Faculty of Engineering, Architecture and Science
Department of Electrical and Computer Engineering
Laboratory Report Cover Page

Course Number	COE891
Course Title	Software Testing & QA
Semester/Year	Winter 2023
Instructor	Reza Semavi
TA Name	Hamed Karimi

Lab/Tutorial Report No.	1

Section No.	012
Group No.	N/A
Submission Date	Jan 23th, 2023
Due Date	Jan 23th, 2023

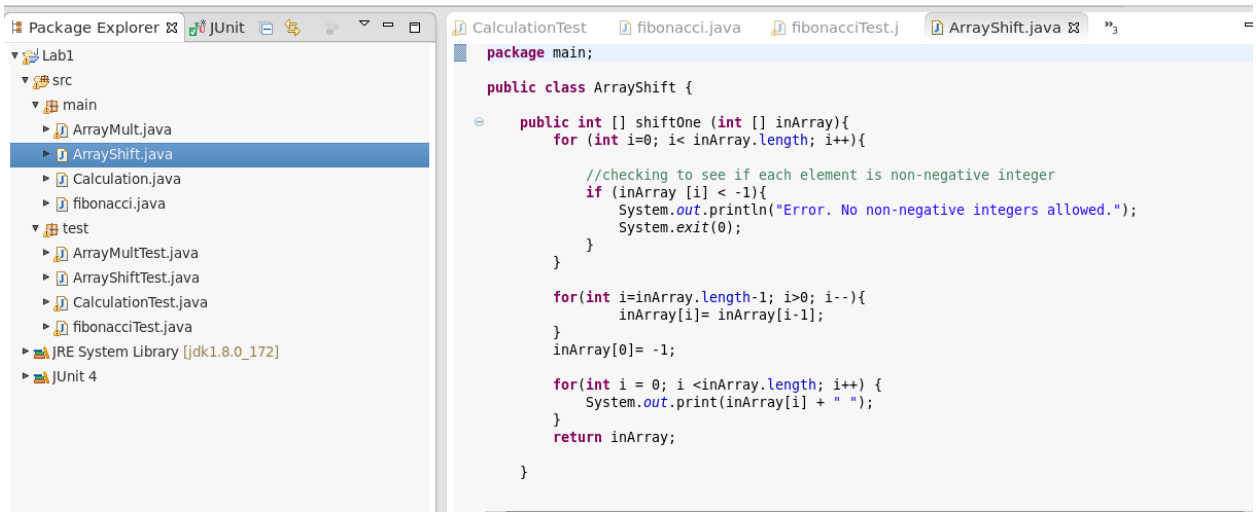
Student Name	Student ID	Signature
Hamza Iqbal	500973673	H.I

Lab Description

The purpose of this lab is to introduce ourselves with unit testing concepts within the JUnit framework of Java. Using the Test-Driven Development process, we will write automated test cases for various programs in order to ensure the units of code are working correctly. Initially, tests will be added for program code and run, in order to see if any tests fail. The code will then be updated if needed in order to pass the new tests. Next, the test will be run again, and the code will be refactored once again if any tests fail and repeat.

Lab Questions

Q1.




```java
package test;

import main.ArrayShift;

public class ArrayShiftTest extends TestCase{

    private ArrayShift as;
    private int[] inArray = {4, 6, 0, 3, 4, 5, 4, 4, 6, 2 };

    public final void setUp() {
        as = new ArrayShift();
    }

    @Test
    public final void testShiftOne() {

        int[] outArray = as.shiftOne(inArray);
        int[] expArray = {-1,4,6,0,3,4,5,4,4,6};

        assertArrayEquals(expArray, outArray);
    }
```

```java
    @Test
    public final void testShiftElements() {
        int[] expArray = {-1,4,6,0,3,4,5,4,4,6};
        int[] outArray = as.shiftOne(inArray);
        if (expArray[0] != -1  &&  outArray[0] != -1) {

            fail("Index values of output array do not match with original array");
        }
    }

    @Test
    public final void testArrayLength() {
        int[] outArray = as.shiftOne(inArray);
        if (outArray.length!= inArray.length) {

            fail("Length of output array do not match with original array");
        }
    }
}
```

Q2.

```java
package main;

public class ArrayMult {

    public int[] mult(int[] array1, int[] array2) {

            int[] short_array = array1;
            int[]  long_array = array2;

            if (array2.length < array1.length) {
                long_array  = array1;
                 short_array  = array2;
             }

             int[] outArray = new int[long_array.length];

             for (int i = 0; i < long_array.length; i++) {
                 if (i < short_array.length) {
                     outArray[i] = array1[i] * array2[i];
                 }
                 else {
                     outArray[i] = long_array [i];
                 }
             }
```

```java
             for(int i = 0; i <outArray.length; i++) {
                 System.out.print(outArray[i] + " ");
             }
             System.out.println("");
             return outArray;

    }
```

```java
package test;

import org.junit.Test;

import main.ArrayMult;
import static org.junit.Assert.*;
import junit.framework.TestCase;

import org.junit.Assert;

import static org.junit.Assert.assertArrayEquals;

public class ArrayMultTest extends TestCase{

    private ArrayMult as;
     int[] first_array = {1, 3, 5, 4};
     int[] second_array = {1, 4, 5, 2};

    public final void setUp() {
        as = new ArrayMult();
    }


    @Test
    public final void testMult() {

        int[] multArray1 = as.mult(first_array, second_array);
```

```java
    @Test
    public final void testMult() {

        int[] multArray1 = as.mult(first_array, second_array);
        int[] multArray2 = as.mult(second_array, first_array);
        int[] expArray = {1, 12,25,8};

        System.out.println("Test for multiplication:");
       assertArrayEquals(expArray, multArray1);
       assertArrayEquals(expArray, multArray2);
    }


    @Test
    public final void testLength() {
        int[] first_array = {1, 3, 5, 4, 6 ,8};
        int[] second_array = {1, 4, 5, 2};

        int[] multArray1 = as.mult(first_array, second_array);

        int[] multArray2 = as.mult(second_array, first_array);

        System.out.println("Test for Array Lengths:");
        assertEquals(first_array.length, multArray1.length);
        assertEquals(first_array.length, multArray2.length);

        assertEquals(first_array[first_array.length-1], multArray1[multArray1.length-1]);
```

```java
        assertEquals(first_array.length, multArray1.length);
        assertEquals(first_array.length, multArray2.length);

        assertEquals(first_array[first_array.length-1], multArray1[multArray1.length-1]);


    }

    @Test
    public final void testValues () {

        int[] multArray1 = as.mult(first_array, second_array);
        int[] multArray2 = as.mult(second_array, first_array);

        for (int i = 0;i < first_array.length; i++) {
            assertEquals(first_array[i] * second_array[i], multArray1[i]);
        }

        for (int i = 0; i < first_array.length; i++) {
            assertEquals(first_array[i] * second_array[i], multArray2[i]);
        }

    }

}
```

Q3.

The reason the findmax () method didnt work initially is because the local variable max in findmax () method was initialized to 0. Any values less than are therefore not accounted for in the loop. Setting the variable max to any value within the array allows it to be initialized without any sort of restrictions and allows for all ranges of values to be assessed.

```java
package main;

public class Calculation {

    public static int findMax(int arr[]){
        int max=arr [0];

        for(int i = 1;i<arr.length;i++){
            if(max<arr[i]){
                max=arr[i];
            }
         }
        System.out.println ("Max is: " + max);
        return max;
     }

    public static int cube(int n){
        System.out.println ("The cubed of " + n + " is: " + n*n*n);
        return n*n*n;
    }
}
```

```java
package test;

import static org.junit.Assert.*;

import org.junit.AfterClass;
import org.junit.After;
import org.junit.BeforeClass;
import org.junit.Before;
import org.junit.Test;

import main.ArrayMult;
import main.Calculation;
import static org.junit.Assert.*;
import junit.framework.TestCase;

import java.lang.Math;

import org.junit.Assert;

import static org.junit.Assert.assertArrayEquals;

public class CalculationTest extends TestCase {
```

```java
@BeforeClass
public static void BeforeClass() {
    System.out.println ("Before Class Running");
}

@Before
public static void BeforeTest() {
    System.out.println ("Before Test");
}


@Test
public void testMax() {
    System.out.println ("Testing FindMax Method:");
    int[] testArray1 = {1, 12,25,8};
    int maxArray1 = Calculation.findMax(testArray1);
    int expMax1 = 25;
    assertEquals(expMax1, maxArray1);
}

@Test
public void testMax2() {
    System.out.println ("Testing FindMax2 Method:");
    int[] testArray2 = {7, 0,7,3};
```

```java
        System.out.println ("Testing FindMax2 Method:");
        int[] testArray2 = {7, 0,7,3};
        int maxArray2 = Calculation.findMax(testArray2);
        int expMax2 = 7;
        assertEquals(expMax2, maxArray2);
        if (expMax2 != maxArray2) {
            fail("expected max should be the same as output max");
        }
    }

    @Test
    public void testMaxNeg() {
        System.out.println ("Testing FindMaxNeg Method");
        int[] testNegArray = {-12, -3,-4, -2};
        int maxArray = Calculation.findMax(testNegArray);
        int expMax1 = -2;
        assertEquals(expMax1, maxArray);
    }

    @Test
    public void testCube() {
        System.out.println ("Before Testing Cube Method");
        int testCube = Calculation.cube(3);
        int expVal = 3*3*3;
        assertEquals(expVal, testCube);
    }
    @Test
```

```java
    public void testCube() {
        System.out.println ("Before Testing Cube Method");
        int testCube = Calculation.cube(3);
        int expVal = 3*3*3;
        assertEquals(expVal, testCube);
    }
        @Test
    public void testCube2() {
        System.out.println ("Testing testCube2 Method:");
        int testCube = Calculation.cube(2);
        int cubeValue =2;
        if (cubeValue != Math.cbrt(testCube)) {
            fail("incorrect cube value.");

        }
    }

    @AfterClass
    public static void AfterClass() {
        System.out.println ("After Class Running");
    }

    @After
    public static void AfterTest() {
        System.out.println ("After Test");
    }
}
```

Q4.

```java
package main;

public class fibonacci {

    public static int fibonacci_iterative(int n)
    {
        if (n <= 1){
            return 1;
        }
        return fibonacci_iterative(n - 1) + fibonacci_iterative(n - 2);
    }


    public static void main(String args[])
    {
        int n = 9;
        System.out.println(fibonacci_iterative(n));
    }

}
```

```java
package test;

import static org.junit.Assert.*;


public class fibonacciTest extends TestCase {

    @Test
    public void test1() {
        int index = 9;
        int accVal = fibonacci.fibonacci_iterative(index);
        System.out.println(fibonacci.fibonacci_iterative(index));
        int expVal = 55;
        assertEquals (expVal,accVal);
    }

    @Test
    public void test2() {
        int index = -1;
        int accVal = fibonacci.fibonacci_iterative(index);
        System.out.println(fibonacci.fibonacci_iterative(index));
        int expVal = 1;
        assertEquals (expVal,accVal);
    }
```

```java
    @Test
    public void test3() {
        int index = 6;
        int expVal = 13;

        assertEquals(expVal, fibonacci.fibonacci_iterative(index));
    }

    //Test case to compare the first two values of fib seqeunce returned by the function
    @Test
    public void test4() {

        int firstIndex = fibonacci.fibonacci_iterative(0);
        int secondIndex = fibonacci.fibonacci_iterative(1);
         if (firstIndex != secondIndex) {
                fail("First two values of fib seq should match (=1)");
            }
    }
```