**Cognizant\_Digital Nurture 4.0\_Deep Skilling**

**Programming Languages**

**Module 4 – Test driven development and Logging framework**

**JUnit Testing Exercises**

**Project Name: JUintExample**

**Exercise 1: Setting Up Junit**

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

**CalculatorTest.java**

import static org.junit.Assert.\*;

import org.junit.Test;

public class CalculatorTest {

*@Test*

public void testAdd() {

Calculator calc = new Calculator();

*assertEquals*(5, calc.add(2, 3));

}

*@Test*

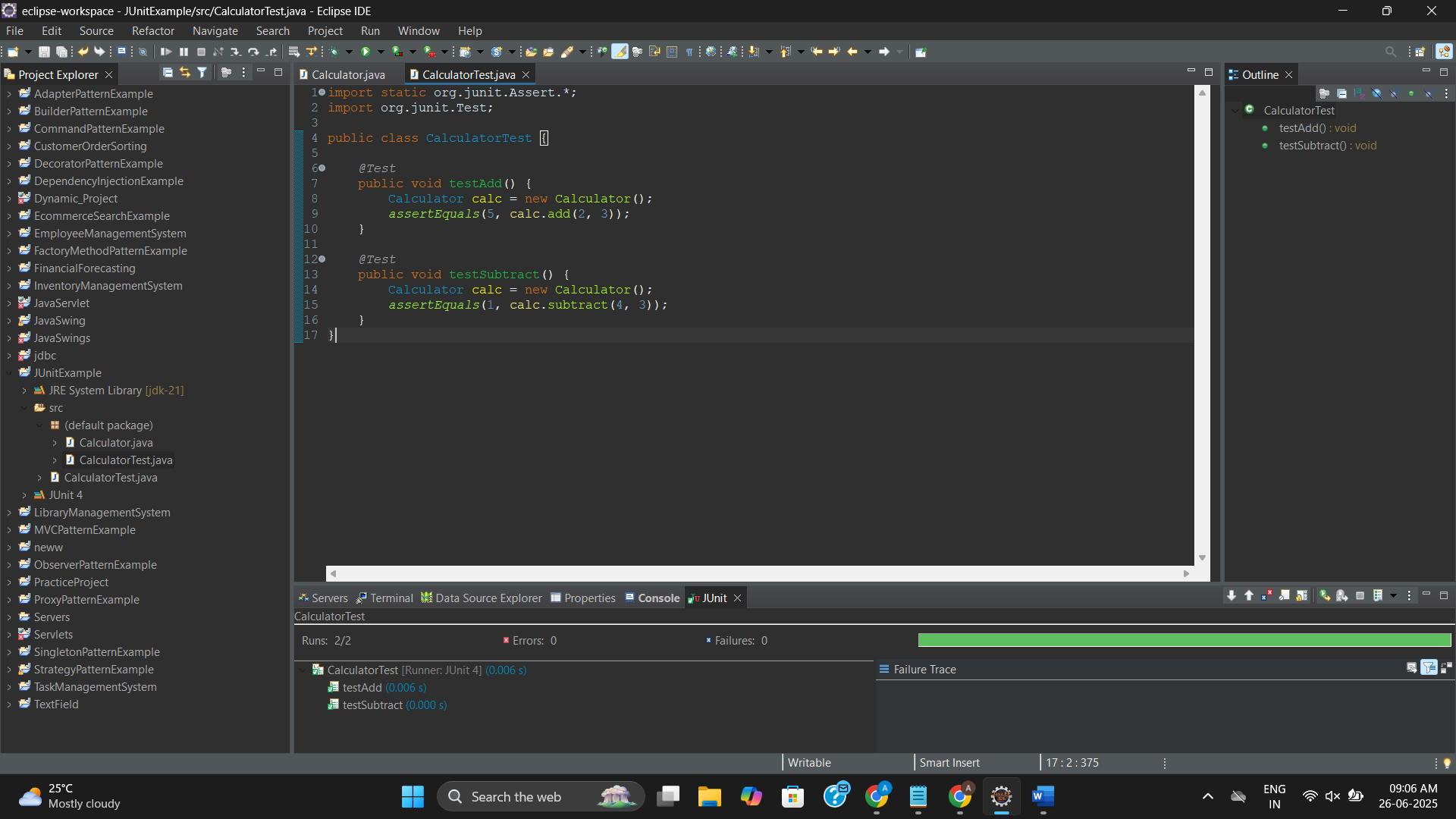
public void testSubtract() {

Calculator calc = new Calculator();

*assertEquals*(1, calc.subtract(4, 3));

}

}

****

**Exercise 2: Writing Basic JUnit Tests**

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

public int multiply(int a, int b) {

return a \* b;

}

public int divide(int a, int b) {

if (b == 0) throw new IllegalArgumentException("Division by zero");

return a / b;

}

}

**CalculatorTest.java**

import static org.junit.Assert.\*;

import org.junit.Test;

public class CalculatorTest {

*@Test*

public void testAdd() {

Calculator calc = new Calculator();

*assertEquals*(5, calc.add(2, 3));

}

*@Test*

public void testSubtract() {

Calculator calc = new Calculator();

*assertEquals*(1, calc.subtract(4, 3));

}

*@Test*

public void testMultiply() {

Calculator calc = new Calculator();

*assertEquals*(15, calc.multiply(5, 3));

}

*@Test*

public void testDivide() {

Calculator calc = new Calculator();

*assertEquals*(2, calc.divide(6, 3));

}

*@Test*(expected = IllegalArgumentException.class)

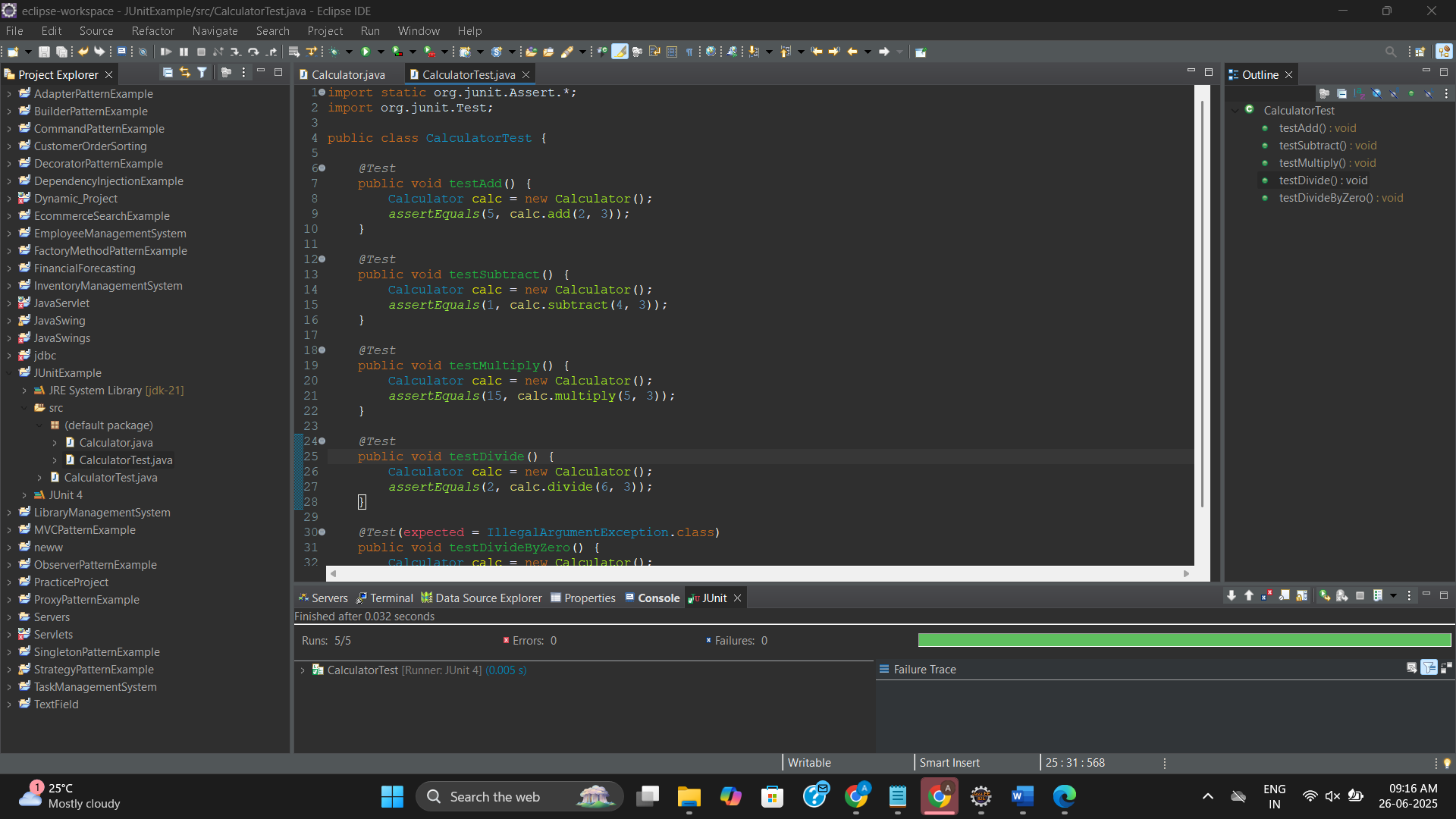
public void testDivideByZero() {

Calculator calc = new Calculator();

calc.divide(10, 0);

}

}

****

**Exercise 3: JUnit Assertions**

**AssertionsTest.java**

// AssertionsTest.java

import static org.junit.Assert.\*;

import org.junit.Test;

public class AssertionsTest {

*@Test*

public void testEquals() {

*assertEquals*("Sum should be 5", 5, 2 + 3);

}

*@Test*

public void testTrue() {

*assertTrue*("Should be true", 10 > 1);

}

*@Test*

public void testFalse() {

*assertFalse*("Should be false", 2 > 10);

}

*@Test*

public void testNull() {

String str = null;

*assertNull*("Should be null", str);

}

*@Test*

public void testNotNull() {

String str = "JUnit";

*assertNotNull*("Should not be null", str);

}

*@Test*

public void testSameObject() {

String a = "Hello";

String b = a;

*assertSame*("Should be same object", a, b);

}

*@Test*

public void testNotSameObject() {

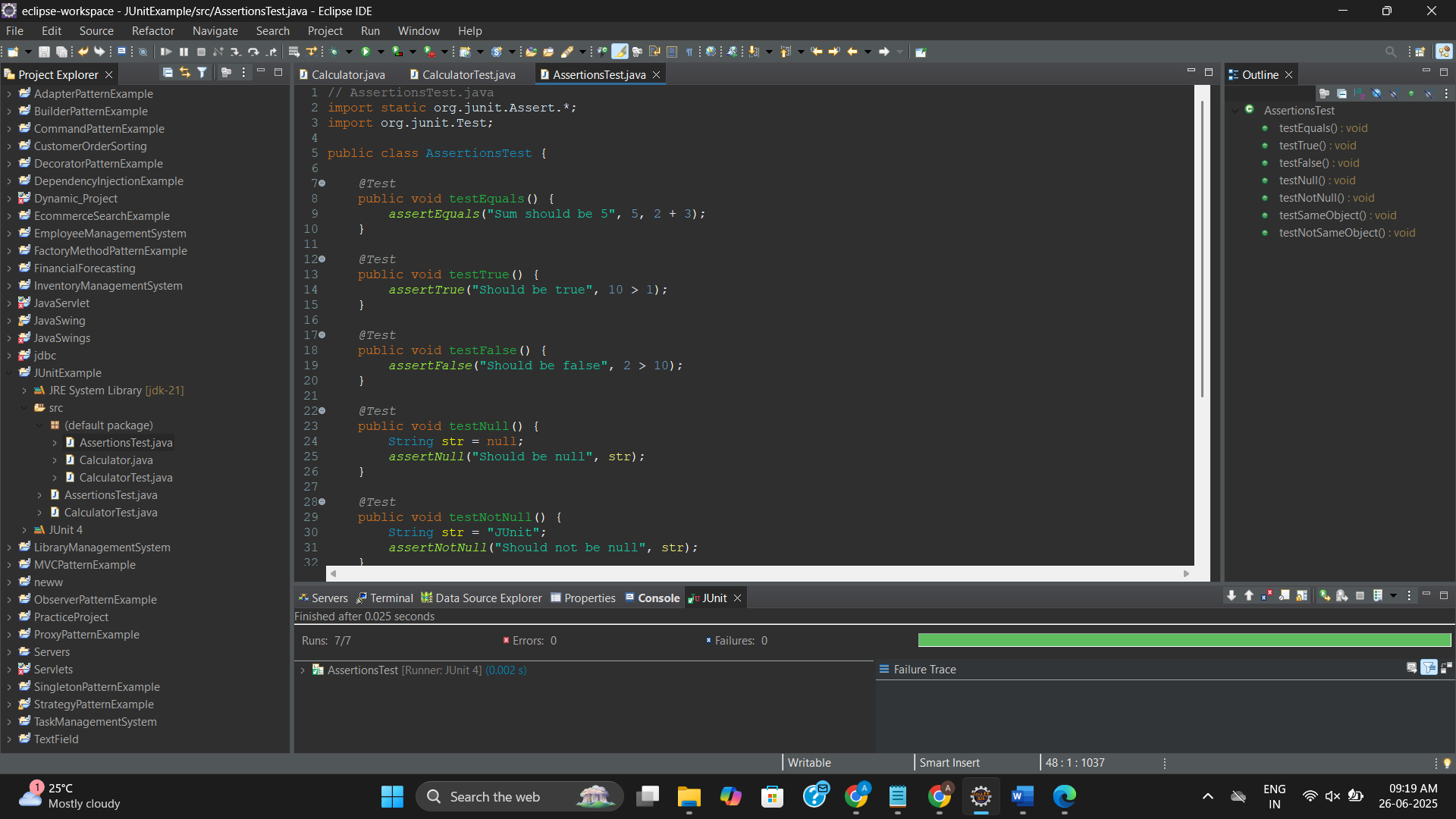
String a = new String("Hello");

String b = new String("Hello");

*assertNotSame*("Should not be same object", a, b);

}

}

****

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

**CalculatorAAATest.java**

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

public class CalculatorAAATest {

private Calculator calculator;

//Setup - runs before every test

*@Before*

public void setUp() {

calculator = new Calculator();

}

// Teardown - runs after every test

*@After*

public void tearDown() {

calculator = null;

}

*@Test*

public void testAdd() {

// Arrange done in setUp

// Act

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

// Assert

*assertEquals*(5, result);

}

*@Test*

public void testSubtract() {

int result = calculator.subtract(4, 3);

*assertEquals*(1, result);

}

*@Test*

public void testMultiply() {

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

*assertEquals*(15, result);

}

*@Test*

public void testDivide() {

int result = calculator.divide(6, 3);

*assertEquals*(2, result);

}

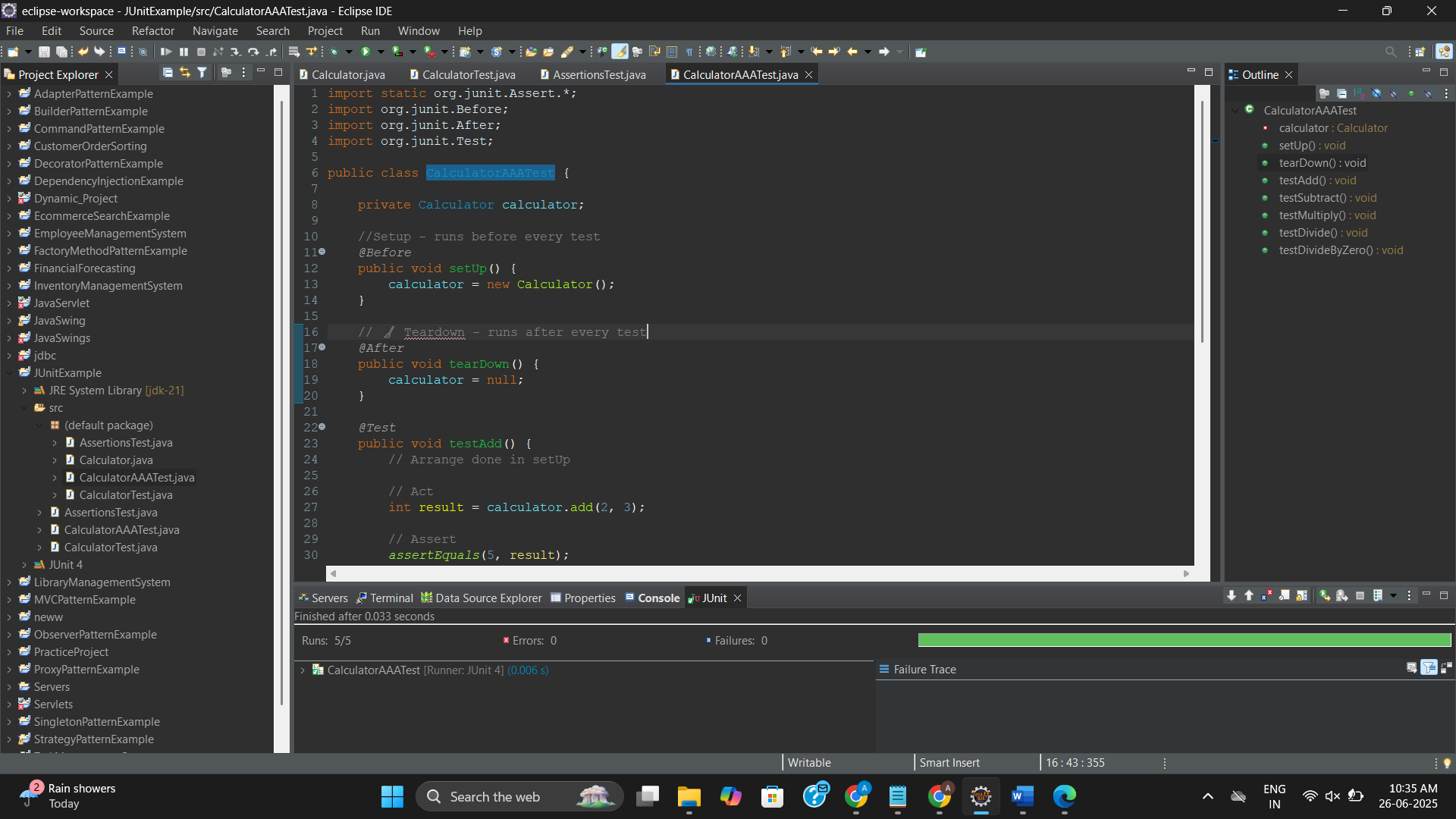
*@Test*(expected = IllegalArgumentException.class)

public void testDivideByZero() {

calculator.divide(10, 0);

}

}

****

**Advanced JUnit Testing Exercises**

**Project Name: AdvancedJUnitExample**

**Exercise 1: Parameterized Tests**

**EvenChecker.Java**

public class EvenChecker {

public boolean isEven(int number) {

return number % 2 == 0;

}

}

**EvenCheckerTest.Java**

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

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;

public class EvenCheckerTest {

*@ParameterizedTest*

*@ValueSource*(ints = {2, 4, 6, 8, 10})

void testEvenNumbers(int number) {

EvenChecker checker = new EvenChecker();

*assertTrue*(checker.isEven(number));

}

*@ParameterizedTest*

*@ValueSource*(ints = {1, 3, 5, 7, 9})

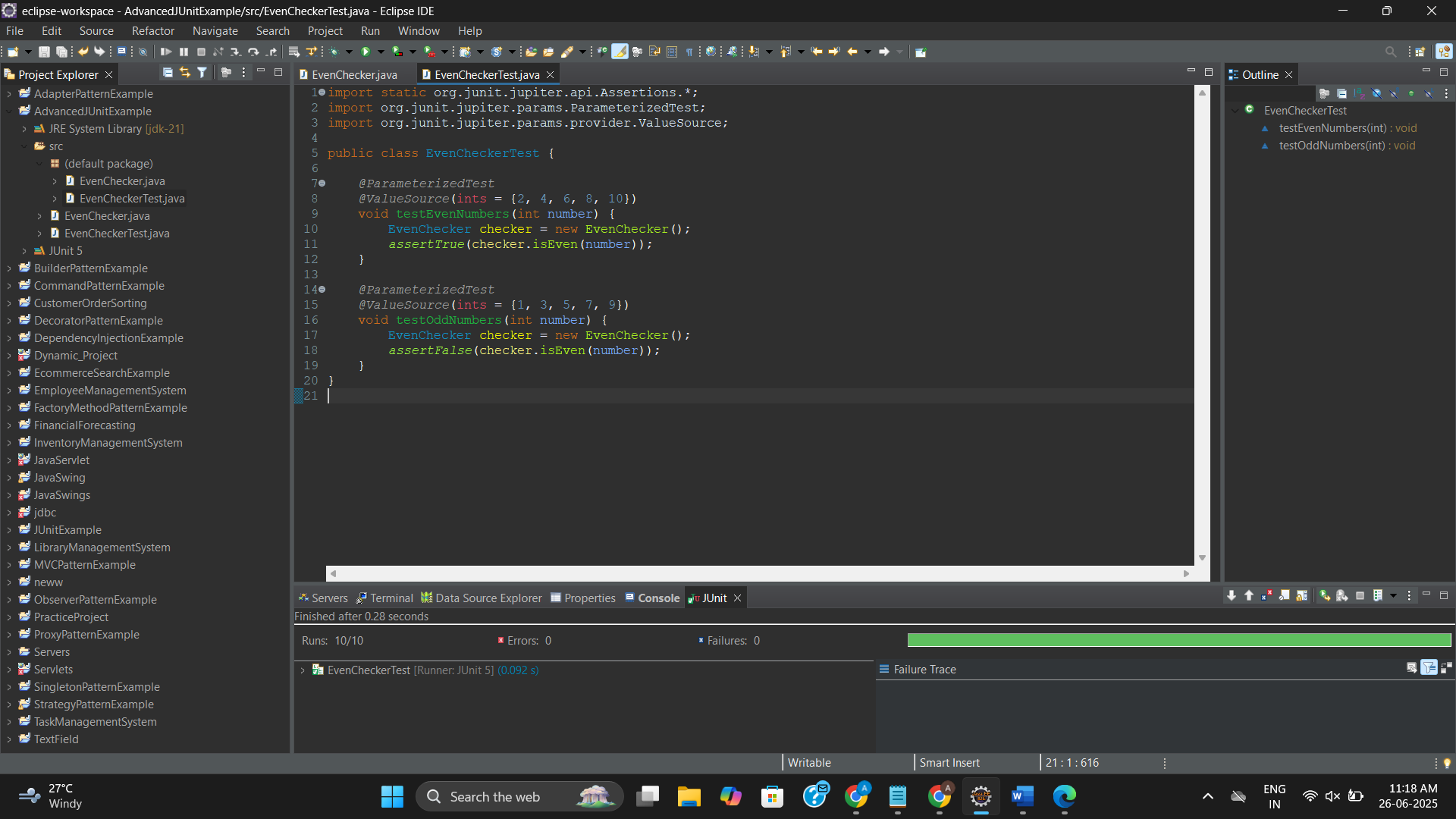
void testOddNumbers(int number) {

EvenChecker checker = new EvenChecker();

*assertFalse*(checker.isEven(number));

}

}

****

**Exercise 2: Test Suites and Categories**

**AllTests.java**

// AllTests.java

import org.junit.platform.suite.api.SelectClasses;

import org.junit.platform.suite.api.Suite;

// This tells JUnit to treat this as a test suite

*@Suite*

*@SelectClasses*({

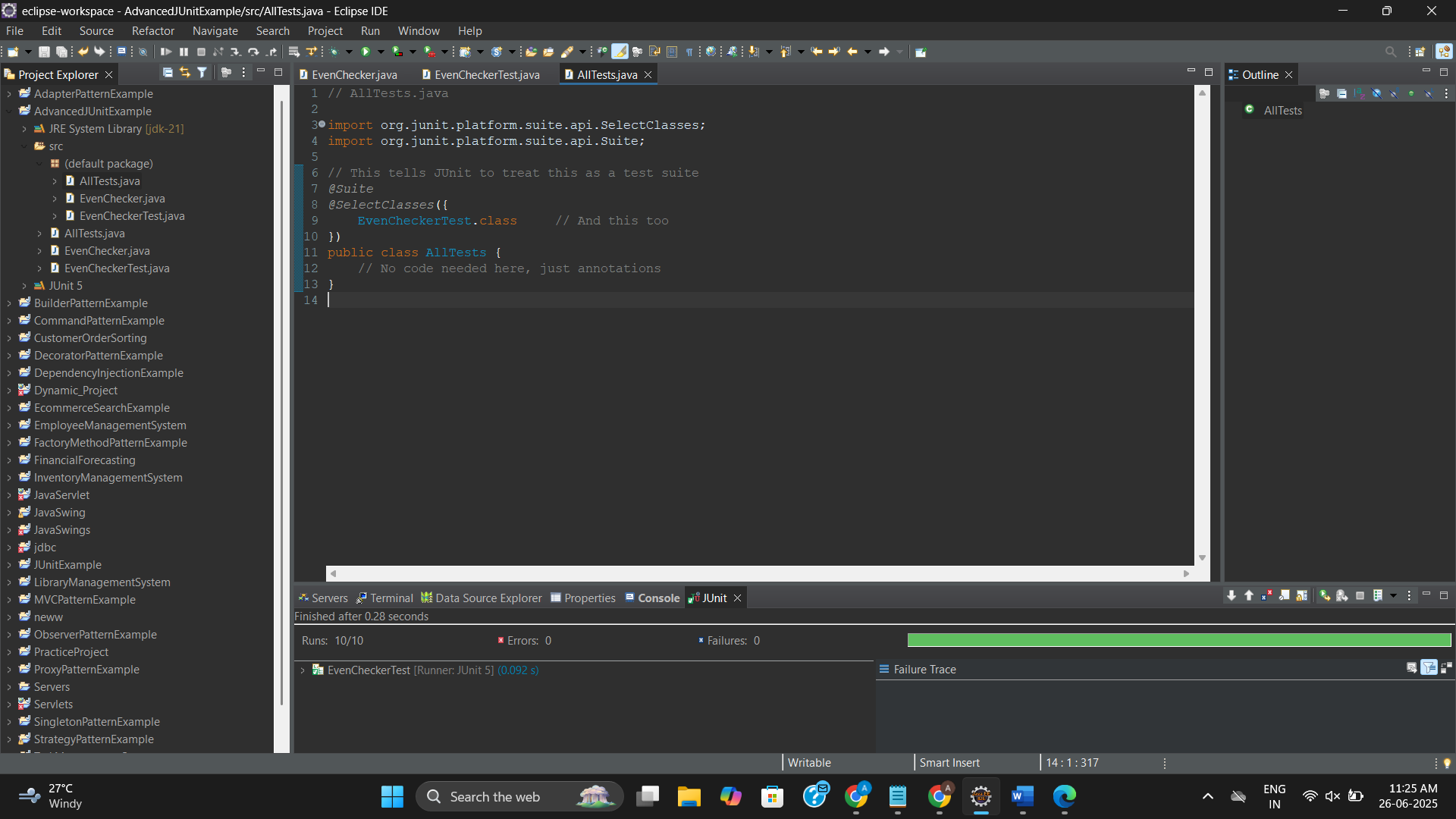
EvenCheckerTest.class // And this too

})

public class AllTests {

// No code needed here, just annotations

}



**Exercise 3: Test Execution Order**

**OrderedTests.java**

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

@TestMethodOrder(MethodOrderer.OrderAnnotation.class)

public class OrderedTests {

@Test

@Order(1)

void testFirst() {

System.out.println("Running First Test");

}

@Test

@Order(3)

void testThird() {

System.out.println("Running Third Test");

}

@Test

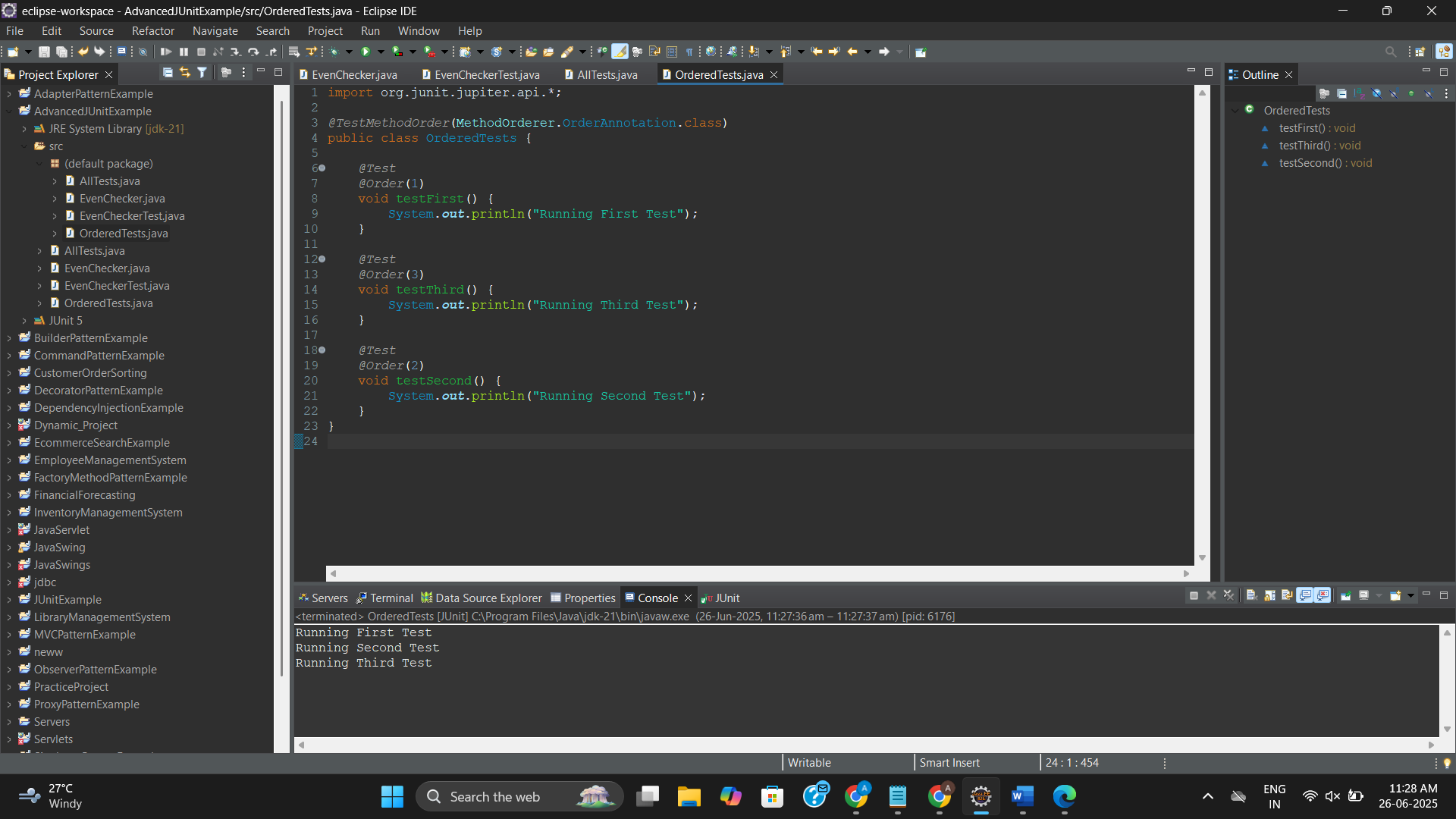
@Order(2)

void testSecond() {

System.out.println("Running Second Test");

}

}



**Exercise 4: Exception Testing**

**ExceptionThrower.java**

public class ExceptionThrower {

public void throwException() {

throw new IllegalArgumentException("Invalid input!");

}

}

**ExceptionThrowerTest.java**

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

import org.junit.jupiter.api.Test;

public class ExceptionThrowerTest {

*@Test*

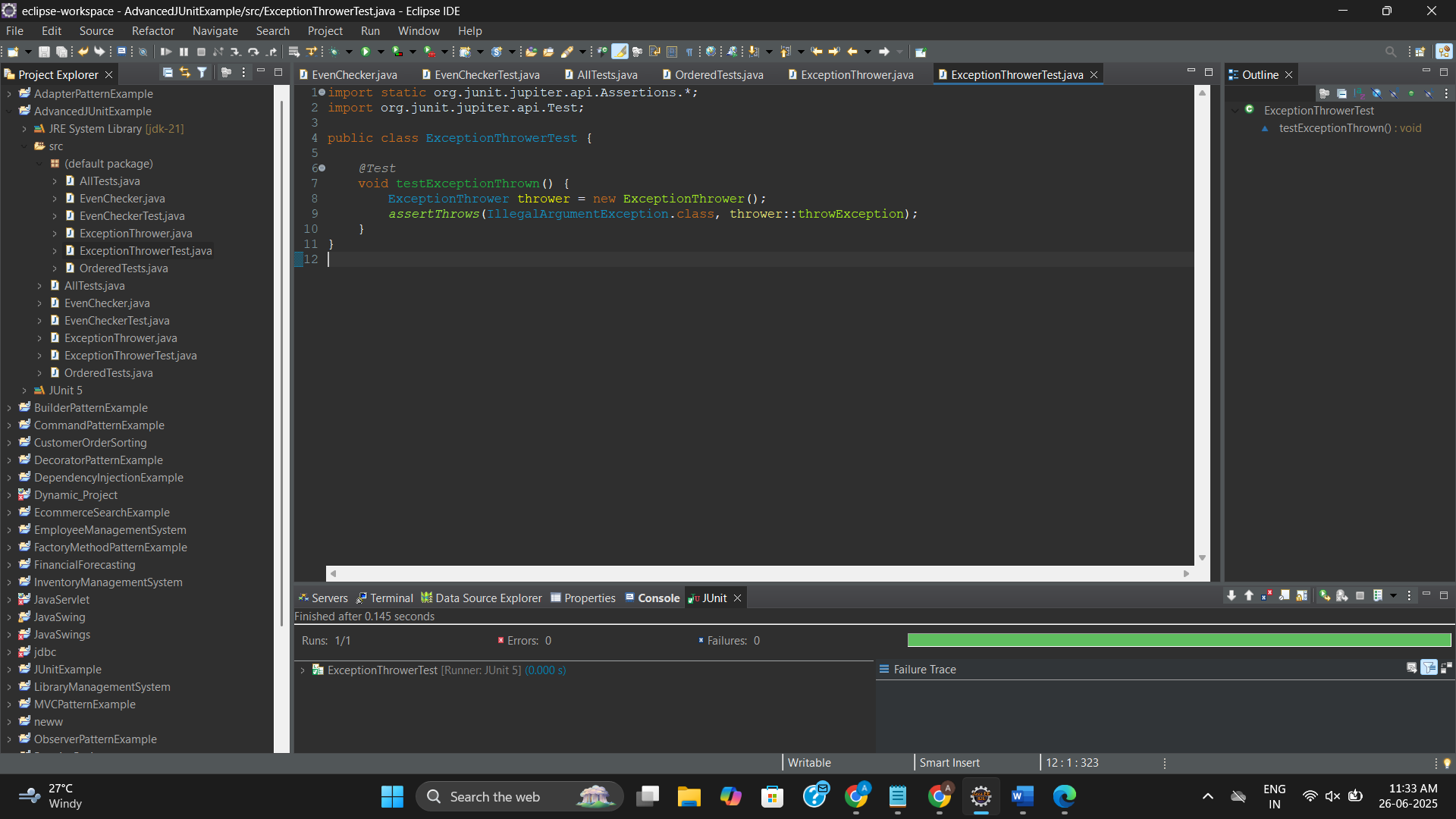
void testExceptionThrown() {

ExceptionThrower thrower = new ExceptionThrower();

*assertThrows*(IllegalArgumentException.class, thrower::throwException);

}

}



**Exercise 5: Timeout and Performance Testing**

**PerformanceTester.java**

public class PerformanceTester {

public void performTask() {

try {

Thread.*sleep*(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

**PerformanceTesterTest.java**

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

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.Timeout;

import java.util.concurrent.TimeUnit;

public class PerformanceTesterTest {

*@Test*

*@Timeout*(value = 200, unit = *TimeUnit*.***MILLISECONDS***)

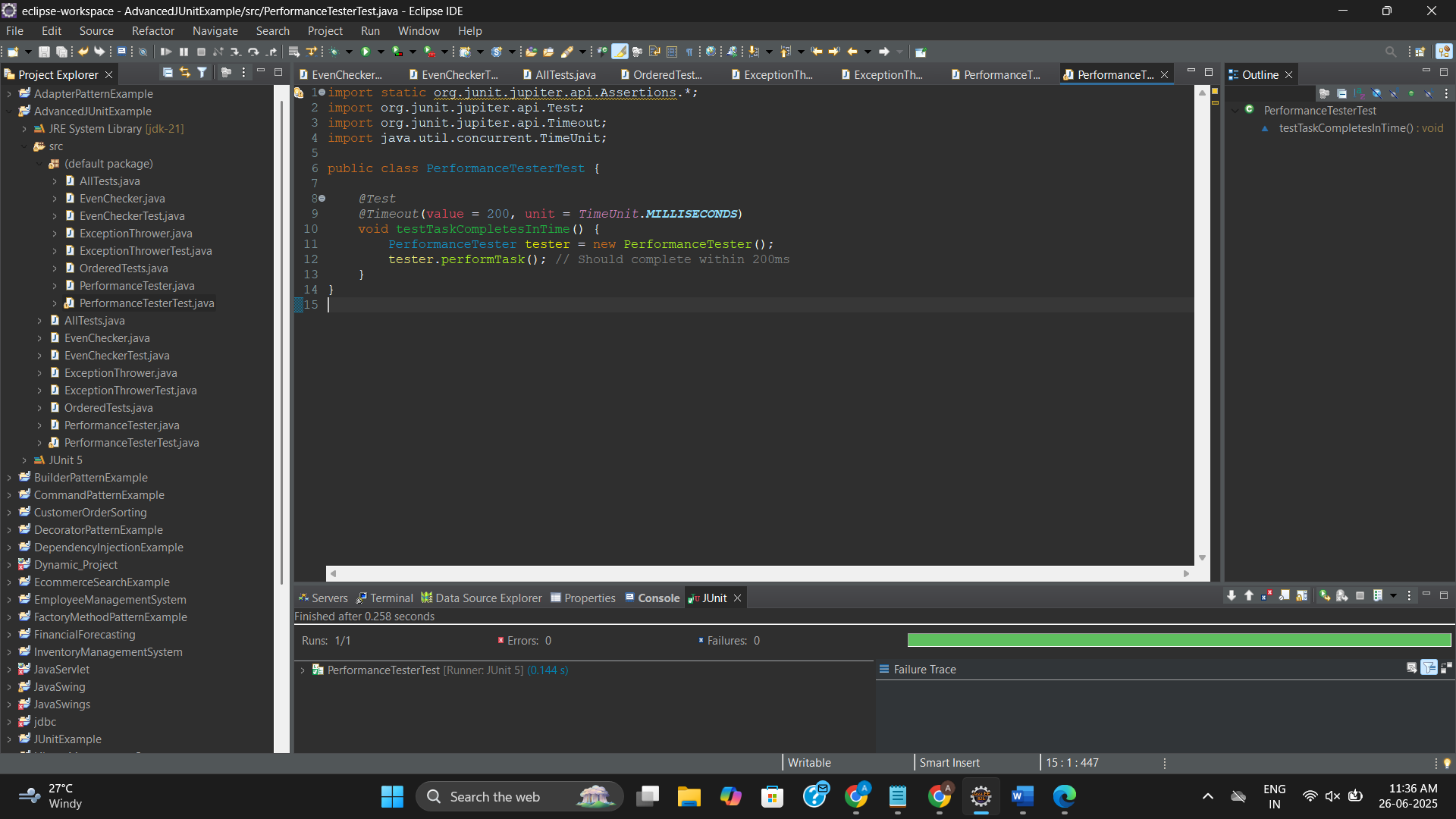
void testTaskCompletesInTime() {

PerformanceTester tester = new PerformanceTester();

tester.performTask(); // Should complete within 200ms

}

}



**Mockito Hands-On Exercises**

**Project Name: MockitoProject**

**Exercise 1: Mocking and Stubbing**

**ExternalApi.java**

public interface ExternalApi {

String getData();

}

**MyService.java**

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

*@Test*

public void testExternalApi() {

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

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

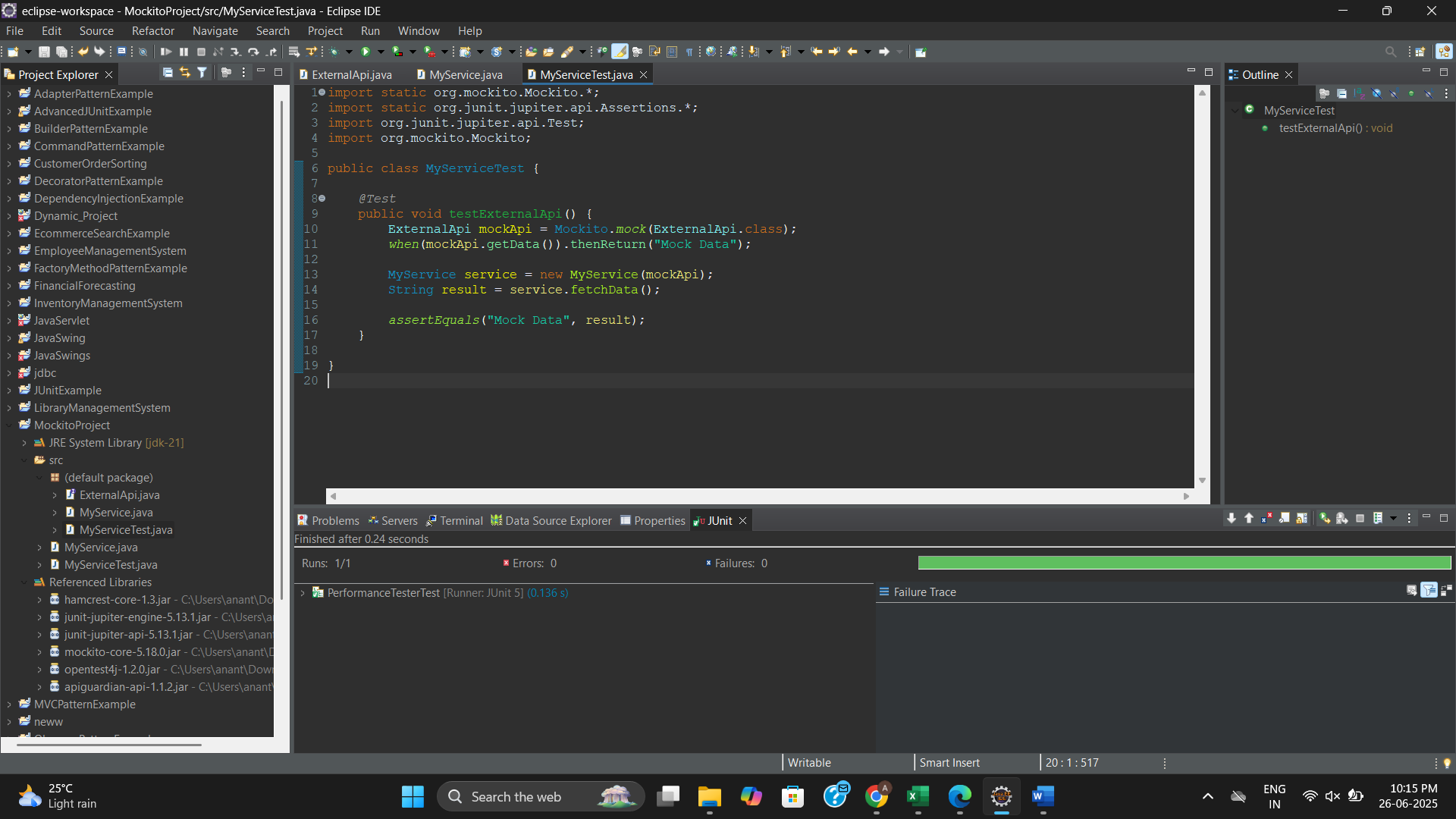
MyService service = new MyService(mockApi);

String result = service.fetchData();

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

}

}

****

**Exercise 2: Verifying Interactions**

**ExternalApi.java**

public interface ExternalApi {

String getData();

}

**MyService.java**

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

*@Test*

public void testExternalApi() {

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

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

MyService service = new MyService(mockApi);

String result = service.fetchData();

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

}

*@Test*

public void testVerifyInteraction() {

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

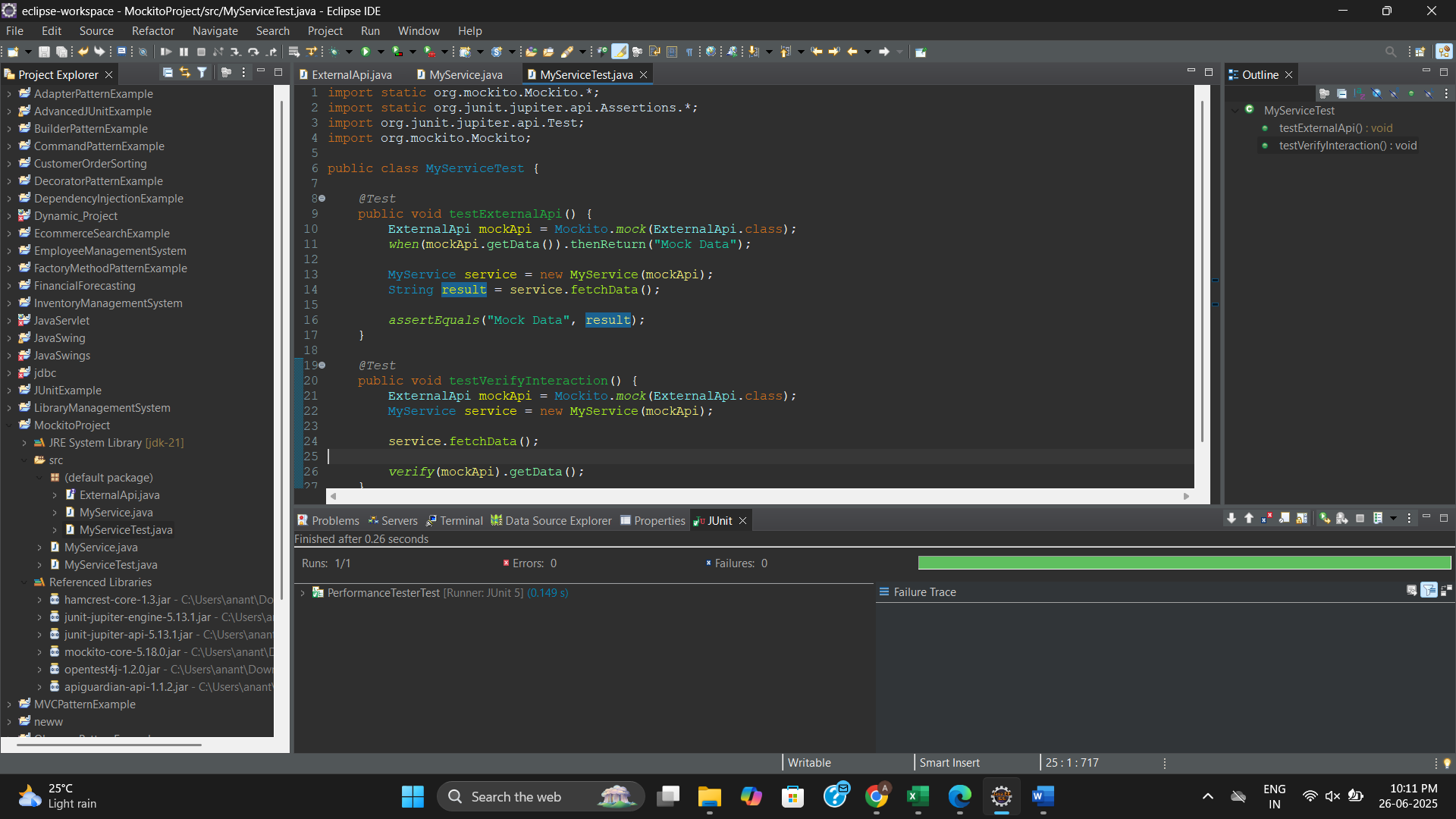
MyService service = new MyService(mockApi);

service.fetchData();

*verify*(mockApi).getData();

}

}



**Logging using SLF4J**

**Project Name: SLF4JExample**

**Exercise 1: Logging Error Messages and Warning Levels**

**LoggingExample.java**

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggingExample {

private static final Logger ***logger*** = LoggerFactory.*getLogger*(LoggingExample.class);

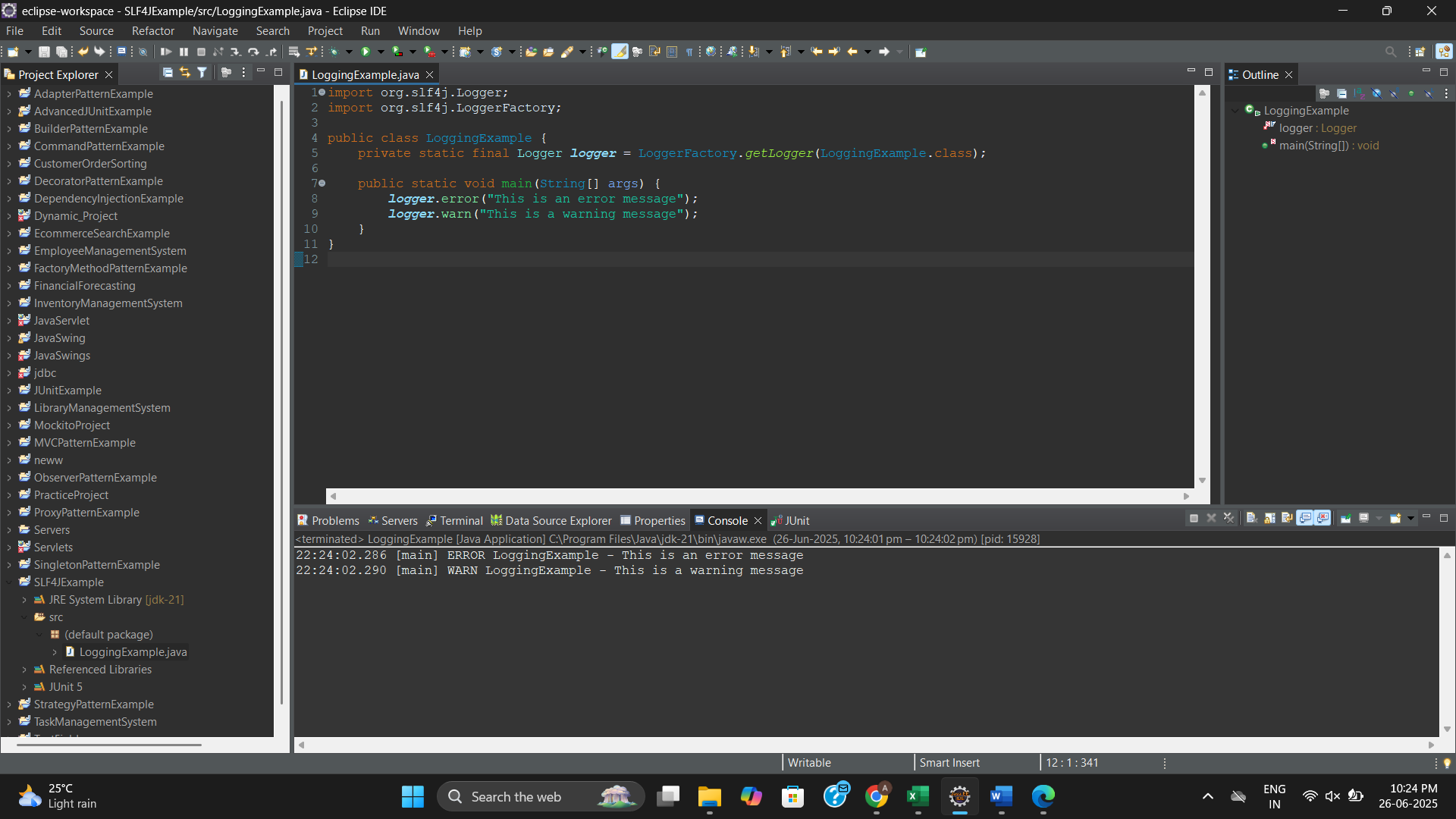
public static void main(String[] args) {

***logger***.error("This is an error message");

***logger***.warn("This is a warning message");

}

}

****