Ayesha Quinto Gammuac 241789
Minh Nguyen Phuong 241841

24292 - Object-Oriented Programming

LAB 3: Extending the World design

REPORT

THE PROGRAM

1. MAIN PURPOSE AND DESCRIPTION

The aim of this report is to describe how we have done lab 3, including all the problems we encountered and our solutions. The objective of the program is to create a map of a world which is similar to the previous lab, but this time with different continents, countries instead of polygonal regions and cities which are created by using coordinates on the screen. In this lab, we made a copy of some classes from the previous lab, which are *Point.java*, *PolygonalRegion.java*, *Continent.java*, *World.java, MyMap.java,* and *MyWindow.java*. For this lab, we implemented the new classes *City.java*, *Country.java* (using Polygonal Region), *Lake.java* and *River.java* and the equivalent testing classes.

1.1. Country

Extending the PolygonalRegion, we created the class Country using the following public methods: *Country(), getName(), getCapital(), addCity(), addNeighbor()* and *drawCountry()*. The attributes for this class are the name of the country *name*, the capital of the country *capital*, a list of the cities in the country *ListCities* and a list of neighboring countries *ListNeighbors.* In the drawing country part, we implemented a for loop to draw the capital, as we did with the previous labs.

We also changed the definition of the class Continent so that they consist of countries instead of PolygonalRegion and so that it also draws countries, drawing forth its corresponding cities and other elements.

1.2. City

Inheriting from the class Point, we implemented the City class so that each city will be a geometrical point. The class, apart from the attributes of Point, only has one attribute *numhab* which will be the number of habitants. The public methods are *City(), getNumHab()* and *drawCity()*. We wanted to make it easier to see the capital of each country, so we chose to draw the capital in red, while other cities are green. We also used the function *drawString()* to add the name of each city above itself. The following image shows how we used the different graphics methods to draw our cities.

```
public void drawCity( Graphics g, int x, int y, boolean capital ) {
    if ( capital ) {
        g.setColor(Color.red);
        g.fillOval( x, y, 7, 7 );
        g.setColor(Color.blue);
        g.setFont(new Font("AvantGarde", Font.PLAIN | Font.BOLD, 10));
        g.drawString( name , x, y );
    } else {
        g.setColor(Color.green);
        g.fillOval( x, y, 7, 7 );
        g.setColor(Color.blue);
        g.setFont(new Font("AvantGarde", Font.PLAIN, 10));
        g.drawString( name , x, y );
    }
    g.setColor(Color.black);
    g.drawOval( x, y, 7, 7 );


}
```

2. OPTIONAL PART

As an addition, we have also represented lakes and rivers in our world. In the document provided by the teacher, it stated to draw oceans and lakes, but we decided to draw rivers instead of oceans. Thus, we implemented the classes *River.java* and *Lake.java*.

The River class is composed of a name and a LinkedList of Point. These points are how we will represent the river's path and it will be drawn using the graphics method *drawPolyline()* which uses the list of points separated by x-coordinates and y-coordinates. We also show the name of the river using *drawString()*, both draw functions inside our class' method *drawRiver()*.

For the Lake class, we decided to use our already existing Point class and extend it since we have decided to represent lakes as points in our map. Therefore, apart from the attributes inherited from Point, it has the integers *width* and *height*. As our rivers and cities, our lakes will also have their names printed on the map. In our method *drawLake()*, we call the graphics method *fillOval()* which receives our x and y coordinates and the width and height, and *drawString()*. For both classes mentioned above, we have used the color blue to draw them, calling the method *setColor(Color.blue)*.

After implementing the classes, we had to make the necessary changes in the other classes to make the whole project work. For example, we added the attributes *ListRivers* and *ListLakes* in the Country class.

3. CONCLUSION

In general, we were able to complete the lab without many difficulties. Also, in order to make the most of our map, we have researched methods involving the Graphics from Java. For example, we got to learn how to draw strings and change their font and size, the color for each element, to draw a line of multiple points, etc. We've been able to apply some things we have been taught in class such as the inheritance and interfaces.