Ayesha Quinto Gammuac 241789
Minh Nguyen Phuong 241841

24292 - Object-Oriented Programming

LAB 1: Implementing a class

REPORT

THE PROGRAM

1. MAIN PURPOSE AND DESCRIPTION

The aim of this report is to describe how we have done lab 1, including all the problems we encountered and our solutions. The objective of the program is to create a matrix of distances. In this lab, we implemented classes representing geometrical points and distance matrices. As a result, the program should print a matrix with cities introduced by the user and the distance between each of them. There are two classes that we had to define: *DistanceMatrix.java* and *GP.java* (GeometricPoint). Also, two classes were given by the teacher: *Matrix.java* and *DisplayMatrix.java*. And we also had to create 3 more classes to test the first two.

1.1. Geometric Point

To represent the cities, we created the geometric point class since a city in a map can be described as a point of two coordinates. In GP class, we have three private attributes which are *xCoor* (x coordinate), *yCoor* (y coordinate) and *name* (name of the city). We used the constructor method to create the instance and also getter methods to get each attribute of the point. Then, a method to get the distance between two geometrical points using themathematical formula. And also a method to print the attributes of the point. Once this class was implemented, we created another class called *TestPoint.java* to check if the class was correct. We used the static method with instances of GP created with the constructor and printed all the results of these methods to the screen and it successfully worked.

1.2. Distance Matrix

Next, we had to implement a class that represents the matrix containing the distances: *DistanceMatrix.java*. In this class, we followed the instructions and used the existing Java class Linked List to create two attributes: *matrix* and *listOfCities*. The constructor in this case is a void method since the *matrix* and *listOfCities* are already

declared and initialized at the beginning of the program. Then, we did similar with how we implemented the GP class; with getter methods for the city name *getCityName()*, the total number of cities *getNoOfCities()* and a specific distance *getDistance()*. A function *addCity()* to add cities into the list was necessary for later to be able to construct the matrix. Thus, we also created a function named *createDistanceMatrix()* which is called after adding a city in the *listOfCities*. This method uses the loop "*for*" to add a list of distances into an element in the *matrix*, in other words, for each city added it creates a list of distances between the rest of the cities and adds this list to the main list of the *matrix*. Finally, we did the same as with the GP class, we created a class called *TestMatrix.java* to check if the *DistanceMatrix* class had been correctly implemented.

1.3. Display Matrix

We also proceeded to do the optional and final part of the lab: to display the Distance Matrix. The purpose of this class *DisplayMatrix.java* is to show the user a graphical interface where points can be introduced and the matrix will be updated. For that, we followed the instructions which were to put all the files of the program on the same package (package *distancematrix*) and connect the *DistanceMatrix.java* with the *Matrix.java* one, using "DistanceMatrix *implements* Matrix" in the header of the *DistanceMatrix* class. After checking that the definitions for both classes were the same (names, arguments and types), we created the *TestDisplayMatrix.java* with the given code in the pdf and successfully finished the program.

2. POSSIBLE SOLUTIONS AND PROBLEMS ENCOUNTERED

There is more than one possible solution for this program that we thought of. On one hand, for the list of cities in the class *DistanceMatrix*, the possible solution we came up with was using a *LinkedList* of Geometric Points. It is the most convenient way to declare this list since we can use the methods from the already created class *LinkedList* (functions such as add, size and get).

On the other hand, our team discussed that for the distance *matrix* of cities we could use either a matrix of doubles which is declared as matrix[][] or a Linked List of Linked Lists of doubles, that is, LinkedList<LinkedList<Double>>. We first tried the first option since it seemed more simple to use. However, we found that the matrix of doubles needs to have limitations indicated before calling it. This meant that we had to declare the matrix with a fixed size from the beginning but that is not functional for our program since the main purpose is to have the distances between as many cities as the user introduces.

Therefore, we could not have a fixed size for the matrix. The only way we could think of to solve this problem was to use Linked Lists because these types of lists needn't have a fixed size. We proceeded to declare our matrix as a Linked List which elements were also Linked Lists but of doubles, creating sort of a matrix (a list of lists). This approach, as we have stated before, has the advantage of using the incorporated methods of the LinkedList class and that can make the program more efficient, short and understandable.

After deciding which was the most efficient way to approach the attributes of the main program *DistanceMatrix*, the rest of the solution was to implement the necessary methods for each class: the constructors, the getters and more other functions that we have explained (1).

3. CONCLUSION

In general, the lab has gone quite well. However, we were a bit struggling with the Linked List in DistanceMatrix part as we mentioned above (2), but we were able to fix it in the end, trying to find other ways to define some methods or declaring a variable and solving the errors encountered in the process.

Throughout the lab, we had learned important points to follow when creating a program. First, to think of the design before actually starting the code, that is, to think what classes we need, how they are connected, what would their attributes and main methods be, etc. Second, when we start to code, we need to check the implementation of the different classes which each of them use different methods but some can be very similar. Therefore, the types of arguments, types of returns if they have, the names, etc need to be revised since any little mistake can make the program not work. Third and final, it is crucial before, during and after creating the program to understand what we are actually doing and coding, understand how we have implemented the solution and if something could be modified to make it easier or more efficient. Also, trying different approaches is a good way to find weak points or holes in our code.