# GTU Department of Computer Engineering

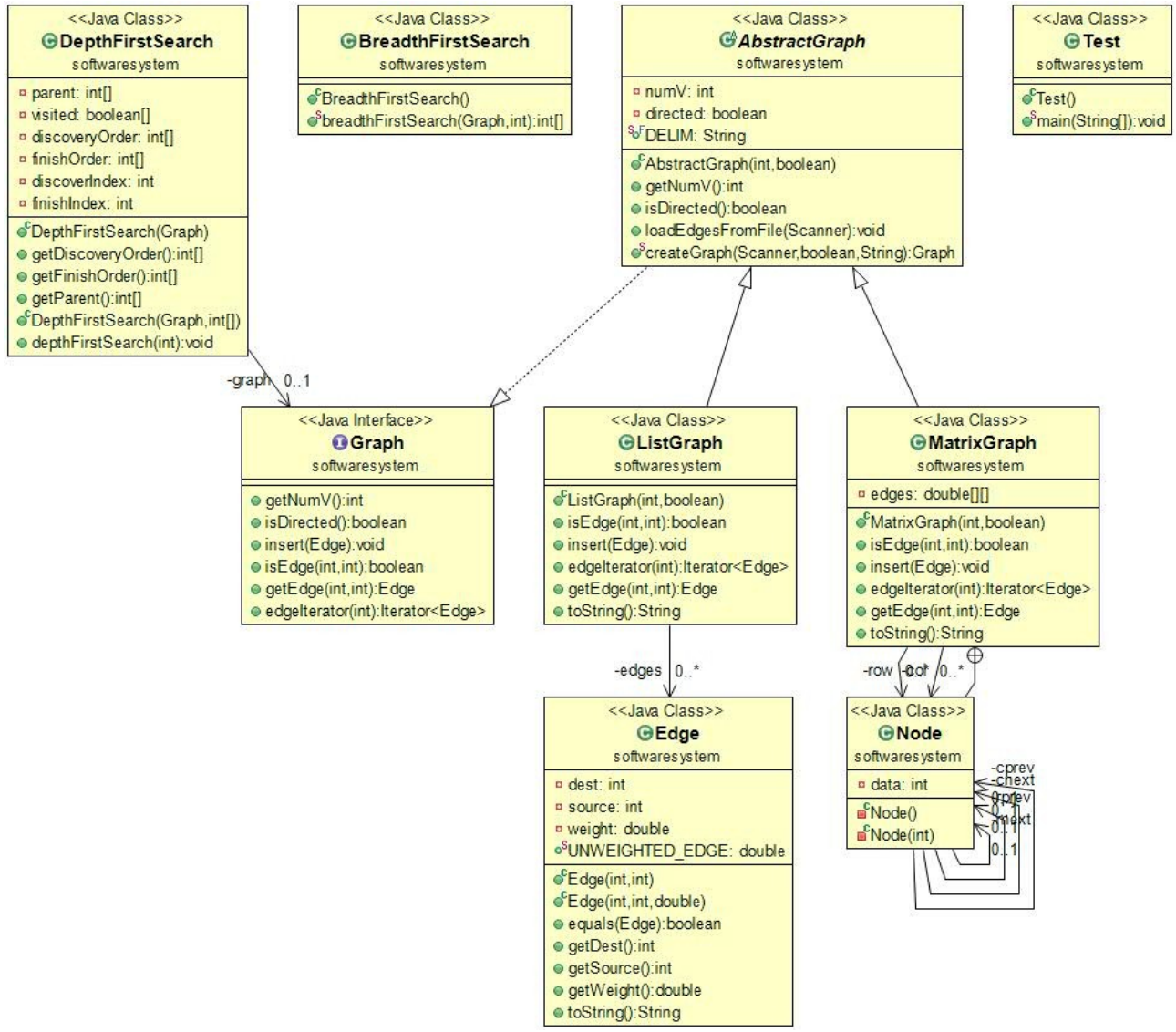
CSE 222/505 - Spring 2020

Homework 8-Report

Mert Can BEŞİRLİ
1801042663

# #Question 2#

# 1. CLASS DIAGRAMS




# 2. PROBLEM SOLUTION APPROACH

In this problem, we implemented the graph structure with 2d linked list. There are two links for the line node, rnext and rprev. There are two links for the line node, cnext and cprev.We proceeded using the matrix graph and changed the inside of the matrix graph class. We have added and searched process. Breadth-first search of the graph and depth-first search of the graph.

# 3. TEST CASES

| Test Case ID | Test Scenario | Test Steps | Test Data | Excepted Results | Actual Results | Pass/Fail |
|---|---|---|---|---|---|---|
| T1 | Check create undirected graph and inserting Edge(source and vertex) | Enter elements for inserting Vertices number is 6 | (0,1),(1,2),(2,3),(3,4),(4,5) | Insertig elements successfully and printing | As Excepted | Pass |
| T2 | Check create directed graph and inserting Edge(source and vertex) | Enter elements for inserting Vertices number is 6 | (0,1),(1,2),(2,3),(3,4),(4,5) | Insertig elements successfully and printing | As Excepted | Pass |
| T3 | Check breadth first search | Parameter as Graph | (0,1),(1,2),(2,3),(3,4),(4,5) | Searching Results Showing | As Excepted | Pass |
| T4 | Check depth first search | Parameter as Graph | (0,1),(1,2),(2,3),(3,4),(4,5) | Searching Results Showing Discovery Order and Finish Order | As Excepted | Pass |

# 4. RUNNING AND RESULTS

```
For Rows Lists
[0]--> [0, 1]
[1]--> [1, 2]
[2]--> [2, 3]
[3]--> [3, 4]
[4]--> [4, 5]
For column Lists
[1]--> [0, 1]
[2]--> [1, 2]
[3]--> [2, 3]
[4]--> [3, 4]
[5]--> [4, 5]

BreadthFirstSearch:
0, 1, 2, 3, 4,
DepthFirstSearch:
Discovery Order: 0, 1, 2, 3, 4, 5,
Finish Order: 5, 4, 3, 2, 1, 0,
```

```
For Rows Lists
[0]--> [0, 1]
[1]--> [1, 0]
[1, 2]
[2]--> [2, 1]
[2, 3]
[3]--> [3, 2]
[3, 4]
[4]--> [4, 3]
[4, 5]
[5]--> [5, 4]
For column Lists
[0]--> [1, 0]
[1]--> [0, 1]
[2, 1]
[2]--> [1, 2]
[3, 2]
[3]--> [2, 3]
[4, 3]
[4]--> [3, 4]
[5, 4]
[5]--> [4, 5]

BreadthFirstSearch:
0, 1, 2, 3, 4,
DepthFirstSearch:
Discovery Order: 0, 1, 2, 3, 4, 5,
Finish Order: 5, 4, 3, 2, 1, 0,
```