Practice Exercise #19: Overlapping Rectangles Version 3

http://www.comp.nus.edu.sg/~cs1020/4_misc/practice.html

Objectives:

1. Using **Point** class and **Math** class
2. Creating your own class
3. Problem solving

Task statement:

This is an extension of Practice Exercise #18.

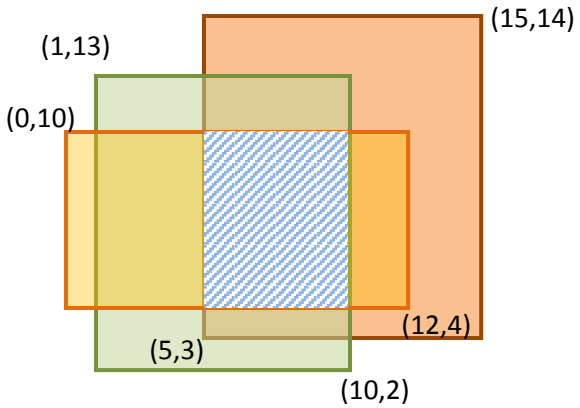
For this exercise, you should use the same **MyRect** class you wrote in Practice #18 without any modification. The class should contain two attributes **vertex1** and **vertex2** which are the two opposite vertices of a rectangle. Your should complete your **MyRect** class with the usual constructors, accessors, mutators, and overriding methods **toString()** and **equals()**.

We can represent a rectangle (whose sides are parallel to the *x*-axis or *y*-axis) with its two opposite vertices, where each vertex is a **Point** object. For example, a rectangle with vertices at (3, -1), (3, 6), (15, 6) and (15, -1) may be represented by any of these 4 pairs of points: (3, -1) and (15, 6); (15, 6) and (3, -1); (3, 6) and (15, -1); or (15, -1) and (3, 6).

You are to write a program **OverlapRectanglesV3.java** to read in an integer indicating the number of rectangles. You may assume that this value is at least 1.

Your program then reads in the values of the rectangles, and compute the overlap area of all the rectangles. (Hint: When you read the second rectangle, compute the overlap of it with the first rectangle. This overlap is itself a rectangle. When you read the third rectangle, compute the overlap of it with the previous overlap. Hence, each time a next rectangle is read, you determine the new overlap rectangle. After reading all the rectangles, you have the final overlap and you can then easily compute the area of this overlap.)

For example, given the 3 rectangles shown below (not drawn to scale), the final overlap, which is also a rectangle, has vertices at (5, 4) and (10, 10), with an area of 30.



Make sure you consider the *degenerate case* where two rectangles have no overlap, that is, the overlap is an "empty" rectangle. See sample run #2 for an example. How would you want to represent such an rectangle object in your program so that the final computation is correct?

Study the skeleton programs provided. You are to submit both **MyRect.java** and **OverlapRectanglesV3.java**.

Sample run #1:

```
How many rectangles? 3
Enter 2 opposite vertices of first rectangle: 1 13 10 2
Enter 2 opposite vertices of next rectangle: 15 14 5 3
Enter 2 opposite vertices of next rectangle: 12 4 0 10
Overlap rectangle: [(5, 4); (10, 10)]
Overlap area = 30
```

Sample run #2:

```
How many rectangles? 2
Enter 2 opposite vertices of first rectangle: 6 6 10 10
Enter 2 opposite vertices of next rectangle: 1 1 8 5
No overlap
```