Week 2 Mandatory Hands-on Exercise Answers

# TDD using JUnit5 and Mockito - Exercise 1: Setting Up JUnit

**CODE:**

import java.util.\*;

class Calculator {

public int add(int a, int b) {

return a + b;

}

}

class Main {

public static void main(String[] args) {

Calculator calculator = new Calculator();

int result = calculator.add(5, 3);

if (result == 8) {

System.out.println("testAddition PASSED");

} else {

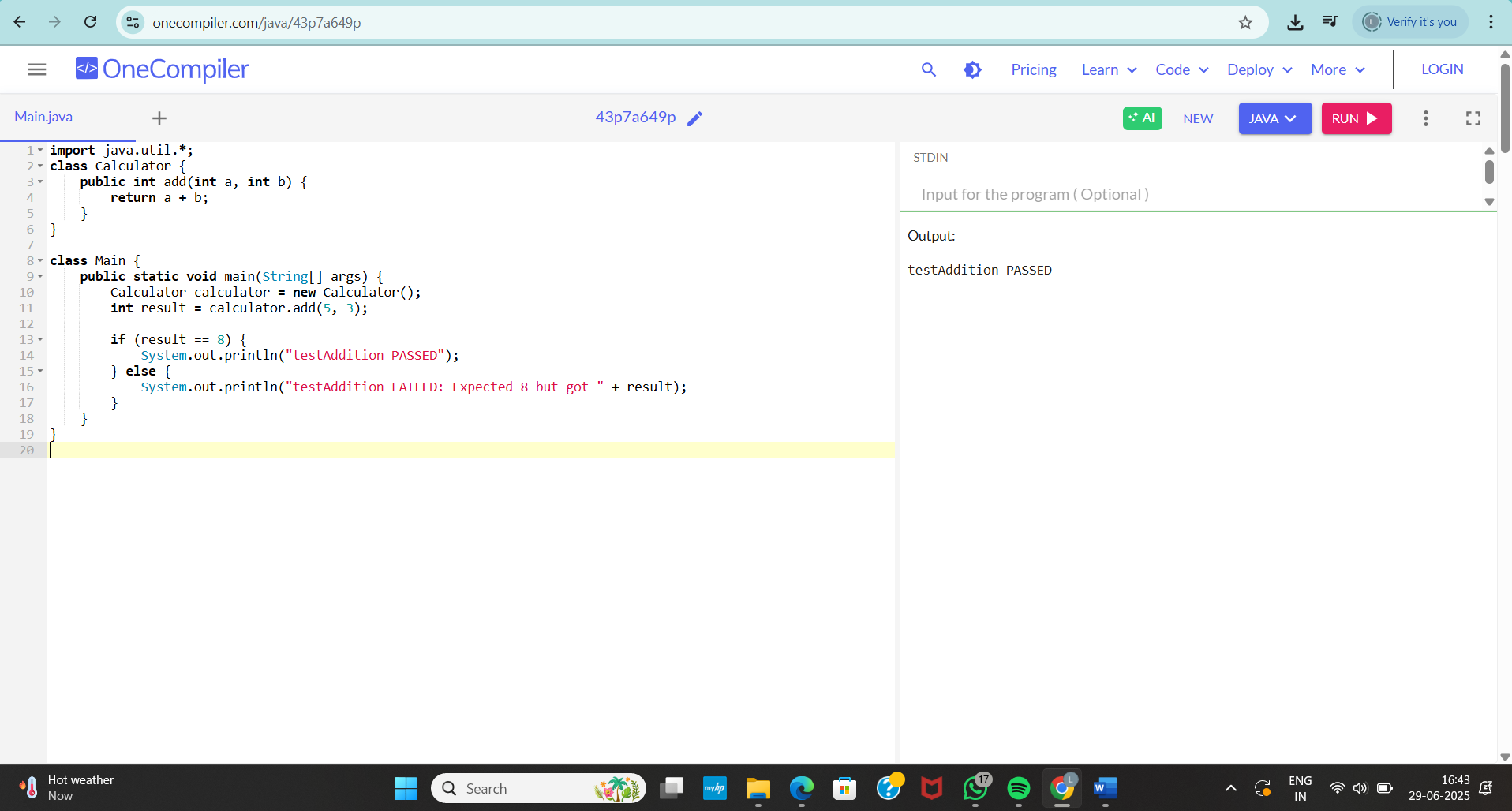
System.out.println("testAddition FAILED: Expected 8 but got " + result);

}

}

}

**OUTPUT:**



# TDD using JUnit5 and Mockito - Exercise 3: Assertions in JUnit

**CODE:**

import java.util.\*;

class MathUtils {

public int multiply(int a, int b) {

return a \* b;

}

public boolean isEven(int number) {

return number % 2 == 0;

}

}

public class Main {

public static void main(String[] args) {

MathUtils mathUtils = new MathUtils();

// Simulate assertEquals

int expected = 20;

int actual = mathUtils.multiply(4, 5);

if (expected == actual) {

System.out.println("testMultiply PASSED");

} else {

System.out.println("testMultiply FAILED: Expected " + expected + ", but got " + actual);

}

// Simulate assertTrue

if (mathUtils.isEven(6)) {

System.out.println("testIsEvenTrue PASSED");

} else {

System.out.println("testIsEvenTrue FAILED");

}

// Simulate assertFalse

if (!mathUtils.isEven(5)) {

System.out.println("testIsEvenFalse PASSED");

} else {

System.out.println("testIsEvenFalse FAILED");

}

// Simulate assertNotNull

if (mathUtils != null) {

System.out.println("testNotNull PASSED");

} else {

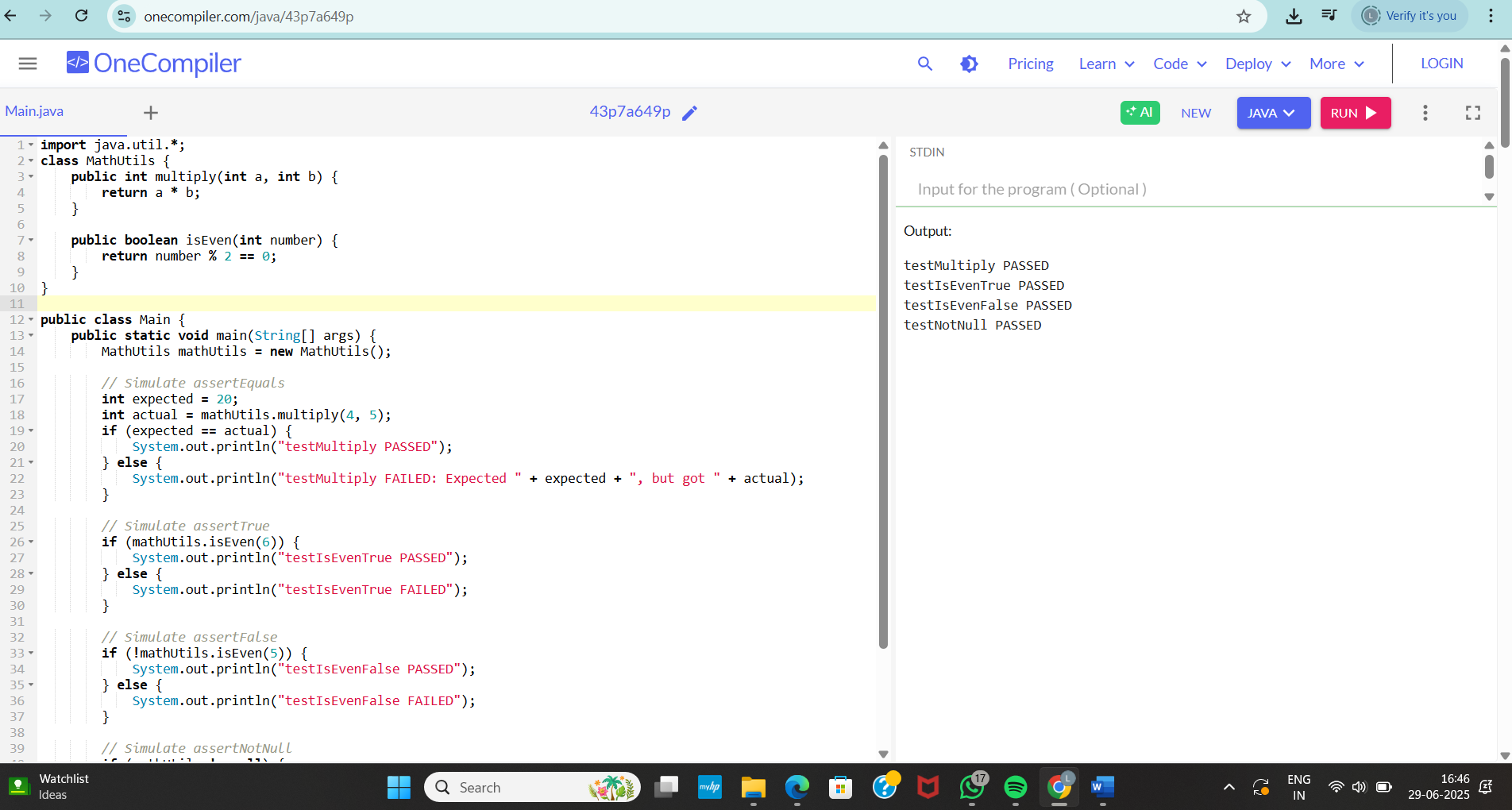
System.out.println("testNotNull FAILED");

}

}

}

**OUTPUT:**



# TDD using JUnit5 and Mockito - Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

**CODE:**

import java.util.\*;

class StringUtils {

public String toUpperCase(String input) {

return input == null ? null : input.toUpperCase();

}

public boolean isEmpty(String input) {

return input == null || input.trim().isEmpty();

}

}

public class Main {

static StringUtils stringUtils;

// Simulate @BeforeEach

public static void setUp() {

stringUtils = new StringUtils();

System.out.println("Setup before test");

}

// Simulate @AfterEach

public static void tearDown() {

System.out.println("Teardown after test\n");

}

public static void main(String[] args) {

testToUpperCase();

testIsEmptyWithNull();

testIsEmptyWithSpaces();

testIsEmptyWithText();

}

public static void testToUpperCase() {

setUp();

// Arrange

String input = "hello";

// Act

String result = stringUtils.toUpperCase(input);

// Assert

if ("HELLO".equals(result)) {

System.out.println("testToUpperCase PASSED");

} else {

System.out.println("testToUpperCase FAILED: Expected HELLO but got " + result);

}

tearDown();

}

public static void testIsEmptyWithNull() {

setUp();

boolean result = stringUtils.isEmpty(null);

if (result) {

System.out.println("TestIsEmptyWithNull PASSED");

} else {

System.out.println("testIsEmptyWithNull FAILED");

}

tearDown();

}

public static void testIsEmptyWithSpaces() {

setUp();

boolean result = stringUtils.isEmpty(" ");

if (result) {

System.out.println("testIsEmptyWithSpaces PASSED");

} else {

System.out.println("testIsEmptyWithSpaces FAILED");

}

tearDown();

}

public static void testIsEmptyWithText() {

setUp();

boolean result = stringUtils.isEmpty("hi");

if (!result) {

System.out.println("testIsEmptyWithText PASSED");

} else {

System.out.println("testIsEmptyWithText FAILED");

}

tearDown();

}

}

**OUTPUT:**

