**SLF4J logging framework**

Exercise 1: Logging Error Messages and Warning Levels

Logging-example/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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>logging-example</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>

LoggingExample.java

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");

}

}

Output:

10:36:09.953 [main] ERROR com.example.LoggingExample - This is an error message

10:36:09.963 [main] WARN com.example.LoggingExample - This is a warning message

**TDD using JUnit5 and Mockito**

Exercise 1: Mocking and Stubbing

Mockito-demo/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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>mocikto-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.7.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>3.9.0</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

ExternalApi.java

package com.example.service;

public interface ExternalApi {

String getData();

}

MyService.java

package com.example.service;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData(); // This calls the API

}

}

MyServiceTest.java

package com.example.service;

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

*@Test*

public void testExternalApi() {

ExternalApi mockApi = Mockito.*mock*(ExternalApi.class);

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

MyService service = new MyService(mockApi);

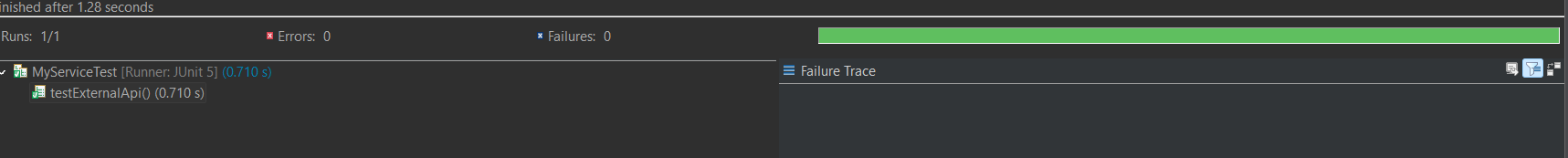
String result = service.fetchData();

*assertEquals*("Mock Data", result);

}

}

Output:



Exercise 2: Verifying Interactions

MyServiceTest.java

package com.example.service;

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

*@Test*

public void testVerifyInteraction() {

// 1. Create mock

ExternalApi mockApi = Mockito.*mock*(ExternalApi.class);

// 2. Inject into MyService

MyService service = new MyService(mockApi);

// 3. Call method

service.fetchData();

// 4. Verify if getData() was called

*verify*(mockApi).getData();

}

*@Test*

public void testExternalApi() {

ExternalApi mockApi = Mockito.*mock*(ExternalApi.class);

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

MyService service = new MyService(mockApi);

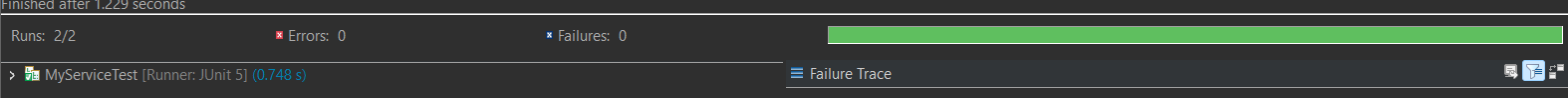
String result = service.fetchData();

*assertEquals*("Mock Data", result);

}

}

Output:



Exercise 1: Setting Up JUnit

Junit-setup/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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>junit-setup</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<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();

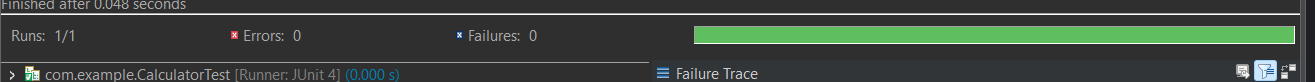
int result = calc.add(2, 3);

*assertEquals*(5, result); // 2 + 3 should be 5

}

}

Output:



Exercise 3: Assertions in JUnit

AssertionsTest.java

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

*@Test*

public void testAssertions() {

*assertEquals*(5, 2 + 3);

*assertTrue*(5 > 3);

*assertFalse*(5 < 3);

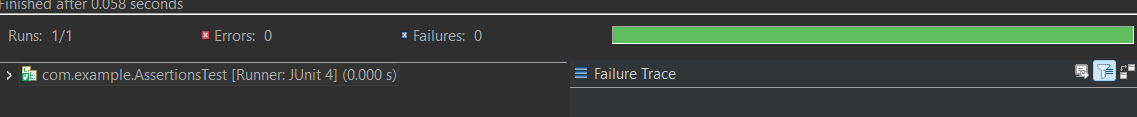
*assertNull*(null);

*assertNotNull*(new Object());

}

}

Output:



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

MathUtil.java

package com.example;

public class MathUtil {

public int multiply(int a, int b) {

return a \* b;

}

public int divide(int a, int b) {

return a / b;

}

}

MathUtilTest.java

package com.example;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class MathUtilTest {

private MathUtil mathUtil;

*@Before*

public void setUp() {

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

mathUtil = new MathUtil();

}

*@After*

public void tearDown() {

System.***out***.println("Tearing down...\n");

}

*@Test*

public void testMultiply() {

int result = mathUtil.multiply(4, 5);

*assertEquals*(20, result);

}

*@Test*

public void testDivide() {

int result = mathUtil.divide(10, 2);

*assertEquals*(5, result);

}

}

Console Output:

Setting up...

Tearing down...

Setting up...

Tearing down...

Junit Output:

