WEEK\_2 \_JUNIT\_Mockito\_SLF4J

Project Setup:

Exercise 1: Setting Up JUnit Scenario: You need to set up JUnit in your Java project to start writing unit tests. Steps: 1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse). 2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml.

mvn archetype:generate -DgroupId=com.adarsh -DartifactId=demo -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

Project Folder Structure

//pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<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.adarsh.junitdemo</groupId>

<artifactId>junitsetup</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

Exercise 3: Assertions in Junit

/ src/main/java/com/ayush/Calculator.java

package com.ayush;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

// src/test/java/com/ayush/CalculatorTest.java

package com.ayush;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class CalculatorTest {

@Test

public void testAdd() {

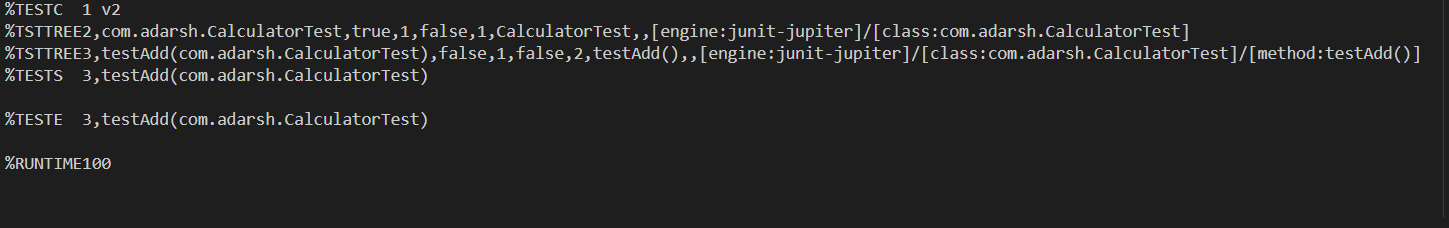
Calculator calc = new Calculator();

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

assertEquals(5, result);

}

}



// -- src/test/java/com/ayush/AssertionsTest.java

package com.ayush;

import static org.junit.Assert.assertEquals;

import static org.junit.Assert.assertFalse;

import static org.junit.Assert.assertNotNull;

import static org.junit.Assert.assertNull;

import static org.junit.Assert.assertTrue;

import org.junit.Test;

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);

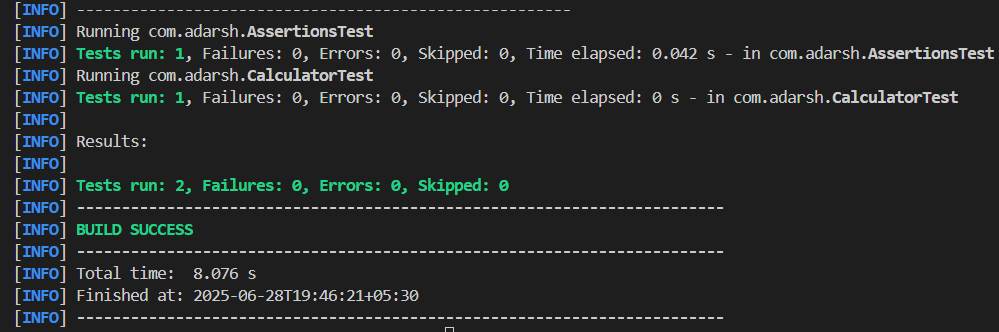
// Assert not null

assertNotNull(new Object());

}

}

Output:



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

package com.adarsh;

import org.junit.jupiter.api.AfterEach;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

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

public class CalculatorLifecycleTest  {

        private Calculator calculator;

    @BeforeEach

    public void setUp() {

        // Setup: runs before each test

        calculator = new Calculator();

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

    }

    @AfterEach

    public void tearDown() {

        // Teardown: runs after each test

        calculator = null;

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

    }

    @Test

    public void testAddition() {

        // Arrange

        int a = 10;

        int b = 5;

        // Act

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

        // Assert

        assertEquals(15, result);

    }

    @Test

    public void testAdditionWithNegativeNumbers() {

        // Arrange

        int a = -3;

        int b = -7;

        // Act

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

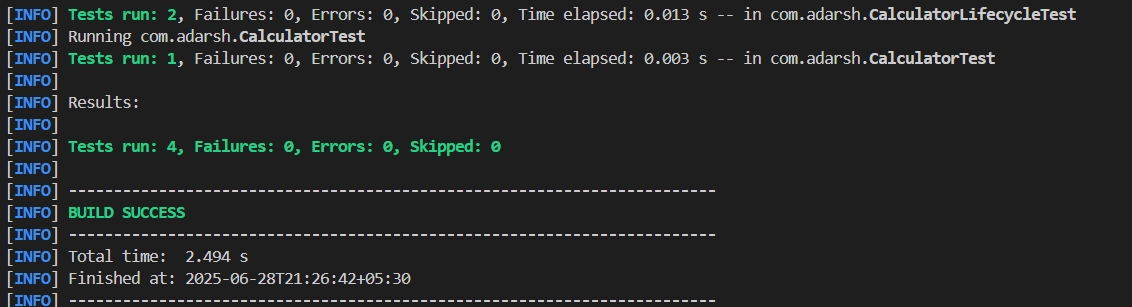
        // Assert

        assertEquals(-10, result);

    }

}

OUTPUT:



**//Mockito**

**Exercise 1: Mocking and Stubbing**

**//pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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.adarsh</groupId>

<artifactId>mockito</artifactId>

<version>1.0-SNAPSHOT</version>

<name>mockito</name>

<url>http://www.example.com</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.release>17</maven.compiler.release>

</properties>

<dependencies>

<!-- JUnit Jupiter API (annotations like @Test) -->

<dependency>

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

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

<version>5.11.0</version>

<scope>test</scope>

</dependency>

<!-- JUnit Jupiter Engine (to run tests) -->

<dependency>

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

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

<version>5.11.0</version>

<scope>test</scope>

</dependency>

<!-- Mockito Core -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>maven-compiler-plugin</artifactId>

<version>3.13.0</version>

</plugin>

<plugin>

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

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

<version>3.0.0-M8</version>

</plugin>

</plugins>

</build>

</project>

**//src/main/java/com/adarsh/ExternalApi.java**

**//ExternalApi.java**

package com.adarsh;

public interface ExternalApi {

String getData();

}

**//src/main/java/com/adarsh/MyService.java**

**//MyService.java**

package com.adarsh;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**//src/test/java/com/adarsh/MyServiceTest.java**

**// MyServiceTest.java**

package com.adarsh;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

public class MyServiceTest {

@Test

public void testExternalApi() {

// Arrange: Create a mock of ExternalApi

ExternalApi mockApi = mock(ExternalApi.class);

// Stub: Define return value for getData()

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

// Act: Use mock in MyService

MyService service = new MyService(mockApi);

String result = service.fetchData();

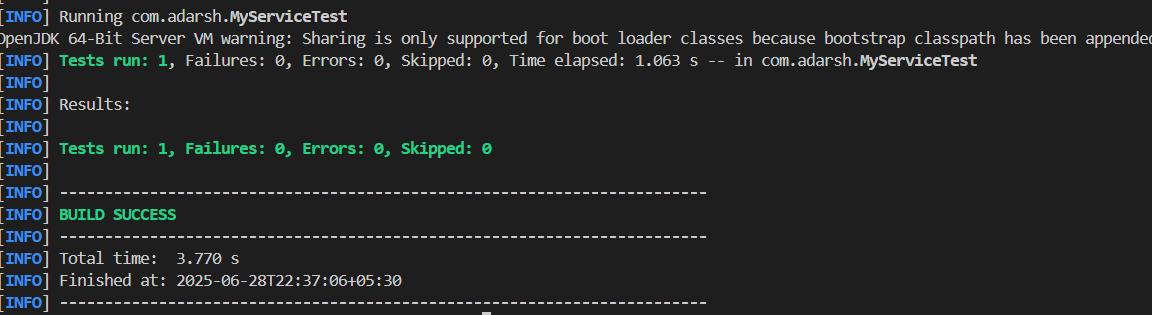
// Assert: Check expected result

assertEquals("Mock Data", result);

}

}

**Output:**



**Exercise 2: Verifying Interactions**

package com.ayush;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

// Step 1: Create a mock

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Use the mock in the service

MyService service = new MyService(mockApi);

service.fetchData(); // internally calls mockApi.getData()

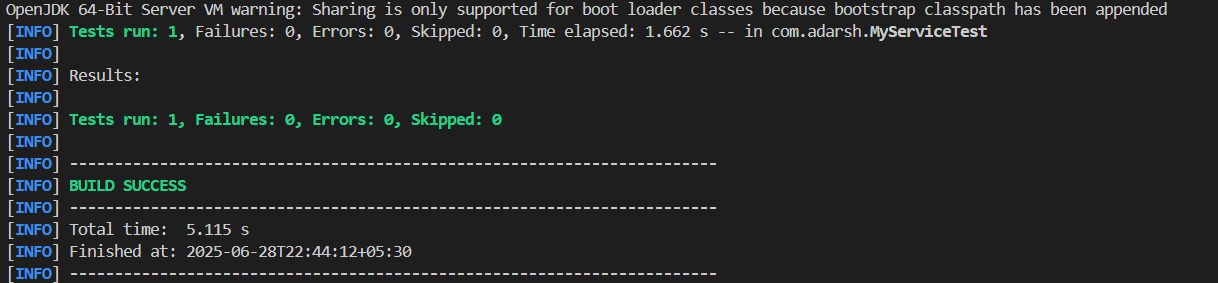
// Step 3: Verify interaction

verify(mockApi).getData();

}

}

**Output:**



**//SL4J Logging**

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

//folder structure

//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.ayush</groupId>

<artifactId>slf4j-logging</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<!-- SLF4J API -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<!-- Logback (SLF4J Implementation) -->

<dependency>

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

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<mainClass>com.ayush.LoggingExample</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

**//LoggingExample.java**

package com.ayush;

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) {

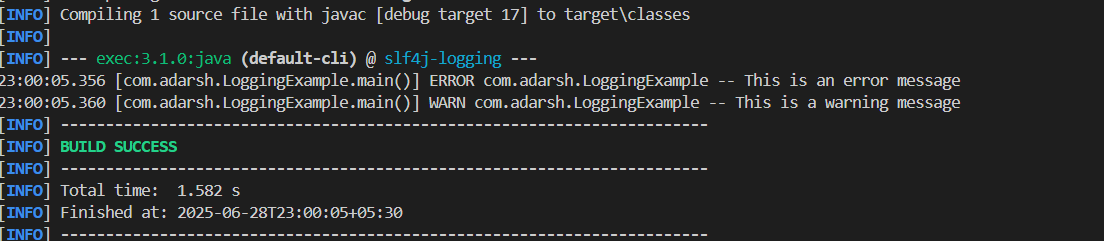
logger.error("This is an error message");

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

}

}

**Output:**

****