**JUnit Testing Exercises**

**Exercise 1: Setting Up Junit**

**Step 1: Create a Java Project**

We created a new Java Maven project in Eclipse to serve as the foundation for writing and running Java code.

**Step 2: Add JUnit Dependency**

We added the JUnit 4.13.2 dependency in the pom.xml file so that the project can use JUnit’s testing features.

**Step 3: Create a Java Class to Test**

1. Right-click on src/main/java → New > Package → Name it: com.example.
2. Right-click on the package → New > Class → Name it: Calculator.

**Calculator.java:**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**Step 4: Create a Test Class**

1.Right-click on src/test/java → **New > Package** → Name it: com.example.

2.Right-click on the package → **New > JUnit Test Case**.

3.Name it CalculatorTest, select JUnit 4 → Click **Finish**.

**CalculatorTest.java:**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

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

assertEquals(5, result);

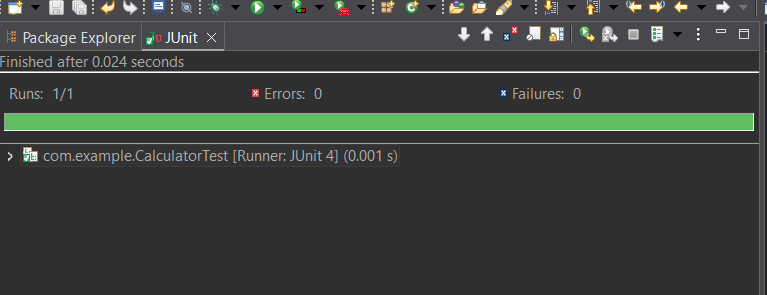
}

}

**Step 5: Run the Test**

1. Right-click CalculatorTest.java → **Run As > JUnit Test**.
2. You should see a **green bar** in the JUnit view, which means the test passed.

**OUTPUT:**

****

**Exercise 3: Assertions in Junit**

**Step 1: Create a JUnit Test Class**

Create a Java class named AssertionsTest inside src/test/java/com/example (or any package you’re using):

**AssertionsTest.java:**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

// Assert equals: checks if 2 + 3 equals 5

assertEquals(5, 2 + 3);

// Assert true: checks if 5 > 3 is true

assertTrue(5 > 3);

// Assert false: checks if 5 < 3 is false

assertFalse(5 < 3);

// Assert null: checks if the value is null

assertNull(null);

// Assert not null: checks if the object is not null

assertNotNull(new Object());

}

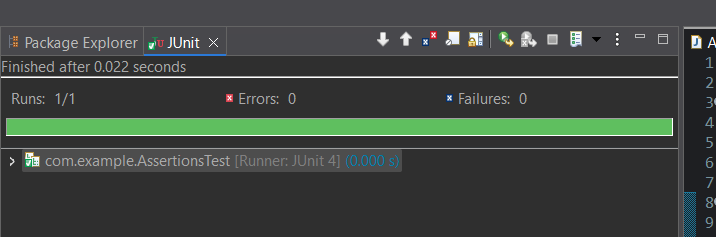
}

**Step 2:** Add JUnit Dependency in pom.xml

**Step 3:** Run the Test

1. Right-click the AssertionsTest.java file.
2. Choose Run As > JUnit Test.

**OUTPUT:**

****

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

1. Use the **Arrange-Act-Assert (AAA)** pattern in your test methods.

2. Use **@Before** and **@After** annotations in JUnit to setup and teardown test fixtures (i.e., shared objects used across multiple tests).

**Step 1:** **Create a New Class to Be Tested**

**MathService.java**:

package com.example;

public class MathService {

public int subtract(int a, int b) {

return a - b;

}

public int square(int x) {

return x \* x;

}

}

**Step 2: Create the Test Class**

create the test class using the **AAA pattern** and **setup/teardown methods**.

**MathServiceTest.java:**

package com.example;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class MathServiceTest {

private MathService mathService;

// Setup method: runs before each test

@Before

public void initialize() {

mathService = new MathService();

System.out.println("Setup: MathService instance created.");

}

// Teardown method: runs after each test

@After

public void cleanup() {

mathService = null;

System.out.println("Teardown: MathService instance set to null.");

}

@Test

public void testSubtraction() {

// Arrange

int a = 10;

int b = 4;

// Act

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

// Assert

assertEquals(6, result);

}

@Test

public void testSquare() {

// Arrange

int input = 5;

// Act

int result = mathService.square(input);

// Assert

assertEquals(25, result);

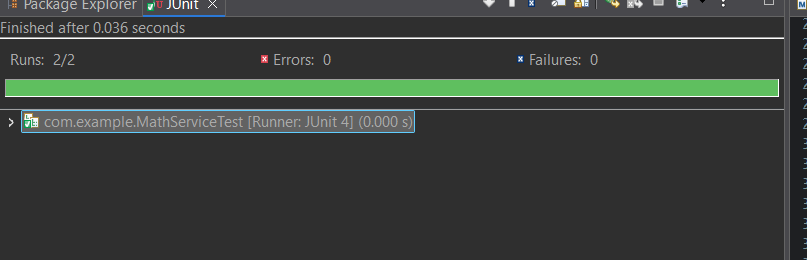
}

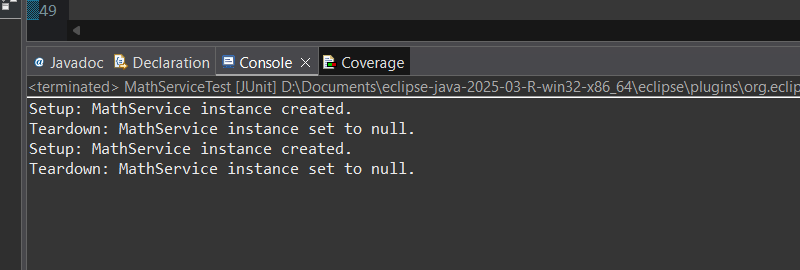
}

**Step 3: Run the Test**

1. Right-click CalculatorTest.java → **Run As > JUnit Test**.
2. You should see a **green bar** in the JUnit view, which means the test passed.

**OUTPUT:**

****

****

**Mockito Hands-On Exercises**

**Exercise 1: Mocking and Stubbing**

**Step 1:** Add Mockito and JUnit 5 Dependencies

**Step 2:** Create the External API Interface **src/main/java/com/example/ExternalApi.java**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**Step 3:** Create the Service Class That Uses the API

**MyService.java:**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**Step 4:** Create the Test Class Using Mockito

**MyServiceTest.java:**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

public class MyServiceTest {

@Test

public void testExternalApi() {

// Step 1: Create mock object

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Stub the method

when(mockApi.getData()).thenReturn("Mock Data");

// Step 3: Inject mock into the service

MyService service = new MyService(mockApi);

// Step 4: Call the method and assert

String result = service.fetchData();

assertEquals("Mock Data", result); }}

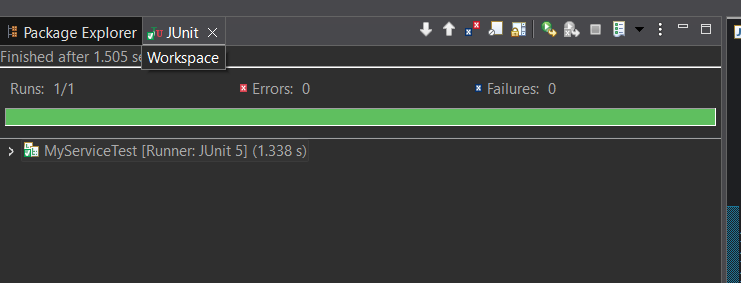
**Step 4:** Run the Test

1. Open MyServiceTest.java.

2. Right-click inside the file → Choose Run As → JUnit Test.

3. Eclipse runs the test and shows the JUnit panel with a green bar if all tests pass.

**OUTPUT:**

****

**Exercise 2: Verifying Interactions**

**Step 1:** Add Mockito and JUnit 5 Dependencies

**Step 2:** Create the External API Interface **src/main/java/com/example/ExternalApi.java**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**Step 3:** Create the MyService Class That Uses It.

**MyService.java:**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData(); // This method should be called and verified

}

}

**Step 4:** Create the Test Class with Verification.

**MyServiceTest.java:**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

// Step 1: Create a mock

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Create service using mock

MyService service = new MyService(mockApi);

// Step 3: Call the method

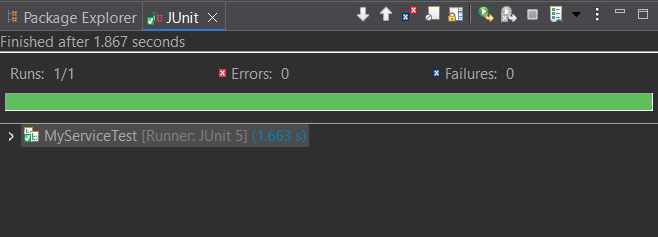
service.fetchData();

// Step 4: Verify interaction

verify(mockApi).getData(); // Verifies that getData() was called

}}

**OUTPUT:**

****