**JUnit Testing Exercises (Contains both Mandatory and Additional Exercises)**

**Exercise 1: Setting Up JUnit (Mandatory Exercise)**

**Scenario:**

**You need to set up JUnit in your Java project to start writing unit tests.**

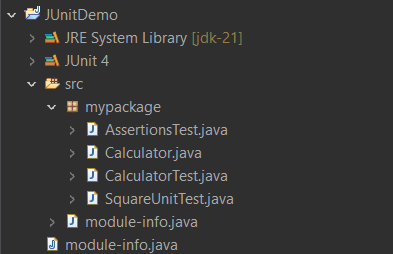
**Steps:**

1. **Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).**
2. **Add JUnit dependency to your project.**

For Eclipse (non-Maven):

Right-click project → Build Path → Add Libraries → JUnit → Select JUnit 4.

1. **Create a new test class in your project.**

****

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

**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.**

**Calculator.java**

package mypackage;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int square(int a) {

return a\*a;

}

}

**CalculatorTest.java**

package mypackage;

import static org.junit.Assert.\*;

import org.junit.Test;

public class CalculatorTest {

@Test

public void test() {

Calculator calc = new Calculator();

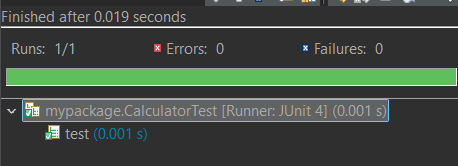
int res = calc.add(2, 3);

assertEquals(5, res);

}

}

**Output:**

****

**Exercise 3: Assertions in JUnit (Mandatory Exercise)**

**Scenario:**

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

**Steps:**

1. **Write tests using various JUnit assertions.**

**AssertionsTest.java**

package mypackage;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AssertionsTest {

*@Test*

public void testAssertions() {

*assertEquals*(5, 2 + 3);

*assertTrue*(5 > 3);

*assertFalse*(5 < 3);

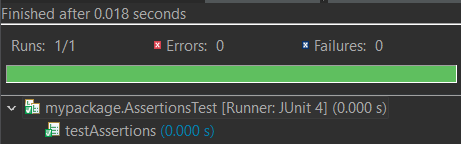
*assertNull*(null);

*assertNotNull*(new Object());

}

}

**Output:**

****

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

**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.**

**SquareUnitTest.java**

package mypackage;

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

public class SquareUnitTest {

private Calculator calc;

*@Before*

public void setUp() {

calc = new Calculator(); // Arrange

}

*@After*

public void tearDown() {

calc = null; // Clean up

}

*@Test*

public void testSquareOf3() {

int result = calc.square(3); // Act

*assertEquals*(9, result); // Assert

}

*@Test*

public void testSquareOf0() {

*assertEquals*(0, calc.square(0));

}

*@Test*

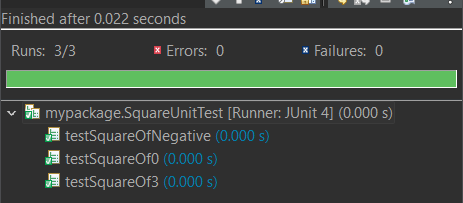
public void testSquareOfNegative() {

*assertEquals*(4, calc.square(-2));

}

}

**Output:**

****