SOFTWARE QUALITY & METRICS QUIZ 2

- ASHISH KUMAR
- 2K18/SE/041

Tasks:

Ques 1. Write two Junit test cases for the following findMax() method:

a. Test case that passes.

b. Test case that fails.

c. Modify the source code to fix the defect(s) that made the test fails.

Source Code of input java file:

Calculation.java

```
package com.simplilearn.mavenproject;

public class Calculation {

        public static int findMax(int arr[]) {

                int max = 0;

                if (arr.length == 0)

                        return null;

                for (int i = 0; i < arr.length; i++) {

                        if (max < arr[i])

                                max = arr[i];

                }

                return max;

        }

        }
```

THEORY: JUnit is a popular unit-testing framework in the Java ecosystem. JUnit is a Java library to help you perform unit testing. Unit testing is the process of examining a small "unit" of software (usually a single class) to verify that it meets its expectations or specification. A unit test generally consists of various testing methods that each interacts with the class under test in some specific way to make sure it works as expected.

My Testing Code:

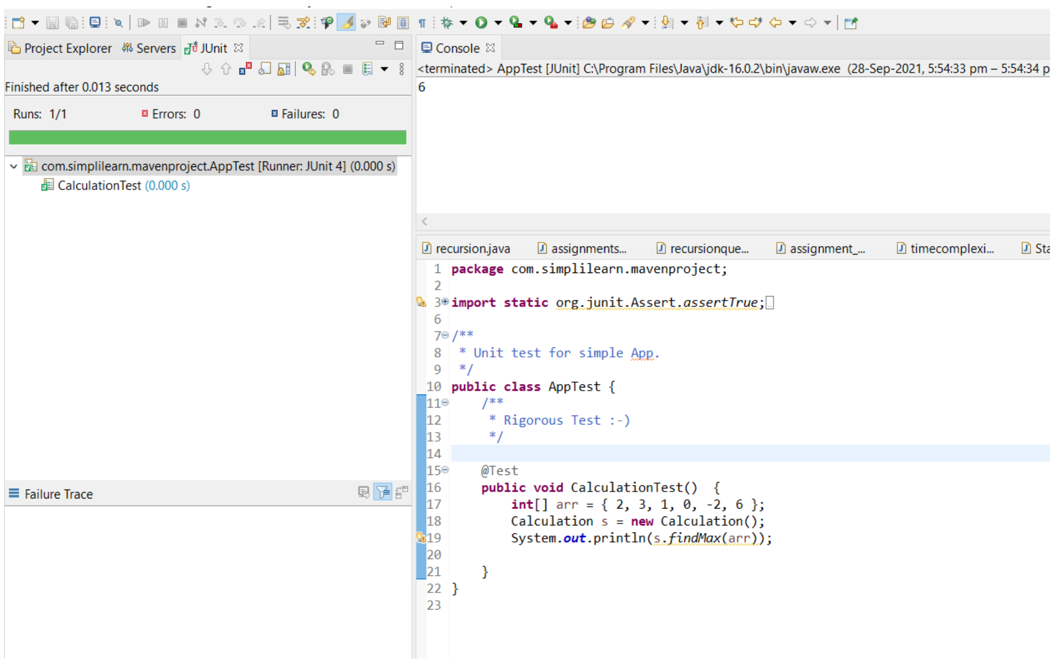
I have written testing code in a java file named as "**AppTest.java**" and below is the code of it:

```
package com.simplilearn.mavenproject;

import static org.junit.Assert.assertTrue;

import org.junit.Test;


public class AppTest {

@Test

        public void CalculationTest()  {

                int[] arr = { 2, 3, 1, 0, -2, 6 };           //input array contains +ve & -ve numbers

                //int[] arr = { };                            //empty array

                Calculation s = new Calculation();    //creating an object of class Calculation

                System.out.println(s.findMax(arr));     //calling findMax() function and displaying
                                                        //the result in console.

                assert (s.findMax(arr) = = 6);      // checking whether maxm number comes out to
                                                    //be 6 or not

                //assert (s.findMax(arr) = = 0);  //checking whether output comes out to be 0 or
                                                    //not  for empty array

        }

}
```

OUTPUT:

1. Test case that passes

There is **no error** because syntax and program is correct. Also maximum number in given array is also displayed on the console i.e., 6.

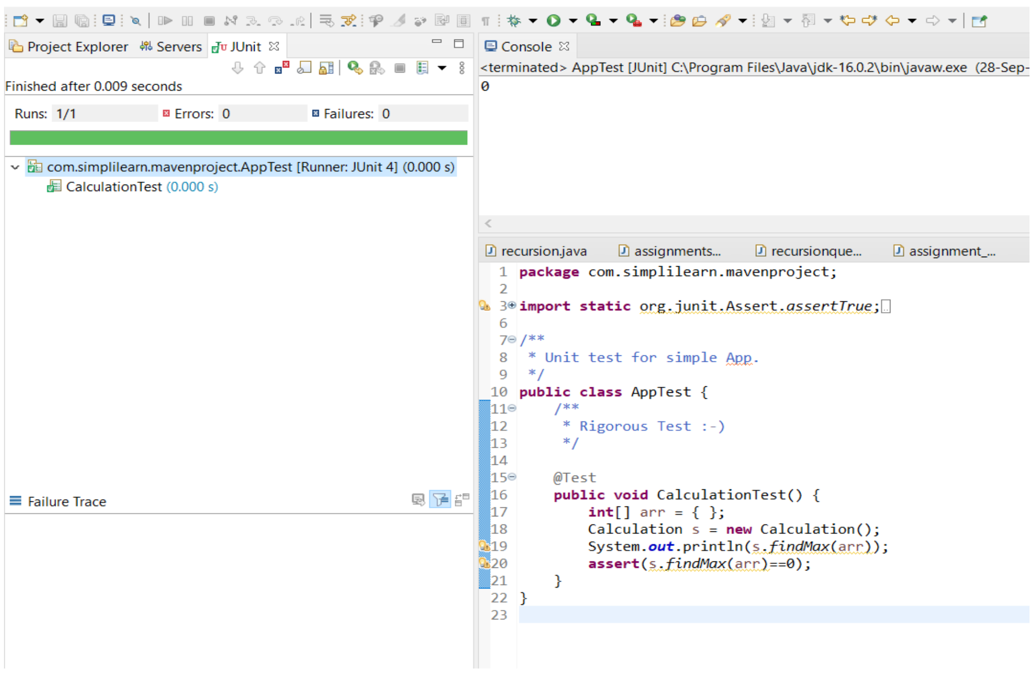


```
Finished after 0.013 seconds
Runs: 1/1    Errors: 0    Failures: 0
com.simplilearn.mavenproject.AppTest [Runner: JUnit 4] (0.000 s)
    CalculationTest (0.000 s)
```

```
<terminated> AppTest [JUnit] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (28-Sep-2021, 5:54:33 pm – 5:54:34 p
6
```

```
package com.simplilearn.mavenproject;

import static org.junit.Assert.assertTrue;

/**
 * Unit test for simple App.
 */
public class AppTest {
    /**
     * Rigorous Test :-)
     */

    @Test
    public void CalculationTest()  {
        int[] arr = { 2, 3, 1, 0, -2, 6 };
        Calculation s = new Calculation();
        System.out.println(s.findMax(arr));

    }
}
```

When array is empty, there is no error. findMax() is also working for this testcase. It is returning 0 as an output (after modifying the code).



```
Finished after 0.009 seconds
Runs: 1/1    Errors: 0    Failures: 0
com.simplilearn.mavenproject.AppTest [Runner: JUnit 4] (0.000 s)
    CalculationTest (0.000 s)
```

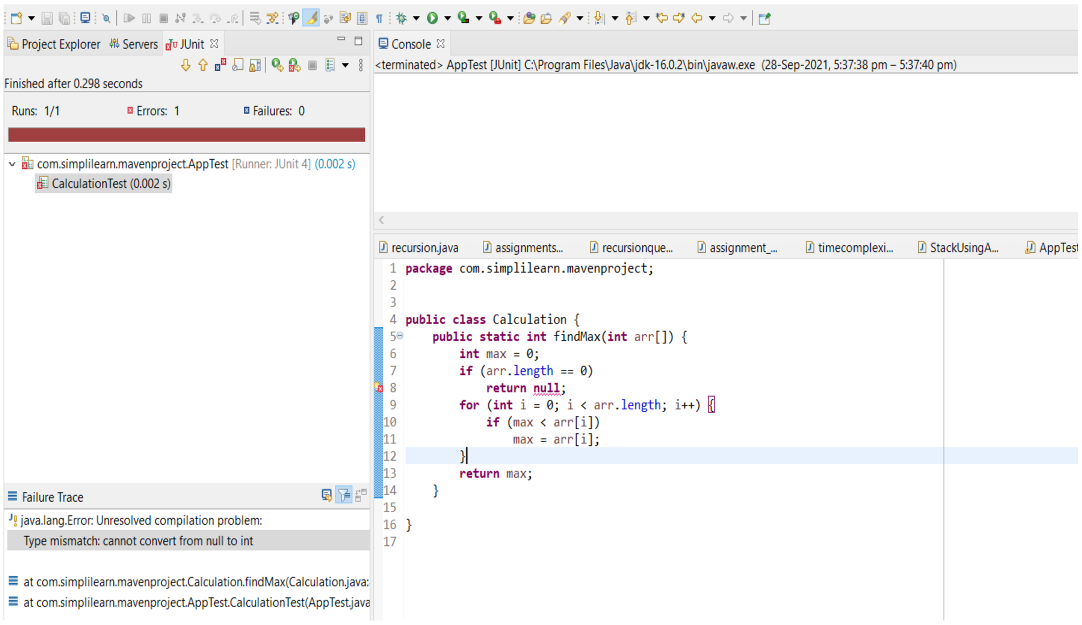
```
<terminated> AppTest [JUnit] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (28-Sep-
0
```

```
package com.simplilearn.mavenproject;

import static org.junit.Assert.assertTrue;

/**
 * Unit test for simple App.
 */
public class AppTest {
    /**
     * Rigorous Test :-)
     */

    @Test
    public void CalculationTest() {
        int[] arr = { };
        Calculation s = new Calculation();
        System.out.println(s.findMax(arr));
        assert(s.findMax(arr)==0);
    }
}
```

2. Test case that fails

There is **an error in line no. 8**. As the return type of the findMax() is of integer type, so it can't return null as an output. That's why we are getting an error "Type mismatch: cannot convert from null to int".

Now to resolve this issue, just replace null with zero in line no. 8.



Project Explorer Servers JUnit

Finished after 0.298 seconds

Runs: 1/1 Errors: 1 Failures: 0

com.simplilearn.mavenproject.AppTest [Runner: JUnit 4] (0.002 s)
CalculationTest (0.002 s)

Failure Trace

```
java.lang.Error: Unresolved compilation problem:
    Type mismatch: cannot convert from null to int

at com.simplilearn.mavenproject.Calculation.findMax(Calculation.java:
at com.simplilearn.mavenproject.AppTest.CalculationTest(AppTest.java
```

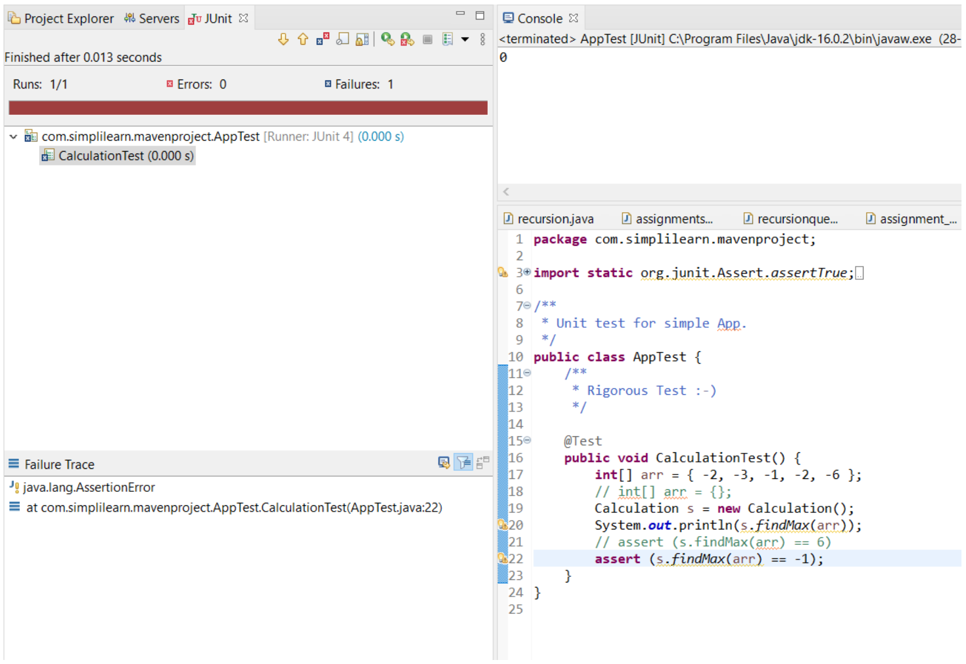
Console

```
<terminated> AppTest [JUnit] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (28-Sep-2021, 5:37:38 pm – 5:37:40 pm)
```

recursion.java assignments... recursionque... assignment_... timecomplexi... StackUsingA... AppTest

```java
1  package com.simplilearn.mavenproject;
2
3
4  public class Calculation {
5      public static int findMax(int arr[]) {
6          int max = 0;
7          if (arr.length == 0)
8              return null;
9          for (int i = 0; i < arr.length; i++) {
10             if (max < arr[i])
11                 max = arr[i];
12         }
13         return max;
14     }
15
16 }
17
```

When I provide only negative numbers in input array. It is giving wrong output i.e, 0 but the expected output will be -1. So, **it is a failure**.



Project Explorer Servers JUnit

Finished after 0.013 seconds

Runs: 1/1 Errors: 0 Failures: 1

com.simplilearn.mavenproject.AppTest [Runner: JUnit 4] (0.000 s)
CalculationTest (0.000 s)

Failure Trace

```
java.lang.AssertionError
at com.simplilearn.mavenproject.AppTest.CalculationTest(AppTest.java:22)
```

Console

```
<terminated> AppTest [JUnit] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (28-
0
```

recursion.java assignments... recursionque... assignment_...

```java
1  package com.simplilearn.mavenproject;
2
3⊕import static org.junit.Assert.assertTrue;
6
7⊖/**
8   * Unit test for simple App.
9   */
10 public class AppTest {
11⊖    /**
12      * Rigorous Test :-)
13      */
14
15⊖    @Test
16     public void CalculationTest() {
17         int[] arr = { -2, -3, -1, -2, -6 };
18         // int[] arr = {};
19         Calculation s = new Calculation();
20         System.out.println(s.findMax(arr));
21         // assert (s.findMax(arr) == 6)
22         assert (s.findMax(arr) == -1);
23     }
24 }
25
```

To resolve this issue, do this change in line no. 6.

```
int max = arr[0];
```

3. Modify the source code to fix the defect(s) that made the test fails

I have already covered it in 2nd Test case.

So, the **modify** source code is:

It is also handling the case of duplicate numbers existed in the array.

```
public class Calculation {
    public static int findMax(int arr[]) {
            int max = arr[0];
            if (arr.length == 0)
                    return 0;
            for (int i = 0; i < arr.length; i++) {
                    if (max < arr[i])
                            max = arr[i];
            }
            return max;
    }
    }
```

Ques 2: Consider the following requirements and code:

The program shall take as input an array of three integer numbers.

The program shall output the greatest number among the elements of the array.

The program shall order the elements of the array in decreasing order.

```
1.   int[] order(int v[]) {
2.        int tmp;
3.        if (v[0]<v[1]) {
4.        tmp = v[0];
5.        v[1] = v[1];
6.        v[1] = tmp;
7.        }
8.        if (v[1]<v[2]) {
9.         tmp = v[0];
10.       v[1] = v[2];
11.       v[2] = tmp;
12.       }
13.       return v;
14.  }
```

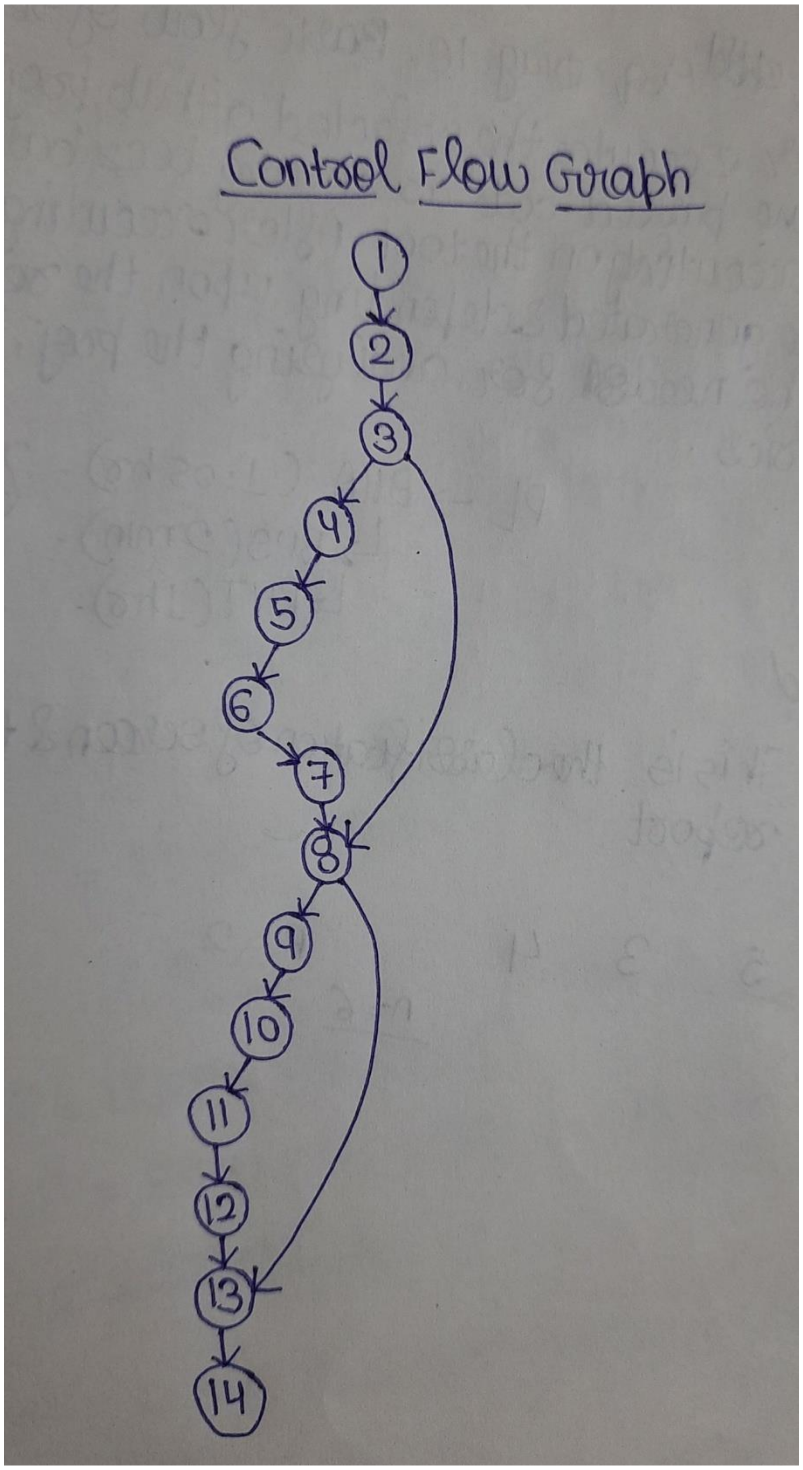
a) Create the control flow graph.

b) Write the minimum number of test cases for each of the coverage criteria:

- 100% Node (statement) coverage,
- 100% Edge (branch) coverage,
- 100% Path coverage

Ans:

a)



b) i) For 100% node coverage, following array can be taken as test case :

v[] = {10, 11, 12}

ii) For 100% edge coverage, following arrays can be taken as test case :

v[] = {10, 11, 12}

v[] = {12, 11, 10}

iii) For 100% Path coverage, following arrays can be taken as test case :

v[] = {10, 11, 12)}

v[] = {11, 12, 12}

v[] = {5, 5, 6}

v[] = {7, 7, 7}