**WEEK – 02**

1. **JUnit\_Basic Testing Exercises**

**Superset ID: 6262264**

**Exercise 1: Setting Up Junit**

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

**Solution :**

Steps Followed in Eclipse to Run the JUnit Test

**1.Created a New Java Project**

* Opened Eclipse IDE.
* Went to File > New > Java Project.
* Named the project: JUnitSetupExample.

**2. Added JUnit Library**

* Right-clicked the project > Properties.
* Selected Java Build Path > Libraries tab.
* Clicked Add Library… > JUnit.
* Chose JUnit 4 and clicked Finish.

**pom.xml**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

**3. Created the Test Class**

* Right-clicked src/test/java.
* Chose New > Class.
* Named it CalculatorTest.
* Added this code:

**CalculatorTest.java**

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAddition() {

int result = 2 + 3;

assertEquals(5, result);

}

}

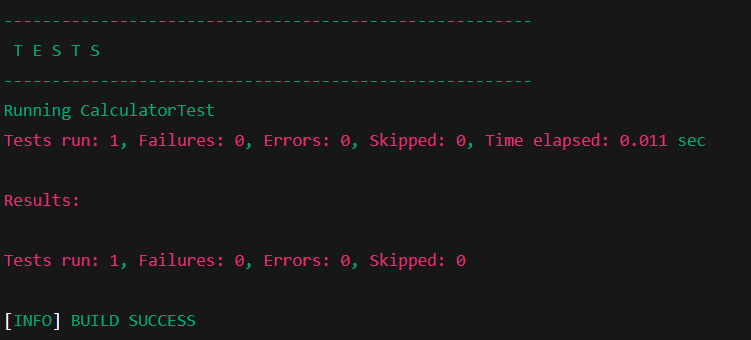
**4. Ran the Test**

* Right-clicked CalculatorTest.java.
* Chose Run As > JUnit Test.

**5. Observed Output**

* Eclipse JUnit view showed:
  + Green bar indicating success.
  + 1 test run, 0 failures, 0 errors.

**OUTPUT:**

****

**EXPLANATION:**

This exercise sets up JUnit in a Java project by adding the dependency to pom.xml.  
A simple test class verifies basic functionality with an assertion.  
Running the test confirms JUnit is correctly integrated and working.

**Exercise 3: Assertions in JUnit**

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

1.Write tests using various JUnit assertions.

**SOLUTION:**

import org.junit.Test;

import static org.junit.Assert.\*;

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());

}

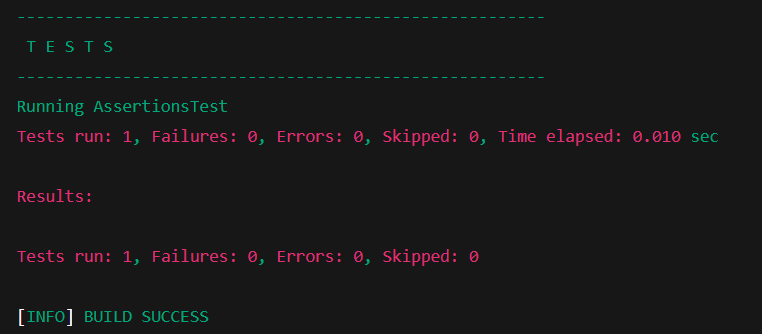
}

**EXPLANATION:**

This JUnit test demonstrates how to use multiple assertions:

* assertEquals() checks that 2 + 3 equals 5.
* assertTrue() and assertFalse() validate boolean conditions.
* assertNull() ensures a value is null.
* assertNotNull() confirms an object is not null.  
  All these assertions together help verify expected results in unit tests.

**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.

**SOLUTION :**

**Calculator Class :**

public class Calculator {

public int add(int x, int y) {

return x + y;

}

public int subtract(int x, int y) {

return x - y;

}

}

**CalculatorTest Class Using AAA and Setup/Teardown :**

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

**// Setup method runs before each test**

**@Before**

public void setUp() {

calculator = new Calculator();

System.out.println("Setup completed.");

}

**// Teardown method runs after each test**

**@After**

public void tearDown() {

calculator = null;

System.out.println("Teardown completed.");

}

**@Test**

public void testAddition() {

**// Arrange**

int a = 2;

int b = 3;

**// Act**

int result = calculator.add(a, b);

**// Assert**

assertEquals(5, result);

}

**@Test**

public void testSubtraction() {

**// Arrange**

int a = 5;

int b = 3;

**// Act**

int result = calculator.subtract(a, b);

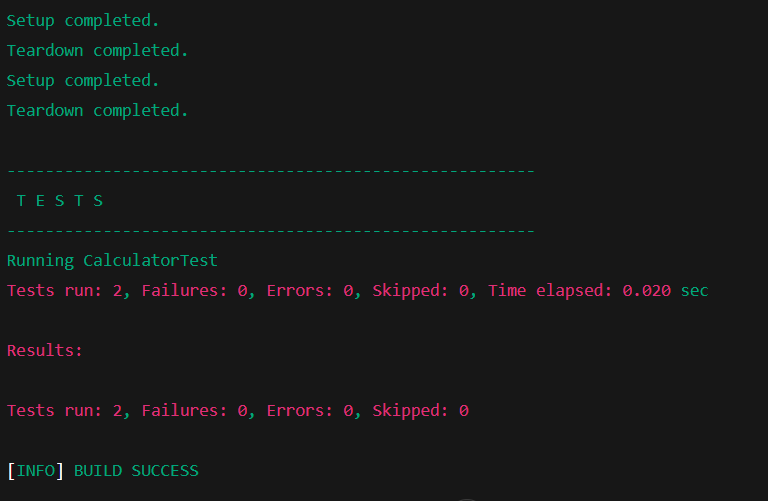
**// Assert**

assertEquals(2, result);

}

}

**OUTPUT :**

****

**EXPLANATION :**

This example shows how to write tests with the Arrange-Act-Assert pattern for clarity.  
@Before sets up the calculator before each test, and @After cleans up afterward.  
Each test method follows AAA to verify correct behavior.