1. (25%) Presume there is a version of EmptyList.java in which setNext, getNext, setItem, and getItem all throw EmptyListE (which extends Exception.java), and the following (assume item1 and item2 are non-null).

EmptyList empty = new EmptyList();
Node node1 = new Node(item1, empty);
Node node2 = new Node(item2, empty);
List l;

The variable l is non-null, and has either been assigned some node or is empty and you don’t know which. Do not make inquiries into the contents of l (i.e. do not call any getters or use instanceof). Write code that will attempt to perform l.setNext(node2) and if that fails because of an EmptyListE, alternatively sets l = node1 and then retry l.setNext(node2).

2. The following compiles. What would be the output? Use a sequence diagram in your answer, showing the succession of method calls and where the flow of control goes during the execution. Use the following blank page to answer.

import java.util.Observable;
import java.util.Observer;

class Blame extends Exception {
    Politician accuser;
    Politician accused;

    public Blame(String excuse, Politician accuser, Politician accused) {
        super(excuse);
        this.accuser = accuser;
        this.accused = accused;
    }

    Politician getAccuser() { return accuser; }
    Politician getAccused() { return accused; }
}

abstract class Politician {
    abstract void antagonize(Politician p) throws Blame;
}

class Republican extends Politician {
    void antagonize(Politician p) throws Blame {
        throw new Blame("Democrats are soft on security", this, p);
    }
}

class Democrat extends Politician {
    void antagonize(Politician p) throws Blame {
        throw new Blame("Republicans are destroying the environment", this, p);
    }
}
class BlameGame {
    public static void main (String[] args) {
        Politician p1 = new Democrat();
        Politician p2 = new Republican();

        while (true) {
            try { p1.antagonize(p2); }
            catch (Blame b) {
                System.err.println(b.getMessage());
                p2 = b.getAccuser();
                p1 = b.getAccused();
            }
        }
    }
}

3. (25%) Class C has method public void m() which can throw two types of exception (BadLuckE and BadKarmaE). Write method public void attemptM(C c) which calls m() on the argument c. If a BadLuckE occurs, it just prints “oops!”. If a BadKarmaE occurs, it prints “fooie!” and second call m() is made. Note that attemptM() does not throw any exceptions.

    public void attemptM(C c) {

4. (25%) Assume class A implements interface I, and the implementation of method m is complicated, proprietary, and not to be replicated.

    public interface I { public void m(); }

Next, class B1 extends class B. Classes A and B are unrelated (neither extends the other). Use the Adaptor Pattern to have B1 implement I using the version of m in A.

    public class B1 extends B implements I {