**Logging using SLF4J**

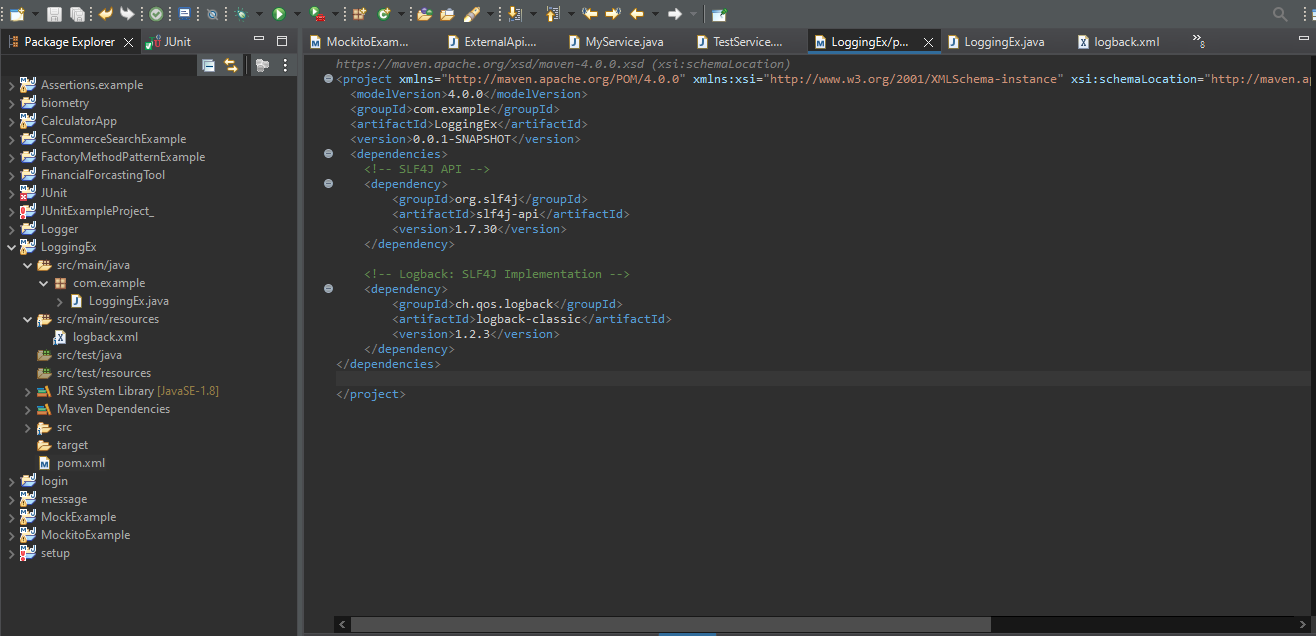
**Exercise 1: Logging Error Messages and Warning Levels Task:**

**SCENARIO :**

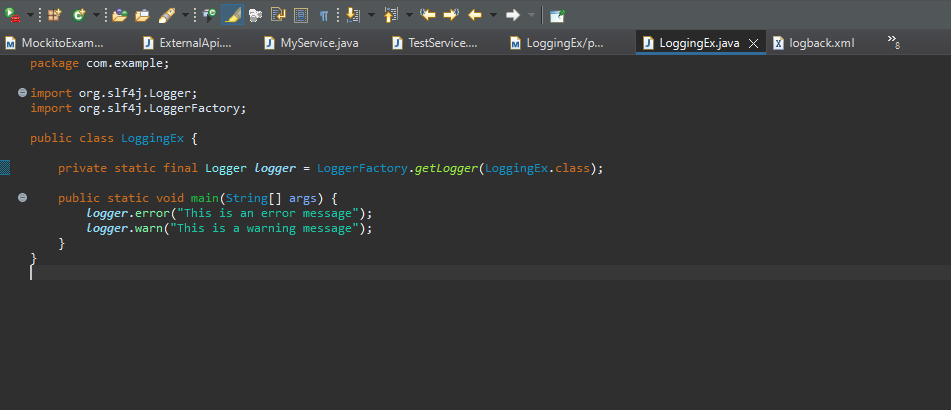
Write a Java application that demonstrates logging error messages and warning levels using SLF4J.

Step-by-Step Solution:

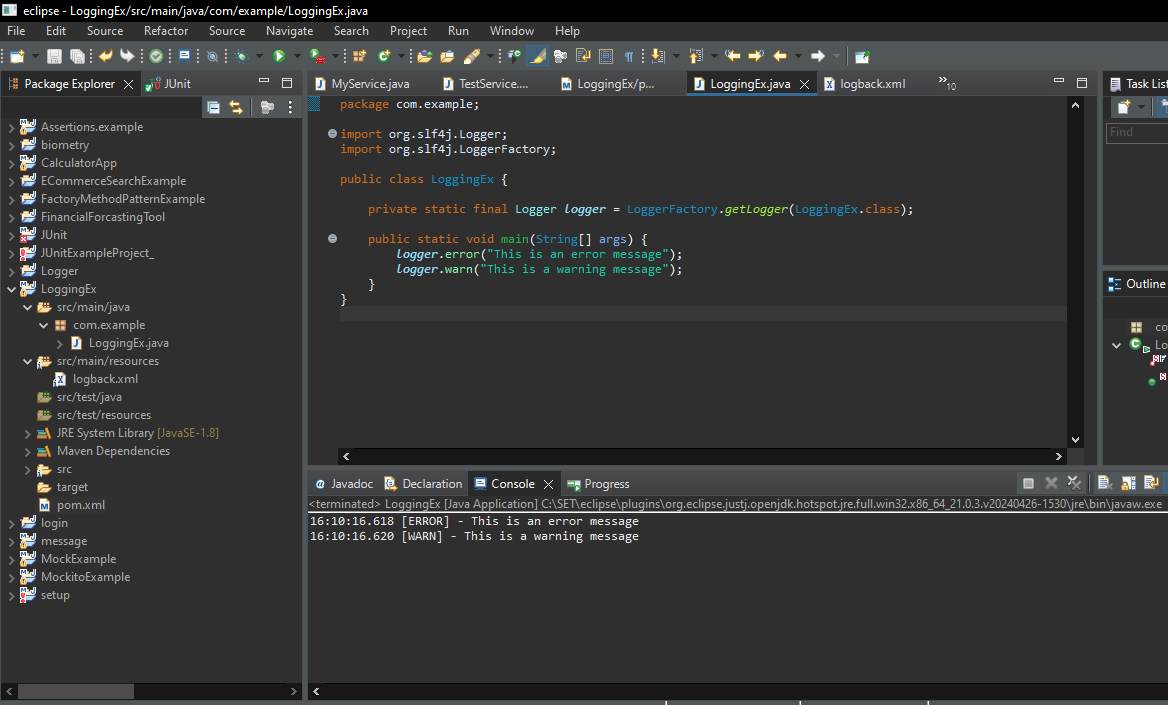
1. Add SLF4J and Logback dependencies to your `pom.xml` file.

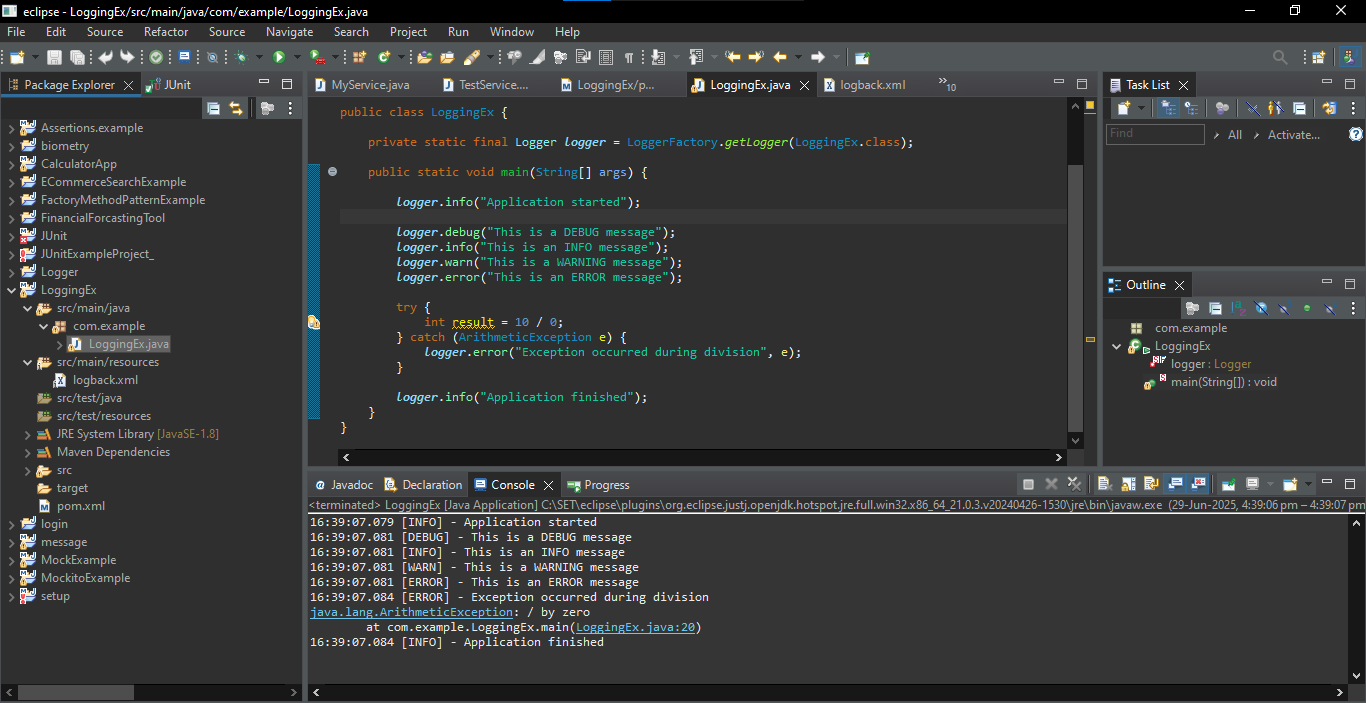


1. Create a Java class that uses SLF4J for logging.



**OUTPUT:**



****

JUnit Testing Exercises

**Exercise 3: Assertions in JUnit**

**Scenario :**

Assertions in JUnit Scenario: You need to use different assertions in JUnit to validate your test results.

**AssertionsTest.java :**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

*@Test*

public void testAssertions() {

// Assert equals

int sum = 2 + 3;

System.***out***.println("Sum of 2 + 3 = " + sum);

*assertEquals*(5, sum);

// Assert true

boolean isGreater = 5 > 3;

System.***out***.println("Is 5 > 3? " + isGreater);

*assertTrue*(isGreater);

// Assert false

boolean isLess = 5 < 3;

System.***out***.println("Is 5 < 3? " + isLess);

*assertFalse*(isLess);

// Assert null

Object obj1 = null;

System.***out***.println("Object 1 is null: " + (obj1 == null));

*assertNull*(obj1);

// Assert not null

Object obj2 = new Object();

System.***out***.println("Object 2 is not null: " + (obj2 != null));

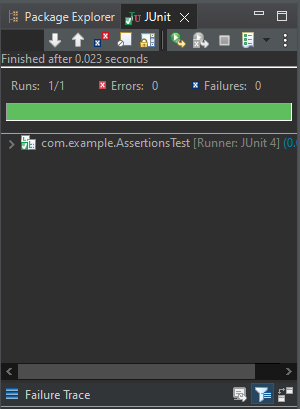
*assertNotNull*(obj2);

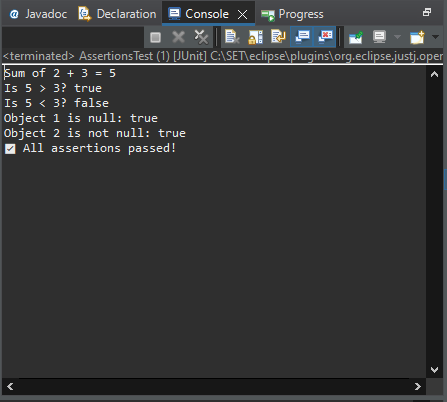
System.***out***.println("All assertions passed!");

}

}

**OUTPUT :**





**JUnit Testing**

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

**Scenario:**

You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

Steps: 1. Write tests using the AAA pattern.

2. Use @Before and @After annotations for setup and teardown methods.

**Calculator.java :**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int sub(int a, int b) {

return a - b;

}

public int mul(int a, int b) {

return a \* b;

}

public int div(int a, int b) {

if (b == 0) throw new ArithmeticException("Division by zero");

return a / b;

}

}

**CalculatorTest.java :**

package com.example;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

*@Before*

public void setUp() {

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

calculator = new Calculator(); // Arrange

}

*@After*

public void tearDown() {

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

calculator = null;

}

*@Test*

public void testAddition() {

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

System.***out***.println("Addition Result: 2 + 3 = " + result);

*assertEquals*(5, result); // Assert

}

*@Test*

public void testSubtraction() {

int result = calculator.sub(10, 4);

System.***out***.println("Subtraction Result: 10 - 4 = " + result);

*assertEquals*(6, result);

}

*@Test*

public void testMultiplication() {

int result = calculator.mul(3, 4);

System.***out***.println("Multiplication Result: 3 \* 4 = " + result);

*assertEquals*(12, result);

}

*@Test*

public void testDivision() {

int result = calculator.div(20, 4);

System.***out***.println("Division Result: 20 / 4 = " + result);

*assertEquals*(5, result);

}

*@Test*(expected = ArithmeticException.class)

public void testDivisionByZero() {

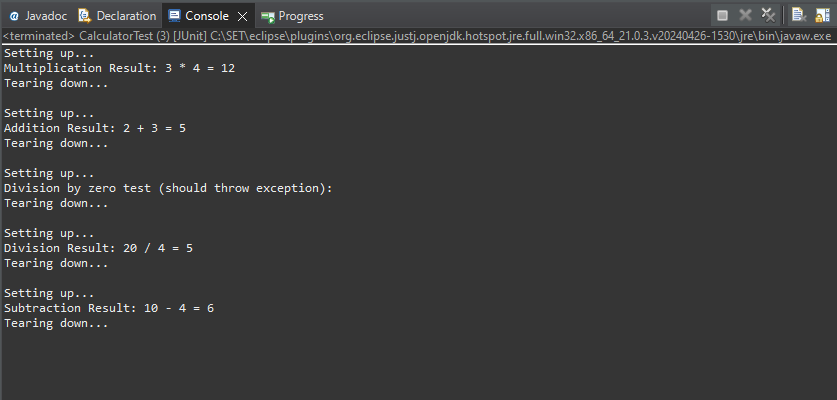
System.***out***.println("Division by zero test (should throw exception):");

calculator.div(10, 0);

}

}

**Output :**

****