Pablo Fernández
Clara López
S101, P101

Lab 2 Report

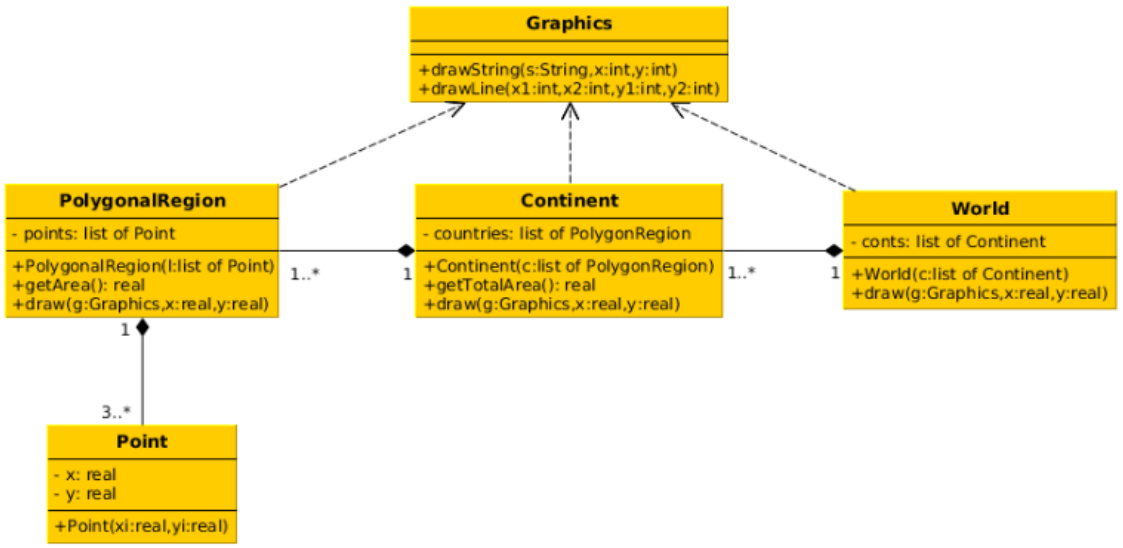
Introduction

This report presents the work done in the second laboratory project of the Object-Oriented Programming course.

The aim of this second project was to create a program that related regions, continents and worlds by applying the concepts learned in theory sessions. It also had to draw each of those classes using the Graphics class. To do so, we had to implement the classes and relations we had designed in the previous seminar session, which were Polygonal Region, Continent and World.

These classes are related to each other, for instance, a Continent is a set of countries (Polygonal region).

When creating a class, it is important to take into account the relation they may have between them. In this laboratory we can see that between the classes *Point* and *PolygonalRegion*, *PolygonalRegion* and *Continent* and between *Continent* and *World*, there is a Composition of many to one relation.
Moreover, *PolygonalRegion*, *Continent* and *World*, have a Dependency relation with *Graphics*, that is because they need this class for their *draw* function.



Pablo Fernández
Clara López
S101, P101

Description

While doing the practice, we noticed that there were different options to do the draw function:

Option 1: drawPolygon

At the beginning we attempted to use this function in order to implement the draw function in PolygonalRegion. Nevertheless, we found it too hard because we had to import other libraries, as well as to use the Polygon class, so we finally decided to implement Option 2.

Option 2: drawLine

For this option, we had to change the *double* parameters in the Point class to *int* in order to make the drawline function work.

For the rest of the code, we did not discuss other solutions, as what we did was code together and build the code step by step.
Finally, the solution we got to is the following:

1. Point

The first part of this practice consisted in implementing the code for the *Point* class.

```
public class Point {
    private int x;
    private int y;
```

First, we defined the attributes of the *Point* class, which are the int variables *x* and *y*.

```
public Point (int initX, int initY){
    x = initX;
    y = initY;
}
```

This is the Constructor method, which is where the value of the attribute is assigned.

```
public int getX() {
    return x;
}

public int getY() {
    return y;
}
```

Getters. This method returns the current value of an attribute, and we have to create a specific one for each attribute.

```
public void setX(int x1) {
    x = x1;
}

public void setY(int y1) {
    y = y1;
}
```

Setters, which are used to change the value of the attributes. The same as the getters, we have to create one per attribute.

Pablo Fernández
Clara López
S101, P101

```java
public double distance(Point p2){
    return Math.sqrt(Math.pow(x - p2.x,2) + Math.pow(y - p2.y,2));
}
```

The function distance is used to calculate the distance between two points.

```java
public void printPoint(){
    System.out.println("(" + x + "," + y + ")");

}
```

The last function, printPoint is used to print the coordinates of X and Y of the point.

2. Polygonal Region

Secondly, we had to implement the *PolygonalRegion* class. This class is used to calculate the area of the different regions (in this practice we assume that all of them are convex polygons), in order to be able to draw them.

```java
public class PolygonalRegion {
    private LinkedList<Point> points;
```

We define the attribute *points* as a List of Points

```java
public PolygonalRegion( LinkedList<Point> initPoints) {
    points = initPoints;
}
```

Constructor method, the same way as in point, in order to assign a value to the attribute.

```java
public double getArea() {
    double area = 0.0;

    int size = points.size();
    int j;

    for (int i = 0; i < size; i++) {
        if (i == size-1) {
            j = 1;
        } else {
            j = i+1;
        }
        area += points.get(i).getX()*points.get(j).getY();
        area -= points.get(i).getY()*points.get(j).getX();
    }

    area = area/2;
    return area;
}
```

The function getArea is used to compute the area of the form that the list of points creates, which is a convex polygon. We use a for loop in order to develop the formula for these polygons.

Pablo Fernández
Clara López
S101, P101

```
public void draw(java.awt.Graphics g) {
    for (int k = 0; k < points.size()-1; k++) {
        g.drawLine(points.get(k).getX(), points.get(k).getY(), points.get(k+1).getX(), points.get(k+1).getY());
    }

    g.drawLine(points.get(points.size()-1).getX(), points.get(points.size()-1).getY(), points.get(0).getX(), points.get(0).getY());
}
```

Lastly, the function draw is used to draw the form that the polygon has.

In order to compute this function, we have used the *drawLine* option which consists of drawing each line between a pair of points. Apart from the loop, we need to draw the last line between the last and the first point.

3. Continent

Then, we had to implement the *Continent* class.

```
public class Continent {
    private LinkedList<PolygonalRegion> countries;
```

First we defined the attribute *countries* as a List of Polygonal Regions

```
public Continent(LinkedList<PolygonalRegion> initCountries) {
    countries = initCountries;
}
```

Constructor method, in order to assign a value to the attribute

```
public double getTotalArea(){
    double totalArea = 0;
    for(int i = 0; i < countries.size(); i++){
        totalArea += countries.get(i).getArea();
    }
    return totalArea;
}
```

This method is used to calculate the total area as the sum of the area of each different continent.

```
ublic void draw(java.awt.Graphics g) {
   for (int j = 0; j < countries.size(); j++)
       countries.get(j).draw(g);
   }
```

The draw method is used to draw all the countries defined.

4. World

Finally, we implemented the *World* class.

```
public class World {
    private LinkedList<Continent> continents;
```

In here we defined the attribute *continents* as a List of Continent

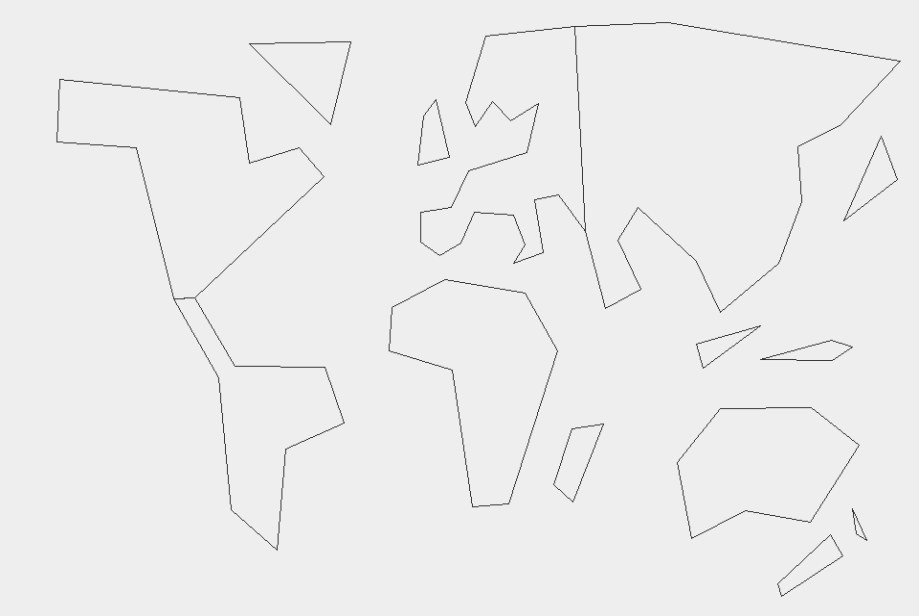
```
public World(LinkedList<Continent> initContinents){
    continents = initContinents;
}
```

Constructor method, to initialize the value of the attribute.

```
public void draw(java.awt.Graphics g){
    for (int i = 0; i < continents.size(); i++){
        continents.get(i).draw(g);
    }
}
```

The draw method is used to draw the whole world, drawing all the continents we have defined.

Final result



We manually drew a world map in Paint in order to get the coordinates of the points we could use to represent an approximate shape of each continent, because we wanted to make the most out of our program.

Pablo Fernández
Clara López
S101, P101

Conclusion

On the whole, we both are really satisfied with the final result, even though we had some difficulties at the beginning, especially when it came to calculating the area of the polygon, but with time and effort, we were able to overcome this problem and at the end, we are happy with the work we have done.