**WEEK - 2**

**PL/SQL programming**

**Exercise 1: Control Structures**

**Scenario 1: Discount for Senior Customers**

BEGIN

FOR cust IN (SELECT CustomerID, Age FROM Customers) LOOP

IF cust.Age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

**Scenario 2: Promote to VIP Based on Balance**

BEGIN

FOR cust IN (SELECT CustomerID, Balance FROM Customers) LOOP

IF cust.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

**Scenario 3: Send Loan Due Reminders**

DECLARE

v\_due\_date DATE;

v\_name VARCHAR2(100);

BEGIN

FOR loan IN (

SELECT l.CustomerID, l.DueDate, c.Name

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan for ' || loan.Name ||

' is due on ' || TO\_CHAR(loan.DueDate, 'DD-MON-YYYY'));

END LOOP;

END;

**Exercise 3: Stored Procedures**

**Scenario 1: Process Monthly Interest**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'SAVINGS';

COMMIT;

END;

**Scenario 2: Update Employee Bonus**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_dept\_id IN NUMBER,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* p\_bonus\_percent / 100)

WHERE DepartmentID = p\_dept\_id;

COMMIT;

END;

**Scenario 3: Transfer Funds Between Accounts**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_from\_account;

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in source account.');

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account;

COMMIT;

END;

**JUnit Basic Testing Exercises**

**Exercise 1: Setting Up JUnit**

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorTest.java**

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CalculatorTest {

@Test

public void testAdd() {

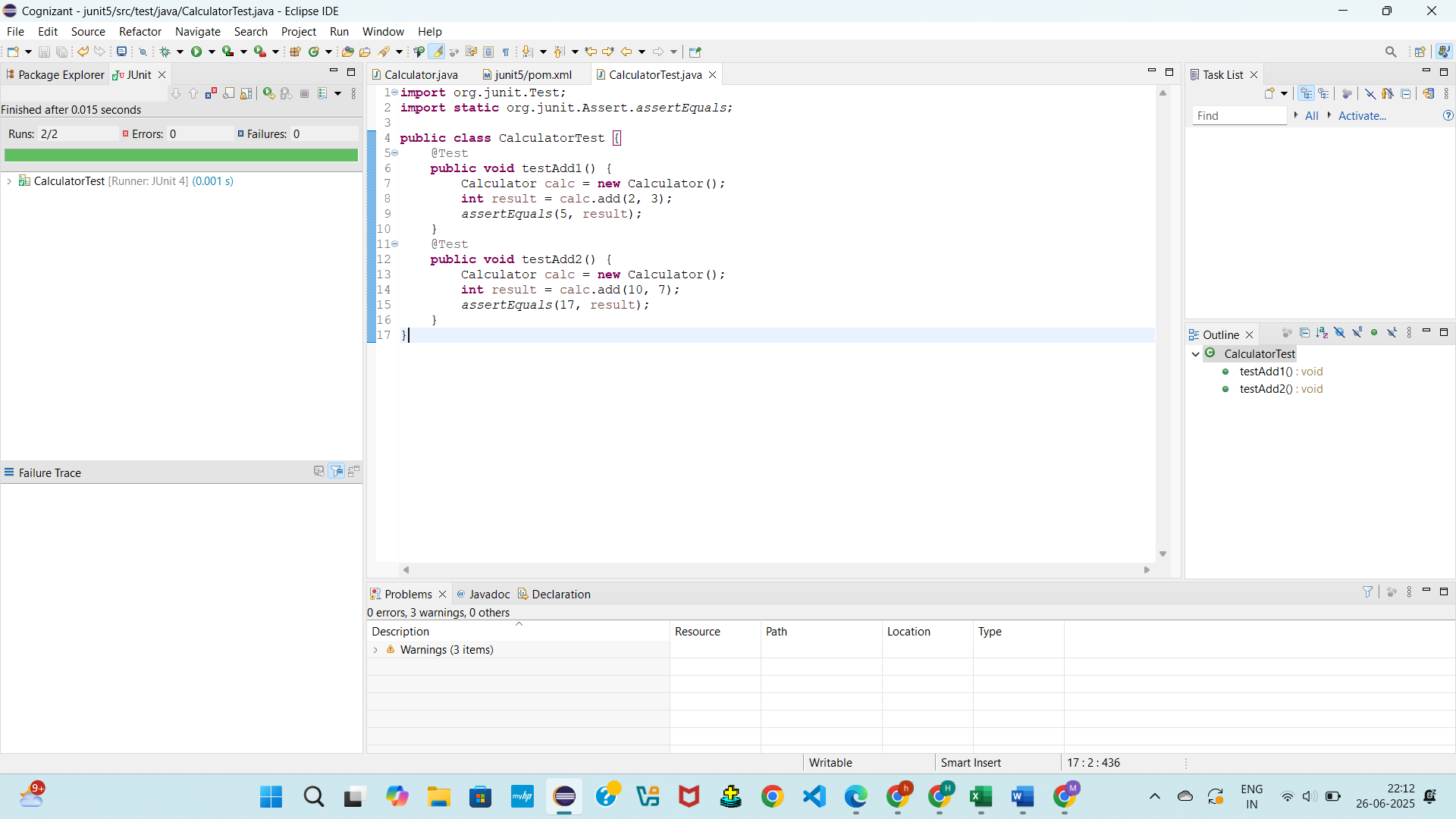
Calculator calc = new Calculator();

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

}

}

**OUTPUT:**

****

**Exercise 3: Assertions in JUnit**

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public boolean isPositive(int number) {

return number > 0;

}

public Object getNullValue() {

return null;

}

public Object getNonNullValue() {

return new Object();

}

}

**AssertionsTest.java**

import static org.junit.Assert.\*;

import org.junit.Test;

public class AssertionsTest {

@Test

public void testAssertions() {

// Assert equals

assertEquals(5, 2 + 3);

// Assert true

assertTrue(5 > 3);

// Assert false

assertFalse(5 < 3);

// Assert null

assertNull(null);

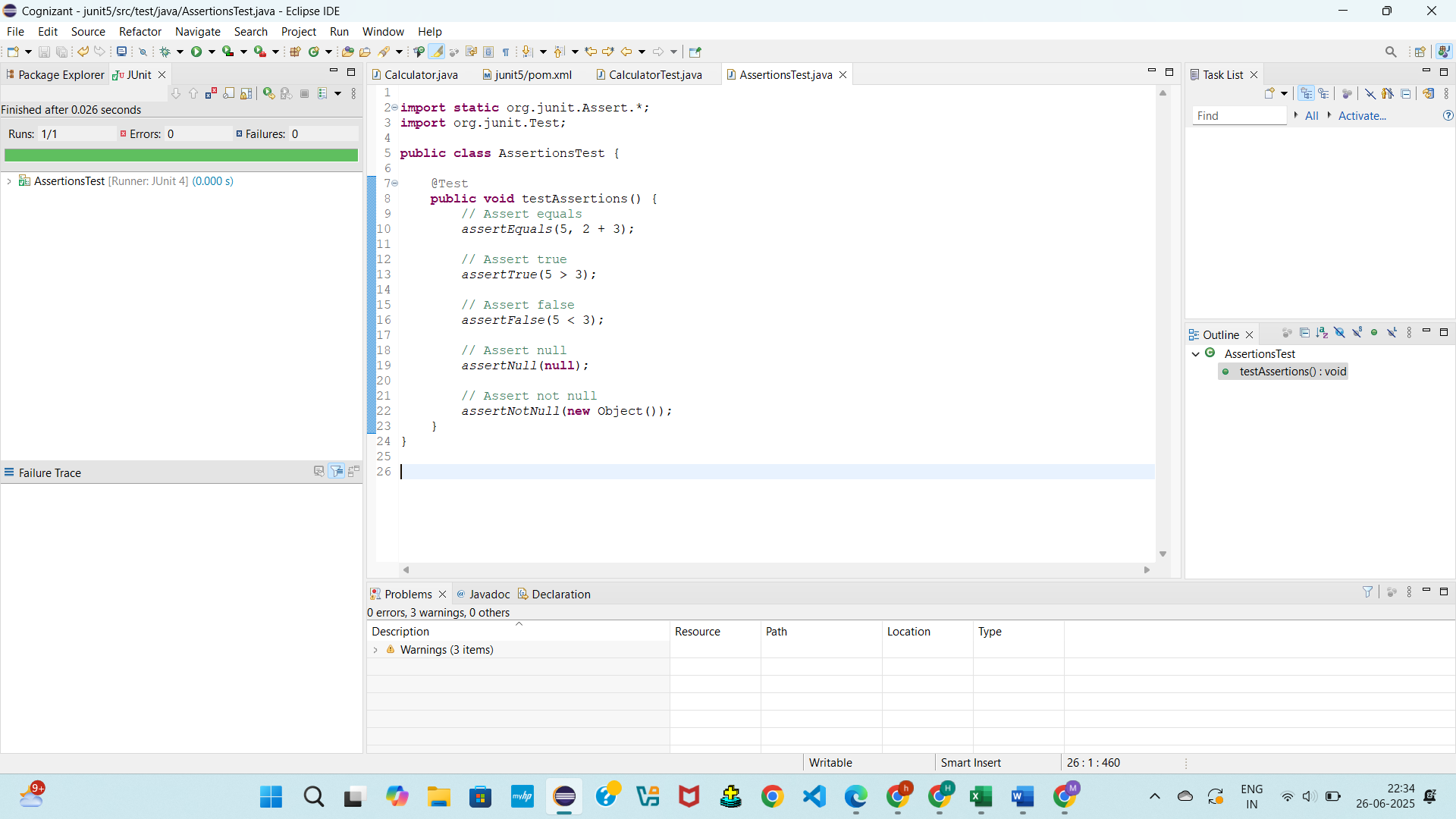
// Assert not null

assertNotNull(new Object());

}

}

**OUTPUT:**

****

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

**Calculator.java**  
public class Calculator {

public int multiply(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.Before;

import org.junit.After;

import org.junit.Test;

public class CalculatorTest {

private Calculator calc;

@Before

public void setUp() {

calc = new Calculator();

System.out.println("Setup done");

}

@After

public void tearDown() {

calc = null;

System.out.println("Teardown done");

}

@Test

public void testMultiply() {

// Arrange

int a = 4;

int b = 5;

// Act

int result = calc.multiply(a, b);

// Assert

assertEquals(20, result);

}

@Test

public void testSubtract() {

int a = 10;

int b = 3;

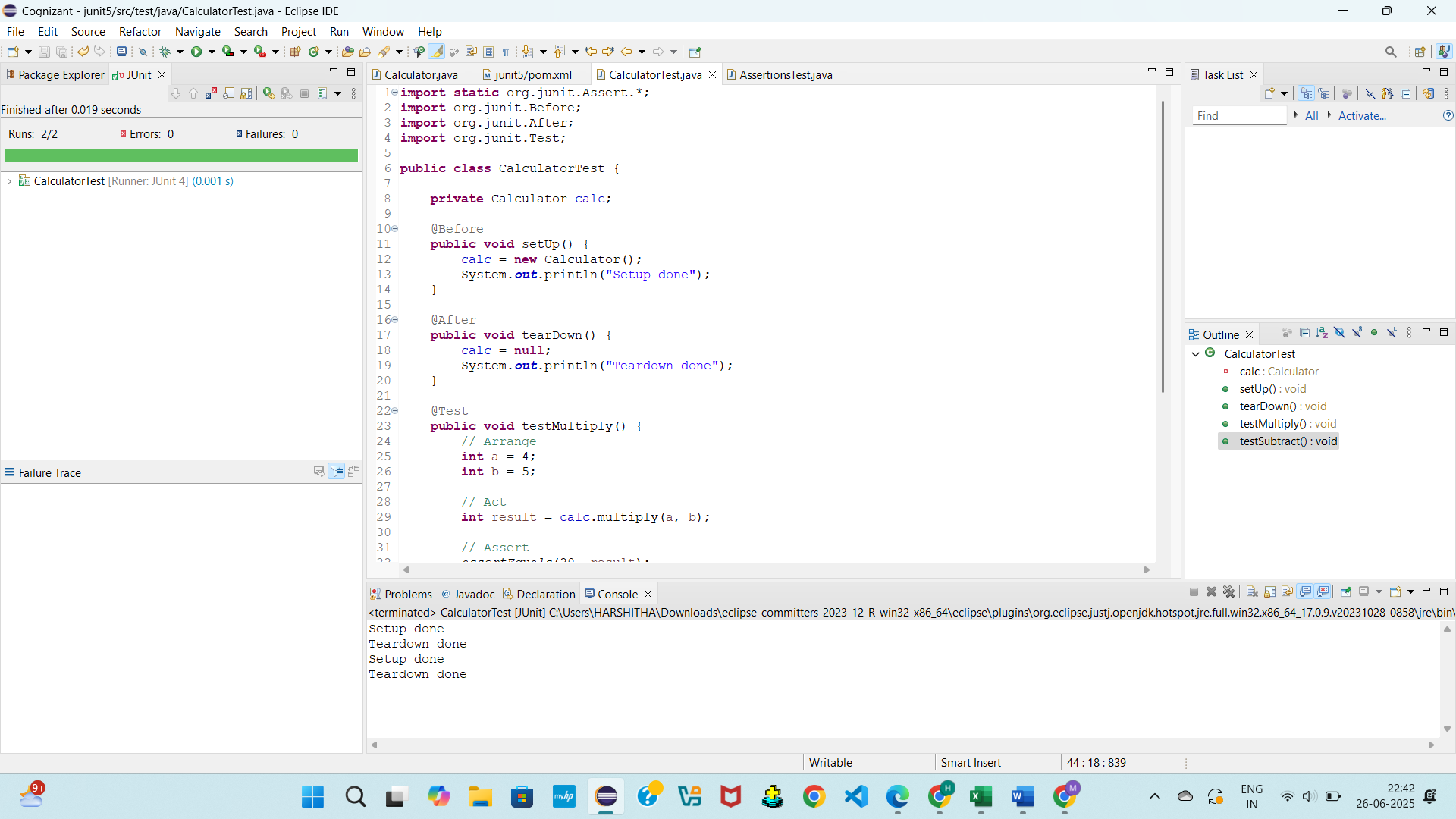
int result = calc.subtract(a, b);

assertEquals(7, result);

}

}

**OUTPUT:**



**Mockito exercises**

**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");

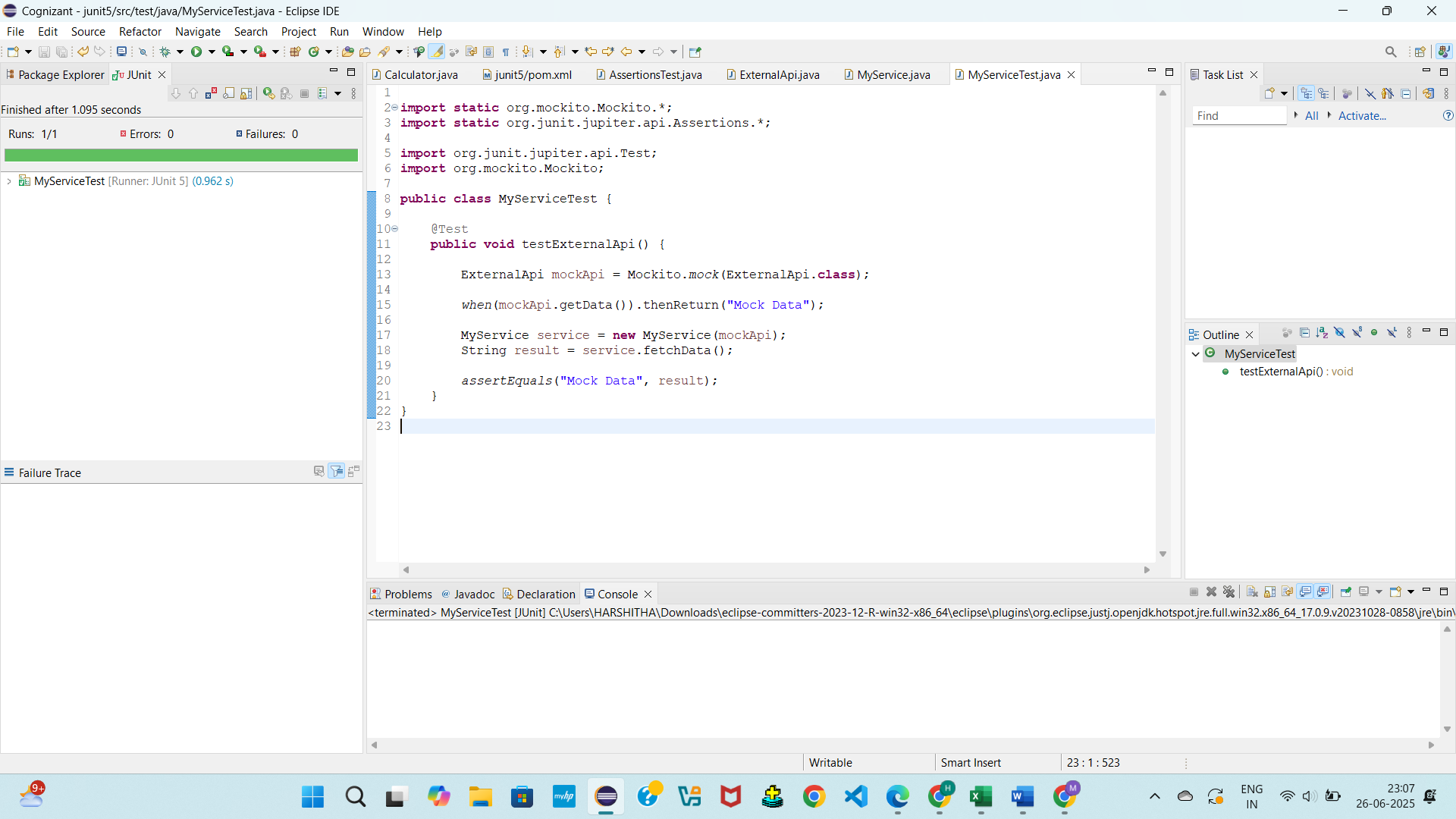
MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**OUTPUT:**  


**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 void fetchData() {

api.getData();

}

}

**MyServiceTest.java**

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);

MyService service = new MyService(mockApi);

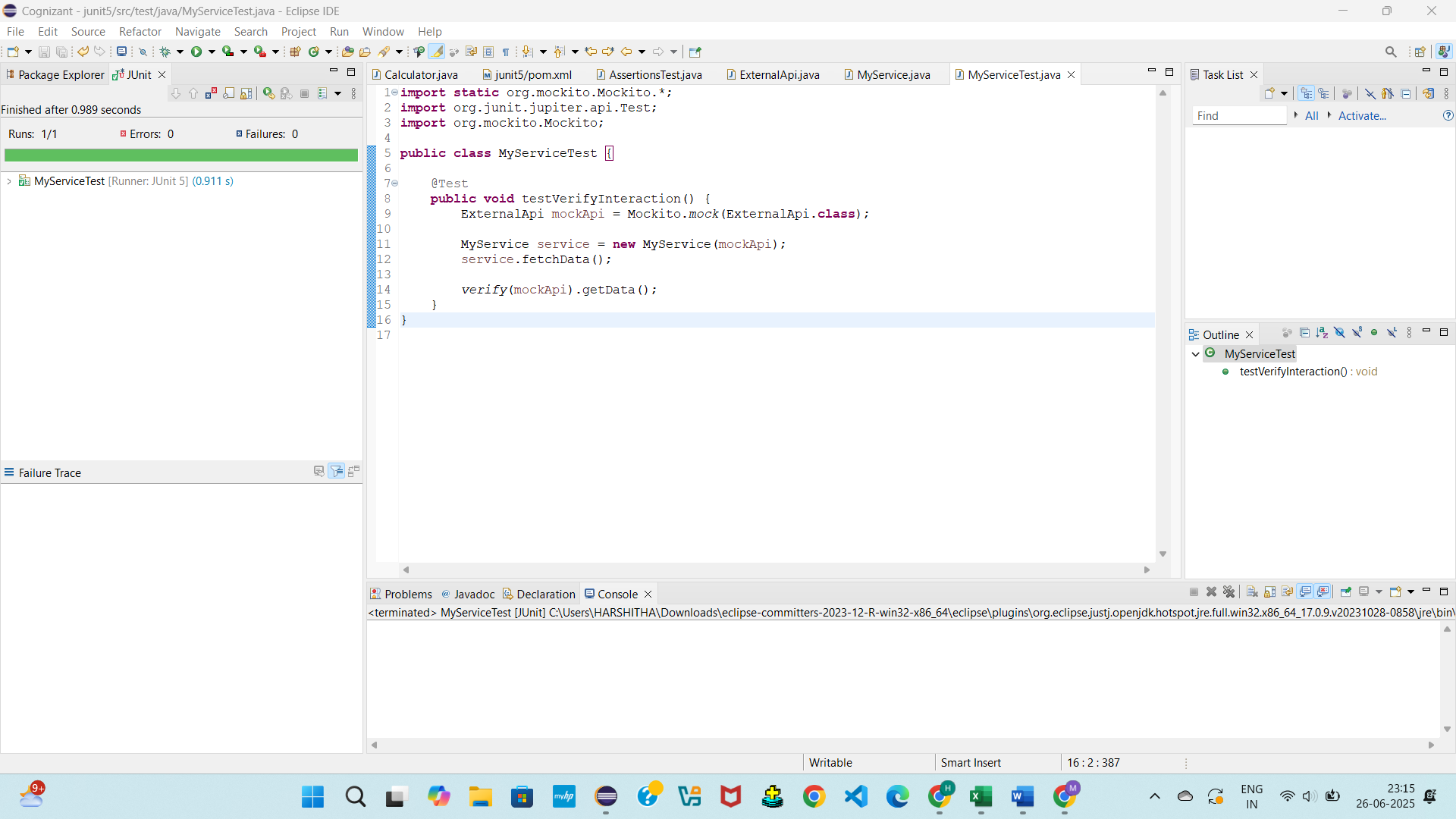
service.fetchData();

verify(mockApi).getData();

}

}

**OUTPUT:**



**SL4J Logging exercises**

**Exercise 1: Logging Error Messages and Warning Level**s

**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) {

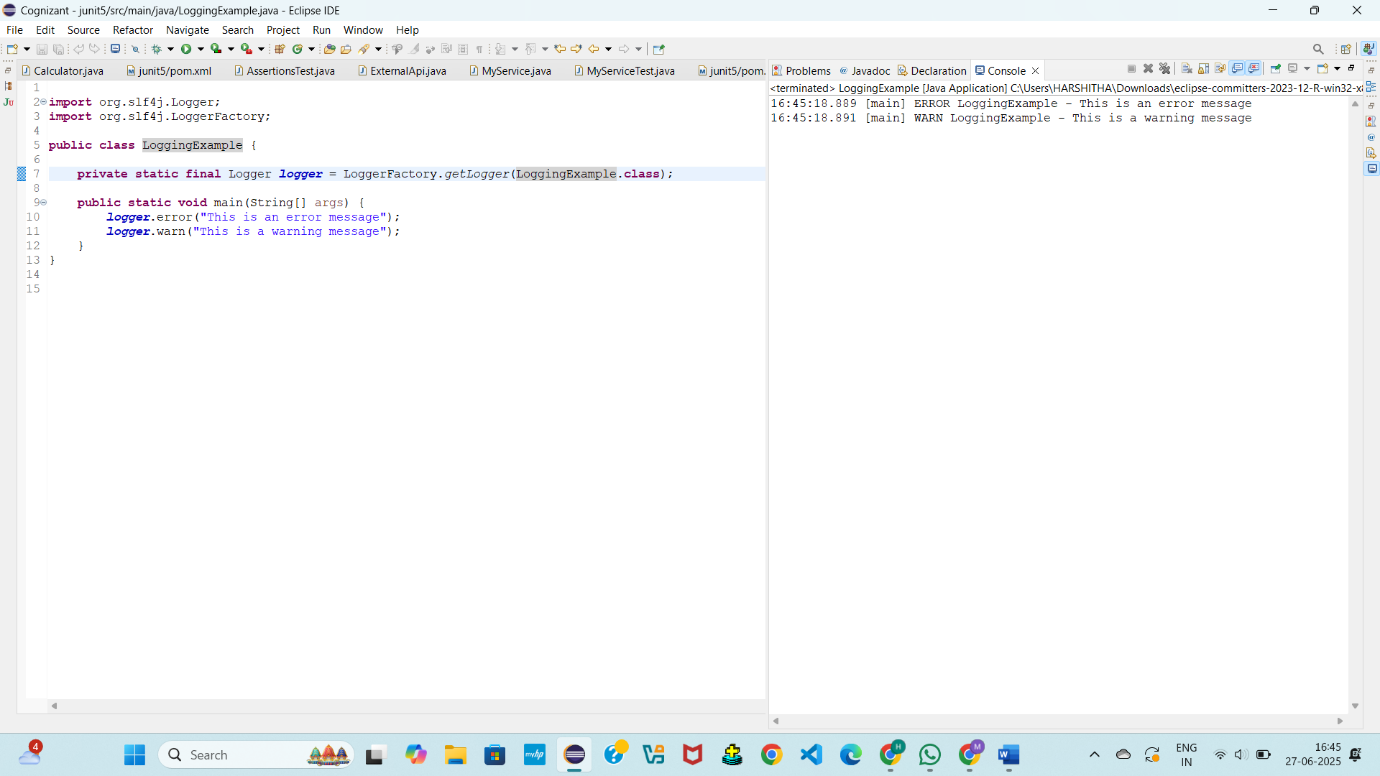
logger.error("This is an error message");

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

}

}

**OUTPUT:**

****