class Main {

public static void main(String[] args) {

Graph graph = new Graph(4, true); // Removed named parameters (Java doesn't support them)

graph.addEdge(0, 1);

graph.addEdge(0, 2);

graph.addEdge(0, 3);

graph.addEdge(1, 2);

System.out.println(graph.neighbors(1)); // fixed method name typo

graph.printGraph(); // method name corrected

}

}

class Graph {

int[][] graph;

int vertices;

boolean isDirected;

Graph(int vertices, boolean isDirected) {

graph = new int[vertices][vertices];

this.vertices = vertices;

this.isDirected = isDirected;

}

void removeEdge(int src, int dest) {

if (isValid(src, dest)) {

graph[src][dest] = 0;

if (!isDirected) {

graph[dest][src] = 0;

}

} else {

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

}

}

void addEdge(int src, int dest) {

if (isValid(src, dest)) {

graph[src][dest] = 1;

if (!isDirected) {

graph[dest][src] = 1;

}

} else {

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

}

}

// Method renamed from invalid to isValid and logic reversed for clarity

boolean isValid(int src, int dest) {

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

}

// Added neighbors method which returns a list of neighbors for a node

java.util.List<Integer> neighbors(int node) {

java.util.List<Integer> result = new java.util.ArrayList<>();

if (node >= 0 && node < vertices) {

for (int i = 0; i < vertices; i++) {

if (graph[node][i] == 1) {

result.add(i);

}

}

}

return result;

}

void printGraph() {

System.out.println("Matrix representation of graph:");

for (int i = 0; i < vertices; i++) {

for (int j = 0; j < vertices; j++) {

System.out.print(graph[i][j] + " ");

}

System.out.println();

}

}

}