

In this lab, we implement the classes 'GeometricalPoints' and 'DistanceMatrices' that we design in Seminar 1. The aim of this programme is to get the geometric points of some cities, and then calculate the distance that it is between them. For each class we have to write all its attributes in private and then, the methods in public.

For 'GeometricPoint', the attributes are the two coordinates, and its name. The main method in this class is the one that we use to calculate the distance between the two points.

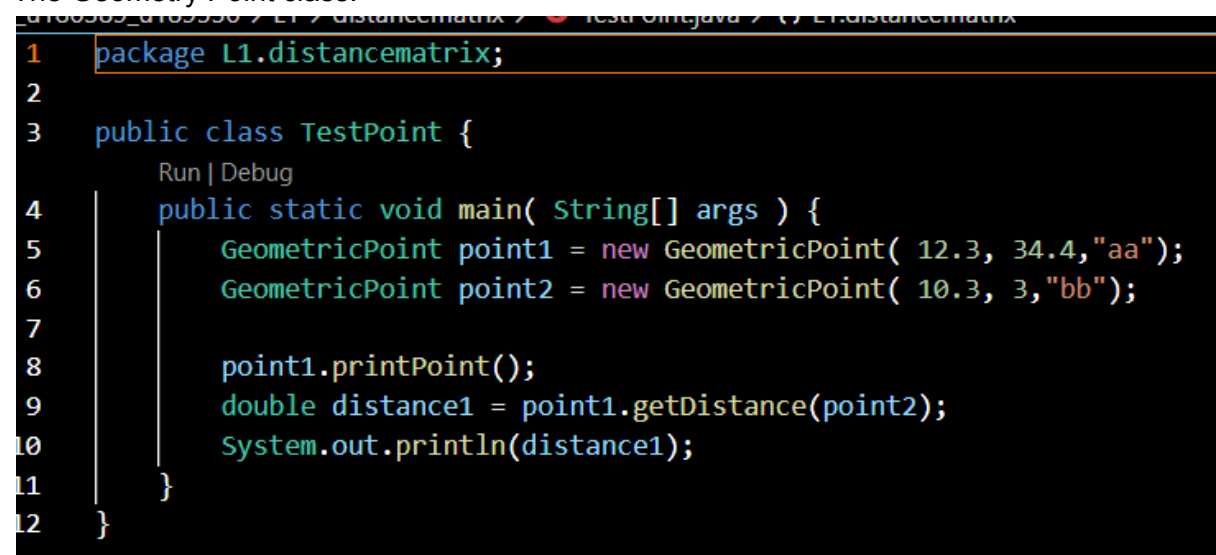
In 'MatrixDistance', the attributes are the points, that are the linked list of the class geometric points and then we have the matrix that is the array of the arrays. The methods in this class are the following ones: DistanceMatrix(), addCity(x,y,name), getCityName(index), getNoOfCities(),createDistanceMatrix() and finally the getDistance().

The part where we discussed most was whether we use the static linked list or dynamic linked list. The difference between these two linked lists is that in the static data structure, you have the memory fixed, whereas in Dynamic data Structure, the size of the memory can be updated during the execution time. Knowing that, we decided to work with the dynamic linked list, so we control the memory of the code while running it.

Once we decided which one we were going to use, we needed to implement the 'DistanceMatrices' class with the linked list. At first it was a little bit complicated, but with some time we understood it and we obtained the result that we were asked to.

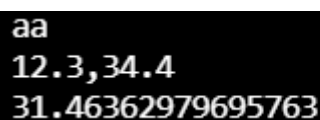
For each class we implemented a test file, where we can see if the classes were correctly programmed. With that implementation we saw that our classes are correctly done. Here we will attach some photos of the results.

The Geometry Point class:

A screenshot of a Java IDE with a dark theme. The code is for a class named TestPoint in the package L1.distancematrix. It contains a main method that creates two GeometricPoint objects, point1 (12.3, 34.4, "aa") and point2 (10.3, 3, "bb"). It then calls point1.printPoint() and calculates the distance between point1 and point2 using point1.getDistance(point2), printing the result to the console.

```
1 package L1.distancematrix;
2
3 public class TestPoint {
4     Run | Debug
5     public static void main( String[] args ) {
6         GeometricPoint point1 = new GeometricPoint( 12.3, 34.4,"aa");
7         GeometricPoint point2 = new GeometricPoint( 10.3, 3,"bb");
8
9         point1.printPoint();
10        double distance1 = point1.getDistance(point2);
11        System.out.println(distance1);
12    }
13 }
```

The result

A screenshot of the program's output in a console window. It shows the name of the first point, its coordinates, and the calculated distance between the two points.

```
aa
12.3,34.4
31.46362979695763
```

The Distance Matrix class;

```
00505_0105550 / L1 / distancematrix / TestDistanceMatrix.java / 17
package L1.distancematrix;

public class TestDistanceMatrix{
    Run | Debug
    public static void main(String[]args){
        DistanceMatrix p1 = new DistanceMatrix();

        p1.addCity(0,0,"Madrid");
        p1.addCity(0,750, "Barcelona");
        System.out.println(p1.getCityName(0));
        System.out.println(p1.getNoOfCities());
        System.out.println(p1.getDistance(0,1));
    }
}
```

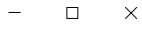
The result

```
Madrid
2
750.0
```

The 'optional' DisplayMatrix class

```
1 package L1.distancematrix;
2
3 public class TestDisplayMatrix {
    Run | Debug
4     public static void main(String[] args) {
5         DistanceMatrix matrix = new DistanceMatrix();
6         DisplayMatrix display = new DisplayMatrix(matrix);
7         display.setVisible(true);
8     }
9 }
```

The result



x-coord

y-coord

name

Enter new point!

	Barcelona	Madrid	Bilbao
Barcelona	0,00	180	424
Madrid	180	0,00	602
Bilbao	424	602	0,00