```
/* Program 4
From a given source vertex in a weighted connected graph, find shortest paths to other vertices
using Dijkstra's algorithm.
*/
import java.util.*;
public class DijkstrasClass {

        final static int MAX = 20;
        final static int infinity = 9999;
        static int n;                           // No. of vertices of G
        static int a[][];                       // Cost matrix
        static Scanner scan = new Scanner(System.in);

        public static void main(String[] args) {
                int s = 0;                      // starting vertex
                System.out.println("*****DIJKSTRA'S ALGORTIHM*****");
                ReadMatrix();
                System.out.println("Enter starting vertex: ");
                s = scan.nextInt();
                Dijkstras(s);                   // find shortest path
        }
        static void ReadMatrix() {
                a = new int[MAX][MAX];
                System.out.println("Enter the number of vertices:");
                n = scan.nextInt();
                System.out.println("Enter the cost adjacency matrix:");
                        for (int i = 1; i <= n; i++)
                                for (int j = 1; j <= n; j++)
                                        a[i][j] = scan.nextInt();
         }
         static void Dijkstras(int s) {
                int S[] = new int[MAX];
                int d[] = new int[MAX];
                int u, v;
                int i;
                for (i = 1; i <= n; i++) {
                        S[i] = 0;
                        d[i] = a[s][i];
                }
                S[s] = 1;
                d[s] = 1;
                i = 2;
                while (i <= n) {
                        u = Extract_Min(S, d);
                        S[u] = 1;
                        i++;
                        for (v = 1; v <= n; v++) {
                                if (((d[u] + a[u][v] < d[v]) && (S[v] == 0)))
                                        d[v] = d[u] + a[u][v];
                        }
                }
                System.out.println("The shortest distance from source vertex "+s+" to all other vertices
are:");
                for (i = 1; i <= n; i++)
                        if (i != s)
                                System.out.println(i + ":" + d[i]);
        }

        static int Extract_Min(int S[], int d[]) {
                int i, j = 1, min;
                min = infinity;
                for (i = 1; i <= n; i++) {
                        if ((d[i] < min) && (S[i] == 0)) {
                                min = d[i];
                                j = i;
                        }
                }
                return (j);
        }
}
```

OUTPUT:

```
*****DIJKSTRA'S ALGORTIHM*****
Enter the number of vertices:
5
Enter the cost adjacency matrix:
0 4 8 999 999
4 0 2 5 999
8 2 0 5 9
999 5 5 0 4
999 999 9 4 0
Enter starting vertex:
1
The shortest distance from source vertex 1 to all other vertices are:
2:4
3:6
4:9
5:13
```