Lab 12: Graph

Objective(s)

- Weighted Graph definition and operations.
- implement a weighted graph using array

Tool(s)/Software

Java programming language with NetBeans IDE.

Description

- Operations
  - Adjacency Matrix
  - Count edges
  - Count vertex
  - Find the path

Defining the structure of graph in Java:

a. **Weighted Graph class**

```
public class WeightedGraph {
    public int[][] edge;
    public String[] vertice;
    public int eSize=0;
    public int vSize=0;
    public WeightedGraph(int size){
        edge = new int [size][size];
        vertice = new String[size];
        eSize = vSize = size;
    }
```

b. Weighted Graph methods

```
public int getEdgeSize(){
    return eSize;
}
public int getVertSize(){
    return vSize;
}
public boolean isEdge(int Row, int Col){
    return edge[Row][Col]>0;
}

public void setVertice(int Row, String str){
    vertice[Row]=str;
}
public String getVertice(int Row){
    return vertice[Row];
}
public void setEdge(int Row, int Col, int Weight){
    edge[Row][Col] = Weight;
}
public int getEdge(int Row, int Col){
    return edge[Row][Col];
}

public void removeEdge(int Row, int Col){
    edge[Row][Col]=0;
}
```

c. Count the connected vertices

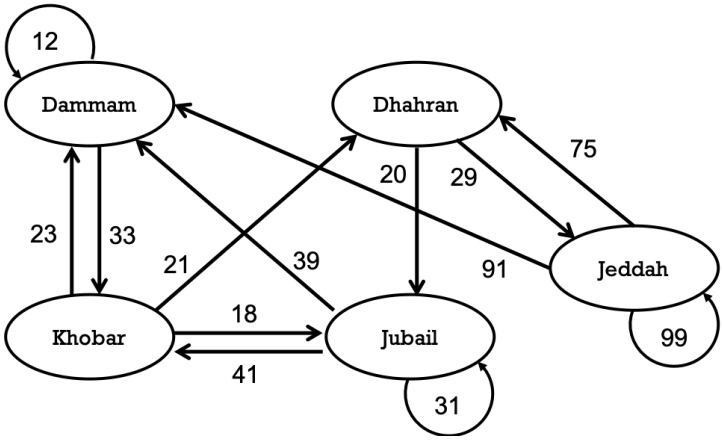
```
    public void countVertice(){
        int count;
        for (int Row=0; Row<vSize; Row++){
            count = 0;
            for (int Col=0; Col<vSize; Col++)
                if (isEdge(Row, Col)) count++;
        System.out.println(vertice[Row]+": "+ count);
        }
    }
```

d. Print the vertices

```
public void printVertice(){
    for (int Row=0; Row<vSize; Row++){
        System.out.print(vertice[Row]+"->");
        for(int Col=0; Col<vSize; Col++){
            if(isEdge(Row, Col))
                System.out.print(vertice[Col]+": "+edge[Row][Col]+", ");
        }
        System.out.println();
    }
}
```

e. Main Method

```
public static void main(String[] args) {
    WeightedGraph wg = new WeightedGraph(5);
    int r = 0;
    wg.setVertice(r++, "Dammam");
    wg.setVertice(r++, "Khobar");
    wg.setVertice(r++, "Zahran");
    wg.setVertice(r++, "Jubail");
    wg.setVertice(r++, "Jaddah");

    wg.setEdge(0, 0 , 12);
    wg.setEdge(0, 1 , 33);
    wg.setEdge(1, 0 , 23);
    wg.setEdge(1, 2 , 21);
    wg.setEdge(1, 3 , 18);
    wg.setEdge(2, 3 , 20);
    wg.setEdge(2, 4 , 29);
    wg.setEdge(3, 0 , 39);
    wg.setEdge(3, 1 , 41);
    wg.setEdge(3, 3 , 31);
    wg.setEdge(4, 0 , 91);
    wg.setEdge(4, 2 , 75);
    wg.setEdge(4, 4 , 99);

    System.out.println(" -- Print the vertices of the Graph");
    wg.printVertice();

    System.out.println(" \n\n-- Print number of connected vertices of the Graph");
    wg.countVertice();

}
```



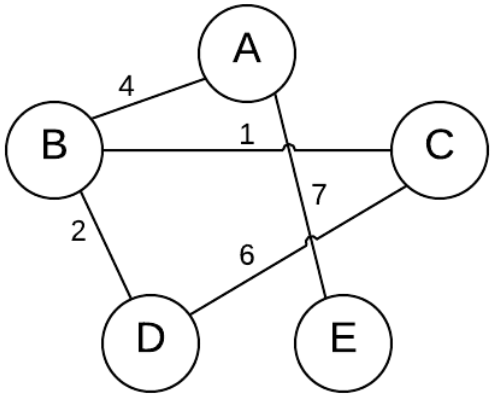
f. Output

```
 -- Print the vertices of the Graph
Dammam->
Dammam: 12,
Khobar: 33,

Khobar->
Dammam: 23,
Dhahran: 21,
Jubail: 18,

Dhahran->
Jubail: 20,
Jeddah: 29,

Jubail->
Dammam: 39,
Khobar: 41,
Jubail: 31,

Jeddah->
Dammam: 91,
Dhahran: 75,
Jeddah: 99,

-- Print the number of connected vertices of the Graph
Dammam: 2
Khobar: 3
Dhahran: 2
Jubail: 3
Jeddah: 3
```

Tasks/Assignments(s)

1- Write the code for the following methods:

- ☒ Create the graph in the below figure.




```
run:
 -- Print the vertices of the Graph
A->
B: 4,
E: 7,

B->
A: 4,
C: 1,
D: 2,

C->
B: 1,
D: 6,

D->
B: 2,
C: 6,

E->
A: 7,

-- Print the number of connected vertices of the Graph
A: 2
B: 3
C: 2
D: 2
E: 1
```

Deliverables(s)

You are required to implement and deliver a Java program(s) as described in the previous section.