**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>

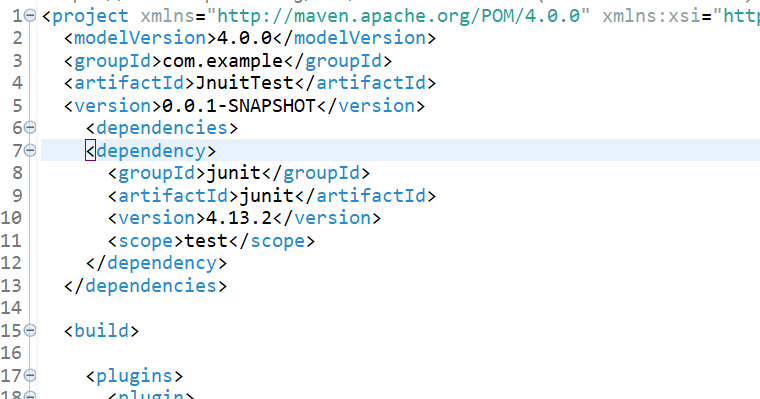
</dependency>

3. Create a new test class in your project.

**Project Structure:**



**pom.xml:**



**Test Class:**

**package** test.java;

import static org.junit.Assert.*assertEquals*;

import org.junit.Test;

public class SampleTest {

@Test

public void testAddition() {

int a = 29;

int b = 32;

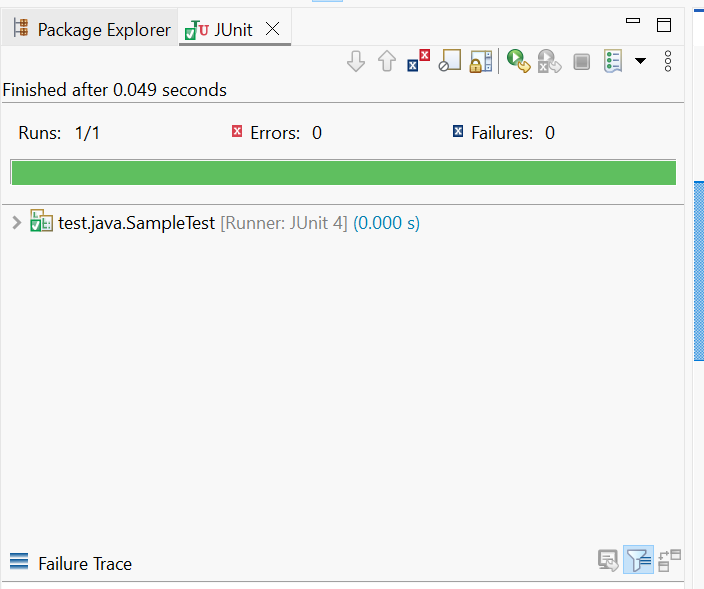
int expectedSum = 61;

*assertEquals*(expectedSum, a + b);

}

}

**Output:**



**2) Exercise 2: Writing Basic JUnit Tests**

**Scenario:**

You need to write basic JUnit tests for a simple Java class.

**Steps:**

1. Create a new Java class with some methods to test.

2. Write JUnit tests for these methods.

**Code:**

**Math Class:**

package com.example.math;

public class Math {

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("Division by zero");

return a / b;

}

}

**MathTest Class:**

**package** com.example.math;

**import** **static** org.junit.Assert.\*;

**import** org.junit.Test;

**public** **class** MathTest {

Math math = **new** Math();

@Test

**public** **void** testAdd() {

*assertEquals*(2, math.add(2, 0));

}

@Test

**public** **void** testSubtract() {

*assertEquals*(1, math.subtract(5, 4));

}

@Test

**public** **void** testMultiply() {

*assertEquals*(20, math.multiply(4, 5));

}

@Test

**public** **void** testDivide() {

*assertEquals*(2, math.divide(10, 5));

}

@Test(expected = IllegalArgumentException.**class**)

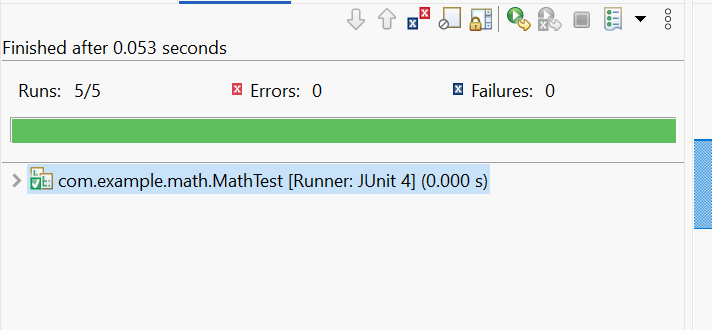
**public** **void** testDivideByZero() {

math.divide(5, 0);

}

}

**Output:**

****

**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());

}

}

**Code :**

**Assertions Class:**

package com.example.calculator;

public class Assertions {

public int add(int a, int b) {

return a + b;

}

public boolean isGreater(int a, int b) {

return a > b;

}

public boolean isLess(int a, int b) {

return a < b;

}

public Object getNullObject() {

return null;

}

public Object getNotNullObject() {

return new Object();

}

}

**AssertionsTest Class:**

**package** com.example.calculator;

**import** **static** org.junit.Assert.\*;

**import** org.junit.Test;

**public** **class** AssertionsTest {

Assertions calc = **new** Assertions();

@Test

**public** **void** testAdd() {

*assertEquals*("2 + 3 should equal 5", 5, calc.add(2, 3));

}

@Test

**public** **void** testIsGreater() {

*assertTrue*("5 should be greater than 3", calc.isGreater(5, 3));

}

@Test

**public** **void** testIsLess() {

*assertFalse*("5 should not be less than 3", calc.isLess(5, 3));

}

@Test

**public** **void** testGetNullObject() {

*assertNull*("Should return null", calc.getNullObject());

}

@Test

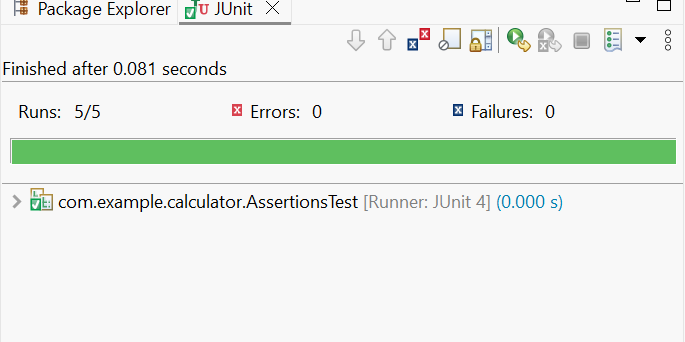
**public** **void** testGetNotNullObject() {

*assertNotNull*("Should not return null", calc.getNotNullObject());

}

}

**Output:**

****

**4) 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.

**Code:**

**Math Class:**

**package** com.example.math;

**public** **class** Math {

**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("Division by zero");

**return** a / b;

}

}

**MathTestAAA Class:**

**package** com.example.math;

**import** **static** org.junit.Assert.\*;

**import** org.junit.Before;

**import** org.junit.After;

**import** org.junit.Test;

**public** **class** MathTestAAA {

**private** Math math;

@Before

**public** **void** setUp() {

// Arrange

math = **new** Math();

System.***out***.println("Setup: New SimpleMath created.");

}

@After

**public** **void** tearDown() {

math = **null**;

System.***out***.println("Teardown: SimpleMath reference cleared.");

}

@Test

**public** **void** testAdd() {

// Arrange

**int** a = 2;

**int** b = 3;

// Act

**int** result = math.add(a, b);

// Assert

*assertEquals*("2 + 3 should equal 5", 5, result);

}

@Test

**public** **void** testSubtract() {

// Arrange

**int** a = 10;

**int** b = 4;

// Act

**int** result = math.subtract(a, b);

// Assert

*assertEquals*("10 - 4 should equal 6", 6, result);

}

}

**Output:**

