Assignment Review Paper

To complete the second assignment for OOP, processing multiple files and extracting data to a graph, which stored information about nodes and edges was necessary. Nodes represent 15 locations in Sarajevo between which shortest paths are calculated. Different types of constraints are also taken into account, causing certain paths to be blocked.

My approach to solving this task was to start with creating a graph. I have created a class, *Graph.java*, which initially consisted only of its parameters, number of nodes and an adjacency list, private class Node, a constructor, and two methods - *addNode*() and *printGraph*(). To save data from files to a graph, I created a method *createGraph*() in main. After that, I added another class - *ShortestPath.java,* with a method that applies Dijkstra's algorithm. To perform this, the program uses a PriorityQueue to keep track of nodes to be explored next, to ensure that the path with the shortest distance from the current node is explored first. *Distance* and *parent* maps store the current shortest distance from the source to each node and parent for each node, respectively. Method *shortestPath*() traverses the graph keeping information in both the maps and the queue. To test the functionality of this method, I have also implemented four tests that check if the algorithm finds a path between all nodes, and how the algorithm performs if inadequate variables are input (for example a node not present in places map). Finally, I have added a couple of functions to read and apply constraints. For each graph, a random variable is calculated, depending on which constraints are either applied or not applied.

For me, the most challenging aspects of this task were deciding how to apply constraints to a complete graph, and devising a strategy to produce output. Because of certain tendencies to overcomplicate, my thought process and strategy might appear strange. Deciding what exactly to test in my class was also difficult. Additionally, I also struggled with error correction.

Initially, I believed that creating a graph and writing the *shortestPath*() method would be more difficult than it was. Writing the Graph class was rather easy, and after some research, Dijkstra's algorithm was not too difficult to implement either.

Appropriately resetting my report in test cases, and in my code in general, was not something I expected to struggle with. Writing tests also appeared as an easy concept, but was something I had difficulty understanding thoroughly. Creating a Maven project is also worth mentioning. I rearranged the locations of files, and tests were not recognized.

If I started this assignment again, I would probably change the return types of certain methods and modify them in general. Perhaps I would omit certain functions to simplify and clean up the code. I believe my approach to the task would be similar.