JUnit Testing Exercises:

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

3. Create a new test class in your project.

JUnit Dependency to pom.xml:

Code:

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>junit-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- JUnit 4 Dependency -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

Calculator:

Code:

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

Test Class:

package com.example;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

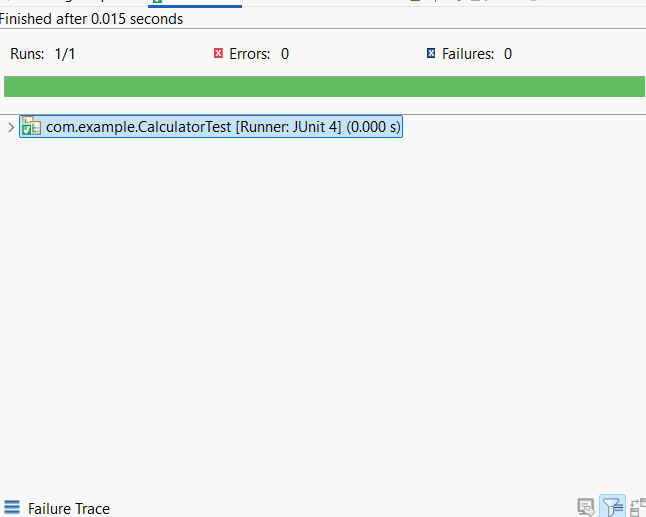
int result = calc.add(2, 3);

assertEquals(5, result); // Test should pass

}

}

Output:



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:

package com.example;

public class MathUtil {

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) {

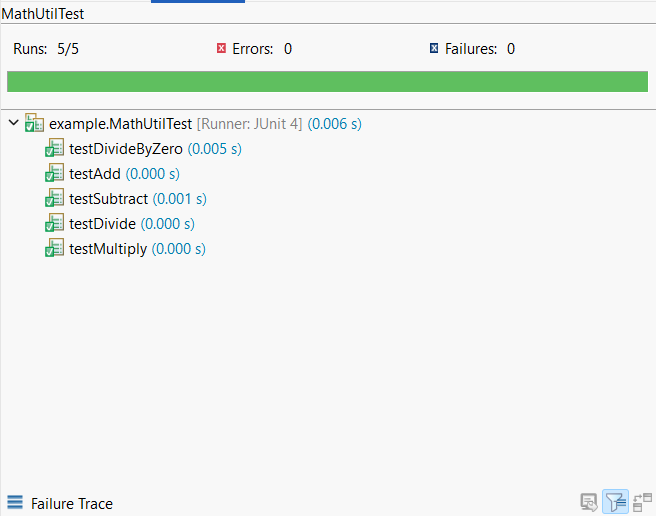
if (b == 0) throw new ArithmeticException("Cannot divide by zero");

return a / b;

}

}

Output:



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.
2. Solution Code:
3. public class AssertionsTest {
4. @Test public void testAssertions()
5. { // Assert equals assertEquals(5, 2 + 3);
6. // Assert true assertTrue(5 > 3);
7. // Assert false assertFalse(5 < 3);
8. // Assert null assertNull(null);
9. // Assert not null assertNotNull(new Object());
10. }
11. }

Code:

package com.example;

import static org.junit.Assert.\*;

import org.junit.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);

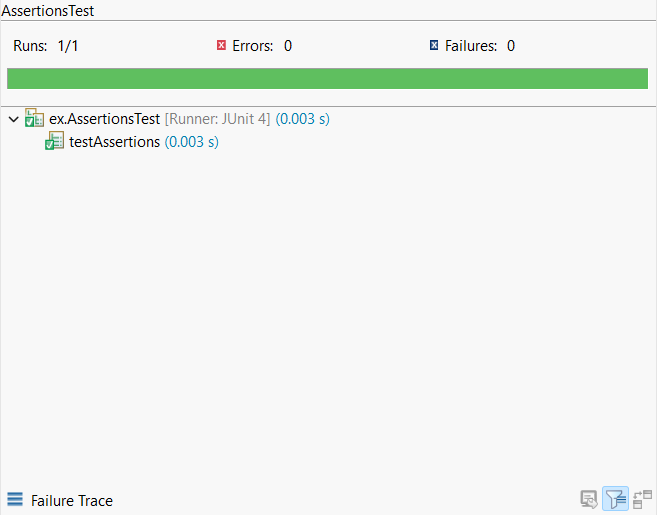
// Assert not null

assertNotNull(new Object());

}

}

Output:



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:

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

Use AAA Pattern with Setup and Teardown:

package com.example;

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

public class CalculatorTest {

private Calculator calc;

// Setup method – runs before each test

@Before

public void setUp() {

System.out.println("Setting up Calculator object...");

calc = new Calculator();

}

// Teardown method – runs after each test

@After

public void tearDown() {

System.out.println("Cleaning up after test...");

calc = null;

}

@Test

public void testAdd() {

// Arrange: (Already done in setUp)

int a = 5;

int b = 3;

// Act

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

// Assert

assertEquals(8, result);

}

@Test

public void testSubtract() {

// Arrange

int a = 10;

int b = 4;

// Act

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

// Assert

assertEquals(6, result);

}

}

Output:

