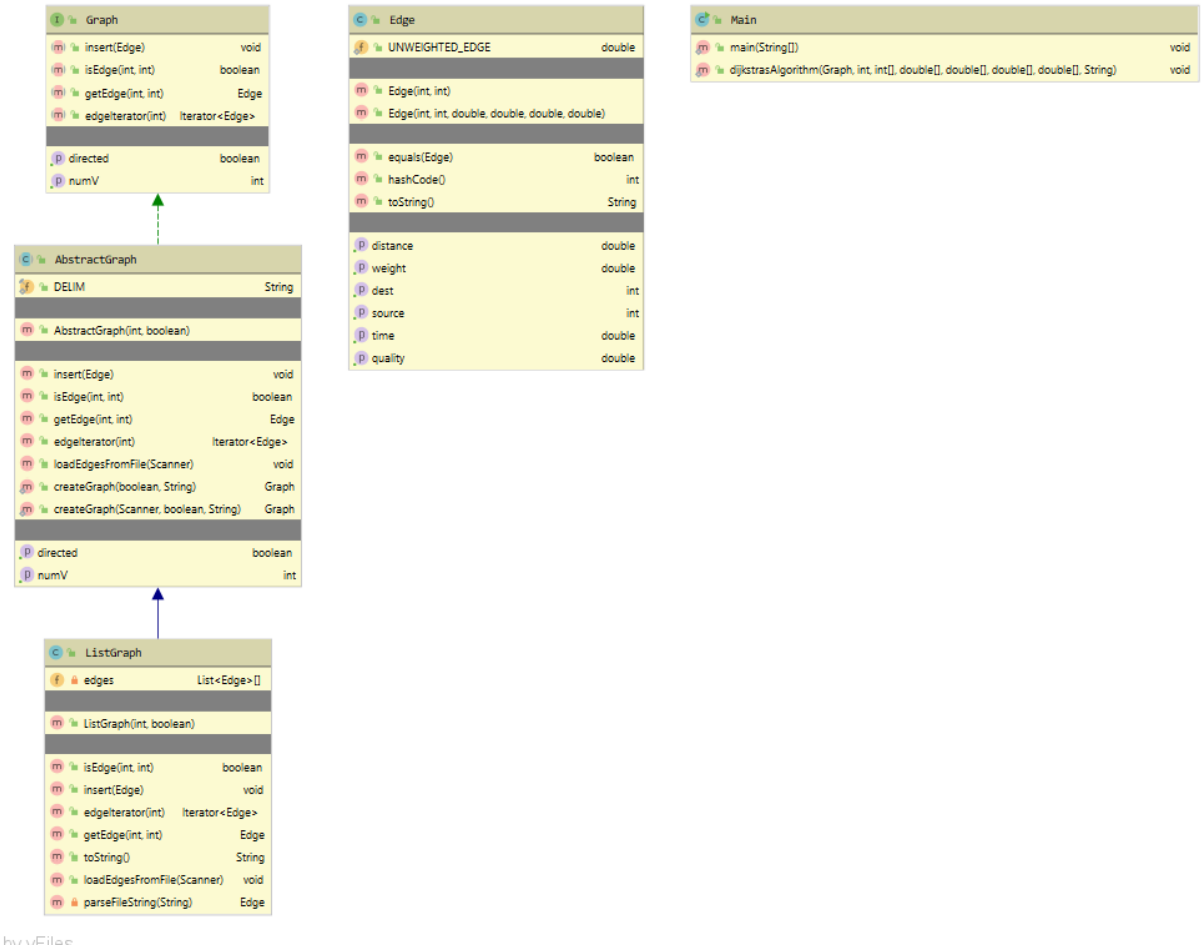
# GIT Department of Computer Engineering
# CSE 222/505 - Spring 2021

# Homework 8 Report
## *part 1*
***Refik Orkun Arslan***
***151044063***

# CLASS DİAGRAM




# Problem solutions approach

***I applied the dijkstra algorithm on both the list and the matrix. Then I added measures such as distance, time, quality along with the weight. Then we measured those lengths separately.addition ,multiplication and * operation.***

# *Test cases*

```
Graph g=null;
try {
    File myObj = new File( pathname: "./src/graph.txt");
    Scanner scan = new Scanner(myObj);
     g = AbstractGraph.createGraph(scan, isDirected: false,  type: "List");
    int numV = g.getNumV();
    int[] pred = new int[numV];
    double[] dist = new double[numV];
    double[] dis = new double[numV];
    double[] tim = new double[numV];
    double[] q = new double[numV];

    dijkstrasAlgorithm(g, start: 0, pred, dist,dis,tim,q, str: "addition");
    for(int i = 0; i < pred.length; i++){
        System.out.println(i + ":\t" + pred[i] + "\t" + dist[i]+ "\t" + dis[i]+ "\t" + tim[i]+ "\t" + q[i]);
    }
    System.out.print("\n");
//  System.out.println(g.toString());
} catch(Exception e){
    e.printStackTrace();
}
System.out.println("\n");
```

```
Graph gr=null;
try {
    File myObj = new File( pathname: "./src/graph.txt");
    Scanner scan = new Scanner(myObj);
    gr = AbstractGraph.createGraph(scan, isDirected: false,  type: "List");
    int numV = g.getNumV();
    int[] pred = new int[numV];
    double[] dist = new double[numV];
    double[] dis = new double[numV];
    double[] tim = new double[numV];
    double[] q = new double[numV];

    dijkstrasAlgorithm(g, start: 0, pred, dist,dis,tim,q, str: "multiplication");
    for(int i = 0; i < pred.length; i++){
        System.out.println(i + ":\t" + pred[i] + "\t" + dist[i]+ "\t" + dis[i]+ "\t" + tim[i]+ "\t" + q[i]);
    }
    System.out.print("\n");
    //System.out.println(g.toString());
} catch(Exception e){
    e.printStackTrace();
}
```

# Running command and results

```
addition
0:  0   0.0 0.0 0.0 0.0
1:  0   3.0 3.0 3.0 3.0
2:  5   33.0    33.0    2.0 2.0
3:  6   4.0 56.0    2.0 2.0
4:  5   12.0    66.0    2.0 2.0
5:  0   1.0 1.0 1.0 1.0
6:  5   35.0    35.0    2.0 2.0
```

```
addition operator
start   distance   weight   distance  time quality
vertex   vertex
```

```
multiplication
0:  0   0.0 0.0 0.0 0.0
1:  6   3.0 3.0 1.0 1.0
2:  5   32.0    32.0    1.0 1.0
3:  4   4.0 715.0   1.0 1.0
4:  5   12.0    65.0    1.0 1.0
5:  0   1.0 1.0 1.0 1.0
6:  5   34.0    34.0    1.0 1.0
```

```
multiplication opearator
start    distance  weight   distance time quality
vertex    vertex
```

```
*
0:  0   0.0 0.0 0.0 0.0
1:  6   1.0 1.0 1.0 1.0
2:  5   1.0 1.0 1.0 1.0
3:  5   1.0 1.0 1.0 1.0
4:  5   1.0 1.0 1.0 1.0
5:  0   1.0 1.0 1.0 1.0
6:  5   1.0 1.0 1.0 1.0
```