6

Methods: A Deeper Look

The greatest invention of the nineteenth century was the invention of the method of invention.
—Alfred North Whitehead

Call me Ishmael.
—Herman Melville

When you call me that, smile!
—Owen Wister

Answer me in one word.
—William Shakespeare

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

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

OBJECTIVES

In this chapter you will learn:

- How `static` methods and fields are associated with an entire class rather than specific instances of the class.
- To use common `Math` methods available in the Java API.
- To understand the mechanisms for passing information between methods.
- How the method call/return mechanism is supported by the method-call stack and activation records.
- 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 is 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) In Java, it is 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: 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:

a) the variable `randomNumbers`.

ANS: class body.

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:

```
1  // Exercise 6.3 Solution: 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.sqrt( 9.0 ) = %f\n", Math.sqrt( 9.0 ) );
39       System.out.printf( "Math.tan( 0.0 ) = %f\n", Math.tan( 0.0 ) );
40    } // end main
41 } // 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.sqrt( 9.0 ) = 3.000000
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)
```
int g()
{
   System.out.println( "Inside method g" );
   int h()
   {
      System.out.println( "Inside method h" );
   }
}
```

```
1  // Exercise 6.5a Solution: PartAError.java
2  public class PartAError
3  {
4     void g()
5     {
6        System.out.println( "Inside method g" );
```

```
7         void h()
8         {
9            System.out.println( "Inside method h" );
10        } // end method h
11     } // end method g
12  } // end class PartAError
```

```
PartAError.java:7: illegal start of expression
      void h()
      ^
PartAError.java:10: ';' expected
      } // end method h
       ^
2 errors
```

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

```
1  // Exercise 6.5a Solution: PartACorrect.java
2  public class PartACorrect
3  {
4     void g()
5     {
6        System.out.println( "Inside method g" );
7     } // end method g
8
9     void h()
10    {
11       System.out.println( "Inside method h" );
12    } // end method h
13 } // end class PartACorrect
```

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;
```

```
1  // Exercise 6.5b Solution: PartBCorrect.java
2  public class PartBCorrect
3  {
4     int sum( int x, int y )
5     {
6        return x + y;
```

```
7      } // end method sum
8   } // end class PartBCorrect
```

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

```
1  // Exercise 6.5c Solution: PartCError.java
2  public class PartCError
3  {
4     void f( float a );
5     {
6        float a;
7        System.out.println( a );
8     } // end method f
9  } // end class PartCError
```

```
PartCError.java:4: missing method body, or declare abstract
   void f( float a );
        ^
1 error
```

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;`.

```
1  // Exercise 6.5c Solution: PartCCorrect.java
2  public class PartCCorrect
3  {
4     void f( float a )
5     {
6        System.out.println( a );
7     } // end method f
8  } // end class PartCCorrect
```

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;
   }
```

```
1  // Exercise 6.5d Solution: PartDError.java
2  public class PartDError
```

```
   {
      void product()
      {
         int a = 6, b = 5, c = 4, result;
         result = a * b * c;
         System.out.println( "Result is " + result );
         return result;
      } // end method product
   } // end class PartDError
```

```
PartDError.java:9: cannot return a value from method whose result type is
void
      return result;
             ^
1 error
```

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

```
// Exercise 6.5d Solution: PartDCorrect.java
public class PartDCorrect
{
   int product()
   {
      int a = 6, b = 5, c = 4, result;
      result = a * b * c;
      System.out.println( "Result is " + result );
      return result;
   } // end method product
} // end class PartDCorrect
```

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:

```
// Exercise 6.6 Solution: Sphere.java
// Calculate the volume of a sphere.
import java.util.Scanner;

public class Sphere
{
   // obtain radius from user and display volume of sphere
   public void determineSphereVolume()
   {
      Scanner input = new Scanner( System.in );

```

```
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 Solution: 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

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

6.8 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. The program should use the method `calculateCharges` to determine the charge for each customer.

ANS:

```
1  // Exercise 6.8 Solution: Garage.java
2  // Program calculates charges for parking
3  import java.util.Scanner;
4
5  public class Garage
6  {
7     // begin calculating charges
8     public void startCharging()
9     {
10       Scanner input = new Scanner( System.in );
11
12       double totalReceipts = 0.0; // total fee collected for the day
13       double fee; // the charge for the current customer
14       double hours; // hours for the current customer
15
16       // read in the first customer's hours
17       System.out.print(
18          "Enter number of hours (a negative to quit): " );
19       hours = input.nextDouble();
20
21       while ( hours >= 0.0 )
22       {
23          // calculate and print the charges
24          fee = calculateCharges( hours );
25          totalReceipts += fee;
26          System.out.printf(
27             "Current charge: $%.2f, Total receipts: $%.2f\n",
28             fee, totalReceipts );
29
30          // read in the next customer's hours
31          System.out.print(
32             "Enter number of hours (a negative to quit): " );
33             hours = input.nextDouble();
34       } // end while loop
35    } // end method startCharging
36
37    // determines fee based on time
38    public double calculateCharges( double hours )
39    {
40       // apply minimum charge
41       double charge = 2.0;
42
43       // add extra fees as applicable
44       if ( hours > 3.0 )
45          charge = 2.0 + 0.5 * Math.ceil( hours - 3.0 );
46
```

```
47         // apply maximum value if needed
48         if ( charge > 10.0 )
49            charge = 10.0;
50
51         return charge;
52      } // end method calculateCharges
53   } // end class Garage
```

```
1   // Exercise 6.8 Solution: GarageTest.java
2   // Test application for class Garage
3   public class GarageTest
4   {
5      public static void main( String args[] )
6      {
7         Garage application = new Garage();
8         application.startCharging();
9      } // end main
10  } // end class GarageTest
```

```
Enter number of hours (a negative to quit): 2
Current charge: $2.00, Total receipts: $2.00
Enter number of hours (a negative to quit): 10
Current charge: $5.50, Total receipts: $7.50
Enter number of hours (a negative to quit): 20
Current charge: $10.00, Total receipts: $17.50
Enter number of hours (a negative to quit): -1
```

6.9 An application of method `Math.floor` is rounding a value to the nearest integer. The statement

```
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.

ANS:

```
1   // Exercise 6.9 Solution: RoundingTest.java
2   // Program tests Math.floor.
3   import java.util.Scanner;
4
5   public class RoundingTest
6   {
7      public static void main( String args[] )
8      {
9         Scanner input = new Scanner( System.in );
10
11        System.out.printf( "%s\n%s\n   %s\n   %s\n",
12           "Enter decimal numbers.",
13           "Type the end-of-file indicator to terminate input:",
```

```
            "On UNIX/Linux/Mac OS X type <ctrl> d then press Enter",
            "On Windows type <ctrl> z then press Enter" );

         while ( input.hasNext() )
         {
            double x = input.nextDouble();

            System.out.printf( "Number: %f\tMath.floor( x + .5 ): %f\n",
               x, Math.floor( x + .5 ) );
         } // end while loop
      } // end method main
   } // end class RoundingTest
```

```
Enter decimal numbers.
Type the end-of-file indicator to terminate input:
   On UNIX/Linux/Mac OS X type <ctrl> d then press Enter
   On Windows type <ctrl> z then press Enter
5.49
Number: 5.490000        Math.floor( x + .5 ): 5.000000
6.28
Number: 6.280000        Math.floor( x + .5 ): 6.000000
6.86
Number: 6.860000        Math.floor( x + .5 ): 7.000000
^Z
```

6.10 `Math.floor` may be used to round a number to a specific decimal place. The statement

```
y = Math.floor( x * 10 + 0.5 ) / 10;
```

rounds `x` to the tenths position (i.e., the first position to the right of the decimal point). The statement

```
y = Math.floor( x * 100 + 0.5 ) / 100;
```

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.

ANS:

```
// Exercise 6.10 Solution: Round.java
// Program tests rounding with Math.floor
import java.util.Scanner;

public class Round
{
```

```
7     // prints the various roundings for a number
8     public void printRoundings()
9     {
10       Scanner input = new Scanner( System.in );
11
12       System.out.printf( "%s\n%s\n   %s\n   %s\n",
13          "Enter decimal numbers.",
14          "Type the end-of-file indicator to terminate input:",
15          "On UNIX/Linux/Mac OS X type <ctrl> d then press Enter",
16          "On Windows type <ctrl> z then press Enter" );
17
18       while ( input.hasNext() )
19       {
20          double x = input.nextDouble();
21
22          // print the various roundings
23          System.out.printf( "The number: %f\n", x);
24          System.out.printf( "rounded to Integer: %f\n",
25             roundToInteger( x ) );
26          System.out.printf( "rounded to Tenth: %f\n",
27             roundToTenths( x ) );
28          System.out.printf( "rounded to Hundredth: %f\n",
29             roundToHundredths( x ) );
30          System.out.printf( "rounded to Thousandth: %f\n",
31             roundToThousandths( x ) );
32       } // end while loop
33    } // end method printRoundings
34
35    // round to ones place
36    public double roundToInteger( double number )
37    {
38       return( Math.floor( number + .5 ) );
39    } // end method roundToInteger
40
41    // round to tenths place
42    public double roundToTenths( double number )
43    {
44       return( Math.floor( number * 10 + .5 ) / 10 );
45    } // end method roundToTenths
46
47    // round to hundredths place
48    public double roundToHundredths( double number )
49    {
50       return( Math.floor( number * 100 + .5 ) / 100 );
51    } // end method roundToHundredths
52
53    // round to thousandths place
54    public double roundToThousandths( double number )
55    {
56       return( Math.floor( number * 1000 + .5 ) / 1000 );
57    } // end method roundToThousandths
58 } // end class Round
```

```
1  // Exercise 6.10 Solution: RoundTest.java
2  // Test application for class Round
3  public class RoundTest
4  {
5     public static void main( String args[] )
6     {
7        Round application = new Round();
8        application.printRoundings();
9     } // end main
10 } // end class RoundTest
```

```
Enter decimal numbers.
Type the end-of-file indicator to terminate input:
   On UNIX/Linux/Mac OS X type <ctrl> d then press Enter
   On Windows type <ctrl> z then press Enter
10.234
The number: 10.234000
rounded to Integer: 10.000000
rounded to Tenth: 10.200000
rounded to Hundredth: 10.230000
rounded to Thousandth: 10.234000
6.228
The number: 6.228000
rounded to Integer: 6.000000
rounded to Tenth: 6.200000
rounded to Hundredth: 6.230000
rounded to Thousandth: 6.228000
^Z
```

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 );`

```
1  // Exercise 6.12 Solution: RandomRange.java
2  import java.util.Random;
3
4  public class RandomRange
5  {
6     public static void main( String args[] )
7     {
8        Random randomNumbers = new Random();
9
10       // a)
11       System.out.println( 1 + randomNumbers.nextInt( 2 ) );
12
13       // b)
14       System.out.println( 1 + randomNumbers.nextInt( 100 ) );
15
16       // c)
17       System.out.println( randomNumbers.nextInt( 10 ) );
18
19       // d)
20       System.out.println( 1000 + randomNumbers.nextInt( 113 ) );
21
22       // e)
23       System.out.println( -1 + randomNumbers.nextInt( 3 ) );
24
25       // f)
26       System.out.println( -3 + randomNumbers.nextInt( 15 ) );
27    } // end main
28 } // end class RandomRange
```

```
2
18
0
1061
1
7
```

```
2
63
5
1071
1
-2
```

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 );`

```
1  // Exercise 6.13 Solution: RandomSet.java
2  import java.util.Random;
3
4  public class RandomSet
5  {
6     public static void main( String args[] )
7     {
8        Random randomNumbers = new Random();
9
10       // a)
11       System.out.println( 2 + randomNumbers.nextInt( 5 ) * 2 );
12
13       // b)
14       System.out.println( 3 + randomNumbers.nextInt( 5 ) * 2 );
15
16       // c)
17       System.out.println( 6 + randomNumbers.nextInt( 5 ) * 4 );
18    } // end main
19 } // end class RandomSet
```

```
10
3
14
```

```
2
11
18
```

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

$$base^{exponent}$$

For example, `integerPower( 3, 4 )` calculates 3^4 (or `3 * 3 * 3 * 3`). Assume that `exponent` is a positive, nonzero integer and that `base` is an integer. Method `integerPower` should use a `for` or `while` statement to control the calculation. Do not use any math library methods. Incorporate this method into an application that reads integer values for `base` and `exponent` and performs the calculation with the `integerPower` method.

ANS:

```
1  // Exercise 6.14 Solution: Power.java
2  // Program calculates an exponent
3  import java.util.Scanner;
4
5  public class Power
6  {
7     // begin calculating integer powers
8     public void calculate()
```

```
9      {
10        Scanner input = new Scanner( System.in );
11
12        System.out.print( "Enter base: " );
13        int base = input.nextInt();
14
15        System.out.print( "Enter exponent (negative to quit): " );
16        int exponent = input.nextInt();
17
18        // use a negative exponent as a sentinel
19        while ( exponent >= 0 )
20        {
21           System.out.printf( "%d to the %d is %d\n",
22              base, exponent, integerPower( base, exponent ) );
23
24           System.out.print( "Enter base: " );
25           base = input.nextInt();
26
27           System.out.print( "Enter exponent (negative to quit): " );
28           exponent = input.nextInt();
29        } // end while loop
30     } // end method calculate
31
32     // raise integer base to the exponent power
33     public int integerPower( int base, int exponent )
34     {
35        int product = 1;
36
37        for ( int i = 1; i <= exponent; i++ )
38           product *= base;
39
40        return product;
41     } // end method integerPower
42  } // end class Power
```

```
1   // Exercise 6.14 Solution: PowerTest.java
2   // Test application for class Power
3   public class PowerTest
4   {
5      public static void main( String args[] )
6      {
7         Power application = new Power();
8         application.calculate();
9      } // end main
10  } // end class PowerTest
```

```
Enter base: 10
Enter exponent (negative to quit): 3
10 to the 3 is 1000
Enter base: 0
Enter exponent (negative to quit): -1
```

6.15 Define a method `hypotenuse` that calculates the length of the hypotenuse of a right triangle when the lengths of the other two sides are given. (Use the sample data in Fig. 6.26.) The method should take two arguments of type `double` and return the hypotenuse as a `double`. Incorporate this method into an application that reads values for `side1` and `side2` and performs the calculation with the `hypotenuse` method. Determine the length of the hypotenuse for each of the triangles in Fig. 6.26.

Triangle	Side 1	Side 2
1	3.0	4.0
2	5.0	12.0
3	8.0	15.0

Fig. 6.26 | Values for the sides of triangles in Exercise 6.15.

ANS:

```
1  // Exercise 6.15 Solution: Triangle.java
2  // Program calculates the hypotenuse of a right triangle.
3  import java.util.Scanner;
4
5  public class Triangle
6  {
7     // reads in two sides and prints the hypotenuse
8     public void calculateHypotenuse()
9     {
10       Scanner input = new Scanner( System.in );
11
12       double side1; // first side of triangle
13       double side2; // second side of triangle
14
15       System.out.print( "Enter side 1 (negative to quit): " );
16       side1 = input.nextDouble();
17
18       while ( side1 > 0 )
19       {
20          System.out.print( "Enter side 2: " );
21          side2 = input.nextDouble();
22
23          System.out.printf( "Hypotenuse is: %f\n",
24             hypotenuse( side1, side2 ) );
25
26          System.out.print( "Enter side 1 (negative to quit): " );
27          side1 = input.nextDouble();
28       } // end while
29    } // end method calculateHypotenuse
30
31    // calculate hypotenuse given lengths of two sides
32    public double hypotenuse( double side1, double side2 )
33    {
```

```
34         double hypotenuseSquared = Math.pow( side1, 2 ) +
35            Math.pow( side2, 2 );
36
37         return Math.sqrt( hypotenuseSquared );
38      } // end method hypotenuse
39   } // end class Triangle
```

```
1  // Exercise 6.15 Solution: TriangleTest.java
2  // Test application for class Triangle
3  public class TriangleTest
4  {
5     public static void main( String args[] )
6     {
7        Triangle application = new Triangle();
8        application.calculateHypotenuse();
9     } // end main
10 } // end class TriangleTest
```

```
Enter side 1 (negative to quit): 8
Enter side 2: 15
Hypotenuse is: 17.000000
Enter side 1 (negative to quit): 5
Enter side 2: 12
Hypotenuse is: 13.000000
Enter side 1 (negative to quit): 3
Enter side 2: 4
Hypotenuse is: 5.000000
Enter side 1 (negative to quit): -1
```

6.16 Write a method `multiple` 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 `false` otherwise. [*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.

ANS:

```
1  // Exercise 6.16 Solution: Multiplicity.java
2  // Determines if the second number entered is a multiple of the first.
3  import java.util.Scanner;
4
5  public class Multiplicity
6  {
7     // checks if the second number is a multiple of the first
8     public void checkMultiples()
9     {
10       Scanner input = new Scanner( System.in );
11
12       int first; // the first number
13       int second; // the second number
14
```

```
15         System.out.print( "Enter first number (0 to exit): " );
16         first = input.nextInt();
17
18         // use 0 as the sentinel value, since we cannot divide by zero
19         while ( first != 0 )
20         {
21            System.out.print( "Enter second number: " );
22            second = input.nextInt();
23
24            if ( multiple( first, second ) )
25               System.out.printf( "%d is a multiple of %d\n",
26                  second, first );
27            else
28               System.out.printf( "%d is not a multiple of %d\n",
29                  second, first );
30
31            System.out.print( "Enter first number (0 to exit): " );
32            first = input.nextInt();
33         } // end while loop
34      } // end method checkMultiples
35
36      // determine if first int is a multiple of the second
37      public boolean multiple( int firstNumber, int secondNumber )
38      {
39         return secondNumber % firstNumber == 0;
40      } // end method multiple
41   } // end class Multiplicity
```

```
1   // Exercise 6.16 Solution: MultiplicityTest.java
2   // Test application for class Multiplicity
3   public class MultiplicityTest
4   {
5      public static void main( String args[] )
6      {
7         Multiplicity application = new Multiplicity();
8         application.checkMultiples();
9      } // end main
10  } // end class MultiplicityTest
```

```
Enter first number (0 to exit): 5
Enter second number: 100
100 is a multiple of 5
Enter first number (0 to exit): 5
Enter second number: 101
101 is not a multiple of 5
Enter first number (0 to exit): 0
```

6.17 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.

ANS:

```
// Exercise 6.17 Solution: EvenOdd.java
// Program determines if a number is odd or even.
import java.util.Scanner;

public class EvenOdd
{
   // determines whether numbers are even or odd
   public void checkEvenOdd()
   {
      Scanner input = new Scanner( System.in );

      System.out.printf( "%s\n%s\n   %s\n   %s\n",
         "Enter numbers to determine if they are even or odd.",
         "Type the end-of-file indicator to terminate input:",
         "On UNIX/Linux/Mac OS X type <ctrl> d then press Enter",
         "On Windows type <ctrl> z then press Enter" );

      while ( input.hasNext() )
      {
         int number = input.nextInt();

         if ( isEven( number ) )
            System.out.printf( "%d is even\n", number );
         else
            System.out.printf( "%d is odd\n", number );
      } // end while loop
   } // end method checkEvenOdd

   // return true if number is even
   public boolean isEven( int number )
   {
      return number % 2 == 0;
   } // end method isEven
} // end class EvenOdd
```

```
// Exercise 6.17 Solution: EvenOddTest.java
// Test application for class EvenOdd
public class EvenOddTest
{
   public static void main( String args[] )
   {
      EvenOdd application = new EvenOdd();
      application.checkEvenOdd();
   } // end main
} // end class EvenOddTest
```

```
Enter numbers to determine if they are even or odd.
Type the end-of-file indicator to terminate input:
   On UNIX/Linux/Mac OS X type <ctrl> d then press Enter
   On Windows type <ctrl> z then press Enter
11
11 is odd
6
6 is even
13
13 is odd
5
5 is odd
^Z
```

6.18 Write a method `squareOfAsterisks` 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 the `squareOfAsterisks` method.

ANS:

```
1  // Exercise 6.18 Solution: Square.java
2  // Program draws a square of asterisks.
3  import java.util.Scanner;
4
5  public class Square
6  {
7     // obtain value from user
8     public void drawSquare()
9     {
10       Scanner input = new Scanner( System.in );
11
12       System.out.print( "Enter square size: " );
13       int size = input.nextInt();
14
15       squareOfAsterisks( size );
16    } // end method drawSquare
17
18    // draw a square of asteriks
19    public void squareOfAsterisks( int side )
20    {
21       for ( int count = 1; count <= side * side; count++ ) {
22          System.out.print( "*" );
23
24          if ( count % side == 0 )
25             System.out.println();
26       } // end for loop
```

```
27      }  // end method squareOfAsterisks
28  } // end class Square
```

```
1   // Exercise 6.18 Solution: SquareTest.java
2   // Test application for class Square
3   public class SquareTest
4   {
5      public static void main( String args[] )
6      {
7         Square application = new Square();
8         application.drawSquare();
9      } // end main
10  } // end class SquareTest
```

```
Enter square size: 8
********
********
********
********
********
********
********
********
```

```
Enter square size: 12
************
************
************
************
************
************
************
************
************
************
************
************
```

6.19 Modify the method created in Exercise 6.18 to form the square out of whatever character is contained in character parameter `fillCharacter`. Thus, if `side` is `5` and `fillCharacter` is "`#`", the method should display

```
#####
#####
#####
#####
#####
```

ANS:

```
1   // Exercise 6.19 Solution: Square2.java
2   // Program draws a square of asterisks
```

```
3  import java.util.Scanner;
4
5  public class Square2
6  {
7     // obtain value from user
8     public void drawSquare()
9     {
10       Scanner input = new Scanner( System.in );
11
12       System.out.print( "Enter square size: " );
13       int size = input.nextInt();
14
15       System.out.print( "Enter fill character: " );
16       char fillCharacter = input.next().charAt( 0 );
17
18       fillSquare( size, fillCharacter );
19    } // end method drawSquare
20
21    // draw a square of asteriks
22    public void fillSquare( int side, char fillCharacter )
23    {
24       for ( int count = 1; count <= side * side; count++ ) {
25          System.out.print( fillCharacter );
26
27          if ( count % side == 0 )
28             System.out.println();
29       } // end for loop
30    } // end method fillSquare
31 } // end class Square2
```

```
1  // Exercise 6.19 Solution: Square2Test.java
2  // Test application for class Square2
3  public class Square2Test
4  {
5     public static void main( String args[] )
6     {
7        Square2 application = new Square2();
8        application.drawSquare();
9     } // end main
10 } // end class Square2Test
```

```
Enter square size: 5
Enter fill character: #
#####
#####
#####
#####
#####
```

6.20 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.

ANS:

```
1  // Exercise 6.20 Solution: Circle.java
2  // Program calculates the area of a circle.
3  import java.util.Scanner;
4
5  public class Circle
6  {
7     // calculate the areas of circles
8     public void calculateAreas()
9     {
10       Scanner input = new Scanner( System.in );
11
12       System.out.print( "Enter the radius (negative to quit): " );
13       double radius = input.nextDouble();
14
15       while ( radius >= 0 )
16       {
17          circleArea( radius );
18
19          System.out.print( "Enter the radius (negative to quit): " );
20          radius = input.nextDouble();
21       } // end while loop
22    } // end method calculateAreas
23
24    // calculate area
25    public void circleArea( double radius )
26    {
27       System.out.printf( "Area is %f\n", Math.PI * radius * radius );
28    } // end method circleArea
29 } // end class Circle
```

```
1  // Exercise 6.20 Solution: CircleTest.java
2  // Test application for class Circle
3  public class CircleTest
4  {
5     public static void main( String args[] )
6     {
7        Circle application = new Circle();
8        application.calculateAreas();
9     } // end main
10 } // end class CircleTest
```

```
Enter the radius (negative to quit): 10
Area is 314.159265
Enter the radius (negative to quit): -1
```

6.21 Write program segments 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 program pieces developed in parts (a) and (b) to write a method `displayDigits` that receives an integer between `1` and `99999` and displays it as a sequence of digits, separating each pair of digits by two spaces. For example, the integer `4562` should appear as

```
4  5  6  2
```

d) Incorporate the method developed in part (c) into an application that inputs an integer and calls `displayDigits` by passing the method the integer entered. Display the results.

ANS:

```
1  // Exercise 6.21 Solution: Digits.java
2  // Program separates a five-digit number
3  // into its individual digits.
4  import java.util.Scanner;
5
6  public class Digits
7  {
8     // displays the individual digits of a number
9     public void separateDigits()
10    {
11       Scanner input = new Scanner( System.in );
12
13       System.out.print( "Enter the integer (0 to exit): " );
14       int number = input.nextInt();
15
16       while ( number != 0 )
17       {
18          if ( number <= 99999 && number >= 1 )
19            displayDigits( number );
20          else
21             System.out.println( "number must be between 1 and 99999" );
22
23          System.out.print( "Enter the integer (0 to exit): " );
24          number = input.nextInt();
25       } // end while loop
26    } // end method separateDigits
27
28    // part A
29    public int quotient( int a, int b )
30    {
31       return a / b;
32    } // end method quotient
33
34    // part B
35    public int remainder( int a, int b )
36    {
37       return a % b;
38    } // end method remainder
39
40    // part C
41    public void displayDigits( int number )
42    {
43       int divisor = 1, digit;
```

```
44         String result = "";
45
46         // Loop for highest divisor
47         for ( int i = 1; i < number; i *= 10 )
48            divisor = i;
49
50         while ( divisor >= 1 )
51         {
52            digit = quotient( number, divisor );
53
54            result += digit + "  ";
55
56            number = remainder( number, divisor );
57            divisor = quotient( divisor, 10 );
58         } // end while loop
59
60         System.out.println( result );
61      } // end method displayDigits
62   } // end class Digits
```

```
1  // Exercise 6.21 Solution: DigitsTest.java
2  // Test application for class Digits
3  public class DigitsTest
4  {
5     public static void main( String args[] )
6     {
7        Digits application = new Digits();
8        application.separateDigits();
9     } // end main
10 } // end class DigitsTest
```

```
Enter the integer (0 to exit): 60450
6  0  4  5  0
Enter the integer (0 to exit): 0
```

6.22 Implement the following integer methods:

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

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

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

```
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.

ANS:

```
// Exercise 6.22 Solution: Convert.java
// Program converts Fahrenheit to Celsius and vice versa.
import java.util.Scanner;

public class Convert
{
   // convert temperatures
   public void convertTemperatures()
   {
      Scanner input = new Scanner( System.in );

      int choice; // the user's choice in the menu

      do
      {
         // print the menu
         System.out.println( "1. Fahrenheit to Celsius" );
         System.out.println( "2. Celsius to Fahrenheit" );
         System.out.println( "3. Exit" );
         System.out.print( "Choice: " );
         choice = input.nextInt();

         if ( choice != 3 )
         {
            System.out.print( "Enter temperature: " );
            int oldTemperature = input.nextInt();

            // convert the temperature appropriately
            switch ( choice )
            {
               case 1:
                  System.out.printf( "%d Fahrenheit is %d Celsius\n",
                     oldTemperature, celsius( oldTemperature ) );
                  break;

               case 2:
                  System.out.printf( "%d Celsius is %d Fahrenheit\n",
                    oldTemperature, fahrenheit( oldTemperature ) );
                  break;
            } // end switch
         } // end if
      } while ( choice != 3 );
   } // end method convertTemperatures

   // return Celsius equivalent of Fahrenheit temperature
   public int celsius( int fahrenheitTemperature )
   {
      return ( (int) ( 5.0 / 9.0 * ( fahrenheitTemperature - 32 ) ) );
   } // end method celsius
```

```
50
51       // return Fahrenheit equivalent of Celsius temperature
52       public int fahrenheit( int celsiusTemperature )
53       {
54          return ( (int) ( 9.0 / 5.0 * celsiusTemperature + 32 ) );
55       } // end method fahrenheit
56    } // end class Convert
```

```
1  // Exercise 6.22 Solution: ConvertTest.java
2  // Test application for class Convert
3  public class ConvertTest
4  {
5     public static void main( String args[] )
6     {
7        Convert application = new Convert();
8        application.convertTemperatures();
9     } // end main
10 } // end class ConvertTest
```

```
1. Fahrenheit to Celsius
2. Celsius to Fahrenheit
3. Exit
Choice: 1
Enter temperature: 32
32 Fahrenheit is 0 Celsius
1. Fahrenheit to Celsius
2. Celsius to Fahrenheit
3. Exit
Choice: 2
Enter temperature: 100
100 Celsius is 212 Fahrenheit
1. Fahrenheit to Celsius
2. Celsius to Fahrenheit
3. Exit
Choice: 3
```

6.23 Write a method `minimum3` that returns the smallest of three floating-point numbers. Use the `Math.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.

ANS:

```
1  // Exercise 6.23 Solution: Min.java
2  // Program finds the minimum of 3 numbers
3  import java.util.Scanner;
4
5  public class Min
6  {
7     // find the minimum of three numbers
8     public void findMinimum()
9     {
10       Scanner input = new Scanner( System.in );
```

```
11
12       double one; // first number
13       double two; // second number
14       double three; // third number
15
16       System.out.printf( "%s\n   %s\n   %s\n",
17          "Type the end-of-file indicator to terminate",
18          "On UNIX/Linux/Mac OS X type <ctrl> d then press Enter",
19          "On Windows type <ctrl> z then press Enter" );
20       System.out.print( "Or enter first number: " );
21
22       while ( input.hasNext() )
23       {
24          one = input.nextDouble();
25          System.out.print( "Enter second number: " );
26          two = input.nextDouble();
27          System.out.print( "Enter third number: " );
28          three = input.nextDouble();
29
30          System.out.printf( " Minimum is %f\n",
31             minimum3( one, two, three ) );
32
33          System.out.printf( "\n%s\n   %s\n   %s\n",
34             "Type the end-of-file indicator to terminate",
35             "On UNIX/Linux/Mac OS X type <ctrl> d then press Enter",
36             "On Windows type <ctrl> z then press Enter" );
37          System.out.print( "Or enter first number: " );
38       } // end while
39    } // end method findMinimum
40
41    // determine the smallest of three numbers
42    public double minimum3( double one, double two, double three )
43    {
44       // use a nested pair of min statements
45       return Math.min( Math.min( one, two ), three );
46    } // end method minimum3
47 } // end class Min
```

```
1  // Exercise 6.23 Solution: MinTest.java
2  // Test application for class Min
3  public class MinTest
4  {
5     public static void main( String args[] )
6     {
7        Min application = new Min();
8        application.findMinimum();
9     } // end main
10 } // end class MinTest
```

```
Type the end-of-file indicator to terminate
   On UNIX/Linux/Mac OS X type <ctrl> d then press Enter
   On Windows type <ctrl> z then press Enter
Or enter first number: 1.1
Enter second number: 2.2
Enter third number: 3.3
Minimum is 1.100000

Type the end-of-file indicator to terminate
   On UNIX/Linux/Mac OS X type <ctrl> d then press Enter
   On Windows type <ctrl> z then press Enter
Or enter first number: 3.3
Enter second number: 2.2
Enter third number: 1.1
Minimum is 1.100000

Type the end-of-file indicator to terminate
   On UNIX/Linux/Mac OS X type <ctrl> d then press Enter
   On Windows type <ctrl> z then press Enter
Or enter first number: 2.2
Enter second number: 1.1
Enter third number: 3.3
Minimum is 1.100000

Type the end-of-file indicator to terminate
   On UNIX/Linux/Mac OS X type <ctrl> d then press Enter
   On Windows type <ctrl> z then press Enter
Or enter first number: ^Z
```

6.24 An integer number is said to be a *perfect 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 `perfect` that determines whether parameter `number` is a perfect number. Use this method in an application that determines and 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.

ANS:

```
1  // Exercise 6.24 Solution: PerfectNumber.java
2  // Program displays all perfect numbers between 1 and 1000.
3
4  public class PerfectNumber
5  {
6     // finds all the perfect numbers from 2 to 1000
7     public void findPerfects()
8     {
9        for ( int number = 2; number <= 1000; number++ )
10       {
11          String result = perfect( number );
12
13          if ( result != null )
14             System.out.printf ( "%d is perfect.\n\tFactors: %s\n",
15                number, result );
16       } // end for
```

```
17      } // end main
18
19      // returns a string of factors if parameter is a
20      // perfect number, or null if it isn't.
21      public String perfect( int value )
22      {
23         int factorSum = 1;
24         String factors = "1 ";
25
26         for ( int test = 2; test <= value / 2; test++ )
27         {
28            if ( value % test == 0 )
29            {
30               factorSum += test;
31               factors += test + " ";
32            } // end if
33         } // end for
34
35         if ( factorSum == value )
36            return factors;
37
38         return null;
39      } // end method perfect
40   } // end class PerfectNumber
```

```
1   // Exercise 6.24 Solution: PerfectNumberTest.java
2   // Test application for class PerfectNumber
3   public class PerfectNumberTest
4   {
5      public static void main( String args[] )
6      {
7         PerfectNumber application = new PerfectNumber();
8         application.findPerfects();
9      } // end main
10  } // end class PerfectNumberTest
```

```
6 is perfect.
        Factors: 1 2 3
28 is perfect.
        Factors: 1 2 4 7 14
496 is perfect.
        Factors: 1 2 4 8 16 31 62 124 248
```

6.25 An integer is said to be *prime* if it is divisible by only 1 and itself. For example, 2, 3, 5 and 7 are prime, but 4, 6, 8 and 9 are not.

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 have found all the primes?

ANS:

```
1  // Exercise 6.25 Part A and B Solution: PrimeNum.java
2  // Program calculates prime numbers
3  public class PrimeNum
4  {
5     // find the prime numbers between 1 and 10,000
6     public void findPrimes()
7     {
8        System.out.println( "Prime numbers between 2 and 10,000 are: " );
9
10       // test all numbers between 2 and 10000
11       for ( int m = 2; m <= 10000; m++ )
12          if ( prime( m ) )
13             System.out.println( m );
14    } // end method findPrimes
15
16    // a helper method for determining if a number is prime
17    // (This is the solution to 6.25, Part A.)
18    public boolean prime( int n )
19    {
20       for ( int v = 2; v < n ; v++ )
21          if ( n % v == 0 )
22             return false;
23
24       return true;
25    } // end method prime
26 } // end class PrimeNum
```

```
1  // Exercise 6.25 Part A and B Solution: PrimeNumTest.java
2  // Test application for class PrimeNum
3  public class PrimeNumTest
4  {
5     public static void main( String args[] )
6     {
7        PrimeNum application = new PrimeNum();
8        application.findPrimes();
9     } // end main
10 } // end class PrimeNumTest
```

```
Prime numbers between 2 and 10,000 are:
2
3
5
7
11

.
.
.

9941
9949
9967
9973
```

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

ANS:

```
1  // Exercise 6.25 Part C Solution: PrimeNum2.java
2  // Program calculates prime numbers more efficiently
3  public class PrimeNum2
4  {
5     // find the prime numbers between 1 and 10,000
6     public void findPrimes()
7     {
8        System.out.println( "Prime numbers between 2 and 10,000 are: " );
9
10       // test all numbers between 2 and 10000
11       for ( int m = 2; m <= 10000; m++ )
12          if ( prime( m ) )
13             System.out.println( m );
14    } // end method findPrimes
15
16    // a helper method for determining if a number is prime
17    public boolean prime( int n )
18    {
19       int max = (int) Math.sqrt( n ); // the highest number to test
20
21       for ( int v = 2; v <= max; v++ )
22          if ( n % v == 0 )
23             return false;
24
25       return true;
26    } // end method prime
27 } // end class PrimeNum2
```

```
1  // Exercise 6.25 Part C Solution: PrimeNum2Test.java
2  // Test application for class PrimeNum2
```

```
3  public class PrimeNum2Test
4  {
5     public static void main( String args[] )
6     {
7        PrimeNum application = new PrimeNum2();
8        application.findPrimes();
9     } // end main
10 } // end class PrimeNum2Test
```

```
Prime numbers between 2 and 10,000 are:
2
3
5
7
11

.
.
.

9941
9949
9967
9973
```

6.26 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.

ANS:

```
1  // Exercise 6.26 Solution: Reverse.java
2  // Program takes a number and prints it out
3  // with its digits reversed.
4  import java.util.Scanner;
5
6  public class Reverse
7  {
8     // reverses an Integer
9     public void reverseInteger()
10    {
11       Scanner input = new Scanner( System.in );
12
13       System.out.print( "Enter an integer (-1 to exit): " );
14       int number = input.nextInt();
15
16       while ( number != -1 )
17       {
18          System.out.printf( "%d reversed is %d\n",
19          number, reverseDigits( number ) );
20
21          System.out.print( "Enter an integer (-1 to exit): " );
22          number = input.nextInt();
```

```
23         } // end while loop
24      } // end method reverseInteger
25
26      // print parameter number with digits reversed
27      public int reverseDigits( int number )
28      {
29         int reverseNumber = 0; // the number in reverse order
30         int placeValue; // the value at the current place
31
32         while ( number > 0 )
33         {
34            placeValue = number % 10;
35            number = number / 10;
36            reverseNumber = reverseNumber * 10 + placeValue;
37         } // end while loop
38
39         return reverseNumber;
40      } // end method reverseDigits
41   } // end class Reverse
```

```
1  // Exercise 6.26 Solution: ReverseTest.java
2  // Test application for class Reverse
3  public class ReverseTest
4  {
5     public static void main( String args[] )
6     {
7        Reverse application = new Reverse();
8        application.reverseInteger();
9     } // end main
10 } // end class ReverseTest
```

```
Enter an integer (-1 to exit): 54321
54321 reversed is 12345
Enter an integer (-1 to exit): -1
```

6.27 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 the algorithm at `en.wikipedia.org/wiki/Euclidean_algorithm`.] Incorporate the method into an application that reads two values from the user and displays the result.

ANS:

```
1  // Exercise 6.27 Solution: Divisor.java
2  // Program finds the greatest common divisor of two numbers.
3  import java.util.Scanner;
4
5  public class Divisor
6  {
7     // finds the gcd of two numbers
8     public void findGCD()
```

```
9      {
10        Scanner input = new Scanner( System.in );
11
12        int num1; // first number
13        int num2; // second number
14
15        System.out.print( "Enter first number (-1 to exit): " );
16        num1 = input.nextInt();
17
18        while ( num1 != -1 )
19        {
20           System.out.print( "Enter second number: " );
21           num2 = input.nextInt();
22
23           System.out.printf( "GCD is: %d\n", gcd( num1, num2 ) );
24
25           System.out.print( "Enter first number (-1 to exit): " );
26           num1 = input.nextInt();
27        } // end while
28     } // end method findGCD
29
30     // calculate the greatest common divisor using Euclid's Algorithm
31     // alternatively, you can simply check every number up to the
32     // lesser of x or y to see if it divides both x and y.
33     public int gcd( int x, int y )
34     {
35        int mod; // remainder of x / y
36
37        while ( y != 0 )
38        {
39           mod = x % y;
40           x = y;
41           y = mod;
42        } // end while loop
43
44        return x;
45     } // end method gcd
46  } // end class Divisor
```

```
1   // Exercise 6.27 Solution: DivisorTest.java
2   // Test application for class Divisor
3   public class DivisorTest
4   {
5      public static void main( String args[] )
6      {
7         Divisor application = new Divisor();
8         application.findGCD();
9      } // end main
10  } // end class DivisorTest
```

```
Enter first number (-1 to exit): 25
Enter second number: 100
GCD is: 25
Enter first number (-1 to exit): 10
Enter second number: 2
GCD is: 2
Enter first number (-1 to exit): 99
Enter second number: 27
GCD is: 9
Enter first number (-1 to exit): 11
Enter second number: 3
GCD is: 1
Enter first number (-1 to exit): -1
```

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

ANS:

```
1  // Exercise 6.28 Solution: Average.java
2  // Program displays a number
3  // representing the student's average.
4  import java.util.Scanner;
5
6  public class Average
7  {
8     public void computePoints()
9     {
10       Scanner input = new Scanner( System.in );
11
12       System.out.print( "Enter average (-1 to quit): " );
13       int inputNumber = input.nextInt();
14
15       while ( inputNumber != -1 )
16       {
17          if ( inputNumber >= 0 && inputNumber <= 100 )
18             System.out.printf( "Point is: %d\n",
19                qualityPoints( inputNumber ) );
20          else
21             System.out.println( "Invalid input." );
22
23          System.out.print( "Enter average (-1 to quit): " );
24          inputNumber = input.nextInt();
25       } // end while loop
26    } // end method actionPerformed
27
28    // return single-digit value of grade
29    public int qualityPoints( int grade )
30    {
31       if ( grade >= 90 )
32          return 4;
```

```
33         else if ( grade >= 80 )
34            return 3;
35         else if ( grade >= 70 )
36            return 2;
37         else if ( grade >= 60 )
38            return 1;
39         else
40            return 0;
41      } // end method qualityPoints
42   } // end class Average
```

```
1   // Exercise 6.28 Solution: AverageTest.java
2   // Test application for class Average
3   public class AverageTest
4   {
5      public static void main( String args[] )
6      {
7         Average application = new Average();
8         application.computePoints();
9      } // end main
10  } // end class AverageTest
```

```
Enter average (-1 to quit): 90
Point is: 4
Enter average (-1 to quit): 80
Point is: 3
Enter average (-1 to quit): -1
```

6.29 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 `false` for tails and `true` for heads. [*Note*: If the program realistically simulates coin tossing, each side of the coin should appear approximately half the time.]

ANS:

```
1   // Exercise 6.29 Solution: Coin.java
2   // Program simulates tossing a coin.
3   import java.util.*;
4
5   public class Coin
6   {
7      private Random randomNumbers = new Random();
8
9      // flips a coin many times
10     public void flipCoins()
11     {
12        Scanner input = new Scanner( System.in );
13
14        int heads = 0; // the number of times heads shows up
15        int tails = 0; // the number of times tails shows up
```

```
16         int choice; // the user's choice
17
18         do
19         {
20            // display a menu
21            System.out.println( "1. Toss Coin" );
22            System.out.println( "2. Exit" );
23            System.out.print( "Choice: " );
24            choice = input.nextInt();
25
26            if ( choice == 1 )
27            {
28               if ( flip() )
29                  heads++;
30               else
31                  tails++;
32
33               System.out.printf( "Heads: %d, Tails: %d\n", heads, tails );
34            } // end if
35
36         } while ( choice != 2 );
37      } // end method flipCoins
38
39      // simulate flipping
40      public boolean flip()
41      {
42         return randomNumbers.nextInt( 2 ) == 1;
43      } // end method flip
44   } // end class Coin
```

```
1   // Exercise 6.29 Solution: CoinTest.java
2   // Test application for class Coin
3   public class CoinTest
4   {
5      public static void main( String args[] )
6      {
7         Coin application = new Coin();
8         application.flipCoins();
9      } // end main
10  } // end class CoinTest
```

```
1. Toss Coin
2. Exit
Choice: 1
Heads: 0, Tails: 1
1. Toss Coin
2. Exit
Choice: 1
Heads: 0, Tails: 2

.
.
.

1. Toss Coin
2. Exit
Choice: 1
Heads: 23, Tails: 24
1. Toss Coin
2. Exit
Choice: 1
Heads: 24, Tails: 24
1. Toss Coin
2. Exit
Choice: 2
```

6.30 Computers are playing an increasing role in education. Write a program that will help an elementary school student learn multiplication. Use a `Random` 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 is 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.

ANS:

```
1  // Exercise 6.30 Solution: Multiply.java
2  // Program generates single-digit multiplication problems
3  import java.util.*;
4
5  public class Multiply
6  {
7     Random randomNumbers = new Random();
8
9     int answer; // the correct answer
10
11    // ask the user to answer multiplication problems
12    public void quiz()
13    {
14       Scanner input = new Scanner( System.in );
```

```
15
16       int guess; // the user's guess
17
18       createQuestion(); // display the first question
19
20       System.out.println( "Enter your answer (-1 to exit):" );
21       guess = input.nextInt();
22
23       while ( guess != -1 )
24       {
25          checkResponse( guess );
26
27          System.out.println( "Enter your answer (-1 to exit):" );
28          guess = input.nextInt();
29       } // end while
30    } // end method
31
32    // prints a new question and stores the corresponding answer
33    public void createQuestion()
34    {
35       // get two random numbers between 0 and 9
36       int digit1 = randomNumbers.nextInt( 10 );
37       int digit2 = randomNumbers.nextInt( 10 );
38
39       answer = digit1 * digit2;
40       System.out.printf( "How much is %d times %d?\n",
41          digit1, digit2 );
42    } // end method createQuestion
43
44    // checks if the user answered correctly
45    public void checkResponse( int guess )
46    {
47       if ( guess != answer )
48          System.out.println( "No. Please try again." );
49       else
50       {
51          System.out.println( "Very Good!" );
52          createQuestion();
53       } // end else
54    } // end method checkResponse
55 } // end class Multiply
```

```
1  // Exercise 6.30 Solution: MultiplyTest.java
2  // Test application for class Multiply
3  public class MultiplyTest
4  {
5     public static void main( String args[] )
6     {
7        Multiply application = new Multiply();
8        application.quiz();
9     } // end main
10 } // end class MultiplyTest
```

```
How much is 8 times 4?
Enter your answer (-1 to exit):
32
Very Good!
How much is 4 times 9?
Enter your answer (-1 to exit):
38
No. Please try again.
Enter your answer (-1 to exit):
36
Very Good!
How much is 0 times 2?
Enter your answer (-1 to exit):
-1
```

6.31 The use of computers in education is referred to as *computer-assisted instruction* (*CAI*). One problem that develops in CAI environments is student fatigue. This problem can be eliminated by varying the computer's responses to hold the student's attention. Modify the program of Exercise 6.30 so that the various comments are displayed for each correct answer and each incorrect answer as follows:

Responses to a correct answer:

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

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 an appropriate response to each answer. Use a `switch` statement to issue the responses.

ANS:

```
1  // Exercise 6.31 Solution: Multiply2.java
2  // Program generates single-digit multiplication problems
3  import java.util.*;
4
5  public class Multiply2
6  {
7     Random randomNumbers = new Random();
8
9     int answer; // the correct answer
10
11    // ask the user to answer multiplication problems
12    public void quiz()
13    {
14       Scanner input = new Scanner( System.in );
15
16       int guess; // the user's guess
```

```
17
18          createQuestion(); // display the first question
19
20          System.out.println( "Enter your answer (-1 to exit):" );
21          guess = input.nextInt();
22
23          while ( guess != -1 )
24          {
25             checkResponse( guess );
26
27             System.out.println( "Enter your answer (-1 to exit):" );
28             guess = input.nextInt();
29          } // end while
30       } // end method
31
32       // prints a new question and stores the corresponding answer
33       public void createQuestion()
34       {
35          // get two random numbers between 0 and 9
36          int digit1 = randomNumbers.nextInt( 10 );
37          int digit2 = randomNumbers.nextInt( 10 );
38
39          answer = digit1 * digit2;
40          System.out.printf( "How much is %d times %d?\n",
41             digit1, digit2 );
42       } // end method createQuestion
43
44       // create a new response
45       public String createResponse( boolean correct )
46       {
47          if ( correct )
48             switch ( randomNumbers.nextInt( 4 ) )
49             {
50                case 0:
51                   return( "Very good!" );
52
53                case 1:
54                   return( "Excellent!" );
55
56                case 2:
57                   return( "Nice work!" );
58
59                case 3:
60                   return( "Keep up the good work!" );
61             } // end switch
62
63          // otherwise, assume incorrect
64          switch ( randomNumbers.nextInt( 4 ) )
65          {
66             case 0:
67                return( "No. Please try again." );
68
69             case 1:
70                return( "Wrong. Try once more." );
```

```
71
72           case 2:
73              return( "Don't give up!" );
74
75           case 3: default:
76              return( "No. Keep trying." );
77        } // end switch
78     } // end method createResponse
79
80     // checks if the user answered correctly
81     public void checkResponse( int guess )
82     {
83        if ( guess != answer )
84           System.out.println( createResponse( false ) );
85        else
86        {
87           System.out.println( createResponse( true ) );
88           createQuestion();
89        } // end else
90     } // end method checkResponse
91  } // end class Multiply2
```

```
1  // Exercise 6.31 Solution: Multiply2Test.java
2  // Test application for class Multiply2
3  public class Multiply2Test
4  {
5     public static void main( String args[] )
6     {
7        Multiply2 application = new Multiply2();
8        application.quiz();
9     } // end main
10 } // end class Multiply2Test
```

```
How much is 2 times 3?
Enter your answer (-1 to exit):
6
Very good!
How much is 0 times 9?
Enter your answer (-1 to exit):
0
Nice work!
How much is 2 times 8?
Enter your answer (-1 to exit):
13
No. Please try again.
Enter your answer (-1 to exit):
14
No. Keep trying.
Enter your answer (-1 to exit):
16
Excellent!
How much is 1 times 8?
Enter your answer (-1 to exit):
-1
```

6.32 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 6.31 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 of correct responses. If the percentage is lower than 75%, display `Please ask your instructor for extra help` and reset the program so another student can try it.

ANS:

```
1  // Exercise 6.32 Solution: Multiply3.java
2  // Program generates single-digit multiplication problems
3  import java.util.*;
4
5  public class Multiply3
6  {
7     Random randomNumbers = new Random();
8
9     int answer; // the correct answer
10    int count; // number of questions answered
11    int correct; // number of correct answers
12
13    // ask the user to answer multiplication problems
14    public void quiz()
15    {
16       Scanner input = new Scanner( System.in );
17
18       int guess; // the user's guess
19
20       createQuestion(); // display the first question
21
22       System.out.println( "Enter your answer (-1 to exit):" );
23       guess = input.nextInt();
24
25       while ( guess != -1 )
26       {
27          checkResponse( guess );
28
29          System.out.println( "Enter your answer (-1 to exit):" );
30          guess = input.nextInt();
31       } // end while
32    } // end method
33
34    // prints a new question and stores the corresponding answer
35    public void createQuestion()
36    {
37       // get two random numbers between 0 and 9
38       int digit1 = randomNumbers.nextInt( 10 );
39       int digit2 = randomNumbers.nextInt( 10 );
40
41       answer = digit1 * digit2;
42       System.out.printf( "How much is %d times %d?\n",
43          digit1, digit2 );
44    } // end method createQuestion
45
46    // create a new response
```

```
47    public String createResponse( boolean correct )
48    {
49       if ( correct )
50          switch ( randomNumbers.nextInt( 4 ) )
51          {
52             case 0:
53                return( "Very good!" );
54
55             case 1:
56                return( "Excellent!" );
57
58             case 2:
59                return( "Nice work!" );
60
61             case 3:
62                return( "Keep up the good work!" );
63          } // end switch
64
65       // otherwise, assume incorrect
66       switch ( randomNumbers.nextInt( 4 ) )
67       {
68          case 0:
69             return( "No. Please try again." );
70
71          case 1:
72             return( "Wrong. Try once more." );
73
74          case 2:
75             return( "Don't give up!" );
76
77          case 3: default:
78             return( "No. Keep trying." );
79       } // end switch
80    } // end method createResponse
81
82    // determine how the user is faring
83    public double calculatePercentage()
84    {
85       return ( double ) correct / count;
86    } // end method calculatePercentage
87
88    // checks if the user answered correctly
89    public void checkResponse( int guess )
90    {
91
92       count++;
93
94       if ( guess != answer )
95          System.out.println( createResponse( false ) );
96       else
97       {
98          correct++;
99          System.out.println( createResponse( true ) );
100         if ( count < 10 )
```

```
                createQuestion();
            } // end else

            if ( count >= 10 )
            {
               System.out.printf( "You scored a %d\n",
                  ( int ) ( calculatePercentage()*100 ) );

               if ( calculatePercentage() < 0.75 )
                  System.out.println(
                     "Please ask your instructor for extra help." );

               // start over
               System.out.println();
               correct = 0;
               count = 0;
               createQuestion();
            } // end if
         } // end method checkResponse
      } // end class Multiply3
```

```
// Exercise 6.32 Solution: Multiply3Test.java
// Test application for class Multiply3
public class Multiply3Test
{
   public static void main( String args[] )
   {
      Multiply3 application = new Multiply3();
      application.quiz();
   } // end main
} // end class Multiply3Test
```

```
How much is 0 times 4?
Enter your answer (-1 to exit):
0
Nice work!
How much is 1 times 5?
Enter your answer (-1 to exit):
6
Don't give up!

.
.
.

How much is 0 times 1?
Enter your answer (-1 to exit):
0
Keep up the good work!
You scored a 90

How much is 4 times 1?
Enter your answer (-1 to exit):
-1
```

```
How much is 6 times 0?
Enter your answer (-1 to exit):
0
Keep up the good work!
How much is 9 times 8?
Enter your answer (-1 to exit):
72
Excellent!
How much is 7 times 4?
Enter your answer (-1 to exit):
26
No. Please try again.
Enter your answer (-1 to exit):
30
Don't give up!
Enter your answer (-1 to exit):
22
Don't give up!
Enter your answer (-1 to exit):
28
Nice work!
How much is 8 times 1?
Enter your answer (-1 to exit):
8
Keep up the good work!
How much is 4 times 8?
Enter your answer (-1 to exit):
30
Don't give up!
Enter your answer (-1 to exit):
32
Excellent!
How much is 7 times 0?
Enter your answer (-1 to exit):
0
Very Good!
You scored a 60
Please ask your instructor for extra help.
```

6.33 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, display `Congratulations. 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 16, Searching and Sorting.]

ANS:

```
1  // Exercise 6.33 Solution: Guess.java
2  // Program plays guess the number.
3  import java.util.*;
4
```

```
5  public class Guess
6  {
7     Random randomNumbers = new Random();
8     int answer; // the answer to be guessed
9
10    // play games of guess the number
11    public void play()
12    {
13       Scanner input = new Scanner( System.in );
14       int userGuess; // the guess made by the user
15
16       newGame();
17
18       System.out.print( "Guess (0 to exit): " );
19       userGuess = input.nextInt();
20
21       while ( userGuess != 0 )
22       {
23          checkUserGuess( userGuess );
24
25          System.out.print( "Guess (0 to exit): " );
26          userGuess = input.nextInt();
27       } // end while
28    } // end method play
29
30    // create a new number to guess
31    public int getNumber()
32    {
33       return 1 + randomNumbers.nextInt( 1000 );
34    } // end method getNumber
35
36    // starts a new game
37    public void newGame()
38    {
39       answer = getNumber();
40       System.out.println( "Guess a number between 1 and 1000" );
41    } // end method newGame
42
43    // checks user input
44    public void checkUserGuess( int userGuess )
45    {
46       if ( userGuess < answer )
47          System.out.printf( "%d is too low. Try again.\n", userGuess );
48       else if ( userGuess > answer )
49          System.out.printf( "%d is too high. Try again.\n", userGuess );
50       else
51       {
52          System.out.println(
53             "Congratulations. You guessed the number!\n" );
54          // new game
55          newGame();
56       } // end else
57    } // end method checkUserGuess
58 } // end class Guess
```

```
1  // Exercise 6.33 Solution: PlayGuess.java
2  // Application to play a game of guess the number
3  public class PlayGuess
4  {
5     public static void main( String args[] )
6     {
7        Guess application = new Guess();
8        application.play();
9     } // end main
10 } // end class PlayGuess
```

```
Guess a number between 1 and 1000
Guess (0 to exit): 500
500 is too low. Try again.
Guess (0 to exit): 750
750 is too low. Try again.
Guess (0 to exit): 875
875 is too high. Try again.
Guess (0 to exit): 812
812 is too high. Try again.
Guess (0 to exit): 781
781 is too low. Try again.
Guess (0 to exit): 796
796 is too high. Try again.
Guess (0 to exit): 788
788 is too high. Try again.
Guess (0 to exit): 784
784 is too high. Try again.
Guess (0 to exit): 782
Congratulations. You guessed the number!

Guess a number between 1 and 1000
Guess (0 to exit): 0
```

6.34 Modify the program of Exercise 6.33 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.

ANS:

```
1  // Exercise 6.34 Solution: Guess2.java
2  // Program plays guess the number.
3  import java.util.*;
4
5  public class Guess2
6  {
7     Random randomNumbers = new Random();
8     int answer; // the answer to be guessed
9     int guesses; // the number of guesses the user has made
10
```

```
11      // play games of guess the number
12      public void play()
13      {
14         Scanner input = new Scanner( System.in );
15         int userGuess; // the guess made by the user
16
17         newGame();
18
19         System.out.print( "Guess (0 to exit): " );
20         userGuess = input.nextInt();
21
22         while ( userGuess != 0 )
23         {
24            guesses++;
25            checkUserGuess( userGuess );
26
27            System.out.print( "Guess (0 to exit): " );
28            userGuess = input.nextInt();
29         } // end while
30      } // end method play
31
32      // create a new number to guess
33      public int getNumber()
34      {
35         return 1 + randomNumbers.nextInt( 1000 );
36      } // end method getNumber
37
38      // starts a new game
39      public void newGame()
40      {
41         guesses = 0;
42         answer = getNumber();
43         System.out.println( "Guess a number between 1 and 1000" );
44      } // end method newGame
45
46      // checks user input
47      public void checkUserGuess( int userGuess )
48      {
49         if ( userGuess < answer )
50            System.out.printf( "%d is too low. Try again.\n", userGuess );
51         else if ( userGuess > answer )
52            System.out.printf( "%d is too high. Try again.\n", userGuess );
53         else
54         {
55            displayMessage();
56            // new game
57            newGame();
58         } // end else
59      } // end method checkUserGuess
60
61      // print a message based on the number of tries
62      public void displayMessage()
63      {
64         System.out.printf( "You guessed the number in %d tries\n",
```

```
65           guesses );
66
67        if ( guesses < 10 )
68           System.out.println(
69              "Either you know the secret or you got lucky!\n" );
70        else if ( guesses == 10 )
71          System.out.println( "Ahah! You know the secret!\n" );
72        else
73           System.out.println( "You should be able to do better!\n" );
74     } // end method displayMessage
75  } // end class Guess2
```

```
1  // Exercise 6.34 Solution: PlayGuess2.java
2  // Application to play a game of Guess2 the number
3  public class PlayGuess2
4  {
5     public static void main( String args[] )
6     {
7        Guess2 application = new Guess2();
8        application.play();
9     } // end main
10 } // end class PlayGuess2
```

```
Guess a number between 1 and 1000
Guess (0 to exit): 500
500 is too low. Try again.

.
.
.

Guess (0 to exit): 890
You guessed the number in 9 tries
Either you know the secret or you got lucky!

Guess a number between 1 and 1000
Guess (0 to exit): 0
```

6.35 Exercise 6.30 through Exercise 6.32 developed a computer-assisted instruction program to teach an elementary school student multiplication. Perform the following enhancements:

a) Modify the program to allow the user to enter a school grade-level capability. A grade level of 1 means that the program should use only single-digit numbers in the problems, a grade level of 2 means that the program should use numbers as large as two digits, and so on.

ANS:

```
1  // Exercise 6.35 Part A Solution: Multiply4.java
2  // Program generates single-digit multiplication problems
3  import java.util.*;
4
```

```
5  public class Multiply4
6  {
7     Random randomNumbers = new Random();
8
9     int answer; // the correct answer
10    int count; // number of questions answered
11    int correct; // number of correct answers
12    int grade; // the grade of the user
13
14    // ask the user to answer multiplication problems
15    public void quiz()
16    {
17       Scanner input = new Scanner( System.in );
18
19       int guess; // the user's guess
20
21       // prompts the user to enter a grade level
22       do
23       {
24          System.out.print( "Enter the grade (1 or 2) of the user: " );
25          grade = input.nextInt();
26       } while ( ( grade != 1 ) && ( grade != 2 ) );
27
28
29       createQuestion(); // display the first question
30
31       System.out.println( "Enter your answer (-1 to exit):" );
32       guess = input.nextInt();
33
34       while ( guess != -1 )
35       {
36          checkResponse( guess );
37
38          System.out.println( "Enter your answer (-1 to exit):" );
39          guess = input.nextInt();
40       } // end while
41    } // end method
42
43    // prints a new question and stores the corresponding answer
44    public void createQuestion()
45    {
46       int range = 10; // the range of possible numbers
47       if ( grade == 2 )
48          range = 100;
49
50       // get two random numbers less than range
51       int digit1 = randomNumbers.nextInt( range );
52       int digit2 = randomNumbers.nextInt( range );
53
54       answer = digit1 * digit2;
55       System.out.printf( "How much is %d times %d?\n",
56          digit1, digit2 );
57    } // end method createQuestion
58
```

```
59    // create a new response
60    public String createResponse( boolean correct )
61    {
62       if ( correct )
63          switch ( randomNumbers.nextInt( 4 ) )
64          {
65             case 0:
66                return( "Very good!" );
67
68             case 1:
69                return( "Excellent!" );
70
71             case 2:
72                return( "Nice work!" );
73
74             case 3:
75                return( "Keep up the good work!" );
76          } // end switch
77
78       // otherwise, assume incorrect
79       switch ( randomNumbers.nextInt( 4 ) )
80       {
81          case 0:
82             return( "No. Please try again." );
83
84          case 1:
85             return( "Wrong. Try once more." );
86
87          case 2:
88             return( "Don't give up!" );
89
90          case 3: default:
91             return( "No. Keep trying." );
92       } // end switch
93    } // end method createResponse
94
95    // determine how the user is faring
96    public double calculatePercentage()
97    {
98       return ( double ) correct / count;
99    } // end method calculatePercentage
100
101   // checks if the user answered correctly
102   public void checkResponse( int guess )
103   {
104      if ( guess != answer )
105         System.out.println( createResponse( false ) );
106      else
107      {
108         correct++;
109         System.out.println( createResponse( true ) );
110         createQuestion();
111      } // end else
112
```

```
113         count++;
114
115         if ( count >= 10 )
116         {
117            System.out.printf( "You scored a %d\n",
118               ( int ) ( calculatePercentage()*100 ) );
119
120            if ( calculatePercentage() < 0.75 )
121               System.out.println(
122                  "Please ask your instructor for extra help." );
123
124            // start over
125            System.out.println();
126            correct = 0;
127            count = 0;
128            createQuestion();
129         } // end if
130      } // end method checkResponse
131   } // end class Multiply4
```

```
1  // Exercise 6.35 Part A Solution: Multiply4Test.java
2  // Test application for class Multiply4
3  public class Multiply4Test
4  {
5     public static void main( String args[] )
6     {
7        Multiply4 application = new Multiply4();
8        application.quiz();
9     } // end main
10 } // end class Multiply4Test
```

```
Enter the grade (1 or 2) of the user: 1
How much is 8 times 0?
Enter your answer (-1 to exit):
0
Nice work!
How much is 3 times 2?
Enter your answer (-1 to exit):
6
Nice work!
How much is 3 times 3?
Enter your answer (-1 to exit):
-1
```

```
Enter the grade (1 or 2) of the user: 2
How much is 28 times 38?
Enter your answer (-1 to exit):
1064
Nice work!
How much is 83 times 61?
Enter your answer (-1 to exit):
-1
```

b) Modify the program to allow the user to pick the type of arithmetic problems he or she wishes to study. An option of `1` means 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 problems of all these types.

ANS:

```
1  // Exercise 6.35 Part B Solution: Multiply5.java
2  // Program generates single-digit multiplication problems
3  import java.util.*;
4
5  public class Multiply5
6  {
7     Random randomNumbers = new Random();
8
9     int answer; // the correct answer
10    int count; // number of questions answered
11    int correct; // number of correct answers
12    int grade; // the grade of the user
13    int operationNum; // the operations to use
14
15    // ask the user to answer multiplication problems
16    public void quiz()
17    {
18       Scanner input = new Scanner( System.in );
19
20       int guess; // the user's guess
21
22       // prompts the user to enter a grade level
23       do
24       {
25          System.out.print( "Enter the grade (1 or 2) of the user: " );
26          grade = input.nextInt();
27       } while ( ( grade != 1 ) && ( grade != 2 ) );
28
29       // prompts the user to enter an operation level
30       do
31       {
32          System.out.println( "1 = addition" );
33          System.out.println( "2 = subtraction" );
34          System.out.println( "3 = mutiplication" );
35          System.out.println( "4 = division" );
36          System.out.println( "5 = mixed operations" );
37          System.out.print( "Enter the operation (1 to 5): " );
```

```
38            operationNum = input.nextInt();
39         } while ( ( operationNum < 1 ) || ( operationNum > 5 ) );
40
41         createQuestion(); // display the first question
42
43         System.out.println( "Enter your answer (-1 to exit):" );
44         guess = input.nextInt();
45
46         while ( guess != -1 )
47         {
48            checkResponse( guess );
49
50            System.out.println( "Enter your answer (-1 to exit):" );
51            guess = input.nextInt();
52         } // end while
53      } // end method
54
55      // prints a new question and stores the corresponding answer
56      public void createQuestion()
57      {
58         int range = 10; // the range of possible numbers
59         if ( grade == 2 )
60            range = 100;
61
62         // get two random numbers less than range
63         int digit1 = randomNumbers.nextInt( range );
64         int digit2 = randomNumbers.nextInt( range );
65
66         // generate the appropriate answer and operation
67         int op = operationNum; // the operation number
68         String operation = ""; // string corresponding to the operation
69
70         if ( op == 5 ) // random operation
71            op = 1 + randomNumbers.nextInt( 4 );
72
73         switch ( op )
74         {
75            case 1:
76               operation = "plus";
77               answer = digit1 + digit2;
78               break;
79
80            case 2:
81               // don't use negatives so simply swap the two
82               if ( digit1 < digit2 )
83               {
84                  int temp = digit1;
85                  digit1 = digit2;
86                  digit2 = temp;
87               } // end if
88
89               operation = "minus";
90               answer = digit1 - digit2;
91               break;
```

```
92
93             case 3:
94                operation = "times";
95                answer = digit1 * digit2;
96                break;
97
98             case 4:
99                if ( digit2 == 0 )
100                   digit2 = 1;
101               operation =  "divided by";
102               answer = digit1 / digit2;
103               break;
104         } // end switch
105
106         System.out.printf( "How much is %d %s %d?\n",
107            digit1, operation, digit2 );
108      } // end method createQuestion
109
110      // create a new response
111      public String createResponse( boolean correct )
112      {
113         if ( correct )
114            switch ( randomNumbers.nextInt( 4 ) )
115            {
116               case 0:
117                  return( "Very Good!" );
118
119               case 1:
120                  return( "Excellent!" );
121
122               case 2:
123                  return( "Nice work!" );
124
125               case 3:
126                  return( "Keep up the good work!" );
127            } // end switch
128
129         // otherwise, assume incorrect
130         switch ( randomNumbers.nextInt( 4 ) )
131         {
132            case 0:
133               return( "No. Please try again." );
134
135            case 1:
136               return( "Wrong. Try once more." );
137
138            case 2:
139               return( "Don't give up!" );
140
141            case 3: default:
142               return( "No. Keep trying." );
143         } // end switch
144      } // end method createResponse
145
```

```
146    // determine how the user is faring
147    public double calculatePercentage()
148    {
149       return ( double ) correct / count;
150    } // end method calculatePercentage
151
152    // checks if the user answered correctly
153    public void checkResponse( int guess )
154    {
155       if ( guess != answer )
156          System.out.println( createResponse( false ) );
157       else
158       {
159          correct++;
160          System.out.println( createResponse( true ) );
161          createQuestion();
162       } // end else
163
164       count++;
165
166       if ( count >= 10 )
167       {
168          System.out.printf( "You scored a %d\n",
169             ( int ) ( calculatePercentage()*100 ) );
170
171          if ( calculatePercentage() < 0.75 )
172             System.out.println(
173                "Please ask your instructor for extra help." );
174
175          // start over
176          System.out.println();
177          correct = 0;
178          count = 0;
179          createQuestion();
180       } // end if
181    } // end method checkResponse
182 } // end class Multiply5
```

```
1  // Exercise 6.35 Part B Solution: Multiply5Test.java
2  // Test application for class Multiply5
3  public class Multiply5Test
4  {
5     public static void main( String args[] )
6     {
7        Multiply5 application = new Multiply5();
8        application.quiz();
9     } // end main
10 } // end class Multiply5Test
```

```
Enter the grade (1 or 2) of the user: 1
1 = addition
2 = subtraction
3 = mutiplication
4 = division
5 = mixed operations
Enter the operation (1 to 5): 5
How much is 0 times 1?
Enter your answer (-1 to exit):
0
Keep up the good work!
How much is 1 times 5?
Enter your answer (-1 to exit):
5
Excellent!
How much is 9 plus 6?
Enter your answer (-1 to exit):
54
Don't give up!
Enter your answer (-1 to exit):
15
Keep up the good work!
How much is 3 plus 3?
Enter your answer (-1 to exit):
6
Nice work!
How much is 9 minus 2?
Enter your answer (-1 to exit):
7
Very good!
How much is 4 plus 9?
Enter your answer (-1 to exit):
13
Very good!
How much is 9 divided by 7?
Enter your answer (-1 to exit):
1
Nice work!
How much is 4 divided by 5?
Enter your answer (-1 to exit):
0
Excellent!
How much is 3 times 2?
Enter your answer (-1 to exit):
6
Excellent!
You scored a 90

How much is 5 times 2?
Enter your answer (-1 to exit):
-1
```

```
Enter the grade (1 or 2) of the user: 2
1 = addition
2 = subtraction
3 = mutiplication
4 = division
5 = mixed operations
Enter the operation (1 to 5): 1
How much is 56 plus 87?
Enter your answer (-1 to exit):
143
Very good!
How much is 90 plus 18?
Enter your answer (-1 to exit):
118
No. Keep trying.
Enter your answer (-1 to exit):
108
Very good!
How much is 19 plus 71?
Enter your answer (-1 to exit):
-1
```

6.36 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.

ANS:

```
1  // Exercise 6.36 Solution: Points.java
2  // Program calculates the distance between two points.
3  import java.util.Scanner;
4
5  public class Points
6  {
7     // calculates the distance between two points
8     public void calculateDistance()
9     {
10       Scanner input = new Scanner( System.in );
11
12       System.out.printf( "%s\n   %s\n   %s\n",
13          "Type the end-of-file indicator to terminate",
14          "On UNIX/Linux/Mac OS X type <ctrl> d then press Enter",
15          "On Windows type <ctrl> z then press Enter" );
16       System.out.print( "Or Enter X1: " );
17
18       // continually read in inputs until the user terminates
19       while ( input.hasNext() )
20       {
21          double x1 = input.nextDouble();
22          System.out.print( "Enter Y1: " );
23          double y1 = input.nextDouble();
24          System.out.print( "Enter X2: " );
25          double x2 = input.nextDouble();
```

```
26           System.out.print( "Enter Y2: " );
27           double y2 = input.nextDouble();
28
29           double distance = distance( x1, y1, x2, y2 );
30           System.out.printf( "Distance is %f\n\n", distance );
31
32           System.out.printf( "%s\n   %s\n   %s\n",
33              "Type the end-of-file indicator to terminate",
34              "On UNIX/Linux/Mac OS X type <ctrl> d then press Enter",
35              "On Windows type <ctrl> z then press Enter" );
36           System.out.print( "Or Enter X1: " );
37        } // end while
38     } // end method calculateDistance
39
40     // calculate distance between two points
41     public double distance( double x1, double y1, double x2, double y2 )
42     {
43        return Math.sqrt( Math.pow( ( x1 - x2 ), 2 ) +
44           Math.pow( ( y1 - y2 ), 2 ) );
45     } // end method distance
46  } // end class Points
```

```
1  // Exercise 6.36 Solution: PointsTest.java
2  // Test application for class Points
3  public class PointsTest
4  {
5     public static void main( String args[] )
6     {
7        Points application = new Points();
8        application.calculateDistance();
9     } // end main
10 } // end class PointsTest
```

```
Type the end-of-file indicator to terminate
   On UNIX/Linux/Mac OS X type <ctrl> d then press Enter
   On Windows type <ctrl> z then press Enter
Or Enter X1: 1
Enter Y1: 1
Enter X2: 4
Enter Y2: 5
Distance is 5.000000

Type the end-of-file indicator to terminate
   On UNIX/Linux/Mac OS X type <ctrl> d then press Enter
   On Windows type <ctrl> z then press Enter
Or Enter X1: ^Z
```

6.37 Modify the craps program of Fig. 6.9 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 is not, have the user reenter `wager` until a valid `wager` is entered. After a correct `wager` is entered, run one game of craps. If the player wins, increase `bankBalance` by `wager` and display the new `bankBalance`. If the player loses, decrease `bankBalance` 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.

ANS:

```
1  // Exercise 6.37: Craps.java
2  // Craps class simulates the dice game Craps.
3  // allows the user to place bets on games
4  import java.util.*;
5
6  public class Craps
7  {
8     // create random number generator for use in method rollDice
9     private Random randomNumbers = new Random();
10
11    int balance; // the current balance
12    int wager; // the current wager
13
14    // enumeration with constants that represent the game status
15    private enum Status { CONTINUE, WON, LOST };
16
17    // allows the user to bet on games of Craps
18    public void bet()
19    {
20       Scanner input = new Scanner( System.in );
21
22       balance = 1000; // start the user off with 1000
23
24       do
25       {
26          // prompt the user for a wager
27          System.out.printf( "Current balance is %d\n", balance );
28          System.out.print( "Enter wager (-1 to quit): " );
29          wager = input.nextInt();
30
31          if ( wager >= 0 )
32          {
33             if ( wager > balance )
34                System.out.println( "You don't have enough money!" );
35             else
36             {
37                System.out.println ( chatter() );
38
39                play(); // play a game
40
41                if ( balance <= 0 )
42                   System.out.println( "Sorry. You busted!" );
43             } // end else
44
45             // reset the wager
46             wager = 0;
47             System.out.println();
48          } // end if
```

```
49          // terminate if the user quits or runs out of money
50          } while ( ( wager != -1 ) && ( balance > 0 ) );
51       } // end method newGame
52
53       // plays one game of craps
54       public void play()
55       {
56          int sumOfDice = 0; // sum of the dice
57          int myPoint = 0; // point if no win or loss on first roll
58
59          Status gameStatus; // can contain CONTINUE, WON or LOST
60
61          sumOfDice = rollDice(); // first roll of the dice
62
63          // determine game status and point based on sumOfDice
64          switch ( sumOfDice )
65          {
66             case 7: // win with 7 on first roll
67             case 11: // win with 11 on first roll
68                gameStatus = Status.WON;
69                break;
70             case 2: // lose with 2 on first roll
71             case 3: // lose with 3 on first roll
72             case 12: // lose with 12 on first roll
73                gameStatus = Status.LOST;
74                break;
75             default: // did not win or lose, so remember point
76                gameStatus = Status.CONTINUE; // game is not over
77                myPoint = sumOfDice; // store the point
78                System.out.printf( "Point is %d\n", myPoint );
79                break; // optional for default case at end of switch
80          } // end switch
81
82          // while game is not complete ...
83          while ( gameStatus == Status.CONTINUE )
84          {
85             sumOfDice = rollDice(); // roll dice again
86
87             // determine game status
88             if ( sumOfDice == myPoint ) // win by making point
89                gameStatus = Status.WON;
90             else
91                if ( sumOfDice == 7 ) // lose by rolling 7
92                   gameStatus = Status.LOST;
93          } // end while
94
95          // display won or lost message and change the balance
96          if ( gameStatus == Status.WON )
97          {
98             System.out.println( "Player wins" );
99             balance += wager;
100         } // end if
101         else
102         {
```

```
103          System.out.println( "Player loses" );
104          balance -= wager;
105       } // end else
106    } // end method play
107
108    // roll dice, calculate sum and display results
109    public int rollDice()
110    {
111       // pick random die values
112       int die1 = 1 + randomNumbers.nextInt( 6 );
113       int die2 = 1 + randomNumbers.nextInt( 6 );
114
115       int sum = die1 + die2; // sum die values
116
117       // display results of this roll
118       System.out.printf( "Player rolled %d + %d = %d\n",
119          die1, die2, sum );
120
121       return sum; // return sum of dice
122    } // end method rollDice
123
124    // randomly chooses a phrase to respond to the player's action
125    public String chatter()
126    {
127       switch ( randomNumbers.nextInt( 5 ) )
128       {
129          case 0:
130             return "Oh, you're going for broke huh?";
131
132          case 1:
133             return "Aw cmon, take a chance!";
134
135          case 2:
136             return
137                "You're up big. Now's the time to cash in your chips!";
138
139          case 3:
140             return
141                "You're way too lucky! I think you're a cheat!!!";
142
143          default:
144             return "I'm betting all my money on you.";
145       } // end switch
146    } // end method chatter
147 } // end class Craps
```

```
1  // Exercise 6.37 Solution: PlayCraps.java
2  // Application to play a game of Craps
3  public class PlayCraps
4  {
5     public static void main( String args[] )
6     {
7        Craps application = new Craps();
```

```
        application.bet();
     } // end main
  } // end class PlayCraps
```

```
Current balance is 1000
Enter wager (-1 to quit): 500
Aw cmon, take a chance!
Player rolled 6 + 1 = 7
Player wins

Current balance is 1500
Enter wager (-1 to quit): 750
Oh, you're going for broke huh?
Player rolled 2 + 1 = 3
Player loses

Current balance is 750
Enter wager (-1 to quit): 750
Aw cmon, take a chance!
Player rolled 3 + 6 = 9
Point is 9
Player rolled 4 + 6 = 10
Player rolled 2 + 3 = 5
Player rolled 2 + 1 = 3
Player rolled 3 + 2 = 5
Player rolled 3 + 1 = 4
Player rolled 1 + 2 = 3
Player rolled 5 + 3 = 8
Player rolled 1 + 4 = 5
Player rolled 4 + 3 = 7
Player loses
Sorry. You busted!
```

6.38 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 are not familiar with these number systems, read Appendix E first.

ANS:

```
// Exercise 6.38 Solution: NumberSystem.java
// Converting a decimal number to binary, octal and hexadecimal.
public class NumberSystem
{
   // displays conversions in binary, octal, and hexadecimal
   public void displayConversions()
   {
      System.out.printf( "%-8s%-12s%-8s%-8s\n",
         "Decimal", "Binary", "Octal", "Hexadecimal" );

      // print out binary, octal and hexadecimal representation
      // for each number
      for ( int i = 1; i <= 256; i++ )
      {
         String binary = convertToBinary( i ); // binary representation
```

```
16            String octal = convertToOctal( i ); // octal representation
17            String hexadecimal = convertToHex( i ); // hex representaion
18
19            System.out.printf( "%-8d%-12s%-8s%-8s\n",
20               i, binary, octal, hexadecimal );
21         } // end for loop
22      } // end method displayConversions
23
24      // returns a String represention of the decimal number in binary
25      public String convertToBinary( int decimal )
26      {
27         String binary = "";
28
29         while ( decimal >= 1 )
30         {
31            int value = decimal % 2;
32            binary = value + binary;
33            decimal /= 2;
34         } // end binary while loop
35
36         return binary;
37      } // end method convertToBinary
38
39      // returns a String represention of the number in octal
40      public String convertToOctal( int decimal )
41      {
42         String octal = "";
43
44         // get octal representation
45         while ( decimal >= 1 )
46         {
47            int value = decimal % 8;
48            octal = value + octal;
49            decimal /= 8;
50         } // end octal while loop
51
52         return octal;
53      } // end method convertToOctal
54
55      // returns a String represention of the number in hexadecimal
56      public String convertToHex( int decimal )
57      {
58         String hexadecimal = "";
59
60         // get hexadecimal representation
61         while ( decimal >= 1 )
62         {
63            int value = decimal % 16;
64
65            switch ( value )
66            {
67               case 10:
68                  hexadecimal = "A" + hexadecimal;
69                  break;
```

```
70
71             case 11:
72                hexadecimal = "B" + hexadecimal;
73                break;
74
75             case 12:
76                hexadecimal = "C" + hexadecimal;
77                break;
78
79             case 13:
80                hexadecimal = "D" + hexadecimal;
81                break;
82
83             case 14:
84                hexadecimal = "E" + hexadecimal;
85                break;
86
87             case 15:
88                hexadecimal = "F" + hexadecimal;
89                break;
90
91             default:
92                hexadecimal = value + hexadecimal;
93                break;
94          } // end switch
95
96          decimal /= 16;
97       } // end hexadecimal while loop
98
99       return hexadecimal;
100   } // end method convertToHex
101 } // end class NumberSystem
```

```
1  // Exercise 6.38 Solution: NumberSystemTest.java
2  // Test application for class NumberSystem
3  public class NumberSystemTest
4  {
5     public static void main(String[] args)
6     {
7        NumberSystem application = new NumberSystem();
8        application.displayConversions();
9     } // end main
10 } // end class NumberSystemTest
```

```
Decimal Binary      Octal   Hexadecimal
1       1           1       1
2       10          2       2
3       11          3       3
4       100         4       4

.
.
.

252     11111100    374     FC
253     11111101    375     FD
254     11111110    376     FE
255     11111111    377     FF
256     100000000   400     100
```

(Optional) GUI and Graphics Case Study

6.1 Using method `fillOval`, draw a bull's-eye that alternates between two random colors, as in Fig. 6.27. Use the constructor `Color( int r, int g, int b )` with random arguments to generate random colors.

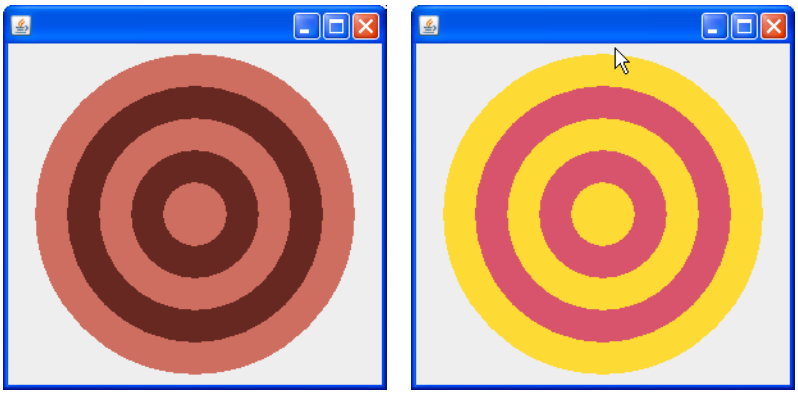


Fig. 6.27 | A bull's-eye with two alternating, random colors.

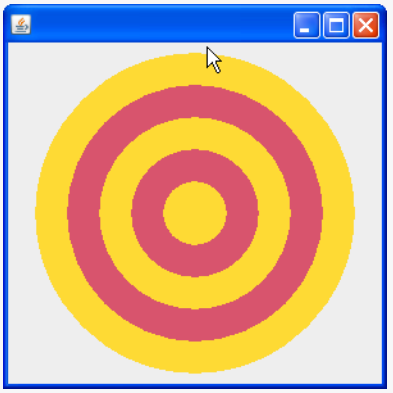
ANS:

```
1  // GCS Exercise 6.1: AlternateColors.java
2  // Demonstrates colors and filled shapes.
3  import java.awt.Color;
4  import java.awt.Graphics;
5  import java.util.Random;
6  import javax.swing.JPanel;
7
8  public class AlternateColors extends JPanel
9  {
10    private Random randomNumbers = new Random();
```

```
   private boolean colorChoice; // flag for which color to use
   private Color color1; // first color
   private Color color2; // second color

   // no-argument constructor
   public AlternateColors()
   {
      // randomly generate two colors
      color1 = new Color( randomNumbers.nextInt( 256 ),
         randomNumbers.nextInt( 256 ), randomNumbers.nextInt( 256 ) );
      color2 = new Color( randomNumbers.nextInt( 256 ),
         randomNumbers.nextInt( 256 ), randomNumbers.nextInt( 256 ) );
   } // end AlternateColors constructor

   // draws a "bull's-eye" with two alternating colors
   public void paintComponent( Graphics g )
   {
      super.paintComponent( g );

      int circles = 5; // number of circles
      int radius = 25; // radius of a circle

      // find the middle of the panel
      int centerX = getWidth() / 2;
      int centerY = getHeight() / 2;

      colorChoice = true; // Set the first color to be used

      // draws circles starting with the outermost
      for ( int counter = circles; counter > 0; counter-- )
      {
         // set the colors based on the current color choice
         if ( colorChoice )
            g.setColor( color1 );
         else
            g.setColor( color2 );

         colorChoice = !colorChoice; // flip the choice of colors

         // draw the oval
         g.fillOval( centerX - counter * radius,
            centerY - counter * radius,
            counter * radius * 2, counter * radius * 2 );
      } // end for
   } // end method paintComponent
} // end class AlternateColors
```

```
// GCS Exercise 6.1: AlternateColorsTest.java
// Test application that displays class AlternateColors.
import javax.swing.JFrame;

public class AlternateColorsTest
{
```

```
7      public static void main( String args[] )
8      {
9         AlternateColors panel = new AlternateColors();
10        JFrame application = new JFrame();
11
12        application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
13        application.add( panel );
14        application.setSize( 300, 300 );
15        application.setVisible( true );
16     } // end main
17  } // end class AlternateColorsTest
```



6.2 Create a program that draws 10 random filled shapes in random colors, positions and sizes (Fig. 6.19). `Method paintComponent` should contain a loop that iterates 10 times. In each iteration, the loop should determine whether to draw a filled rectangle or an oval, create a random color and choose coordinates and dimensions at random. The coordinates should be chosen based on the panel's width and height. Lengths of sides should be limited to half the width or height of the window.

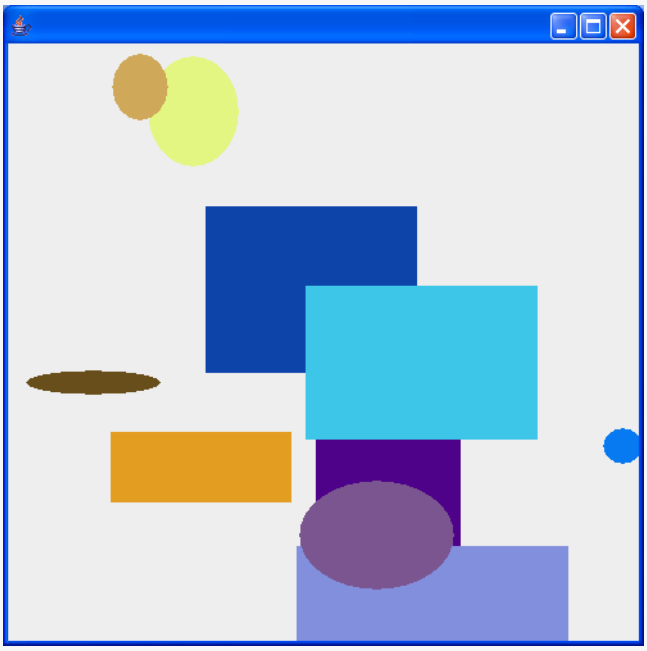


Fig. 6.28 | Randomly generated shapes.

ANS:

```
1  // GCS Exercise 6.2: DrawPanel.java
2  // Demonstrates drawing random shapes and random colors.
3  import java.awt.Color;
4  import java.awt.Graphics;
5  import java.util.Random;
6  import javax.swing.JPanel;
7
8  public class DrawPanel extends JPanel
9  {
10    private Random randomNumbers = new Random();
11
12    // draws random shapes in random colors
13    public void paintComponent( Graphics g )
14    {
15       int maxWidth = getWidth(); // get the width of the panel
16       int maxHeight = getHeight(); // get the height of the panel
17
```

```
18         // draw ten random shapes
19         for ( int i = 0; i < 10; i++ )
20         {
21            // generate a random color
22            Color color = new Color( randomNumbers.nextInt( 256 ),
23               randomNumbers.nextInt( 256 ),
24               randomNumbers.nextInt( 256 ) );
25            g.setColor( color ); // set the color to the random color
26
27            // pick a shape at random
28            switch ( randomNumbers.nextInt( 2 ) )
29            {
30               case 0: // draw a random rectangle
31                  g.drawRect( randomNumbers.nextInt( maxWidth ) + 1,
32                     randomNumbers.nextInt( maxHeight ) + 1,
33                     randomNumbers.nextInt( maxWidth / 2 ) + 1,
34                     randomNumbers.nextInt( maxHeight / 2 ) + 1 );
35                  break;
36               case 1: // draw a random oval
37                  g.drawOval( randomNumbers.nextInt( maxWidth ) + 1,
38                     randomNumbers.nextInt( maxHeight ) + 1,
39                     randomNumbers.nextInt( maxWidth / 2 ) + 1,
40                     randomNumbers.nextInt( maxHeight / 2 ) + 1 );
41                  break;
42            } // end switch
43         } // end for
44      } // end method paintComponent
45   } // end class DrawPanel
```

```
1  // GCS Exercise 6.2: DrawPanelTest.java
2  // Test application that displays class DrawPanel.
3  import javax.swing.JFrame;
4
5  public class DrawPanelTest
6  {
7     public static void main( String args[] )
8     {
9        DrawPanel panel = new DrawPanel();
10       JFrame application = new JFrame();
11
12       application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
13       application.add( panel );
14       application.setSize( 500, 500 );
15       application.setVisible( true );
16    } // end main
17 } // end class DrawPanelTest
```