**SE 317**

**Lab 2 Group 1**

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

1. **Count Positive**

Step 1: Run and understand the CountPositive.java in the group1 package.

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

*package group1;*

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

*import org.junit.\*;*

*import java.util.\*;*

*public class CountPositiveTest {*

*@Test public void arrayContainsZeroes()*

*{*

*int arr[] = {-4, 2, -1, 2};*

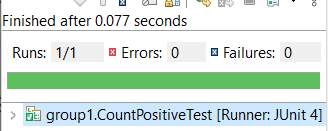
*assertEquals("Array contains zeroes",2, CountPositive.countPositive(arr));*

*}*

*}*

After running the code the code, the JUnit will pass successfully and will not reveal any errors. This does not mean the code is error-free.

Remember: Good tests fail

**

*Example of a bad test*

Step 3: Carefully inspect the code and identify the error in CountPositive.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 CountPositive.java.
    3. Take screenshots of your work showing the bad and good test results

1. **FindLast**

Step 1: Run and understand the FindLast.java in the group1 package.

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

***public******class*** *FindLastTest {*

*@Test*

***public******void*** *lastOccurrenceInFirstElement()*

*{*

***int*** *arr[] = {2, 3, 5};*

***int*** *y = 3;*

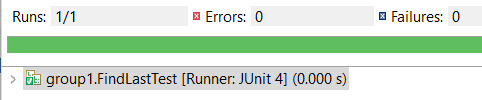
*assertEquals("Last occurence in first element", 1, FindLast.findLast(arr, y));*

*}*

*}*

After running, the JUnit will pass successfully and will not reveal any errors. This does not mean the code is error free.

Remember: Good tests fail

**

*Example of a bad test*

Step 3: Carefully inspect the code and identify the error in FindLast.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 FindLast.java.
    3. Take screenshots of your work showing the bad and good test results.

**(c) LastZero**

Step 1: Run and understand the LastZero.java in the group1 package.

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

*public class LastZeroTest {*

*@Test public void multipleZeroes()*

*{*

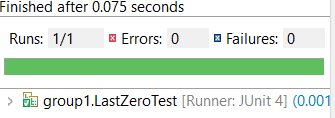
*int arr[] = {0, 1, 0};*

*assertEquals("Multiple zeroes: should find last one", 0, lastZero.lastZero(arr));*

*}*

*}*

After running the code, the JUnit will pass successfully and will not reveal any errors. This does not mean the code is error free.

**

*Example of a bad test*

Step 3: Carefully inspect the code and identify the error in LastZero.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 LastZero.java.
    3. Take screenshots of your work showing the bad and good test results.

**(d) OddOrPos**

Step 1: Run and understand the OddOrPos.java in the group1 package.

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

*public class OddOrPostest {*

*@Test public void negativeOddNumbers()*

*{*

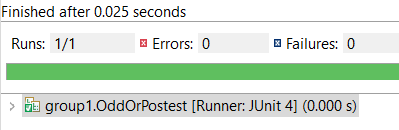
*int arr[] = {1, 2, 3};*

*assertEquals("Negative odd numbers in array", 3, OddOrPos.oddOrPos(arr));*

*}*

*}*

After running, the JUnit passed and did not reveal the error.

**

*Example of a bad test*

Step 3: Carefully inspect the code and identify the error in OddOrPos.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 OddOrPos.java.
    3. Take screenshots of your work showing the bad and good test results.

**Submission:**

**For each of the 4 cases above (a, b, c, d), provide the following (taken from step 4 of each case):**

* **A description of the fault in code**
* **The good test that reveals the error**
* **Screenshots of bad and good test results**

**Include all your answers in ONE Word document and use the link on the course canvas to upload it**