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

/**
 * @author Amar Bessedik
 * This class designs a graph. Each instance reads in data
 * which represents a graph from a text file.
 * Tests the validity of the data.
 * Extracts the number of vertices from the first line of the file.
 * and generates an edge from each subsequent line.
 * Put the constructed edges into a list.
 */
public final class Graph {
   private ArrayList<Edge> edges;
   private int numOfVertices;
   private String data;

   /**
    * Constructor
    * @param filename
    * DESIGN CHOICES:
    * An list is used as we don't necessarily know how many edges are there.
    * This saves O(n) time that is needed to count the edges first.
    */
   public Graph(String filename) {
      this.data = filename;
      this.edges = new ArrayList<>();
      this.numOfVertices = 0;
      readData(data);
   }//end Constructor

   /** @param data */
   public void readData(String data) {
      int vertex1, vertex2, weight;

      try {
         Scanner reader = new Scanner(new File(data));
         //read first line and get # of vertices
         numOfVertices = Integer.parseInt(reader.nextLine());

         //A line has the form: VERTEX1 VERTEX2 WEIGHT
         while (reader.hasNextInt()) {
            vertex1 = reader.nextInt();
            vertex2 = reader.nextInt();
            weight  = reader.nextInt();

            //Alert if a given edge's parameters are invalid
            //such as vertices are the same or weight is negative
            validateEdge(vertex1, vertex2, weight);
            //Generate an edge and add it to the list of edges
            edges.add(new Edge(vertex1, vertex2, weight));
         }
      } catch (FileNotFoundException | NumberFormatException e) {
         System.out.println(e);
      }
   }//end readData

```

```
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
```