**Section 6.1 Introduction**

• Experience has shown that the best way to develop and maintain a large program is to construct it from small, simple pieces, or modules. This technique is called divide and conquer (p. [201](http://proquest.safaribooksonline.com/9780133813036/ch06_html#page_201)).

#### Section 6.2 Program Modules in Java

• Methods are declared within classes. Classes are typically grouped into packages so they can be imported and reused.

• Methods allow you to modularize a program by separating its tasks into self-contained units. The statements in a method are written only once and hidden from other methods.

• Using existing methods as building blocks to create new programs is a form of software reusability (p. [202](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec2_html#page_202)) that allows you to avoid repeating code within a program.

#### Section 6.3 static Methods, static Fields and ClassMath

• A method call specifies the name of the method to call and provides the arguments that the called method requires to perform its task. When the method call completes, the method returns either a result, or simply control, to its caller.

• A class may contain static methods to perform common tasks that do not require an object of the class. Any data astatic method might require to perform its tasks can be sent to the method as arguments in a method call. A staticmethod is called by specifying the name of the class in which the method is declared followed by a dot (.) and the method name, as in

ClassName.methodName(arguments)

• Class Math provides static methods for performing common mathematical calculations.

• The constant Math.PI (p. [204](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec3_html#page_204); 3.141592653589793) is the ratio of a circle’s circumference to its diameter. The constantMath.E (p. [204](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec3_html#page_204); 2.718281828459045) is the base value for natural logarithms (calculated with static Math methodlog).

• Math.PI and Math.E are declared with the modifiers public,final and static. Making them public allows you to use these fields in your own classes. A field declared with keyword final (p. [205](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec5_html#page_205)) is constant—its value cannot be changed after it’s initialized. Both PI and E are declaredfinal because their values never change. Making these fields static allows them to be accessed via the class nameMath and a dot (.) separator, just like class Math’s methods.

• All the objects of a class share one copy of the class’s staticfields. Together the class variables (p. [204](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec3_html#page_204)) and instance variables represent the fields of a class.

• When you execute the Java Virtual Machine (JVM) with thejava command, the JVM loads the class you specify and uses that class name to invoke method main. You can specify additional command-line arguments (p. [205](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec5_html#page_205)) that the JVM will pass to your application.

• You can place a main method in every class you declare—only the main method in the class you use to execute the application will be called by the java command.

#### Section 6.4 Declaring Methods with Multiple Parameters

• When a method is called, the program makes a copy of the method’s argument values and assigns them to the method’s corresponding parameters. When program control returns to the point in the program where the method was called, the method’s parameters are removed from memory.

• A method can return at most one value, but the returned value could be a reference to an object that contains many values.

• Variables should be declared as fields of a class only if they’re required for use in more than one method of the class or if the program should save their values between calls to the class’s methods.

• When a method has more than one parameter, the parameters are specified as a comma-separated list. There must be one argument in the method call for each parameter in the method declaration. Also, each argument must be consistent with the type of the corresponding parameter. If a method does not accept arguments, the parameter list is empty.

• Strings can be concatenated (p. [208](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec10_html#page_208)) using operator +, which places the characters of the right operand at the end of those in the left operand.

• Every primitive value and object in Java can be represented as a String. When an object is concatenated with a String, the object is converted to a String, then the two Strings are concatenated.

• If a boolean is concatenated with a String, the word"true" or the word "false" is used to represent theboolean value.

• All objects in Java have a special method named toStringthat returns a String representation of the object’s contents. When an object is concatenated with a String, the JVM implicitly calls the object’s toString method to obtain the string representation of the object.

• You can break large String literals into several smallerStrings and place them on multiple lines of code for readability, then reassemble the Strings using concatenation.

#### Section 6.5 Notes on Declaring and Using Methods

• There are three ways to call a method—using a method name by itself to call another method of the same class; using a variable that contains a reference to an object, followed by a dot (.) and the method name to call a method of the referenced object; and using the class name and a dot (.) to call a staticmethod of a class.

• There are three ways to return control to a statement that calls a method. If the method does not return a result, control returns when the program flow reaches the method-ending right brace or when the statement

return;

is executed. If the method returns a result, the statement

return expression;

evaluates the expression, then immediately returns the resulting value to the caller.

#### Section 6.6 Method-Call Stack and Stack Frames

• Stacks (p. [209](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec5_html#page_209)) are known as last-in, first-out (LIFO) data structures—the last item pushed (inserted) onto the stack is the first item popped (removed) from the stack.

• A called method must know how to return to its caller, so the return address of the calling method is pushed onto the method-call stack when the method is called. If a series of method calls occurs, the successive return addresses are pushed onto the stack in last-in, first-out order so that the last method to execute will be the first to return to its caller.

• The method-call stack (p. [210](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec6_html#page_210)) contains the memory for the local variables used in each invocation of a method during a program’s execution. This data is known as the method call’s stack frame or activation record. When a method call is made, the stack frame for that method call is pushed onto the method-call stack. When the method returns to its caller, its stack frame call is popped off the stack and the local variables are no longer known to the program.

• If there are more method calls than can have their stack frames stored on the method-call stack, an error known as a stack overflow (p. [210](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec6_html#page_210)) occurs. The application will compile correctly, but its execution causes a stack overflow.

#### Section 6.7 Argument Promotion and Casting

• Argument promotion (p. [210](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec6_html#page_210)) converts an argument’s value to the type that the method expects to receive in its corresponding parameter.

• Promotion rules (p. [210](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec6_html#page_210)) apply to expressions containing values of two or more primitive types and to primitive-type values passed as arguments to methods. Each value is promoted to the “highest” type in the expression. In cases where information may be lost due to conversion, the Java compiler requires you to use a cast operator to explicitly force the conversion to occur.

#### Section 6.9 Case Study: Secure Random-Number Generation

• Objects of class SecureRandom (package java.security; p.[213](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec8_html#page_213)) can produce nondeterministic random values.

• SecureRandom method nextInt (p. [214](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec11_html#page_214)) generates a random int value.

• Class SecureRandom provides another version of methodnextInt that receives an int argument and returns a value from 0 up to, but not including, the argument’s value.

• Random numbers in a range (p. [214](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec11_html#page_214)) can be generated with

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0233pro01a)

int number = shiftingValue +randomNumbers.nextInt(scalingFactor);

where shiftingValue specifies the first number in the desired range of consecutive integers, and scalingFactorspecifies how many numbers are in the range.

• Random numbers can be chosen from nonconsecutive integer ranges, as in

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0233pro02a)

int number = shiftingValue +  
   differenceBetweenValues \*randomNumbers.nextInt(scalingFactor);

where shiftingValue specifies the first number in the range of values, differenceBetweenValues represents the difference between consecutive numbers in the sequence and scalingFactor specifies how many numbers are in the range.

#### Section 6.10 Case Study: A Game of Chance; Introducing enum Types

• An enum type (p. [221](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec20_html#page_221)) is introduced by the keyword enum and a type name. As with any class, braces ({ and }) delimit the body of an enum declaration. Inside the braces is a comma-separated list of enum constants, each representing a unique value. The identifiers in an enum must be unique. Variables of an enum type can be assigned only constants of that enum type.

• Constants can also be declared as private static finalvariables. Such constants are declared by convention with all capital letters to make them stand out in the program.

#### Section 6.11 Scope of Declarations

• Scope (p. [222](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec23_html#page_222)) is the portion of the program in which an entity, such as a variable or a method, can be referred to by its name. Such an entity is said to be “in scope” for that portion of the program.

• The scope of a parameter declaration is the body of the method in which the declaration appears.

• The scope of a local-variable declaration is from the point at which the declaration appears to the end of that block.

• The scope of a local-variable declaration that appears in the initialization section of a for statement’s header is the body of the for statement and the other expressions in the header.

• The scope of a method or field of a class is the entire body of the class. This enables a class’s methods to use simple names to call the class’s other methods and to access the class’s fields.

• Any block may contain variable declarations. If a local variable or parameter in a method has the same name as a field, the field is shadowed (p. [223](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec11_html#page_223)) until the block terminates execution.

#### Section 6.12 Method Overloading

• Java allows overloaded methods (p. [225](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec11_html#page_225)) in a class, as long as the methods have different sets of parameters (determined by the number, order and types of the parameters).

• Overloaded methods are distinguished by their signatures (p.[226](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec26_html#page_226))—combinations of the methods’ names and the number, types and order of their parameters, but not their return types.

### Self-Review Exercises

[**6.1**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec17_html#ch06ans1) Fill in the blanks in each of the following statements:

a) A method is invoked with a(n) \_\_\_\_\_\_\_\_ .

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

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

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

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

f) Stacks are known as \_\_\_\_\_\_\_\_ data structures; the last item pushed (inserted) onto the stack is the first item popped (removed) from the stack.

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

h) An object of class \_\_\_\_\_\_\_\_ produces truly random numbers.

i) The method-call 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 method-call stack, is known as the \_\_\_\_\_\_\_\_ or \_\_\_\_\_\_\_\_ of the method call.

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

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

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 \_\_\_\_\_\_\_\_ .

[**6.2**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec17_html#ch06ans2) For the class Craps in [Fig. 6.8](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec10_html#ch06fig08), state the scope of each of the following entities:

a) the variable randomNumbers.

b) the variable die1.

c) the method rollDice.

d) the method main.

e) the variable sumOfDice.

[**6.3**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec17_html#ch06ans3) Write an application that tests whether the examples of theMath class method calls shown in [Fig. 6.2](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec3_html#ch06fig02) actually produce the indicated results.

[**6.4**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec17_html#ch06ans4) 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.

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

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.]

d) Method intToFloat, which takes integer argument numberand returns a float.

[**6.5**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec17_html#ch06ans5) Find the error in each of the following program segments. Explain how to correct the error.

a)

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0235pro01a)

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

b)

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

c)

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

d)

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0235pro02a)

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

[**6.6**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec17_html#ch06ans6) Declare method sphereVolume to calculate and return the volume of the sphere. Use the following statement to calculate the volume:

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0235pro03a)

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

Write a Java application that prompts the user for the doubleradius of a sphere, calls sphereVolume to calculate the volume and displays the result.

### Answers to Self-Review Exercises

[**6.1**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec16_html#ch06que1)

a) method call.

b) local variable.

c) return.

d) void.

e) top.

f) last-in, first-out (LIFO).

g) return; or return expression; or encountering the closing right brace of a method.

h) SecureRandom.

i) stack frame, activation record.

j) stack overflow.

k) scope.

l) method overloading.

[**6.2**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec16_html#ch06que2)

a) class body.

b) block that defines method rollDice’s body.

c) class body.

d) class body.

e) block that defines method main’s body.

[**6.3**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec16_html#ch06que3) The following solution demonstrates the Math class methods in [Fig. 6.2](http://proquest.safaribooksonline.com/9780133813036/ch06lev2sec3_html#ch06fig02):

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0235pro04a)

**1**   // Exercise 6.3: MathTest.java  
 **2**   // Testing the Math class methods.  
 **3**   public class MathTest  
 **4**   {  
 **5**      public static void main(String[] args)  
 **6**      {  
 **7**         System.out.printf("Math.abs(23.7) = %f%n", Math.abs(23.7));  
 **8**         System.out.printf("Math.abs(0.0) = %f%n", Math.abs(0.0));  
 **9**         System.out.printf("Math.abs(-23.7) = %f%n", Math.abs(-23.7));  
**10**         System.out.printf("Math.ceil(9.2) = %f%n", Math.ceil(9.2));  
**11**         System.out.printf("Math.ceil(-9.8) = %f%n", Math.ceil(-9.8));  
**12**         System.out.printf("Math.cos(0.0) = %f%n", Math.cos(0.0));  
**13**         System.out.printf("Math.exp(1.0) = %f%n", Math.exp(1.0));  
**14**         System.out.printf("Math.exp(2.0) = %f%n", Math.exp(2.0));  
**15**         System.out.printf("Math.floor(9.2) = %f%n", Math.floor(9.2));  
**16**         System.out.printf("Math.floor(-9.8) = %f%n", Math.floor(-9.8));  
**17**         System.out.printf("Math.log(Math.E) = %f%n", Math.log(Math.E));  
**18**         System.out.printf("Math.log(Math.E \* Math.E) = %f%n",  
**19**            Math.log(Math.E \* Math.E));  
**20**         System.out.printf("Math.max(2.3, 12.7) = %f%n", Math.max(2.3, 12.7));  
**21**         System.out.printf("Math.max(-2.3, -12.7) = %f%n",  
**22**            Math.max(-2.3, -12.7));  
**23**         System.out.printf("Math.min(2.3, 12.7) = %f%n", Math.min(2.3, 12.7));  
**24**         System.out.printf("Math.min(-2.3, -12.7) = %f%n",  
**25**            Math.min(-2.3, -12.7));  
**26**         System.out.printf("Math.pow(2.0, 7.0) = %f%n", Math.pow(2.0, 7.0));  
**27**         System.out.printf("Math.pow(9.0, 0.5) = %f%n", Math.pow(9.0, 0.5));  
**28**         System.out.printf("Math.sin(0.0) = %f%n", Math.sin(0.0));  
**29**         System.out.printf("Math.sqrt(900.0) = %f%n", Math.sqrt(900.0));  
**30**         System.out.printf("Math.tan(0.0) = %f%n", Math.tan(0.0));  
**31**      } // end main  
**32**   } // 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**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec16_html#ch06que4)

a) double hypotenuse(double side1, double side2)

b) int smallest(int x, int y, int z)

c) void instructions()

d) float intToFloat(int number)

[**6.5**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec16_html#ch06que5)

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

b) 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) 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) Error: The method returns a value when it’s not supposed to.

Correction: Change the return type from void to int.

[**6.6**](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec16_html#ch06que6) The following solution calculates the volume of a sphere, using the radius entered by the user:

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0237pro01a)

**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 static void main(String[] args)   
 **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 static double sphereVolume(doubleradius)   
**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

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

### Exercises

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

a) x = Math.abs(7.5);

b) x = Math.floor(7.5);

c) x = Math.abs(0.0);

d) x = Math.ceil(0.0);

e) x = Math.abs(-6.4);

f) x = Math.ceil(-6.4);

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

**6.8 (Parking Charges)** A parking garage charges a $2.00 minimum fee to park for up to three hours. The garage charges an additional $0.50 per hour for each hour or part thereof in excess of three hours. The maximum charge for any given 24-hour period is $10.00. Assume that no car parks for longer than 24 hours at a time. Write an application that calculates and displays the parking charges for each customer who parked in the garage yesterday. You should enter the hours parked for each customer. The program should display the charge for the current customer and should calculate and display the running total of yesterday’s receipts. It should use the method calculateCharges to determine the charge for each customer.

**6.9 (Rounding Numbers)** Math.floor can be used to round values to the nearest integer—e.g.,

y = Math.floor(x + 0.5);

will round the number x to the nearest integer and assign the result to y. Write an application that reads double values and uses the preceding statement to round each of the numbers to the nearest integer. For each number processed, display both the original number and the rounded number.

**6.10 (Rounding Numbers)** To round numbers to specific decimal places, use a statement like

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0238pro01a)

y = Math.floor(x \* 10 + 0.5) / 10;

which rounds x to the tenths position (i.e., the first position to the right of the decimal point), or

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0238pro02a)

y = Math.floor(x \* 100 + 0.5) / 100;

which rounds x to the hundredths position (i.e., the second position to the right of the decimal point). Write an application that defines four methods for rounding a number x in various ways:

a) roundToInteger(number)

b) roundToTenths(number)

c) roundToHundredths(number)

d) roundToThousandths(number)

For each value read, your program should display the original value, the number rounded to the nearest integer, the number rounded to the nearest tenth, the number rounded to the nearest hundredth and the number rounded to the nearest thousandth.

**6.11** Answer each of the following questions:

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

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

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

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

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

a) 1 ≤n ≤ 2.

b) 1 ≤n ≤ 100.

c) 0 ≤n ≤ 9.

d) 1000 ≤n ≤ 1112.

e) –1 ≤n ≤ 1.

f) –3 ≤n ≤ 11.

**6.13** Write statements that will display a random number from each of the following sets:

a) 2, 4, 6, 8, 10.

b) 3, 5, 7, 9, 11.

c) 6, 10, 14, 18, 22.

**6.14 (Exponentiation)** Write a method integerPower(base, exponent) that returns the value of

baseexponent

For example, integerPower(3, 4) calculates 34 (or 3 \* 3 \* 3 \* 3). Assume that exponent is a positive, nonzero integer and thatbase is an integer. Use a for or while statement to control the calculation. Do not use any Math class methods. Incorporate this method into an application that reads integer values for base andexponent and performs the calculation with the integerPowermethod.

**6.15 (Hypotenuse Calculations)** Define a method hypotenusethat calculates the hypotenuse of a right triangle when the lengths of the other two sides are given. The method should take two arguments of type double and return the hypotenuse as adouble. Incorporate this method into an application that reads values for side1 and side2 and performs the calculation with thehypotenuse method. Use Math methods pow and sqrt to determine the length of the hypotenuse for each of the triangles in [Fig. 6.15](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec18_html#ch06fig15). [Note: Class Math also provides method hypot to perform this calculation.]

**Fig. 6.15** | Values for the sides of triangles in [Exercise 6.15](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec18_html#ch06que15).

**6.16 (Multiples)** Write a method isMultiple that determines, for a pair of integers, whether the second integer is a multiple of the first. The method should take two integer arguments and return true if the second is a multiple of the first and falseotherwise. [Hint: Use the remainder operator.] Incorporate this method into an application that inputs a series of pairs of integers (one pair at a time) and determines whether the second value in each pair is a multiple of the first.

**6.17 (Even or Odd)** Write a method isEven that uses the remainder operator (%) to determine whether an integer is even. The method should take an integer argument and return true if the integer is even and false otherwise. Incorporate this method into an application that inputs a sequence of integers (one at a time) and determines whether each is even or odd.

**6.18****(Displaying a Square of Asterisks)** Write a methodsquareOfAsterisks that displays a solid square (the same number of rows and columns) of asterisks whose side is specified in integer parameter side. For example, if side is 4, the method should display

\*\*\*\*  
\*\*\*\*  
\*\*\*\*  
\*\*\*\*

Incorporate this method into an application that reads an integer value for side from the user and outputs the asterisks with thesquareOfAsterisks method.

**6.19 (Displaying a Square of Any Character)** Modify the method created in [Exercise 6.18](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec18_html#ch06que18) to receive a second parameter of type char called fillCharacter. Form the square using the charprovided as an argument. Thus, if side is 5 and fillCharacter is#, the method should display

#####  
#####  
#####  
#####  
#####

Use the following statement (in which input is a Scanner object) to read a character from the user at the keyboard:

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0239pro01a)

char fill = input.next().charAt(0);

**6.20 (Circle Area)** Write an application that prompts the user for the radius of a circle and uses a method called circleArea to calculate the area of the circle.

**6.21 (Separating Digits)** Write methods that accomplish each of the following tasks:

a) Calculate the integer part of the quotient when integer a is divided by integer b.

b) Calculate the integer remainder when integer a is divided by integer b.

c) Use the methods developed in parts (a) and (b) to write a method displayDigits that receives an integer between 1and 99999 and displays it as a sequence of digits, separating each pair of digits by two spaces. For example, the integer4562 should appear as

4  5  6  2

Incorporate the methods into an application that inputs an integer and calls displayDigits by passing the method the integer entered. Display the results.

**6.22 (Temperature Conversions)** Implement the following integer methods:

a) Method celsius returns the Celsius equivalent of a Fahrenheit temperature, using the calculation

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0240pro01a)

celsius = 5.0 / 9.0 \* (fahrenheit - 32);

b) Method fahrenheit returns the Fahrenheit equivalent of a Celsius temperature, using the calculation

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p0240pro02a)

fahrenheit = 9.0 / 5.0 \* celsius + 32;

c) Use the methods from parts (a) and (b) to write an application that enables the user either to enter a Fahrenheit temperature and display the Celsius equivalent or to enter a Celsius temperature and display the Fahrenheit equivalent.

**6.23****(Find the Minimum)** Write a method minimum3 that returns the smallest of three floatingpoint numbers. Use theMath.min method to implement minimum3. Incorporate the method into an application that reads three values from the user, determines the smallest value and displays the result.

**6.24 (Perfect Numbers)** An integer number is said to be aperfect number if its factors, including 1 (but not the number itself), sum to the number. For example, 6 is a perfect number, because 6 = 1 + 2 + 3. Write a method isPerfect that determines whether parameter number is a perfect number. Use this method in an application that displays all the perfect numbers between 1 and 1000. Display the factors of each perfect number to confirm that the number is indeed perfect. Challenge the computing power of your computer by testing numbers much larger than 1000. Display the results.

**6.25 (Prime Numbers)** A positive integer is prime if it’s divisible by only 1 and itself. For example, 2, 3, 5 and 7 are prime, but 4, 6, 8 and 9 are not. The number 1, by definition, is not prime.

a) Write a method that determines whether a number is prime.

b) Use this method in an application that determines and displays all the prime numbers less than 10,000. How many numbers up to 10,000 do you have to test to ensure that you’ve found all the primes?

c) Initially, you might think that n/2 is the upper limit for which you must test to see whether a number n is prime, but you need only go as high as the square root of n. Rewrite the program, and run it both ways.

**6.26 (Reversing Digits)** Write a method that takes an integer value and returns the number with its digits reversed. For example, given the number 7631, the method should return 1367. Incorporate the method into an application that reads a value from the user and displays the result.

**6.27 (Greatest Common Divisor)** The greatest common divisor(GCD) of two integers is the largest integer that evenly divides each of the two numbers. Write a method gcd that returns the greatest common divisor of two integers. [Hint: You might want to use Euclid’s algorithm. You can find information about it aten.wikipedia.org/wiki/Euclidean\_algorithm.] Incorporate the method into an application that reads two values from the user and displays the result.

**6.28** Write a method qualityPoints that inputs a student’s average and returns 4 if it’s 90–100, 3 if 80–89, 2 if 70–79, 1 if 60–69 and 0 if lower than 60. Incorporate the method into an application that reads a value from the user and displays the result.

**6.29 (Coin Tossing)** Write an application that simulates coin tossing. Let the program toss a coin each time the user chooses the “Toss Coin” menu option. Count the number of times each side of the coin appears. Display the results. The program should call a separate method flip that takes no arguments and returns a value from a Coin enum (HEADS and TAILS). [Note: If the program realistically simulates coin tossing, each side of the coin should appear approximately half the time.]

**6.30 (Guess the Number)** Write an application that plays “guess the number” as follows: Your program chooses the number to be guessed by selecting a random integer in the range 1 to 1000. The application displays the prompt Guess a number between 1 and 1000. The player inputs a first guess. If the player's guess is incorrect, your program should display Too high. Try again. or Too low. Try again. to help the player “zero in” on the correct answer. The program should prompt the user for the next guess. When the user enters the correct answer, displayCongratulations. You guessed the number!, and allow the user to choose whether to play again. [Note: The guessing technique employed in this problem is similar to a binary search, which is discussed in [Chapter 19](http://proquest.safaribooksonline.com/9780133813036/ch19_html#ch19), [Searching, Sorting and Big O](http://proquest.safaribooksonline.com/9780133813036/ch19_html#ch19).]

**6.31 (Guess the Number Modification)** Modify the program of[Exercise 6.30](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec18_html#ch06que30) to count the number of guesses the player makes. If the number is 10 or fewer, display Either you know the secret or you got lucky! If the player guesses the number in 10 tries, display Aha! You know the secret! If the player makes more than 10 guesses, display You should be able to do better! Why should it take no more than 10 guesses? Well, with each “good guess,” the player should be able to eliminate half of the numbers, then half of the remaining numbers, and so on.

**6.32 (Distance Between Points)** Write method distance to calculate the distance between two points (x1, y1) and (x2, y2). All numbers and return values should be of type double. Incorporate this method into an application that enables the user to enter the coordinates of the points.

**6.33 (Craps Game Modification)** Modify the craps program of[Fig. 6.8](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec10_html#ch06fig08) to allow wagering. Initialize variable bankBalance to 1000 dollars. Prompt the player to enter a wager. Check that wager is less than or equal to bankBalance, and if it’s not, have the user reenter wager until a valid wager is entered. Then, run one game of craps. If the player wins, increase bankBalance by wager and display the new bankBalance. If the player loses, decreasebankBalance by wager, display the new bankBalance, check whether bankBalance has become zero and, if so, display the message "Sorry. You busted!" As the game progresses, display various messages to create some “chatter,” such as "Oh, you're going for broke, huh?" or "Aw c'mon, take a chance!" or"You're up big. Now's the time to cash in your chips!". Implement the “chatter” as a separate method that randomly chooses the string to display.

**6.34****(Table of Binary, Octal and Hexadecimal Numbers)**Write an application that displays a table of the binary, octal and hexadecimal equivalents of the decimal numbers in the range 1 through 256. If you aren’t familiar with these number systems, read online Appendix J first.

### Making a Difference

As computer costs decline, it becomes feasible for every student, regardless of economic circumstance, to have a computer and use it in school. This creates exciting possibilities for improving the educational experience of all students worldwide, as suggested by the next five exercises. [Note: Check out initiatives such as the One Laptop Per Child Project ([www.laptop.org](http://www.laptop.org/)). Also, research “green” laptops—what are some key “going green” characteristics of these devices? Look into the Electronic Product Environmental Assessment Tool (www.epeat.net), which can help you assess the “greenness” of desktops, notebooks and monitors to help you decide which products to purchase.]

**6.35 (Computer-Assisted Instruction)** The use of computers in education is referred to as computer-assisted instruction (CAI). Write a program that will help an elementary school student learnmultiplication. Use a SecureRandom object to produce two positive one-digit integers. The program should then prompt the user with a question, such as

How much is 6 times 7?

The student then inputs the answer. Next, the program checks the student’s answer. If it’s correct, display the message "Very good!" and ask another multiplication question. If the answer is wrong, display the message "No. Please try again." and let the student try the same question repeatedly until the student finally gets it right. A separate method should be used to generate each new question. This method should be called once when the application begins execution and each time the user answers the question correctly.

**6.36 (Computer-Assisted Instruction: Reducing Student Fatigue)** One problem in CAI environments is student fatigue. This can be reduced by varying the computer’s responses to hold the student’s attention. Modify the program of [Exercise 6.35](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec19_html#ch06que35) so that various comments are displayed for each answer as follows:

Possible responses to a correct answer:

Very good!  
Excellent!  
Nice work!  
Keep up the good work!

Possible responses to an incorrect answer:

No. Please try again.  
Wrong. Try once more.  
Don't give up!  
No. Keep trying.

Use random-number generation to choose a number from 1 to 4 that will be used to select one of the four appropriate responses to each correct or incorrect answer. Use a switch statement to issue the responses.

**6.37 (Computer-Assisted Instruction: Monitoring Student Performance)** More sophisticated computer-assisted instruction systems monitor the student’s performance over a period of time. The decision to begin a new topic is often based on the student’s success with previous topics. Modify the program of [Exercise 6.36](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec19_html#ch06que36)to count the number of correct and incorrect responses typed by the student. After the student types 10 answers, your program should calculate the percentage that are correct. If the percentage is lower than 75%, display "Please ask your teacher for extra help.", then reset the program so another student can try it. If the percentage is 75% or higher, display "Congratulations, you are ready to go to the next level!", then reset the program so another student can try it.

**6.38 (Computer-Assisted Instruction: Difficulty Levels)**[Exercises 6.35](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec19_html#ch06que35)–[6.37](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec19_html#ch06que37) developed a computer-assisted instruction program to help teach an elementary school student multiplication. Modify the program to allow the user to enter a difficulty level. At a difficulty level of 1, the program should use only single-digit numbers in the problems; at a difficulty level of 2, numbers as large as two digits, and so on.

**6.39****(Computer-Assisted Instruction: Varying the Types of Problems)** Modify the program of [Exercise 6.38](http://proquest.safaribooksonline.com/9780133813036/ch06lev1sec19_html#ch06que38) to allow the user to pick a type of arithmetic problem to study. An option of 1means addition problems only, 2 means subtraction problems only, 3 means multiplication problems only, 4 means division problems only and 5 means a random mixture of all these types.