```
package hw4;
import java.util.*;

/** Graph represents a mutable graph of nodes with directed edges.*/
public class Graph {

    private ArrayList<ArrayList<ArrayList<String>>> matrix;
    private ArrayList<String> names;
    private int num_edges;
    private int num_nodes;

    // Abstraction Function:
    //   A Graph g is a arraylist matrix of strings, such that for matrix[x][y][z],
    //   x is the origin node and y is the node pointed to, with z being the
arraylist of nodes with parent x and child y.
    //   An empty arraylist in the adjacency matrix = no edge between node, and all
else signifies a present edge.

    // Representation invariant:
    //   assert (num_edges >= 0 && num_nodes >= 0);
    //
    //   Set<String> nameSet = new HashSet<>(names);
    //   assert (names.size() == nameSet.size());
    //
    //  for (int i = 0; i < matrix.get(0).size(); i++){
    //      for (int j = 0; j < matrix.get(0).size(); j++){
    //          Set<String> labelSet = new HashSet<>(matrix.get(i).get(j));
    //          assert(matrix.get(i).get(j).size() == labelSet.size());
    //      }
    //   }
    //
    //
    //   In other words,
    //     * There is always 0 or more nodes and edges (non negative)
    //     * There are no duplicates in names (nodes)
    //     * There are no duplicates edges (edges with same parent, same child, same
label)
    // (A representation invariant tells us something that is true for all valid
instances of a Graph)
```

```
    /** @effects Constructs a new Graph with zero nodes and zero edges.*/
    public Graph() {
        throw new RuntimeException("Not yet implemented.");
    }

    /** @param another another Graph to be copied.
        @requires another != null and this != null
        @effects Sets this new Graph's attributes to the same ones as another
Graph's. Includes number of nodes, number of edges, adjacency matrix, name of nodes.
    */
    public Graph(Graph another) {
        throw new RuntimeException("Not yet implemented.");
    }

    /**
     * Checks that the representation invariant holds.
     * @ensures representation invariant holds
     * @throws RuntimeException if the rep invariant is violated.
     **/
    // Throws a RuntimeException if the rep invariant is violated.
    private void checkRep() throws RuntimeException {
        throw new RuntimeException("Not yet implemented.");
    }

    /**@return number of nodes in the Graph
     * @requires this!=null
    */
    public int getNumNodes() {
        throw new RuntimeException("Not yet implemented.");
    }

     /** @return number of edges in the Graph
      * @requires this!=null
    */
    public int getNumEdges(){
        throw new RuntimeException("Not yet implemented.");
    }
    /** Node Search Operation.
        @param node
        @requires this!=null
        @return index if edge exists in Graph, return -1 if edge does not exist.
    */
    public int findNode(String node) {
        throw new RuntimeException("Not yet implemented.");
    }

    /** Node Addition Operation.
     * @requires this!=null
        @param s The label of the node to be added.
        @modifies Graph to insert node s into arraylist of names and adjacency matrix
of nodes.
        @modifies Resizes the matrix to be (old size + 1) * (old size + 1).
        @modifies Set Each new index as empty Arraylist to signify no edge present.
        Does nothing if node s is already in the Graph.
    */
    public void addNode(String s) {
        throw new RuntimeException("Not yet implemented.");
    }

    /** Edge Addition Operation.
     * @requires this!=null
        @param from The node which the edge stems from
        @param to The node which the edge points to
        @param label The label of the edge
        @throws RuntimeException if from or to don't exist as nodes in the list of
labels
        @modifies Adds the edge into matrix. Does nothing if edge with same label,
parent, and child already exists.
    */
    public void addEdge(String from, String to, String label){
        throw new RuntimeException("Not yet implemented.");
    }
```

```java
    /** Returns edge label.
     * @requires this!=null
        @param from The node which the edge stems from
        @param to The node which the edge points to
        @throws RuntimeException if from or to don't exist as nodes in the list of
labels
        @returns arraylist of edge labels of specified edge.
    */
    public ArrayList<String> getEdgeLabel(String from, String to){
        throw new RuntimeException("Not yet implemented.");
    }

    /** Gets list of children of Node s.
     * @requires this!=null
        @param s the node that we will find children of
        @throws RuntimeException if node s is not in the graph
        @return an arraylist of the names of the children
    */
    public ArrayList<String> getChildren(String s){
        throw new RuntimeException("Not yet implemented.");
    }

    /** Gets list of all nodes in Graph.
     * @requires this!=null
        @return a copy of names, the arraylist of all nodes in the graph
    */
    public ArrayList<String> getNodeList(){
        throw new RuntimeException("Not yet implemented.");
    }
    /**@requires this!=null
     * @return a String representing the graph.
        The returned string shows the adjacency matrix for the nodes
        Shows the edge label, and "m" if there is more than one. If there is nothing,
show "x".
    */
    @Override
    public String toString() {
        throw new RuntimeException("Not yet implemented.");
    }
}
```