**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: junit junit 4.13.2 test Answer

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;

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

import static org.junit.Assert.\*;

import org.junit.Test;

class Calculator {

public int add(int a, int b) {

return a + b;

}

public boolean isEven(int num) {

return num % 2 == 0;

}

}

public class Main {

@Test

public void testAddition() {

Calculator calc = new Calculator();

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

assertEquals(0, calc.add(2, -2));

}

@Test

public void testEvenNumbers() {

Calculator calc = new Calculator();

assertTrue(calc.isEven(2));

assertTrue(calc.isEven(100));

}

@Test

public void testOddNumbers() {

Calculator calc = new Calculator();

assertFalse(calc.isEven(3));

assertFalse(calc.isEven(101));

}

public static void main(String[] args) {

Main test = new Main();

test.testAddition();

test.testEvenNumbers();

test.testOddNumbers();

System.out.println("All tests passed");

}

}

**Output:**

All tests passed

**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());

}

}

**Answer :**

import static org.junit.Assert.\*;

import org.junit.Test;

public class Main {

@Test

public void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 3);

assertFalse(5 < 3);

assertNull(null);

assertNotNull(new Object());

}

public static void main(String[] args) {

Main test = new Main();

test.testAssertions();

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

}

}

**Output:**

All assertions passed

**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.

**Answer:**

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

class Calculator {

public int multiply(int a, int b) {

return a \* b;

}

}

public class Main {

Calculator calc;

@Before

public void setUp() {

calc = new Calculator();

}

@After

public void tearDown() {

calc = null;

}

@Test

public void testMultiplyPositive() {

int result = calc.multiply(3, 4);

assertEquals(12, result);

}

@Test

public void testMultiplyWithZero() {

int result = calc.multiply(5, 0);

assertEquals(0, result);

}

public static void main(String[] args) {

Main test = new Main();

test.setUp();

test.testMultiplyPositive();

test.tearDown();

test.setUp();

test.testMultiplyWithZero();

test.tearDown();

System.out.println("All AAA pattern tests passed with setup and teardown");

}

}

**Output :**

All AAA pattern tests passed with setup and teardown