import java.util.HashSet;

class Edge {

int src;

int dest;

int weight;

Edge(int src, int dest, int weight) {

this.src = src;

this.dest = dest;

this.weight = weight;

}

Edge(int src, int dest) {

this(src, dest, 1); // Default weight = 1

}

}

class Graph {

HashSet<Edge> graph;

int vertices;

boolean isDirected;

Graph(int vertices, boolean isDirected) {

this.vertices = vertices;

this.isDirected = isDirected;

graph = new HashSet<>();

}

public void addEdge(int src, int dest) {

if (isValid(src, dest)) {

Edge e = new Edge(src, dest);

graph.add(e);

if (!isDirected) {

Edge e2 = new Edge(dest, src);

graph.add(e2);

}

} else {

System.out.println("Invalid source and destination.");

}

}

public void addEdge(int src, int dest, int weight) {

if (isValid(src, dest)) {

Edge e = new Edge(src, dest, weight);

graph.add(e);

if (!isDirected) {

Edge e2 = new Edge(dest, src, weight);

graph.add(e2);

}

} else {

System.out.println("Invalid source and destination.");

}

}

private boolean isValid(int src, int dest) {

return src >= 0 && dest >= 0 && src < vertices && dest < vertices && src != dest;

}

}

public class Main {

public static void main(String[] args) {

Graph g = new Graph(5, false); // Undirected graph with 5 vertices

g.addEdge(0, 1);

g.addEdge(0, 2, 10);

g.addEdge(3, 4);

// You can add code here to display or test the graph

System.out.println("Edges in the graph:");

for (Edge e : g.graph) {

System.out.println(e.src + " -> " + e.dest + " (weight: " + e.weight + ")");

}

}

}