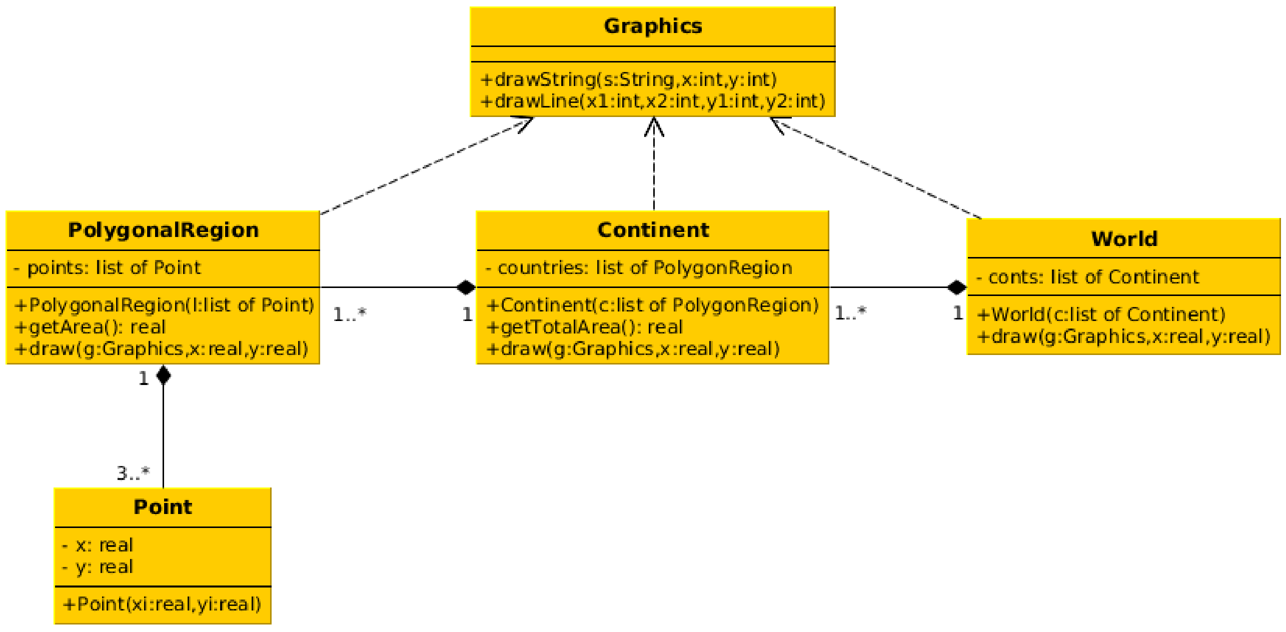
Lab exercise 2: Implementing a program design
24292-Object Oriented Programming

1 Introduction

The aim of this lab session is to implement part of the program design of Seminar 2. The session is mandatory and you have to deliver the source code of the Java project and a document describing the implementation decisions and details.

Since the program design from Seminar 2 involves a lot of classes, we will implement the code in steps. To start with, we will only consider the classes representing polygons, continents and worlds. The following figure shows a fragment of the program design from Seminar 2. You are free to implement your own design for this subset of classes, but the descriptions in the following sections refer to the sample solution.



Graphics
+drawString(s:String,x:int,y:int)
+drawLine(x1:int,x2:int,y1:int,y2:int)
PolygonalRegion
- points: list of Point
+PolygonalRegion(l:list of Point)
+getArea(): real
+draw(g:Graphics,x:real,y:real)
Continent
- countries: list of PolygonRegion
+Continent(c:list of PolygonRegion)
+getTotalArea(): real
+draw(g:Graphics,x:real,y:real)
World
- conts: list of Continent
+World(c:list of Continent)
+draw(g:Graphics,x:real,y:real)
Point
- x: real
- y: real
+Point(xi:real,yi:real)
1..*
1
1..*
1
1
3..*

The first part of the implementation is to create a graphical interface in Java. The supplied code includes classes MyWindow and MyMap that provide the skeleton for the graphical user interface.

2 Polygonal regions

Although the design included both polygonal and ellipsoidal regions, we are only going to implement polygonal regions in this lab session. You do not have to implement the classes “Region” and “EllipsoidalRegion”.

To compute the area of a polygon, you may assume that the polygon is *convex*. The following page shows how to compute the area of a convex polygon:

`http://www.mathwords.com/a/area_convex_polygon.htm`

To draw a polygon, you can either use the method `drawLine` of Graphics to draw each line between a pair of points of the polygon, or the method `drawPolygon` which draws the entire polygon at once. Note that the method `drawPolygon` assumes that the points are given by two arrays of integers: one for the x-coordinates, and one for the y-coordinates. Hence the method `draw` in `PolygonalRegion` has to first convert the points to the correct representation.

Once you have implemented the PolygonalRegion class you should test it in the graphical interface. To do so, we are going to modify the source code of MyMap. To the class MyMap we are going to add an attribute which is a polygonal region. We are also going to modify the constructor method. The resulting code should look like this:

```
private PolygonalRegion region;

public MyMap() {
    initComponents();

    LinkedList< Point > points = new LinkedList< Point >();
    points.add( new Point( 10, 100 ) );
    points.add( new Point( 150, 10 ) );
    points.add( new Point( 290, 100 ) );
    points.add( new Point( 290, 200 ) );
    points.add( new Point( 150, 290 ) );
    points.add( new Point( 10, 200 ) );

    region = new PolygonalRegion( points );
    System.out.println( region.getArea() );
}
```

Next, modify the method `paint` of MyMap. This method should look as follows:

```
public void paint( java.awt.Graphics g ) {
    super.paint( g );
    region.draw( g );
}
```

If everything works as intended, running the program should open a window in which the region is drawn, and also print the total area of the region in the terminal.

3 Continents and World

Next, implement the classes Continent and World and change the graphical interface so that instead of a single country, the entire world is drawn. To do this, you have to perform the following steps:

1. Implement the class Continent such that it includes a list of polygonal regions. The draw method should simply draw all the regions of the continent.
2. Implement the class World such that it includes a list of continents. The draw method should simply draw all the continents.
3. Modify the class MyMap such that the attribute is a World rather than a single region. The main method has to create *all* the regions, continents as well as the world.

Note that the map has size 1000×1000, so it is a good idea to use coordinates in the interval $[0, 1000]$.