Lab 12 – Classes with Methods

Start with your completed Lab 9. (If you haven't completed Lab 9, you can use the starting code for Lab 9 instead, but you will likely have difficulty understanding this lab.) Copy it to a folder called `Lab12_lastname_firstname`.

Part 1

You will be updating the `Point` class so that it uses private fields, has setters and getters and has a constructor. Perform the following steps:

1. Make the two fields private. Compile the project to see the error that now occurs. Let's fix it!
2. Add four methods: `setX`, `getX`, `setY` and `getY`. These methods should be in the `Point` class.
3. Modify the code in the class `Part1` so that it properly uses the getters and setters.
    - Where you previously could read or update the x value by using "`.x`", you will now need to use either "`.getX()`" if you want to read the value or "`.setX(newValue)`" if you want to update the value.
4. Compile and run the program and ensure that it still works.
5. Add a constructor that takes in the x and y values for the point. After doing this, you will find that the class `Part1` has compilation issues again. That is because we have "changed the rules" so that now you must have an x and y value to create a new point.
6. Modify the code in the class `Part1` so that it properly uses the new constructor.
    - You will need to modify `readPoint` so that it asks the user for the x and y value first and then creates a point second.
7. Compile and run the program and ensure that it still works.

Part 2

You will be adding new methods that use the `Point` class and are in the point class. Perform the following steps:

1. Create a new method to compute the distance between the two points using Pythagorean's theorem. This method should reside in `Part1`. Add code to the `run` method to test out this new method.
2. Create a method called `toString` that returns a nicely formatted string for the point. The string should be in the form `"(12, 8)"`. Update the `run` method to print out the points using this method. The code to print the points should look something like:
```
System.out.println("The distance from point " + start +
    " to point " + end + " is " + distance);
```
3. Create a copy of your new distance function inside the `Point` class called `distanceTo`. It should take a single point as an argument. Hint: the second point for the distance formula will be "`this`". Update the `run` method to use this method instead of the method in `Part1`.

4. Create another method inside the `Point` class called `distanceToOrigin` and have it compute and return the distance from the point to the origin at (0, 0). This function should make use of your `distanceTo` method from the previous step.
5. Create a new `describe` method that returns a description of the point as a string. An example output of the function might be `"point (3, 4) which is 5 from the origin"`. Add some code to make use of this method.

Submission

Submit your work in the usual way.

Part 3

This part will not be submitted, but is recommended. You will be creating a new `LineSegment` class that is defined by two points. Perform the following steps:

1. Create a new class called `LineSegment`. This class will have two private fields "start" and "end" that are both of type `Point`.
2. Add a constructor and getters for the new class.
3. Create a method called `toString` that returns a nicely formatted string for the line segment. The string should be in the form `"(12, 8) -> (2, 3)"`.
4. Add code to your run method to read in a line segment. This code will need to read in two x values, two y values, construct two points and then create a line segment. Then add some code to print out your line segment.
5. Add a method called length to the `LineSegment` class. This method should return the length of the line segment. Hint: this method can make use of your `distanceTo` method in the `Point` class. Add some code to make use of this new method.