TDD using JUnit5 and Mockito

**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:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

3. Create a new test class in your project

// 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;

}

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;

}

public int modulus(int a, int b) {

return a % b;

}

public int square(int a) {

return a \* a;

}

public int cube(int a) {

return a \* a \* a;

}

}

// CalculatorTest.java

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

Calculator calc = new Calculator();

@Test

public void testAdd() {

assertEquals(8, calc.add(5, 3));

}

@Test

public void testSubtract() {

assertEquals(2, calc.subtract(5, 3));

}

@Test

public void testMultiply() {

assertEquals(15, calc.multiply(5, 3));

}

@Test

public void testDivide() {

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

}

@Test(expected = IllegalArgumentException.class)

public void testDivideByZero() {

calc.divide(10, 0);

}

@Test

public void testModulus() {

assertEquals(1, calc.modulus(10, 3));

}

@Test

public void testSquare() {

assertEquals(16, calc.square(4));

}

@Test

public void testCube() {

assertEquals(27, calc.cube(3));

}

}

**Exercise 3: Assertions in JUnit**

Scenario:

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

Steps: 1. Write tests using various JUnit assertions.

Solution Code:

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());

}

}

**Ans:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 3);

assertFalse(5 < 3);

assertNull(null);

assertNotNull(new Object());

String s = "hello";

String t = s;

assertSame(s, t);

assertNotSame(new String("abc"), new String("abc"));

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**

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

**Ans:**

package calculator.ex;

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 int divide(int a, int b) {

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

return a / b;

}

}

package calculator.ex;

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() {

calc = new Calculator();

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

}

@After

public void tearDown() {

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

}

@Test

public void testAddition() {

int a = 5;

int b = 3;

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

assertEquals(8, result);

}

@Test

public void testSubtraction() {

int a = 10;

int b = 4;

int result = calc.subtract(a, b);

assertEquals(6, result);

}

@Test

public void testMultiplication() {

int x = 6;

int y = 7;

int product = calc.multiply(x, y);

assertEquals(42, product);

}

@Test(expected = IllegalArgumentException.class)

public void testDivideByZero() {

int dividend = 10;

int divisor = 0;

calc.divide(dividend, divisor);

}

@Test

public void testDivision() {

int dividend = 20;

int divisor = 4;

int quotient = calc.divide(dividend, divisor);

assertEquals(5, quotient);

}

}