

### Lab Exercise - JUnit

SE3010 – SEPQM

Semester 1

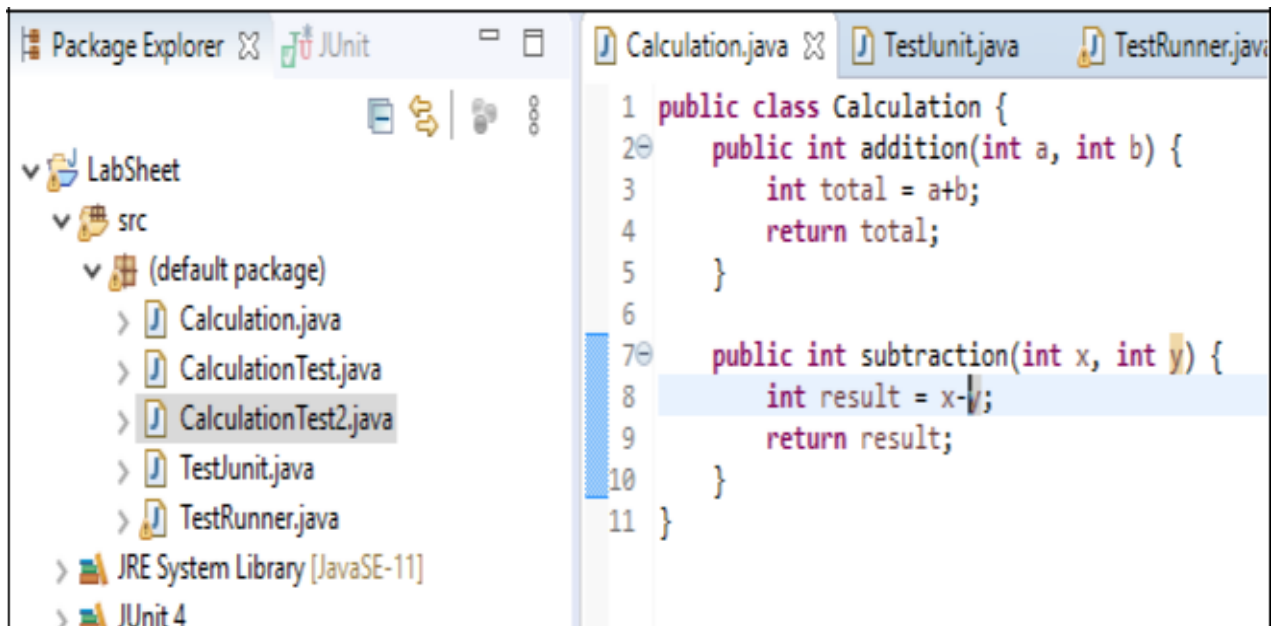
**JUnit** is an open source unit testing framework for JAVA. It is useful for java developers to write and run repeatable tests. Using JUnit, you can easily and incrementally build a test suite that will help you measure your progress, spot unintended side effects, and focus on your development efforts.

#### Key JUnit Notions

- Tested Class – The class that is being tested
- Tested Method – The method that is tested
- Test Case – The testing of a class's method against some specified conditions
- Test Case Class – A class performing the test cases
- Test Case Method – A Test Case Class's method implementing a test case
- Test Suite – A collection of test cases that can be tested in a single batch

#### Step 1:

There should be a class to test. Thus, first create a java project and create a class called "Calculation.java" to add two numbers.



The screenshot shows an IDE with a Package Explorer on the left and a code editor on the right. The Package Explorer shows a project named 'LabSheet' with a source folder 'src' containing a 'default package' with files: 'Calculation.java', 'CalculationTest.java', 'CalculationTest2.java', 'TestJUnit.java', and 'TestRunner.java'. The code editor shows the 'Calculation.java' file with the following code:

```
1 public class Calculation {
2     public int addition(int a, int b) {
3         int total = a+b;
4         return total;
5     }
6
7     public int subtraction(int x, int y) {
8         int result = x-y;
9         return result;
10    }
11 }
```

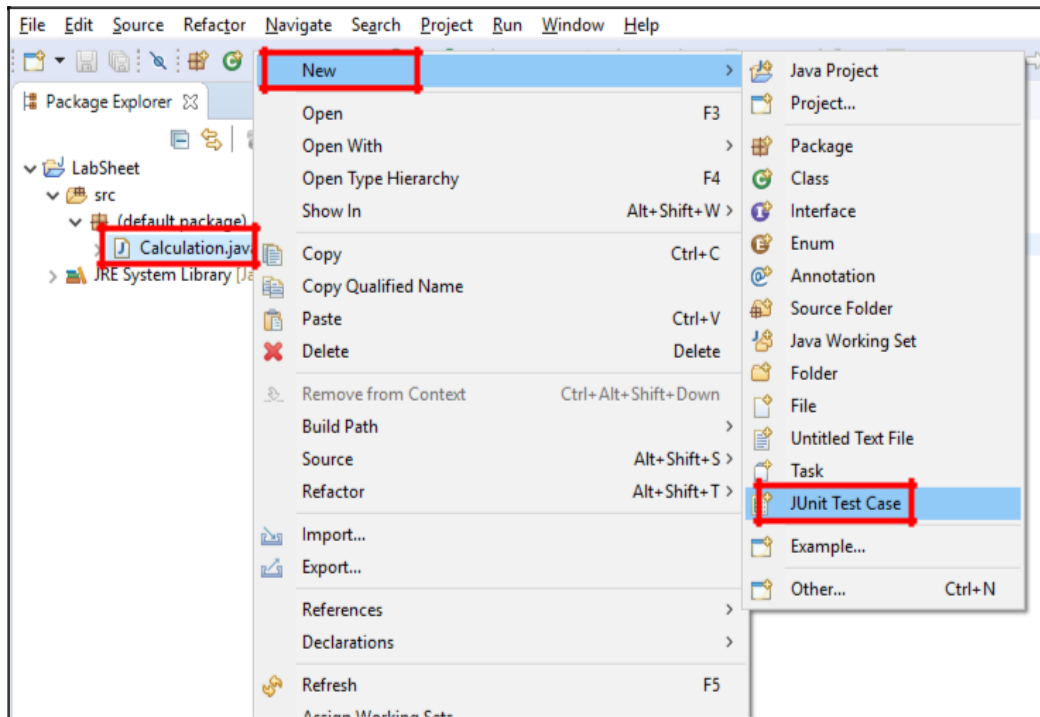
### Lab Exercise - JUnit

SE3010 – SEPQM

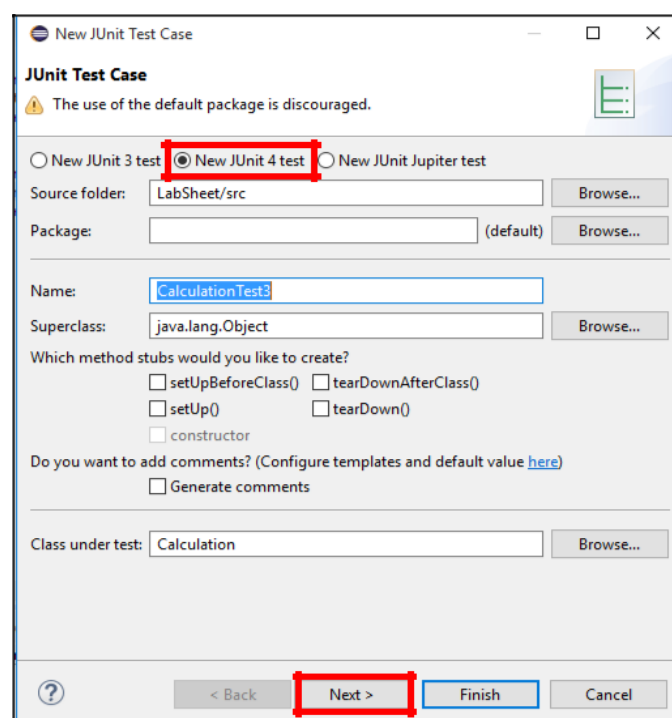
Semester 1

#### Step 2: Adding a test case to the project

1. Right click on the created class → Select “New” → Click “JUnit Test Case”



2. Then you would be navigated to the following window. Select the radio button “New JUnit 4 test” and then click on the “next” button.

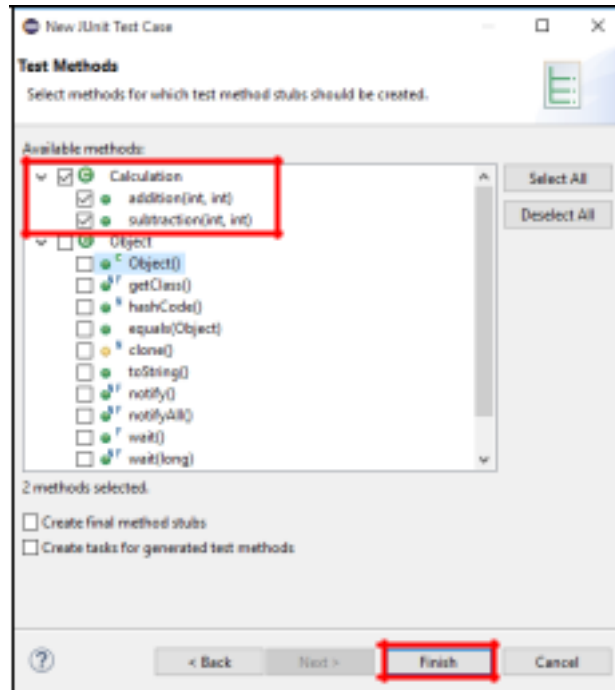


### Lab Exercise - JUnit

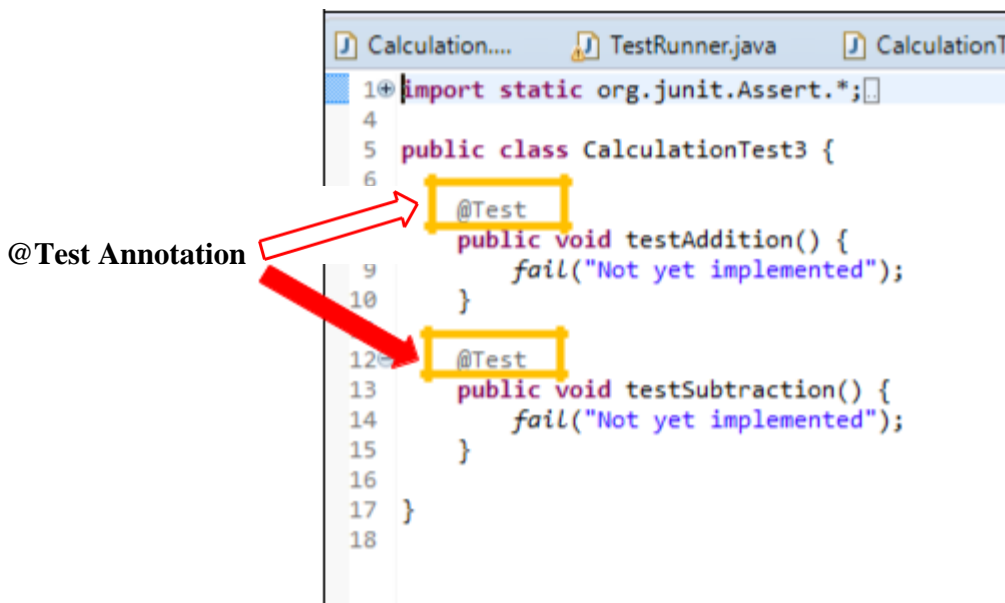
SE3010 – SEPQM

Semester 1

3. Now you would get following window. Select the units (methods) you want to test and click finish.



4. Now you would be able to see the “TestCase class”.



### Lab Exercise - JUnit

**SE3010 – SEPQM**

**Semester 1**

#### 5. Assertions

The **org.junit.Assert** class provides a group of assertion methods which are useful for writing tests. Some important methods of the Assert class are follows:

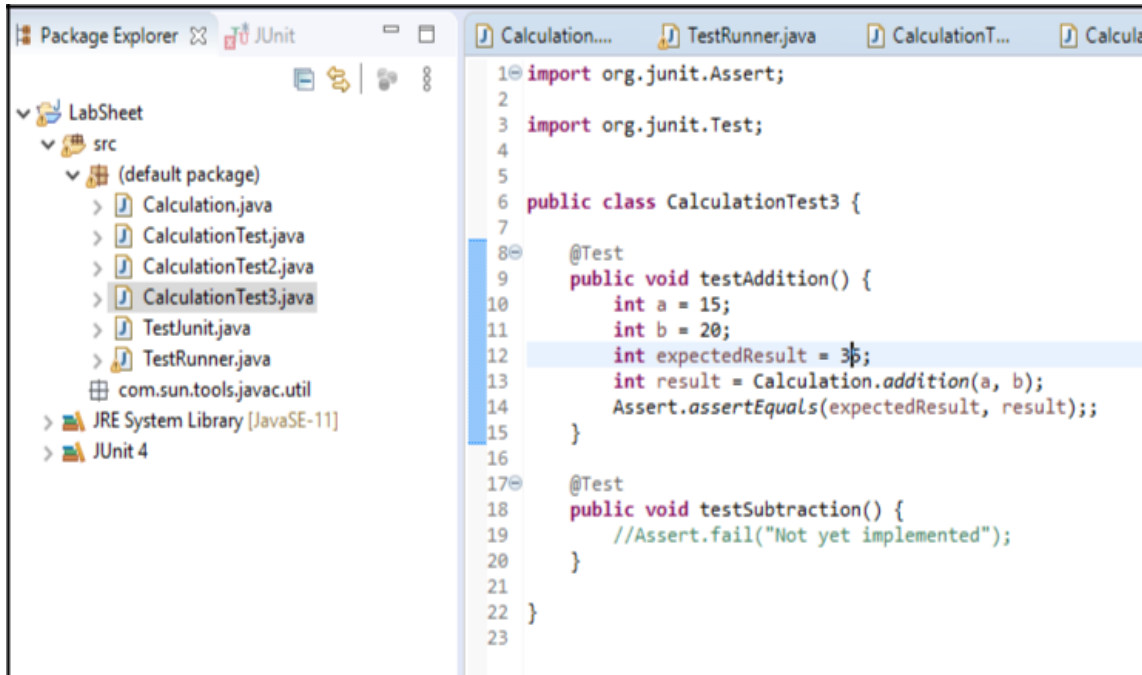
Method	Description
<code>assertEquals(boolean expected, boolean actual)</code>	Is used to test the equality of two primitives/objects.
<code>assertArrayEquals(expectedArray, resultArray)</code>	Is used to test whether two arrays are equal.
<code>assertTrue(boolean condition)</code>	Is used to test whether a condition is true.
<code>assertFalse(boolean condition)</code>	Is used to test whether a condition is false.
<code>assertNull(Object object)</code>	Is used to test whether an object is null.
<code>assertNotNull(Object object)</code>	Is used to test whether an object is not null.
<code>assertSame(object1, object2)</code>	Is used to test whether two object references are pointing to the same object
<code>assertNotSame(object1, object2)</code>	Is used to test whether two object references are not pointing to the same object

## Lab Exercise - JUnit

SE3010 – SEPQM

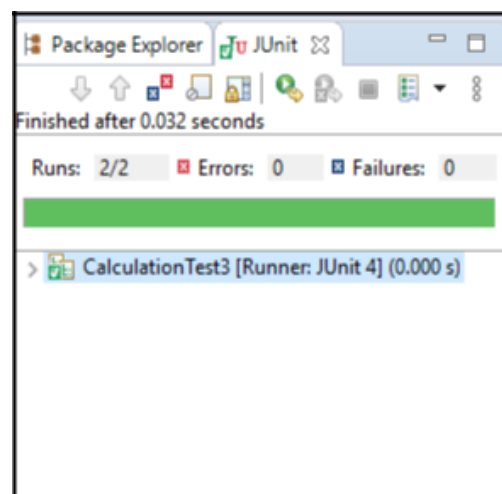
Semester 1

5.1. Include the appropriate assert method(s) and complete the code.



```
1 import org.junit.Assert;
2
3 import org.junit.Test;
4
5
6 public class CalculationTest3 {
7
8     @Test
9     public void testAddition() {
10         int a = 15;
11         int b = 20;
12         int expectedResult = 35;
13         int result = Calculation.addition(a, b);
14         Assert.assertEquals(expectedResult, result);
15     }
16
17     @Test
18     public void testSubtraction() {
19         //Assert.fail("Not yet implemented");
20     }
21
22 }
23
```

5.2. Next, right click on your test case class and run it as a JUnit test. Then, you would get a window such as the following:



### Lab Exercise - JUnit

**SE3010 – SEPQM**

**Semester 1**

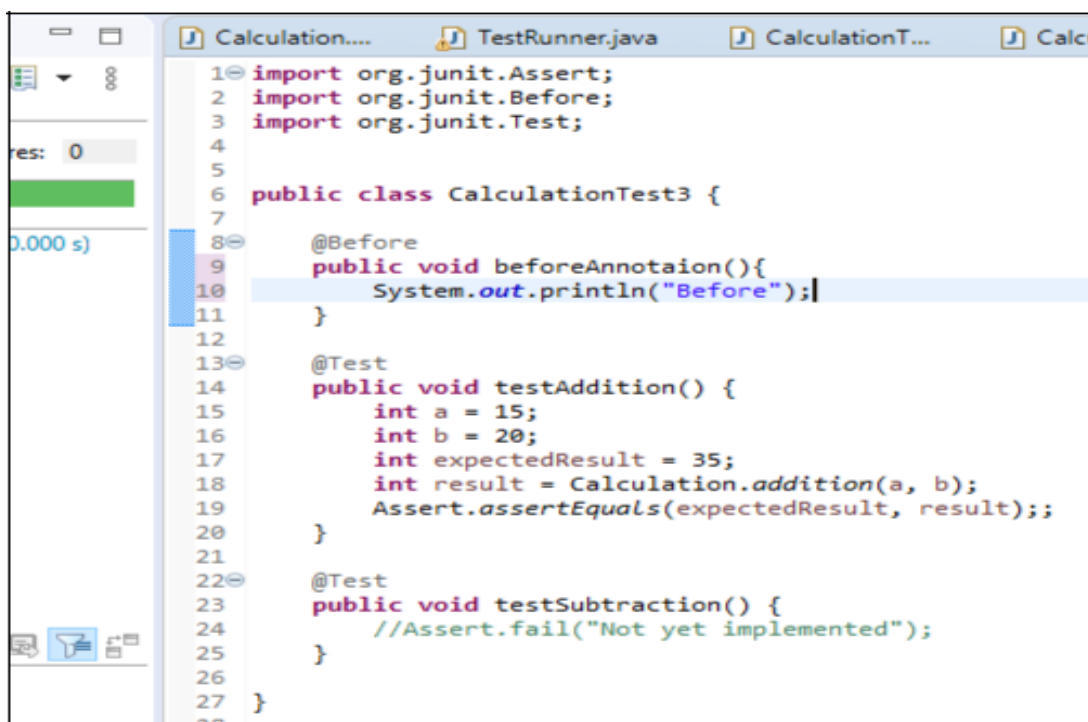
6. **Annotations:** Annotations were introduced in Junit4. They are like meta-tags which can be added to a java code for better readability. They provide the following information about test methods:

- The methods which are going to run before and after test methods
- The methods which run before and after all the methods
- The methods or classes that will be ignored during the execution

Following table provides a list of important and frequently used annotations:

Annotation	Description
@Test	This annotation indicates that public void method to which it is attached can be executed as a test case.
@Before	This annotation can be used to execute some statement(s) such as preconditions before each test case.
@BeforeClass	This annotation can be used to execute some statement(s) before all the test cases.
@After	This annotation can be used to execute some statement(s) after each test case.
@AfterClass	This annotation can be used to execute some statement(s) after all test cases.
@Ignores	This annotation can be used to ignore some statement(s) during test execution.
@Test(timeout=500)	This annotation can be used to specify a time limit for the execution of a test.

6.1. Add the @Before annotation to your code and run it.



```

1 import org.junit.Assert;
2 import org.junit.Before;
3 import org.junit.Test;
4
5
6 public class CalculationTest3 {
7
8     @Before
9     public void beforeAnnotation(){
10         System.out.println("Before");
11     }
12
13     @Test
14     public void testAddition() {
15         int a = 15;
16         int b = 20;
17         int expectedResult = 35;
18         int result = Calculation.addition(a, b);
19         Assert.assertEquals(expectedResult, result);
20     }
21
22     @Test
23     public void testSubtraction() {
24         //Assert.fail("Not yet implemented");
25     }
26
27 }
  
```

### Lab Exercise - JUnit

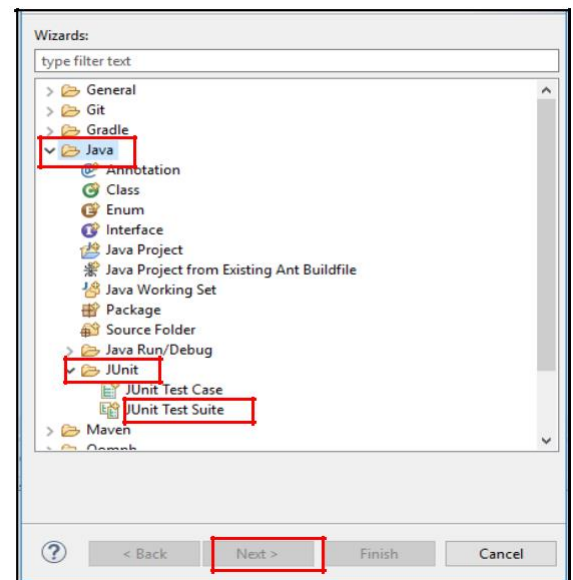
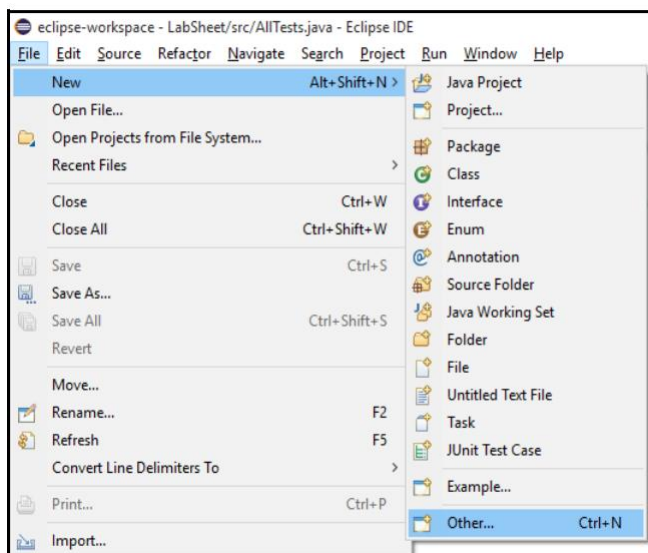
SE3010 – SEPQM

Semester 1

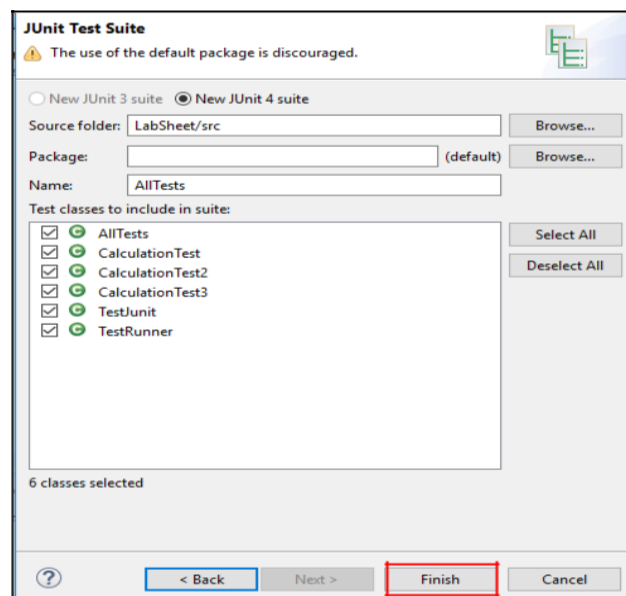
#### Step 3: Test Suite

We have performed tests on only one class, i.e. we have tested methods under the consideration that they belong to the same class. However, in large projects there are many classes with methods that should be tested. For testing multiple classes, Eclipse and JUnit introduced the concept of Test Suite. A Test Suite is a simple way running a collection of test cases as a single batch. To open a Test Suite, follow the following steps:

1. Select File → New → Other... → Java → JUnit → JUnit Test Suite → Next



2. Next, select which test classes that you wish to test together and click finish.

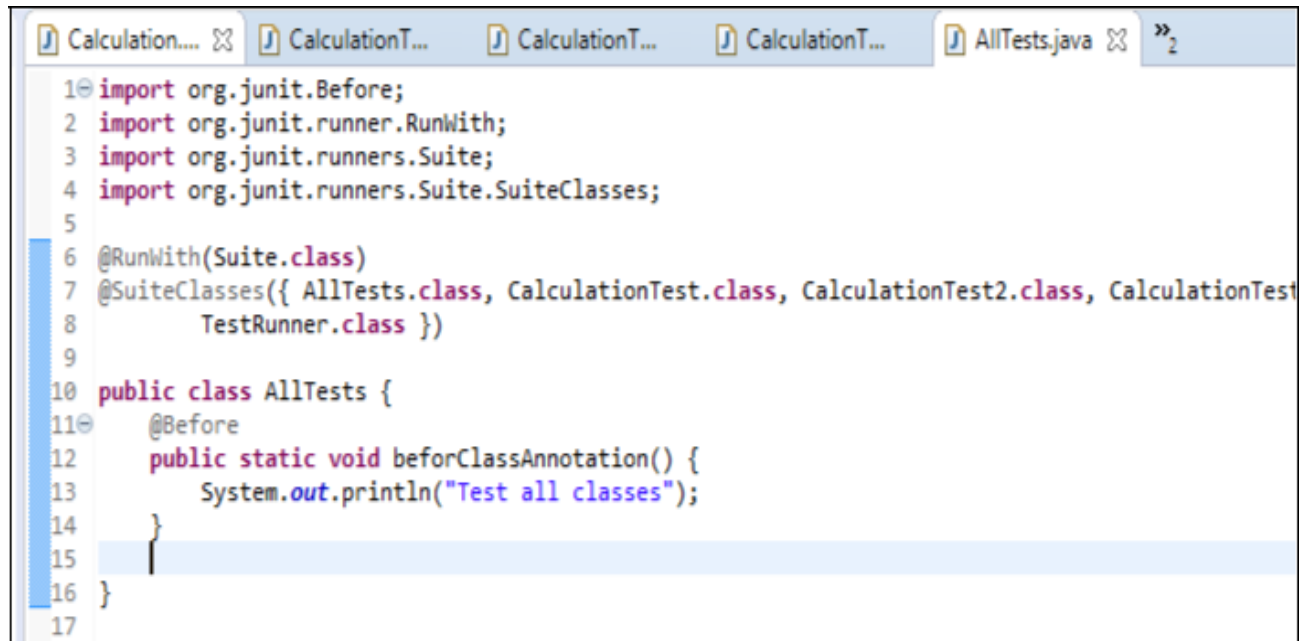


### Lab Exercise - JUnit

SE3010 – SEPQM

Semester 1

3. Then, you would get the code related to the test suite, which can be altered according to your needs.



```
1 import org.junit.Before;
2 import org.junit.runner.RunWith;
3 import org.junit.runners.Suite;
4 import org.junit.runners.Suite.SuiteClasses;
5
6 @RunWith(Suite.class)
7 @SuiteClasses({ AllTests.class, CalculationTest.class, CalculationTest2.class, CalculationTest
8     TestRunner.class })
9
10 public class AllTests {
11     @Before
12     public static void beforClassAnnotation() {
13         System.out.println("Test all classes");
14     }
15 }
16
17
```