Problem 1

Graph

```java
package hw4;

import java.util.*;

/**
 * <b>Graph</b> represents an <b>mutable</b>
 * graph consists of nodes and edges
 * It represents a multi-bidirectional graph
 */
public class Graph {
    private HashMap<String, HashSet<Edge>> graph;

    // Abstraction Function:
    // Make use of adjacent list, we have a map that represent the start
nodes corresponded with
    // a set of edges connected with it

    // Representation invariant for every Graph g:
    // forall list of graph.values() :: forall edge in list ::
graph.keys().contains(edge.getTo())
    // An empty graph is allowed

    /**
     * @effects Construct an empty graph
     */
    public Graph() {
        throw new RuntimeException("Graph constructor not implemented");
    }

    /**
     * add new nodes and a edge to the graph
     * @param a String represent Node which the edge starts at
     * @param b String represent Node which the edge ends at
     * @param edgeName String the name of the edge
     * @return boolean true iff the edge successfully added to the graph
     */
    public boolean connect(String a, String b, String edgeName) {
        throw new RuntimeException("Connect not implemented");
    }

    /**
     * Return a Set of edge that start from given Node a
     * @param a String represent Node where the edge start from
     * @return A set of edge that start from given Node a.
     *  An empty  set will return if node "a" cannot be found in the graph
     */
    public Set<Edge> connectedEdge(String a) {
        throw new RuntimeException("ConnectedEdge not implemented");
    }
}
```

Edge

```
package hw4;

/** <b>Edge</b> represents an <b>immutable</b> edge in the graph
 * it set up a bridge between two node, and can be labeled with a name
 * There many kinds of edge, include reflexible edge, which the Node "from"
and "to"
 *  reference to the same node
 * An edge can be compared by its name
 */
public class Edge {
    private final String from;
    private final String  to;
    private final String name = "";

    // Abstraction Function:
    // an edge that start from node "from" and ends at Node "to"
    // named with String "name"

    // Representation invariant for every Edge:
    // from != NULL && to != NULL && name != NULL

    /** @param a is the String represent node that  the edge start from
     * @param b is the String represent node that the edge ends at
          @effects Constructs a new Edge that connects Node a and Node b
         with name n
     */
    public Edge (String a, String b, String n) {
        throw new RuntimeException("Edge constructor Not Implemented");
    }

    /** Returns the start from node
          @return source node of this edge
     */
    public String getFrom() {
        throw new RuntimeException("getFrom Not Implemented");
    }

    /**
     *  Returns the end node of the edge
     * @return Node Destination of this edge
     */
    public String getTo() {
        throw new RuntimeException("getTo Not Implemented");
    }
    /**
     * Returns the name of the edge
     * @return String the name of this edge
     */
    public String getName() {
        throw new RuntimeException("getName Not Implemented");
    }
}
```