```
import java.util.ArrayList;
import java.util.PriorityQueue;

public class Dijkshtra {

    public static class Edge {

        int src;
        int dest;
        int wt;

        public Edge(int s, int d, int wt){

        this.src=s;
        this.dest=d;
        this.wt=wt;

        }
    public static void createGraph(ArrayList<Edge> graph[]){

    for (int i = 0; i < graph.length; i++) {

        graph[i]= new ArrayList<Edge>();
    }

    graph[0].add(new Edge(0, 1,2));
    graph[0].add(new Edge(0, 2,4));

    graph[1].add(new Edge(1, 2,1));
    graph[1].add(new Edge(1, 3,7));

    graph[2].add(new Edge(2, 4,3));

    graph[3].add(new Edge(3, 5,1));

    graph[4].add(new Edge(4, 3,2));
    graph[4].add(new Edge(4, 5,5));

    }

    public static class Pair implements Comparable <Pair>{

        int node;
        int dist;

        public Pair(int n, int d){
            this.node=n;
```

```
            this.dist=d;
        }

        @Override
        public int compareTo(Pair p2) {

            return this.dist-p2.dist; //ascending order
        }

    }

    public static void priorityQueue(ArrayList<Edge> graph[], int src, int v){

    PriorityQueue <Pair> pq= new PriorityQueue<Pair>();
    int dist []= new int [v];

    for (int i = 0; i < v; i++) {
        if (i != src) {
            dist[i] = Integer.MAX_VALUE;
        } else {
            dist[i] = 0;
        }
    }
    boolean visited []= new boolean[v];

    pq.add(new Pair(0, 0));

    while (!pq.isEmpty()) {

        Pair curr=pq.remove();

        if (!visited[curr.node]) {

            visited[curr.node]=true;

            for (int i = 0; i < graph[curr.node].size(); i++) {

                Edge e= graph[curr.node].get(i);
                int u=e.src;
                int wt=e.wt;
                int vi=e.dest;

                if (dist[u]+wt<dist[vi]) {

                dist[vi]=dist[u]+wt;
                pq.add(new Pair(vi, dist[vi]));
```

```
                }

            }
        }

    }
    System.out.println();
        for (int i = 0; i <v; i++) {

            System.out.print(dist[i]+" ");
    }


    }



    public static void main(String[] args) {


        int v=6;
        System.out.println();
        ArrayList<Edge> graph[]= new ArrayList[v];
        createGraph(graph);

        priorityQueue(graph, 0, v);

    }

    }
}
```