**PROGRAM:**

package com.mycompany.project1;

import java.util.\*;

class Graph {

public int n; // number of vertices

public int[][] g = new int[10][10]; // adjacency matrix

Graph(int x) // constructor

{

this.n = x;

int i, j;

// initializing each element of the adjacency matrix to zero

for (i = 0; i < n; ++i)

{

for (j = 0; j < n; ++j)

{

g[i][j] = 0;

}

}

}

public void displayAdjacencyMatrix()

{

System.out.print("\n\n Adjacency Matrix:");

// displaying the 2D array

for (int i = 0; i < n; ++i)

{

System.out.println();

for (int j = 0; j < n; ++j)

{

System.out.print(" " + g[i][j]);

}

}

}

public void addEdge(int x, int y)

{

// checks if the vertex exists in the graph

if ((x >= n) || (y > n))

{

System.out.println("Vertex does not exists!");

}

// checks if the vertex is connecting to itself

if (x == y)

{

System.out.println("Same Vertex!");

}

else

{

// connecting the vertices

g[y][x] = 1;

g[x][y] = 1;

}

}

public void removeedge(int x,int y)

{

if((x>=n) || (y>=n))

{

System.out.println("Edge not present!");

}

else if(x==y)

{

System.out.println("Same vertex!");

}

else

{

g[x][y] = 0;

g[y][x] = 0;

}

}

public void addVertex()

{

// increasing the number of vertices

n++;

int i;

// initializing the new elements to 0

for (i = 0; i < n; ++i)

{

g[i][n - 1] = 0;

g[n - 1][i] = 0;

}

int j=n-1;

System.out.println("v"+j+" Vertex has been created!");

}

public void removeVertex(int x)

{

// checking if the vertex is present

if (x > n)

{

System.out.println("Vertex not present!");

return;

}

else

{

int i;

// removing the vertex

while (x < n)

{

// shifting the rows to left side

for (i = 0; i < n; ++i)

{

g[i][x] = g[i][x + 1];

}

// shifting the columns upwards

for (i = 0; i < n; ++i)

{

g[x][i] = g[x + 1][i];

}

x++;

}

// decreasing the number of vertices

n--;

}

}

public void dfs(int start,boolean[] visited)

{

System.out.print(start +" ");

visited[start] = true;

for(int i=0;i< g[start].length; i++)

{

if(g[start][i]==1 && (!visited[i]))

{

dfs(i,visited);

}

}

}

public void BFS(int start)

{

boolean[] visited = new boolean[n];

Arrays.fill(visited, false);

List<Integer> q = new ArrayList<>();

q.add(start);

// Set source as visited

visited[start] = true;

int vis;

while (!q.isEmpty())

{

vis = q.get(0);

// Print the current node

System.out.print(vis + " ");

q.remove(q.get(0));

// For every adjacent vertex to the current vertex

for(int i = 0; i < n; i++)

{

if (g[vis][i] == 1 && (!visited[i]))

{

// Push the adjacent node to the queue

q.add(i);

// Set

visited[i] = true;

}

}

}

}

}

public class Project1 {

public static void main(String[] args)

{

//object of class Graph

Graph obj = new Graph(1);

while(true)

{

System.out.println("1.Add vertex\n2.Remove vertex\n3.Add edge \n4.Remove edge \n5.BFS \n6.DFS \n7.display graph \n8.Exit \nenter your choice\n");

Scanner s= new Scanner(System.in);

int i=s.nextInt();

switch(i)

{

case 1:

obj.addVertex();

System.out.println();

break;

case 2:

System.out.println("enter the vertex to be deleted");

int v=s.nextInt();

obj.removeVertex(v);

System.out.println();

break;

case 3:

System.out.println("enter the vertices between which you want edge");

int x,y;

System.out.println("enter the 1st vertex");

x=s.nextInt();

System.out.println("enter the 2nd vertex");

y=s.nextInt();

obj.addEdge(x,y);

System.out.println();

break;

case 4:

System.out.println("enter the vertices between which you want to delete the edge");

int p,q;

System.out.println("enter the 1st vertex");

p=s.nextInt();

System.out.println("enter the 2nd vertex");

q=s.nextInt();

obj.removeedge(p,q);

System.out.println();

break;

case 5:

System.out.println("enter the source vertex");

int u;

u=s.nextInt();

obj.BFS(u);

System.out.println();

break;

case 6:

boolean[] visited=new boolean[obj.n];

System.out.println("enter the source vertex");

int k;

k=s.nextInt();

obj.dfs(k,visited);

System.out.println();

break;

case 7:

obj.displayAdjacencyMatrix();

System.out.println();

break;

case 8:

System.exit(0);

default:

System.out.println("Wrong choice");

}

}

}

}

**OUTPUT:**

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

1

v1 Vertex has been created!

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

1

v2 Vertex has been created!

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

1

v3 Vertex has been created!

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

1

v4 Vertex has been created!

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

3

enter the vertices between which you want edge

enter the 1st vertex

0

enter the 2nd vertex

3

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

3

enter the vertices between which you want edge

enter the 1st vertex

1

enter the 2nd vertex

2

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

3

enter the vertices between which you want edge

enter the 1st vertex

0

enter the 2nd vertex

1

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

3

enter the vertices between which you want edge

enter the 1st vertex

3

enter the 2nd vertex

2

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

3

enter the vertices between which you want edge

enter the 1st vertex

2

enter the 2nd vertex

4

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

7

Adjacency Matrix:

0 1 0 1 0

1 0 1 0 0

0 1 0 1 1

1 0 1 0 0

0 0 1 0 0

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

5

enter the source vertex

0

0 1 3 2 4

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

6

enter the source vertex

0

0 1 2 3 4

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

4

enter the vertices between which you want to delete the edge

enter the 1st vertex

0

enter the 2nd vertex

3

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

7

Adjacency Matrix:

0 1 0 0 0

1 0 1 0 0

0 1 0 1 1

0 0 1 0 0

0 0 1 0 0

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

2

enter the vertex to be deleted

4

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

7

Adjacency Matrix:

0 1 0 0

1 0 1 0

0 1 0 1

0 0 1 0

1.Add vertex

2.Remove vertex

3.Add edge

4.Remove edge

5.BFS

6.DFS

7.display graph

8.Exit

enter your choice

8

------------------------------------------------------------------------

BUILD SUCCESS