CSE 216 Homework II

This homework does not require you to write a lot of new code, but instead, write the correct amount of code carefully within a given framework. The concepts being explored in this assignment are the different kinds of abstraction that you should be familiar with, together with the use of subtype polymorphism and parametric polymorphism. Carefully read the entire document and the given codebase before you start coding. A lot of relevant information is provided in the codebase in the form of comments and documentation.

This homework is designed in a way where the best implementation requires you to start with the big picture and only then begin to implement the details. If you start with implementing specific parts right away, you may end up in a situation where the complete codebase in the end does not work as intended.

Development Environment

It is highly recommended that you use IntelliJ IDEA (available at https://www.jetbrains.com/idea/) by either downloading the free community edition, or create a student account and get access to the Ultimate edition for free for one year (after that, you can renew it if you want). You can, if you really want, use a different IDE, but if things go wrong there, you may be on your own.

Programming Language

Starting with this homework, and for the remainder of this course, all Java code must be JDK 1.8 compliant. That is, you may have a higher version of Java installed, but the “language level” must be set to Java 8. This can be easily done in IntelliJ IDEA by going to “Project Structure” and selecting the appropriate “Project language level". This is a very important requirement, since Java 9 and beyond have additional language features that will not compile with a Java 8 compiler.

- You may import core Java libraries available with JDK 1.8. For example, anything within `java.util.*` is safe to import. But again, make sure your language level is set to Java 8, since later releases may have additional libraries not available to Java 8 (and then, your code will not compile).

Codebase, Tasks, and Rubric

There is some amount of code already given to you, and a large part of this is very similar to the code examples we covered in the lectures on polymorphism. In the given code, you will find the following abstract classes or interfaces

AbstractPrinter, ***Point***, ***Positionable***, ***TwoDShape***,

and classes

Printer, **Circle**, **Triangle**, **Quadrilateral**, **TwoDPoint**, **ThreeDPoint**, **Ordering**.

The main driver class is Ordering, where a sample main method is provided. This code is extensively commented, and divided into three main sections. Please read this very carefully. This method also serves as a test scenario for your complete code.

The given code is incomplete, and several places are marked with a "TODO" tag. Your task is to follow the documentation and the comments to complete the codebase. A few things to keep in mind during this code completion:

- The assignment uses the Comparable and Comparator interfaces. The comments in the main driver class will illustrate how they are used, and why both are needed.
- Any fields you need/want to add to the classes must be private, with the corresponding getter and setter methods added by you (if needed).
- You may add more methods or fields to the classes if you feel the need. But ***do not modify any class or interface where the comment specifies not to change the code***.
- For numeric computations, two decimal-places are expected in this assignment. You do not have to be more precise than that, and your code will not be tested with input points that have more than two decimal places. You need not round the `area()` and `perimeter()` outputs, though.

1. Completion of **`Circle`** code (4 points):

   `Circle#setPosition(List)` (1)

   `Circle#getPosition()` (1)

   `Circle#area()` (1)

   `Circle#perimeter()` (1)

2. Completion of Triangle code (21 points):

   `Triangle` constructor (1)

   `Triangle#setPosition(List)` (3)

   `Triangle#getPosition()` (3)

   `Triangle#area()` (2)

   `Triangle#perimeter()` (4)

   `Triangle#snap()` (4)

   `Triangle#isMember()` (4)

3. Completion of Quadrilateral code (21 points):

   `Quadrilateral` constructor (1)

   `Quadrilateral#setPosition(List)` (3)

`Quadrilateral#getPosition()` (3)

`Quadrilateral#area()` (2)

`Quadrilateral#perimeter()` (4)

`Quadrilateral#snap()` (4)

`Quadrilateral#isMember()` (4)

For the purpose of this assignment, you may assume that quadrilaterals are always convex (i.e., you are not expected to handle irregular/concave polygons, and your code will not be tested with concave polygon inputs).

4. Completion of TwoDPoint code (8 points):

   `TwoDPoint` constructor (2)

   `TwoDPoint#coordinates()` (2)

   `TwoDPoint#ofDoubles(double...)` (4)

5. Completion of ThreeDPoint code (4 points):

   `ThreeDPoint` constructor (2)

   `ThreeDPoint#coordinates()` (2)

6. Fix the definitions of the two lists in `main(String[])`: shapes and points. (4)

7. Completion of the portion required to run "SECTION 1" (16 points):

   (a) Fix the `Ordering#copy(List, List)` method so that it works with **note-1** in the given code. (3)

   (b) Complete the `XLocationShapeComparator` implementation as per the given documentation, so that calling `shapes.sort(new XLocationComparator())` works as expected. (3)

   (c) Finish the required implementation in the codebase so that the call to `Collections.sort(shapes)` works as expected. The expected behavior of this sorting is provided in the "TODO" comment. (3)

   (d) The call to `points.sort(new XLocationPointComparator())` should work as expected. The expected behavior of this comparator is provided in the "TODO" comment. (3)

   (e) Implement the natural ordering so that the call to `Collections.sort(points)` works. The natural ordering is described in the "TODO" comment. (4)

8. Completion of the portion required to run "SECTION 2". For this, you must fix the `Ordering#copy(List, List)` method so that it works with **note-2**, **note-3**, and **note-4** as well. (3)

9. Completion of the portion required to run "SECTION 3" (19 points):

   (a) What method(s) do you need to override in each class such that the printer automatically prints the human-readable strings for each object? Implement this for the Circle, Triangle, and Quadrilateral classes. The expected string representations are explained in the comments provided with the code. (9)

   (b) What must you implement so that the `printAllAndReturnLeast(List, AbstractPrinter)` method can make sense of the notion of "least"? Think carefully about ***where*** you must implement this so that the call to `printAllAndReturnLeast(lst, new Printer())` in the main method works. (6)

   (c) The `printAllAndReturnLeast(lst, new Printer())` will compile after your above implementation, but there are other aspects of this method that need to be fixed. This has to do with introducing the correct generic parameters. Once you do this, the method will run properly. (4)

The **`Ordering#main(String[])`** method serves as a test code of sorts. If, after following the instructions in the comments, your code does not compile or run properly, then it indicates a denite error in your implementation(s). Keep in mind, though, that a complete suite of tests is not provided here. You are, of course, free to add your own tests to check whether or not your completed code is running as expected.

NOTES

- **Late submissions or uncompilable code will not be accepted**.
- Please remember to verify what you are submitting. Make sure you are, indeed, submitting what you think you are submitting!
- **What to submit?** A single .zip file comprising the completed codebase. *Some parts of this assignment will be graded by a script, so be absolutely sure that the submission follows this structure.*
  - In the submission, please include YOUR uncommented version of Ordering.java. If you have any additional code (e.g., creating instances or making method calls), you can include that as well, but these should all be placed before the line marked as "any additional code you write must be above this". (line num. 30 in the code given to you). *Do NOT include any additional print statements in the submitted code, however*.