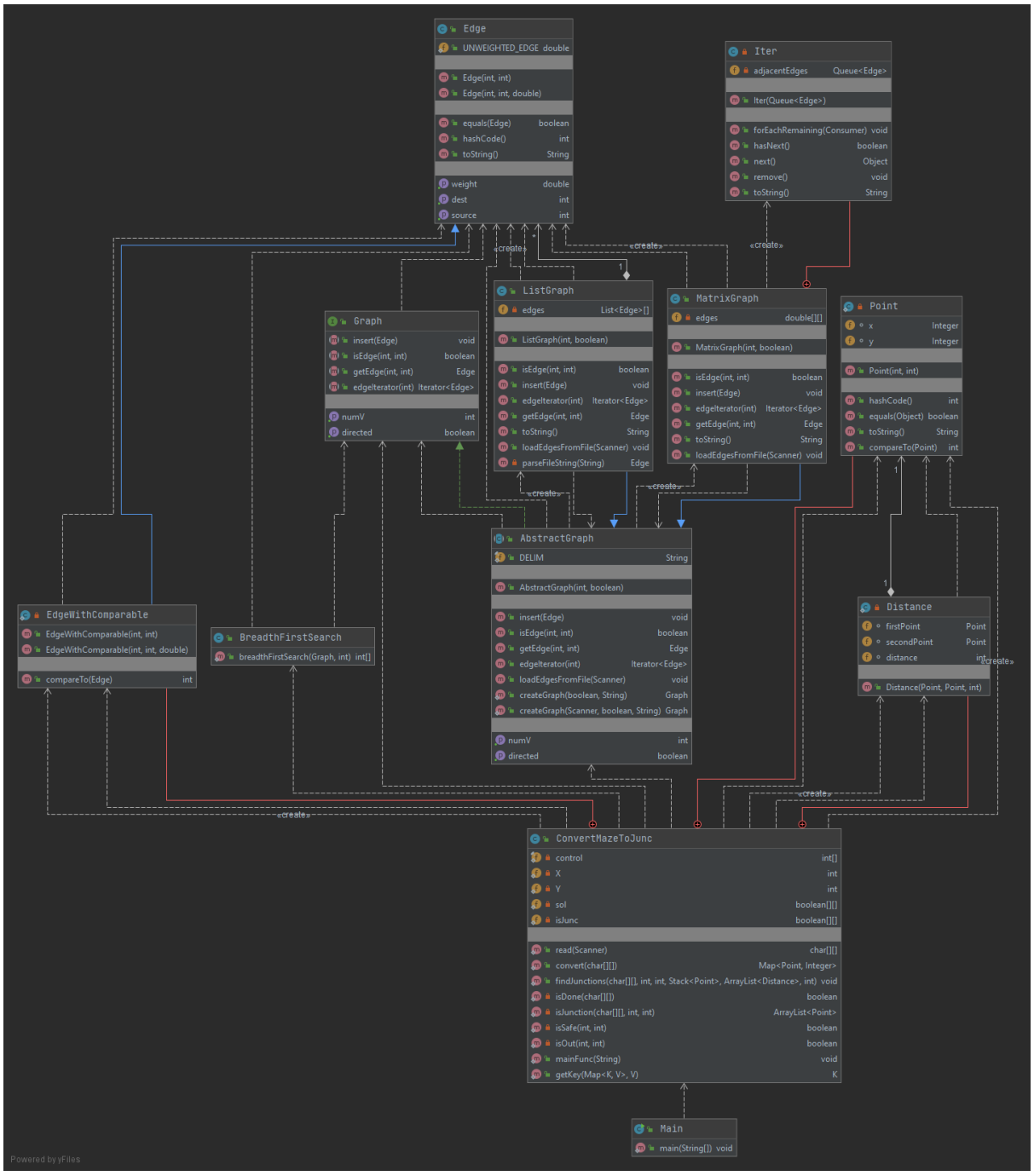
GIT Department of Computer Engineering
CSE 222/505 - Spring 2020
Homework 8 Question 3 Report

Melihcan Çilek
1801042092

1. CLASS DIAGRAM



I will put a copy of class diagram to this file if there is a problem about not being able to see some class fields or methods.

,

2. PROBLEM SOLUTION APPROACH

In this question, we asked for creating a Maze Solver which solves a maze. We have to read maze that is constructedby binary representation where 1's represents closed ways and 0's represents ways that we can go from. We have to convert that maze to weighted graph and we have to find best way that goes to end. For this representation, there are some difficulties such as converting this maze to graph and after this conversion, getting parent array with Breath First Search and after than, give it to stack and get order of traverse then printing it. While converting this maze into graph, I used depth first search perspective which means, while traversing 0 cells, if isJunction function returns true, I add that point into stack that's number is same number as number of ways that depth first search can go so depth first search traverse these ways consequently. While traversing graph, program marks sol boolean array's as true so program can understand whether it come from that position or not. In second array, isJunc, keeps if a cell is junction or not. Reason I have to keep sol and isJunc boolean arrays is that, when program traverse, program have to differenciate if that cell is a junction or not junction. Because of that, I seperated them. If program comes to a junction, program adds that Distance class variable into Arraylist so I can keep all of the distances between two points. After getting all distances between all nodes,program puts them into Set so all nodes are unique node. After putting into Set, program maps all of these nodes into an integer value. After map all of the vertex's, I search them in Distance Arraylist. If program finds edge with that vertex, I write them into a file named graphWillBeRead.txt . After writing all vertex's with weights, I give scanner of that file so program gives parent array. With pushing parent's into stack, program finds the route of junctions from beginning to the end.

3. TEST CASES

For testing, when compiling file, user should give name of file as compilation parameter so program can take it as input.

Example:

```
javac Main.java

java Main graph.txt
```

4. RUNNING AND RESULTS

For given example at the pdf, output of given file is at the below

```
Successfully wrote to the file.
The shortest path is:
Point{x=0, y=0}
Point{x=1, y=1}
Point{x=1, y=5}
Point{x=7, y=1}
Point{x=11, y=5}
Point{x=17, y=3}
Point{x=21, y=11}
Point{x=15, y=11}
Point{x=17, y=15}
Point{x=21, y=17}
```