Object-Oriented Programming
Tutorial 2

Exercise 1: Cities and countries

A country is composed of several cities, and exactly one capital city.

Question 1.1: Propose a class diagram for the following classes: City, Capital and Country.

Question 1.2: Write the Capital and City classes in Java.

Question 1.3: How to ensure that a Country has exactly one capital city?

The java.lang package contains an interface called Enumeration. This interface declares two methods:

- public boolean hasMoreElements()
- public Object nextElement()

Question 1.4: Let us assume that there is a method to add a city to a country. Write the function that allows to add to a country several cities. The function will be declared as: public boolean addCities(Enumeration cities).

Exercise 2: Geometrical shapes - interface and implementation

We want to implement geometrical shapes. To do this, we need a class that implements the points of the shapes. We have two ways to implement points: either by cartesian coordinates, or by polar coordinates (i.e., module and argument of a complex number). However, each way should allow us to do the following:

- to access to the easting (value on the x-axis) and northing (value on the y-axis), and to the module and to the argument,
- to compute the distance between two points,
- to compare whether two points are equal or not,
- to rotate points, by giving the center of the rotation and the angle,
- to compute a homothety of the point, from the center and the ratio,
- to print the point using cartesian coordinates.

Question 2.1: Write a class Cartesian2DPoint and a class Polar2DPoint. You do not need to implement all methods in details, but you need their declaration.

Now, we want to implement triangles. A triangle is represented by three points. We want to be able to do the following:

- to access all three points of the triangle from a single method,
- to perform a homothety by giving the center of the homothety, from the center and the ratio,
- to perform a rotation by giving the center of the rotation and the angle,
- to print the triangle.

Question 2.2: Write a CartesianTriangle class and a PolarTriangle class.

Question 2.3: If we now implement a Point3D class, how to manipulate triangles using these 3D points?

Question 2.4: Propose a method where you implement the Triangle class only once. Your implementation should allow to manipulate triangles from any type of point. In other words, this method should not be modified when we implement a new implementation of points.

Exercise 3: Geometrical shapes - multiple inheritance and overloading

Here are the operations we plan to use on geometrical shapes:

- to compute the area size,
- to perform a rotation of the shape, given the center and the angle,
- to perform a homothety of the shape, given the center and the ratio,
- to print the shape as a string.

We also want to implement some shapes with centers. For those shapes, we need to be able to access to the center.

We start by considering only the following shapes: triangles (no center), quadrangles (no center), and circles (with a center).

Question 3.1: Propose a class diagram that corresponds to this specification.

We now add to our specifications rings. A ring is a circle with a circle-shaped hole. The ring and the hold have the same center.

Question 3.2: Add the Ring class to your previous class diagram.

Question 3.3: Implement the java classes for circles and rings. To create a circle, you need the center and the radius. To create a ring, you need the main circle, and the radius of the hole.

Now, we want to implement animated shapes. A shape is animated if it implements the Runnable interface, containing the method void run(), which enables the animation.

Question 3.4: Add this specification to your class diagram. Note that not all shapes are animated.

Question 3.5: When an animated shape is printed, the string should mention that it is an animated object. Write the corresponding methods, for animated circles and animated rings.