**Week 2 Mandatory Hands-on**

**JUnit\_Basic Testing Exercises**

**Exercise 1: Setting Up Junit**

Scenario: You need to set up JUnit in your Java project to start writing unit tests.

* JUnit setup made.

**first.java**

**package** com.fse.junit.ex;

**public** **class** first {

**public** **int** add(**int** a, **int** b) {

**return** a + b;

}

**public** **int** subtract(**int** a, **int** b) {

**return** a - b;

}

}

**firstTest.java**

package com.fse.junit.ex;

import static org.junit.Assert.\*;

import org.junit.Test;

public class firstTest {

@Test

public void testAdd() {

first calc = new first();

assertEquals(5, calc.add(2, 3));

}

@Test

public void testSubtract() {

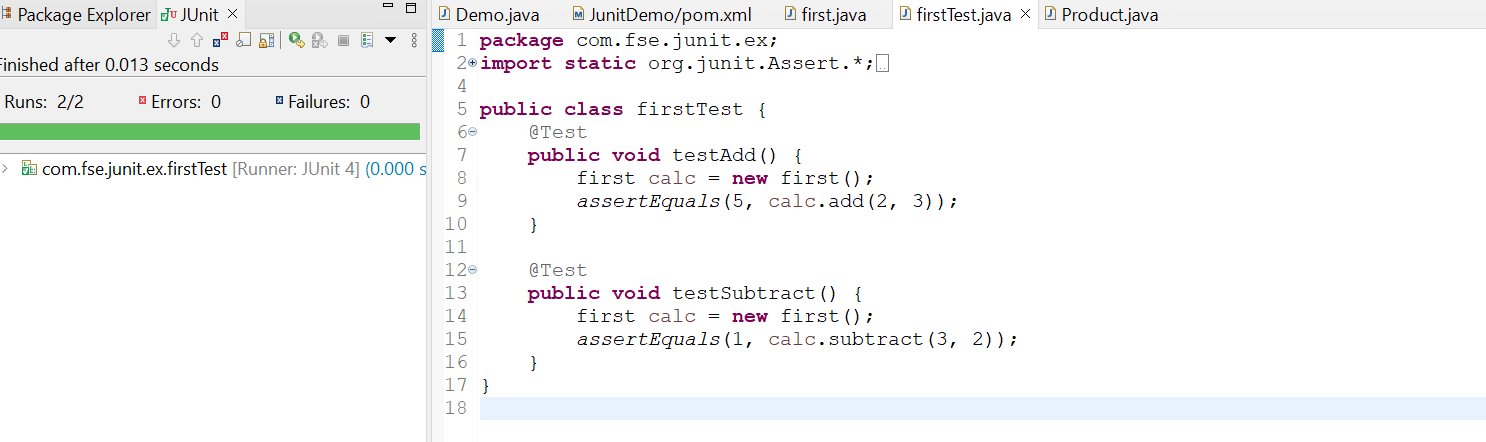
first calc = new first();

assertEquals(1, calc.subtract(3, 2));

}

}

**Output:**

****

**Description:**

* In testAdd() : 2+3=5 🡺 The test is passed.
* In testSubtract(): 3-2=1 🡺 The test is passed.

**Exercise 2: Writing Basic JUnit Tests**

Scenario: You need to write basic JUnit tests for a simple Java class.

**Base.java**

**package** com.fse.junit.ex;

**public** **class** Base {

**public** **static** **boolean** isEven(**int** number) {

**return** number % 2 == 0;

}

}

**BaseTest.java**

**package** com.fse.junit.ex;

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

**import** org.junit.Test;

**public** **class** BaseTest {

@Test

**public** **void** testIsEven1() {

System.***out***.println("Checking Test Case 1");

*assertFalse*(Base.*isEven*(6));

}

@Test

**public** **void** testIsEven2() {

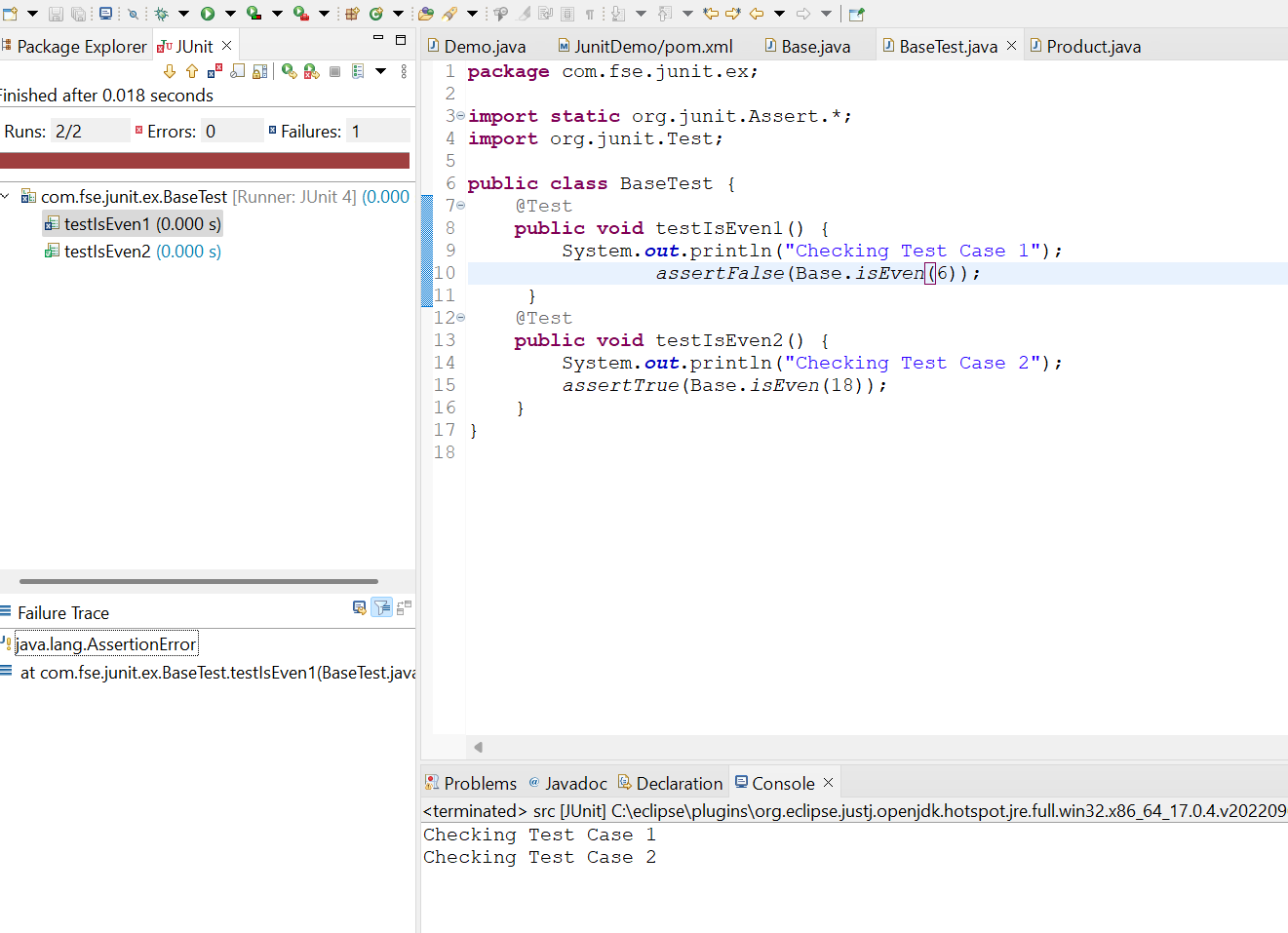
System.***out***.println("Checking Test Case 2");

*assertTrue*(Base.*isEven*(18));

}

}

**Output:**



**Description:**

* In testIsEven(): 6 = Even 🡺 True
* Hence test is failed
* In testIsEven(): 18 = Even 🡺 True
* Hence test is passed

**Exercise 3: Assertions in JUnit**

Scenario: You need to use different assertions in JUnit to validate your test results.

**Code:**

**package** com.fse.junit.ex;

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

**import** org.junit.Test;

**public** **class** AssertionsTest {

@Test

**public** **void** testAssertions() {

// Assert equals

System.***out***.println("Assertions Test running");

*assertEquals*(5, 2 + 3);

// Assert true

*assertTrue*(5 > 3);

// Assert false

*assertFalse*(5 < 3);

// Assert null

*assertNull*(**null**);

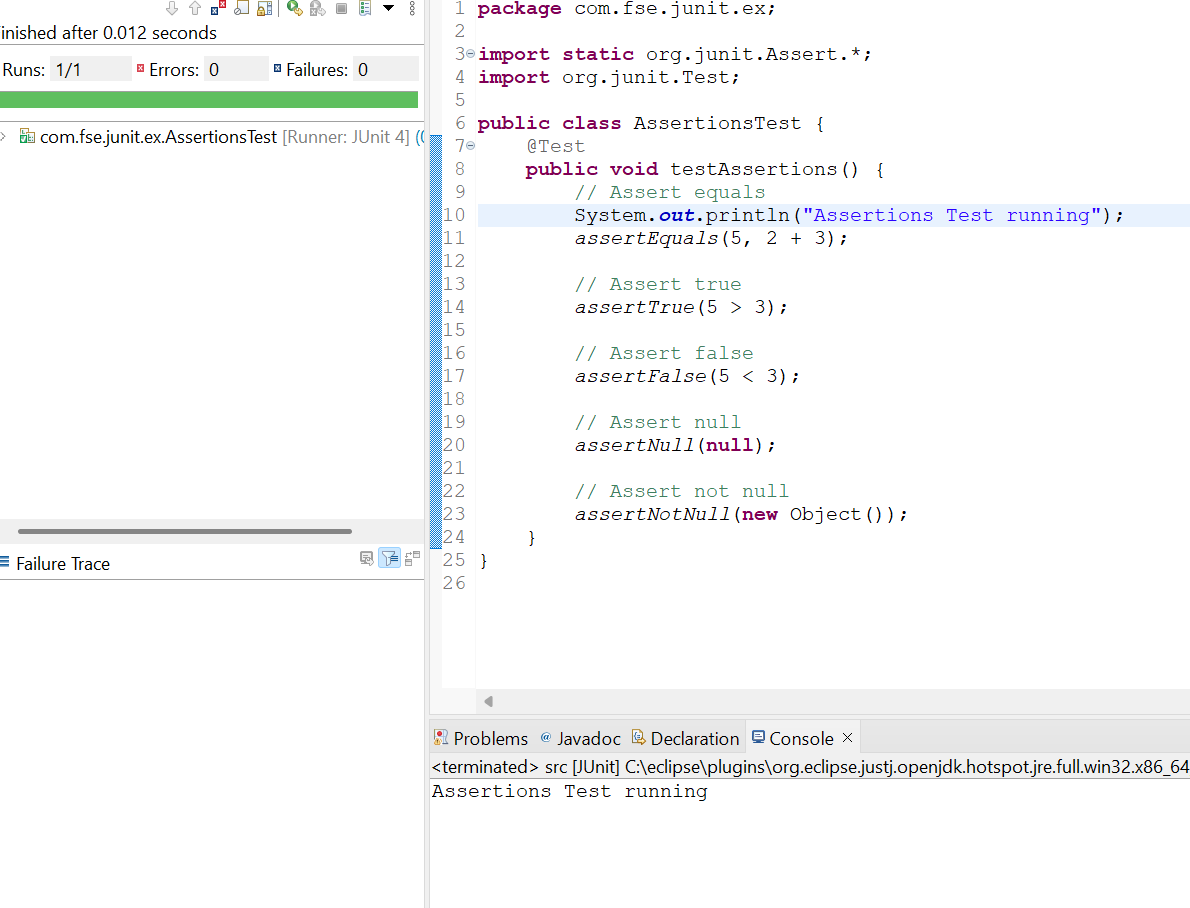
// Assert not null

*assertNotNull*(**new** Object());

}

}

**Output:**



**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

Scenario: You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

**Calculator.java**

**package** com.fse.junit.ex;

**public** **class** Calculator {

**public** **int** add(**int** a, **int** b) {

**return** a + b;

}

**public** **int** subtract(**int** a, **int** b) {

**return** a - b;

}

}

**CalculatorAAATest.java**

package com.fse.junit.ex;

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

public class CalculatorAAATest {

private Calculator calc;

// Setup method - runs before each @Test

@Before

public void setUp() {

calc = new Calculator();

System.out.println("Setup done");

}

// Teardown method - runs after each @Test

@After

public void tearDown() {

System.out.println("Teardown done");

}

@Test

public void testAdd() {

int result = calc.add(2, 3);

assertEquals(5, result);

System.out.println("Test running");

}

@Test

public void testSubtract() {

// Act

int result = calc.subtract(5, 2);

assertEquals(3, result);

System.out.println("Test running");

}

}

**Output:**

