**EXERCISE 1: SETTING UP JUNIT**

**PROGRAM:**

**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>JUnitManualProject</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 4.13.2 -->

<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;

}

}

**CalculatorTest.java**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

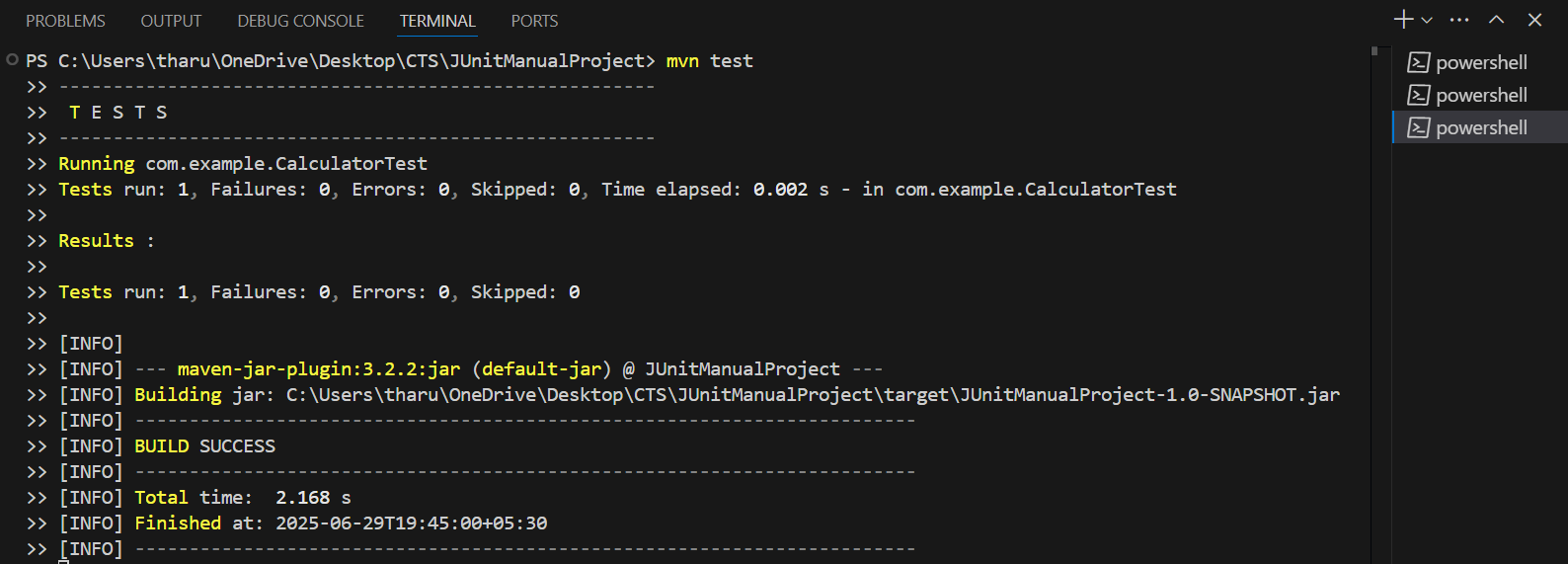
int result = calc.add(2, 3);

assertEquals(5, result);

}

}

**OUTPUT:**



**EXERCISE 3: ASSERTIONS IN JUNIT**

**PROGRAM:**

**pom.xml**

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

**AssertionsTest.java**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

// assertEquals

assertEquals(5, 2 + 3);

// assertTrue

assertTrue(5 > 3);

// assertFalse

assertFalse(5 < 3);

// assertNull

assertNull(null);

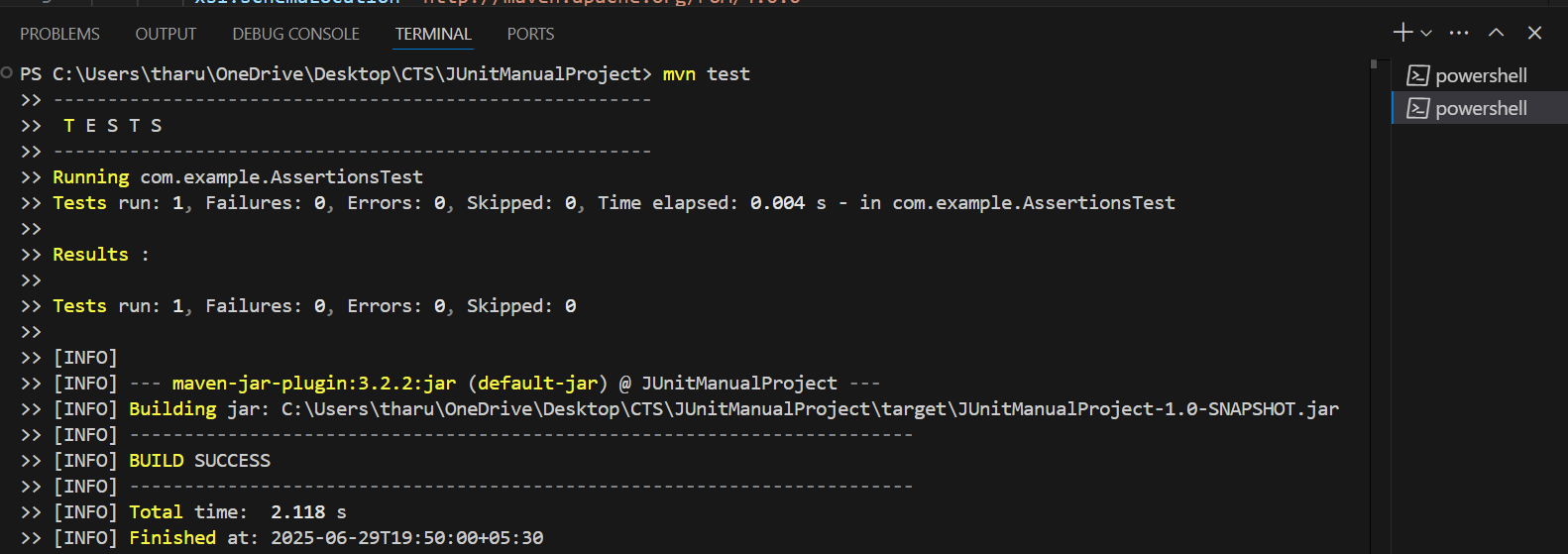
// assertNotNull

assertNotNull(new Object());

}

}

**OUTPUT:**



**EXERCISE 4: ARRANGE-ACT-ASSERT (AAA) PATTERN, TEST FIXTURES, SETUP AND TEARDOWN METHODS IN JUNIT**

**PROGRAM:**

**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;

}

}

**CalculatorTest.java**

package com.example;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calc;

@Before

public void setUp() {

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

calc = new Calculator(); // Arrange

}

@After

public void tearDown() {

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

}

@Test

public void testAdd() {

// Act

int result = calc.add(4, 5);

// Assert

assertEquals(9, result);

}

@Test

public void testSubtract() {

// Act

int result = calc.subtract(10, 3);

// Assert

assertEquals(7, result);

}

}

**pom.xml**

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

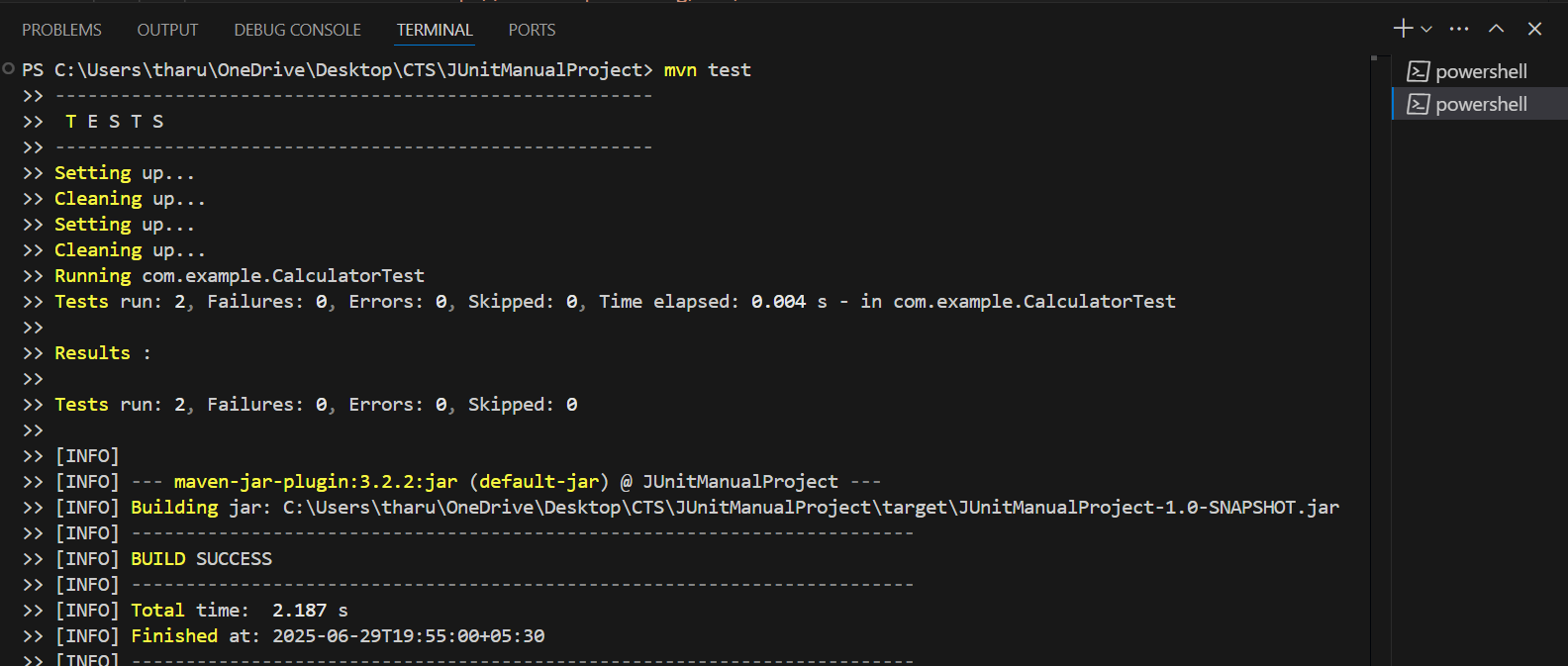
<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

**OUTPUT**



**EXERCISE 1: MOCKING AND STUBBING**

**PROGRAM:**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testExternalApi() {

// Step 1: Create mock

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Stub method

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

// Step 3: Use service

MyService service = new MyService(mockApi);

String result = service.fetchData();

// Step 4: Assert

assertEquals("Mock Data", result);

}

}

**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>JUnitManualProject</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 4 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.12.0</version>

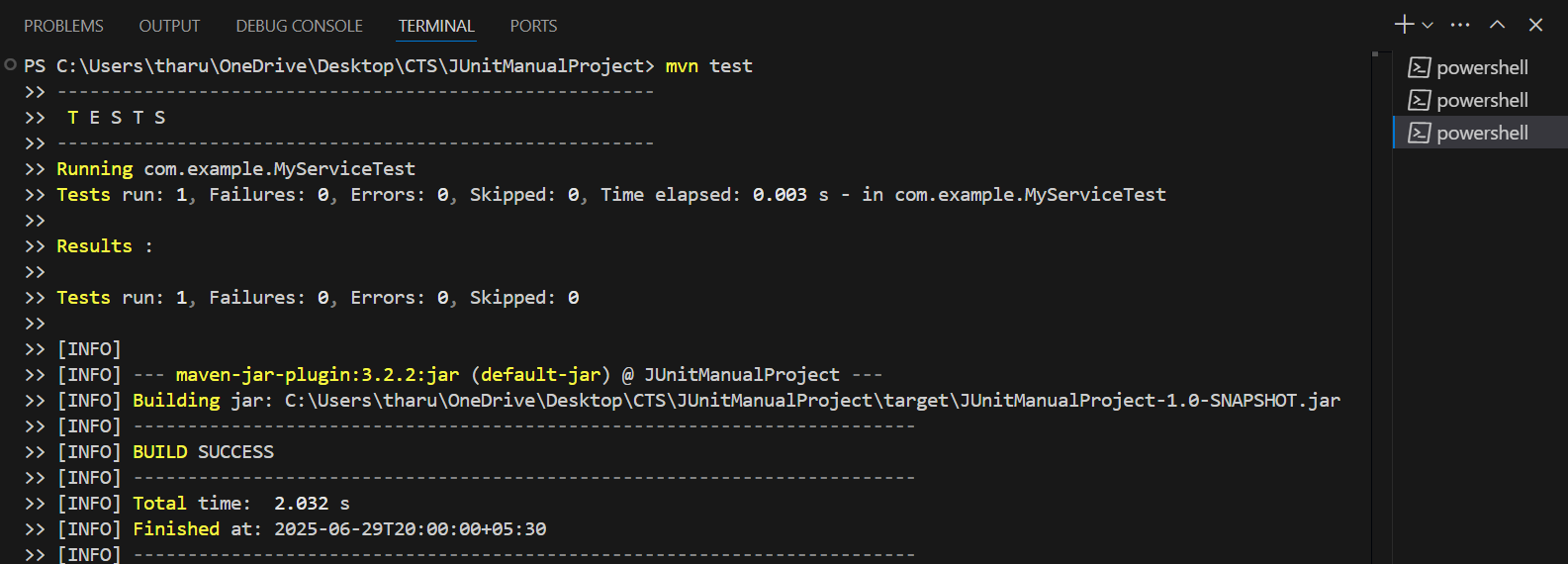
<scope>test</scope>

</dependency>

</dependencies>

</project>

**OUTPUT:**

****

**EXERCISE 2: VERIFYING INTERACTIONS**

**PROGRAM**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

// Step 1: Create mock object

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Call method using the mock

MyService service = new MyService(mockApi);

service.fetchData();

// Step 3: Verify method interaction

verify(mockApi).getData(); // Passes if getData() is called exactly once

}

}

**pom.xml**

<dependencies>

<!-- 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>

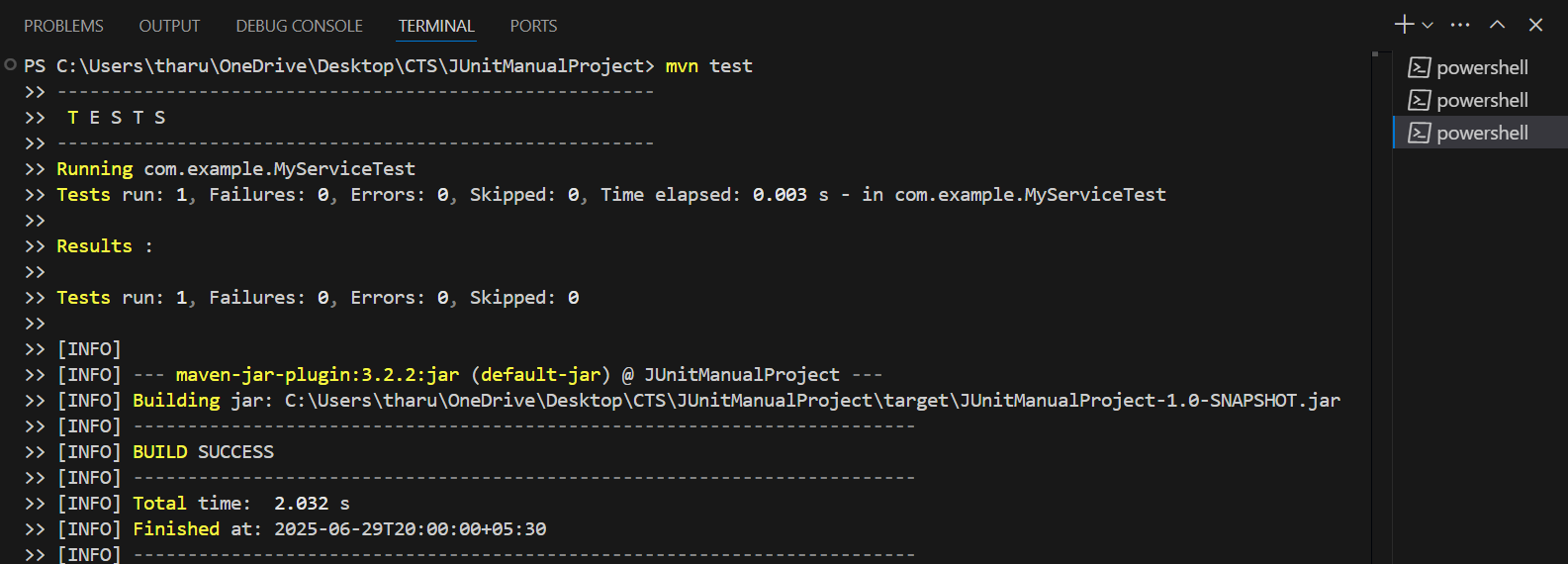
<version>5.12.0</version>

<scope>test</scope>

</dependency>

</dependencies>

**OUTPUT:**

****