Data Structures
Final Individual Exercise 1

Required:

Organize the following classes in a project. Provide appropriate javadoc comments and other comments. Show the output of the main class in a multiline comment above the declaration of the main class.

```
package graphdemo1;

public class Edge {
  private Node start;
  private Node end;
  private double weight;
  private int id;
  public int getId() {
    return this.id;
  }
  public Node getStart() {
    return this.start;
  }
  public int getIdOfStartNode() {
    return this.start.getNodeId();
  }
  public Node getEnd() {
    return this.end;
  }
  public int getIdOfEndNode() {
    return this.end.getNodeId();
  }

  public double getWeight() {
    return this.weight;
  }
  public Edge(Node s, Node e, double w, int id) {
    this.start = s;
    this.end = e;
    this.weight = w;
    this.id = id;
  }

  public String toString(){
    return ("("+start+","+end+")");
  }
}
```

```
package graphdemo1;
import java.util.*;
public class Node {
  private int id;
  private List<Edge> neighbours = new ArrayList<Edge>();
  public int getNodeId() {
    return this.id;
  }
  public void addNeighbour(Edge e) {
    if(this.neighbours.contains(e)) {
      System.out.println("This edge has already been used for this node.");
    } else {
      System.out.println("Successfully added " + e);
      this.neighbours.add(e);
    }
  }
  public void getNeighbours() {
    System.out.println("List of all edges that node " + this.id +" has: ");
    System.out.println("================================");
    for (int i = 0; i < this.neighbours.size(); i++ ){
      System.out.println("ID of Edge: " + neighbours.get(i).getId() + "\nID of the first node: " +
neighbours.get(i).getIdOfStartNode() +
          "\nID of the second node: " + neighbours.get(i).getIdOfEndNode());
      System.out.println();
    }
    System.out.println(neighbours);
  }
  public Node(int id) {
    this.id = id;
  }

  public String toString(){
    return ""+id;
  }
}
```

```java
package graphdemo1;

import java.util.*;
public class Graph {
  private List<Node> nodes = new ArrayList<Node>();
  private int numberOfNodes = 0;
  public boolean checkForAvailability() { // will be used in TestGraph.java
    return this.numberOfNodes > 1;
  }
  public void createNode(Node node) {
    this.nodes.add(node);
    this.numberOfNodes++; // a node has been added
  }
  public int getNumberOfNodes() {
    return this.numberOfNodes;
  }
}
```

—----------

```java
package graphdemo1;

public class TestGraph {
  public static void main(String args[]) {
    Graph graph = new Graph();
    Node node1 = new Node(1); // create a new node that contains id of 1
    Node node2 = new Node(2); // create a new node that contains id of 2
    Node node3 = new Node(3); // create a new node that contains id of 3
    graph.createNode(node1); // numberOfNodes should increment by 1
    graph.createNode(node2); // numberOfNodes should increment by 1
    graph.createNode(node3); // numberOfNodes should increment by 1
    Edge e12 = new Edge(node1, node2, 5, 1); // create an edge that connects node1 to node2 and contains weight of 5
    Edge e13 = new Edge(node1, node3, 10, 2); // create an edge that connects node1 to node3 and contains weight of 10
    if (graph.checkForAvailability()) {
      // two nodes can be connected via edge
      node1.addNeighbour(e12); // connect 1 and 2 (nodes)
      node1.addNeighbour(e13);
      node1.getNeighbours();
    } else {
      System.out.println("There are less than 2 nodes. Add more to connect.");
    }
  }
}
```