**WEEK 2**

**Module 4 – Test driven development and Logging framework**

**JUnit Testing Exercises**

**Exercise 1: Setting Up Junit**

**CODE:**

**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>org.example</groupId>

<artifactId>JUnitDemo</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>24</maven.compiler.source>

<maven.compiler.target>24</maven.compiler.target>

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

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**src/main/java**

**Calculator.java**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

    }

}

**src/test/java**

**CalculatorTest.java**

package com.example;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

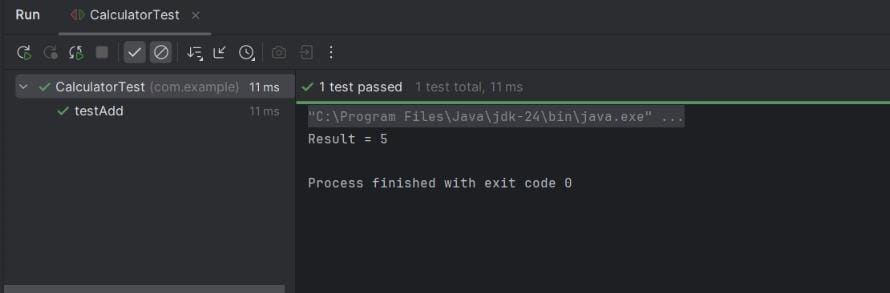
int result = calc.add(2, 3);

System.out.println("Result = " + result);

assertEquals(5, result);

    }

}

**OUTPUT:**

**Exercise 2: Writing Basic JUnit Tests**

**CODE:**

**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>org.example</groupId>

<artifactId>Math</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>24</maven.compiler.source>

<maven.compiler.target>24</maven.compiler.target>

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

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**src/main/java**

**MathUtils.java**

package com.example;

public class MathUtils {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

public int multiply(int a, int b) {

return a \* b;

}

public int divide(int a, int b) {

if (b == 0) throw new IllegalArgumentException("Cannot divide by zero");

return a / b;

}

}

**src/test/java**

**MathUtilsTest.java**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class MathUtilsTest {

@Test

public void testAdd() {

MathUtils math = new MathUtils();

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

}

@Test

public void testSubtract() {

MathUtils math = new MathUtils();

assertEquals(1, math.subtract(4, 3));

}

@Test

public void testMultiply() {

MathUtils math = new MathUtils();

assertEquals(12, math.multiply(3, 4));

}

@Test

public void testDivide() {

MathUtils math = new MathUtils();

assertEquals(2, math.divide(6, 3));

}

@Test(expected = IllegalArgumentException.class)

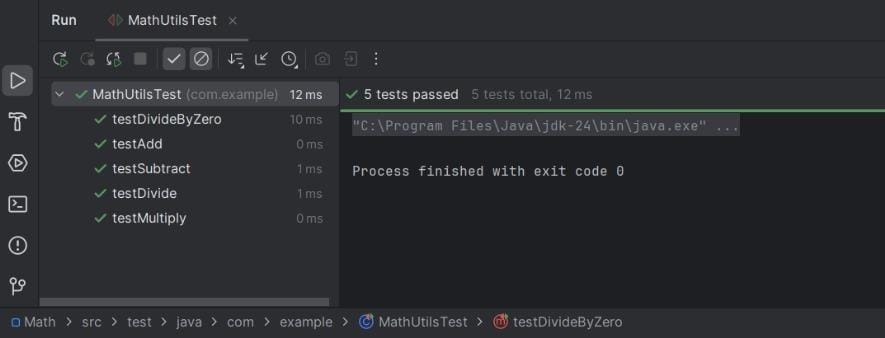
public void testDivideByZero() {

MathUtils math = new MathUtils();

math.divide(10, 0); // should throw exception

}

}

**OUTPUT:**

**Exercise 3: Assertions in Junit**

**CODE:**

**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>org.example</groupId>

<artifactId>JUnitAssertionsDemo</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>24</maven.compiler.source>

<maven.compiler.target>24</maven.compiler.target>

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

</properties>

<dependencies>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**src/test/java**

**AssertionsTest.java**

package com.example;

import org.junit.jupiter.api.Test;

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

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);

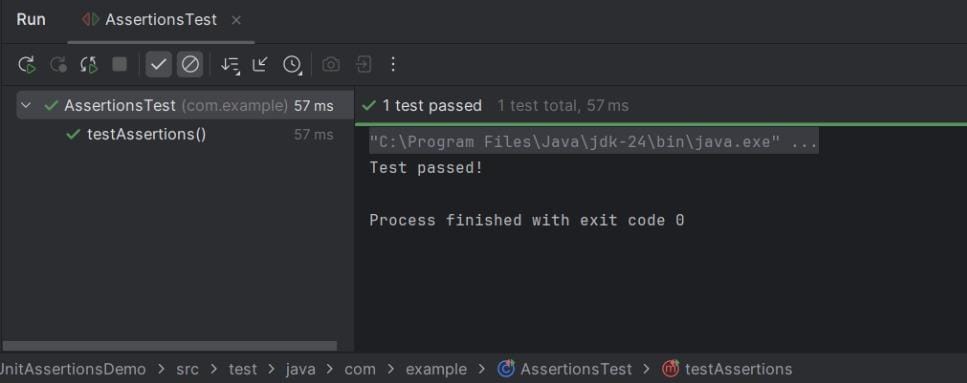
// Assert not null

assertNotNull(new Object());

System.out.println("Test passed!");

}

}

**OUTPUT:**

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

**CODE:**

**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>org.example</groupId>

<artifactId>AAA</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>AAA</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**src/main/java**

**Calculator.java**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

**src/test/java**

**CalculatorTest.java**

package com.example;

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 CalculatorTest {

private Calculator calculator;

@BeforeEach

public void setUp() {

calculator = new Calculator();

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

}

@AfterEach

public void tearDown() {

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

}

@Test

public void testAdd() {

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

assertEquals(5, result);

}

@Test

public void testSubtract() {

int result = calculator.subtract(7, 4);

assertEquals(3, result);

}

}

**src/test/java**

**AppTest.java**

package org.example;

import org.junit.jupiter.api.Test;

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

public class AppTest {

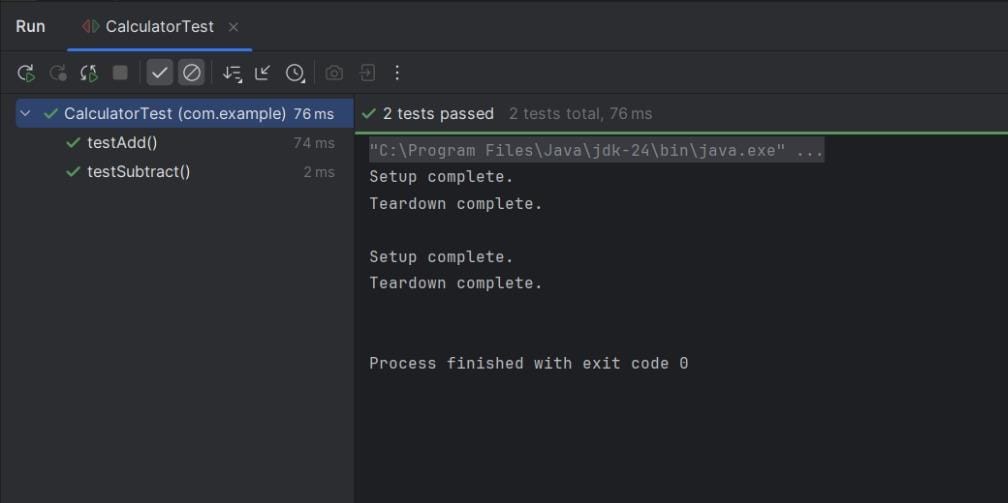
@Test

public void testTrueCondition() {

assertTrue(5 > 3);

}

}

**OUTPUT:**

**Mockito Hands-On Exercises**

**Exercise 1: Mocking and Stubbing**

**CODE:**

**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>org.example</groupId>

<artifactId>MockitoExample</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>MockitoExample</name>

<url>http://maven.apache.org</url>

<properties>

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

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

</properties>

<dependencies>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<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-surefire-plugin</artifactId>

<version>3.2.5</version>

</plugin>

</plugins>

</build>

</project>

**src/main/java**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**src/main/java**

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**src/test/java**

**MyServiceTest.java**

package com.example;

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);

}

}

**src/test/java**

**AppTest.java**

package org.example;

import org.junit.jupiter.api.Test;

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

public class AppTest {

@Test

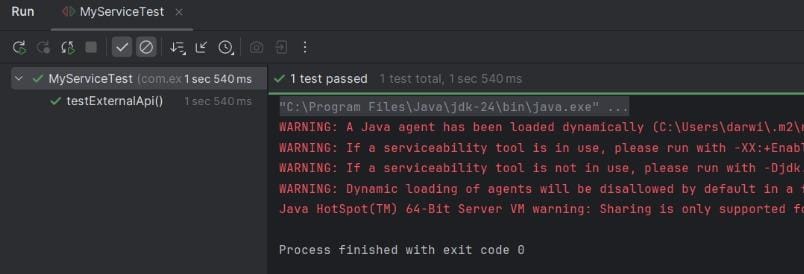
public void testApp() {

assertTrue(true);

}

}

**OUTPUT:**



**Exercise 2: Verifying Interactions**

**CODE:**

**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>org.example</groupId>

<artifactId>VerifyingInteractions</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>VerifyingInteractions</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**src/main/java**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**src/main/java**

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData(); }

}

**src/test/java**

**MyServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

}

}

**src/test/java**

**AppTest.java**

package org.example;

import org.junit.jupiter.api.Test;

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

public class AppTest {

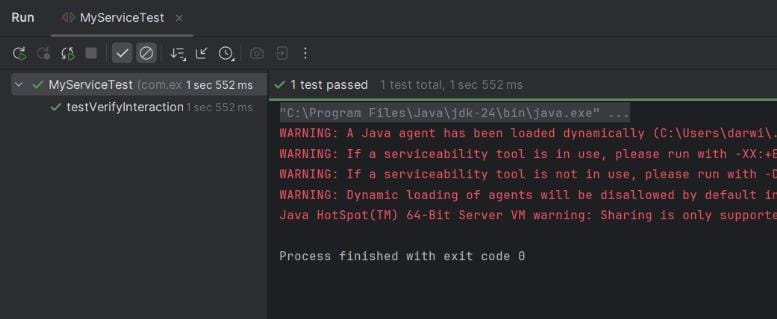
@Test

public void testApp() {

assertTrue(true);

}

}

**OUTPUT:**

**Exercise 3: Argument Matching**

**CODE:**

**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>org.example</groupId>

<artifactId>ArgumentMatching</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>ArgumentMatching</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**src/main/java**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

void sendMessage(String message);

}

**src/main/java**

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public void process() {

api.sendMessage("Hello, Mockito!");

}

}

**src/test/java**

**MyServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

import static org.mockito.ArgumentMatchers.eq;

public class MyServiceTest {

@Test

public void testArgumentMatching() {

ExternalApi mockApi = mock(ExternalApi.class);

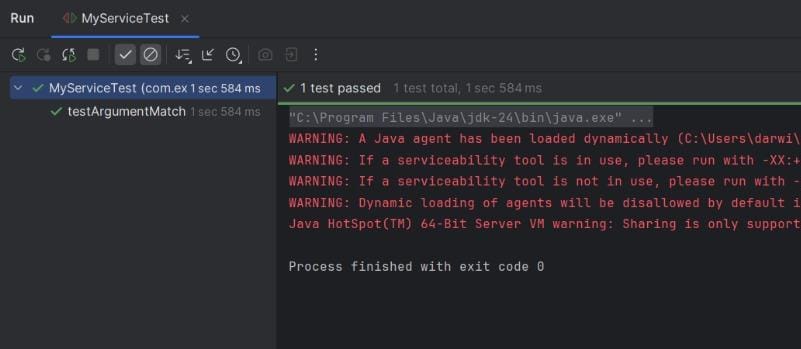
MyService service = new MyService(mockApi);

service.process();

verify(mockApi).sendMessage(eq("Hello, Mockito!"));

}

}

**OUTPUT:**

**Exercise 4: Handling Void Methods**

**CODE:**

**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>org.example</groupId>

<artifactId>HandlingVoidMethods</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>HandlingVoidMethods</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**src/main/java**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

void logMessage(String message);

}

**src/main/java**

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public void doWork() {

api.logMessage("Processing started...");

}

}

**src/test/java**

**MyServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVoidMethodInteraction() {

ExternalApi mockApi = mock(ExternalApi.class);

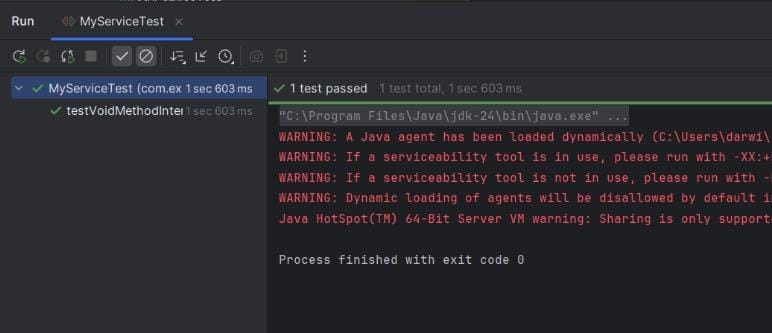
MyService service = new MyService(mockApi);

service.doWork();

verify(mockApi).logMessage("Processing started...");

}

}

**OUTPUT:**

**Exercise 5: Mocking and Stubbing with Multiple Returns**

**CODE:**

**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>org.example</groupId>

<artifactId>MockingAndStubbing</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>MockingAndStubbing</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**src/main/java**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**src/main/java**

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String[] fetchTwice() {

String first = api.getData();

String second = api.getData();

return new String[]{first, second};

}

}

**src/test/java**

**MyServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

public class MyServiceTest {

@Test

public void testMultipleReturns() {

ExternalApi mockApi = mock(ExternalApi.class);

when(mockApi.getData())

.thenReturn("First Call")

.thenReturn("Second Call");

MyService service = new MyService(mockApi);

String[] result = service.fetchTwice();

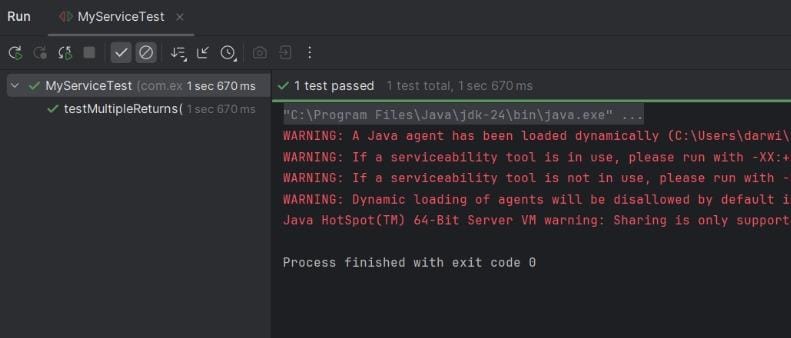
assertEquals("First Call", result[0]);

assertEquals("Second Call", result[1]);

}

}

**OUTPUT:**



**Exercise 6: Verifying Interaction Order**

**CODE:**

**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>org.example</groupId>

<artifactId>VerifyingInteractionOrder</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>VerifyingInteractionOrder</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**src/main/java**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

void stepOne();

void stepTwo();

}

**src/main/java**

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public void executeSteps() {

api.stepOne();

api.stepTwo();

}

}

**src/test/java**

**MyServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

import org.mockito.InOrder;

public class MyServiceTest {

@Test

public void testInteractionOrder() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.executeSteps();

InOrder inOrder = inOrder(mockApi);

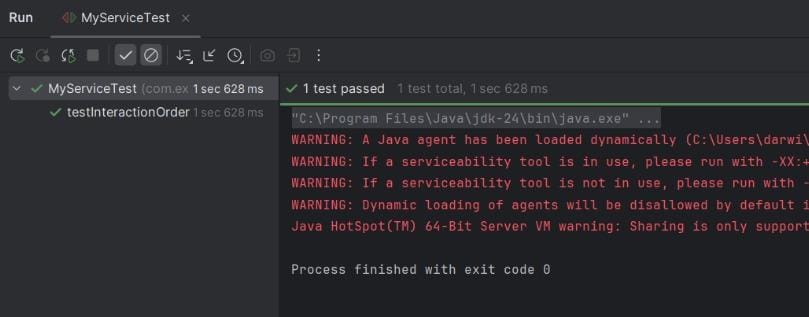
inOrder.verify(mockApi).stepOne();

inOrder.verify(mockApi).stepTwo();

}

}

**OUTPUT:**



**Exercise 7: Handling Void Methods with Exceptions**

**CODE:**

**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>org.example</groupId>

<artifactId>HandlingVoidMethodswithExceptions</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>HandlingVoidMethodswithExceptions</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**src/main/java**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

void deleteRecord(String id);

}

**src/main/java**

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public void remove(String id) {

api.deleteRecord(id); }

}

**src/test/java**

**MyServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

public class MyServiceTest {

@Test

public void testVoidMethodThrowsException() {

ExternalApi mockApi = mock(ExternalApi.class);

doThrow(new RuntimeException("Delete failed"))

.when(mockApi).deleteRecord("123");

MyService service = new MyService(mockApi);

RuntimeException thrown = assertThrows(

RuntimeException.class,

() -> service.remove("123")

);

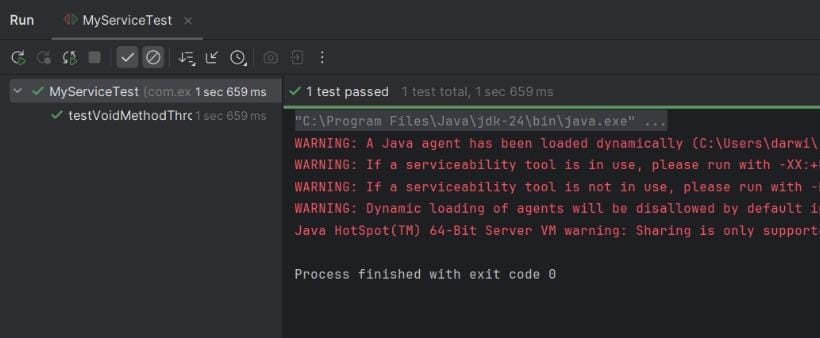
assertEquals("Delete failed", thrown.getMessage());

verify(mockApi).deleteRecord("123");

}

}

**OUTPUT:**



**Logging using SLF4J**

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

**CODE:**

**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>logging-example</artifactId>

<version>1.0-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>

**src/main/java**

**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");

}

}

**src/main/resources**

**logback.xml**

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

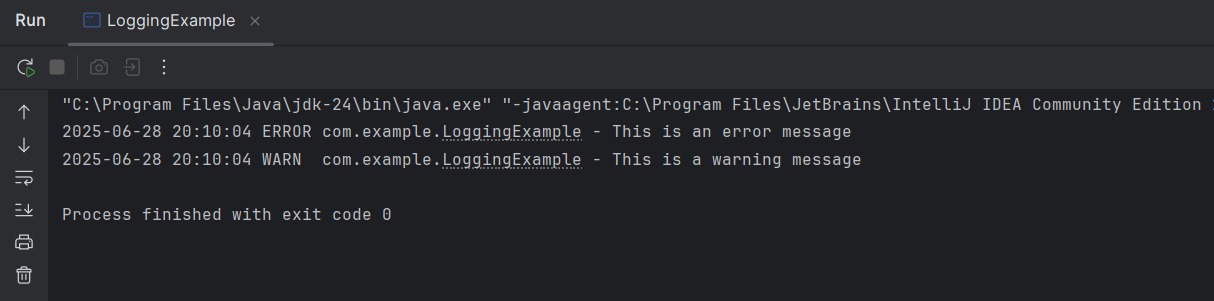
<root level="debug">

<appender-ref ref="STDOUT" />

</root>

</configuration>

**OUTPUT:**



**Exercise 2: Parameterized Logging**

**CODE:**

**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>org.example</groupId>

<artifactId>ParameterizedLogging</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>ParameterizedLogging</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<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>

**src/main/java**

**LoggingParameterizedExample.java**

package com.example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggingParameterizedExample {

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

public static void main(String[] args) {

String username = "john\_doe";

int loginAttempts = 3;

double balance = 1245.75;

logger.info("User {} has attempted to login {} times", username, loginAttempts);

logger.warn("User {} has low balance: ${}", username, balance);

logger.error("Failed login attempt for user: {}", username);

}

}

**src/main/resources**

**logback.xml**

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

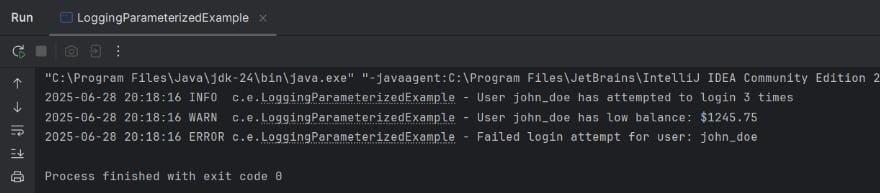
<root level="debug">

<appender-ref ref="STDOUT" />

</root>

</configuration>

**OUTPUT:**



**Exercise 3: Using Different Appenders**

**CODE:**

**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>org.example</groupId>

<artifactId>DifferentAppenders</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>DifferentAppenders</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<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>

**src/main/java**

**LoggingWithAppenders.java**

package com.example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggingWithAppenders {

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

public static void main(String[] args) {

logger.debug("Debug message for development");

logger.info("Information log for system monitoring");

logger.warn("Warning: low memory");

logger.error("Error encountered during operation");

}

}

**src/main/resources**

**logback.xml**

<configuration>

<appender name="console" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<appender name="file" class="ch.qos.logback.core.FileAppender">

<file>app.log</file>

<append>true</append>

<encoder>

<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<root level="debug">

<appender-ref ref="console" />

<appender-ref ref="file" />

</root>

</configuration>

**OUTPUT:**

