EE360T/EE382V: Software Testing
Problem Set 2

Out: March 2, 2016; **Due: March 12, 2016 11:59pm**
Submission: *.zip via Canvas
Maximum points: 40

1 Testing contracts

Consider the following code snippet that declares a class `C`:

```
package pset2;

public class C {
    int f;

    public C(int f) {
        this.f = f;
    }

    @Override
    public boolean equals(Object o) {
        // returns a boolean consistent with the Java contract for equals method;
        // returns true if and only if o is an object of class
        // C and has the same value for field f as this.f

        // your code goes here

    }

    @Override
    public int hashCode() {
        // returns an integer consistent with the Java contract for hashCode method;
        // does not return a constant value

        // your code goes here

    }
}
```

Consider next the following code snippet that declares a class `D` as a subclass of `C`:

```
package pset2;

public class D extends C {
    int g;

    public D(int f, int g) {
        super(f);
        this.g = g;
    }

    @Override
    public boolean equals(Object o) {
```

```
        // returns a boolean consistent with the Java contract for equals method;
        // returns true if and only if o is an object of class D and
        // has the same value for field f as this.f and the same value
        // for field g as this.g

        // your code goes here

    }

    @Override
    public int hashCode() {
        // returns an integer consistent with the Java contract for hashCode method;
        // does not return a constant value

        // your code goes here

    }
}
```

According to the contract for `java.lang.Object`, any correct Java program must satisfy certain properties with respect to the `equals` methods; these properties include[1]:

P_1: For any non-null reference value x, $x.equals(null)$ should return false;

P_2: It is reflexive: for any non-null reference value x, $x.equals(x)$ should return true;

P_3: It is symmetric: for any non-null reference values x and y, $x.equals(y)$ should return true if and only if $y.equals(x)$ returns true; and

P_4: It is transitive: for any non-null reference values x, y, and z, if $x.equals(y)$ returns true and $y.equals(z)$ returns true, then $x.equals(z)$ should return true;

1.1 Implementing `equals` and `hashCode` [8 points]

Complete the implementations of classes `C` and `D` as specified in the comments.

1.2 Testing `equals` [8 points]

You are to implement a test suite that checks one of the four properties – namely P_4 – with respect to the `equals` methods implemented in the three classes `pset2.C`, `pset2.D`, and `java.lang.Object`. Specifically, implement test methods in the following class `EqualsTester` such that: (1) each test method has exactly one invocation of `assertTrue(...)` or `assertFalse(...)`; (2) the property is tested with respect to each of the three `equals` methods, i.e., each `equals` method must be executed by some test; (3) each property is tested with respect to each combination of the three object types (`C`, `D`, or `Object`) for the inputs to `equals`, i.e., each `equals` method must be executed using an object of each type as actual argument by some test:

```
package pset2;

import static org.junit.Assert.*;
import org.junit.Test;

public class EqualsTester {
    /*
     * P1: For any non-null reference value x, x.equals(null) should return false.
     */

    // an example test for P1
    @Test public void t0() {
        assertFalse(new Object().equals(null));
```

[1] http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html

```
    }

    /*
     * P4: It is transitive: for any non-null reference values x, y, and z,
     * if x.equals(y) returns true and y.equals(z) returns true, then
     * x.equals(z) should return true.
     */

    // your test methods for P4 go here

}
```

1.3 Testing `hashCode` [4 points]

The contract for `java.lang.Object` additionally requires the following property that relates `equals` and `hashCode`[1]:

> P_5 : If two objects are equal according to the *equals*(*Object*) method, then calling the *hashCode* method on each of the two objects must produce the same integer result.

Implement test methods in the following class `HashCodeTester` such that: (1) each test method has exactly one invocation of `assertTrue(...)` or `assertFalse(...)`; (2) the property is tested with respect to each `hashCode` method, i.e., each `hashCode` method must be executed by some test:

```
package pset2;

import static org.junit.Assert.*;
import org.junit.Test;

public class HashCodeTester {
    /*
     * P5: If two objects are equal according to the equals(Object)
     * method, then calling the hashCode method on each of
     * the two objects must produce the same integer result.
     */

     // your test methods go here

}
```

2 Textbook exercises

Solve the following problems from the Software Testing textbook:

1. [7 points] Exercises – Section 3.3 Question 3 – answer this question with respect to CACC instead of GACC (Pages 130–131)
2. [6 points] Exercises – Section 4.1 Question 2 (Page 159) – answer this question with respect to only the method Push using exactly two characteristics where each characteristic has two blocks in its partition
3. [7 points] Exercises – Section 4.2 Question 4 (Pages 163–164)