```java
import java.util.*;

public class Prim {
    private boolean unsettled[];
    private boolean settled[];
    private int numVertices;
    private int matrix[][];
    private int key[];
    public static final int INFINITE = 999;
    private int parent[];

    public Prim(int numVertices) {
        this.numVertices = numVertices;
        unsettled = new boolean[numVertices + 1];
        settled = new boolean[numVertices + 1];
        matrix = new int[numVertices + 1][numVertices + 1];
        key = new int[numVertices + 1];
        parent = new int[numVertices + 1];
    }

    public int getUnsettledCount(boolean unsettled[]) {
        int count = 0;
        for (int index = 0; index < unsettled.length; index++) {
            if (unsettled[index]) {
                count++;
            }
        }
        return count;
    }

    public void primsAlgorithm(int adjacencyMatrix[][]) {
        int evaluationVertex;
        for (int source = 1; source <= numVertices; source++) {
            for (int destination = 1; destination <= numVertices;
destination++) {
                this.matrix[source][destination] =
adjacencyMatrix[source][destination];
            }
        }

        for (int index = 1; index <= numVertices; index++) {
```

```
40              key[index] = INFINITE;
41          }
42          key[1] = 0;
43          unsettled[1] = true;
44          parent[1] = 1;
45
46          while (getUnsettledCount(unsettled) != 0) {
47              evaluationVertex =
  getMimumKeyVertexFromUnsettled(unsettled);
48              unsettled[evaluationVertex] = false;
49              settled[evaluationVertex] = true;
50              evaluateNeighbours(evaluationVertex);
51          }
52      }
53
54      private int getMimumKeyVertexFromUnsettled(boolean[]
  unsettled2) {
55          int min = Integer.MAX_VALUE;
56          int node = 0;
57          for (int vertex = 1; vertex <= numVertices; vertex++) {
58              if (unsettled[vertex] == true && key[vertex] < min) {
59                  node = vertex;
60                  min = key[vertex];
61              }
62          }
63          return node;
64      }
65
66      public void evaluateNeighbours(int eVertex) {
67          for (int d = 1; d <= numVertices; d++) {
68              if (settled[d] == false) {
69                  if (matrix[eVertex][d] != INFINITE) {
70                      if (matrix[eVertex][d] < key[d]) {
71                          key[d] = matrix[eVertex][d];
72                          parent[d] = eVertex;
73                      }
74                      unsettled[d] = true;
75                  }
76              }
77          }
78      }
```

```
79
80      public void printMST() {
81          System.out.println("SOURCE  : DESTINATION = WEIGHT");
82          for (int v = 2; v <= numVertices; v++) {
83              System.out.println(parent[v] + "\t:\t" + v +"\t=\t"+
    matrix[parent[v]][v]);
84          }
85      }
86
87      public static void main(String... arg) {
88          int adjacency_matrix[][];
89          int numVer;
90          Scanner scan = new Scanner(System.in);
91
92          try {
93              System.out.println("Number of vertices");
94              numVer = scan.nextInt();
95              adjacency_matrix = new int[numVer + 1][numVer + 1];
96              System.out.println("Weighted Matrix");
97              for (int i = 1; i <= numVer; i++) {
98                  for (int j = 1; j <= numVer; j++) {
99                      adjacency_matrix[i][j] = scan.nextInt();
100                     if (i == j) {
101                         adjacency_matrix[i][j] = 0;
102                         continue;
103                     }
104                     if (adjacency_matrix[i][j] == 0) {
105                         adjacency_matrix[i][j] = INFINITE;
106                     }
107                 }
108             }
109             Prim prims = new Prim(numVer);
110             prims.primsAlgorithm(adjacency_matrix);
111             prims.printMST();
112         }
113         catch (InputMismatchException inputMismatch) {
114             System.out.println("Wrong Input Format");
115         }
116         scan.close();
117     }
118 }
```