**MODULE 2 – TDD JUNIT 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/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>JUnitExample</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**Java Class**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**JUnit Test Class**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

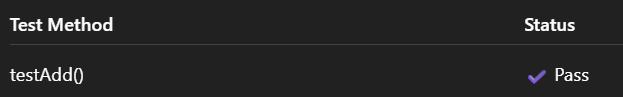
assertEquals(5, calc.add(2, 3));

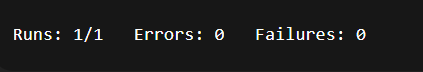
}

}

**Output**

OK (1 test)





**Exercise 3: Assertions in Junit**

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);

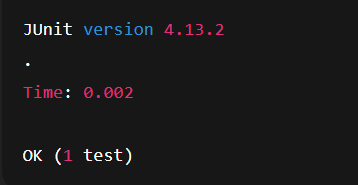
// Assert not null

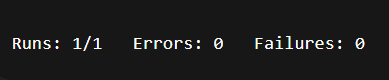
assertNotNull(new Object());

}

}

**Output**





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

package com.example;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTestWithSetup {

private Calculator calculator;

@Before

public void setUp() {

// Setup: runs before each test

calculator = new Calculator();

System.out.println("Setup complete.");

}

@After

public void tearDown() {

// Teardown: runs after each test

calculator = null;

System.out.println("Teardown complete.");

}

@Test

public void testAddition() {

// Arrange

int a = 5;

int b = 3;

// Act

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

// Assert

assertEquals(8, result);

}

@Test

public void testAdditionWithZero() {

// Arrange

int a = 0;

int b = 7;

// Act

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

// Assert

assertEquals(7, result);

}

}

**Calculator.java**

package com.example;

public class Calculator {

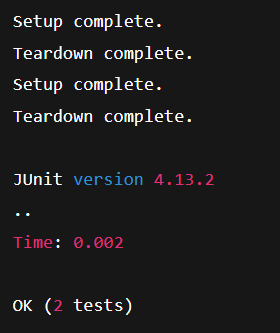
public int add(int a, int b) {

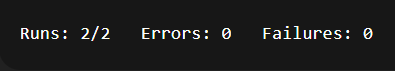
return a + b;

}

}

**Output**





**MOCKITO EXERCISES**

**Exercise 1: Mocking and Stubbing**

package com.example;

public interface ExternalApi {

String getData();

}

package com.example;

public class Myservice {

private ExternalApi api;

public Myservice(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

package com.example.test;

import com.example.ExternalApi;

import com.example.Myservice;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

public class MyserviceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = mock(ExternalApi.class);

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

Myservice service = new Myservice(mockApi);

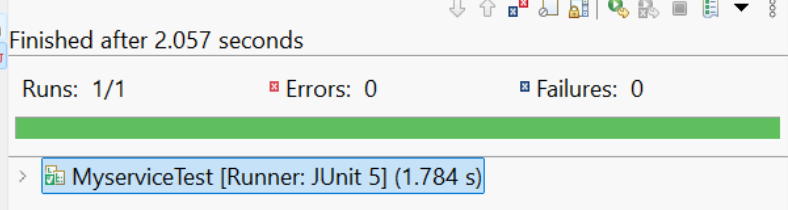
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Output**

****

**Exercise 2: Verifying Interactions**

package com.example;

public interface ExternalApi1 {

String getData();

}

package com.example;

public class Myservice1 {

private ExternalApi1 api;

public Myservice1(ExternalApi1 api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

package com.example.test;

import com.example.ExternalApi1;

import com.example.Myservice1;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyserviceTest1 {

@Test

public void testVerifyInteraction() {

ExternalApi1 mockApi = mock(ExternalApi1.class);

Myservice1 service = new Myservice1(mockApi);

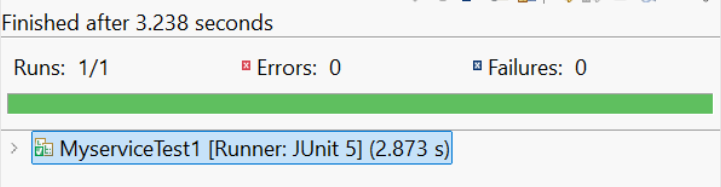
service.fetchData();

verify(mockApi).getData();

}

}

**Output**



**SLF4J logging framework**

**Exercise 1: Logging Error Messages and Warning Levels**

package com.example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class loggingExample {

private static final Logger logger = LoggerFactory.getLogger(loggingExample.class);

public static void main(String[] args) {

logger.error("This is an error message");

logger.warn("This is a warning message");

}

}

<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>loggingDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

</project>

**Output**

