 128.000000
Math.pow( 9.0, 0.5 ) = 3.000000
Math.sin( 0.0 ) = 0.000000
Math.sqrt( 900.0 ) = 30.000000
Math.tan( 0.0 ) = 0.000000
```

6.4 Give the method header for each of the following methods:

a) Method `hypotenuse`, which takes two double-precision, floating-point arguments `side1` and `side2` and returns a double-precision, floating-point result.

ANS: `double hypotenuse( double side1, double side2 )`

b) Method `smallest`, which takes three integers `x`, `y` and `z` and returns an integer.

ANS: `int smallest( int x, int y, int z )`

c) Method `instructions`, which does not take any arguments and does not return a value. [*Note:* Such methods are commonly used to display instructions to a user.]

ANS: `void instructions()`

d) Method `intToFloat`, which takes an integer argument `number` and returns a floating-point result.

ANS: `float intToFloat( int number )`

6.5 Find the error in each of the following program segments. Explain how to correct the error.

a)
```
void g()
{
   System.out.println( "Inside method g" );
   void h()
   {
      System.out.println( "Inside method h" );
   }
}
```

ANS: Error: Method `h` is declared within method `g`.
Correction: Move the declaration of `h` outside the declaration of `g`.

b)
```
int sum( int x, int y )
{
   int result;
   result = x + y;
}
```

ANS: Error: The method is supposed to return an integer, but does not.
Correction: Delete the variable `result`, and place the statement
```
return x + y;
```
in the method, or add the following statement at the end of the method body:
```
return result;
```

c)
```
void f( float a );
{
   float a;
   System.out.println( a );
}
```

ANS: Error: The semicolon after the right parenthesis of the parameter list is incorrect, and the parameter `a` should not be redeclared in the method.
Correction: Delete the semicolon after the right parenthesis of the parameter list, and delete the declaration `float a;`.

d)
```
void product()
{
   int a = 6, b = 5, c = 4, result;
   result = a * b * c;
   System.out.printf( "Result is %d\n", result );
   return result;
}
```

ANS: Error: The method returns a value when it's not supposed to.
Correction: Change the return type from `void` to `int`.

6.6 Write a complete Java application to prompt the user for the `double` radius of a sphere, and call method `sphereVolume` to calculate and display the volume of the sphere. Use the following statement to calculate the volume:

```
double volume = ( 4.0 / 3.0 ) * Math.PI * Math.pow( radius, 3 )
```

ANS: The following solution calculates the volume of a sphere, using the radius entered by the user:

```
1  // Exercise 6.6: Sphere.java
2  // Calculate the volume of a sphere.
3  import java.util.Scanner;
4
5  public class Sphere
6  {
7     // obtain radius from user and display volume of sphere
8     public void determineSphereVolume()
9     {
10       Scanner input = new Scanner( System.in );
11
12       System.out.print( "Enter radius of sphere: " );
13       double radius = input.nextDouble();
14
15       System.out.printf( "Volume is %f\n", sphereVolume( radius ) );
16    } // end method determineSphereVolume
17
18    // calculate and return sphere volume
19    public double sphereVolume( double radius )
20    {
21       double volume = ( 4.0 / 3.0 ) * Math.PI * Math.pow( radius, 3 );
22       return volume;
23    } // end method sphereVolume
24 } // end class Sphere
```

```
1  // Exercise 6.6: SphereTest.java
2  // Calculate the volume of a sphere.
3
4  public class SphereTest
5  {
6     // application starting point
7     public static void main( String[] args )
8     {
9        Sphere mySphere = new Sphere();
10       mySphere.determineSphereVolume();
11    } // end main
12 } // end class SphereTest
```

```
Enter radius of sphere: 4
Volume is 268.082573
```

Exercises

NOTE: Solutions to the programming exercises are located in the **ch06solutions** *folder. Each exercise has its own folder named* **ex06_##** *where ## is a two-digit number representing the exercise number. For example, exercise 6.8's solution is located in the folder* **ex06_08**.

6.7 What is the value of x after each of the following statements is executed?

a) `x = Math.abs( 7.5 );`

ANS: 7.5

b) `x = Math.floor( 7.5 );`

ANS: 7.0

c) `x = Math.abs( 0.0 );`

ANS: 0.0

d) `x = Math.ceil( 0.0 );`

ANS: 0.0

e) `x = Math.abs( -6.4 );`

ANS: 6.4

f) `x = Math.ceil( -6.4 );`

ANS: -6.0

g) `x = Math.ceil( -Math.abs( -8 + Math.floor( -5.5 ) ) );`

ANS: -14.0

6.11 Answer each of the following questions:

a) What does it mean to choose numbers "at random"?

ANS: Every number has an equal chance of being chosen at any time.

b) Why is the `nextInt` method of class `Random` useful for simulating games of chance?

ANS: Because it produces a series of random numbers.

c) Why is it often necessary to scale or shift the values produced by a `Random` object?

ANS: To produce random numbers in a specific range.

d) Why is computerized simulation of real-world situations a useful technique?

ANS: It enables more accurate predictions of random events, such as cars arriving at toll booths and people arriving in lines at a supermarket. The results of a simulation can help determine how many toll booths to have open or how many cashiers to have open at specified times.

6.12 Write statements that assign random integers to the variable n in the following ranges:

a) $1 \leq n \leq 2$

ANS: `n = 1 + randomNumbers.nextInt( 2 );`

b) $1 \leq n \leq 100$

ANS: `n = 1 + randomNumbers.nextInt( 100 );`

c) $0 \leq n \leq 9$

ANS: `n = randomNumbers.nextInt( 10 );`

d) $1000 \leq n \leq 1112$

ANS: `n = 1000 + randomNumbers.nextInt( 113 );`

e) $-1 \leq n \leq 1$

ANS: `n = -1 + randomNumbers.nextInt( 3 );`

f) $-3 \leq n \leq 11$

ANS: `n = -3 + randomNumbers.nextInt( 15 );`

ANS: [Note: See the test program in the `ch06solutions\ex06_12` folder.]

6.13 For each of the following sets of integers, write a single statement that will display a number at random from the set:

a) 2, 4, 6, 8, 10.
ANS: `System.out.println( 2 + randomNumbers.nextInt( 5 ) * 2 );`
b) 3, 5, 7, 9, 11.
ANS: `System.out.println( 3 + randomNumbers.nextInt( 5 ) * 2 );`
c) 6, 10, 14, 18, 22.
ANS: `System.out.println( 6 + randomNumbers.nextInt( 5 ) * 4 );`
ANS: [Note: See the test program in the `ch06solutions\ex06_12` folder.]