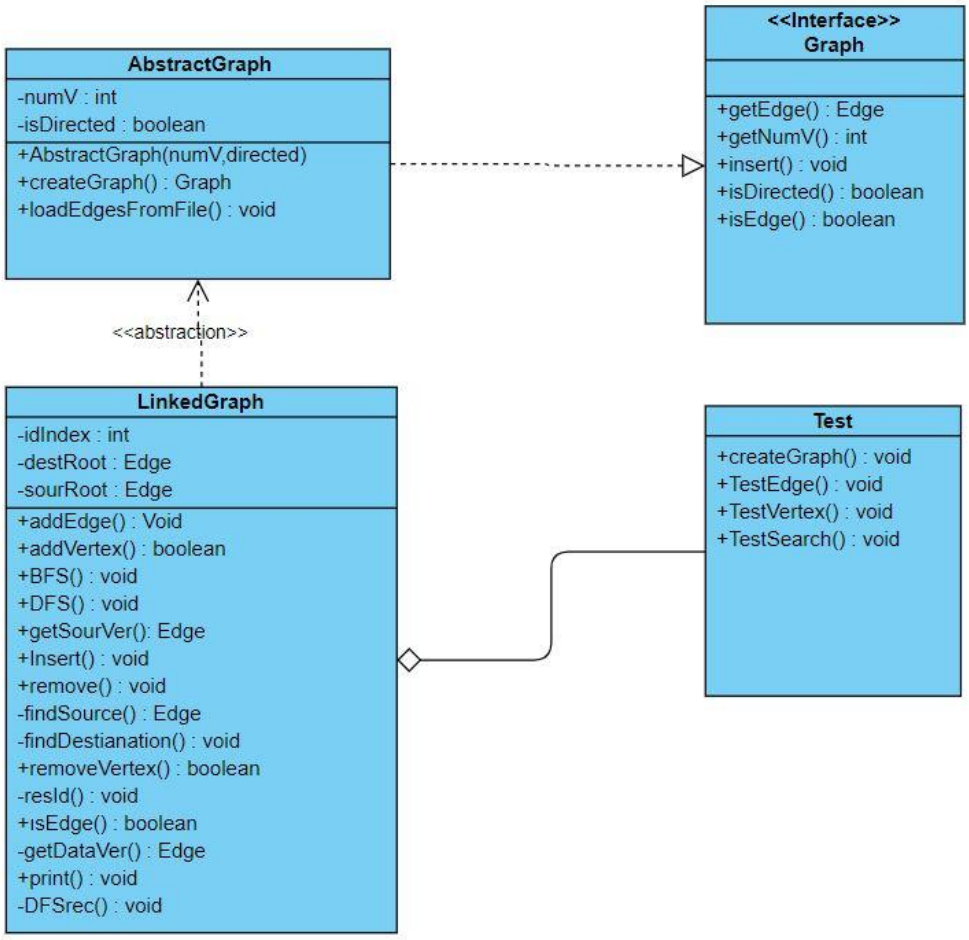
GIT Department of Computer Engineering
CSE 222/505 - Spring 2020
Homework 8 Report

Oğuzhan SEZGİN
1801042005

Q2

I added 2 new parameters to the constructor of the graph class which implement in book to make the tests easier. One of these parameters is the array where the vertices of the graph is held, and the other is the number of vertex. So instead of receiving the vertex number and vertex from the user, I quickly sent them as parameters in the tests.

CLASS DIAGRAM



PROBLEM SOLUTION APPROACH

If user wants to delete non-exist vertex program checks the vertex is exist then return false.

If user wants to add exist vertex , program checks is the vertex is exist then return false.

If user wants to add edge without vertices program checks if the vertices exist and do nothing if does not exist.

If user wants to remove not-exist edge , program checks if the edge exist and do nothing if does not exist.

if user wants to start traverse from vertex which does not exist , program prints nothing.

TEST CASES

VERTEX TEST

Test Case Name	**Test Step**	**Action**	**Test Data**	**Expected Result**	**Test Result**	**Note**
Add Vertex	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
Add Vertex	2.	Choose “addVertex” method	“v1”	System should add vertex to graph	System add vertex to graph	

Test Case Name	**Test Step**	**Action**	**Test Data**	**Expected Result**	**Test Result**	**Note**

Add Exist Vertex	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
Add Exist Vertex	2.	Choose “addVertex” method	“v1”	System should return false	System return false	

Test Case Name	**Test Step**	**Action**	**Test Data**	**Expected Result**	**Test Result**	**Note**
Remove Vertex	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
Remove Vertex	2.	Choose “removeVertex” method	“v1”	System should remove vertex	System remove vertex	

Test Case Name	**Test Step**	**Action**	**Test Data**	**Expected Result**	**Test Result**	**Note**
Remove Non-Exist Vertex	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
Remove Non-Exist Vertex	2.	Choose “removeVertex” method	“v1”	System should return false	System return false	

EDGE TEST

Test Case Name	Test Step	Action	Test Data	Expected Result	Test Result	Note
Add Edge	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
Add Edge	2.	Choose “addEdge” method	“v1”,”v2”	System should add vertex	System add vertex	

Test Case Name	Test Step	Action	Test Data	Expected Result	Test Result	Note
Add Edge Without Vertices	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
Add Edge Without Vertices	2.	Choose “addEdge” method	“v20”,”v53”	System should do nothing	System do nothing	

Test Case Name	Test Step	Action	Test Data	Expected Result	Test Result	Note
Remove Edge	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
Remove Edge	2.	Choose “removeEdge” method	“v1”,”v2”	System should remove edge	System remove edge	

Test Case Name	Test Step	Action	Test Data	Expected Result	Test Result	Note

Remove Not-Exist Edge	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
Remove Not-Exist Edge	2.	Choose “removeEdge” method	“v10”,”v53”	System should do nothing	System do nothing	

SEARCH TEST

Test Case Name	**Test Step**	**Action**	**Test Data**	**Expected Result**	**Test Result**	**Note**
DFS Search	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
DFS Search	2.	Choose “DFS” method	“v1”	System should do deep first search and print DFS traverse	System should do deep first search and print DFS traverse	

Test Case Name	**Test Step**	**Action**	**Test Data**	**Expected Result**	**Test Result**	**Note**
DFS Search with Not-Exist Vertex	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
DFS Search with Not-Exist Vertex	2.	Choose “DFS” method	“v53”	System should print nothing	System print nothing	

Test Case Name	**Test Step**	**Action**	**Test Data**	**Expected Result**	**Test Result**	**Note**

BFS Search	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
BFS Search	2.	Choose “BFS” method	“v1”	System should do breadth first search and print BFS traverse	System should do breadth first search and print BFS traverse	
Test Case Name	**Test Step**	**Action**	**Test Data**	**Expected Result**	**Test Result**	**Note**
BFS Search with Not-Exist Vertex	1	Create directed or non-directed graph	`LinkedGraph<String>`	System should create graph	System create graph	
BFS Search with Not-Exist Vertex	2.	Choose “BFS” method	“v53”	System should print nothing	System print nothing	

RUNNING COMMAND AND RESULT

```
        v1      v2      v3      v4      v5      v6
v1                      E:v1,v3
v2                      E:v2,v3         E:v2,v5
v3
v4              E:v4,v2
v5      E:v5,v1
v6              E:v6,v2

Add exist vertex result (v1) :false
Add non-exist vertex result (v7) :true
Remove non-exist vertex result (v8) :false
Remove exist vertex result (v3):true

After above operation, matrix graph
        v1      v2      v4      v5      v6      v7
v1
v2
v4              E:v4,v2
v5      E:v5,v1
v6              E:v6,v2
v7



-----------UNDIRECTED GRAPH ------------------------
Add edge without vertices 'v1,v50' and 'v40,v30'
Matrix Representation
        v1      v2      v3      v4      v5
v1                      E:v1,v3         E:v1,v5
v2              E:v2,v2 E:v2,v3 E:v2,v4 E:v2,v5
v3      E:v3,v1 E:v3,v2
v4              E:v4,v2
v5      E:v5,v1 E:v5,v2

Dfs Traverse: (Initial vertex = v4)
v5 v1 v3 v2 v4

Remove 'v4,v2' and 'v1,v3' edge from graph

Remove 'v10,v12 and 'v1,v5' edge from graph (not exist)

Bfs traverse after deletion :  (Initial vertex = v3)
v3 v2 v5
```

```
UNDIRECTED GRAPH
        0       1       2       3       4       5       6       7       8       9
0               E:0,1                                   E:0,6           E:0,8
1       E:1,0                   E:1,3                           E:1,7
2                               E:2,3                           E:2,7
3               E:3,1   E:3,2                   E:3,5                   E:3,8
4
5                               E:5,3
6       E:6,0                                                   E:6,7
7               E:7,1   E:7,2                           E:7,6                   E:7,9
8       E:8,0                   E:8,3
9                                                               E:9,7

Initial vertex = 0 - BFS
0 1 6 8 3 7 2 5 9
Initial vertex = 3 - BFS
3 1 2 5 8 0 7 6 9
Initial vertex = 0 - DFS
6 9 7 2 5 8 3 1 0
Initial vertex = 3 - DFS
2 9 7 6 8 0 1 5 3
Initial vertex = 15(non-exist) - DFS

Initial vertex = 53(non-exist) - BFS
```