**Junit Basic Testing**

Exercise 1: Setting Up JUnit

Scenario:

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

Steps:

1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).

2. Add JUnit dependency to your project. If you are using Maven, add the following to your

pom.xml:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

3. Create a new test class in your project.

**Code**

**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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>cts</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>juint\_basic\_testing</name>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**Calculator.java**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

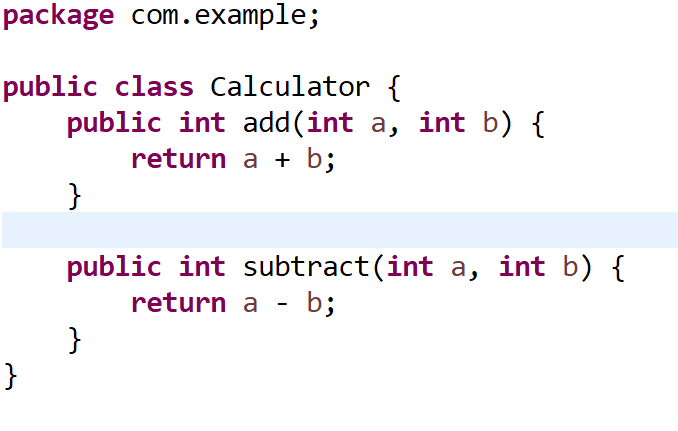
public int subtract(int a, int b) {

return a - b;

}

}

**Output**

****

**Exercise 2**

Scenario:

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

Steps:

1. Create a new Java class with some methods to test.

2. Write JUnit tests for these methods.

**CalculatorTest.java**

package com.example;

import static org.junit.Assert.\*;

import org.junit.Test;

public class CalculatorTest {

Calculator calc=new Calculator();

@Test

public void testAdd() {

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

}

@Test

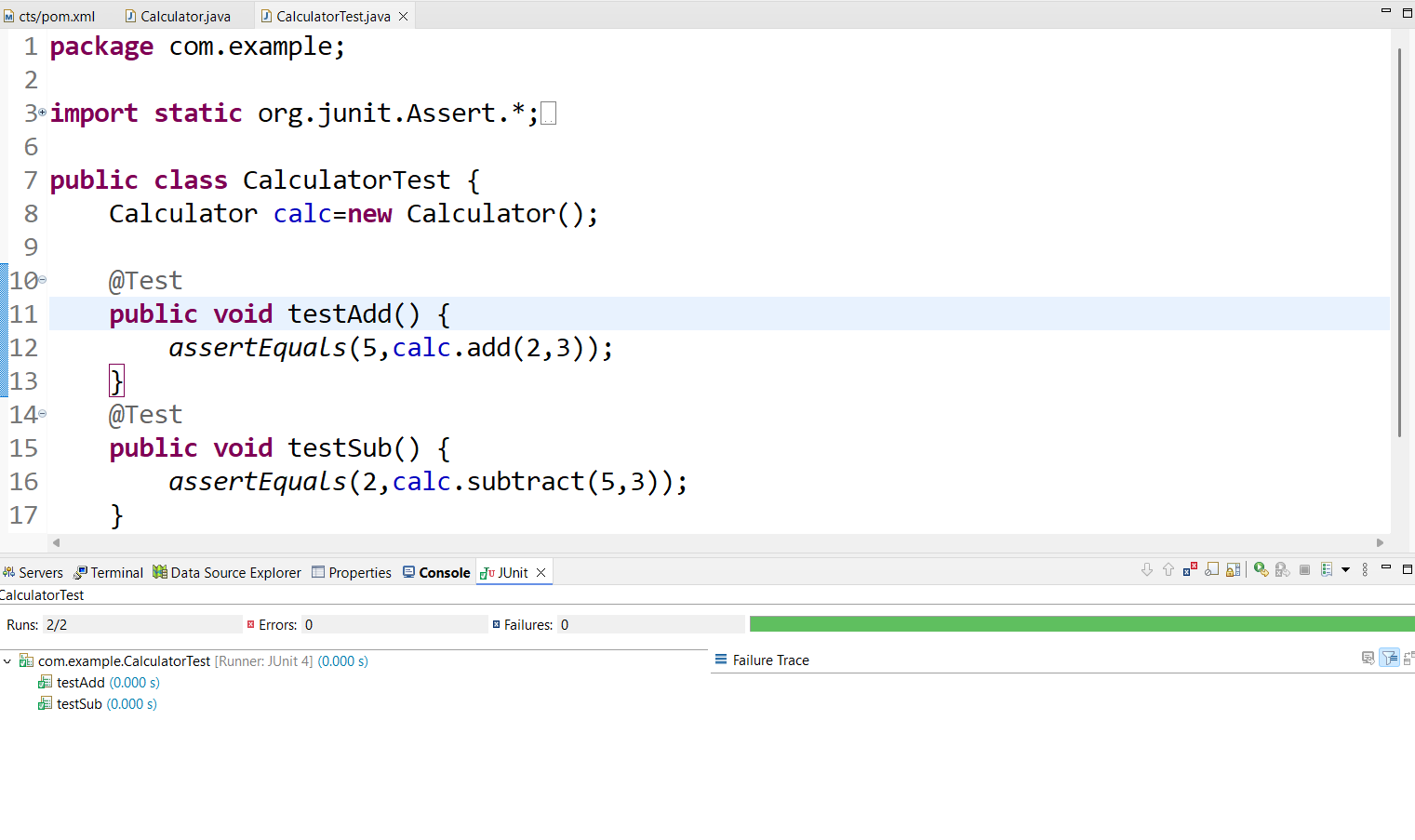
public void testSub() {

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

}

}

**Output**

****

**Exercise 3**

Scenario:

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

Steps:

1. Write tests using various JUnit assertions.

**Code**

public class AssertionsTest {

@Test

public void testAssertions() {

// Assert equals

assertEquals(5, 2 + 3);

// Assert true

assertTrue(5 > 3);

// Assert false

assertFalse(5 < 3);

// Assert null

assertNull(null);

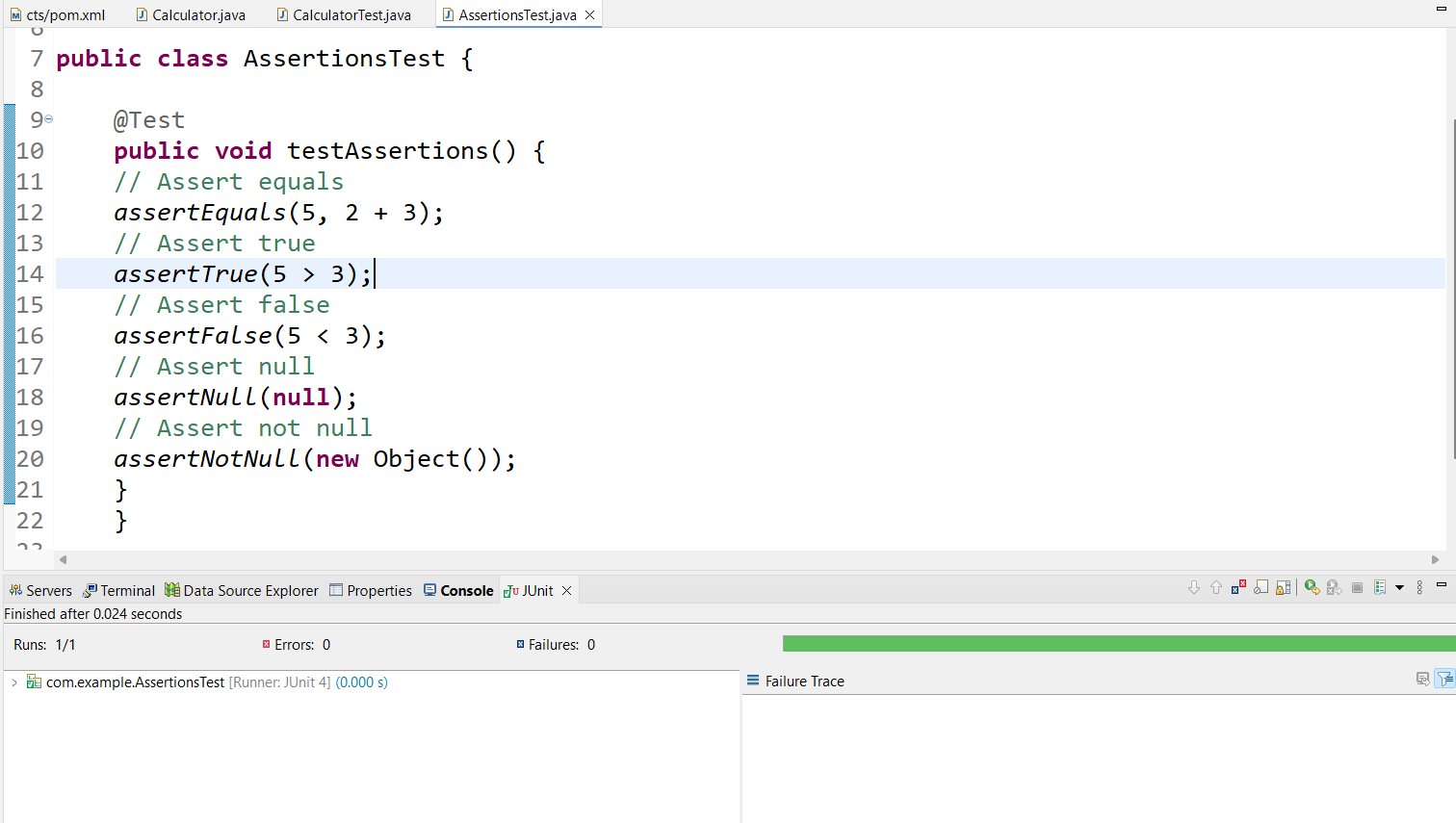
// Assert not null

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.

Steps:

1. Write tests using the AAA pattern.

2. Use @Before and @After annotations for setup and teardown methods

**Code**

**package** com.example;

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

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**public** **class** CalculatorLifeCycleTest {

Calculator calc;

// Setup method – runs BEFORE each test

@Before

**public** **void** setUp() {

System.***out***.println("Setting up calculator...");

calc = **new** Calculator();

}

// Teardown method – runs AFTER each test

@After

**public** **void** tearDown() {

System.***out***.println("Cleaning up calculator...");

calc = **null**;

}

@Test

**public** **void** testAdd() {

// Arrange

**int** a = 5;

**int** b = 3;

// Act

**int** result = calc.add(a, b);

// Assert

*assertEquals*(8, result);

}

@Test

**public** **void** testSubtract() {

// Arrange

**int** a = 10;

**int** b = 6;

// Act

**int** result = calc.subtract(a, b);

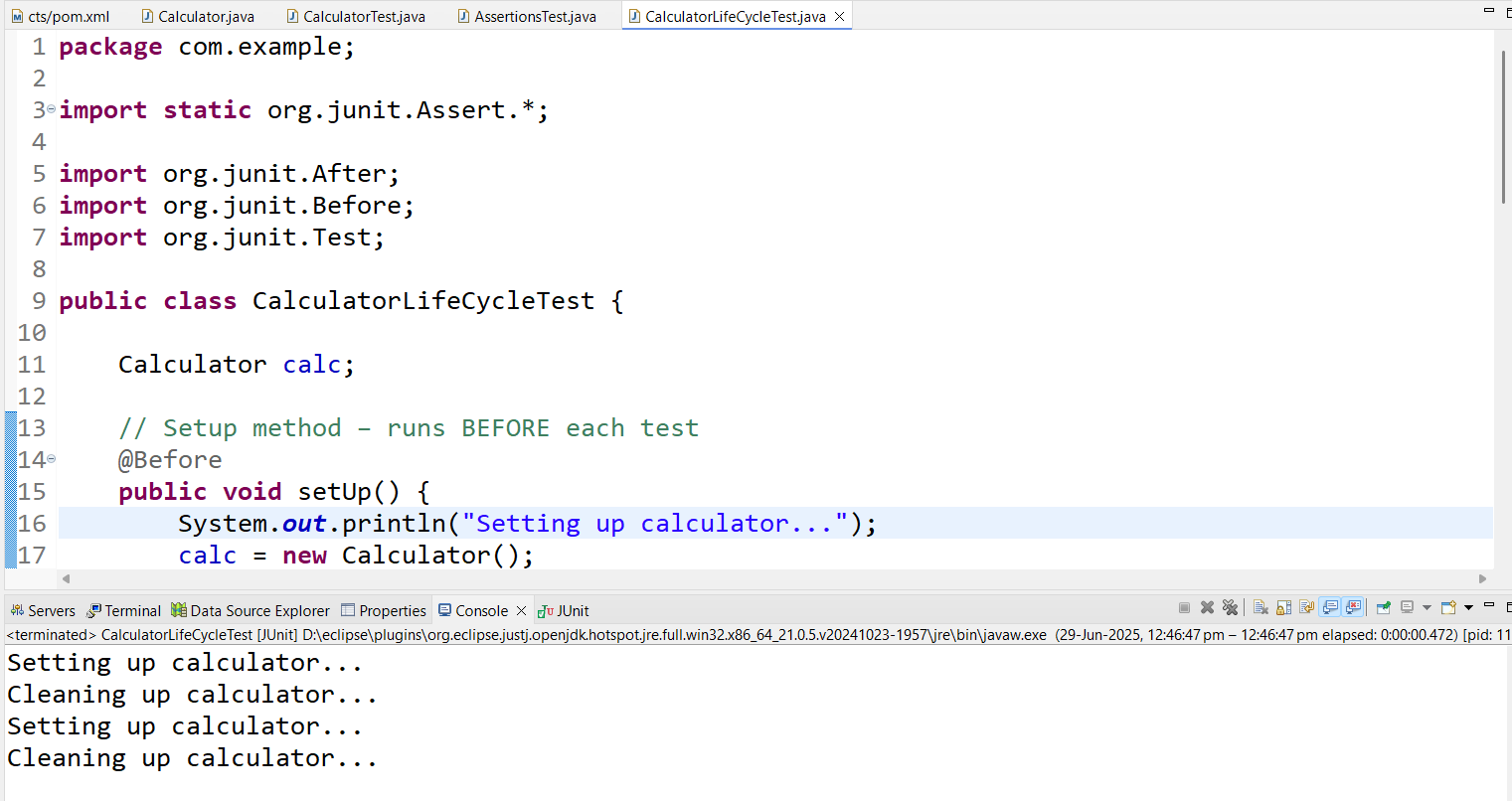
// Assert

*assertEquals*(4, result);

}

}

**Output**

****