1. JUnit\_Basic Testing Exercises

# Exercise 1: Setting Up JUnit

Adding Dependencies in Pom.xml :

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>5.11.0</version> <!-- Or latest -->

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

**SimpleMath.java:**

package mav.mav01;

public class SimpleMath {

public int add(int a, int b) {

return a + b;

}

}

**SimpleMathTest.java:**

package mav.mav01;

import org.junit.jupiter.api.Test;

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

public class SimpleMathTest {

@Test

void testAdd() {

SimpleMath math = new SimpleMath();

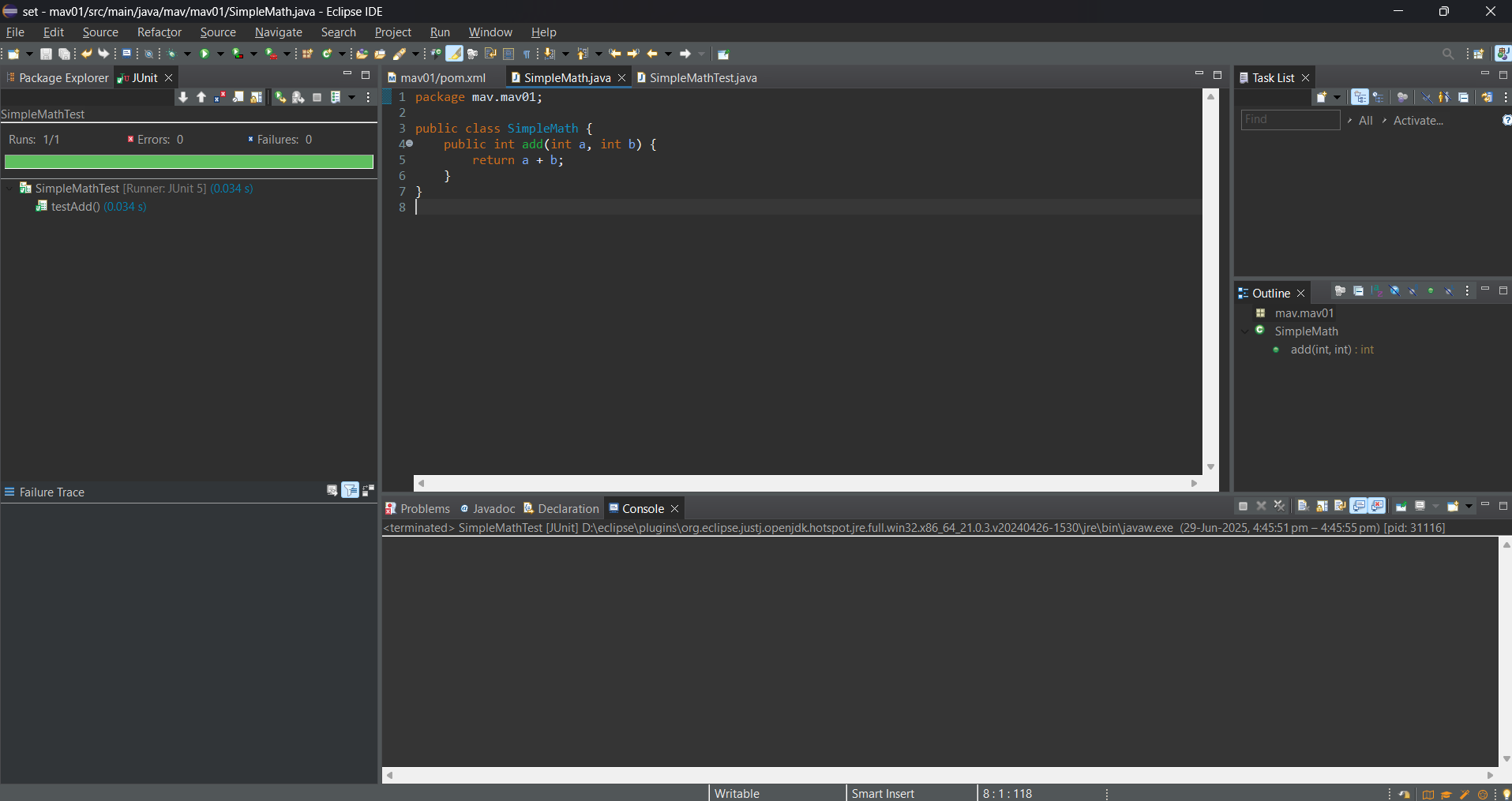
int result = math.add(2, 3);

assertEquals(5, result);

}

}

**OUTPUT :**



# Exercise 3: Asertions in Junit

package mav.mav01;

import org.junit.jupiter.api.Test;

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

public class AssertionsTest {

@Test

void testAssertions() {

// Assert equals: checks if two values are equal

assertEquals(5, 2 + 3);

// Assert true: checks if the condition is true

assertTrue(5 > 3);

// Assert false: checks if the condition is false

assertFalse(5 < 3);

// Assert null: checks if the object is null

assertNull(null);

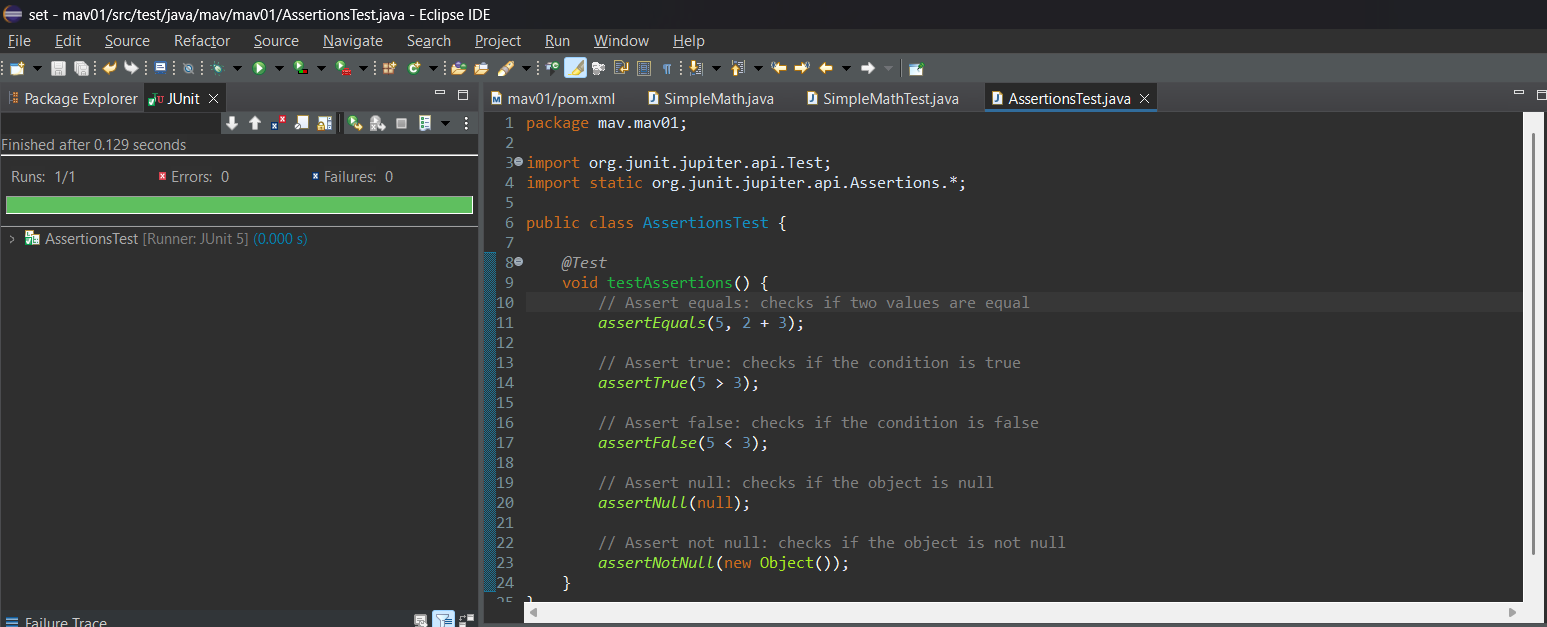
// Assert not null: checks if the object is not null

assertNotNull(new Object());

}

}

**OUPUT:**



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

**MathService.java:**

package mav.mav01;

public class MathService {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

**MathServiceTest.java:**

package mav.mav01;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

public class MyServiceTest {

@Test

void testExternalApi() {

// Step 1: Create mock

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Stub method

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

// Step 3: Inject mock into service

MyService service = new MyService(mockApi);

// Act

String result = service.fetchData();

// Assert

assertEquals("Mock Data", result);

}

}

**OUTPUT:**

