1. **Junit\_Basic\_Testing Exercises**

**Exercise-1 Setting Up JUnit**

**junitdemo/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>com.example</groupId>

<artifactId>junitdemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

**Calculator.java**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}

}

**Calculatortest.java**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

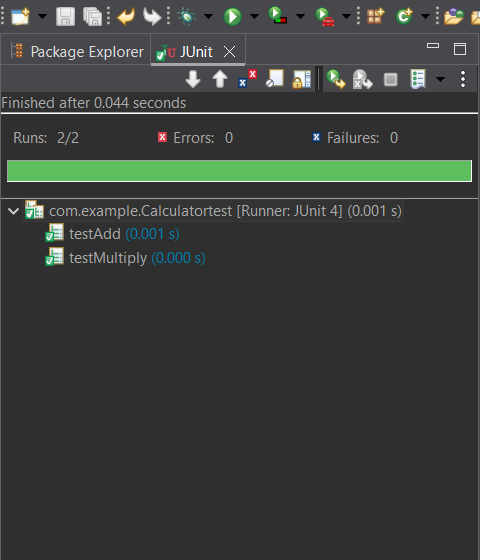
public int multiply(int a, int b) {

return a \* b;

}

}

**Output:**

****

**Exercise-2 Writing Basic Junit Tests**

**MathUtils.java**

package com.example.basic;

public class MathUtils {

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("Cannot divide by zero.");

}

return a / b;

}

}

**MathUtilsTest.java**

package com.example.basic;

import org.junit.Test;

import static org.junit.Assert.\*;

public class MathUtilsTest {

*@Test*

public void testAdd() {

MathUtils mu = new MathUtils();

*assertEquals*(10, mu.add(7, 3));

}

*@Test*

public void testSubtract() {

MathUtils mu = new MathUtils();

*assertEquals*(4, mu.subtract(7, 3));

}

*@Test*

public void testMultiply() {

MathUtils mu = new MathUtils();

*assertEquals*(21, mu.multiply(7, 3));

}

*@Test*

public void testDivide() {

MathUtils mu = new MathUtils();

*assertEquals*(2, mu.divide(6, 3));

}

*@Test*(expected = IllegalArgumentException.class)

public void testDivideByZero() {

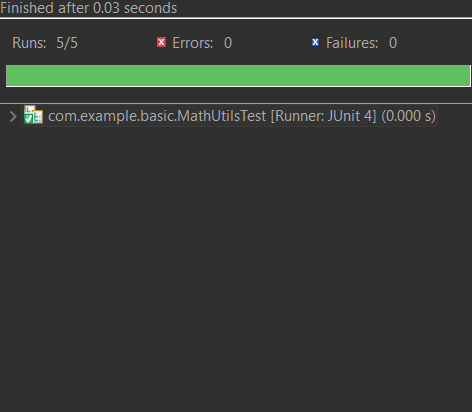
MathUtils mu = new MathUtils();

mu.divide(5, 0);

}

}

**Output:**

****

**Exercise-3 Assertions in JUnit**

**AssertionsTest.java**

package com.example.basic;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

*@Test*

public void testAssertions() {

// assertEquals(expected, actual)

*assertEquals*(5, 2 + 3);

// assertTrue(condition)

*assertTrue*(5 > 3);

// assertFalse(condition)

*assertFalse*(5 < 3);

// assertNull(object)

Object obj = null;

*assertNull*(obj);

// assertNotNull(object)

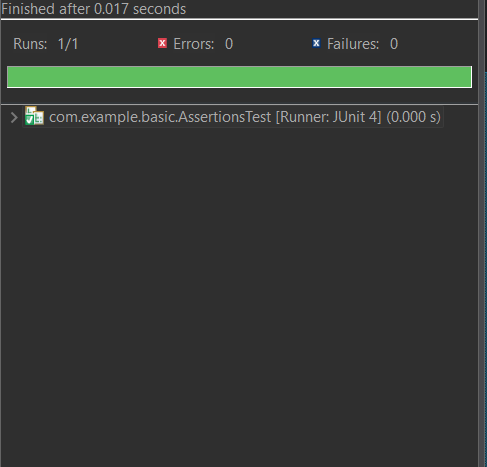
Object notNullObj = new Object();

*assertNotNull*(notNullObj);

}

}

**Output:**

****

**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

**MathUtils.java**

package com.example.basic;

public class MathUtils {

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("Cannot divide by zero.");

}

return a / b;

}

}

**MathUtilsLifeCycleTest.java**

package com.example.basic;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class MathUtilsLifeCycleTest {

private MathUtils mathUtils;

// Setup: runs BEFORE each @Test

*@Before*

public void setUp() {

mathUtils = new MathUtils(); // Arrange

System.***out***.println("Setup: Creating MathUtils instance");

}

// Teardown: runs AFTER each @Test

*@After*

public void tearDown() {

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

}

*@Test*

public void testAdd() {

// Act

int result = mathUtils.add(10, 5);

// Assert

*assertEquals*(15, result);

}

*@Test*

public void testDivide() {

// Act

int result = mathUtils.divide(10, 2);

// Assert

*assertEquals*(5, result);

}

*@Test*(expected = IllegalArgumentException.class)

public void testDivideByZero() {

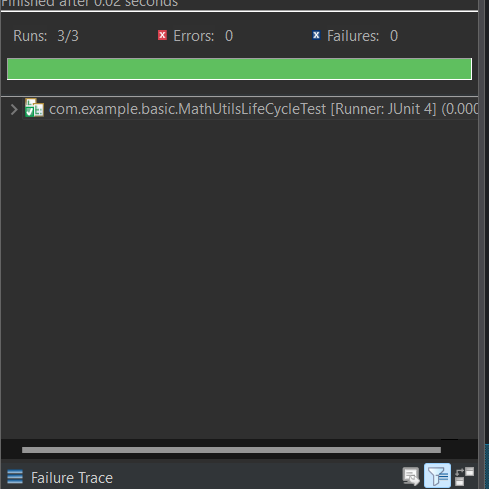
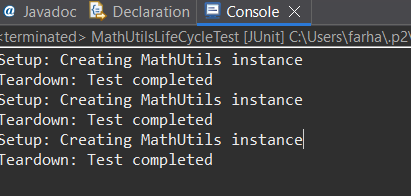
// Act

mathUtils.divide(10, 0); // Should throw

}

}

**Output:**

****