Shortest path between vertices using bellman-ford algorithm.

```
import java.util.Scanner;
 public class BellmanFord
{
    private int distances[ ];
    private int numberofvertices;
    public static final int MAX_VALUE = 999;

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

    public void BellmanFordEvaluation(int source, int destination, int adjacencymatrix[ ][ ])
    {
        for (int node = 1; node <= numberofvertices; node++)
        {
            distances[node] = MAX_VALUE;
        }

       distances[source] = 0;

       for (int node = 1; node <= numberofvertices - 1; node++)
        {
            for (int sourcenode = 1; sourcenode <= numberofvertices; sourcenode++)
            {
                for (int destinationnode = 1; destinationnode <= numberofvertices;
                     destinationnode++)
             {
               if (adjacencymatrix[sourcenode][destinationnode] != MAX_VALUE)
```

Shortest path between vertices using bellman-ford algorithm.

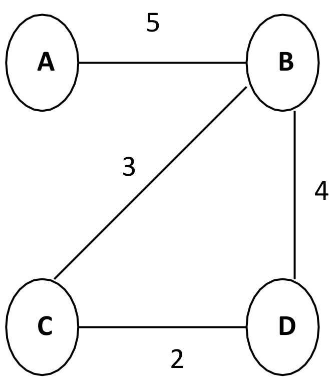
```
{
               if (distances[destinationnode] > distances[sourcenode] +
               adjacencymatrix[sourcenode][destinationnode])
               distances[destinationnode] =
               distances[sourcenode]+adjacencymatrix[sourcenode][destinationnode];

                }
             }
       }
  }

     for (int vertex = 1; vertex <= numberofvertices; vertex++)
     {
         if (vertex == destination)
           System.out.println("Distance of source  " + source + " to "+ vertex + " is " +
                                          distances[vertex]);
     }
  }


  public static void main(String[ ] args)
  {
     int numberofvertices = 0;
     int source, destination;
     Scanner scanner = new Scanner(System.in);
     System.out.println("Enter the number of vertices: ");
     numberofvertices = scanner.nextInt();
     int adjacencymatrix[ ][ ] = new int[numberofvertices + 1][numberofvertices + 1];
     System.out.println("Enter the adjacency matrix");
```

Shortest path between vertices using bellman-ford algorithm.

```
        for (int sourcenode = 1; sourcenode <= numberofvertices; sourcenode++)
        {
            for (int destinationnode = 1; destinationnode <= numberofvertices;
                 destinationnode++)
            {
                adjacencymatrix[sourcenode][destinationnode] = scanner.nextInt();
                if (sourcenode == destinationnode)
                {
                    adjacencymatrix[sourcenode][destinationnode] = 0;
                    continue;
                }
                if (adjacencymatrix[sourcenode][destinationnode] == 0)
                {
                    adjacencymatrix[sourcenode][destinationnode] = MAX_VALUE;
                }
            }
        }
        System.out.println("Enter the source vertex");
        source = scanner.nextInt();
        System.out.println("Enter the destination vertex: ");
        destination = scanner.nextInt();
        BellmanFord bellmanford = new BellmanFord(numberofvertices);
        bellmanford.BellmanFordEvaluation(source, destination, adjacencymatrix);
        scanner.close();
    }
}
```

Shortest path between vertices using bellman-ford algorithm.

Output:



```
Enter the number of vertices:
4
Enter the adjacency matrix
0 5 0 0
5 0 3 4
0 3 0 2
0 4 2 0

Enter the source vertex: 1
Enter the Destination vertex: 4
Distance of source  1 to 4 is 9
```