Solutions to Chapter Exercises

4

Defining Instantiable Classes

4.1. Consider the following instantiable class.

```
class QuestionOne
{
   public   final int  A = 345;
   public   int        b;
   private  float      c;

   private void methodOne( int a)
   {
      b = a;
   }

   public float methodTwo( )
   {
      return 23;
   }
}
```

Identify invalid statements in the following main class. For each invalid statement, state why it is invalid.

```
class Q1Main
{
   public static void main( String[] args )
```

```
      {
          QuestionOne q1;
          q1 = new QuestionOne( );

1 ───►    q1.A = 12;
          q1.b = 12;
2 ───►    q1.c = 12;

3 ───►    q1.methodOne( 12 );
4 ───►    q1.methodOne( );
5 ───►    System.out.println( q1.methodTwo( 12 ) );
6 ───►    q1.c = q1.methodTwo( );
      }
  }
```

1. Data member A *is a constant.*

2. Data member c *is private.*

3. Wrong number of arguments; the method is private.

4. The method is private.

5. Wrong number of arguments.

6. Data member c *is private.*

4.2. What will be the output from the following code?

```
class Q2Main
{
    public static void main( String[] args )
    {
        QuestionTwo q2;
        q2 = new QuestionTwo( );
        q2.init();

        q2.increment();
        q2.increment();

        System.out.println( q2.getCount() );
    }
}

class QuestionTwo
{
   private  int  count;

   public void init( )
   {
      count = 1;
   }

   public void increment( )
   {
      count = count + 1;
   }

   public int getCount( )
   {
      return count;
   }
}
```

3

4.3. What will be the output from the following code? Q3Main and QuestionThree classes are the slightly modified version of Q2Main and QuestionTwo.

```
class Q3Main
{
   public static void main( String[] args )
   {
      QuestionThree q3;
      q3 = new QuestionThree( );
      q3.init();

      q3.count = q3.increment() + q3.increment();

      System.out.println( q3.increment() );
   }
}

class QuestionThree
{
   public  int   count;
```

```
      public void init( )
      {
         count = 1;
      }

      public int increment( )
      {
         count = count + 1;
         return count;
      }
   }
```

```
6
```

4.4. Determine the output of the following program.

```
/*
   Program Question4

*/

import javabook.*;

class Question4
{
   private int x, y, z;
   private MainWindow mainWindow;
   private OutputBox  outputBox;

   public void start ( )
   {
      int    x, y;

      setup();

      x = y = 10;
      modify(x, y);

      printout();
   }


   private void setup( )
   {
      mainWindow = new MainWindow( );
      outputBox = new OutputBox(mainWindow);
      mainWindow.setVisible( true );
      outputBox.setVisible( true );

      x = 100;
      y = 200;
      z = 300;
   }

   private void modify( int x, int y)
   {
      z = x + y;
      x = z;
      y = 2 * z;
   }

   private void printout( )
   {
      outputBox.printLine("x = " + x);
      outputBox.printLine("y = " + y);
      outputBox.printLine("z = " + z);
   }

}
```

```
/* Main Class */
class Q4Main
{
   public static void main( String[ ] args )
   {
      Question4 q4;
      q4 = new Question4( );
      q4.start( );
   }
}
```

```
x = 100
y = 200
z = 20
```

4.5. Improve the following program Question5 by converting the class into two classes: the main Q5Main class and the instantiable Question5 class. Avoid duplicating the same code for computing the circumference of two circles. Define a private method in Question5 that accepts the radius of a circle as its parameter and returns the circumference of the circle.

```
/*
   Program Question5

*/
class Question5
{

   public static void main (String[ ] args )
   {
      double       radius ;
      double       circumference;

      MainWindow   mainWindow;
      OutputBox    outputBox;
      InputBox     inputBox;

      int          smallRadius, largeRadius;
      double       smallCircum, largeCircum;

      mainWindow   = new MainWindow( );
      outputBox    = new OutputBox(mainWindow);
      inputBox     = new InputBox(mainWindow);

      //compute circumference of a smaller circle
      smallRadius = inputBox.getDouble("Radius of smaller circle:");
      radius = smallRadius;
      smallCircum = 2 * Math.PI * radius;

      //compute circumference of a larger circle
      largeRadius = inputBox.getDouble("Radius of larger circle:");
      radius = largeRadius;
      largeCircum = 2 * Math.PI * radius;

      //Display the difference
      outputBox.printLine("Difference in circumference of two circles");
      outputBox.printLine("Circumference of smaller circle: " + smallCircum);
      outputBox.skipLine(2);
      outputBox.printLine("Circumference of larger circle: " + largeCircum);
      ouputBox.skipLine(2);
      outputBox.printLine("Difference: " + (largeCircum - smallCircum));
   }

}
```

```
import javabook.*;

class Question5
{
   MainWindow   mainWindow;
   OutputBox    outputBox;
   InputBox     inputBox;
```

```
    int           smallRadius, largeRadius;
    double        smallCircum, largeCircum;

    public Question5()
    {
       mainWindow   = new MainWindow( );
       outputBox    = new OutputBox(mainWindow);
       inputBox     = new InputBox(mainWindow);

       mainWindow.setVisible( true );
       outputBox.setVisible( true );
    }

    public void start()
    {
       //compute the circumference of a smaller circle
       smallRadius = inputBox.getInteger("Radius of smaller circle:");
       smallCircum = computeCircumference(smallRadius);

       //compute the circumference of a larger circle
       largeRadius = inputBox.getInteger("Radius of larger circle:");
       largeCircum = computeCircumference(largeRadius);

       //Display the difference
       outputBox.printLine("Difference in circumference of two circles");
       outputBox.printLine("Circumference of smaller circle: " + smallCircum);
       outputBox.skipLine(2);
       outputBox.printLine("Circumference of larger circle: " + largeCircum);
       outputBox.skipLine(2);
       outputBox.printLine("Differnce: " + (largeCircum - smallCircum));
    }

    private double computerCircumference( float radius )
    {
       return (2 * Math.PI * radius);
    }
}

class Exercise4_5
{
    public static void main ( String[] args )
    {
        Question5 q5;
        q5 = new Question5();
        q5.start();
    }
}
```

4.6. Write an application that converts centimeters (input) to feet and inches (output). Use InputBox for input and OutputBox for output. 1 inch = 2.54 centimeters. Implement this application by defining and using an instantiable class whose instance is capable of converting metric measurements to U.S. units and vice versa. Some of the possible public methods are

```
public double toFeet      ( double centimeter ) { ... }
public double toInches    ( double centimeter ) { ... }
public double toCentimeter( int    feet,
                            int    inches     ) { ... }
public double toCentimeter( double yard       ) { ... }
```

See Exercise4_6.java.

4.7. Write an application that displays the recommended weight given the user's age and height. The formula for calculating the recommended weight is

```
recommendedWeight = (height - 100 + age % 10) * 0.90
```

See Exercise4_7.java.

4.8. Write an application that computes the total ticket sales of a concert. There are three types of seatings: A, B, and C. The program accepts the number of tickets sold and the price of a ticket for each of the three types of seats. The total sales are computed as

```
totalSales = numberOfA_Seats * pricePerA_Seat +
             numberOfB_Seats * pricePerB_Seat +
             numberOfC_Seats * pricePerC_Seat;
```

Write this application using only one class, the main class of the program.

See Exercise4_8.java.

4.9. Redo exercise 8 by using an instantiable SeatType class. An instance of the SeatType class keeps track of the ticket price for a given type of seat (A, B, or C).

See Exercise4_9.java.

4.10. Write an application that accepts the unit weight of a bag of coffee in pounds and the number of bags sold and displays the total price of the sale, computed as

```
totalPrice        = unitWeight * numberOfUnits * 5.99;
totalPriceWithTax = totalPrice + totalPrice * 0.0725;
```

where 5.99 is the cost per pound and 0.0725 the sales tax. Display the result in the following manner:

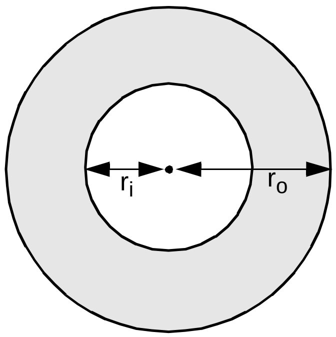
```
Number of bags sold:  32
     Weight per bag:  5 lbs
    Price per pound:  $5.99
          Sales tax:  7.25%

        Total price: $ 1027.884
```

You don't have to worry about formatting.

See Exercise4_10.java.

4.11. Write an application that computes the area of a circular region (the shaded area in the diagram) given the radius of the inner and the outer circles, r_i and r_o, respectively.



We compute the area of the circular region by subtracting the area of the inner circle from the area of the outer circle. Define an instantiable Circle class that has methods to compute the area and circumference. You set the circle's radius with the setRadius method.

See Exercise4_11.java.

4.12. In Section 4.3, we modified the conversion methods of the CurrencyConverter class to include the fee deduction. With the modified methods, the sequence of calls such as

```
yenAmount
   = yenConverter.fromDollar( markConverter.toDollar(100) );
```

will result in charging the fee twice. Modify the class to eliminate this double-charging problem.

The best solution is to define a class that can carry out a direct conversion between any two currencies. However, such solution requires topics we have not covered yet. At this point, one possible solution is to separate the conversion and fee computation, instead of embedding the fee computation as a part of currency conversion.

See Exercise4_12.java.

4.13. Write an instantiable WeightConverter class. An instance of this class is created by passing the gravity of an object relative to the earth's gravity (see exercise 12 on page 132). For example, the moon's gravity is approximately 0.167 of the earth's gravity, so we create a WeightConverter instance for the moon as

```
WeightConverter  moonWeight;
moonWeight = new WeightConverter( 0.167 );
```

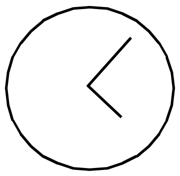
To compute how much you weigh on the moon, you pass your weight on earth to the convert method as

```
yourMoonWeight = moonWeight.convert( 160 );
```

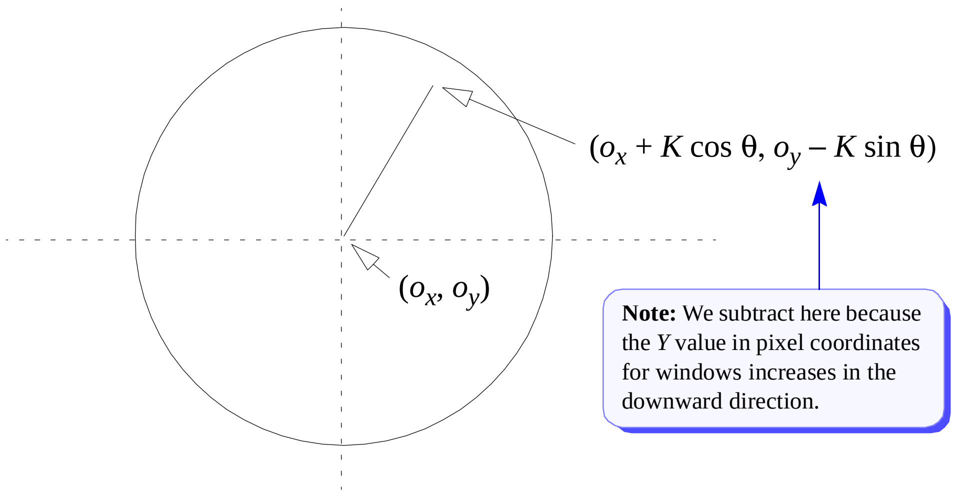
Define the WeightConverter class. Use this class and redo exercise 12 on page 132.

See Exercise4_7.java.

4.14. Write an application that teaches children how to read a clock. Use two InputBox objects to enter the hour and minute. The application program draws a clock that looks something like



To draw a clock hand, you use the drawLine method of the Graphics class. The endpoints of the line are determined as


$(o_x + K \cos \theta, o_y - K \sin \theta)$
(o_x, o_y)
Note: We subtract here because the *Y* value in pixel coordinates for windows increases in the downward direction.

The value for constant K determines the length of the clock hand. Make the K larger for the minute hand than for the hour hand. The angle θ is expressed in radians. The angle θ_{min} of the minute hand is computed as

$$(90 - \text{Minute} \times 6.0)\frac{\pi}{180}$$

and the angle θ_{hr} of the hour hand is computed as

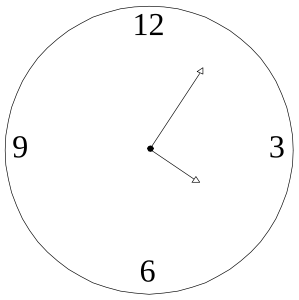
$$\left(90 - \left(\text{Hour} + \frac{\text{Minute}}{60.0}\right) \times 30.0\right)\frac{\pi}{180}$$

where Hour and Minute are input values. The values 6.0 and 30.0 designate the degrees for one minute and one hour (i.e., the minute hand moves 6 degrees in one minute and the hour hand moves 30.0 degrees in one hour). The factor $\frac{\pi}{180}$ converts a degree into the radian equivalent.
You can draw the clock on the main window by getting the window's Graphic object as

```
MainWindow mainWindow = new MainWindow();
mainWindow.setVisible( true );
Graphics graphic = mainWindow.getGraphics();
        //make sure mainWindow is visible on the screen
        //before calling its getGraphics method
...
graphic.drawOval(100, 100, 200, 200);
...
```

See Exercise4_14.java.

4.15. Extend the application in exercise 14 by drawing a more realistic, better looking clock. For example,



Visit Dr. Caffeine's homepage at http://www.drcaffeine.com and see the applet version of the clock.

4.16. In the Turtle exercises from the earlier chapters, we dealt with only one Turtle (e.g., see exercise 27 on page 82). It is possible, however, to let multiple turtles draw on a single drawing window. To associate multiple turtles to a single drawing, we create an instance of TurtleDrawingWindow and add turtles to it as in

```
TurtleDrawingWindow canvas = new TurtleDrawingWindow( );
Turtle winky, pinky, tinky;

winky = new Turtle( ); //create turtles
pinky = new Turtle( );
tinky = new Turtle( );

//now add turtles to the drawing window
canvas.add( winky );
canvas.add( pinky );
canvas.add( tinky );
```

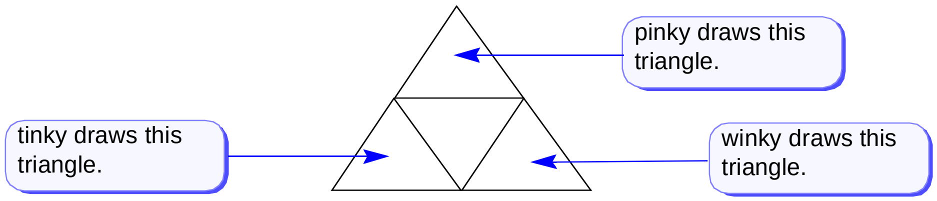
Ordinarily, when you start sending messages such as turn and move to a Turtle, it will begin moving immediately. When you have only one Turtle, this is fine. However, if you have multiple turtles and want them to start moving at the same time, you first have to pause them, give instructions, and then command them to start moving. Here's the basic idea:

```
winky.pause( );
pinky.pause( );
tinky.pause( );

//give instructions to turtles here,
//e.g. pinky.move(50); etc.

//now let the turtles start moving
winky.start( );
pinky.start( );
tinky.start( );
```

Using these Turtle objects, draw the following three triangles:


pinky draws this triangle.
tinky draws this triangle.
winky draws this triangle.

Use a different pen color for each triangle. Run the same program without pausing and describe what happens.