**JUnit Testing Exercises**

Exercise 1: **Setting Up JUnit**

***Scenario:***

You need to set up J Unit in your Java project to start writing unit tests.

CODE:

* Calculator.java

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

* CalculatorTest.java

import org.junit.jupiter.api.Test;

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

public class CalculatorTest {

*@Test*

void testAdd() {

Calculator calc = new Calculator();

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

*assertEquals*(5, result);

}

*@Test*

void testSubtract() {

Calculator calc = new Calculator();

int result = calc.subtract(5, 3);

*assertEquals*(2, result);

}

}

* pom.xml

<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>JUnitExampleProject</groupId>

<artifactId>JUnitExampleProject</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<!-- JUnit 5 Jupiter API and Engine -->

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.13.0</version>

<configuration>

<release>21</release>

</configuration>

</plugin>

<plugin>

<artifactId>maven-war-plugin</artifactId>

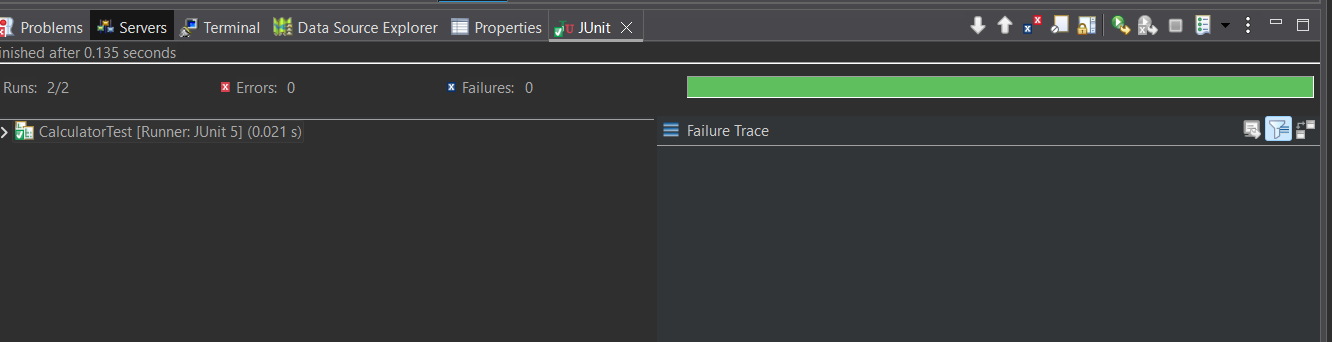
<version>3.2.3</version>

</plugin>

</plugins>

</build>

</project>

**Output:**

Exercise 3: **Assertions in Junit**

***Scenario:***

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

CODE:

* AssertionsTest.java

import org.junit.jupiter.api.Test;

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

public class AssertionsTest {

*@Test*

void testAssertions() {

// Assert equals

*assertEquals*(5, 2 + 3);

// Assert true

*assertTrue*(5 > 3);

// Assert false

*assertFalse*(5 < 3);

// Assert null

*assertNull*(null);

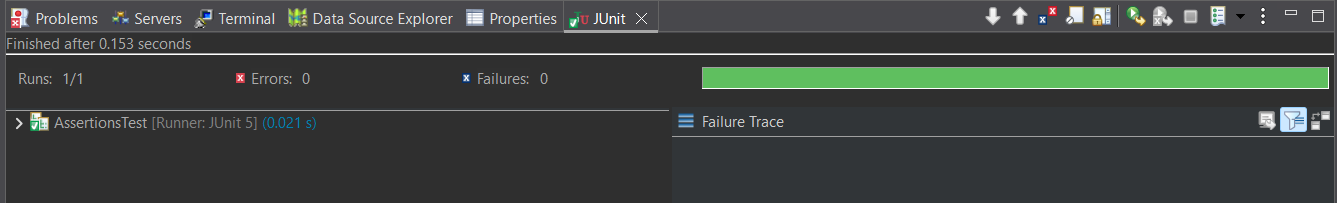
// Assert not null

*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.

CODE:

* Calculator.java

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

* CalculatorTest.java

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

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

public class CalculatorTest {

private Calculator calculator;

// Setup method – runs before each test

*@BeforeEach*

void setUp() {

calculator = new Calculator();

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

}

// Teardown method – runs after each test

*@AfterEach*

void tearDown() {

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

}

*@Test*

void testAdd() {

// Arrange

int a = 2;

int b = 3;

// Act

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

// Assert

*assertEquals*(5, result);

}

*@Test*

void testSubtract() {

// Arrange

int a = 5;

int b = 2;

// Act

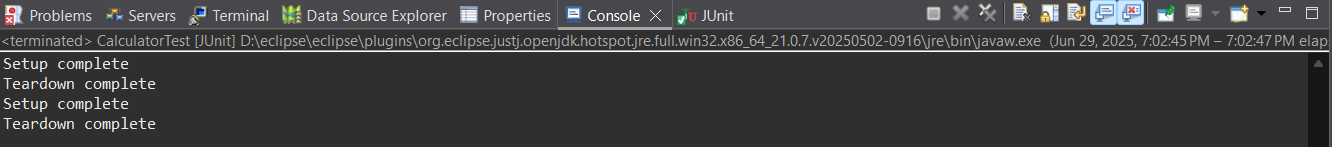
int result = calculator.subtract(a, b);

// Assert

*assertEquals*(3, result);

}

}

OUTPUT: