Homework 2

(Due: March 15, 2023)

The purpose of this homework is to get you started with programming in Java. For each question hand in a solution (i.e., program source code + program output), all zipped together into a single file.

Question 1: 10 points. Write a Java program that solves for all integer pairs $a, b \geq 0$,

$$\sqrt{a} + \sqrt{b} = \sqrt{n} \tag{1}$$

where n = 2023.

Hint: The prime factorization of 2023 is $7 \cdot 17 \cdot 17$. You should be able to use this fact to optimize your computation.

Question 2: 10 points. As shown in Figure 1 below, rectangles may be defined by the (x,y) coordinates of corner points that are diagonally opposite.

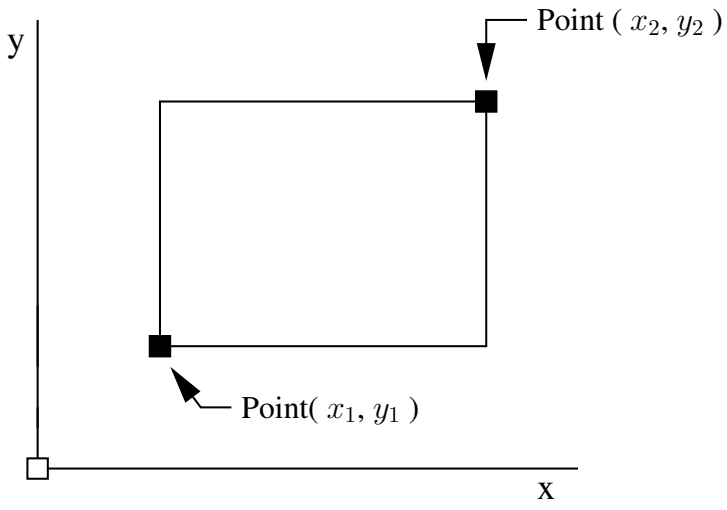

Figure 1: Definition of a rectangle via diagonally opposite corner points.

With this definition in place, the following script of code is a very basic implementation of a class for creating and working with rectangle objects.

```
/*
 *  ========================================================================
 *  Rectangle.java : A library of methods for creating and managing rectangles
 *
 *  double      area() -- returns the area of a rectangle
 *  double perimeter() -- returns the perimeter of a rectangle
 *
 *  Written By : Mark Austin                                    February 2023
 *  ========================================================================
 */

import java.lang.Math;

public class Rectangle {
   protected double dX1, dY1;  // Coordinate (x,y) for corner 1....
   protected double dX2, dY2;  // Coordinate (x,y) for corner 2....

   // Constructor methods ....

   public Rectangle() {}

   public Rectangle( double dX1, double dY1, double dX2, double dY2 ) {
      this.dX1     = dX1; this.dY1     = dY1;
      this.dX2     = dX2; this.dY2     = dY2;
   }

   // Convert rectangle details to a string ...

   public String toString() {
          return "Rectangle: Corner 1: (x,y) = " + "(" + dX1 + "," + dY1 + ")\n" +
                 "           Corner 2: (x,y) = " + "(" + dX2 + "," + dY2 + ")\n";
   }

   // ====================================
   // Compute rectangle area and perimeter
   // ====================================

   public double area() {
       return Math.abs( (dX2-dX1)*(dY2-dY1) );
   }

   public double perimeter() {
       return 2.0*Math.abs(dX2-dX1) + 2.0*Math.abs( dY2-dY1 );
   }

   // Exercise methods in the Rectangle class ....

   public static void main ( String args[] ) {

      System.out.println("Rectangle test program     ");
      System.out.println("==========================");

      // Setup and print details of a small rectangle....

      Rectangle rA = new Rectangle( 1.0, 1.0, 3.0, 4.0 );
```

```
      System.out.println( rA.toString() );

      // Print perimeter and area of the small rectangle....

      System.out.println( "Perimeter = " + rA.perimeter() );
      System.out.println( "Area      = " + rA.area() );
   }
}
```

The script of program output is as follows:

```
prompt >>
prompt >> java Rectangle
Rectangle test program
==========================
Rectangle: Corner 1: (x,y) = (1.0,1.0)
           Corner 2: (x,y) = (3.0,4.0)

Perimeter = 10.0
Area      = 6.0
prompt >> exit
```

The Rectangle class has methods to create objects (i.e, Rectangle), convert the details of a rectangle object into a string format (i.e., toString), and compute the rectangle area and perimeter (i.e., area() and permieter(), respectively). The implementation uses two pairs of doubles (dX1, dY1) and (dX2, dY2) to define the corner points.

Suppose that, instead, the corner points are defined via a Vertex class, where

```
public class Vertex {
    protected double dX, double dY

    ..... details of constructors and other methods removed ...
}
```

The appropriate modification for Rectangle is:

```
public class Rectangle {
   protected Vertex vertex1;  // First corner point....
   protected Vertex vertex2;  // Second corner point....

   ..... details rectangle removed ....
}
```

Fill in the missing details (i.e., constructors and toString() method) of class Vertex. Modify the code in Rectangle to use Vertex class. The resulting program should have essentially has the same functionality as the original version of Rectangle.

Hint. Your implementation should make use of the toString() method in Vertex.

Question 3: 10 points. There are lots of problems in engineering where the position of point needs to be evaluated with respect to a shape. Evaluation procedures can be phrased in terms of questions. For example, is the point inside (or outside) the shape?; Does the point lie on the boundary of the shape?; Does the point lie above/below the shape? Does the point lie to the left or right of the shape?; How far is the point from the perimeter?

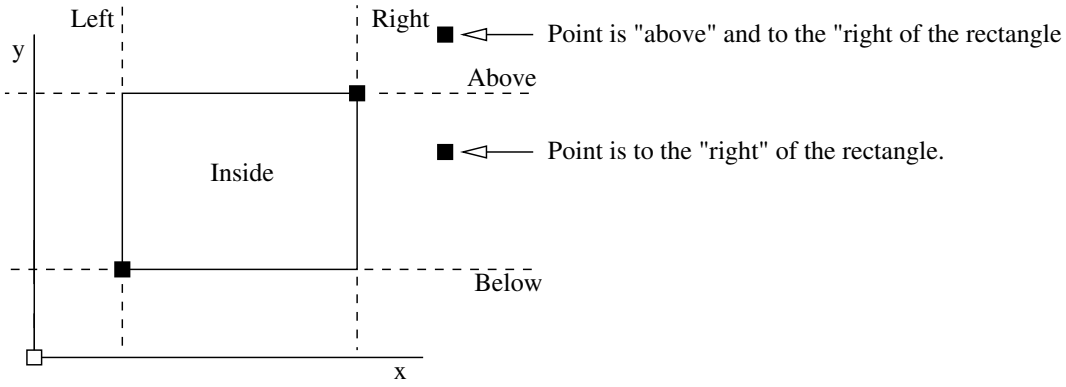

Figure 2: Classification of an (x,y) cooordinate relative to a rectangle

Figure 2 illustrates these ideas for one of the simplest cases possible, one point and a rectangle. Extend the functionality of the Rectangle class so that the possition of a point can be evaluated with respect to a specific rectangle object.

The appropriate method declarations are as follows:

```
public boolean isInside ( Vertex v ) { ... }
public boolean isOutside ( Vertex v ) { ... }
public boolean isOnPerimeter ( Vertex v ) { ... }
public boolean isAbove ( Vertex v ) { ... }
public boolean isBelow  ( Vertex v ) { ... }
public boolean isLeft ( Vertex v ) { ... }
public boolean isRight ( Vertex v ) { ... }
```

If the (x,y) coordinates of a vertex are inside a particular rectangle, then `isInside ()` should return `true`. Otherwise, it should return `false`. From Figure 2 is should evident that some points will result in multiple methods returning true. For example, points in the top right-hand side of the coordinate system will be outside, to the right, and above the rectangle.

Fill in the details of each method, and then develop a test program to exercise the procedures. Perhaps the most straight forward way of doing this is to write an extensive set of tests in the main() method for class Rectangle.

Question 4: 10 points. The left-hand side of Figure 3 shows the essential details of a domain familiar to many children. One by one, rectangular blocks are stacked as high as possible until they come tumbling down – the goal, afterall, is to create a spectacular crash!!

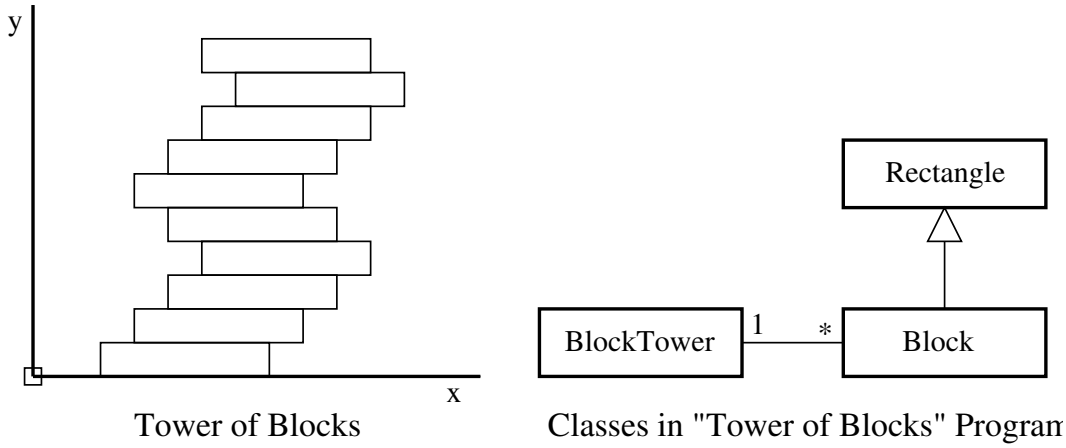

Figure 3: Schematic and classes for "Tower of Blocks"

Suppose that we wanted to model this process and use engineering principles to predict incipient instability of the block tower. Consider the following observations:

1. Rather than start from scratch, it would make sense to create a Block class that inherits the properties of Rectangle, and adds details relevant to engineering analysis (e.g., the density of the block).

2. Then we could develop a BlockTower class that systematically assembles the tower, starting at the base and working upwards. At each step of the tower assembly, analysis procedures should make sure that the tower is still stable.

The right-hand side of Figure 3 shows the relatioship among the classes. One BlockTower program (1) will employ many blocks, as indicated by the asterik (*).

Develop a Java program that builds upon the Rectangle class written in the previous questions. The class Block should store the density of the block (this will be important in determining its weight) and methods to compute the weight and centroid of each block. The BlockTower class will use block objects to build the tower. A straight forward way of modeling the block tower is with a list of block objects. After each block is added, the program should conduct a stability check. If the system is still stable, then add another block should be added. The simulation should cease when the tower of blocks eventually becomes unstable.

Note. To simplify the analysis, assume that adjacent blocks are firmly connected.

Stability Considerations. If the blocks are stacked perfectly on top of each other, then from a mathematical

standpoint the tower will never become unstable. In practice, this never happens. There is always a small offset and, eventually, it's the accumulation of offsets that leads to spectacular disaster.

For the purposes of this question, assume that blocks are five units wide and one unit high. When a new block is added, the block offset should be one unit. To make the question interesting, assume that four blocks are stacked with an offset to the right, then three blocks are added with an offset to the left, then four to the right, three to the left, and so forth. This sequence can be accomplished with the looping construct:

```
offset = Math.floor ((BlockNo - 1)/5.0) + (BlockNo-1)%5 );
if ((BlockNo-1)%5 == 4 ) offset = offset - 2;
```

The tower will become unstable when the center of gravity of blocks above a particular level falls outside the edge of the supporting block.