```java
package primsAlgorithem;

class MST {
    // Number of vertices in the graph
    private static final int V = 5;

    // A utility function to find the vertex with minimum key
    // value, from the set of vertices not yet included in MST
    int minKey(int key[], Boolean mstSet[])
    {
        // Initialize min value
        int min = Integer.MAX_VALUE, min_index = -1;

        for (int v = 0; v < V; v++)
            if (mstSet[v] == false && key[v] < min ) {
                min = key[v];
                min_index = v;
            }

        return min_index;
    }

    // A utility function to print the constructed MST stored in
    // parent[]
    void printMST(int parent[], int graph[][])
    {
        System.out.println("Edge \tWeight");
        for (int i = 1; i < V; i++)
            System.out.println(parent[i] + " - " + i + "\t" + graph[i][parent[i]]);
    }

    // Function to construct and print MST for a graph represented
    // using adjacency matrix representation
    void primMST(int graph[][])
    {
        // Array to store constructed MST
        int parent[] = new int[V];

        // Key values used to pick minimum weight
```

```
39 edge in cut
40         int key[] = new int[V];
41 
42         // To represent set of vertices included in
   MST
43         Boolean mstSet[] = new Boolean[V];
44 
45         // Initialize all keys as INFINITE
46         for (int i = 0; i < V; i++) {
47             key[i] = Integer.MAX_VALUE;
48             mstSet[i] = false;
49         }
50 
51         // Always include first 1st vertex in MST.
52         key[0] = 0; // Make key 0 so that this vertex
    is
53         // picked as first vertex
54         parent[0] = -1; // First node is always root
   of MST
55 
56         // The MST will have V vertices
57         for (int count = 0; count < V - 1; count
   ++) {
58             // Pick thd minimum key vertex from the
   set of vertices
59             // not yet included in MST
60             int u = minKey(key, mstSet);
61 
62             // Add the picked vertex to the MST Set
63             mstSet[u] = true;
64 
65             // Update key value and parent index of
   the adjacent
66             // vertices of the picked vertex.
   Consider only those
67             // vertices which are not yet included in
    MST
68             for (int v = 0; v < V; v++)
69 
70                 // graph[u][v] is non zero only for
   adjacent vertices of m
71                 // mstSet[v] is false for vertices
   not yet included in MST
72                 // Update the key only if graph[u][v
   ] is smaller than key[v]
```

```
 73                 if (graph[u][v] != 0 && mstSet[v
    ] == false && graph[u][v] < key[v]) {
 74                     parent[v] = u;
 75                     key[v] = graph[u][v];
 76                 }
 77         }
 78
 79         // print the constructed MST
 80         printMST(parent, graph);
 81     }
 82
 83     public static void main(String[] args)
 84     {
 85         /* Let us create the following graph
 86         2 3
 87         (0)--(1)--(2)
 88         | / \ |
 89         6| 8/ \5 |7
 90         | /  \ |
 91         (3)-------(4)
 92               9       */
 93         MST t = new MST();
 94         int graph[][] = new int[][] { { 0, 2, 0, 6
    , 0 },
 95                 { 2, 0, 3, 8, 5 },
 96                 { 0, 3, 0, 0, 7 },
 97                 { 6, 8, 0, 0, 9 },
 98                 { 0, 5, 7, 9, 0 } };
 99
100         // Print the solution
101         t.primMST(graph);
102     }
103 }
104
```