Week2

Nunit and Moq

## 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>

1. Create a new test class in your project.

# CODE :

package com.example;

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

public class CalculatorTest { @Test

public void testAdd() {

Calculator calc = new Calculator();

*assertEquals*(5, calc.add(2, 3));

}

@Test

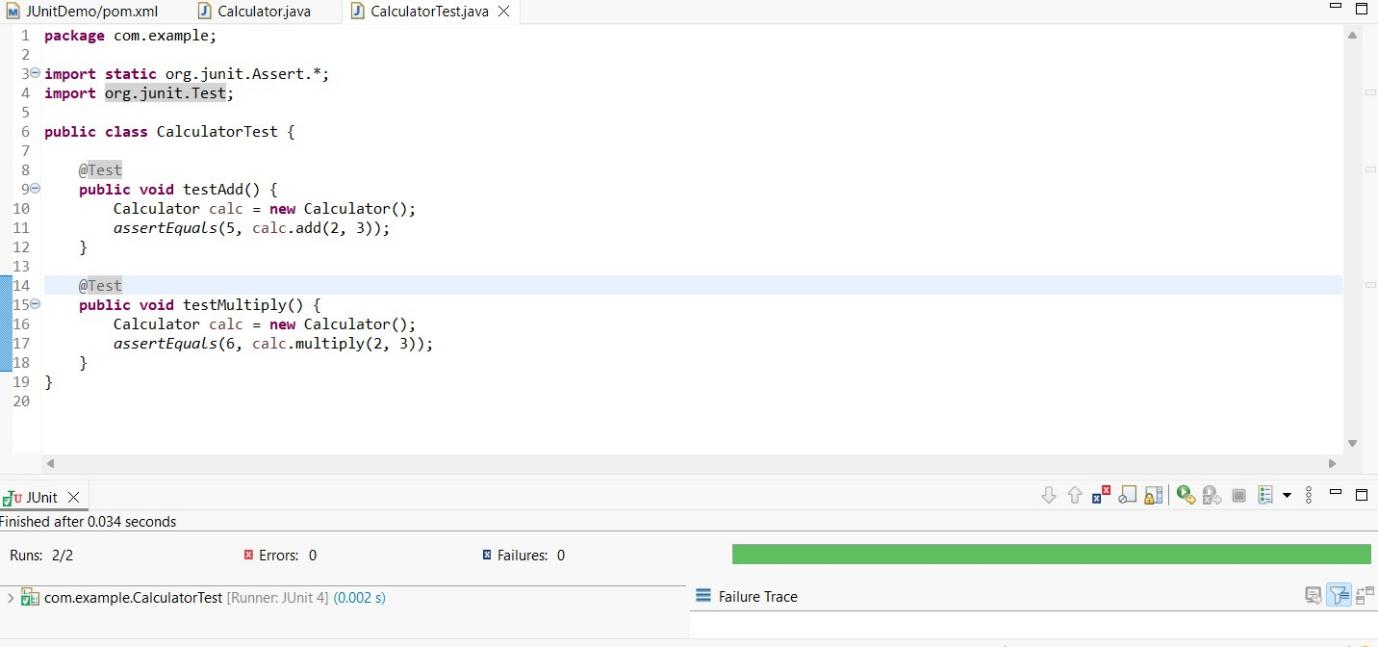
public void testMultiply() { Calculator calc = new Calculator();

*assertEquals*(6, calc.multiply(2, 3));

}

}

# 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.

## 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 :

package com.example.test; 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);

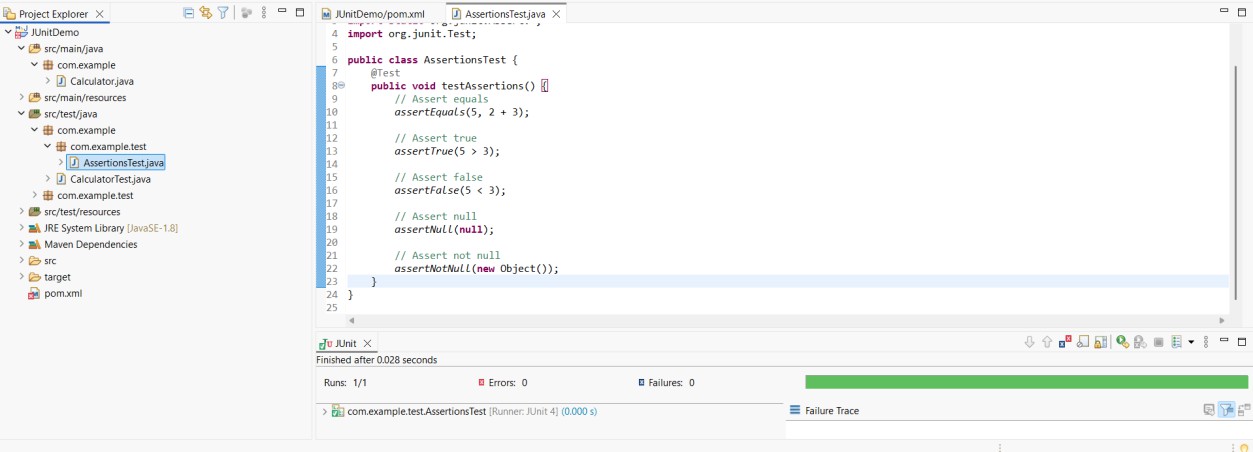
// Assert not null

*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 :

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;

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

import org.junit.After; import org.junit.Test;

public class CalculatorTest { private Calculator calculator; @Before

public void setUp() {

calculator = new Calculator(); // Arrange System.*out*.println("Setup: Calculator created");

}

@After

public void tearDown() { System.*out*.println("Teardown: Test finished\n");

}

@Test

public void testAdd() {

// Act

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

// Assert

*assertEquals*(5, result);

}

@Test

public void testMultiply() {

// Act

int result = calculator.multiply(3, 4);

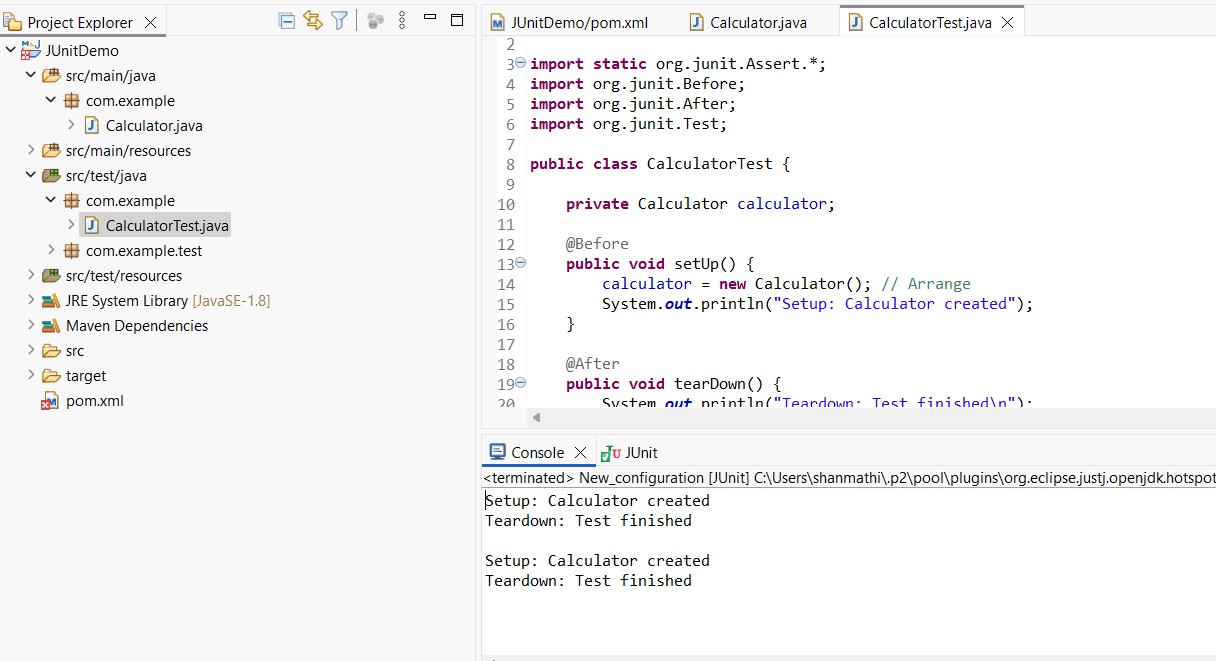
// Assert

*assertEquals*(12, result);

}

}

# OUTPUT :

****