Programming assignment 11

Algorithms

class Rectilinear Region:
private method areConnected:
arguments: 2 rectangles
return: whether the 2 rectangles are connected
The term "connected" is defined as the 2 rectangles fall into one of these scenarios:

- top side of the first rectangle touch the bottom side of the second rectangle, meaning the y-coordinate of the first rectangle's top equals the y-coordinate the second rectangle's bottom
- bottom side of the first rectangle touch the top side of the second rectangle, opposite to scenario 1
- left side of the first rectangle touch the right side of the second rectangle, meaning the x-coordinate of the first rectangle's left equals the x-coordinate the second rectangle's right.
- right side of the first rectangle touch the left side of the second rectangle, opposite to scenario 3

private-package method isConnected: (main method)
argument: nothing
return: whether the rectilinear region is connected
Reminder: the list of rectangles in the class is called *rectangles*.
Pseudocode: (Depth First Search)

```
initialize the set of visited rectangles with the starting element is the first rectangle.
initialize the list of unvisited rectangles with the starting element is the first rectangle.
while (the unvisited list is not null):
        assign rtg to the first element in the unvisited list
        remove the first element in the list
        for each rectangle in rectangles
                if the set of visited rectangles contains rectangle: do nothing, skip
                else if rtg and rectangle are connected:
                        add rectangle to the visited set
                        insert rectangle to the first place of the unvisited list
return (the size of the visited set equals the size of rectangles);
```

Description of data structure used:

For the visited rectangles, Set is used because it can quickly check if a rectangle is in the set.
For the unvisited rectangles, List is used because it is more efficient than

Handle exception:

validate the rectangle, as in the previous assignment, by validating the bottom left and the top right of the rectangle. Throw an illegal argument exception if one of the coordinates is null. The rectilinear region is validated when creating a new one so there is no need to create a method to handle the exception.

Testing

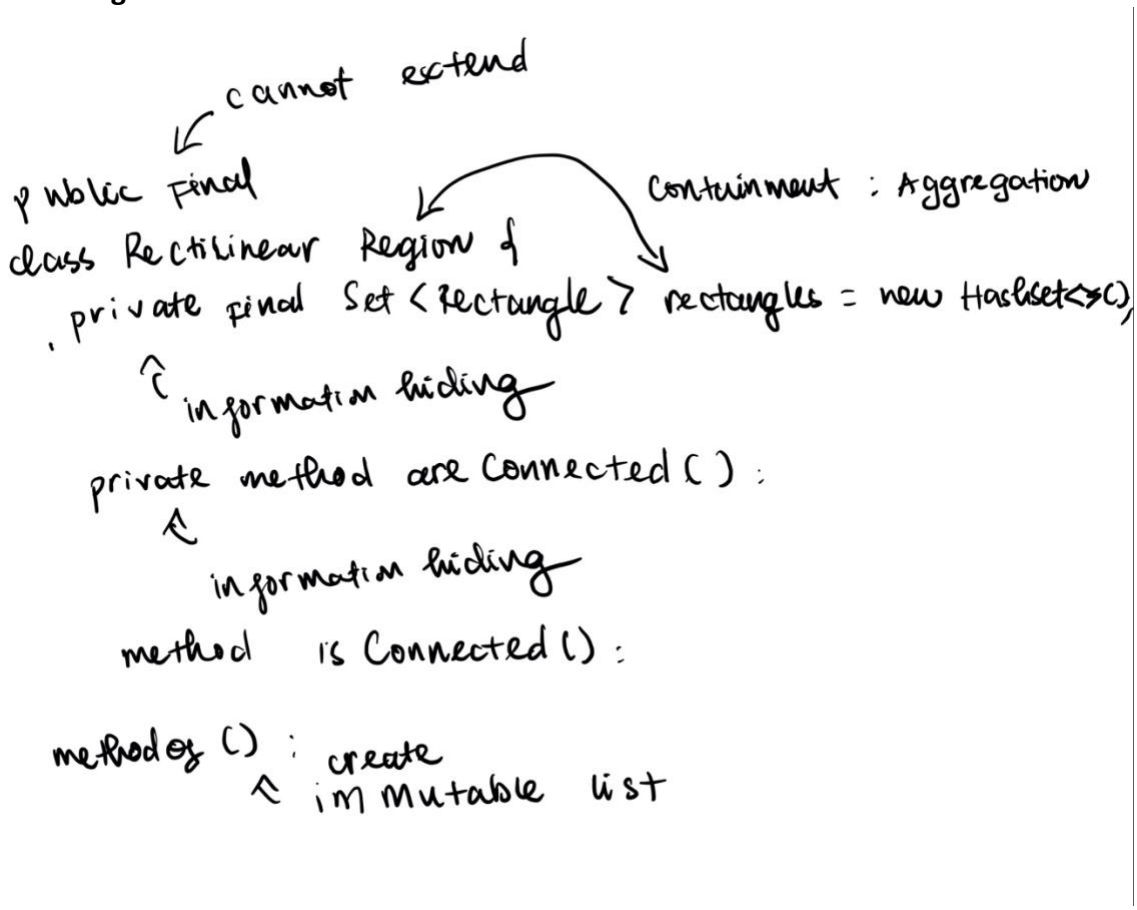
For the private class, create a nested class to test the method. The package-private method should be tested code coverage, branch coverage, and boundary.

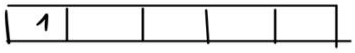
The stress test: Creating 10 test cases manually that are consecutive, meaning that the right of the first one is the left of the second one; the right of the second one is the left of the third one and continue.

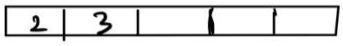
Generate 10 other overlap rectilinear region, which should return illegal argument.

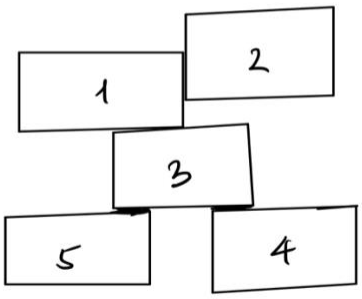
Generate 10 random rectilinear region that is neither connected or overlapped.

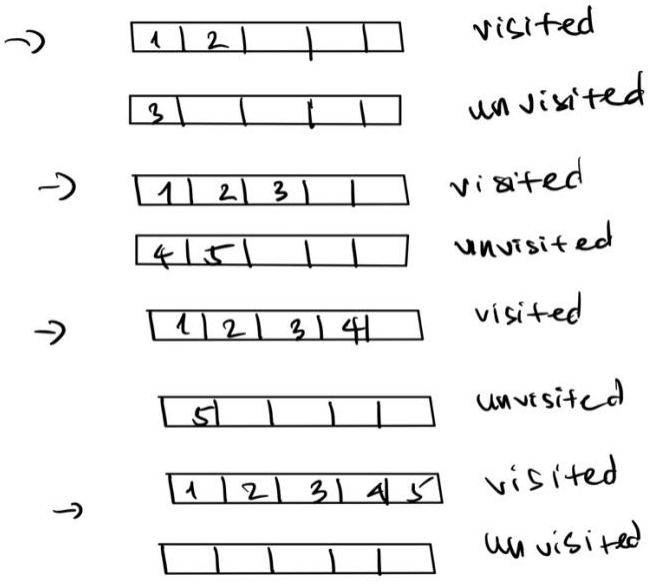
~~Class design~~




1
visited


2
3
unvisited


1
2
3
5
4


1 2
visited
3
unvisited
1 2 3
visited
4 5
unvisited
1 2 3 4
visited
5
unvisited
1 2 3 4 5
visited
unvisited
⇒ the rectilinear region is connected