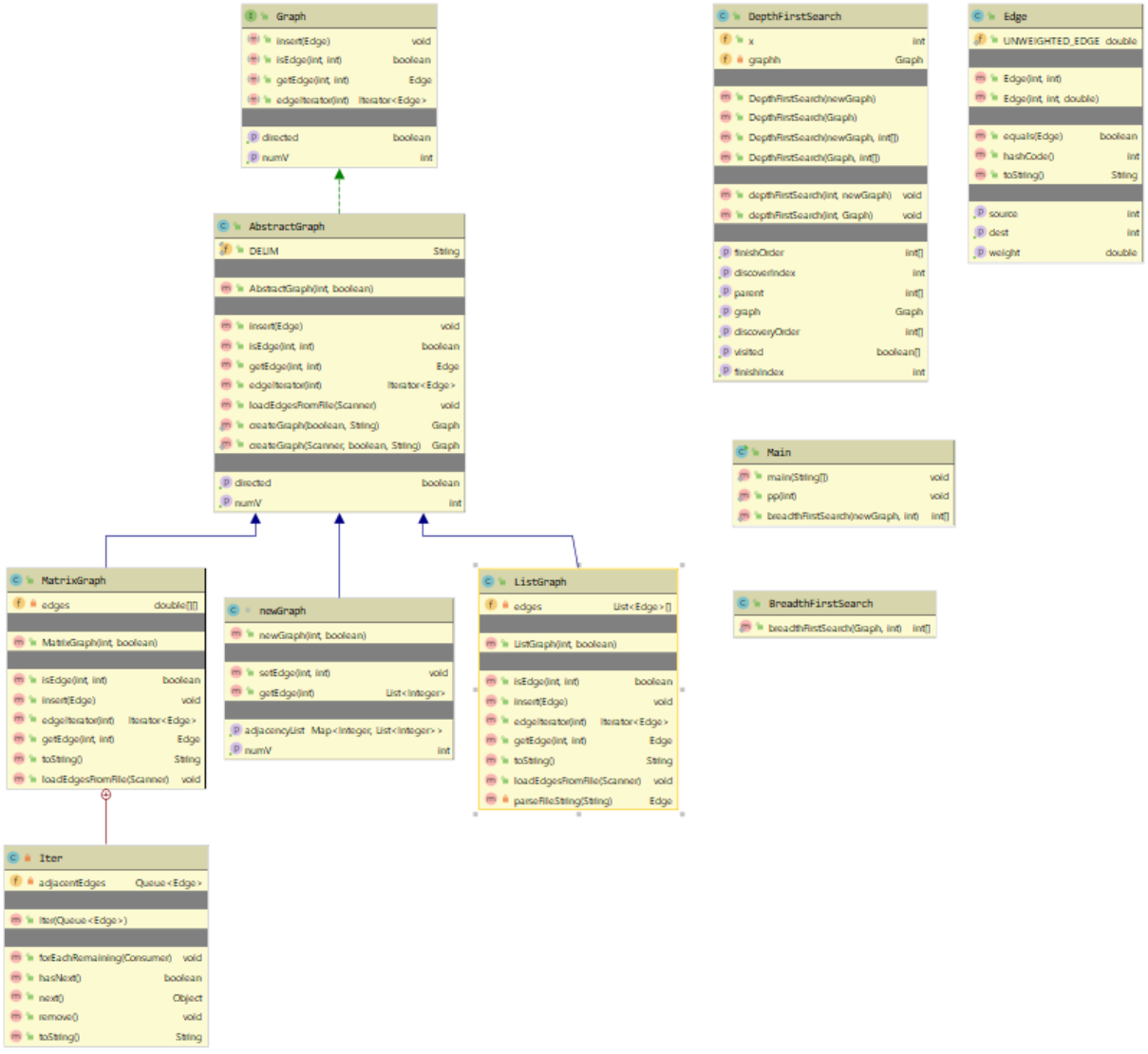
GIT Department of Computer Engineering
CSE 222/505 - Spring 2021
Homework 8 Report
part 2

Refik Orkun Arslan
151044063

Class Diagram



Problem solutions approach

I wrote a new graph and converted it into a class that generates a random graph. Then we calculated the working times using BFS and DFS. As the number of vertex grew, the working time increased. We organized the BFS and DFS functions according to the new graph.

Test cases

```
Graph g = null;
try {
    Scanner scan = new Scanner(new File( pathname: "./src/graph.txt"));
    g = AbstractGraph.createGraph(scan,  isDirected: true,  type: "List");
} catch (IOException ex) {
    ex.printStackTrace();
    System.exit( status: 1);
}


int[] parents = BreadthFirstSearch.breadthFirstSearch(g,  start: 0);
System.out.println("BFS :");
for(int i = 0; i < parents.length; i++){
    System.out.println(i + " " + parents[i]);
}
Graph gr = null;
int n = 0;
try {
    Scanner scan = new Scanner(new File( pathname: "./src/graph.txt"));
    gr = AbstractGraph.createGraph(scan,  isDirected: true,  type: "List");
    n = gr.getNumV();
} catch (IOException ex) {
    ex.printStackTrace();
    System.exit( status: 1);
}
DepthFirstSearch dfs = new DepthFirstSearch(gr);
int[] dOrder = dfs.getDiscoveryOrder();
int[] fOrder = dfs.getFinishOrder();
System.out.println("Discovery and finish order:");
for(int i = 0; i < n; i++){
    System.out.println(dOrder[i] + " " + fOrder[i]);
}


  System.out.println("----1000 vertex-----");
  pp( a: 1000);
  System.out.println("----2000 vertex----");
  pp( a: 2000);
  System.out.println("----5000 vertex----");
  pp( a: 5000);
  System.out.println("----10000 vertex----");
  pp( a: 10000);
```

Running command and results

```
BFS connected componenet : 32
BFS :
0 -1
1 0
2 1
3 0
4 0
5 0
6 1
DFS connected componenet : 32
Discovery and finish order:
0 6
1 5
2 4
3 3
4 2
5 1
6 0
----1000 vertex-----
12414100
----2000 vertex----
14774900
----5000 vertex----
32490000
----10000 vertex----
41494100
```