Assignment 3

Rasika Mohod
rmohod@gmu.edu
G01044774

Introduction to Software Testing (Edition 2): Book by Jeff Offutt and Paul Amman
Exercises Chapter 3, Number 9; Page 52.

Below given is the original code written to check if two points are equal:

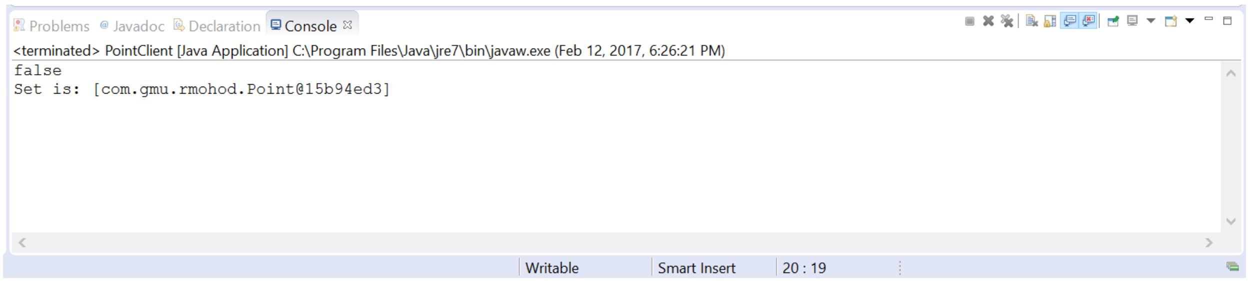
Point.java

```
package com.gmu.rmohod;

/**
 * This class implements a Point with two co-ordinates x & y
 *
 * @author Rasika
 *
 */
public class Point {

	private int x;
	private int y;

	public Point(int x, int y)
	{
		this.x = x;
		this.y = y;
	}

	@Override
	public boolean equals(Object o)
	{
		if (o == null){
			return false;
		}

		if (!(o instanceof Point)){
			return false;
		}

		Point p = (Point)o;
		return p.x == x && p.y == y;
	}
}
```

PointClient.java

```
package com.gmu.rmohod;

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

/**
 * This class is Client class for Class Point.
 * This class exercises the operations on Point class.
 *
 * @author Rasika
 *
 */
public class PointClient {
      public static void main (String args[])
      {
            Point p1 = new Point(10,100);
            Point p2 = new Point(10,100);

            Set<Point> s = new HashSet<Point>();
            s.add(p1);
            System.out.println(s.contains(p2));
            System.out.println("Set is: "+ s);
      }
}
```

The output observed after execution is as follows:



```
<terminated> PointClient [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Feb 12, 2017, 6:26:21 PM)
false
Set is: [com.gmu.rmohod.Point@15b94ed3]
```

a) The problem with Point using HashSet:

- Even if it is possible for set to contain same point equal to p2, these two point objects might be mapped to different hash values.
- The failure is thus to override the hashCode method.
  According to the contract every class that overrides equals must override hashCode. Failure to do so prevents class from functioning properly in conjunction with all hash based collections, including HashMap, HashSet and Hashtable. Here, the code thus gives wrong result with the equals method returning false while checking the reflexivity condition.

b) Mathematical relationship between equals() and hashCode():

- If two objects are considered to be equal, as implemented by equals() method then they both should have same hash values, as computed in hashCode() method.

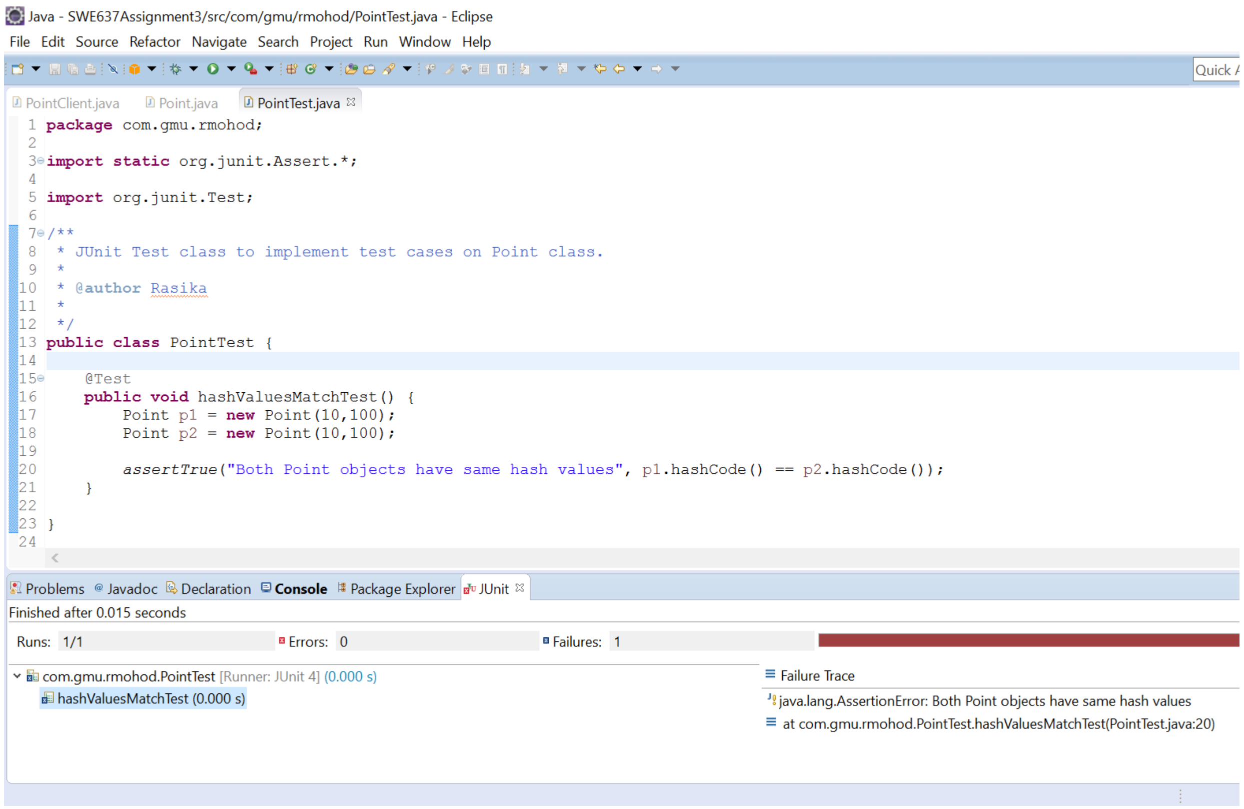
- However, if two objects have same hash values, they do not necessarily be equal objects.

c) JUnit Test case to show that Point object does not enjoy this property:

PointTest.java

```
package com.gmu.rmohod;

import static org.junit.Assert.*;

import org.junit.Test;

/**
 * JUnit Test class to implement test cases on Point class.
 *
 * @author Rasika
 *
 */
public class PointTest {

	@Test
	public void hashValuesMatchTest() {
		Point p1 = new Point(10,100);
		Point p2 = new Point(10,100);

		assertTrue("Both Point objects have same hash values",
p1.hashCode() == p2.hashCode());
	}

}
```

The output of Junit test execution: Test failed (no same hash values found)



Java - SWE637Assignment3/src/com/gmu/rmohod/PointTest.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
PointClient.java
Point.java
PointTest.java
Problems
Javadoc
Declaration
Console
Package Explorer
JUnit

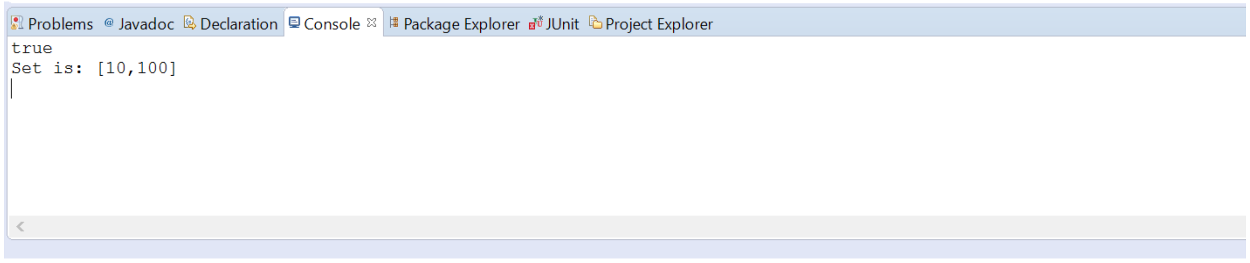
```java
package com.gmu.rmohod;

import static org.junit.Assert.*;

import org.junit.Test;

/**
 * JUnit Test class to implement test cases on Point class.
 *
 * @author Rasika
 *
 */
public class PointTest {

    @Test
    public void hashValuesMatchTest() {
        Point p1 = new Point(10,100);
        Point p2 = new Point(10,100);

        assertTrue("Both Point objects have same hash values", p1.hashCode() == p2.hashCode());
    }

}
```

```
Finished after 0.015 seconds
Runs: 1/1    Errors: 0    Failures: 1

com.gmu.rmohod.PointTest [Runner: JUnit 4] (0.000 s)
    hashValuesMatchTest (0.000 s)

Failure Trace
java.lang.AssertionError: Both Point objects have same hash values
at com.gmu.rmohod.PointTest.hashValuesMatchTest(PointTest.java:20)
```

d) Fault Repair: Override hashCode method and also toString() method

```java
@Override
public int hashCode() {
        int result = 1;
        result = result * 31 + x;
        result = result * 31 + y;
        return result;
}

@Override public String toString() {
        String result = "";
        result = x+","+y;
        return result;
}
```

The output observer after execution of repaired code is as follows:



```
true
Set is: [10,100]
```

e) Rewrite Junit Test Case:

PointTest.java

```
package com.gmu.rmohod;

import static org.junit.Assert.*;

import org.junit.Test;
import org.junit.experimental.theories.DataPoints;
import org.junit.experimental.theories.Theories;
import org.junit.experimental.theories.Theory;
import org.junit.runner.RunWith;

/**
 * JUnit Test class to implement test cases on Point class.
 *
 * @author Rasika
 *
 */
@RunWith(Theories.class)
public class PointTest {

	@DataPoints
	public static Object[] objects = { new Point(10,100), new
	Point(10,100), "SWE637", null, new Point(100,10) };

	@Theory
	public void hashValuesMatchTheoryestTest(Object o1, Object o2) {
		assumeTrue(o1 != null);
		assumeTrue(o2 != null);
		assumeTrue(o1.equals(o2));
		assertTrue("Both Point objects have same hash values",
		o1.hashCode() == o2.hashCode());
	}

}
```

This theory will be evaluated for total 25 values (5x5 for 5 data points).

- 5 values will not go beyond first condition (because of the datapoint: o1 = `null)`
- 4 values will not go beyond second condition (because of the datapoint: o2 = `null)`
- 10 values will not go beyond third condition
- 6 values will satisfy the final condition (because of datapoints: `new Point(10,100),` `new Point(10,100)` with itself and each other and `"SWE637", new Point(100,10)` with itself)