CS5004-5005-SV – Lab 10 – July 9, 2020
Inner Classes and Generics

This week's lab provides an opportunity to explore and practice two concepts recently introduced in lectures: *inner classes* and *generics*.

0. Create a new Java project with a public class called Line. Line can have various constructors, but the main one to create here accepts the coordinates x1 and y1 representing a point in 2-space ***and*** the coordinates x2, y2 for a second point in 2-space. (Two points determine a unique line.) The coordinates for the constructor should be of primitive type double:

    ```
    public Line(double x1, double y1, double x2, double y2) { ... }
    ```

    For this first activity, you should not need to create additional constructors, getters, or setters, although it is fine to do so if you wish.

    a. As part of your definition of Line, create a **private, inner class** called Point. Its constructor should accept two doubles for the x and y coordinates of a point in 2-space. It should override toString to print a point thusly: (x1, y1). It also does not need getters or setters at this time. However, since it is a "helper" method, its members are not accessible outside of the context of Line. Depending on your privacy and encapsulation requirements, this could be viewed as a feature or a bug. Line should store its instance variables in the form of Points.

    b. Create a public method of Line, called slope, which returns the slope of the line as a double. (Be careful about vertical lines!)

    c. Write a main method within Line that creates a few test Lines and accesses the x and y coordinates of their defining points. Notice that you do *not* need public setters or getters to access these private fields. Explain why in your code commentary.

    d. Now write a new implementation of your Line, but this time make Point reside in a separate file, as a top-level class. Document any bugs you encounter. Modify this version so that it still works correctly. Explain the pros and cons of inner class versus top-level class in your code commentary. Both versions should be turned in.

e. [Optional] Write a method to compute the point of intersection of the two lines. (Be careful about parallel lines!)

1. Create a class, IntegerList, using the built-in generic List<E> interface:

   List<E> myList = new ArrayList<E>();

   You can find information about List<E> here:

   https://docs.oracle.com/javase/8/docs/api/java/util/List.html

   Your class should implement the Comparable<T> interface to compare the elements.

   a. Write a method to find the maximum Integer in a given list.

   b. Write a method to find the minimum Integer in a given list.

   c. Write a main method to test your code on several examples.

2. Using #1 above as a model, and starting from a ***copy*** of your Shapes code from prior labs, create a new class which contains a **List** of ***Shapes***.

   a. Your abstract Shape class was hopefully already coded to use the Comparable<T> interface to compare the areas of different shapes.

   b. Write a method to determine which shape on a list of shapes has the minimum area. It should return that shape.

   c. Write a similar method to return the shape with the maximum area.

   d. Write a main method to test your code on several examples of shapes including subclasses such as RightTriangle.

   e. Comment within your file(s) about how much of your code needed to change for this example versus the IntegerList example.

Submit your answers to above items to Canvas for CS5005 *(due next Thursday by midnight). Only your .java files for the above need to be uploaded. Add comments to your files as needed to identify the problems and to answer questions about the code or clarify your approach. Timely submission encouraged, but late is better than never. Then work on any unfinished labs or CS5004 homework assignments. Remember to submit* ***homeworks to Canvas for CS5004*** *and* ***labs to Canvas for CS5005****.*