COGNIZANT

Digital Nurture 4.0

Deep Skilling - Java FSE

WEEK-2 HANDS ON

By Kaviya P

**JUNIT\_BASIC TESTING EXCERCISES**

**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>

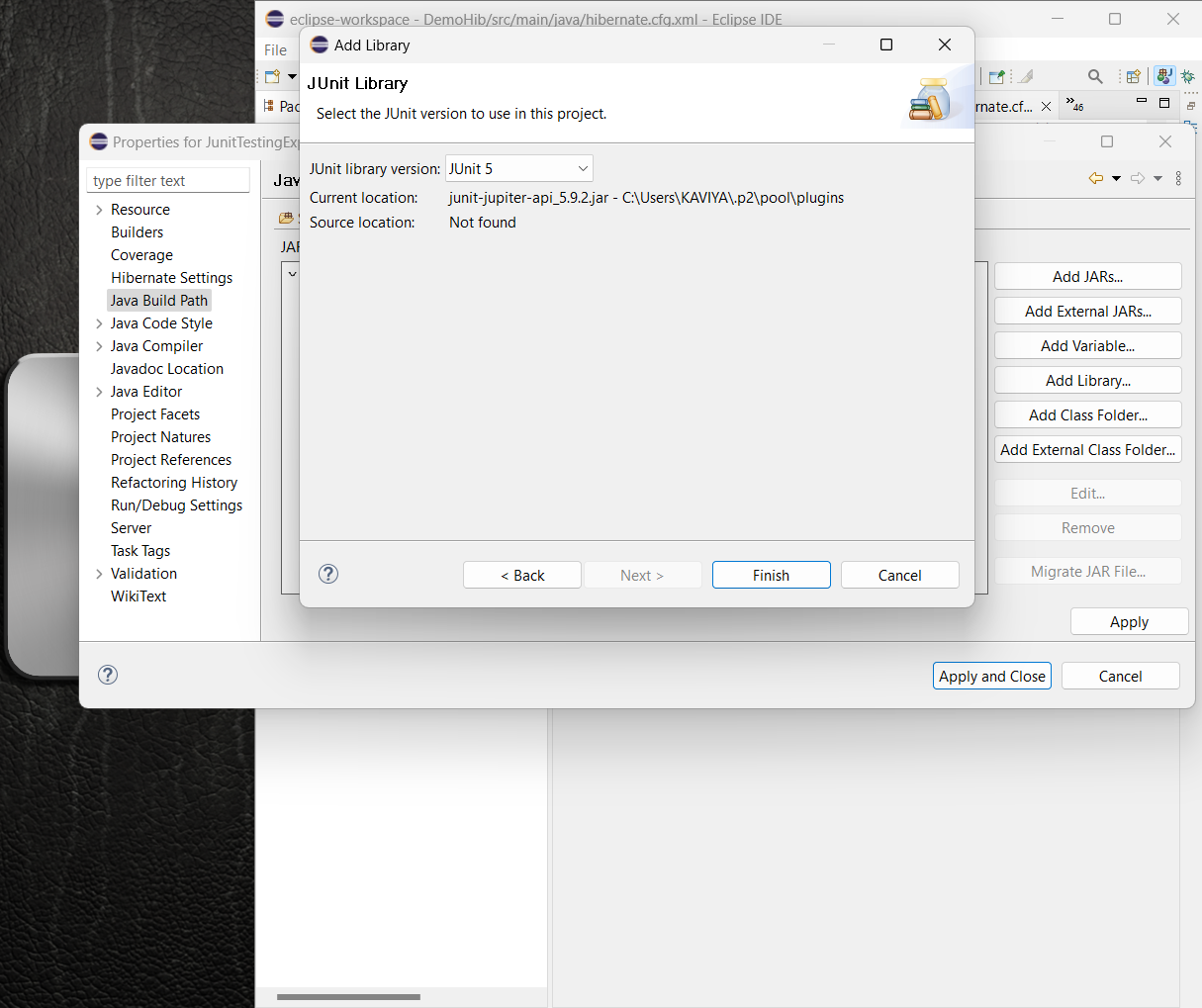
<groupId>junit</groupId>

<artifactId>junit</artifactId>

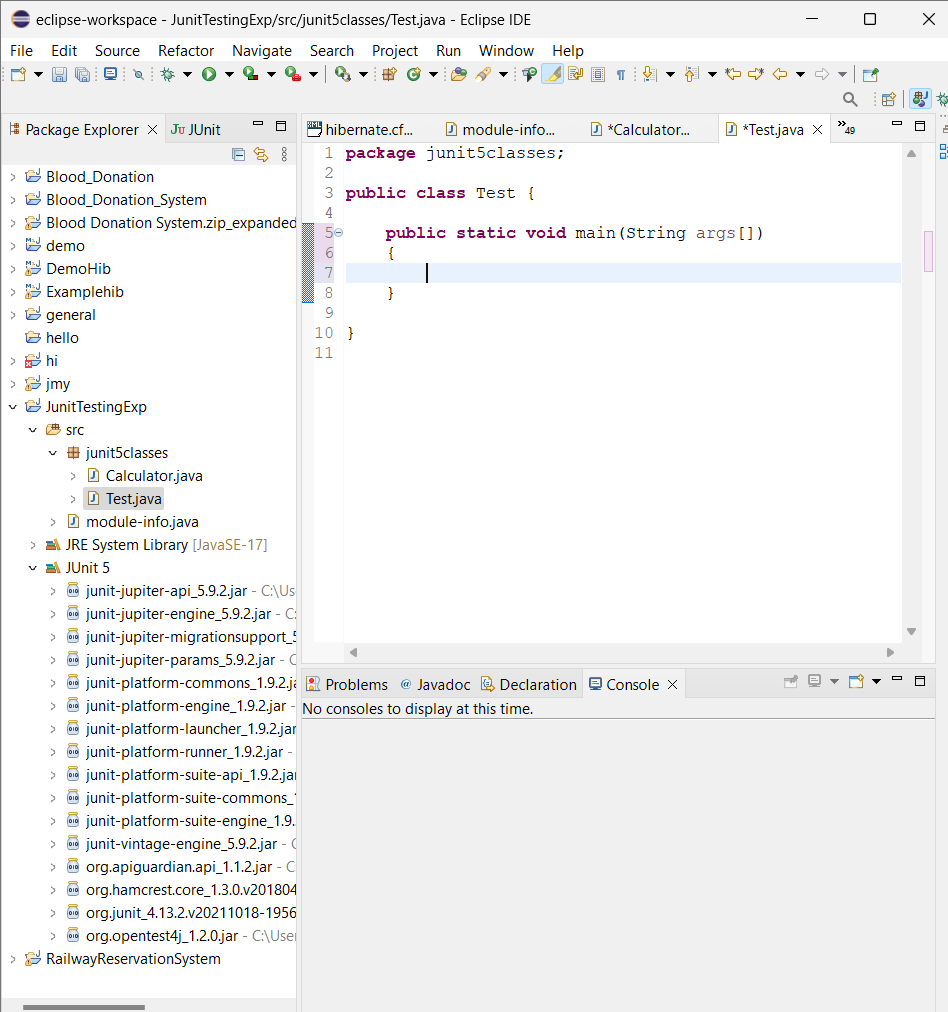
<version>4.13.2</version>

<scope>test</scope>

</dependency>



3. Create a new test class in your project.



**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.**

**package junit5classes;**

**public class Calculator {**

**// TODO Auto-generated constructor stub**

**public int add(int a,int b)**

**{**

**int sum=a+b;**

**return sum;**

**}**

**}**

1. **Write JUnit tests for these methods.**

**package** junti5Tests;

**import** org.junit.jupiter.api.Test;

**import** **static** org.junit.jupiter.api.Assertions.\*;

**import** junit5classes.Calculator;

**public** **class** JunitTest {

@Test

**public** **void** testMethod()

{

Calculator cal=**new** Calculator();

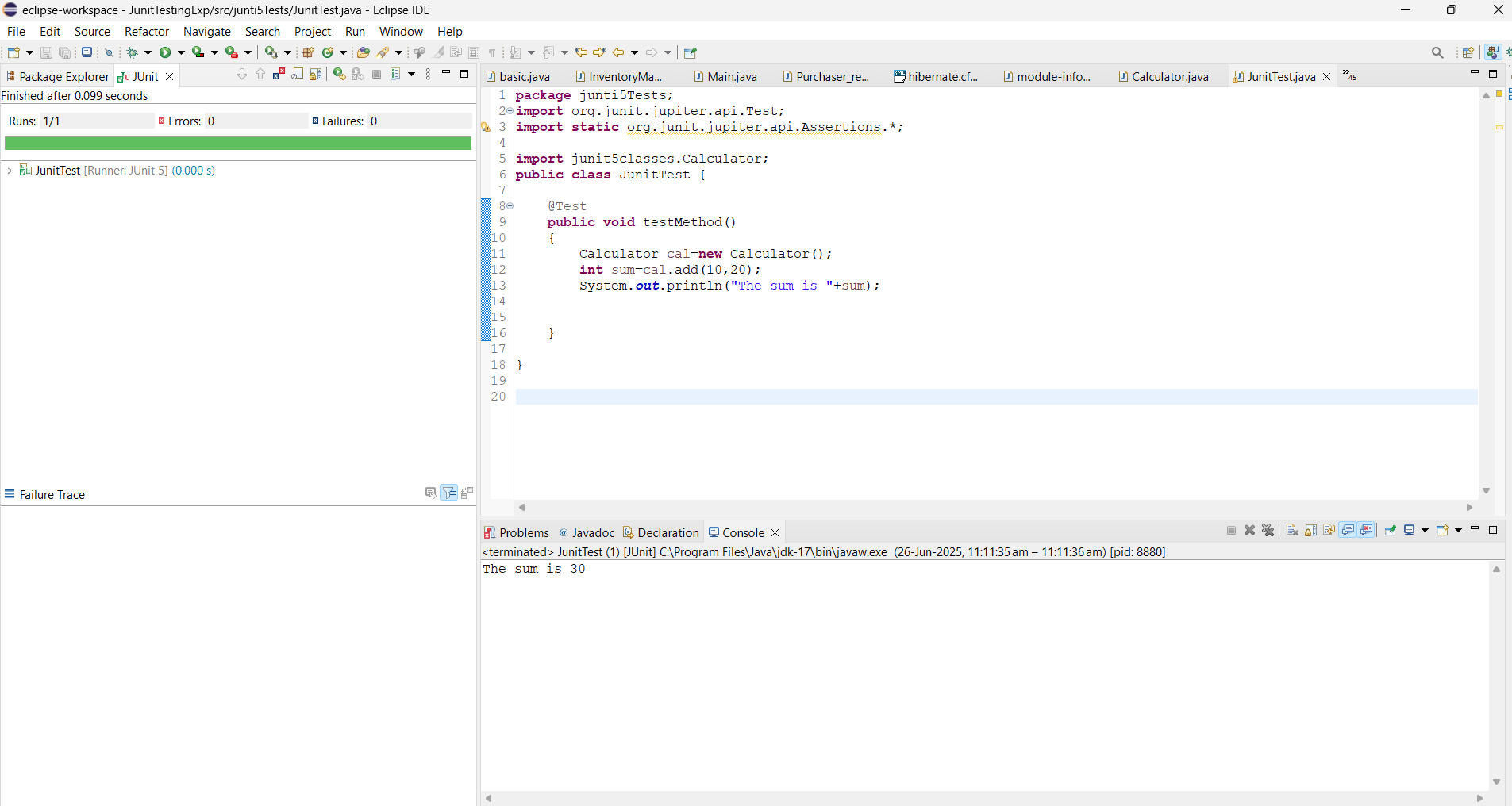
**int** sum=cal.add(10,20);

System.***out***.println("The sum is "+sum);

}

}

**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.

**package** junti5Tests;

**import** org.junit.jupiter.api.Test;

**import** **static** org.junit.jupiter.api.Assertions.\*;

**import** org.junit.jupiter.api.DisplayName;

**import** junit5classes.Calculator;

**public** **class** JunitTest {

@Test

**public** **void** testAssertions() {

*assertEquals*(5, 2 + 3);

*assertTrue*(5 > 3);

*assertFalse*(5 < 3);

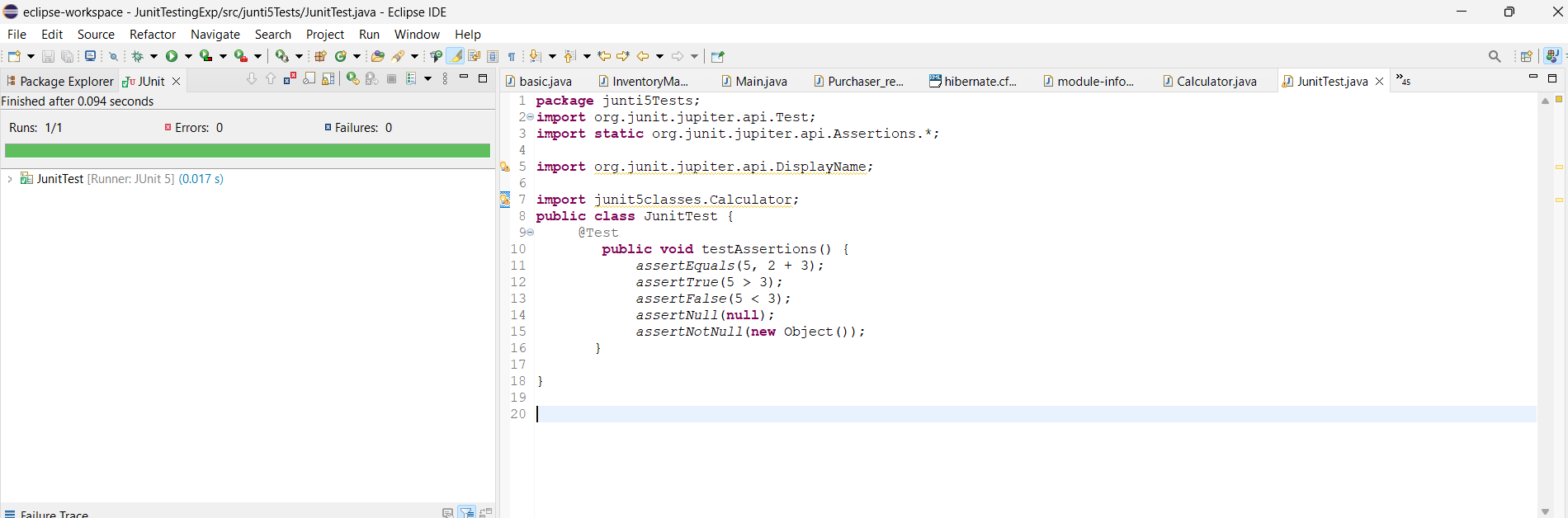
*assertNull*(**null**);

*assertNotNull*(**new** Object());

}

}

**OUTPUT :**



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

Teardown Methods in JUnit

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.

**Calculator.java**

**package** junit5classes;

**public** **class** Calculator {

// **TODO** Auto-generated constructor stub

**public** **int** add(**int** a,**int** b)

{

**int** sum=a+b;

**return** sum;

}

**public** **int** subract(**int** a,**int** b)

{

**int** diff=a-b;

**return** diff;

}

}

**JunitTest.java**

**package** junti5Tests;

**import** org.junit.jupiter.api.Test;

**import** **static** org.junit.jupiter.api.Assertions.\*;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.DisplayName;

**import** junit5classes.Calculator;

**public** **class** JunitTest {

**private** Calculator calculator;

@BeforeEach //SETUP

**public** **void** setUp() {

calculator = **new** Calculator();

System.***out***.println("Setup complete");

}

@AfterEach

**public** **void** tearDown() {

calculator = **null**;

System.***out***.println("Teardown complete");

}

@Test

**public** **void** testAddUsingAAA() {

**int** result = calculator.add(10, 20);

*assertEquals*(30, result);

}

@Test

**public** **void** testSubtractUsingAAA() {

**int** result = calculator.subract(10, 4);

*assertEquals*(6, result);

}

}

**OUTPUT :**

