**WEEK 2 HANDS ON EXERCISE – TDD using JUnit5 and Mockito – Testing Exercises**

**Exercise 1 : Setting Up JUnit**

**PROBLEM STATEMENT:**

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

**CODE :**

**src/main/java**

package com.example.JUnitDemo;

public class Calculator

{

public int add(int a, int b)

{

return a + b;

}

}

**src/test/java**

package com.example.JUnitDemo;

import static org.junit.Assert.\*;

import org.junit.Test;

public class CalculatorTest {

public void testAdd() { Calculator calc = new Calculator();

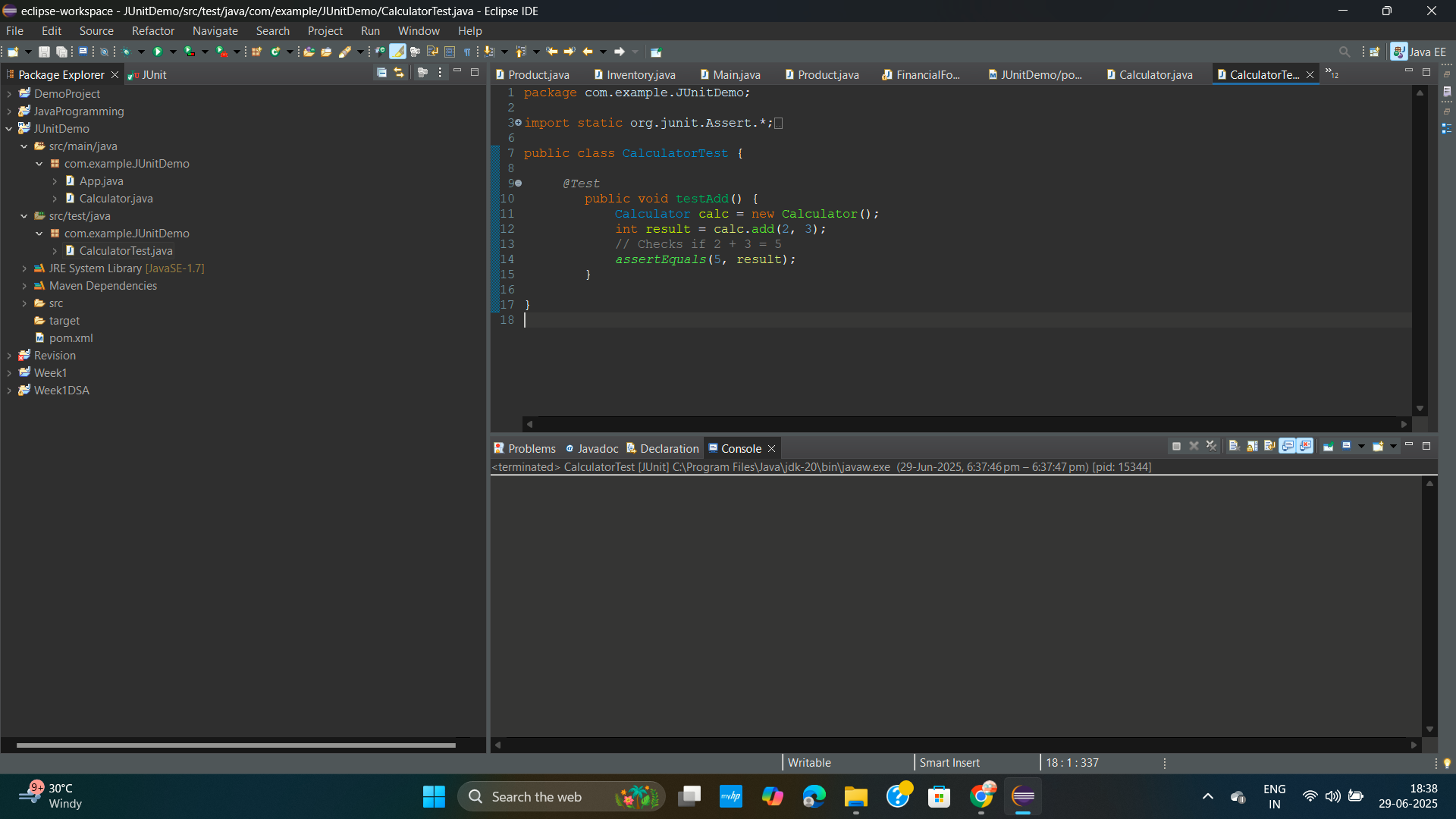
int result = calc.add(2, 3);

// Checks if 2 + 3 = 5

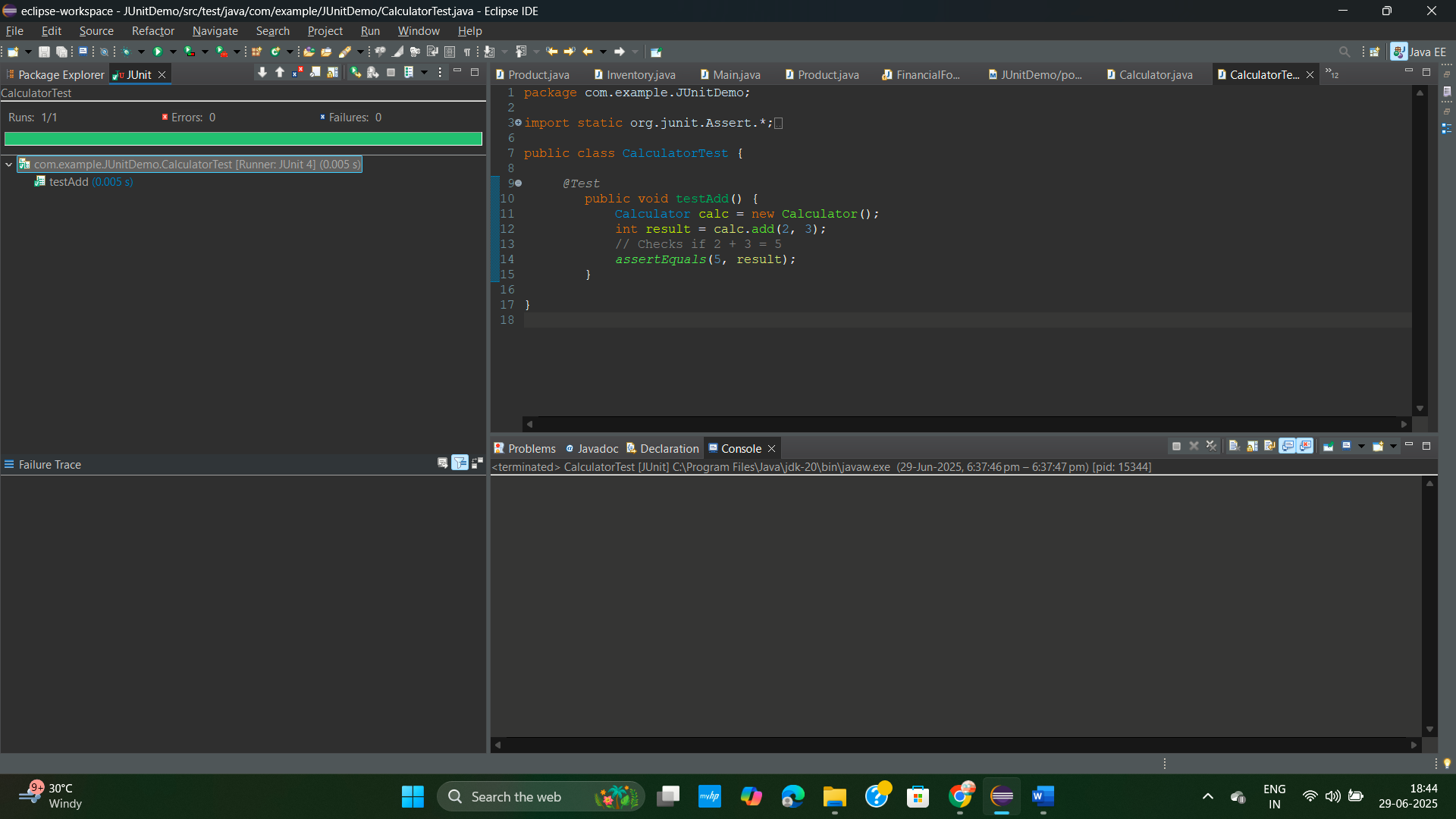
assertEquals(5, result);

}}

**OUTPUT :**



**// shows green bar**



**Exercise 3 : Assertions in JUnit**

**PROBLEM STATEMENT:**

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

**CODE :**

**src/test/java**

package com.example.JUnitDemo;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AssertionsTest {

@Test

public void testAssertions() {

// Assert that 2 + 3 = 5

assertEquals(5, 2 + 3);

// Assert true condition

assertTrue(5 > 3);

// Assert false condition

assertFalse(5 < 3);

// Assert null value

assertNull(null);

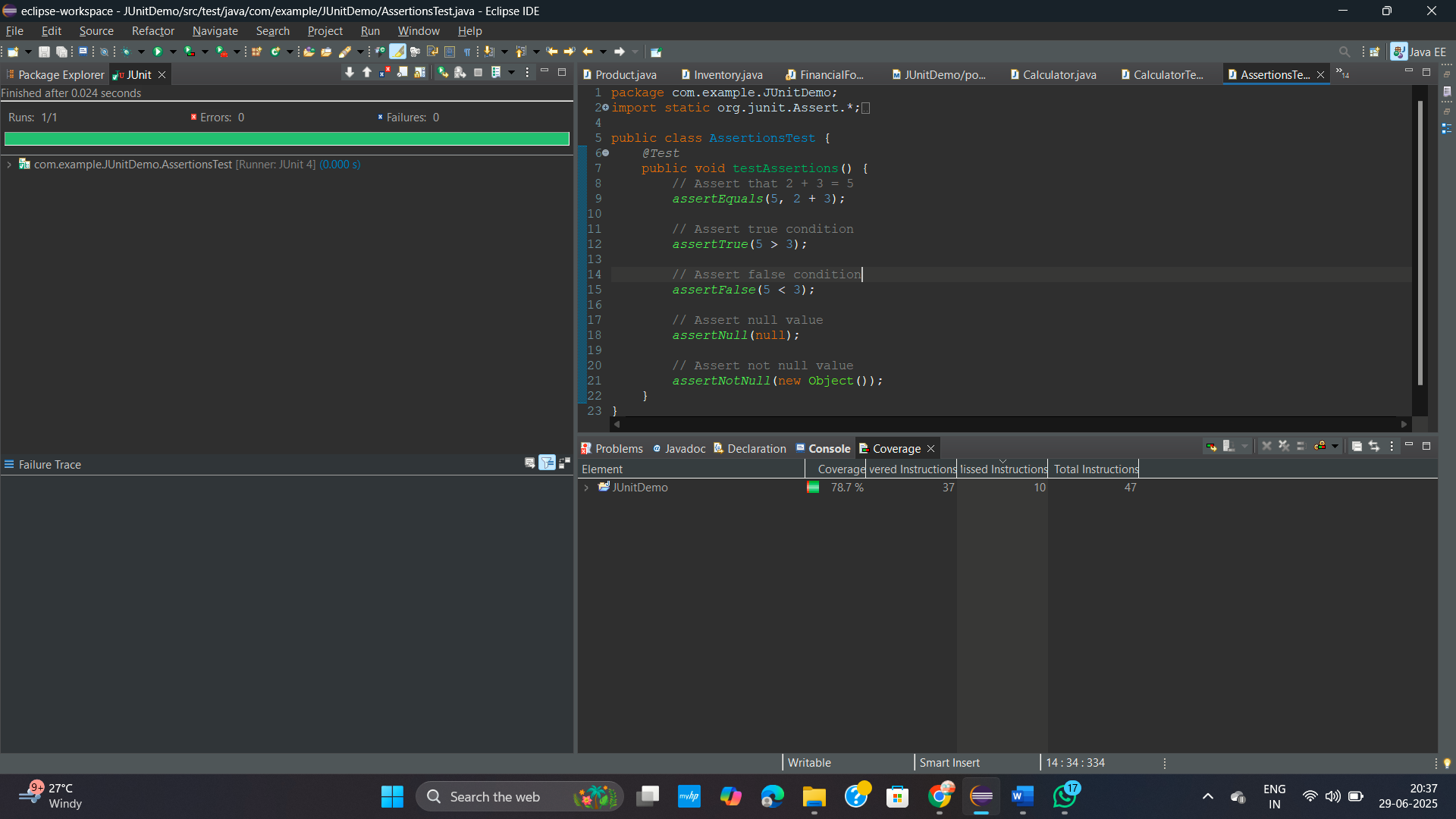
// Assert not null value

assertNotNull(new Object());

}

}

**OUTPUT :**



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

**PROBLEM STATEMENT:**

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

**CODE :**

package com.example.JUnitDemo;

import static org.junit.Assert.\*;

import static org.mockito.Mockito.*mock*;

import static org.mockito.Mockito.*when*;

import org.junit.Test;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.mockito.Mockito.*verify*;

public class CalculatorTest

{

private Calculator calculator;

*@Test*

public void testVerifyInteraction() {

// Step 1: Create mock

ExternalApi mockApi = *mock*(ExternalApi.class);

// Step 2: Inject mock into service

MyService service = new MyService(mockApi);

service.fetchData(); // This should call mockApi.getData()

// Step 3: Verify that getData() was called

*verify*(mockApi).getData();

}

//

*@Test*

public void testExternalApi() {

// Step 1: Create a mock for the ExternalApi

ExternalApi mockApi = *mock*(ExternalApi.class);

// Step 2: Stub the method getData() to return "Mock Data"

*when*(mockApi.getData()).thenReturn("Mock Data");

// Step 3: Inject mock into service and call fetchData()

MyService service = new MyService(mockApi);

String result = service.fetchData();

// Assert the expected output

*assertEquals*("Mock Data", result);

}

// Setup method runs before each test

*@Before*

public void setUp() {

calculator = new Calculator();

System.***out***.println("Setup complete");

}

// Teardown method runs after each test

*@After*

public void tearDown() {

calculator = null;

System.***out***.println("Teardown complete");

}

*@Test*

public void testAddition() {

// Arrange

int a = 5;

int b = 3;

// Act

int result = calculator.add(a, b);

// Assert

*assertEquals*(8, result);

}

*@Test*

public void testSubtraction() {

// Arrange

int a = 10;

int b = 4;

// Act

int result = calculator.subtract(a, b);

// Assert

*assertEquals*(6, result);

}

*@Test*

public void testAdd() {

Calculator calc = new Calculator();

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

// Checks if 2 + 3 = 5

*assertEquals*(5, result);// Now it should work

*assertTrue*(result > 0);

*assertFalse*(result < 0);

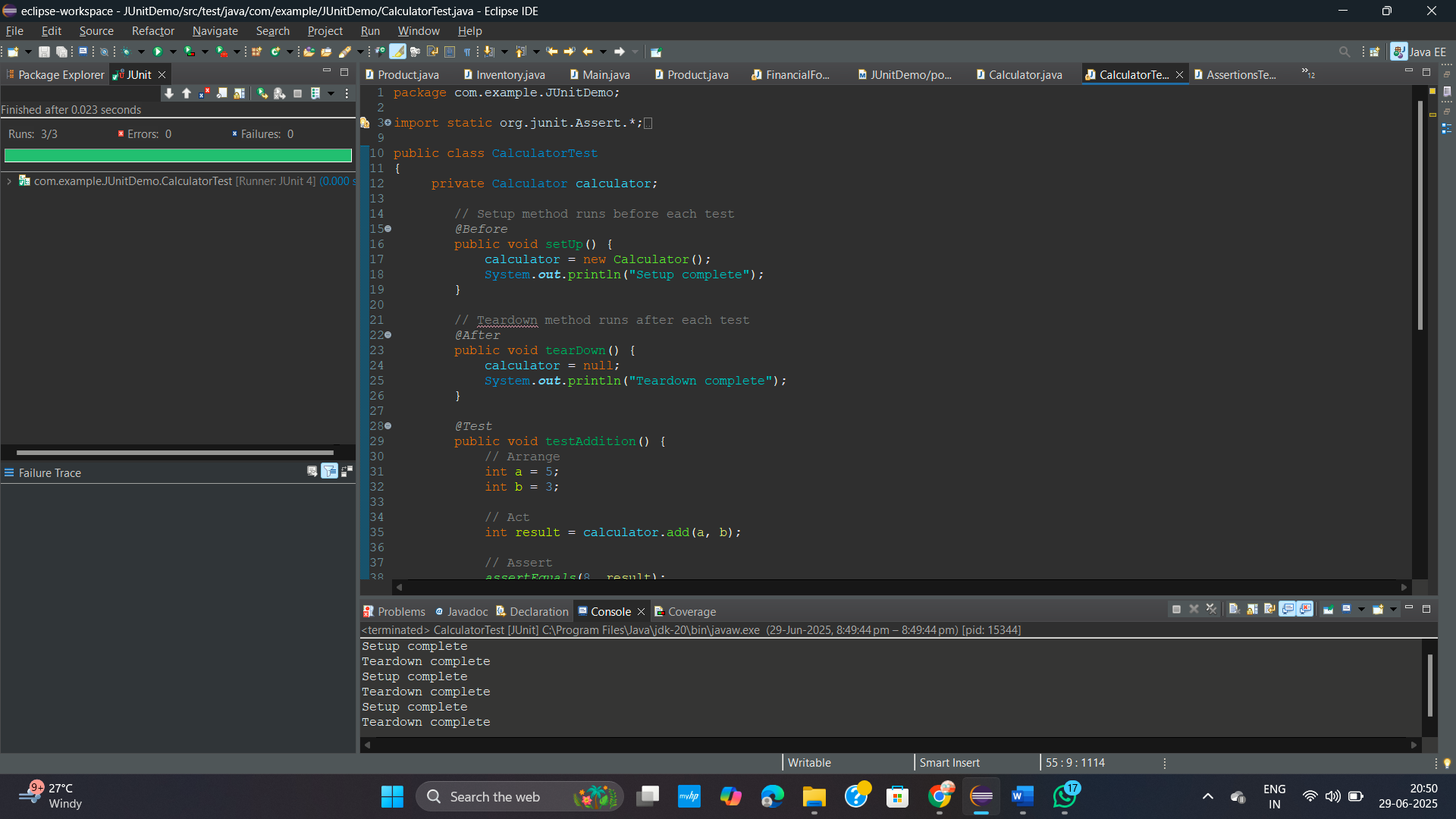
*assertNotNull*(calc);

*assertNull*(null);

}

}

**OUTPUT :**



**DEPENDENCIES :**

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>JUnitDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>JUnitDemo</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.11.0</version>

<scope>test</scope>

</dependency>

<!-- SLF4J API -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<!-- Logback implementation for SLF4J -->

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

</dependencies>

</project>