GTU Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework #8 Report

Hasan Mutlu
1801042673

SYSTEM REQUIREMENTS

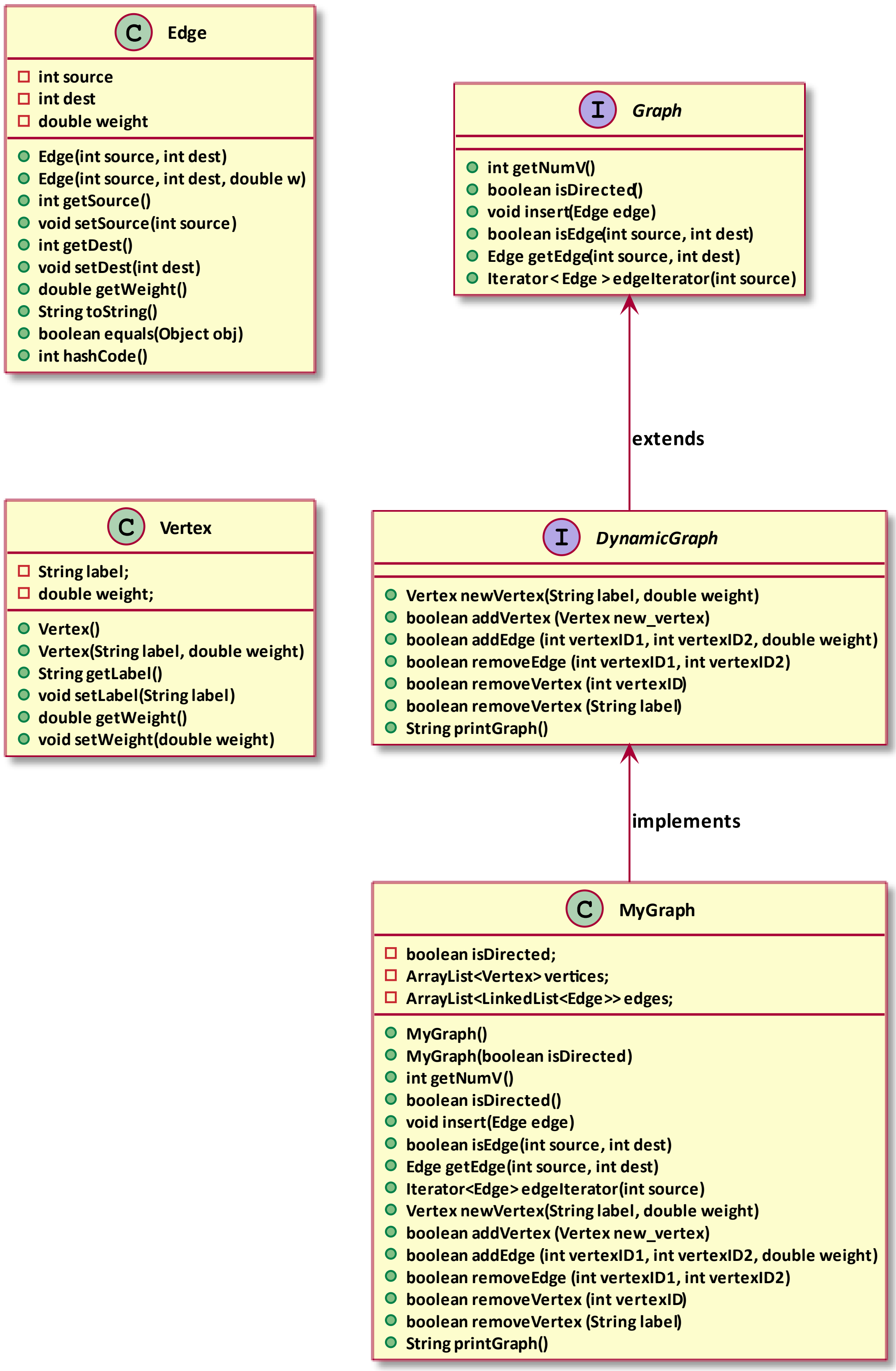
Functional Requirements:

- ➔ User should be able to see the result of hard-coded test cases.

Non-Functional Requirements:

- ➔ Implementation: The programs shall be implemented using VSCode, Ubuntu 18.04 WSL and Java 11.
- ➔ Compiling and Running: The programs should be compiled and run with following commands:
- ➔ -$ javac *.java PartX.java
- ➔ -$ java PartX
- ➔ Efficiency: The algorithms must run as efficiently as possible.
- ➔ Relaibility: The algorithms and the programs must run reliably, should handle every possible valid usage.

CLASS DIAGRAMS


C Edge
int source
int dest
double weight
Edge(int source, int dest)
Edge(int source, int dest, double w)
int getSource()
void setSource(int source)
int getDest()
void setDest(int dest)
double getWeight()
String toString()
boolean equals(Object obj)
int hashCode()
C Vertex
String label;
double weight;
Vertex()
Vertex(String label, double weight)
String getLabel()
void setLabel(String label)
double getWeight()
void setWeight(double weight)
I Graph
int getNumV()
boolean isDirected()
void insert(Edge edge)
boolean isEdge(int source, int dest)
Edge getEdge(int source, int dest)
Iterator< Edge > edgeIterator(int source)
extends
I DynamicGraph
Vertex newVertex(String label, double weight)
boolean addVertex (Vertex new_vertex)
boolean addEdge (int vertexID1, int vertexID2, double weight)
boolean removeEdge (int vertexID1, int vertexID2)
boolean removeVertex (int vertexID)
boolean removeVertex (String label)
String printGraph()
implements
C MyGraph
boolean isDirected;
ArrayList<Vertex> vertices;
ArrayList<LinkedList<Edge>> edges;
MyGraph()
MyGraph(boolean isDirected)
int getNumV()
boolean isDirected()
void insert(Edge edge)
boolean isEdge(int source, int dest)
Edge getEdge(int source, int dest)
Iterator<Edge> edgeIterator(int source)
Vertex newVertex(String label, double weight)
boolean addVertex (Vertex new_vertex)
boolean addEdge (int vertexID1, int vertexID2, double weight)
boolean removeEdge (int vertexID1, int vertexID2)
boolean removeVertex (int vertexID)
boolean removeVertex (String label)
String printGraph()

PROBLEM SOLUTION APPROACH

In part1, I firstly prepared DynamicGraph interface. After that, I have implemented the Edge class from the book with several additional methods and implemented Vertex class. Vertex class has label and weight fields. I didn't understand key-value data structure so I didn't add it. Indexing calculations are done in MyGraph class.

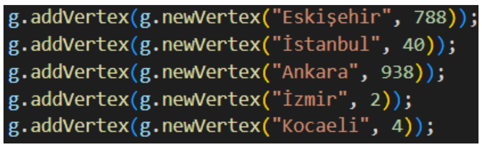
Then, I implemented MyGraph ADT. Vertices are kept in an ArrayList and edges are kept in an array of LinkedLists. There is also boolean isDirected field. When a new vertex is added, it is added to the end of vertices array and its edges are added to the end of edges arraylist. When a new edge is added, the edge is added to the LinkedList of source vertex with vertexID index. It is also added to the LinkedList of destination vertex, if it isn't directed.

While removing edge, the edge is simply removed from the corresponding vertices LinkedLists. When a vertex is removed, however, that vertex's entries are removed from both ArrayLists. Then any edge from/to that vertex is removed. Also, edges that have source/destination value greater than the removed vertex's index are decremented by one, as the ArrayList is shifted down after removal of the vertex.

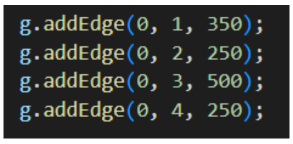
TEST CASES

Test Case #	Test Case Description	Test Data	Expected Result	Actual Result
Test Add Vertex	Add Vertices to MyGraph	5 cities in Turkey and their height.	Vertex added	As Expected
Test Add Edge	Add Edges to MyGraph	The distance between these cities.	Edges are added	As Expected
Test Remove Edge	Remove Edges from MyGraph	Remove edges between 3 cities.	Edges are removed	As Expected
Test Remove Vertex	Remove vertices from MyGraph	Remove 2 vertices	Vertices and any edge from/to that vertex are removed	As Expected

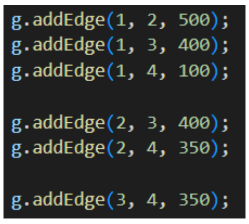
RUNNING AND RESULTS



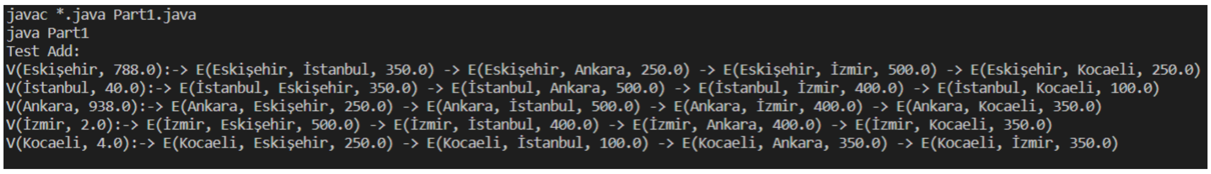
```
g.addVertex(g.newVertex("Eskişehir", 788));
g.addVertex(g.newVertex("İstanbul", 40));
g.addVertex(g.newVertex("Ankara", 938));
g.addVertex(g.newVertex("İzmir", 2));
g.addVertex(g.newVertex("Kocaeli", 4));
```



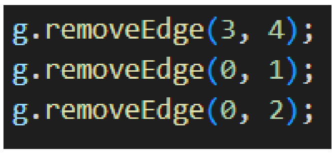
```
g.addEdge(0, 1, 350);
g.addEdge(0, 2, 250);
g.addEdge(0, 3, 500);
g.addEdge(0, 4, 250);
```



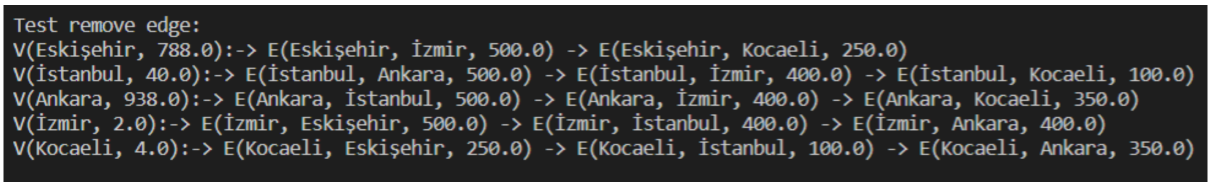
```
g.addEdge(1, 2, 500);
g.addEdge(1, 3, 400);
g.addEdge(1, 4, 100);

g.addEdge(2, 3, 400);
g.addEdge(2, 4, 350);

g.addEdge(3, 4, 350);
```



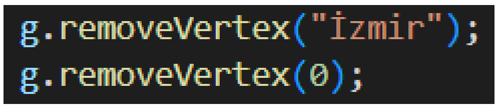
```
javac *.java Part1.java
java Part1
Test Add:
V(Eskişehir, 788.0):-> E(Eskişehir, İstanbul, 350.0) -> E(Eskişehir, Ankara, 250.0) -> E(Eskişehir, İzmir, 500.0) -> E(Eskişehir, Kocaeli, 250.0)
V(İstanbul, 40.0):-> E(İstanbul, Eskişehir, 350.0) -> E(İstanbul, Ankara, 500.0) -> E(İstanbul, İzmir, 400.0) -> E(İstanbul, Kocaeli, 100.0)
V(Ankara, 938.0):-> E(Ankara, Eskişehir, 250.0) -> E(Ankara, İstanbul, 500.0) -> E(Ankara, İzmir, 400.0) -> E(Ankara, Kocaeli, 350.0)
V(İzmir, 2.0):-> E(İzmir, Eskişehir, 500.0) -> E(İzmir, İstanbul, 400.0) -> E(İzmir, Ankara, 400.0) -> E(İzmir, Kocaeli, 350.0)
V(Kocaeli, 4.0):-> E(Kocaeli, Eskişehir, 250.0) -> E(Kocaeli, İstanbul, 100.0) -> E(Kocaeli, Ankara, 350.0) -> E(Kocaeli, İzmir, 350.0)
```



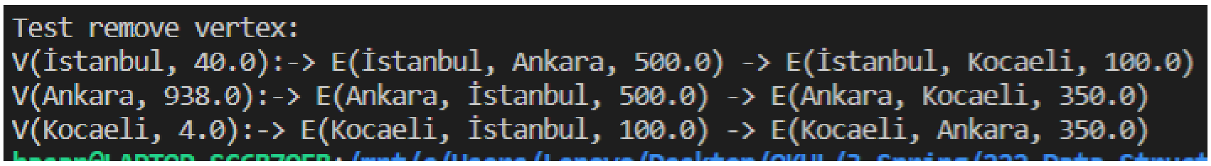
```
g.removeEdge(3, 4);
g.removeEdge(0, 1);
g.removeEdge(0, 2);
```



```
Test remove edge:
V(Eskişehir, 788.0):-> E(Eskişehir, İzmir, 500.0) -> E(Eskişehir, Kocaeli, 250.0)
V(İstanbul, 40.0):-> E(İstanbul, Ankara, 500.0) -> E(İstanbul, İzmir, 400.0) -> E(İstanbul, Kocaeli, 100.0)
V(Ankara, 938.0):-> E(Ankara, İstanbul, 500.0) -> E(Ankara, İzmir, 400.0) -> E(Ankara, Kocaeli, 350.0)
V(İzmir, 2.0):-> E(İzmir, Eskişehir, 500.0) -> E(İzmir, İstanbul, 400.0) -> E(İzmir, Ankara, 400.0)
V(Kocaeli, 4.0):-> E(Kocaeli, Eskişehir, 250.0) -> E(Kocaeli, İstanbul, 100.0) -> E(Kocaeli, Ankara, 350.0)
```



```
g.removeVertex("İzmir");
g.removeVertex(0);
```



```
Test remove vertex:
V(İstanbul, 40.0):-> E(İstanbul, Ankara, 500.0) -> E(İstanbul, Kocaeli, 100.0)
V(Ankara, 938.0):-> E(Ankara, İstanbul, 500.0) -> E(Ankara, Kocaeli, 350.0)
V(Kocaeli, 4.0):-> E(Kocaeli, İstanbul, 100.0) -> E(Kocaeli, Ankara, 350.0)
```