Object-Oriented Programming

Exercise series 3

Exercise 1

Open the `Coordinates.java` and `CoordinatesTest.java` files which are provided along with this document. Have a look at them and answer the following questions.

— How does the `Coordinates` class handle encapsulation ?

— Why are there several constructors ? What mechanism is at play here ?

— Read the `main()` method from the `CoordinatesTest` class. Do `a` and `b` reference distinct objects ? Compile and run the program to test your hypothesis.

— How would you modify the `main()` method in order to change what you observed at the previous point ? Test your idea.

Exercise 2

Create a `LineSegment` class modeling a segment characterized by two endpoints (x_1, y_1) and (x_2, y_2), where x_1, y_1, x_2 and y_2 are integers. This class should implement the following operations.

— Instantiating a new segment, for given values of x_1, y_1, x_2 and y_2.

— Translating a segment by a given translation vector (δ_x, δ_y).

— Computing the length of the segment (as a real value).

— Generating a `String` object describing the segment, e.g., "$[(x_1, y_1),\ (x_2, y_2)]$".

Tips :

— Consider reusing the `Coordinates` class from Exercise 1 without modifying it.

— Consider using the `Math.hypot()` method to compute the length of a segment.

Exercise 3

Create a `PointCloud` class modeling a 2D point cloud, i.e., a collection of (x, y) (integer) coordinates in 2D Euclidean space. This class should implement the following operations.

— Instantiating a new `PointCloud` object.

— Adding a new point (x, y) to the cloud.

— Translating all points of the cloud by a given translation vector (δ_x, δ_y).

— Generating an array of `LineSegment` objects describing line segments that connect all consecutive pairs of points. For instance, if a cloud contains the three points $(0, 1)$, $(2, 2)$ and $(2, 0)$ inserted in that order, this operation should output the segments $[(0, 1), (2, 2)]$ and $[(2, 2), (2, 0)]$.

After having implemented `PointCloud`, create a side class with a `main()` method to test the classes that you have written for this series. For example, you can instantiate a `PointCloud` object, insert some points in it, translate the whole cloud, get the segments joining the points, and finally display them in text format along with their respective length.

Tips and remarks :

— Consider reusing again the `Coordinates` class from Exercise 1.

— Consider using the `Vector` class from the Java library to store the points. The way you will use it can be inspired by the code of the `Groceries` class used in Exercise 2 of the first series. If you are confident enough, you can also review the full interface of the class in the online Java documentation.

— Suppose a third-party developper exploits your `LineSegment` and `PointCloud` classes, knowing their interface but not their implementation. Will they be aware of the `Coordinates` class ? Does this solution respect the philosophy of object-oriented programming ?