**JUnit Testing Exercises**

**Exercise 1: Setting Up JUnit**

<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>junit-exercises</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

<maven.compiler.source>8</maven.compiler.source>

<maven.compiler.target>8</maven.compiler.target>

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

</properties>

<dependencies>

<!-- ✅ JUnit 5 -->

<dependency>

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

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

<version>5.10.2</version>

<scope>test</scope>

</dependency>

<dependency>

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

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

<version>5.10.2</version>

<scope>test</scope>

</dependency>

<!-- ✅ Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.12.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Needed for running JUnit 5 tests -->

<plugin>

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

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

<version>3.2.5</version>

<configuration>

<includes>

<include>\*\*/\*Test.java</include>

</includes>

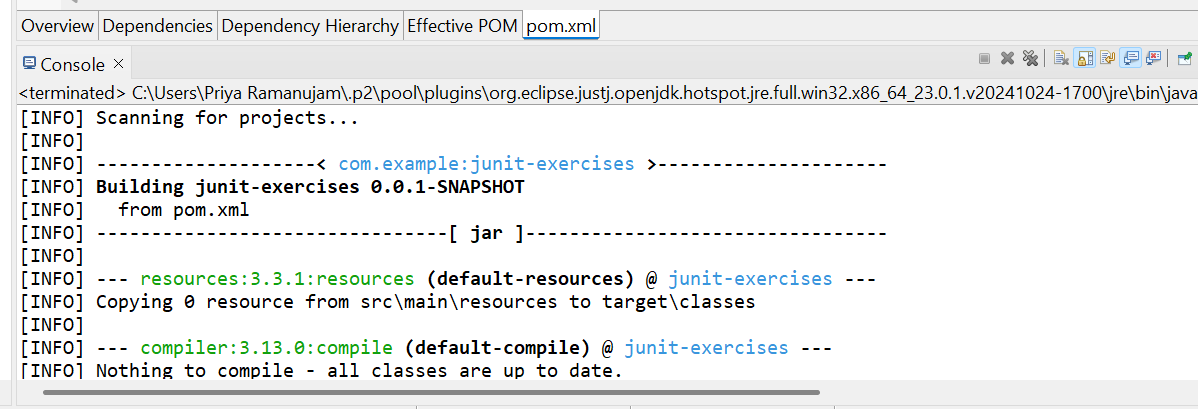
</configuration>

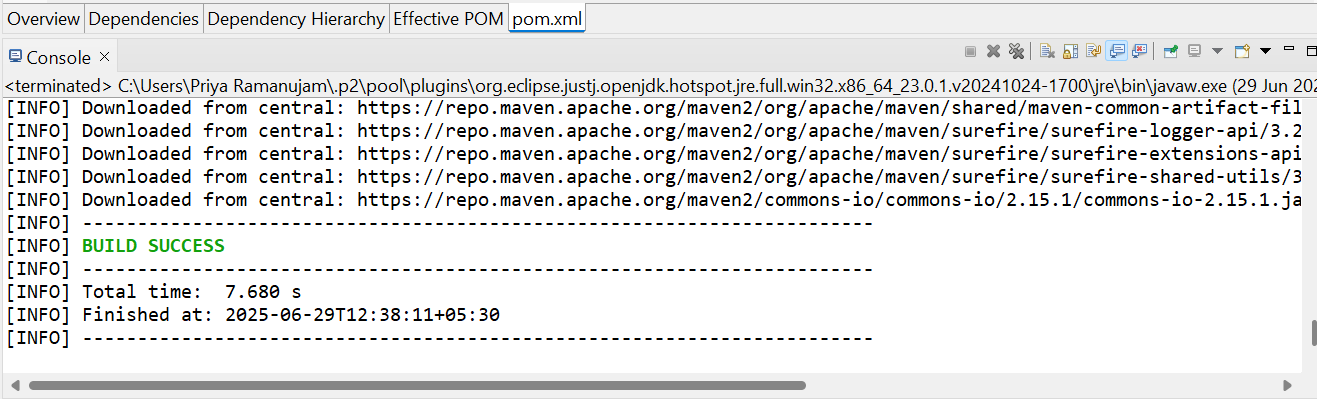
</plugin>

</plugins>

</build>

</project>





**Exercise 2: Writing Basic JUnit Tests**

public class **Calculator** {

**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** **double** divide(**int** a, **int** b) {

**if** (b == 0) {

**throw** **new** IllegalArgumentException("Cannot divide by zero");

}

**return** (**double**) a / b;

}

**public** **boolean** isEven(**int** number) {

**return** number % 2 == 0;

}

**public** **boolean** isPrime(**int** number) {

**if** (number <= 1) {

**return** **false**;

}

**for** (**int** i = 2; i <= Math.*sqrt*(number); i++) {

**if** (number % i == 0) {

**return** **false**;

}

}

**return** **true**;

}

**public** **int** factorial(**int** n) {

**if** (n < 0) {

**throw** **new** IllegalArgumentException("Factorial is not defined for negative numbers");

}

**if** (n == 0 || n == 1) {

**return** 1;

}

**int** result = 1;

**for** (**int** i = 2; i <= n; i++) {

result \*= i;

}

**return** result;

}

**public** String getGrade(**int** score) {

**if** (score < 0 || score > 100) {

**throw** **new** IllegalArgumentException("Score must be between 0 and 100");

}

**if** (score >= 90) **return** "A";

**if** (score >= 80) **return** "B";

**if** (score >= 70) **return** "C";

**if** (score >= 60) **return** "D";

**return** "F";

}

}

**import** org.junit.Test;

**import** org.junit.Before;

**import** org.junit.After;

**import** **static** org.junit.Assert.\*;

**public** **class** CalculatorTest {

**private** Calculator calculator;

@Before

**public** **void** setUp() {

calculator = **new** Calculator();

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

}

@After

**public** **void** tearDown() {

calculator = **null**;

System.***out***.println("Cleaning up test");

}

@Test

**public** **void** testAddPositiveNumbers() {

**int** a = 5;

**int** b = 3;

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

*assertEquals*(8, result);

}

@Test

**public** **void** testAddNegativeNumbers() {

**int** a = -5;

**int** b = -3;

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

*assertEquals*(-8, result);

}

@Test

**public** **void** testAddZero() {

**int** a = 5;

**int** b = 0;

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

*assertEquals*(5, result);

}

@Test

**public** **void** testSubtract() {

**int** a = 10;

**int** b = 4;

**int** result = calculator.subtract(a, b);

*assertEquals*(6, result);

}

@Test

**public** **void** testSubtractNegativeResult() {

**int** a = 3;

**int** b = 8;

**int** result = calculator.subtract(a, b);

*assertEquals*(-5, result);

}

@Test

**public** **void** testMultiply() {

**int** a = 4;

**int** b = 5;

**int** result = calculator.multiply(a, b);

a*ssertEquals*(20, result);

}

@Test

**public** **void** testMultiplyByZero() {

**int** a = 10;

**int** b = 0;

**int** result = calculator.multiply(a, b);

*assertEquals*(0, result);

}

@Test

**public** **void** testDivide() {

**int** a = 10;

**int** b = 2;

**double** result = calculator.divide(a, b);

*assertEquals*(5.0, result, 0.001); // delta for double comparison

}

@Test(expected = IllegalArgumentException.**class**)

**public** **void** testDivideByZero() {

**int** a = 10;

**int** b = 0;

calculator.divide(a, b);

}

@Test

**public** **void** testIsEvenWithEvenNumber() {

**int** number = 4;

**boolean** result = calculator.isEven(number);

*assertTrue*(result);

}

@Test

**public** **void** testIsEvenWithOddNumber() {

**int** number = 5;

**boolean** result = calculator.isEven(number);

*assertFalse*(result);

}

@Test

**public** **void** testIsPrimeWithPrimeNumber() {

**int** number = 7;

**boolean** result = calculator.isPrime(number);

*assertTrue*(result);

}

@Test

**public** **void** testIsPrimeWithNonPrimeNumber() {

**int** number = 8;

**boolean** result = calculator.isPrime(number);

*assertFalse*(result);

}

@Test

**public** **void** testIsPrimeWithOne() {

**int** number = 1;

**boolean** result = calculator.isPrime(number);

*assertFalse*(result);

}

@Test

**public** **void** testFactorial() {

**int** number = 5;

**int** result = calculator.factorial(number);

*assertEquals*(120, result);

}

@Test

**public** **void** testFactorialOfZero() {

**int** number = 0;

**int** result = calculator.factorial(number);

*assertEquals*(1, result);

}

@Test(expected = IllegalArgumentException.**class**)

**public** **void** testFactorialOfNegativeNumber() {

**int** number = -5;

calculator.factorial(number);

}

@Test

**public** **void** testGetGradeA() {

**int** score = 95;

String result = calculator.getGrade(score);

*assertEquals*("A", result);

}

@Test

**public** **void** testGetGradeB() {

**int** score = 85;

String result = calculator.getGrade(score);

*assertEquals*("B", result);

}

@Test

**public** **void** testGetGradeF() {

**int** score = 45;

String result = calculator.getGrade(score);

*assertEquals*("F", result);

}

@Test(expected = IllegalArgumentException.**class**)

**public** **void** testGetGradeInvalidScore() {

**int** score = 105;

calculator.getGrade(score);

}

@Test

**public** **void** testDifferentAssertions() {

*assertEquals*(5, 2 + 3);

*assertTrue*(5 > 3);

*assertFalse*(5 < 3);

*assertNull*(**null**);

*assertNotNull*(**new** Object());

String str1 = "Hello";

String str2 = "Hello";

String str3 = **new** String("Hello");

*assertEquals*(str1, str2); // same content

*assertNotSame*(str1, str3); // different objects

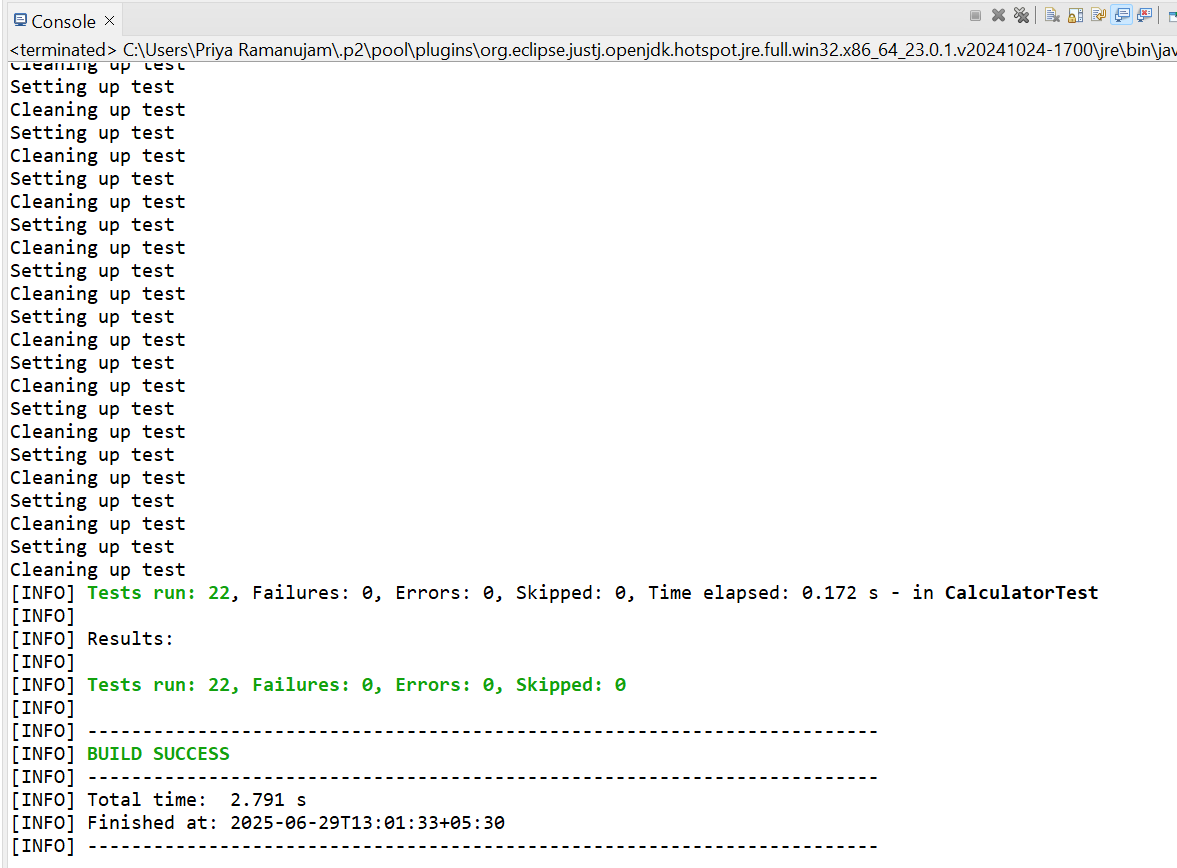
**int**[] expected = {1, 2, 3};

**int**[] actual = {1, 2, 3};

*assertArrayEquals*(expected, actual);

}

}



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

**public** **class** Calculator {

/\*\*

\* Adds two numbers

\* **@param** a first number

\* **@param** b second number

\* **@return** sum of a and b

\*/

**public** **double** add(**double** a, **double** b) {

**return** a + b;

}

/\*\*

\* Subtracts second number from first number

\* **@param** a minuend (number to subtract from)

\* **@param** b subtrahend (number to subtract)

\* **@return** result of a - b

\*/

**public** **double** subtract(**double** a, **double** b) {

**return** a - b;

}

/\*\*

\* Multiplies two numbers

\* **@param** a first number

\* **@param** b second number

\* **@return** product of a and b

\*/

**public** **double** multiply(**double** a, **double** b) {

**return** a \* b;

}

/\*\*

\* Divides first number by second number

\* **@param** a dividend (number to be divided)

\* **@param** b divisor (number to divide by)

\* **@return** quotient of a / b

\* **@throws** ArithmeticException if divisor is zero

\*/

**public** **double** divide(**double** a, **double** b) {

**if** (b == 0.0) {

**throw** **new** ArithmeticException("Cannot divide by zero");

}

**return** a / b;

}

}

**import** org.junit.\*;

**import** **static** org.junit.Assert.\*;

**public** **class** CalculatorTest {

**private** Calculator calculator;

**private** **final** **double** DELTA = 0.001;

@Before

**public** **void** setUp() {

System.***out***.println("Setting up Calculator test fixture...");

calculator = **new** Calculator();

}

@After

**public** **void** tearDown() {

System.***out***.println("Cleaning up Calculator test fixture...");

calculator = **null**;

}

@BeforeClass

**public** **static** **void** setUpClass() {

System.***out***.println("=== Starting Calculator Test Suite ===");

}

@AfterClass

**public** **static** **void** tearDownClass() {

System.***out***.println("=== Finished Calculator Test Suite ===");

}

@Test

**public** **void** testAddPositiveNumbers() {

**double** firstNumber = 10.5;

**double** secondNumber = 5.3;

**double** expectedResult = 15.8;

**double** actualResult = calculator.add(firstNumber, secondNumber);

*assertEquals*("Addition of positive numbers should be correct",

expectedResult, actualResult, DELTA);

}

@Test

**public** **void** testAddNegativeNumbers() {

**double** firstNumber = -7.2;

**double** secondNumber = -3.8;

**double** expectedResult = -11.0;

**double** actualResult = calculator.add(firstNumber, secondNumber);

*assertEquals*("Addition of negative numbers should be correct",

expectedResult, actualResult, DELTA);

}

@Test

**public** **void** testSubtract() {

**double** minuend = 20.0;

**double** subtrahend = 8.5;

**double** expectedResult = 11.5;

**double** actualResult = calculator.subtract(minuend, subtrahend);

*assertEquals*("Subtraction should be correct",

expectedResult, actualResult, DELTA);

}

@Test

**public** **void** testMultiply() {

**double** multiplicand = 4.0;

**double** multiplier = 2.5;

**double** expectedResult = 10.0;

**double** actualResult = calculator.multiply(multiplicand, multiplier);

*assertEquals*("Multiplication should be correct",

expectedResult, actualResult, DELTA);

}

@Test

**public** **void** testDivideValidNumbers() {

**double** dividend = 15.0;

**double** divisor = 3.0;

**double** expectedResult = 5.0;

**double** actualResult = calculator.divide(dividend, divisor);

*assertEquals*("Division should be correct",

expectedResult, actualResult, DELTA);

}

@Test(expected = ArithmeticException.**class**)

**public** **void** testDivideByZero() {

**double** dividend = 10.0;

**double** divisor = 0.0;

calculator.divide(dividend, divisor);

}

@Test

**public** **void** testMultipleOperations() {

**double** a = 10.0;

**double** b = 5.0;

**double** c = 2.0;

**double** expectedResult = 30.0;

**double** sum = calculator.add(a, b);

**double** finalResult = calculator.multiply(sum, c);

*assertEquals*("Multiple operations should work correctly",

expectedResult, finalResult, DELTA);

}

@Test

**public** **void** testZeroOperations() {

**double** zero = 0.0;

**double** number = 42.0;

*assertEquals*("Adding zero should not change the number",

number, calculator.add(number, zero), DELTA);

*assertEquals*("Subtracting zero should not change the number",

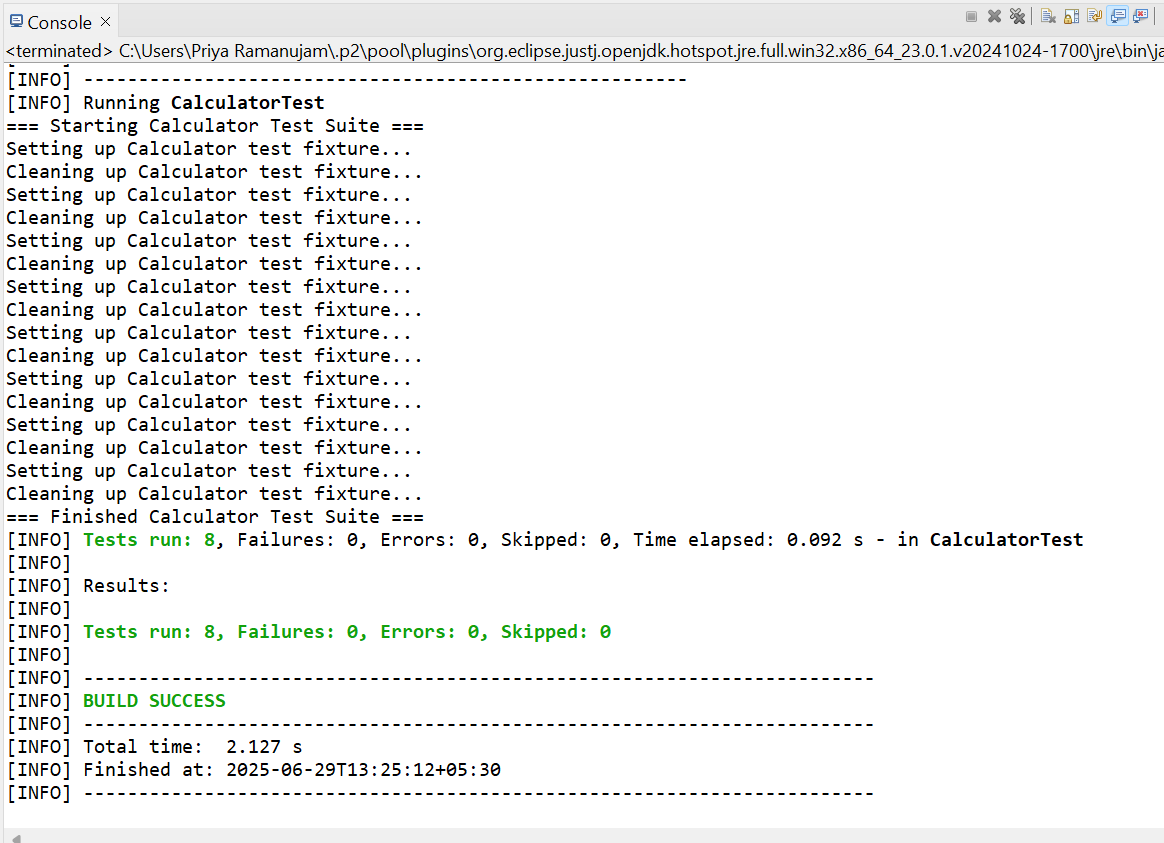
number, calculator.subtract(number, zero), DELTA);

*assertEquals*("Multiplying by zero should result in zero",

0.0, calculator.multiply(number, zero), DELTA);

}

}

****