**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.

**Solution-**

1. Created new Java project Junit\_testing. Package: in.cognizant.

2. Added dependency. Current 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>in.cognizant</groupId>

<artifactId>Junit\_testing</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

3. Created new test class Calculator.

Code:

**package** in.cognizant;

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

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

**return** a+b;

}

}

**Exercise 2: Writing Basic JUnit Tests**

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.

**Solution-**

1. Created Calculator.java with add method to test.

public class Calculator {

public int add(int a, int b) {

return a+b;

}

}

2. JUnit tests written in TestCalculator.java

import org.junit.Test;

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

public class TestCalculator {

@Test

public void testAdd() {

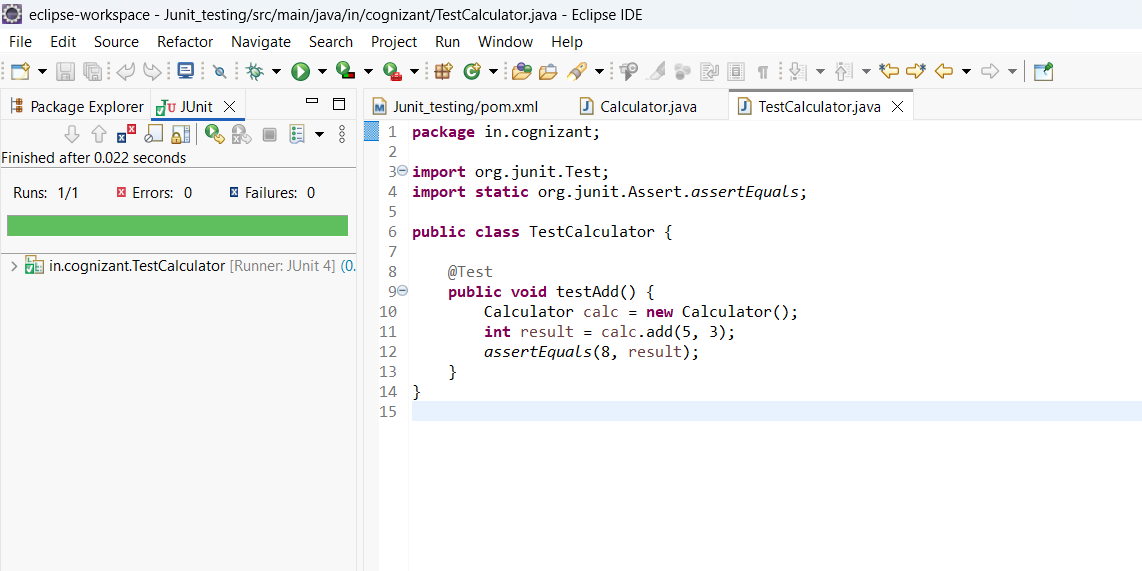
Calculator calc = **new** Calculator();

**int** result = calc.add(5, 3);

*assertEquals*(8, result);

}

}

**Output-**

**Exercise 3: Assertions in JUnit**

Scenario:

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

Steps: 1. Write tests using various JUnit assertions.

**Solution-**

**package** in.cognizant;

**import** org.junit.Test;

**import** **static** org.junit.Assert.\*; // Import all assertion methods

**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**);

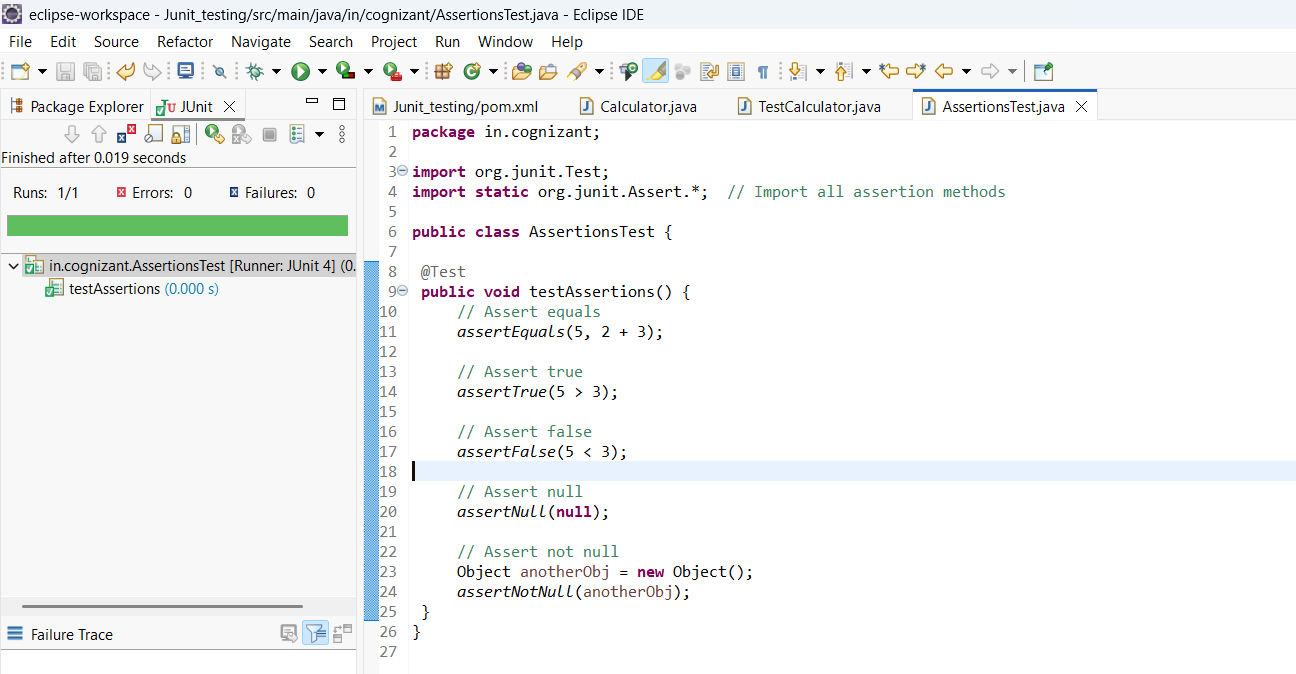
// Assert not null

Object anotherObj = **new** Object();

*assertNotNull*(anotherObj);

}

}

**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.

**Solution-**

We have created Calculator.java with methods add and subtract to test.

**package** in.cognizant;

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

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

**return** a+b;

}

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

**return** a-b;

}

}

**1. Write tests using AAA pattern.**

@Test

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

// Arrange

**int** a = 10;

**int** b = 5;

// Act

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

// Assert

*assertEquals*(15, result);

}

@Test

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

// Arrange

**int** a = 10;

**int** b = 4;

// Act

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

// Assert

*assertEquals*(6, result);

}

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

**package** in.cognizant;

**import** org.junit.\*;

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

**public** **class** TestCalculator {

**private** Calculator calculator;

@Before // Setup

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

calculator = **new** Calculator();

System.***out***.println("Setup: Calculator initialized");

}

@After // Teardown

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

calculator = **null**;

System.***out***.println("Teardown: Calculator cleaned up");

}

@Test

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

// Arrange

**int** a = 10;

**int** b = 5;

// Act

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

// Assert

*assertEquals*(15, result);

}

@Test

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

// Arrange

**int** a = 10;

**int** b = 4;

// Act

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

// Assert

*assertEquals*(6, result);

}

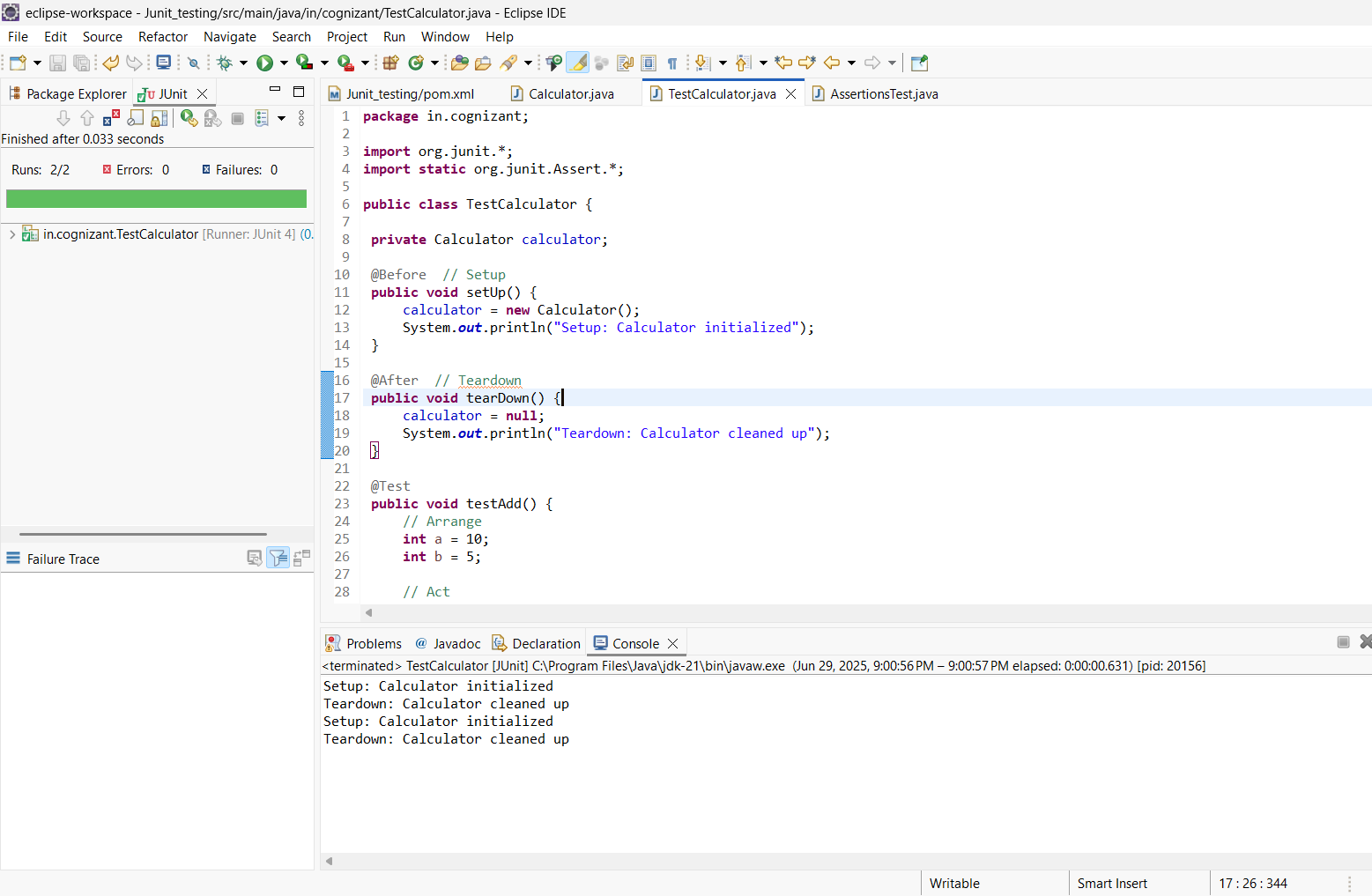
}

Here-

@Before runs **before each test method**.

@After runs **after each test method**.

**Output-**

****