```java
import java.util.*;
import java.lang.*;
import java.io.*;

class Graph {
  class Edge implements Comparable<Edge>
  {
    int src, dest, weight;

    public int compareTo(Edge compareEdge)
    {
      return this.weight - compareEdge.weight;
    }
  };

  class subset
  {
    int parent, rank;
  };

  int V, E;
  Edge edge[];

  Graph(int v, int e)
  {
    V = v;
    E = e;
    edge = new Edge[E];
    for (int i = 0; i < e; ++i)
      edge[i] = new Edge();
  }

  int find(subset subsets[], int i)
  {
    if (subsets[i].parent != i)
      subsets[i].parent
        = find(subsets, subsets[i].parent);

    return subsets[i].parent;
  }

  void Union(subset subsets[], int x, int y)
  {
    int xroot = find(subsets, x);
    int yroot = find(subsets, y);

    if (subsets[xroot].rank
      < subsets[yroot].rank)
      subsets[xroot].parent = yroot;
    else if (subsets[xroot].rank
        > subsets[yroot].rank)
      subsets[yroot].parent = xroot;

    else {
      subsets[yroot].parent = xroot;
      subsets[xroot].rank++;
    }
```

```
}

void KruskalMST()
{
  Edge result[] = new Edge[V];

  int e = 0;

  int i = 0;
  for (i = 0; i < V; ++i)
    result[i] = new Edge();

  Arrays.sort(edge);

  subset subsets[] = new subset[V];
  for (i = 0; i < V; ++i)
    subsets[i] = new subset();

  for (int v = 0; v < V; ++v)
  {
    subsets[v].parent = v;
    subsets[v].rank = 0;
  }

  i = 0;

  while (e < V - 1)
  {
    Edge next_edge = edge[i++];

    int x = find(subsets, next_edge.src);
    int y = find(subsets, next_edge.dest);

    if (x != y) {
      result[e++] = next_edge;
      Union(subsets, x, y);
    }
  }

  System.out.println("Following are the edges in "
        + "the constructed MST");
  int minimumCost = 0;
  for (i = 0; i < e; ++i)
  {
    System.out.println(result[i].src + " -- "
          + result[i].dest
          + " == " + result[i].weight);
    minimumCost += result[i].weight;
  }
  System.out.println("Minimum Cost Spanning Tree "
        + minimumCost);
}

public static void main(String[] args)
{
  int V = 4;
  int E = 5;
  Graph graph = new Graph(V, E);
```

```
116
117     graph.edge[0].src = 0;
118     graph.edge[0].dest = 1;
119     graph.edge[0].weight = 10;
120
121     graph.edge[1].src = 0;
122     graph.edge[1].dest = 2;
123     graph.edge[1].weight = 6;
124
125     graph.edge[2].src = 0;
126     graph.edge[2].dest = 3;
127     graph.edge[2].weight = 5;
128
129     graph.edge[3].src = 1;
130     graph.edge[3].dest = 3;
131     graph.edge[3].weight = 15;
132
133     graph.edge[4].src = 2;
134     graph.edge[4].dest = 3;
135     graph.edge[4].weight = 4;
136
137     graph.KruskalMST();
138   }
139 }
```