**Superset ID: 6394725**

**JUnit\_Basic 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:

<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.

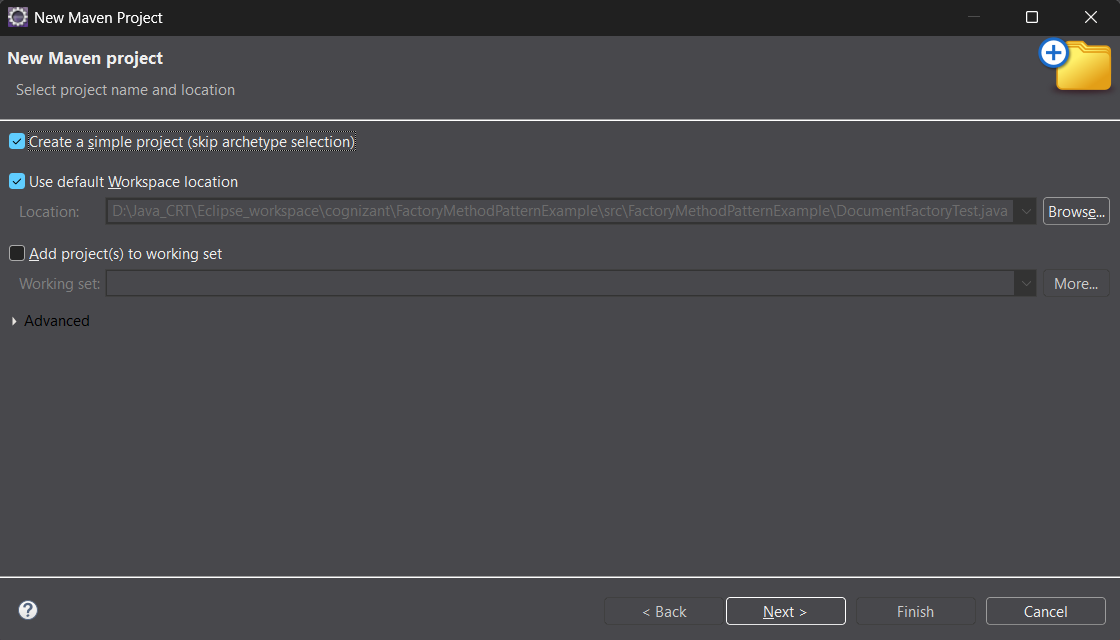
**Solution :**

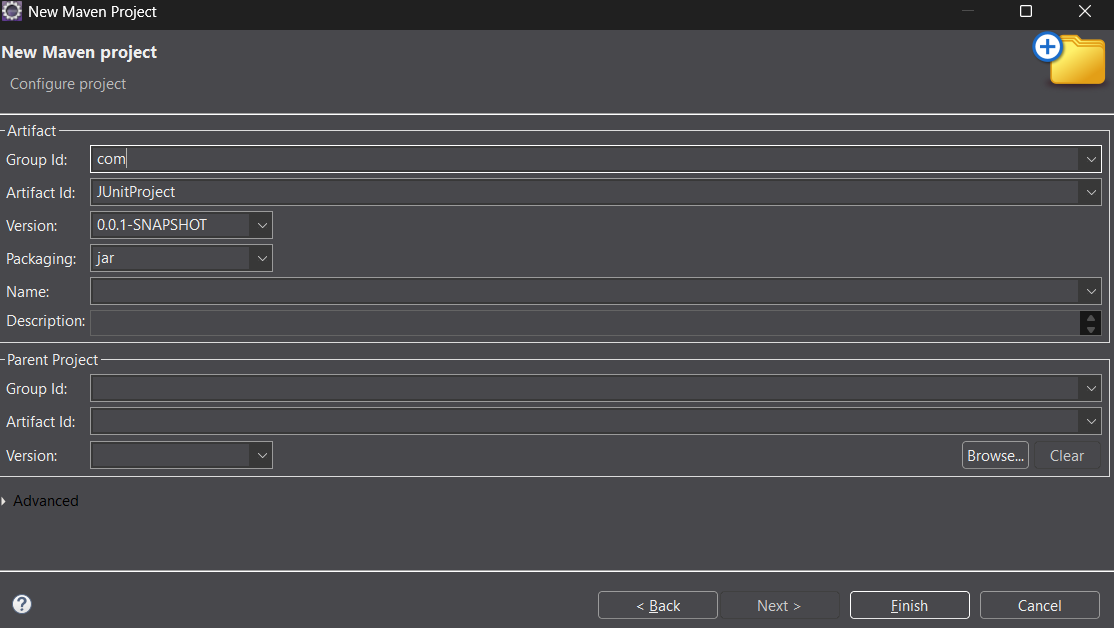
**Step 1:** Create a New Java Project

In Eclipse:

1. File → New → Maven Project.

2. Choose “Create a simple project”.





3. Finish the setup.

**Step 2:** Add JUnit Dependency

Open your pom.xml file and add the following inside <dependencies>:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>



**Step 3:** Create Java Class and Test Class

Create a Java class to test:

File: Calculator.java (under src/main/java)

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

Create a JUnit test class:

File: CalculatorTest.java (under src/test/java)

import org.junit.Test;

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

public class CalculatorTest {

*@Test*

public void testAdd() {

Calculator calc = new Calculator();

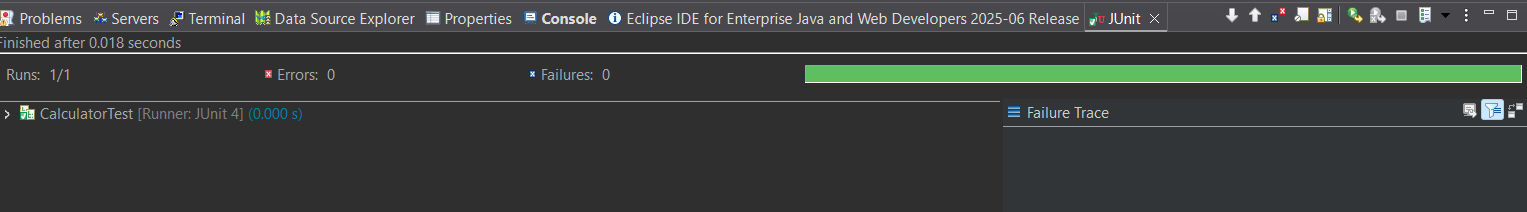
int result = calc.add(2, 3);

*assertEquals*(5, result);

}

}

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:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionTest {

*@Test*

public void testAssertions() {

// Assert equals

*assertEquals*(5, 2 + 3);

// Assert true

*assertTrue*(9 > 3);

// Assert false

*assertFalse*(9 < 3);

// Assert null

*assertNull*(null);

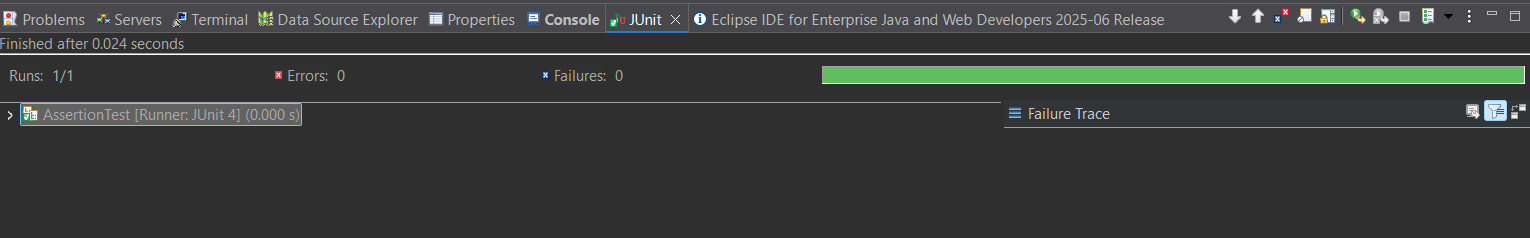
// Assert not null

*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.

**Solution:**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest1 {

private Calculator calculator;

// Setup method runs before each test

*@Before*

public void setUp() {

System.***out***.println("Setting up...");

calculator = new Calculator(); // Arrange

}

// Teardown method runs after each test

*@After*

public void tearDown() {

System.***out***.println("Cleaning up...");

calculator = null;

}

*@Test*

public void testAddition\_PositiveNumbers() {

// Arrange done in setUp()

// Act

int result = calculator.add(5, 3);

// Assert

*assertEquals*(8, result);

}

*@Test*

public void testAddition\_NegativeNumbers() {

// Act

int result = calculator.add(-2, -4);

// Assert

*assertEquals*(-6, result);

}

}

**Output:**

