**Exercise 1: Setting Up JUnit Scenario:**

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

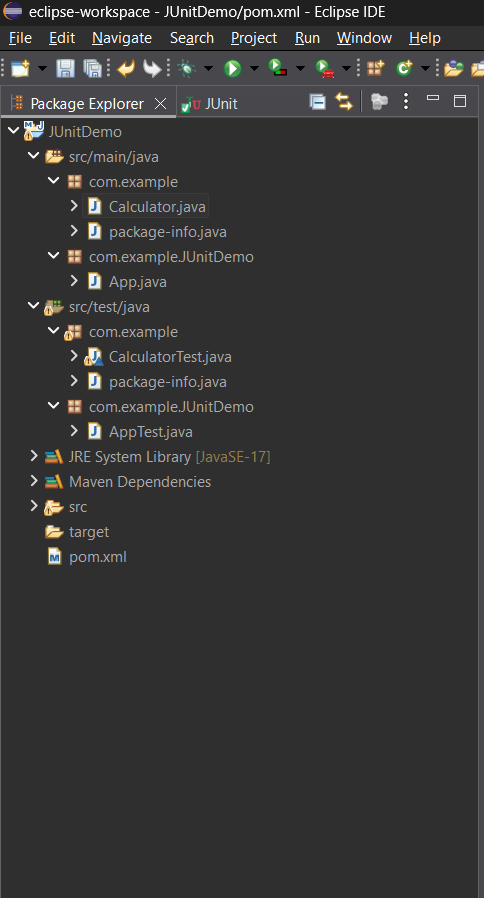
Steps:

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: junit junit 4.13.2 test

3. Create a new test class in your project.

**FOLDER STRUCTURE**

****

**CODE**

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>

<name>JUnitDemo</name>

<url>http://www.example.com</url>

<properties>

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

<maven.compiler.release>17</maven.compiler.release>

<junit.jupiter.version>5.11.0</junit.jupiter.version>

</properties>

<dependencies>

<dependency>

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

<artifactId>junit-jupiter-api</artifactId>

<version>${junit.jupiter.version}</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>junit-jupiter-params</artifactId>

<version>${junit.jupiter.version}</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>junit-jupiter-engine</artifactId>

<version>${junit.jupiter.version}</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.13.0</version>

<configuration>

<release>${maven.compiler.release}</release>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M9</version>

</plugin>

</plugins>

</build>

</project>

**Calculator.java**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import com.example.Calculator;

import static org.junit.jupiter.api.Assertions.\*;

class CalculatorTest {

*@Test*

void testAdd() {

Calculator calc = new Calculator();

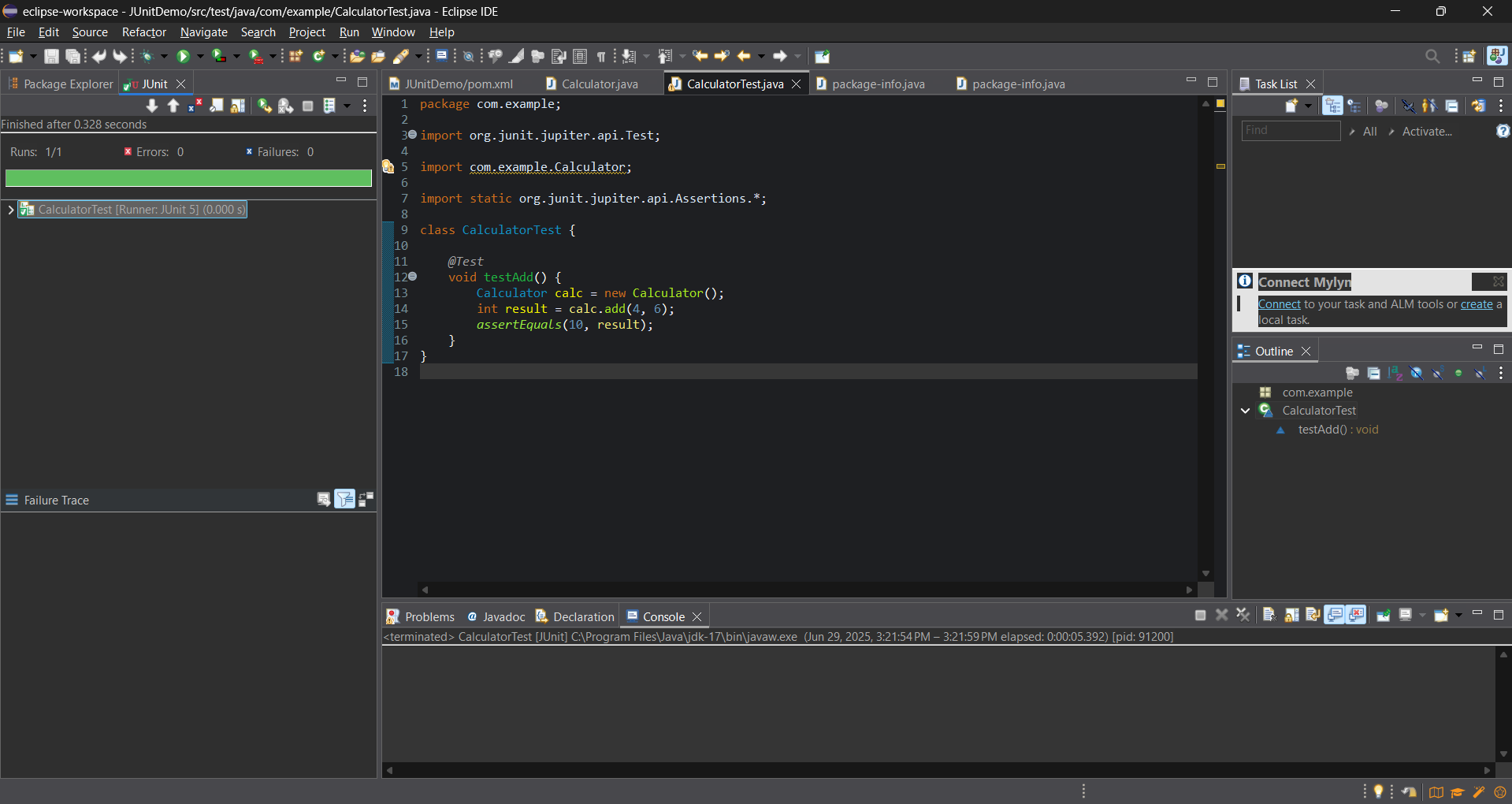
int result = calc.add(4, 6);

*assertEquals*(10, result);

}

}

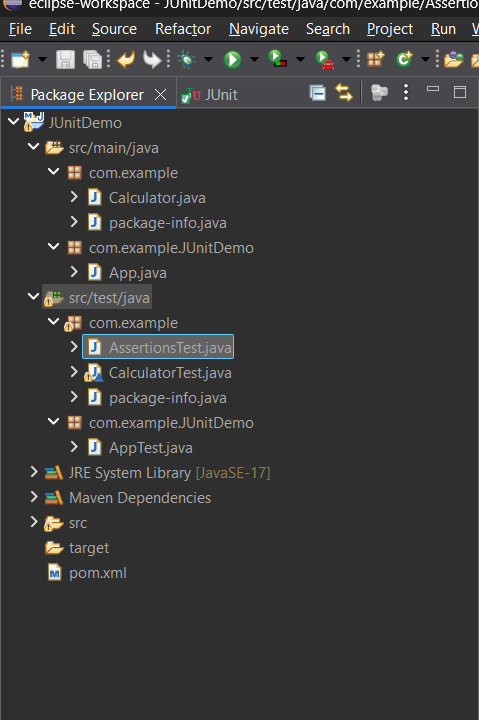
**OUTPUT**

****

**Exercise 3: Assertions in JUnit Scenario:**

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

**FOLDER STRUCTURE**

****

**AssertionsTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

public class AssertionsTest {

*@Test*

public void testAssertions() {

*assertEquals*(5, 2 + 3);

*assertTrue*(5 > 3);

*assertFalse*(5 < 3);

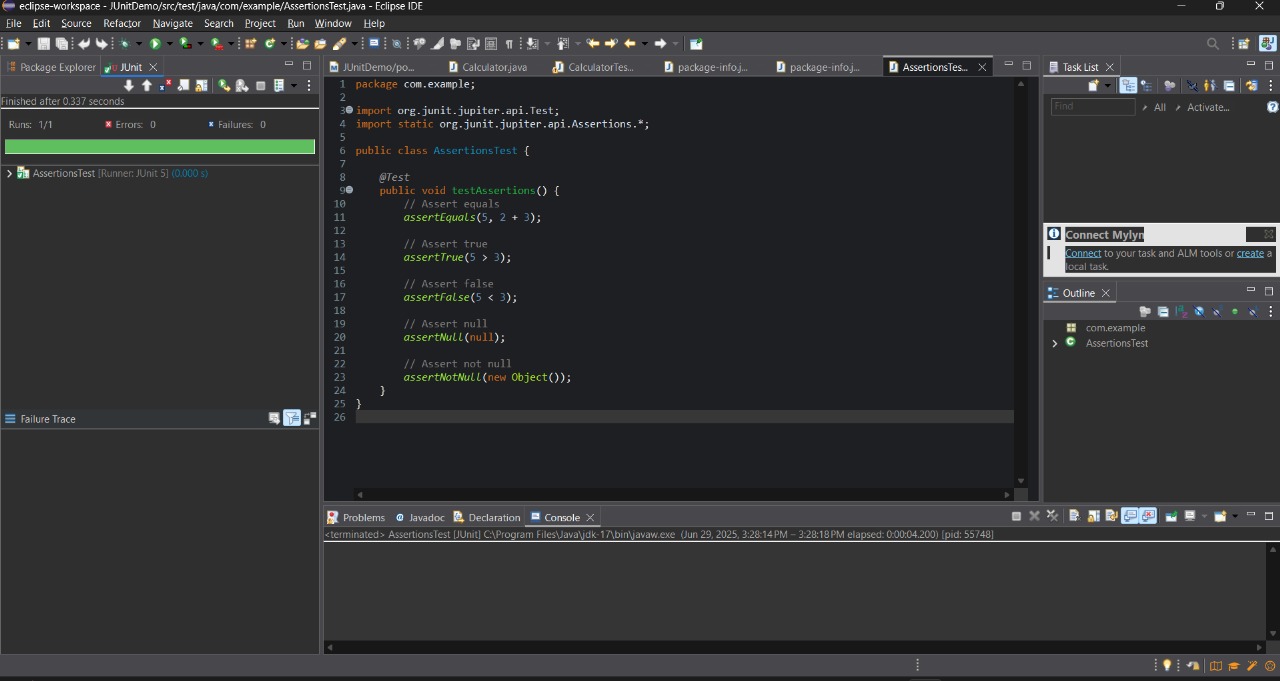
*assertNull*(null);

*assertNotNull*(new Object());

}

}

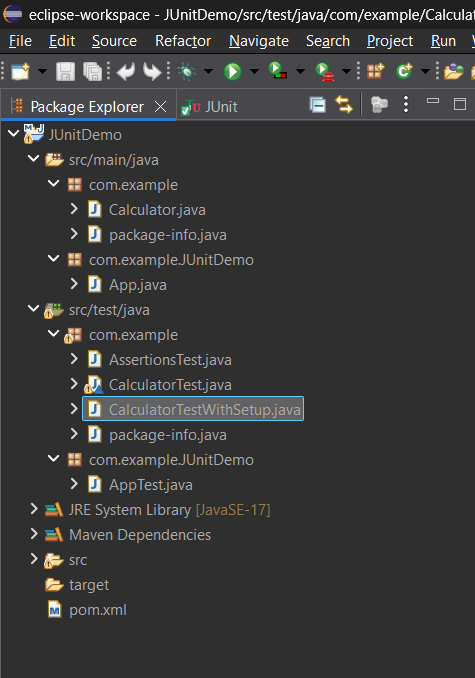
**OUTPUT**



**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.

**FOLDER STRUCTURE**



**CalculatorTestWithSetup.java**

package com.example;

import org.junit.jupiter.api.\*;

import static org.junit.jupiter.api.Assertions.\*;

public class CalculatorTestWithSetup {

private Calculator calculator;

*@BeforeEach*

public void setUp() {

System.*out*.println("Setting up before test...");

calculator = new Calculator(); // Arrange

}

*@AfterEach*

public void tearDown() {

System.*out*.println("Cleaning up after test...");

calculator = null;

}

*@Test*

public void testAddition() {

// Act

int result = calculator.add(10, 5);

// Assert

*assertEquals*(15, result);

}

*@Test*

public void testAnotherAddition() {

// Act

int result = calculator.add(7, 3);

// Assert

*assertEquals*(10, result);

}

}

**OUTPUT**

