EE360T/EE382V: Software Testing
Problem Set 4

Out: Apr 3, 2017; **Due: Apr 12, 2017 11:59pm**
Submission: *.zip via Canvas
Maximum points: 40

1 Implementing a graph data structure

Consider the following partial implementation of a graph data structure:

```
package pset4;

import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

public class Graph {
    private Set<Integer> nodes; // set of nodes in the graph
    private Map<Integer, List<Integer>> edges;
    // map between a node and a list of nodes that are connected to it by outgoing edges
    // class invariant: fields "nodes" and "edges" are non-null;
    //                  "this" graph has no node that is not in "nodes"
    public Graph() {
        nodes = new HashSet<Integer>();
        edges = new HashMap<Integer, List<Integer>>();
    }
    public String toString() {
        return "nodes=" + nodes + "; " + "edges=" + edges;
    }
    public void addNode(int n) {
        // postcondition: adds the node "n" to this graph
        nodes.add(n);
    }
    public void addEdge(int from, int to) {
        // postcondition: adds a directed edge "from" -> "to" to this graph
        // your code goes here
        // ...
    }
    public boolean unreachable(Set<Integer> sources, Set<Integer> targets) {
        if (sources == null || targets == null) throw new IllegalArgumentException();
        // postcondition: returns true if (1) "sources" is a subset of "nodes", (2) "targets" is
        //                            a subset of "nodes", and (3) for each node "m" in set "targets",
        //                            there is no node "n" in set "sources" such that there is a
        //                            directed path that starts at "n" and ends at "m" in "this"; and
        //                            false otherwise
        // your code goes here
        // ...
    }
}
```

1.1 Implementing `addEdge` [4 points]

Implement the method `addEdge` as specified. Make sure your implementation satisfies the class invariant for `Graph` (as given in comments).

1.2 Implementing `unreachable` [11 points]

Implement the method `unreachable` (and any helper methods you need) as specified.

2 Testing your graph implementation [25 points]

Implement a test suite to test the `addEdge` and `unreachable` methods in the following class `GraphTester` as specified:

```
package pset4;

import static org.junit.Assert.*;

import java.util.HashSet;
import java.util.Set;

import org.junit.Test;

public class GraphTester {
    // tests for method "addEdge" in class "Graph"
    @Test public void tae0() {
        Graph g = new Graph();
        g.addEdge(0, 1);
        assertEquals(g.toString(), "nodes: [0, 1]\nedges: {0=[1]}");
    }

    // your tests for method "addEdge" in class "Graph" go here

    // you must provide at least 4 test methods;
    // each test method must have at least 1 invocation of addEdge;
    // each test method must have at least 1 test assertion;
    // your test methods must provide full statement coverage of your
    //   implementation of addEdge and any helper methods
    // no test method directly invokes any method that is not
    //   declared in the Graph class as given in this homework

    // tests for method "unreachable" in class "Graph"
    @Test public void tr0() {
        Graph g = new Graph();
        g.addNode(0);
        Set<Integer> nodes = new HashSet<Integer>();
        nodes.add(0);
        assertTrue(g.unreachable(new HashSet<Integer>(), nodes));
    }

    // your tests for method "unreachable" in class "Graph" go here

    // you must provide at least 6 test methods;
    // each test method must have at least 1 invocation of unreachable;
    // each test method must have at least 1 test assertion;
    // at least 2 test methods must have at least 1 invocation of addEdge;
    // your test methods must provide full statement coverage of your
    //   implementation of unreachable and any helper methods
    // no test method directly invokes any method that is not
    //   declared in the Graph class as given in this homework

}
```