Exercise 1: Setting Up Junit

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:

import org.junit.Test;

import static org.junit.Assert.\*;

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:

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

package com.example;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AdvancedCalculatorTest {

private Calculator calc;

// Setup: runs before each test

@Before

public void setUp() {

calc = new Calculator();

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

}

// Teardown: runs after each test

@After

public void tearDown() {

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

}

// Arrange-Act-Assert Pattern

@Test

public void testAdd\_AAA() {

// Arrange

int a = 4, b = 6;

// Act

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

// Assert

assertEquals(10, result);

}

}