CS 5004 Object Oriented Design
Recitation ICE for May 31, 2023
Points and Lines

1. Goals

- Create a classes from scratch
- Create multiple types of constructors
- Create getters and setters for private variables
- Demonstrate data encapsulation with a setter test
- Perform test driven development using a provided JUnit test file
- Use a provided driver test file as another means of testing

2. Purpose:

- The purpose of this assignment is to reinforce what you have learned in object-oriented programming thus far.
- It takes some of the topics discussed in the earlier modules like testing, using a driver, and using composition within classes. By completing this assignment, you will prove to yourself and the professor that you have a solid understanding of these concepts.
- This exercise should be fun! Team up, work in groups, draw some pictures of how a line and point connect with each other. This is especially useful when trying to determine what quadrants your line is in.

To-Do:

- Review the test file as it is your key to understanding the intended functionality of the classes which you will be creating.
- Discuss and build a frame for this assignment and load the provided test file into a test sources directory.
- Discuss how the design concept of composition will be used.
- Help each other pass the first two point tests:

```
@Test
public void noArgsPointConstructorTest() {
        assertEquals(0,p1.getX() + p1.getY());
}

@Test
public void twoArgsPointConstructorTest(){
        assertEquals(1,p2.getX());
        assertEquals(2,p2.getY());
}
```

3. **Instructions:**

<u>Testing File</u>

For this assignment, I'll give you the testing file instead of asking you to create it. Feel free to reverse engineer these tests to make sure you get it right, but don't change any of the tests. If you add any tests, put them in their own section and make sure to add comments clearly identifying the tests you added. The test file can be found online in Canvas.

<u>Part 1: Point Class</u>

We will get started by creating a class to represent a cartesian point. Here are the requirements for this class.

1. Create a class called point made up of two integer values x and y.
2. Create a no argument and a two argument constructor. Review the test file to get an idea of how to accomplish this.
3. Make sure x and y are private and include getters and setters for access.
4. Make sure the user can not set a value greater than 99 or less than -99. If they try, just set the value to 99/-99. Make sure this can't be circumvented. Use exception handling in the parameterized constructor if the values are greater than 99 or less than -99. Do not worry about resetting the values to 99 or -99 in the constructor, as this can be done in the setter methods.
5. Create a method getQuadrant that will print the quadrant A, B, C, D, or origin that a point is in. If a point is on a border, example (-4, 0), then print "Border."

   If you create a point p1, then calling this method would be:

   ```
   p1.getQuadrant();
   ```

   Create a print method that will print the x and y coordinates of a point. This will be the equivalent of a *toString()* method. A call to this method from point p1 would look like this:

   ```
   p1.printPoint();
   ```

6. Make sure this class passes the provided PointTester class.

<u>Part 2 : Line Class</u>

Class composition is when we use one class as part of another class. A line is made up of two points. We could represent this by having a line incorporate 4 integer values: x1, y1 & x2, y2, but this is redundant. Instead, we will use the already created point class objects to make a line. As long as the point class is in the same folder or the same package they will be accessible to each other.

1. Create a class called “Line” that’s made up of two points p1 and p2.
2. Create a constructor that will accept nothing and set two points at the origin.
3. Create a constructor that will accept two points.
4. Create a constructor that will accept 4 ints and use point constructors to set p1 & p2.
5. Create a print method that will print both points using the print point methods. This will be equivalent to a *toString()* method.
6. Create a *getLength()* method that returns the length of the line. Use the formula:

   $$d = ((x2-x1)^2 + (y2-y1)^2)^{0.5}$$

7. Create a lineSlope() method that returns the slope of a line. The formula is:

   $$m = (y2-y1) / (x2-x1)$$

8. Make sure this class passes all provided LineTester tests.

Part 3 : Driver

Test driven development is useful, but when you first start programming it is also important for you to understand how your classes could be used. A driver class is a class object that builds a usable application instead of representing data. It is the class where your public static void main(...) method is located and where control flow will start. For some of the applications you create, I will ask you to also create a driver. For this assignment the driver has been provided for you. The intent is to reinforce that driver testing is another means of testing classes that you create.

4. Submission:

Submit your files as a single zip file named: `“Your Name”_”Assignment”.zip` on Canvas.