**WEEK - 2**

**JUnit Basic Testing**

**Exercise 1 : Setting Up JUnit**

**-- 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>JUnitExample</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>

</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.assertEquals;

public class calculatortest {

@Test

public void testAdd() {

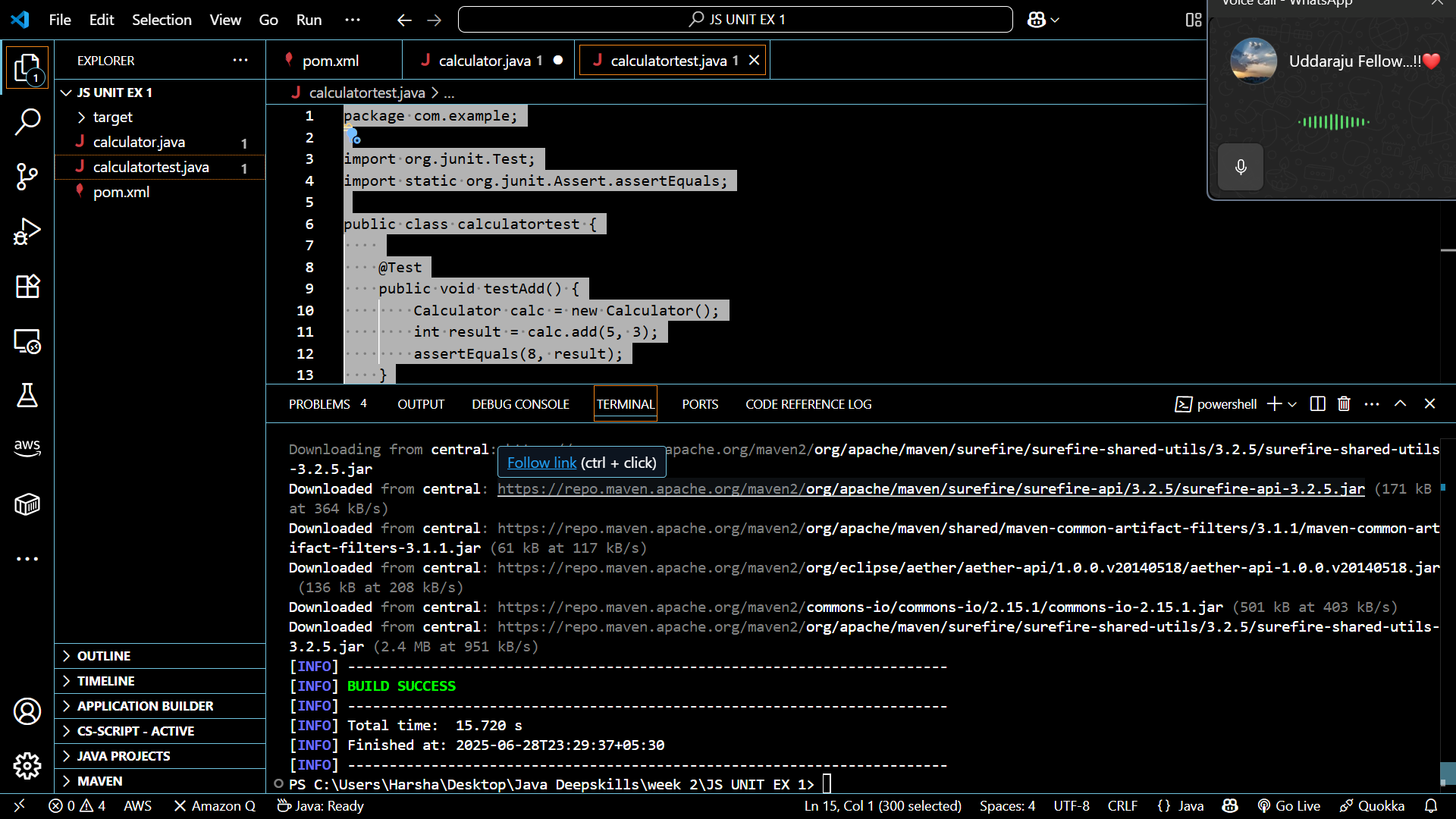
Calculator calc = new Calculator();

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

assertEquals(8, result);

}

}



**Exercise 3: Assertions in JUnit**

**-- AssertionsTest**

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

// Assert equals

assertEquals("Sum should be 5", 5, 2 + 3);

// Assert true

assertTrue("5 should be greater than 3", 5 > 3);

// Assert false

assertFalse("5 should not be less than 3", 5 < 3);

// Assert null

Object obj = null;

assertNull("Object should be null", obj);

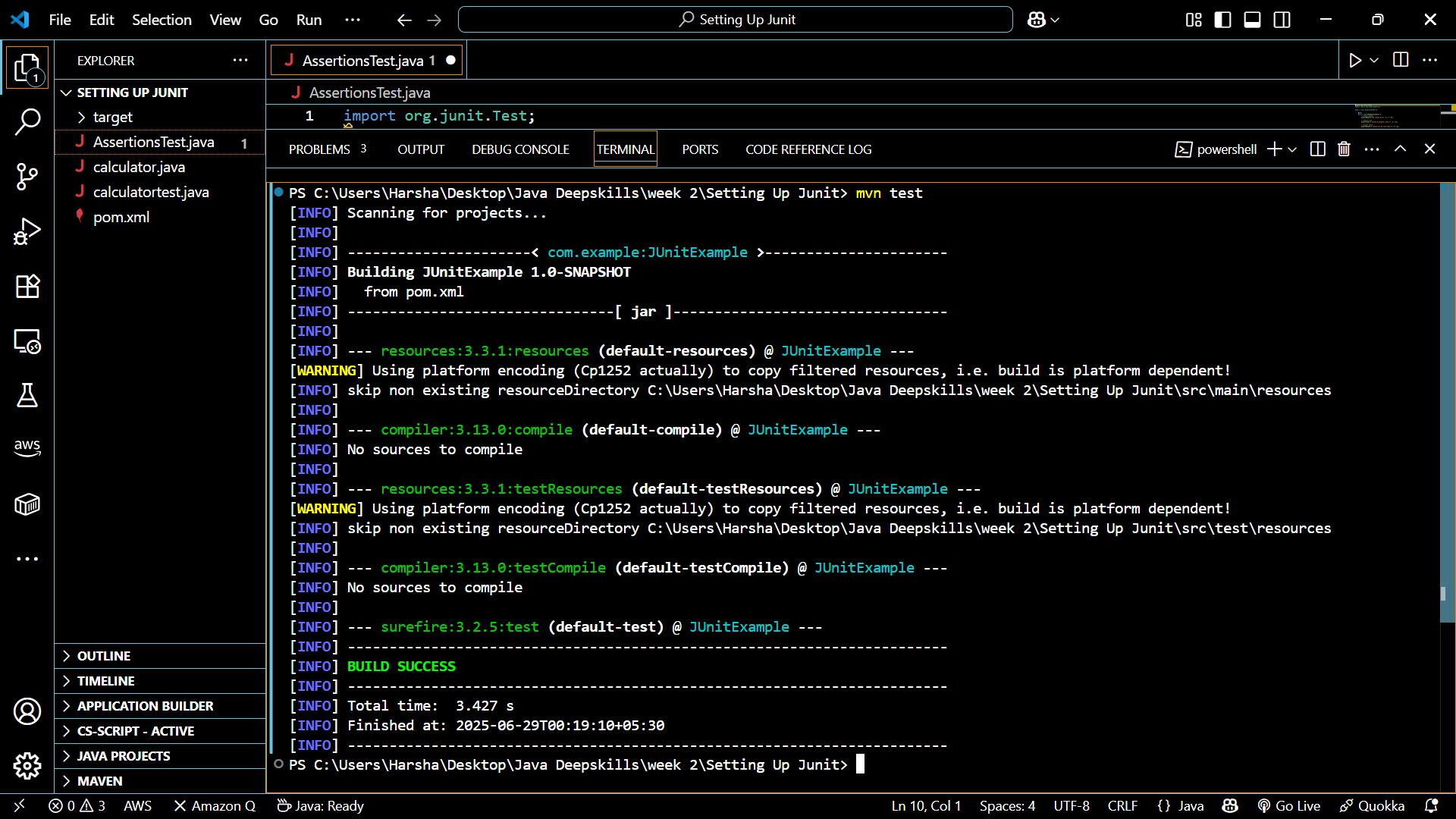
// Assert not null

Object nonNullObj = new Object();

assertNotNull("Object should not be null", nonNullObj);

}

}



**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in Junit**

**Code1: Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

**Code2: CalculatorTest.java**

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("Setup: Calculator instance created");

}

@After

public void tearDown() {

calculator = null;

System.out.println("Teardown: Calculator instance cleared");

}

@Test

public void testAddition() {

// Arrange

int a = 55;

int b = 25;

// Act

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

// Assert

assertEquals(80, result);

}

@Test

public void testSubtraction() {

// Arrange

int a = 50;

int b = 30;

// Act

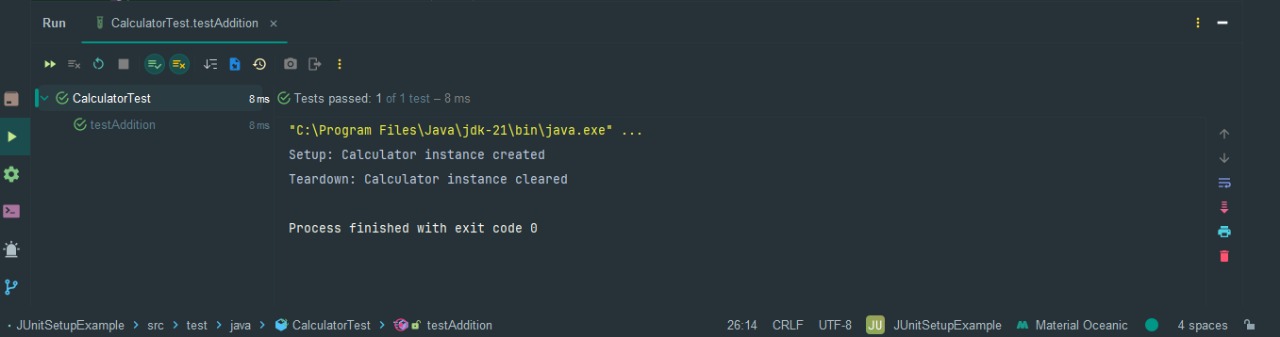
int result = calculator.subtract(a, b);

// Assert

assertEquals(20, result);

}

}



**Mockito**

**Exercise 1: Mocking and Stubbing**

**Code1: ExternalApi.java**

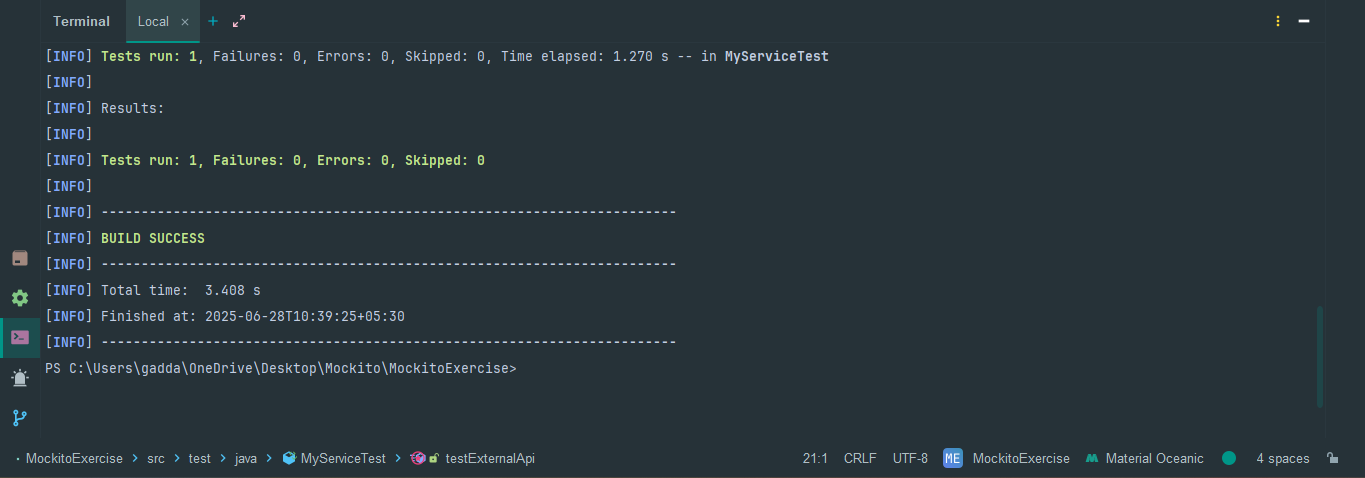
package org.example;  
  
public interface ExternalApi {  
 String getData();  
}

**Code2: MyService.java**

package org.example;  
  
public class MyService {  
 private final ExternalApi api;  
  
 // Constructor injection  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData(); // Calls external API  
 }  
}

**Code3: MyServiceTest.java**

import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
  
import org.example.ExternalApi;  
import org.example.MyService;  
import org.junit.jupiter.api.Test;  
import org.mockito.Mockito;  
  
public class MyServiceTest {  
  
 @Test  
 public void testExternalApi() {  
 // Step 1: Create mock object  
 ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
  
 // Step 2: Stub the getData() method  
 when(mockApi.getData()).thenReturn("Mock Data");  
  
 // Step 3: Inject mock into service  
 MyService service = new MyService(mockApi);  
  
 // Step 4: Call the method and assert result  
 String result = service.fetchData();  
 assertEquals("Mock Data", result); // Assert result matches stubbed value  
 }  
}



**Exercise 2: Verifying Interactions**

**Code1: ExternalApi.java**

package org.example;  
  
public interface ExternalApi {  
 String getData();  
}

**MyService.java**

package org.example;  
  
public class MyService {  
 private ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData(); // This is the call we want to verify  
 }  
}

**Code2: MyServiceTest.java**

import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
  
import org.example.ExternalApi;  
import org.example.MyService;  
import org.junit.jupiter.api.Test;  
import org.mockito.Mockito;  
  
public class MyServiceTest {  
  
 @Test  
 public void testExternalApi() {  
 // Step 1: Create mock object  
 ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
  
 // Step 2: Stub the getData() method  
 when(mockApi.getData()).thenReturn("Mock Data");  
  
 // Step 3: Inject mock into service  
 MyService service = new MyService(mockApi);  
  
 // Step 4: Call the method and assert result  
 String result = service.fetchData();  
 assertEquals("Mock Data", result); // Assert result matches stubbed value  
 }  
}

