Program8 bellmanford

```
import java.util.Scanner;
public class BellmanFord
{
 private int dist[];
private int vertices;
public static final int MAX_VALUE = 999;
public BellmanFord(int vertices)
{
this.vertices = vertices;
dist = new int[vertices + 1];
 }
 public void BellmanFordEval(int source,
int adjmatrix[][])
 {
 for (int node = 1; node <= vertices;
node++)
{
dist[node] = MAX_VALUE;
}
 dist[source] = 0;
for (int node = 1; node <= vertices - 1;
node++)
{
for (int srcnode = 1; srcnode <= vertices;
srcnode++)
 {
 for (int destnode = 1; destnode <=
vertices; destnode++)
{
 if (adjmatrix[srcnode][destnode] !=
MAX_VALUE)
{
 if (dist[destnode] > dist[srcnode] +
adjmatrix[srcnode][destnode])
 dist[destnode] = dist[srcnode] +
adjmatrix[srcnode][destnode];
} } } }
for (int srcnode = 1; srcnode <= vertices;
srcnode++)
 {
 for (int destnode = 1; destnode <=
vertices; destnode++)
{
 if (adjmatrix[srcnode][destnode] !=
MAX_VALUE)
{
 if (dist[destnode] > dist[srcnode] +
adjmatrix[srcnode][destnode])
System.out.println("The Graph contains
negative egde cycle");
} } }
for (int vertex = 1; vertex <= vertices;
vertex++)
{
 System.out.println("distance of source " +
source + " to " + vertex + " is " +
dist[vertex]);
 } }
 public static void main(String... arg)
{
 int nmbrofvertices = 0;
int source; Scanner scanner = new
Scanner(System.in);
System.out.println("Enter the number of
vertices");
 nmbrofvertices = scanner.nextInt();
int adjmatrix[][] = new int[nmbrofvertices
+ 1][nmbrofvertices + 1];
System.out.println("Enter the adjacency
matrix");
for (int srcnode = 1; srcnode <=
nmbrofvertices; srcnode++)
{
 for (int destnode = 1; destnode <=
nmbrofvertices; destnode++)
 {
 adjmatrix[srcnode][destnode] =
scanner.nextInt();
if (srcnode == destnode)
{
adjmatrix[srcnode][destnode] = 0;
 continue;
 }
 if (adjmatrix[srcnode][destnode] == 0)
 {
 adjmatrix[srcnode][destnode] =
MAX_VALUE;
} } }
System.out.println("Enter the source
vertex");
source = scanner.nextInt();
BellmanFord bellmanford = new
BellmanFord(nmbrofvertices);
bellmanford.BellmanFordEval(source,
adjmatrix);
scanner.close();
 } }
```