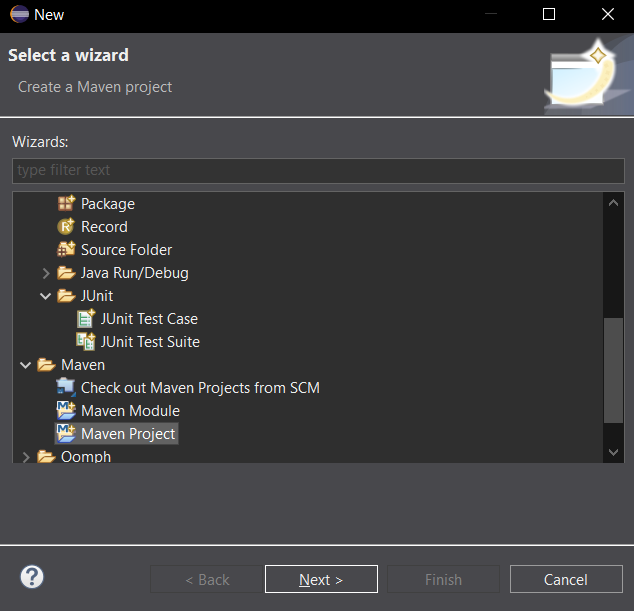
**TDD using Junit5 and Mockito**

**Setting up Junit**

Create a Maven Project using Eclipse IDE



Add Junit dependency to project. add the following to pom.xml:

<dependency>

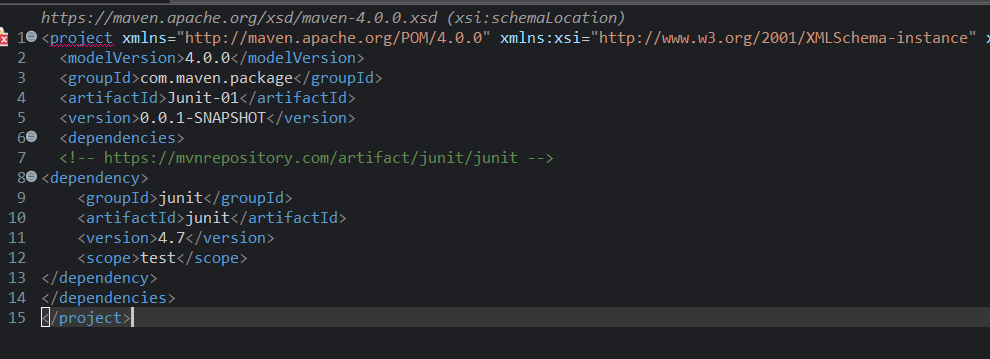
<groupId>junit</groupId>

<artifactId>junit</artifactId>

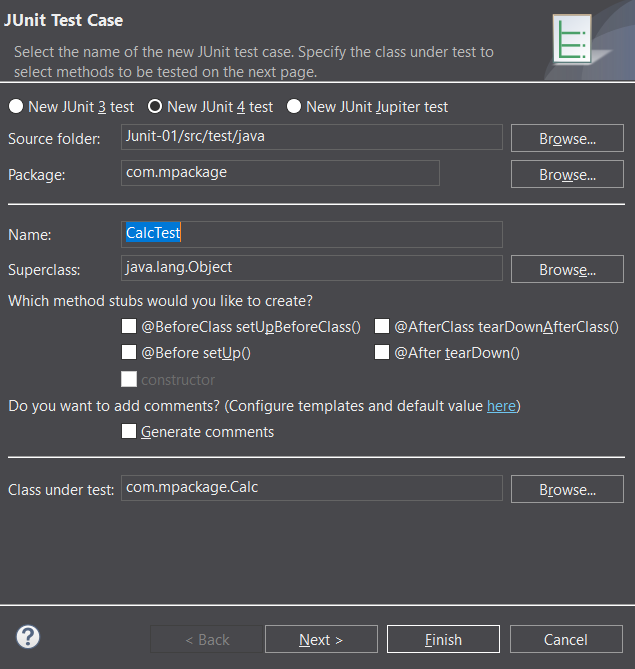
<version>4.7</version>

<scope>test</scope>

</dependency



Create a new test class in project.



**2.ASSERTIONS IN JUNIT**

Calc.java

package com.mpackage;

public class Calc {

public int divide(int num1, int num2)

{

return num1/num2;

}

}

package com.mpackage;

import static org.junit.Assert.\*;

import java.util.Arrays;

import org.junit.Test;

public class AssertionsTest {

*@Test*

public void test() {

Calc opr = new Calc();

*assertEquals*(2,opr.divide(10, 5));

*assertEquals*(5,2+3);

int[] expected = {2,4,6,8};

int[] actual = {4,6,8,2};

Arrays.*sort*(actual);

*assertArrayEquals*(expected,actual);

*assertTrue*(5>3);

*assertFalse*(5<3);

*assertNull*(null);

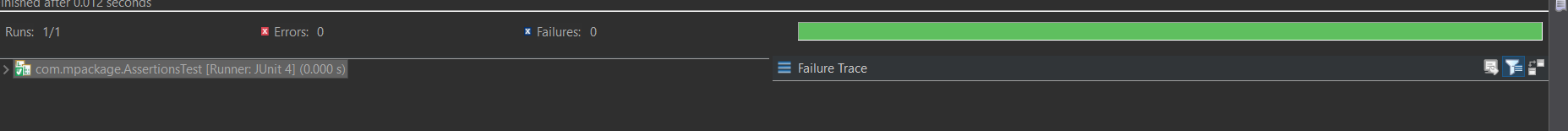
*assertNotNull*(new Calc());

}

}

**Output**

Actual output matches with the expected output so the test cases are passed.



**Arrange-act-assert (AAA) Pattern, Test Fixtures, Setup and teardown methods in junit**

package com.mpackage;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AAATest {

*@Test*

public void test() {

//Arrange

Calc obj = new Calc();

int a= 10, b =5;

//Act

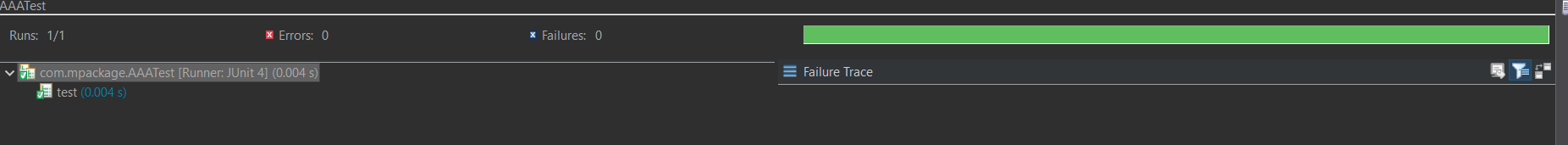
int actual = obj.divide(a, b);

//Assert

*assertEquals*(2,actual);

}

}



package com.mpackage;

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

import org.junit.jupiter.api.AfterAll;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.AfterEach;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Test;

class CalculatorTest {

Calculator obj;

*@BeforeAll*

static void beforeAll() {

System.***out***.println("Before all test cases");

}

*@BeforeEach*

void beforeEach()

{

obj = new Calculator();

System.***out***.println("Before each test case");

}

*@Test*

void divide\_test()

{

int actual = obj.divide(10, 5);

*assertEquals*(2,actual);

System.***out***.println("Checking the divide test");

}

*@AfterEach*

void AfterEach()

{

System.***out***.println("After each test case");

}

*@Test*

void add\_test()

{

int actual = obj.add(2, 8);

*assertEquals*(10,actual);

System.***out***.println("Checking the add test");

}

*@AfterAll*

static void afterall()

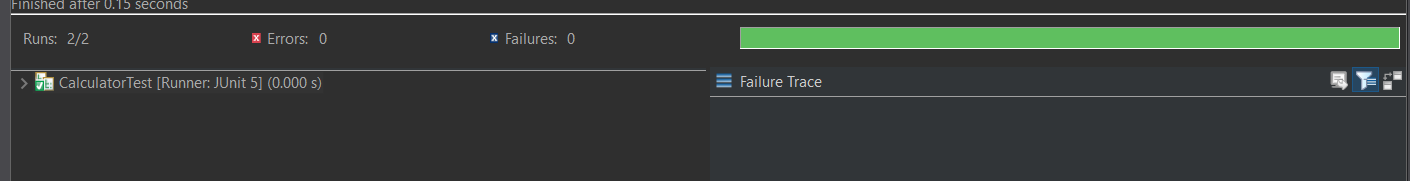
{

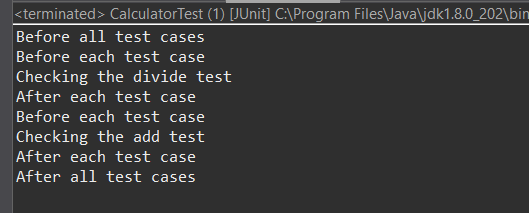
System.***out***.println("After all test cases");

}

}

**s**





**Mocking and Stubbing**

package com.mockito;

public interface ExternalApi {

String getData();

}

package com.mockito;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

package com.mockito;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import static org.junit.jupiter.api.Assertions.\*; // You missed this import in your code

public class MyServiceTest {

*@Test*

public void testExternalApi() {

ExternalApi mockApi = *mock*(ExternalApi.class);

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

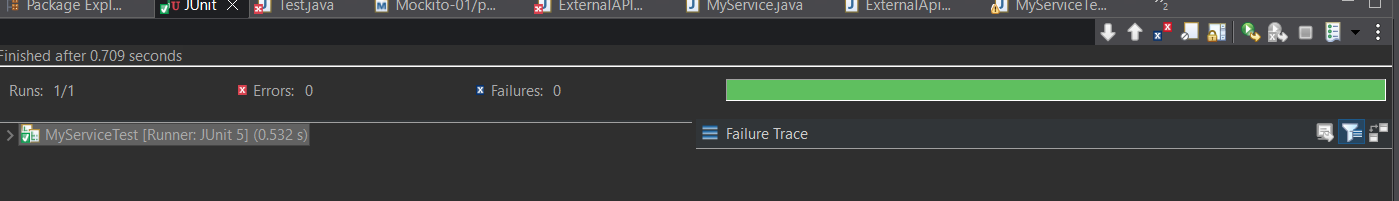
MyService service = new MyService(mockApi);

String result = service.fetchData();

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

}

}



**Verifying Interactions**

package com.mockito;

public interface ExternalApi {

String getData();

}

package com.mockito;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

package com.mockito;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

*@Test*

public void testVerifyInteraction() {

ExternalApi mockApi = Mockito.*mock*(ExternalApi.class); // Step 1: Create mock

MyService service = new MyService(mockApi); // Inject mock

service.fetchData(); // Step 2: Call method

*verify*(mockApi).getData(); // Step 3: Verify interaction

}

}

