**SE 317**

**Lab 2 Group 2**

**\*Do not change the code with fault in the classes.**

1. **NumZero**

Step 1: Run and understand the NumZero.java in the group2 package.

Step 2: Create a JUnit test case for NumZero and name it NumZeroTest. Copy and Run the test case below.

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

*import org.junit.\*;*

*import java.util.\*;*

*public class NumZeroTest*

*{*

*// this test passes*

*@Test public void nonZeroFirstElement()*

*{*

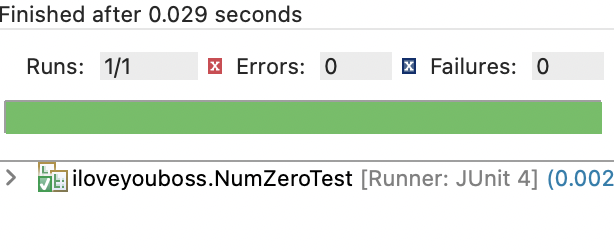
*int arr[] = {2, 7, 0};*

*assertEquals("Nonzero in first element", 1, NumZero.numZero(arr));*

*}*

*}*

After running, the JUnit will pass, not revealing the error.

**

*Example of a bad test*

Step 3: Read the code and identify the error in NumZeroTest.java.

Step 4:

* + 1. What causes the test to fail? Describe the fault in the code.
    2. Write a test that reveals the error in NumZero.java.
    3. Take screenshots of your work showing the bad and good test results

1. **Point, ColorPoint**

Step 1: Run and understand the Point.java and ColorPoint.java in the group2 package.

Step 2: Create a JUnit test case for Point and ColorPoint and name it PointTest. Copy and Run the test case below.

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

*import org.junit.\*;*

*import java.util.\*;*

*public class PointTest {*

*private Point p = new Point(1,2);*

*private ColorPoint cp1 = new ColorPoint(1,2,Color.RED);*

*private ColorPoint cp2 = new ColorPoint(1,2,Color.BLUE);*

*// this test passes*

*@Test public void transitivity() {*

*if (cp1.equals(p) && p.equals(cp2)) {*

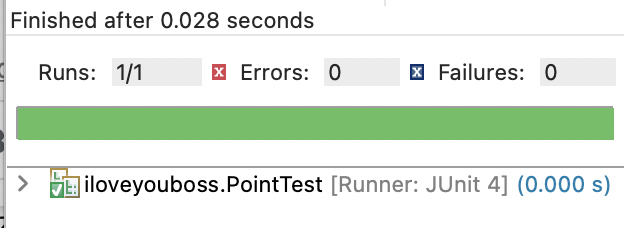
*assertTrue(cp1.equals(cp2));*

*}*

*}*

*}*

After running, the JUnit will pass, not revealing the error.

**

*Example of a bad test*

Step 3: Read the code and identify the error in Point.java and ColorPoint.java

Step 4:

* + 1. What causes the test to fail? Describe the fault in the code.
    2. Write a test that reveals the error in Point.java and ColorPoint.java.
    3. Take screenshots of your work showing the bad and good test results

**(c) Truck, Vehicle**

Step 1: Run and understand the Truck.java and Vehicle.java in the group2 package.

Step 2: Create a JUnit test case for them and name it CloneTest. Copy and Run the test case below.

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

*import org.junit.\*;*

*import java.util.\*;*

*public class CloneTest*

*{*

*// this test passes*

*@Test public void cloneSuper() {*

*Vehicle v = new Vehicle(4);*

*Vehicle w = (Vehicle) v.clone();*

*assertFalse(v == w);*

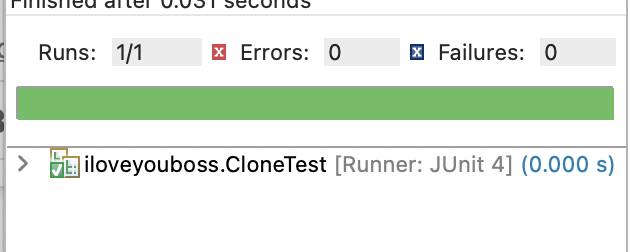
*assertEquals(v.getClass(), w.getClass());*

*assertTrue(v.equals(w));*

*}*

*}*

After running, the JUnit will pass, not revealing the error.

**

*Example of a bad test*

Step 3: Read the code and identify the error in Truck.java and Vehicle.java.

Step 4: Write a test that reveals the error in them. What causes the test to fail? Describe the fault of the code.

Step 4:

* + 1. What causes the test to fail? Describe the fault in the code.
    2. Write a test that reveals the error in Truck.java and Vehicle.java.
    3. Take screenshots of your work showing the bad and good test results

**(d) BigDecimalTest**

Step 1: Create a JUnit test case and name it BigDecimalTest. Copy and Run the test case below.

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

*import org.junit.\*;*

*import java.util.\*;*

*import java.math.\*;*

*public class BigDecimalTest {*

*private BigDecimal x;*

*private BigDecimal y;*

*Set <BigDecimal> tree;*

*Set <BigDecimal> hash;*

*@Before public void setUp() {*

*x = new BigDecimal("1.0");*

*y = new BigDecimal("1.0");*

*// Fact: !x.equals(y), but x.compareTo(y) == 0*

*tree = new TreeSet <BigDecimal> ();*

*hash = new HashSet <BigDecimal> ();*

*}*

*@Test public void inconsistentSets() {*

*tree.add(x); tree.add(y);*

*// TreeSet uses compareTo()*

*hash.add(x); hash.add(y);*

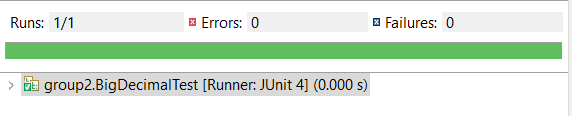
*// HashSet uses equals()*

*assertEquals(tree, hash);*

*}*

*}*

After running, the JUnit will pass, not revealing the error.



*Example of a bad test*

Step 2:

* + 1. What causes the test to fail? Describe the fault in the code.
    2. Write a test that reveals the error in the code.
    3. Take screenshots of your work showing the bad and good test results