Computer Science II, D-BAUG FS 2016
Departement of Computer Science
ETH Zurich

Assignment 7

Out: 11th of April 2016
Due: 19th of April 2016

Imagine you want to use your smart-phone in order to determine where you are. Your phone is capable of receiving GPS signals and determine the position on earth in terms of two-dimensional coordinates. Moreover, you have a map, internally representing countries, cities and islands (etc.) as closed polygons. You are to write a (part of) a program that determines where you are. Moreover, you want to display this graphically. How would you do this? In this exercise we accomplish this task, of course with a simplified setup and scenario.

7.1 Point in Polygon

The idea of this exercise is that you get a coordinate (x,y) for instance from a GPS signal and you want to find out on which island this coordinate is located or if it is in the ocean. Islands in this context are polygons and a coordinate is simply a point. In this exercise we will use the Point-in-Polygon algorithm presented in the class. With the Point-in-Polygon algorithm you can check if a coordinate (x,y) is on an island (inside a polygon). You will implement three functionalities, insert a new island, check if a point is on any of the islands, draw the islands and the point in a graphical viewer (ex 5.2).

To solve this exercise you should download the skeleton project at: `http://lec.inf.ethz.ch/baug/informatik2/2016/ex/ex07/ex07.zip` and import it into eclipse. You can import the project by going File → Import. Then pick "Existing Project into Workspace" under *General* and click *Next*. Then instead of "Select root directory", choose "Select archive file" and navigate to the file ex07.zip you just downloaded. Select all and click *Finish*.

The Skeleton project contains 4 files:

- Main.java
- Polygon.java
- Vertex.java
- ImageViewer.java (can be ignored for part 5.1)

Main.java, contains the Main class which will start your program. You will have to add your code in this file to make the program run. Polygon.java contains the Polygon class and the PointInPolygon algorithm as presented in the lecture. Vertex.java contains the Vertex (Eckpunkt) class which specifies a point with its coordinates (x,y) and has a reference to the next Point in the Polygon. Although the Polygon and Vertex class were presented in the lecture you should have another look at the them to make sure you understand how they work. The ImageViewer.java file is not used in this first part, and you can ignore, also the internals are more complicated and you do not need to understand them.

Island Class

First you should implement the Island class in Main.java, the class should have two variables, the name of the island as a String and the shape of the island as a Polygon. Also add a constructor to the class which takes a String and a Polygon as a parameter and initializes the two variables name and shape.

Main program

After you have implemented the `Island` class, we have a closer look at the main() function in the class Main (in Main.java). The main function will read in the test input and start some function, such as adding a new island, checking if a point is on an island, and drawing the islands. Therefore each input has to start with a command, in this exercise we have the commands *"insert"*, *"onisland"*, *"draw"*.

```
main(String[] args) {
    ...
    while (scanner.hasNext()) {
        String cmd = scanner.next();
        if (cmd.euqals("insert")) { // Add new island
            ...
        }
        else if (cmd.equals("onisland")) { // Check if point is
                                           // on an existing island
            ...
        }
        else if (cmd.equals("draw")) { // Draw islands
            ...
        }
        else {
            System.out.println("Invalid input. terminating...");
            break;
        }
    }
}
```

As you can see in the skeleton code, depending on the command `cmd` a different branch of the if-else block is executed.

Insert Island

You should implement the first command "insert" in the first branch of the if-else block in the main() function in Main.java. The input to add a new island is the following:

```
insert cuba 3
100 100
300 100
100 300
```

The first line specifies the command "insert" followed by the name of the island ("cuba") and the number of coordinates the polygon has (3 in this case). The next lines all specify coordinates (x,y) to define the shape of the island.

You should implement the functionality in

```
...
String cmd = scanner.next();
if (cmd.euqals("insert")) { // Add new island
    // TODO implement this
}
...
```

As you can see the command is already read in with *scanner.next()*. To read in the name of the island also use *scanner.next()*, to read in numbers use *scanner.nextInt()*. The coordinates you read in should be added to a newly created polygon. Once you have read in all coordinates and added them to a polygon you should create a new island with the name and the new polygon. This new island should then be added to the list of islands, called islands. which is defined at the beginning of the main() function, the new island can be added the following way:

```
    // Add new island to list of islands
    islands.add(new Island(name, polygon));
```

Note: With the function call *scanner.hasNextInt()*, you can check if the next symbol from the scanner is of type integer.

Point on Island

In this part you implement the second command "onisland", the second branch of the if-else block in the main() function in Main.java. The input to check if a point is on an island is the following:

```
onisland 200 200
```

The command "onisland" is followed by the coordinates (x,y) of the point to check.

You should implement the functionality in

```
...
else if (cmd.euqals("onisland")) { // // Check if point is on an existing isla
    // TODO implement this
}
...
```

As before the command was already read in with the scanner. So you have to only read in the coordinates, use *scanner.nextInt()* to read in the x,y coordinate. Using the coordinate values you should iterate through all islands in the islands list and call the PointInPolygon(x,y) function on the shape of each island. In case the point is on the island print out the name of the island with *System.out.println()*. The print out should look like this for an island called cuba: "**Point is on cuba.**" If you iterated through all the islands and the point was not on any of them, print out "**Point is not on any island.**"

Note: You can iterate through the island list which is of type `ArrayList<Island>`, the following way:

```
for (int i = 0; i < islands.size(); i++) {
    islands.get(i);
}
```

You can validate your solution here: https://challenge.inf.ethz.ch/team/websubmit.php?problem=IB16071
You have to copy the content from Main.java, Polygon.java and Vertex.java into the submission field on the judge. Copy Main.java first, such that import statements are on the top.

7.2 Drawing Polygons

In this part of the exercise we extend the code from exercise 5.1. The goal is that we can draw all the islands and a point. Thereby you can visually check if this point is on an island or not.

To draw the polygons we make use of the ImageViewer class in ImageViewer.java. You do not need to understand the internals of this class, here we give a brief overview how you can use it to draw polygons and points.

```
    // Create an ImageViewer, with width: 800, and height: 600
    ImageViewer viewer = new ImageViewer(800,600);
    // Draw a line on ImageViewer viewer,
    // from Point (100, 100) to Point (200, 200)
    viewer.line(100, 100, 200, 200);
    // Draw a cross at the coordinate (150, 150);
    viewer.cross(150,150);
    // Update the viewer, required to draw everything
    viewer.update();
```

In the example above we first create a new ImageViewer. You can either draw a line on the ImageViewer with the function line() or draw a cross at a specific coordinate with the function cross(). Once you have drawn all the objects you should update the viewer through the update() function. This makes sure that everything is actually drawn and visible.

To make the drawing of polygons easier for the programmer, we want to extend the Polygon class with a new function called DrawPolygon

```
public void DrawPolygon(ImageViewer viewer)
{
// TODO draw polygon on ImageViewer viewer, using calls to viewer.line()
}
```

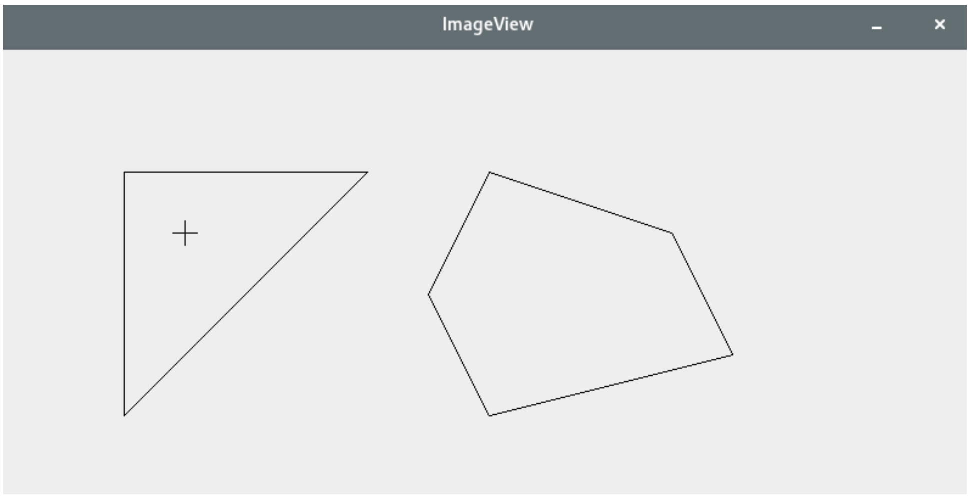
The function `DrawPolygon` should make use of the `line` function, shown above, to draw all lines of the polygon onto the viewer.

After you implemented the `DrawPolygon` function in the Polygon class. You can implement the "draw" command in the main function in Main.java.

You should implement the functionality in

```
...
else if (cmd.euqals("draw")) { // Check if point is
                               // on an existing island
    // TODO implement this
}
...
```

In the "draw" branch of the if-else block, you implement the code which will draw all islands/polygons and also the last point entered with the "onisland" command. First, create a new ImageViewer as in the example code above. Then iterate through all islands in the islands list and call on the shape of each island the DrawPolygon(viewer) function. Once you have drawn all islands/polygons. Also draw the last point entered through the "onisland" command by calling viewer.cross(px,py). Update the viewer through the update() function and you should see a visual representation of all islands and the point similar to the picture below:



This exercise can not be evaluated with the judge. If you have questions about your solution feel free to contact your teaching assistant.