Lab 7

Note : Code for the prob1 & prob4 refer - \\cs5\Public\Courses\CS401-MPP\Renuka\DemoCode-LabSolution.

Prob2 & Prob3 refer the specified package mentioned in the problem- \\cs5\Public\Courses\CS401-MPP\Renuka\DemoCode-LabSolution.

Also available as an attachment in Sakai with your homework.

1. Complete the task (a)
   a) Parts B – D of this Problem refer to code in package `lesson7.labs.prob1`, in which you are trying to remove duplicates from a List and then test that your output is correct. All three attempts to solve this problem are incorrect in some way (when you run the code, output message indicates that the procedure fails (return false). Explain, in each case, what is wrong with the solution. Place each of your answers in a text file in the relevant package.

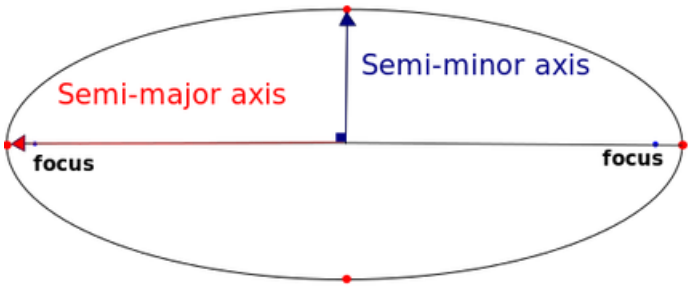
2. The Lesson 5 Demo in `lesson5.lecture.intfaces2` shows how to polymorphically compute the average perimeter of a list of geometric objects by requiring each to implement the `ClosedCurve` interface. Notice that when a closed curve happens to be a polygon, computing the perimeter of polygon is especially easy – you just add up the lengths of the sides.

   If we create an interface `Polygon` having method `double[] getSides()` (which will return the length of each side of the polygon in an array), we could replace `ClosedCurve` in our example with `Polygon` – *if* we didn't have to take into account the computation of the perimeter of *non*-polygons, like `Circles`.

   Copy the classes/interfaces from lesson5.lecture.intfaces2 into a new package for this Lab problem, and create a new `Polygon` interface. Then think of a way to make use of both `ClosedCurve` and `Polygon` so that, when `computeAveragePerimeter` is called on a `ClosedCurve` that implements the `Polygon` interface, the side lengths are added up, but when the object is not a polygon, a different computation of perimeter is done (as in the case of a circle). *Hint.* Create a `default` method in `Polygon to computePerimeter() which returns the sum of lengths.`

   Try out your approach by adding two new `ClosedCurves` to your package: `EquilateralTriangle` and `Ellipse` (an equilateral triangle is a triangle in which all side lengths are equal). Modify `DataMiner` so that it includes in the `objects` list instances of these new classes.

Hint. The perimeter (or circumference) of an ellipse is $4aE$ where a is the length of the semi-major axis and E is the value of the elliptic integral evaluated at the ellipse's eccentricity. You do not need to know these technical concepts; just include a and E as instance variables in your class, of type `double`, and include them as arguments to the `Ellipse` constructor.



3. Rework the Duck Application of Lab 5, Problem 1 by using Java 8 interfaces with default methods. Rewrite your code with this approach.

4. In the lesson7.lab4.prob4 package, there is a class called ForEachExample that specifies, in its main method, a list of Strings. Use the Java 8 forEach method within the main method to print out the list so that *all Strings are in upper case*. To do this, you will need to define your own implementation of the Consumer interface.