1 import java.io.\*;  
 2 import java.util.\*;  
 3   
 4 /\*\*  
 5 \* @author Amar Bessedik  
 6 \* This class designs a graph. Each instance reads in data   
 7 \* which represents a graph from a text file.  
 8 \* Tests the validity of the data.  
 9 \* Extracts the number of vertices from the first line of the file.  
10 \* and generates an edge from each subsequent line.  
11 \* Put the constructed edges into a list.  
12 \*/  
13 public final class Graph {  
14 private ArrayList<Edge> edges;  
15 private int numOfVertices;  
16 private String data;  
17   
18 /\*\*  
19 \* Constructor  
20 \* @param filename  
21 \* DESIGN CHOICES:  
22 \* An list is used as we don't necessarily know how many edges are there.  
23 \* This saves O(n) time that is needed to count the edges first.  
24 \*/  
25 public Graph(String filename) {  
26 this.data = filename;  
27 this.edges = new ArrayList<>();  
28 this.numOfVertices = 0;  
29 readData(data);  
30 }//end Constructor  
31   
32 /\*\* @param data \*/  
33 public void readData(String data) {  
34 int vertex1, vertex2, weight;  
35   
36 try {  
37 Scanner reader = new Scanner(new File(data));  
38 //read first line and get # of vertices  
39 numOfVertices = Integer.parseInt(reader.nextLine());  
40   
41 //A line has the form: VERTEX1 VERTEX2 WEIGHT  
42 while (reader.hasNextInt()) {  
43 vertex1 = reader.nextInt();  
44 vertex2 = reader.nextInt();  
45 weight = reader.nextInt();  
46   
47 //Alert if a given edge's parameters are invalid   
48 //such as vertices are the same or weight is negative  
49 validateEdge(vertex1, vertex2, weight);  
50 //Generate an edge and add it to the list of edges  
51 edges.add(new Edge(vertex1, vertex2, weight));  
52 }  
53 } catch (FileNotFoundException | NumberFormatException e) {  
54 System.out.println(e);  
55 }  
56 }//end readData  
57   
58 /\*\* @return all edges of the graph as an array \*/  
59 public Edge[] getEdges() {  
60 return edges.toArray(new Edge[edges.size()]);  
61 }//end getEdges  
62   
63 /\*\* @return the number of vertices [1 ... n] \*/  
64 public int getVertices() {  
65 return numOfVertices;  
66 }//end getVertices  
67   
68 /\*\*@param vertex1 one the vertices of the graph  
69 \* @param vertex2 the other vertex  
70 \* @param weight of an edge  
71 \*/  
72 public void validateEdge(int vertex1, int vertex2, int weight) {  
73 if ((vertex1 == vertex2) || (vertex1 < 1 || vertex2 < 1)  
74 || (vertex1 > numOfVertices || vertex2 > numOfVertices) || (weight < 0)) {  
75 throw new IllegalArgumentException("Invalid data");  
76 }  
77 }//end validateEdge  
78 }//end Graph class  
79