**Digital Nurture 4.0**

**Week 2-Unit Testing**

# JUnit\_Basic Testing Exercises:

Exercise 1: Setting Up JUnit

* Create a maven project with Groupid and Arifactid.

**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>JUnitExample</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>

* In **src/test/java**,Create java class named MyFirstTest.java.

**MyFirstTest.java:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class MyFirstTest {

*@Test*

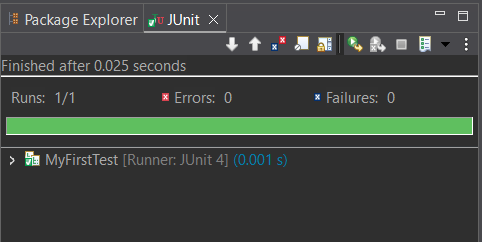
public void sampleTest() {

*assertEquals*(4, 2 + 2);

}

}

**Output:**

****

# Exercise 3: Assertions in JUnit

* In **src/test/java**,Create java class named AssertionsTest.java.

**AssertionsTest.java:**

import static org.junit.Assert.\*;

import org.junit.Test;

public class AssertionsTest {

*@Test*

public void testAssertions() {

*assertEquals*(5, 2 + 3);

*assertTrue*(5 > 3);

*assertFalse*(5 < 3);

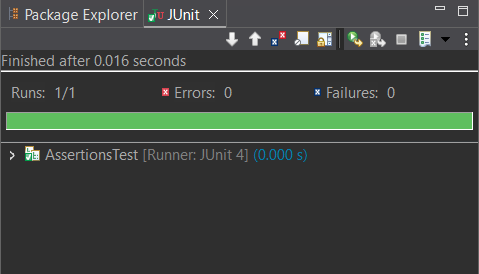
*assertNull*(null);

*assertNotNull*(new Object());

}

}

**Output:**



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

* Create a class named MathOperations.java in src/main/java.

**MathOperations.java:**

public class MathOperations {

public int multiply(int a, int b) {

return a \* b;

}

}

* Create a class named MathOperationsTest.java in src/test/java.

**MathOperationsTest.java:**

import static org.junit.Assert.\*;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

public class MathOperationsTest {

MathOperations math;

*@Before*

public void setUp() {

math = new MathOperations();

System.***out***.println("Calculator instance Setup completed.");

}

*@After*

public void tearDown() {

System.***out***.println("Calculator instance Teardown completed.");

}

*@Test*

public void testMultiply() {

int a = 4;

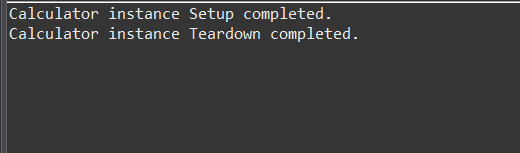
int b = 5;

int result = math.multiply(a, b);

*assertEquals*(20, result);

}

}

**Output:**

3. Mockito exercises:

Exercise 1: Mocking and Stubbing:

* Steup the 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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>MockitoExample</artifactId>

<version>1.0</version>

<dependencies>

<!-- JUnit 5 -->

<dependency>

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

<artifactId>junit-jupiter-api</artifactId>

<version>5.8.2</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>junit-jupiter-engine</artifactId>

<version>5.8.2</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.8.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

</plugin>

</plugins>

</build>

</project>

* Create a Interface named ExternalApi.java in src/main/java.

**ExternalApi.java:**

package com.example;

public interface ExternalApi {

String loadInfo();

}

* Create a Class named MyService.java in src/main/java.

**MyService.java:**

package com.example;

public class MyService {

private ExternalApi dependency;

public MyService(ExternalApi dependency) {

this.dependency = dependency;

}

public String retrieveData() {

return dependency.loadInfo();

}

}

* Create a test class named MyServiceTest.java in src/test/java.

**MyServiceTest.java:**

import com.example.ExternalApi;

import com.example.MyService;

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

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

*@Test*

public void verifyMockData() {

ExternalApi mockDep = *mock*(ExternalApi.class);

*when*(mockDep.loadInfo()).thenReturn("Mock Data Ready");

MyService service = new MyService(mockDep);

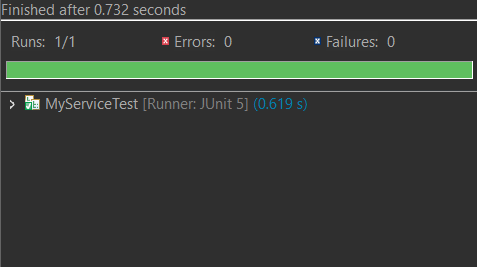
String result = service.retrieveData();

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

}

}

**Output**:



# Exercise 2: Verifying Interactions

* Leave the pom.xml,ExternalApi.java,MyService.java which is created for the previous exercise.
* Create a new test class in src/test/java named MyServiceVerifyTest.java

**MyServiceVerifyTest.java:**

import com.example.ExternalApi;

import com.example.MyService;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceVerifyTest {

*@Test*

public void testVerifyInteraction() {

ExternalApi mockDep = *mock*(ExternalApi.class);

MyService service = new MyService(mockDep);

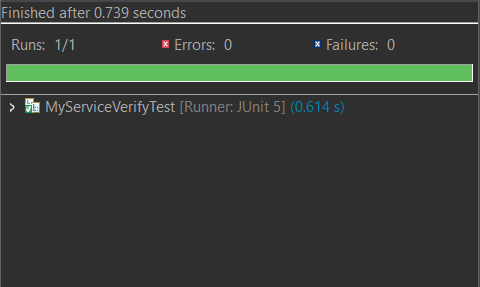
service.retrieveData(); // calls loadInfo()

*verify*(mockDep).loadInfo();

}

}

**Output:**

****

# 6. SL4J Logging exercises

# Exercise 1: Logging Error Messages and Warning Levels

* In pom.xml write the following dependencies.

**Pom.xml:**

package com.logging;

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");

}

}

* Create a package in the src/main/java named com.logging.
* Create a class in the packagecom.logging named LoggingExample.java.

**LoggingExample.java:**

package com.logging;

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) {

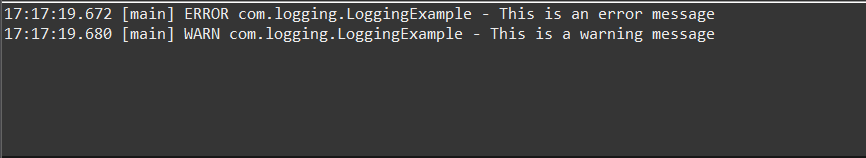
***logger***.error("This is an error message");

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

}

}

**Output:**

****