WEEK 2-JUNIT TESTING-HANDS ON-EXERCISES

**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.*

**Calculator.java**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

1. *Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml.*

**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/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>junitdemo</artifactId>

<packaging>jar</packaging>

<version>1.0-SNAPSHOT</version>

<name>JUnit Demo</name>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<sourceDirectory>src/main/java</sourceDirectory>

<testSourceDirectory>src/test/java</testSourceDirectory>

</build>

</project>

1. *Create a new test class in your project.*

**CalculatorTest.java**

package com.example;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

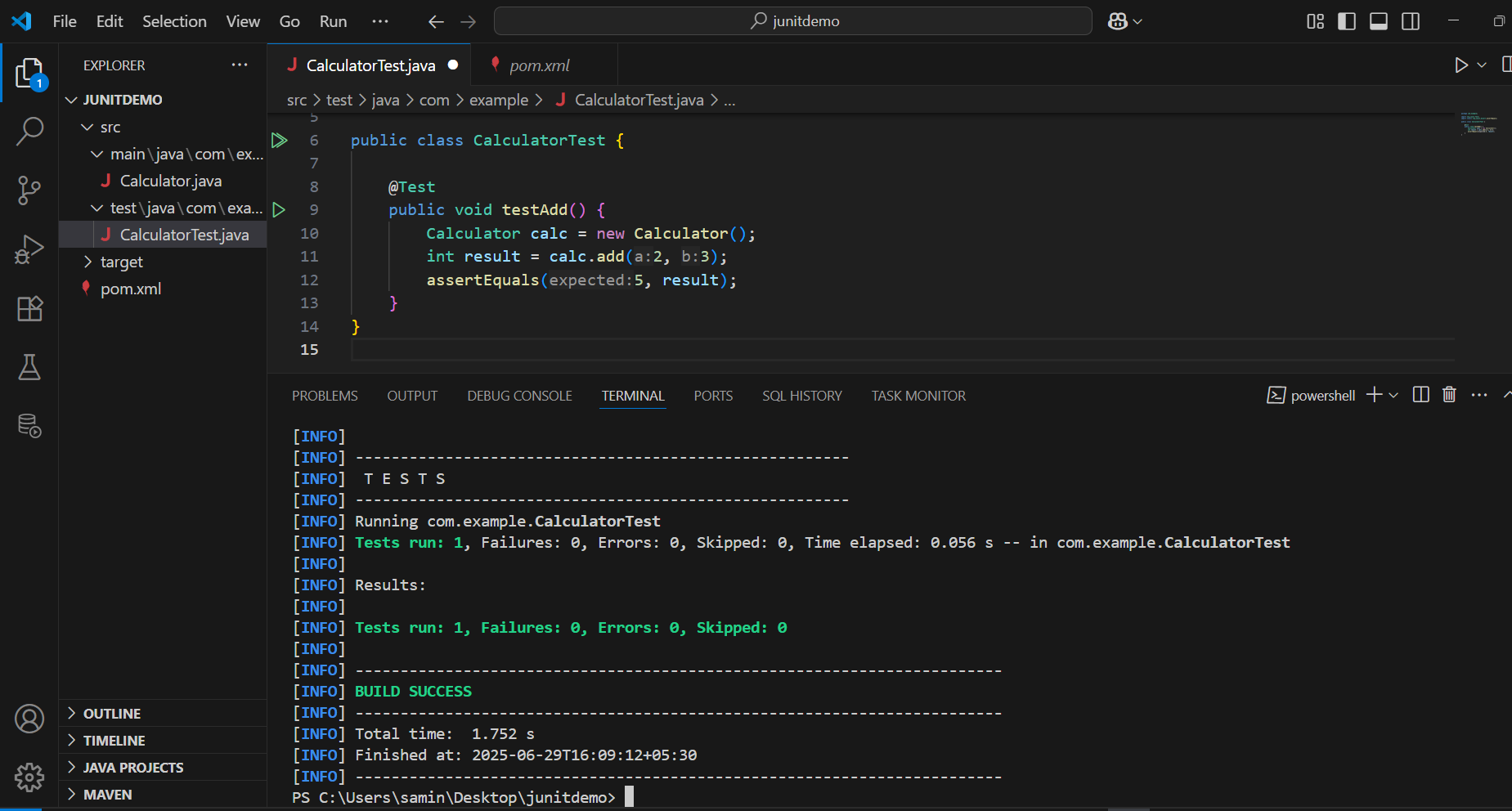
int result = calc.add(2, 3);

assertEquals(5, 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.*

**AssertionsTest.java**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

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);

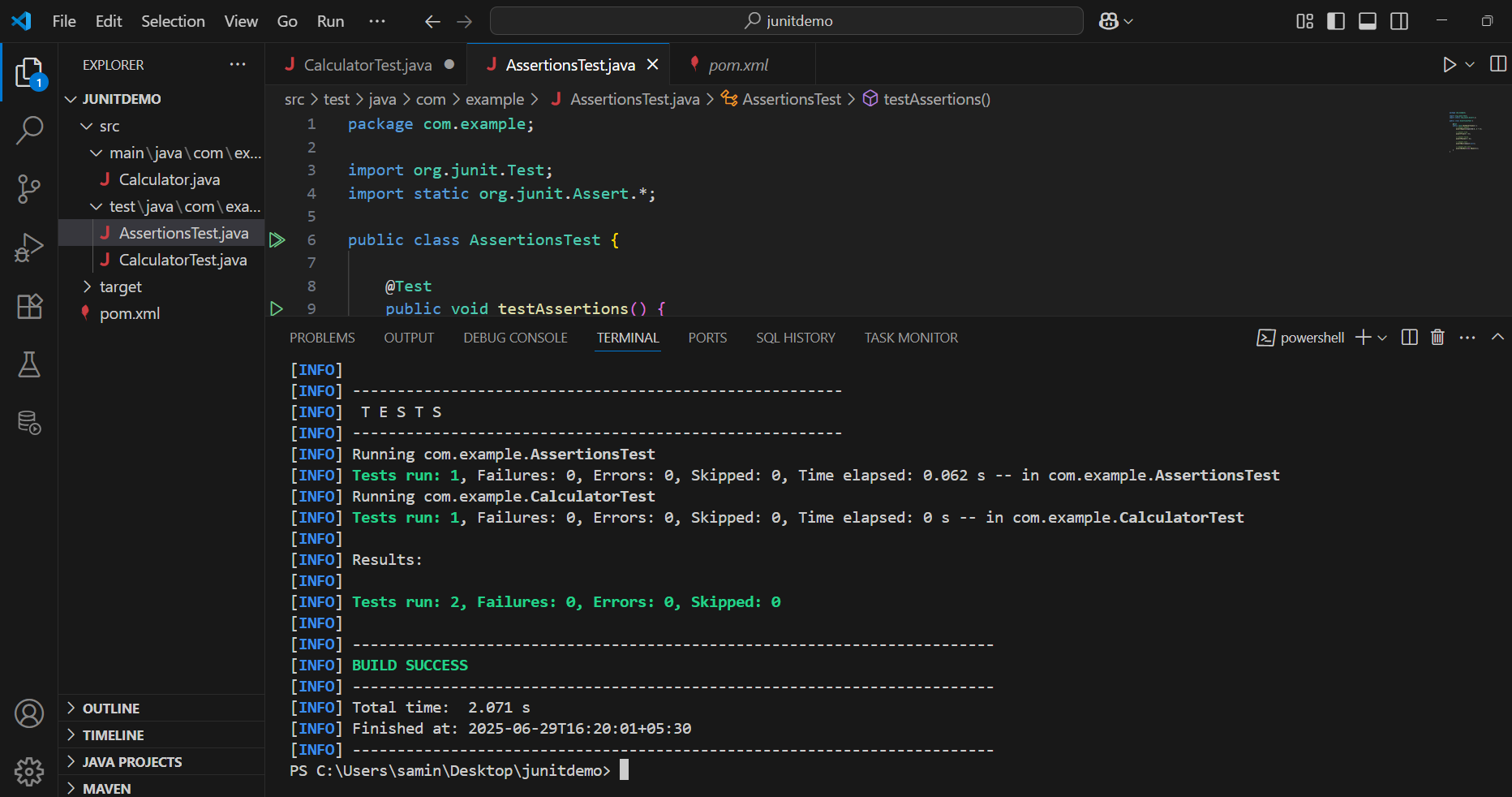
// Assert not null

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 org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

@Before

public void setUp() {

calculator = new Calculator();

System.out.println("Before each test");

}

@After

public void tearDown() {

calculator = null;

System.out.println("After each test");

}

@Test

public void testAdd() {

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

assertEquals(5, result);

}

@Test

public void testAddNegativeNumbers() {

int result = calculator.add(-2, -3);

assertEquals(-5, result);

}

@Test

public void testAddWithZero() {

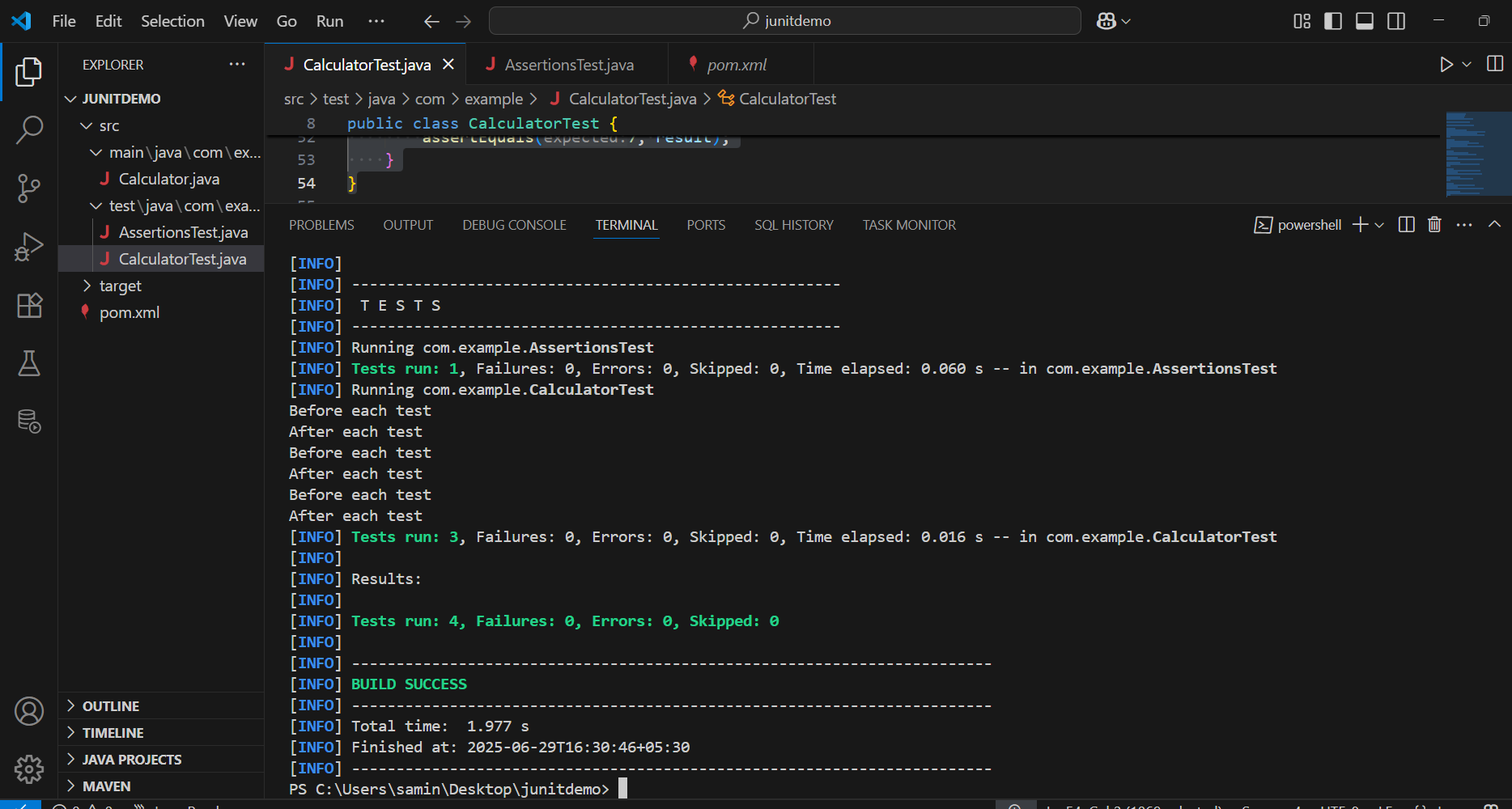
int result = calculator.add(0, 7);

assertEquals(7, result);

}

}

**OUTPUT:**

****