WEEK 2

**1.JUnit Testing Exercises:**

1.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**.**

**Solution:**

**Step 1: Create a New Java Project**

* Open your IDE (such as IntelliJ IDEA or Eclipse).
* Create a new Java project and set up the required directory structure.

**Step 2: Add JUnit Dependency**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

**Step 3: Create a Sample Java Class**

Create a simple class named Calculator.java:

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**Step 4: Create a JUnit Test Class**

Now, create a test class named CalculatorTest.java in the src/test/java folder:

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAddition() {

Calculator calc = new Calculator();

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

assertEquals(5, result);

}

}

**Sample Input:**

calc.add(2, 3);

**Output:**

**A white background with black text

AI-generated content may be incorrect.**

**2.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());

}

}

**Solution:**

**Solution Code:**

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

import org.junit.jupiter.api.Test;

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());

}

}

**Output:**

A black background with white text

AI-generated content may be incorrect.

3.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.

**Solution:**

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

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

public class AAATest {

private int number;

@BeforeEach

public void setUp() {

// Arrange: Initialize test data

number = 5;

System.out.println("Setup: number initialized to 5");

}

@AfterEach

public void tearDown() {

// Cleanup resources if needed

System.out.println("Teardown: Test completed");

}

@Test

public void testAddition() {

// Act: Perform action

int result = number + 5;

// Assert: Verify result

assertEquals(10, result);

}

@Test

public void testSubtraction() {

// Act: Perform action

int result = number - 2;

// Assert: Verify result

assertEquals(3, result);

}

}

**Output:**

