```java
import java.util.Scanner;

public class BellmanFord {
    private int distances[];
    private int numVertices;
    public static final int MAX_VALUE = 999;

    public BellmanFord(int numVertices) {
        this.numVertices = numVertices;
        distances = new int[numVertices + 1];
    }

    public void BellmanFordEvaluation(int source, int destination, int adjacencymatrix[][]) {
        for (int node = 1; node <= numVertices; node++) {
            distances[node] = MAX_VALUE;
        }
        distances[source] = 0;
        for (int node = 1; node <= numVertices - 1; node++) {
            for (int snode = 1; snode <= numVertices; snode++) {
                for (int dnode = 1; dnode <= numVertices; dnode++) {
                    if (adjacencymatrix[snode][dnode] != MAX_VALUE) {
                        if (distances[dnode] > distances[snode] + adjacencymatrix[snode][dnode])
                            distances[dnode] = distances[snode] + adjacencymatrix[snode][dnode];
                    }
                }
            }
        }

        for (int snode = 1; snode <= numVertices; snode++) {
            for (int dnode = 1; dnode <= numVertices; dnode++) {
                if (adjacencymatrix[snode][dnode] != MAX_VALUE) {
                    if (distances[dnode] > distances[snode] + adjacencymatrix[snode][dnode])
                        System.out.println("The Graph contains negative egde cycle");
                }
            }
```

```
36          }
37          for (int vertex = 1; vertex <= numVertices; vertex++) {
38              if (vertex == destination)
39                  System.out.println("distance of source  " + source +
  " to " + vertex + " is " + distances[vertex]);
40          }
41      }
42
43      public static void main(String... arg) {
44          int numVertices = 0;
45          int source, destination;
46          Scanner scanner = new Scanner(System.in);
47          System.out.println("Enter the number of vertices");
48          numVertices = scanner.nextInt();
49          int adjacencymatrix[][] = new int[numVertices + 1]
  [numVertices + 1];
50          System.out.println("Enter the adjacency matrix");
51
52          for (int snode = 1; snode <= numVertices; snode++) {
53              for (int dnode = 1; dnode <= numVertices; dnode++) {
54                  adjacencymatrix[snode][dnode] = scanner.nextInt();
55                  if (snode == dnode) {
56                      adjacencymatrix[snode][dnode] = 0;
57                      continue;
58                  }
59                  if (adjacencymatrix[snode][dnode] == 0) {
60                      adjacencymatrix[snode][dnode] = MAX_VALUE;
61                  }
62              }
63          }
64          System.out.println("Enter the source vertex");
65          source = scanner.nextInt();
66          System.out.println("Enter the destination vertex: ");
67          destination = scanner.nextInt();
68          BellmanFord bellmanford = new BellmanFord(numVertices);
69          bellmanford.BellmanFordEvaluation(source, destination,
  adjacencymatrix);
70          scanner.close();
71      }
72  }
```