**Digital Nurture 4.0 – Week 2**

**(ii).Unit Testing**

**1. JUnit\_Basic Testing Exercises:**

Exercise 1: Setting Up JUnit:

**Step 1:**

Create a maven Project with GroupId(com.CTS) and ArtifactId

**pom.xml:**



**Step 2:**

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

**test.java:**

package com.CTS;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class test {

@Test

public void sampleTest() {

MainApp app = new MainApp();

assertEquals(23, app.add(21, 2));

}

}

**Exercise 3: Assertions in JUnit:**

**Step 1:**

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

**AssertionsTest.java:**

package com.CTS;

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);

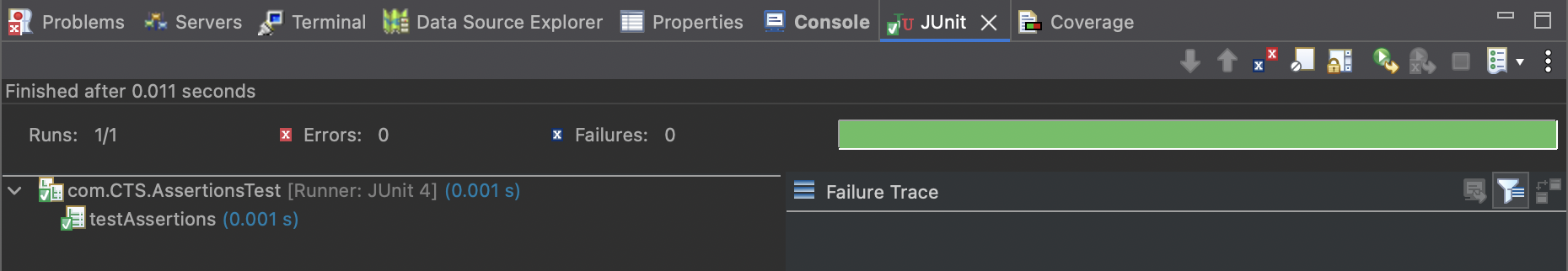
assertNull(null);

assertNotNull(new Object());

}

}

**Output:**



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

**Step 1:**

In **src/main/java/com/CTS,**create a class named Calculator.java.

**Calculator.java:**

package com.CTS;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**Step 2:**

In **src/test/java/com/CTS,**create a class named CalculatorTest.java.

**CalculatorTest.java:**

package com.CTS;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calc;

@Before

public void setUp() {

System.out.println("Setup: Creating Calculator instance");

calc = new Calculator(); // Arrange

}

@After

public void tearDown() {

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

calc = null;

}

@Test

public void testAddition() {

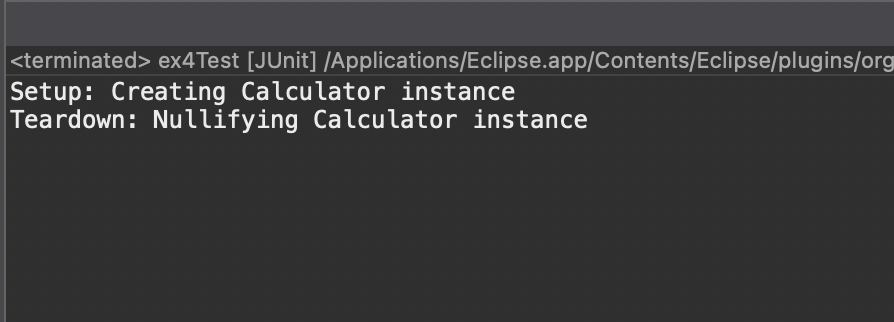
int result = calc.add(10, 20);

assertEquals(30, result);

}

}

**Output:**



**3. Mockito exercises:**

**Exercise 1: Mocking and Stubbing:**

**Step 1:**

Setup 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.CTS</groupId>

<artifactId>MockitoTestProject</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.5.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.0.0-M8</version>

</plugin>

</plugins>

</build>

</project>

**Step 2:**

In **src/main/java/com/example** create an Interface named **ExternalApi.java.**

**ExternalApi.java:**

package com.CTS;

public interface ExternalApi {

String getData();

}

**Step 3:**

In **src/main/java/com/example** create a class named **MyService.java.**

**MyService.java:**

package com.CTS;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**Step 4:**

In **src/test/java/com/example,**create a class named **MyServiceTest.**

**MyServiceTest:**

package com.example;

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);

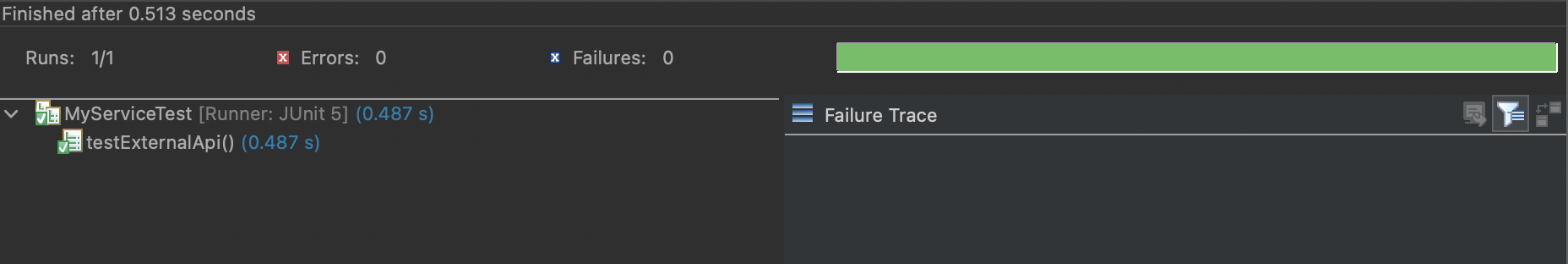
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Output:**



**Exercise 2: Verifying Interactions:**

**Step 1:**

In **src/test/java/com/example,**Create or alter a class named **MyServiceTest.**

package com.example;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

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

MyService service = new MyService(mockApi);

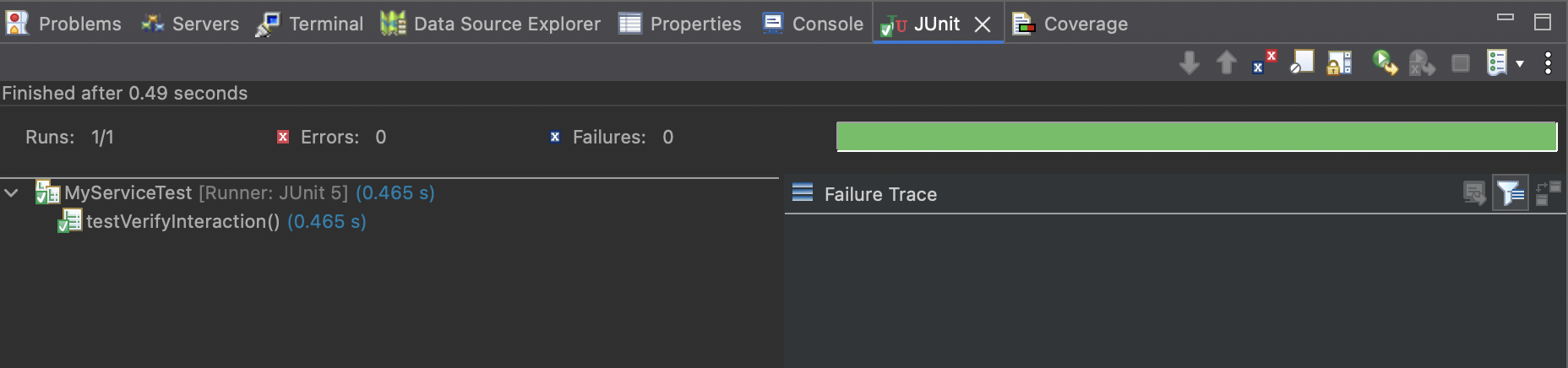
service.fetchData();

verify(mockApi).getData();

}

}

**Ouput:**



**6. SL4J Logging exercises**

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

**Step 1:**

Add the followind dependencies on **pom.xml.**

<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>

**Step 2:**

In **src/main/java/com/example** create an Interface named **LoggingExample.java.**

**LoggingExample.java:**

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");

}

}

**Ouput:**

