GIT Department of Computer Engineering

CSE 222/505 - Spring 2020

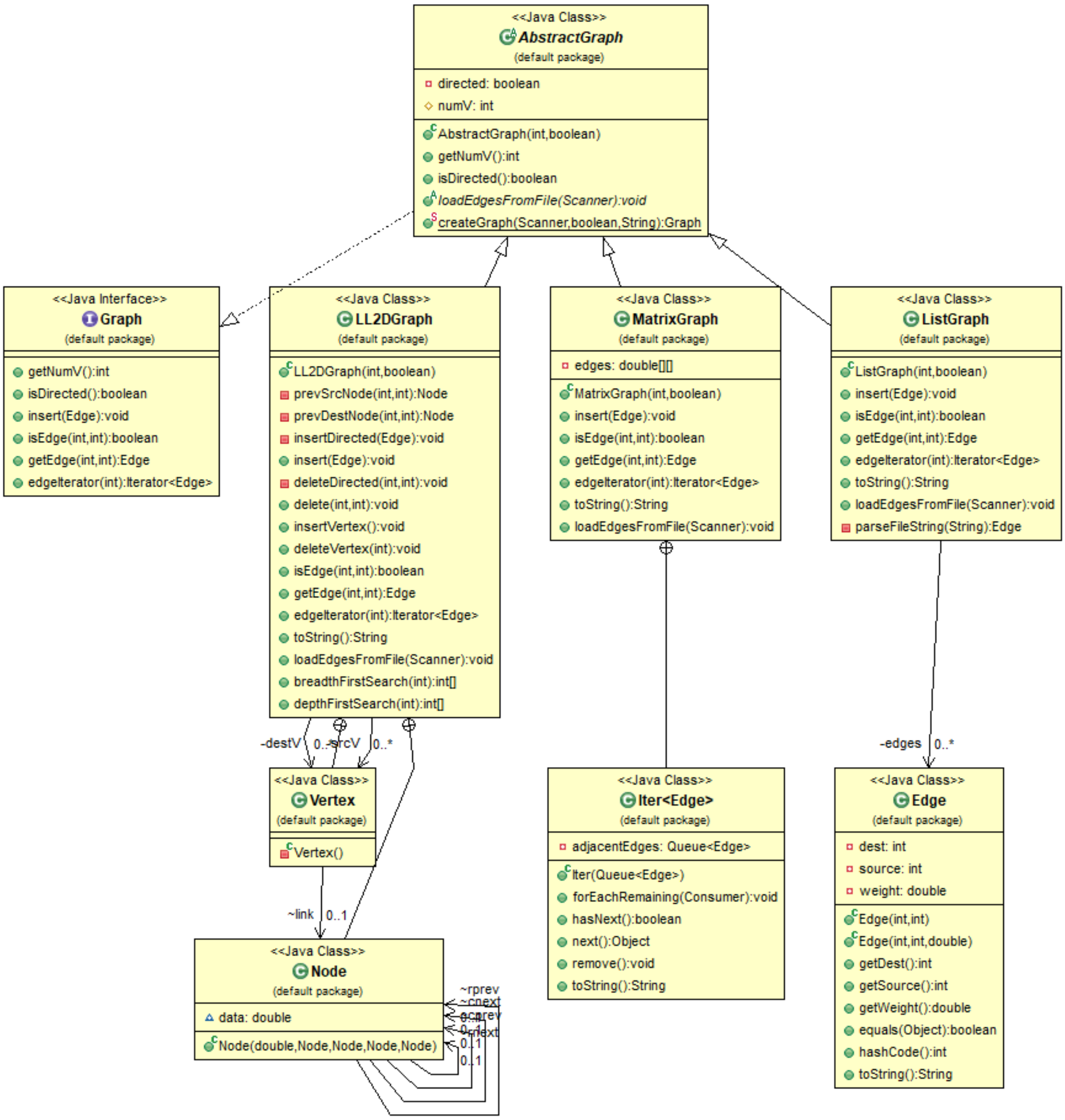
Homework 8 Report

Berke Belgin

171044065

Q2:

Class Diagram:



Problem Solution Approach:

I have 2 linked lists in my graph. One for sources and the other is for destinations. Sources linked list represents the header column and destinations linked list represents header row of a 2d matrix. Every node has 4 links which directs to upper, lower, left, right nodes of itself. When inserting and removing new elements the most trick part was handling with the situations where one of these nodes are null. There is a lot of situation that cause a null pointer exception, so I fixed them. I used two helper methods. One of them is a method that gets previous column of a given node, and the other is its row version. The rest was easy after achieving these steps.

Running Command Results:

```
Insert: Undirected - 0
Node 0-->
         node: 0 is not adjacent to 0
         node: 1, weight: 0.44
         node: 2, weight: 0.67
         node: 3, weight: 0.04
         node: 4, weight: 0.33
         node: 5 is not adjacent to 0
Node 1-->
         node: 0, weight: 0.44
         node: 1 is not adjacent to 1
         node: 2, weight: 0.94
         node: 3 is not adjacent to 1
         node: 4 is not adjacent to 1
         node: 5 is not adjacent to 1
Node 2-->
         node: 0, weight: 0.67
         node: 1, weight: 0.94
         node: 2 is not adjacent to 2
         node: 3 is not adjacent to 2
         node: 4, weight: 0.3
         node: 5 is not adjacent to 2
Node 3-->
         node: 0, weight: 0.04
         node: 1 is not adjacent to 3
         node: 2 is not adjacent to 3
         node: 3 is not adjacent to 3
         node: 4 is not adjacent to 3
         node: 5 is not adjacent to 3
Node 4-->
         node: 0, weight: 0.33
         node: 1 is not adjacent to 4
         node: 2, weight: 0.3
         node: 3 is not adjacent to 4
         node: 4 is not adjacent to 4
         node: 5 is not adjacent to 4
Node 5-->
         node: 0 is not adjacent to 5
         node: 1 is not adjacent to 5
         node: 2 is not adjacent to 5
         node: 3 is not adjacent to 5
         node: 4 is not adjacent to 5
         node: 5 is not adjacent to 5

Delete: Undirected - 0
Node 0-->
         node: 0 is not adjacent to 0
         node: 1 is not adjacent to 0
         node: 2 is not adjacent to 0
         node: 3, weight: 0.04
         node: 4 is not adjacent to 0
         node: 5 is not adjacent to 0
Node 1-->
         node: 0 is not adjacent to 1
         node: 1 is not adjacent to 1
         node: 2, weight: 0.94
         node: 3 is not adjacent to 1
         node: 4 is not adjacent to 1
         node: 5 is not adjacent to 1
Node 2-->
         node: 0 is not adjacent to 2
         node: 1, weight: 0.94
         node: 2 is not adjacent to 2
         node: 3 is not adjacent to 2
         node: 4, weight: 0.3
         node: 5 is not adjacent to 2
Node 3-->
         node: 0, weight: 0.04
         node: 1 is not adjacent to 3
         node: 2 is not adjacent to 3
         node: 3 is not adjacent to 3
         node: 4 is not adjacent to 3
         node: 5 is not adjacent to 3
Node 4-->
         node: 0 is not adjacent to 4
         node: 1 is not adjacent to 4
         node: 2, weight: 0.3
         node: 3 is not adjacent to 4
         node: 4 is not adjacent to 4
         node: 5 is not adjacent to 4
Node 5-->
```

```
         node: 0 is not adjacent to 5
         node: 1 is not adjacent to 5
         node: 2 is not adjacent to 5
         node: 3 is not adjacent to 5
         node: 4 is not adjacent to 5
         node: 5 is not adjacent to 5

Insert: Directed - 0
Node 0-->
         node: 0 is not adjacent to 0
         node: 1, weight: 0.31
         node: 2 is not adjacent to 0
         node: 3 is not adjacent to 0
         node: 4 is not adjacent to 0
         node: 5 is not adjacent to 0
Node 1-->
         node: 0, weight: 0.65
         node: 1 is not adjacent to 1
         node: 2 is not adjacent to 1
         node: 3 is not adjacent to 1
         node: 4 is not adjacent to 1
         node: 5 is not adjacent to 1
Node 2-->
         node: 0, weight: 0.97
         node: 1, weight: 0.22
         node: 2 is not adjacent to 2
         node: 3 is not adjacent to 2
         node: 4, weight: 0.21
         node: 5 is not adjacent to 2
Node 3-->
         node: 0 is not adjacent to 3
         node: 1 is not adjacent to 3
         node: 2, weight: 0.91
         node: 3 is not adjacent to 3
         node: 4 is not adjacent to 3
         node: 5 is not adjacent to 3
Node 4-->
         node: 0 is not adjacent to 4
         node: 1, weight: 0.34
         node: 2 is not adjacent to 4
         node: 3 is not adjacent to 4
         node: 4 is not adjacent to 4
         node: 5 is not adjacent to 4
Node 5-->
         node: 0 is not adjacent to 5
         node: 1 is not adjacent to 5
         node: 2 is not adjacent to 5
         node: 3 is not adjacent to 5
         node: 4 is not adjacent to 5
         node: 5 is not adjacent to 5

Delete: Directed - 0
Node 0-->
         node: 0 is not adjacent to 0
         node: 1, weight: 0.31
         node: 2 is not adjacent to 0
         node: 3 is not adjacent to 0
         node: 4 is not adjacent to 0
         node: 5 is not adjacent to 0
Node 1-->
         node: 0, weight: 0.65
         node: 1 is not adjacent to 1
         node: 2 is not adjacent to 1
         node: 3 is not adjacent to 1
         node: 4 is not adjacent to 1
         node: 5 is not adjacent to 1
Node 2-->
         node: 0, weight: 0.97
         node: 1 is not adjacent to 2
         node: 2 is not adjacent to 2
         node: 3 is not adjacent to 2
         node: 4, weight: 0.21
         node: 5 is not adjacent to 2
Node 3-->
         node: 0 is not adjacent to 3
         node: 1 is not adjacent to 3
         node: 2, weight: 0.91
         node: 3 is not adjacent to 3
         node: 4 is not adjacent to 3
         node: 5 is not adjacent to 3
Node 4-->
         node: 0 is not adjacent to 4
         node: 1 is not adjacent to 4
         node: 2 is not adjacent to 4
         node: 3 is not adjacent to 4
```

```
	node: 4 is not adjacent to 4
	node: 5 is not adjacent to 4
Node 5-->
	node: 0 is not adjacent to 5
	node: 1 is not adjacent to 5
	node: 2 is not adjacent to 5
	node: 3 is not adjacent to 5
	node: 4 is not adjacent to 5
	node: 5 is not adjacent to 5

Insert: Undirected - 1
Node 0-->
	node: 0 is not adjacent to 0
	node: 1, weight: 0.16
	node: 2 is not adjacent to 0
	node: 3 is not adjacent to 0
	node: 4, weight: 0.25
	node: 5 is not adjacent to 0
Node 1-->
	node: 0, weight: 0.16
	node: 1 is not adjacent to 1
	node: 2, weight: 0.05
	node: 3, weight: 0.21
	node: 4, weight: 0.25
	node: 5 is not adjacent to 1
Node 2-->
	node: 0 is not adjacent to 2
	node: 1, weight: 0.05
	node: 2 is not adjacent to 2
	node: 3, weight: 0.25
	node: 4 is not adjacent to 2
	node: 5 is not adjacent to 2
Node 3-->
	node: 0 is not adjacent to 3
	node: 1, weight: 0.21
	node: 2, weight: 0.25
	node: 3 is not adjacent to 3
	node: 4, weight: 0.03
	node: 5 is not adjacent to 3
Node 4-->
	node: 0, weight: 0.25
	node: 1, weight: 0.25
	node: 2 is not adjacent to 4
	node: 3, weight: 0.03
	node: 4 is not adjacent to 4
	node: 5 is not adjacent to 4
Node 5-->
	node: 0 is not adjacent to 5
	node: 1 is not adjacent to 5
	node: 2 is not adjacent to 5
	node: 3 is not adjacent to 5
	node: 4 is not adjacent to 5
	node: 5 is not adjacent to 5

Delete: Undirected - 1
Node 0-->
	node: 0 is not adjacent to 0
	node: 1 is not adjacent to 0
	node: 2 is not adjacent to 0
	node: 3 is not adjacent to 0
	node: 4 is not adjacent to 0
	node: 5 is not adjacent to 0
Node 1-->
	node: 0 is not adjacent to 1
	node: 1 is not adjacent to 1
	node: 2 is not adjacent to 1
	node: 3, weight: 0.21
	node: 4 is not adjacent to 1
	node: 5 is not adjacent to 1
Node 2-->
	node: 0 is not adjacent to 2
	node: 1 is not adjacent to 2
	node: 2 is not adjacent to 2
	node: 3 is not adjacent to 2
	node: 4 is not adjacent to 2
	node: 5 is not adjacent to 2
Node 3-->
	node: 0 is not adjacent to 3
	node: 1, weight: 0.21
	node: 2 is not adjacent to 3
	node: 3 is not adjacent to 3
	node: 4, weight: 0.03
	node: 5 is not adjacent to 3
Node 4-->
	node: 0 is not adjacent to 4
```

```
         node: 1 is not adjacent to 4
         node: 2 is not adjacent to 4
         node: 3, weight: 0.03
         node: 4 is not adjacent to 4
         node: 5 is not adjacent to 4
Node 5-->
         node: 0 is not adjacent to 5
         node: 1 is not adjacent to 5
         node: 2 is not adjacent to 5
         node: 3 is not adjacent to 5
         node: 4 is not adjacent to 5
         node: 5 is not adjacent to 5

Insert: Directed - 1
Node 0-->
         node: 0 is not adjacent to 0
         node: 1 is not adjacent to 0
         node: 2, weight: 0.6
         node: 3 is not adjacent to 0
         node: 4 is not adjacent to 0
         node: 5 is not adjacent to 0
Node 1-->
         node: 0 is not adjacent to 1
         node: 1 is not adjacent to 1
         node: 2, weight: 0.55
         node: 3, weight: 0.02
         node: 4 is not adjacent to 1
         node: 5 is not adjacent to 1
Node 2-->
         node: 0, weight: 0.11
         node: 1 is not adjacent to 2
         node: 2 is not adjacent to 2
         node: 3, weight: 0.29
         node: 4 is not adjacent to 2
         node: 5 is not adjacent to 2
Node 3-->
         node: 0 is not adjacent to 3
         node: 1 is not adjacent to 3
         node: 2 is not adjacent to 3
         node: 3 is not adjacent to 3
         node: 4 is not adjacent to 3
         node: 5 is not adjacent to 3
Node 4-->
         node: 0 is not adjacent to 4
         node: 1, weight: 0.41
         node: 2, weight: 0.78
         node: 3, weight: 0.6
         node: 4 is not adjacent to 4
         node: 5 is not adjacent to 4
Node 5-->
         node: 0 is not adjacent to 5
         node: 1 is not adjacent to 5
         node: 2 is not adjacent to 5
         node: 3 is not adjacent to 5
         node: 4 is not adjacent to 5
         node: 5 is not adjacent to 5

Delete: Directed - 1
Node 0-->
         node: 0 is not adjacent to 0
         node: 1 is not adjacent to 0
         node: 2, weight: 0.6
         node: 3 is not adjacent to 0
         node: 4 is not adjacent to 0
         node: 5 is not adjacent to 0
Node 1-->
         node: 0 is not adjacent to 1
         node: 1 is not adjacent to 1
         node: 2 is not adjacent to 1
         node: 3, weight: 0.02
         node: 4 is not adjacent to 1
         node: 5 is not adjacent to 1
Node 2-->
         node: 0 is not adjacent to 2
         node: 1 is not adjacent to 2
         node: 2 is not adjacent to 2
         node: 3, weight: 0.29
         node: 4 is not adjacent to 2
         node: 5 is not adjacent to 2
Node 3-->
         node: 0 is not adjacent to 3
         node: 1 is not adjacent to 3
         node: 2 is not adjacent to 3
         node: 3 is not adjacent to 3
         node: 4 is not adjacent to 3
         node: 5 is not adjacent to 3
Node 4-->
         node: 0 is not adjacent to 4
         node: 1, weight: 0.41
         node: 2, weight: 0.78
         node: 3, weight: 0.6
         node: 4 is not adjacent to 4
         node: 5 is not adjacent to 4
Node 5-->
         node: 0 is not adjacent to 5
         node: 1 is not adjacent to 5
         node: 2 is not adjacent to 5
         node: 3 is not adjacent to 5
         node: 4 is not adjacent to 5
         node: 5 is not adjacent to 5

Insert: Undirected - 2
Node 0-->
         node: 0 is not adjacent to 0
         node: 1, weight: 0.8
         node: 2 is not adjacent to 0
         node: 3, weight: 0.81
         node: 4 is not adjacent to 0
         node: 5 is not adjacent to 0
Node 1-->
         node: 0, weight: 0.8
         node: 1 is not adjacent to 1
         node: 2 is not adjacent to 1
         node: 3 is not adjacent to 1
         node: 4 is not adjacent to 1
         node: 5 is not adjacent to 1
Node 2-->
         node: 0 is not adjacent to 2
         node: 1 is not adjacent to 2
         node: 2 is not adjacent to 2
         node: 3, weight: 0.21
         node: 4, weight: 0.57
         node: 5 is not adjacent to 2
Node 3-->
         node: 0, weight: 0.81
         node: 1 is not adjacent to 3
         node: 2, weight: 0.21
         node: 3 is not adjacent to 3
         node: 4, weight: 0.06
         node: 5 is not adjacent to 3
Node 4-->
         node: 0 is not adjacent to 4
         node: 1 is not adjacent to 4
         node: 2, weight: 0.57
         node: 3, weight: 0.06
         node: 4 is not adjacent to 4
         node: 5 is not adjacent to 4
Node 5-->
         node: 0 is not adjacent to 5
         node: 1 is not adjacent to 5
         node: 2 is not adjacent to 5
         node: 3 is not adjacent to 5
         node: 4 is not adjacent to 5
         node: 5 is not adjacent to 5

Delete: Undirected - 2
Node 0-->
         node: 0 is not adjacent to 0
         node: 1, weight: 0.8
         node: 2 is not adjacent to 0
         node: 3, weight: 0.81
         node: 4 is not adjacent to 0
         node: 5 is not adjacent to 0
Node 1-->
         node: 0, weight: 0.8
         node: 1 is not adjacent to 1
         node: 2 is not adjacent to 1
         node: 3 is not adjacent to 1
         node: 4 is not adjacent to 1
         node: 5 is not adjacent to 1
Node 2-->
         node: 0 is not adjacent to 2
         node: 1 is not adjacent to 2
         node: 2 is not adjacent to 2
         node: 3 is not adjacent to 2
         node: 4, weight: 0.57
         node: 5 is not adjacent to 2
Node 3-->
         node: 0, weight: 0.81
         node: 1 is not adjacent to 3
```

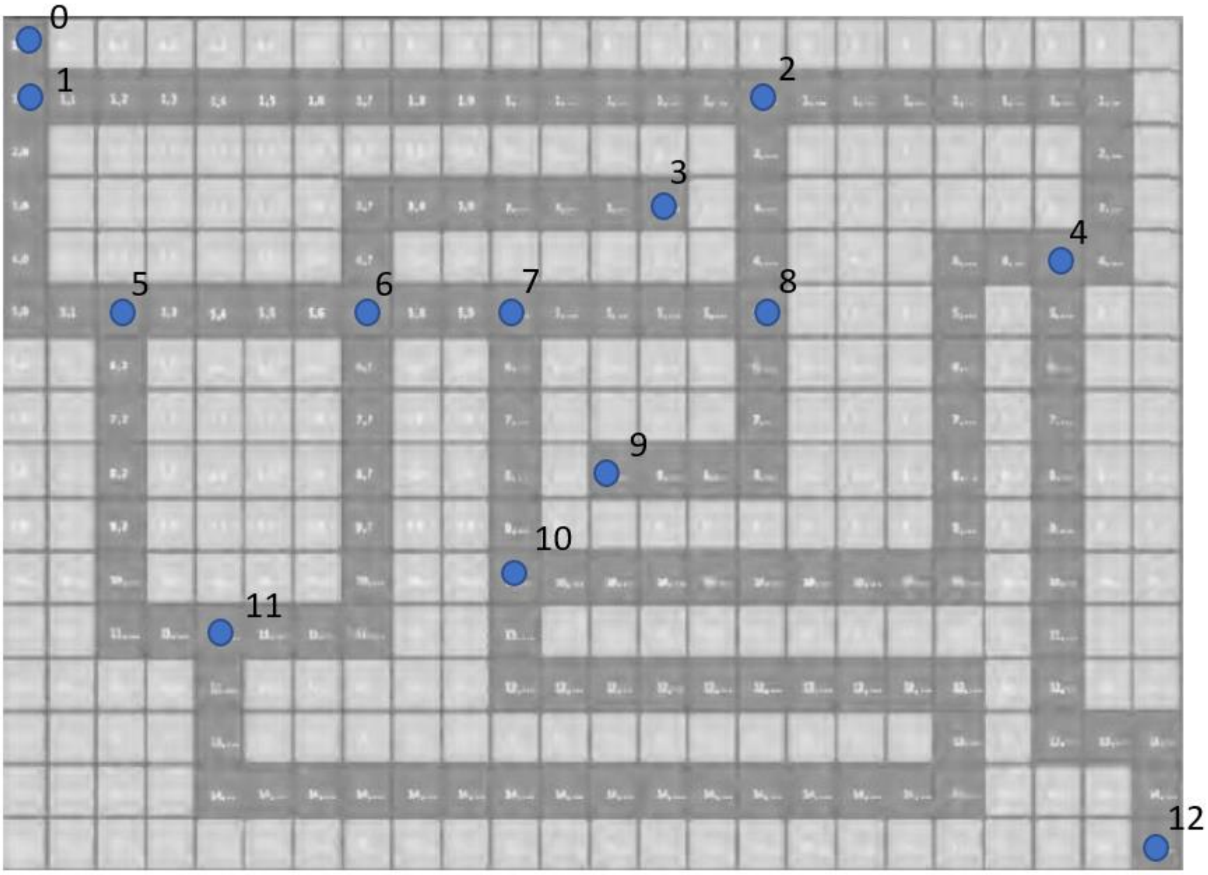
```
        node: 2 is not adjacent to 3
        node: 3 is not adjacent to 3
        node: 4 is not adjacent to 3
        node: 5 is not adjacent to 3
Node 4-->
        node: 0 is not adjacent to 4
        node: 1 is not adjacent to 4
        node: 2, weight: 0.57
        node: 3 is not adjacent to 4
        node: 4 is not adjacent to 4
        node: 5 is not adjacent to 4
Node 5-->
        node: 0 is not adjacent to 5
        node: 1 is not adjacent to 5
        node: 2 is not adjacent to 5
        node: 3 is not adjacent to 5
        node: 4 is not adjacent to 5
        node: 5 is not adjacent to 5

Insert: Directed - 2
Node 0-->
        node: 0 is not adjacent to 0
        node: 1 is not adjacent to 0
        node: 2, weight: 0.15
        node: 3 is not adjacent to 0
        node: 4, weight: 0.58
        node: 5 is not adjacent to 0
Node 1-->
        node: 0, weight: 0.93
        node: 1 is not adjacent to 1
        node: 2 is not adjacent to 1
        node: 3 is not adjacent to 1
        node: 4 is not adjacent to 1
        node: 5 is not adjacent to 1
Node 2-->
        node: 0 is not adjacent to 2
        node: 1, weight: 0.41
        node: 2 is not adjacent to 2
        node: 3 is not adjacent to 2
        node: 4 is not adjacent to 2
        node: 5 is not adjacent to 2
Node 3-->
        node: 0, weight: 0.37
        node: 1 is not adjacent to 3
        node: 2 is not adjacent to 3
        node: 3 is not adjacent to 3
        node: 4, weight: 0.69
        node: 5 is not adjacent to 3
Node 4-->
        node: 0 is not adjacent to 4
        node: 1, weight: 0.51
        node: 2 is not adjacent to 4
        node: 3 is not adjacent to 4
        node: 4 is not adjacent to 4
        node: 5 is not adjacent to 4
Node 5-->
        node: 0 is not adjacent to 5
        node: 1 is not adjacent to 5
        node: 2 is not adjacent to 5
        node: 3 is not adjacent to 5
        node: 4 is not adjacent to 5
        node: 5 is not adjacent to 5

Delete: Directed - 2
Node 0-->
        node: 0 is not adjacent to 0
        node: 1 is not adjacent to 0
        node: 2, weight: 0.15
        node: 3 is not adjacent to 0
        node: 4, weight: 0.58
        node: 5 is not adjacent to 0
Node 1-->
        node: 0 is not adjacent to 1
        node: 1 is not adjacent to 1
        node: 2 is not adjacent to 1
        node: 3 is not adjacent to 1
        node: 4 is not adjacent to 1
        node: 5 is not adjacent to 1
Node 2-->
        node: 0 is not adjacent to 2
        node: 1, weight: 0.41
        node: 2 is not adjacent to 2
        node: 3 is not adjacent to 2
        node: 4 is not adjacent to 2
        node: 5 is not adjacent to 2
```

```
Node 3-->
         node: 0, weight: 0.37
         node: 1 is not adjacent to 3
         node: 2 is not adjacent to 3
         node: 3 is not adjacent to 3
         node: 4, weight: 0.69
         node: 5 is not adjacent to 3
Node 4-->
         node: 0 is not adjacent to 4
         node: 1, weight: 0.51
         node: 2 is not adjacent to 4
         node: 3 is not adjacent to 4
         node: 4 is not adjacent to 4
         node: 5 is not adjacent to 4
Node 5-->
         node: 0 is not adjacent to 5
         node: 1 is not adjacent to 5
         node: 2 is not adjacent to 5
         node: 3 is not adjacent to 5
         node: 4 is not adjacent to 5
         node: 5 is not adjacent to 5
```

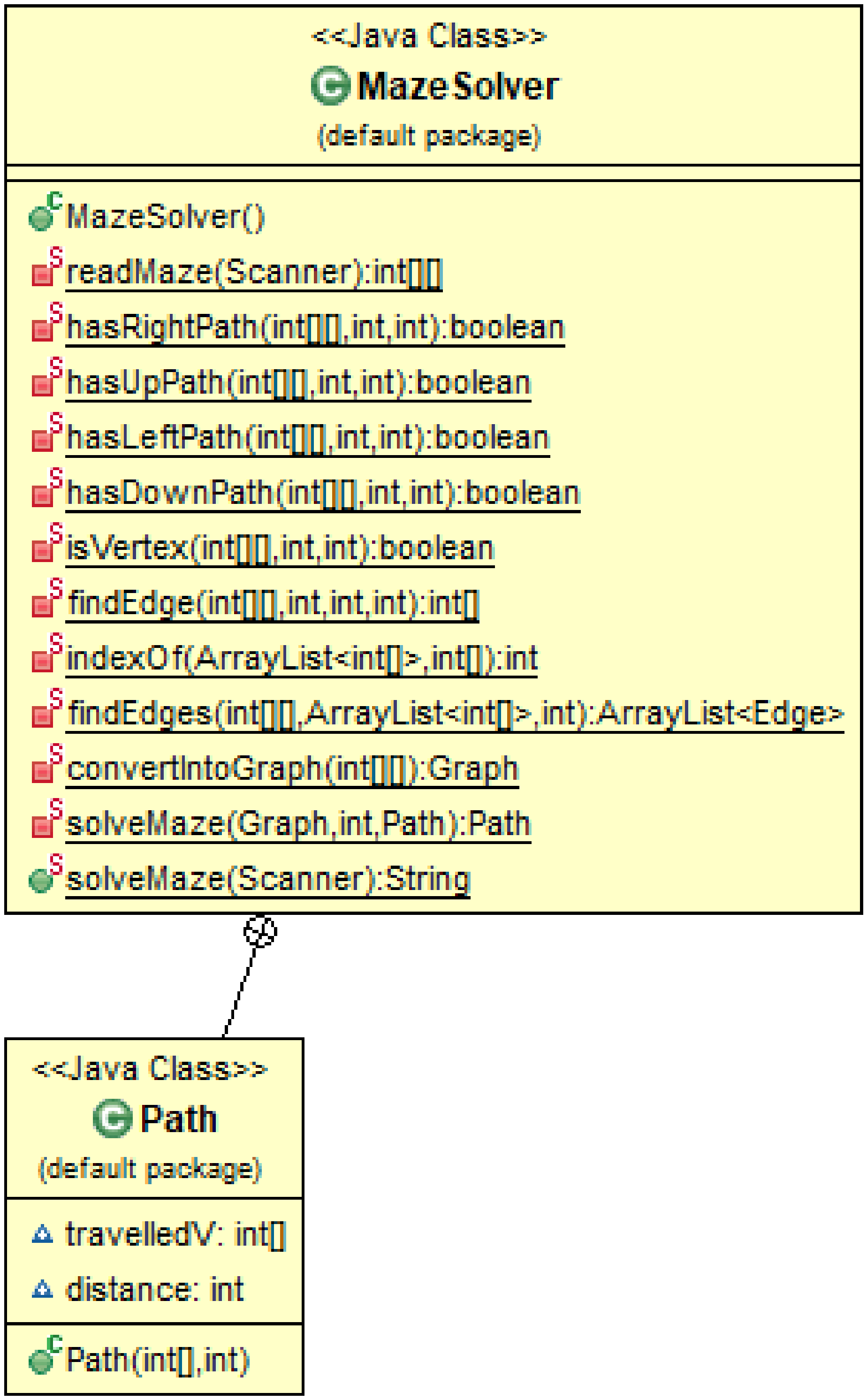
Test Cases:

Test ID	Test Case	Pass/Fail
1	Insert into first row	Pass
2	Insert into first column	Pass
3	Delete a node from the first row	Pass
4	Delete a node from the first column	Pass

Q3:



Class Diagram:


<<Java Class>>
MazeSolver
(default package)
MazeSolver()
readMaze(Scanner):int[][]
hasRightPath(int[][],int,int):boolean
hasUpPath(int[][],int,int):boolean
hasLeftPath(int[][],int,int):boolean
hasDownPath(int[][],int,int):boolean
isVertex(int[][],int,int):boolean
findEdge(int[][],int,int,int):int[]
indexOf(ArrayList<int[]>,int[]):int
findEdges(int[][],ArrayList<int[]>,int):ArrayList<Edge>
convertIntoGraph(int[][]):Graph
solveMaze(Graph,int,Path):Path
solveMaze(Scanner):String
<<Java Class>>
Path
(default package)
travelledV: int[]
distance: int
Path(int[],int)

Problem Solution Approach:

I implemented this part with a class which has only static methods since there is no need to instantiate the class. I firstly read and write the maze into a 2d integer array. Then I created a vertex array list by checking every block in the maze if any block has more than or less than 2 adjacent blocks, if so I added the coordinate into array list. After spotting every vertex in the maze, I traveled every direction possible from every vertex until reaching another vertex. This way I spotted every edge in the maze. Then I created a graph and added every edge I found into the graph. After creating the graph I implemented a recursive function to travel every adjacent vertex of a vertex by remembering the previous visited vertices and not passing the same vertex twice. When found the and, if there is no other solution found, it is the result else which of the results has less distance, it is the result.

Running Command Results:

```
[0, 1, 2, 4, 12] (41)
```

Test Cases:

Test ID	Test Case	Pass/Fail
1	Test solver with given maze	Pass