CPSC 233 – Coding Challenge 2 – Practice 1

This is a practice coding challenge. You may ask other students and your TA any questions you wish. Note that you are expected to complete the actual coding challenge independently. Having a solution for this coding challenge will not help you come up with your own solution for the actual coding challenge. Only if you can solve this practice independently can you be confident that you can complete the actual coding challenge successfully. See previous coding challenges for rules for coding challenges and best practices for success.

Requirements

Remember to always create a 'skeleton' first and compile and test this using the provided JUnit test. For this coding challenge, you will be asked to create two classes. Make sure you create a skeleton of both classes first. WebCAT needs a version of both classes that compile with the tests placed in a zip file. If one of the classes is missing or does not compile, none of the tests will be run and all tests will be marked as failed.

Create two classes, one called Point and the other called Line. (Remember to name the files appropriately as well.)

Requirements for Point class

The point class represents a point on the screen. On the screen, the top-left corner is considered point (0,0). When moving down, the y-coordinate increases, to move right, the x-coordinate increases. The class should contain:

- Instance variables: *xcoord* and *ycoord*, both of type int.
- Constructors: a copy constructor and a constructor that takes initial values for the x- and y-coordinates.
- Methods:
    - Getter and setter methods for both instance variables. x- and y-coordinates must be non-negative.
    - *moveUp, moveDown, moveRight*, and *moveLeft* which all take an argument of type int which represents how far to move.
    - *distance* which takes another Point as an argument and returns the Euclidean distance between itself and the point provided as an argument. The value returned should be of type double.
    - *equals* which takes another Point as an argument and returns true if it has the same x- and y-coordinate as the Point provided as an argument.

Requirements for Line class

- Instance variables: *start* and *end*, both of type Point. Both instance variables must be completely encapsulated. Make sure to prevent all privacy leaks.
- Constructors: one constructor that takes a start and end point as an argument.
- Methods:
    - getter and setter methods for both instance variables.
    - *length* which takes no argument and returns the length of the line as a double. (The distance between the start and end point.)