CS1020 Take-home Lab #1
Exercise #3: Overlapping Rectangles

http://www.comp.nus.edu.sg/~cs1020/3_ca/labs.html

Objectives:

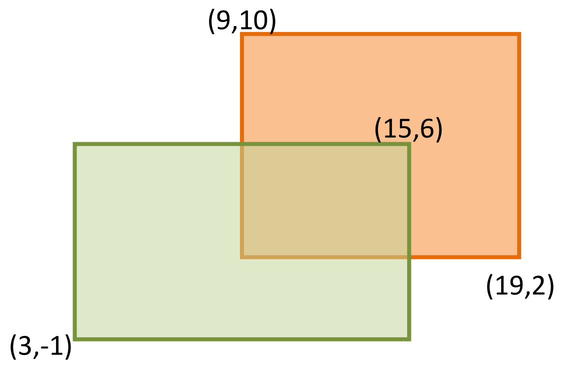
1. Using **Point** class and **Math** class
2. Writing user-defined methods
3. Problem solving

Task statement:

(Note that unless otherwise stated, you may assume that all input data are valid and hence there is no need for you to perform input data validation.)

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 **OverlapRectangles.java** to read in integer values for 4 points: the first 2 points are the opposite vertices of one rectangle, and the next 2 points are the opposite vertices of a second rectangle. The figure below shows one rectangle represented by (3, -1) and (15, 6), and another represented by (19, 2) and (9, 10). You are to use the **Point** class to create these 4 **Point** objects.



Your program should then call the method **overlapArea**(*a*, *b*, *c*, *d*) where *a* and *b* are the opposite vertices of the first rectangle, and *c* and *d* the opposite vertices of the second rectangle. (Of course, *a*, *b*, *c*, *d* should be replaced by more descriptive names in your program.) This method computes the overlap area of the two rectangles, which is **24** in our example.

Refer to the sample runs for the output format.

Number of submissions:

You are given **15** submissions. Only the final submission will be graded.

Sample run #1:

```
Enter 2 opposite vertices of 1st rectangle: 3 -1 15 6
Enter 2 opposite vertices of 2nd rectangle: 19 2 9 10
Overlap area = 24
```

Sample run #2:

```
Enter 2 opposite vertices of 1st rectangle: 15 6 3 -1
Enter 2 opposite vertices of 2nd rectangle: 9 2 19 10
Overlap area = 24
```

Sample run #3:

```
Enter 2 opposite vertices of 1st rectangle: -5 5 -1 1
Enter 2 opposite vertices of 2nd rectangle: 1 2 11 12
Overlap area = 0
```

Hints

This is actually a simplified version of an old CS1010 lab exercise, so it should not be considered hard for CS1020. What you need to do – as we have always emphasised in CS1010 – is to think about the logic and algorithm instead of jumping straight into coding. For this exercise, take out your pencil and paper and draw diagrams showing all the possible scenarios.

To help you out as this is your first take-home exercise, you might have observed that since a rectangle can be represented by any pair of opposite vertices, the logic to compute the overlap area becomes rather complex. It pays to rearrange the two vertices entered by the user in a way that subsequent computation can be simplified. Here, we suggest that you rearrange/modify the two vertices entered by the user such that the first vertex represents the bottom-left corner and the second vertex represents the top-right corner; they still represent the same rectangle entered by the user. For example, if the user enters 19 2 9 10 for a rectangle, the first vertex is (9, 2) and the second vertex is (19, 10).

Such **pre-processing** step is very common in algorithmic problem solving. Once the data are arranged in a certain desired manner, subsequent computation can be greatly simplified.

To do the rearrangement, we suggest you add a method **arrangeVertices**(*a*, *b*) to re-arrange the two vertices *a* and *b* of a rectangle to their desired values.

If you do this and use the appropriate **Math** methods, you do not need a single 'if' statement in your program! Yes, I'm not kidding! Try to achieve this.

In the skeleton program provided, it also includes the **printAllVertices()** method for your testing.

Extension (do not submit)

This exercise can be extended to incorporate OOP concepts that you will learn in week 3, i.e. creating your own class. Create a class called **MyRect** whose attributes are the two opposite vertices **vertex1** and **vertex2**, both are **Point** objects. Add constructors, accessors, mutators and other appropriate methods for this class. Write a client program that solves this exercise, this time creating 2 **MyRect** objects instead. This is only for your own practice.

(In fact, for every exercise you did, you may think of ways to extend it to incorporate new concepts learned in class. This way, you can create your own exercises for practice.)

Notes:

- You are advised to do all take-home exercises by yourself so that you can truly learn, or you may have difficulty with the sit-in labs later.
- You may raise your doubts on the IVLE forum, but please do NOT post your codes (partial or complete) on the forum before the deadline. You may post your codes after the deadline.