**Exercise 1: Control Structures**

Scenario 1: Discount for customers over 60 years old

DECLARE

    CURSOR cust\_cursor IS

        SELECT CustomerID, Age, LoanInterestRate

        FROM Customers

        WHERE Age > 60;

BEGIN

    FOR cust\_rec IN cust\_cursor LOOP

        UPDATE Customers

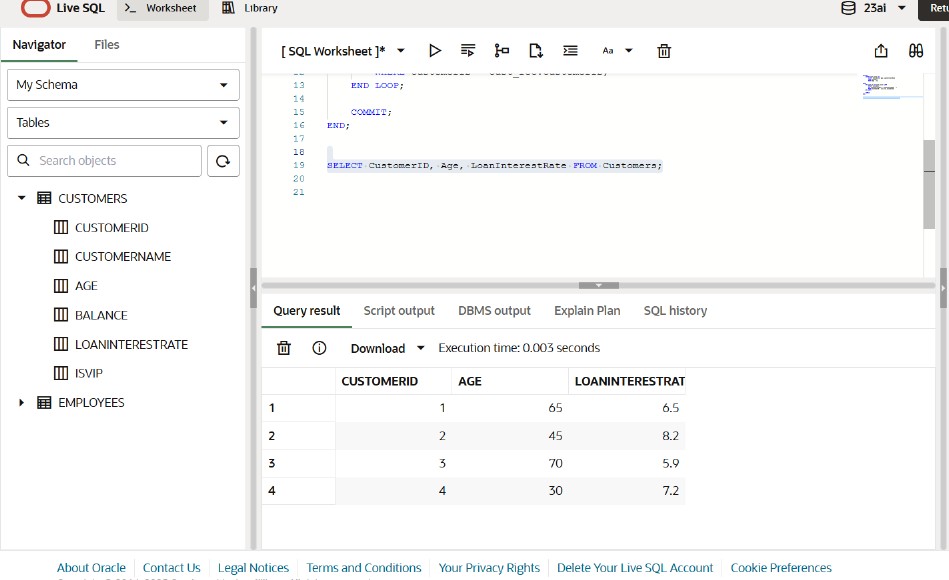
        SET LoanInterestRate = LoanInterestRate - 1

        WHERE CustomerID = cust\_rec.CustomerID;

    END LOOP;

    COMMIT;

END;



Scenario 2: Promote customers to VIP if balance > $10,000

DECLARE

CURSOR vip\_cursor IS

SELECT CustomerID, Balance

FROM Customers

WHERE Balance > 10000;

BEGIN

FOR vip\_rec IN vip\_cursor LOOP

UPDATE Customers

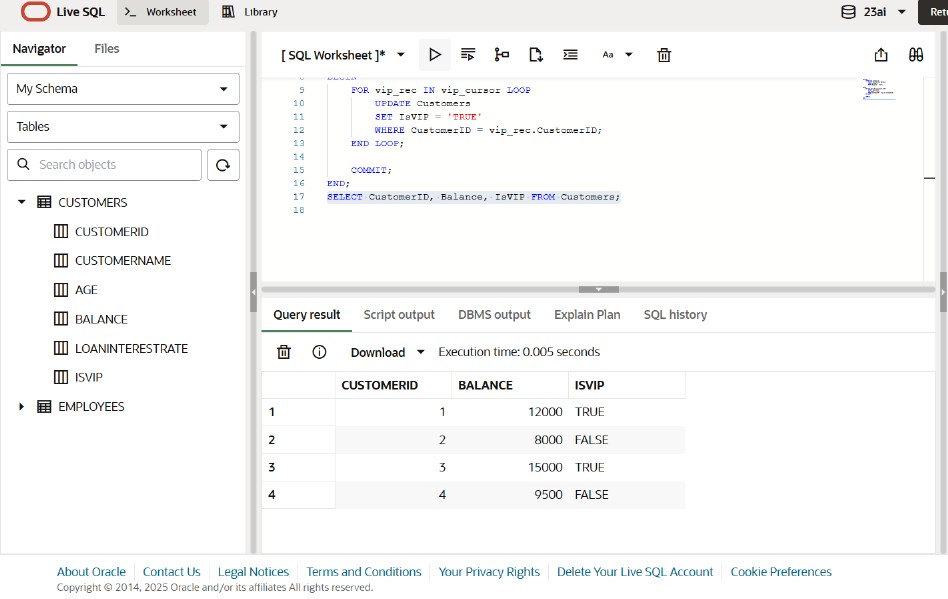
SET IsVIP = 'TRUE'

WHERE CustomerID = vip\_rec.CustomerID;

END LOOP;

COMMIT;

END;



Scenario 3: Reminders for loans due in next 30 days

DECLARE

CURSOR loan\_cursor IS

SELECT l.LoanID, l.DueDate, c.CustomerName

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate <= SYSDATE + 30;

BEGIN

FOR loan\_rec IN loan\_cursor LOOP

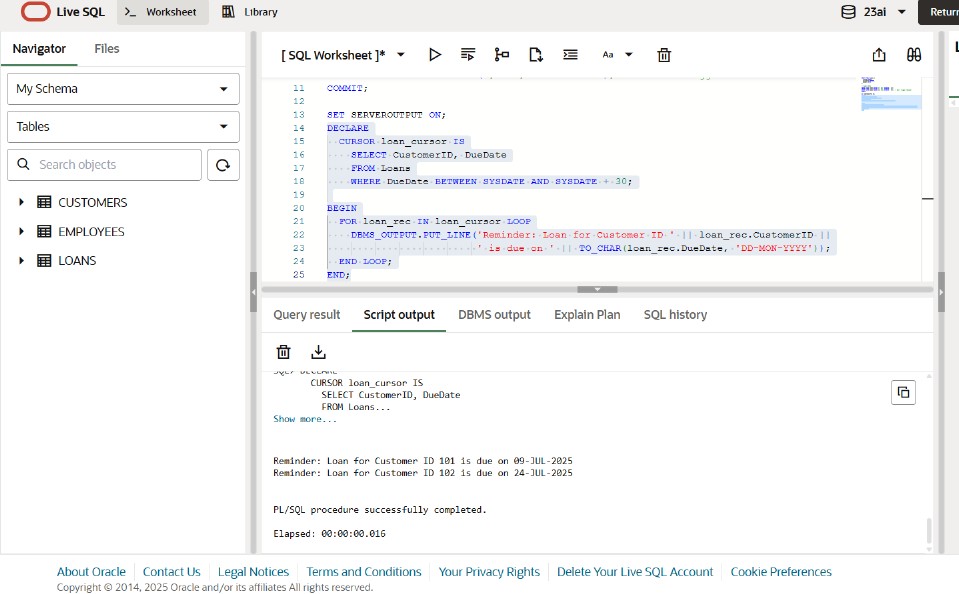
DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || loan\_rec.CustomerName ||

', your loan (ID: ' || loan\_rec.LoanID ||

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

END LOOP;

END;



**Exercise 3: Stored Procedures**

Scenario 1: Process Monthly Interest for Savings Accounts

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

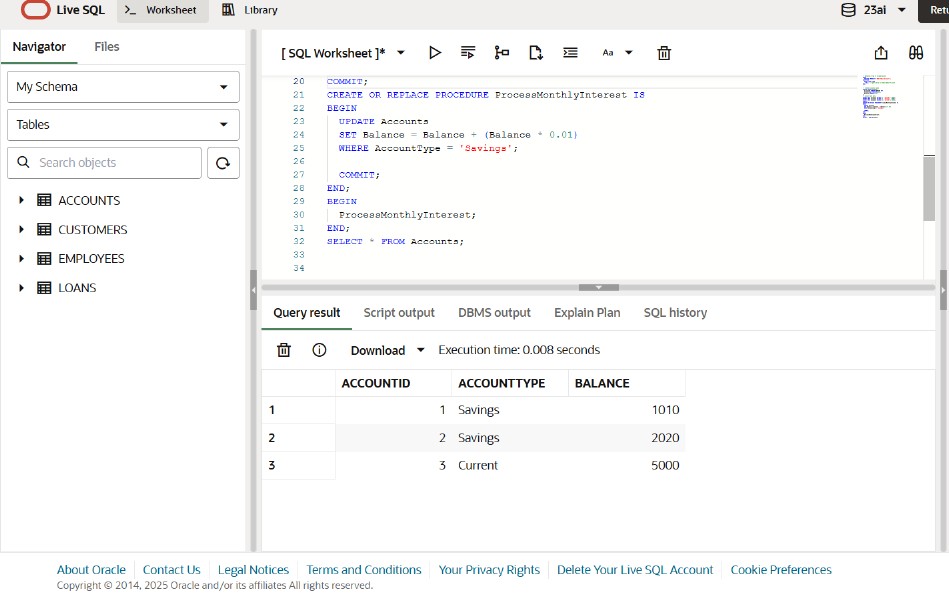
UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

COMMIT;

END;



Scenario 2: Employee Bonus Based on Department

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_DepartmentID IN NUMBER,

p\_BonusPercent IN NUMBER

) IS

BEGIN

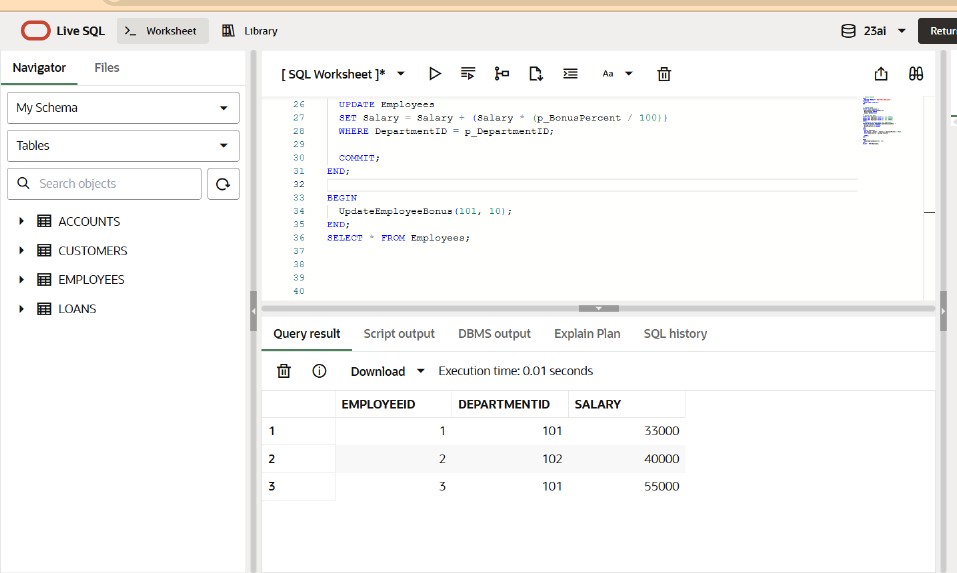
UPDATE Employees

SET Salary = Salary + (Salary \* (p\_BonusPercent / 100))

WHERE DepartmentID = p\_DepartmentID;

COMMIT;

END;



Scenario 3: Transfer Funds Between Accounts

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_FromAccount IN NUMBER,

p\_ToAccount IN NUMBER,

p\_Amount IN NUMBER

) IS

v\_FromBalance NUMBER;

BEGIN

SELECT Balance INTO v\_FromBalance

FROM Accounts

WHERE AccountID = p\_FromAccount;

IF v\_FromBalance >= p\_Amount THEN

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccount;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccount;

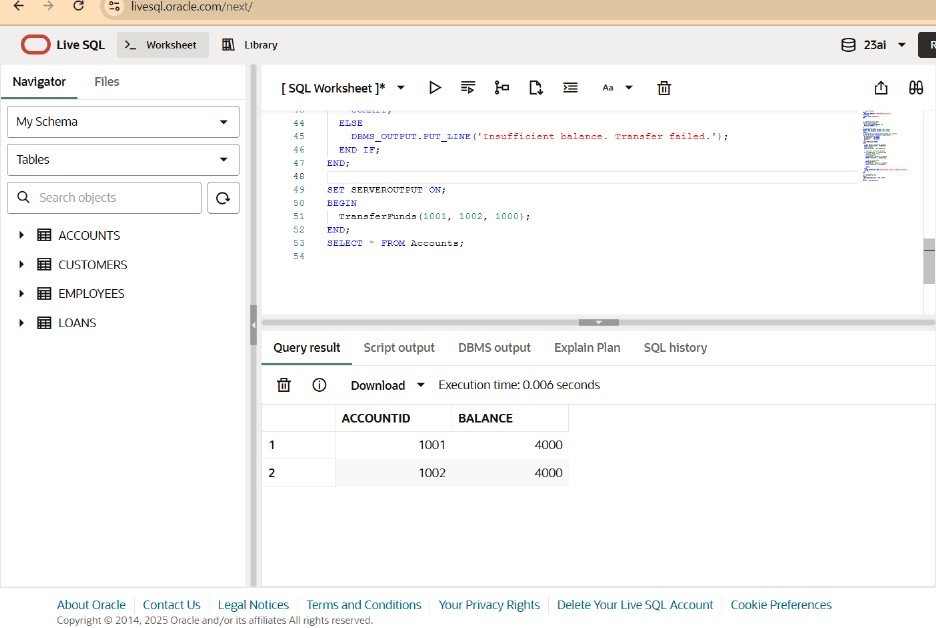
COMMIT;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance. Transfer failed.');

END IF;

END;



**Exercise 1: Setting Up JUnit**

**Calculator.java**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorTest.java**

package com.example;

import static org.junit.Assert.*assertEquals*;

import org.junit.Test;

public class CalculatorTest {

@Test

public void testAdd() {

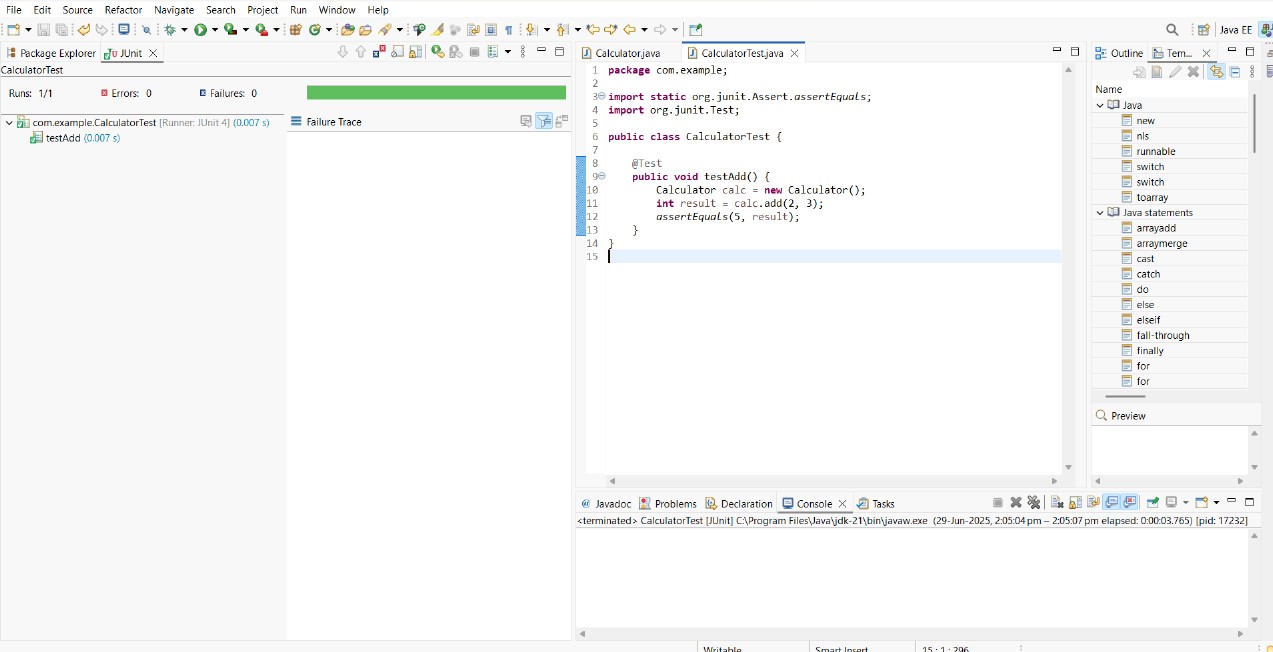
Calculator calc = new Calculator();

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

*assertEquals*(5, result);

}

}



**Exercise 3: Assertions in Junit**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 3);

assertFalse(5 < 3);

String value = null;

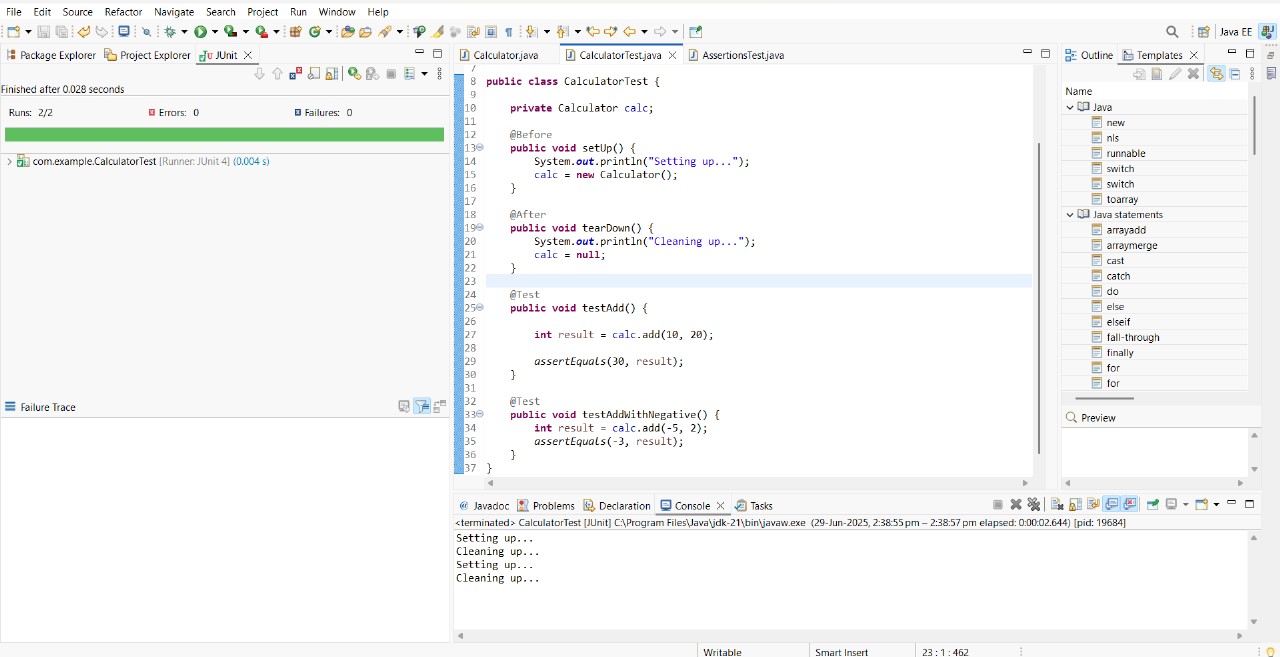
assertNull(value);

Object obj = new Object();

assertNotNull(obj);

}

}



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

**Calculator.java**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorTest.java**

package com.example;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.*assertEquals*;

public class CalculatorTest {

private Calculator calc;

@Before

public void setUp() {

System.*out*.println("Setting up...");

calc = new Calculator();

}

@After

public void tearDown() {

System.*out*.println("Cleaning up...");

calc = null;

}

@Test

public void testAdd() {

int result = calc.add(10, 20);

*assertEquals*(30, result);

}

@Test

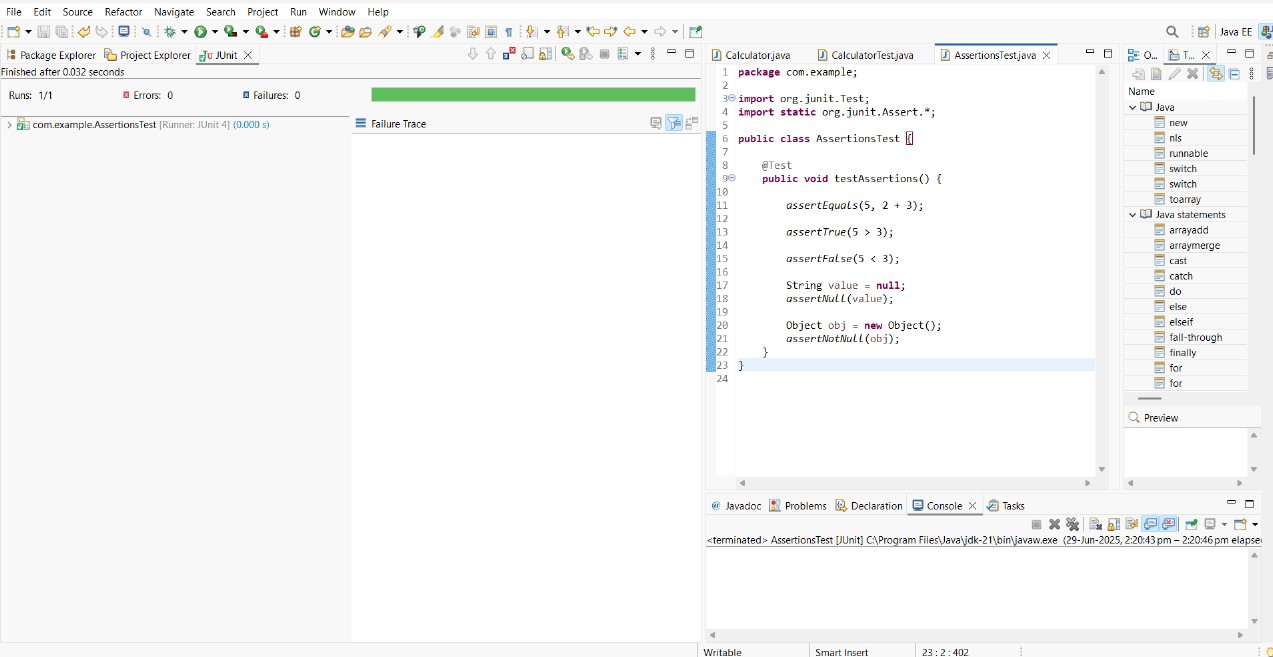
public void testAddWithNegative() {

int result = calc.add(-5, 2);

*assertEquals*(-3, result);

}

}



**Exercise 1: Mocking and Stubbing**

**ExternalAp.java**

package com.example;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.example;

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

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

@Test

public void testExternalApi()

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

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

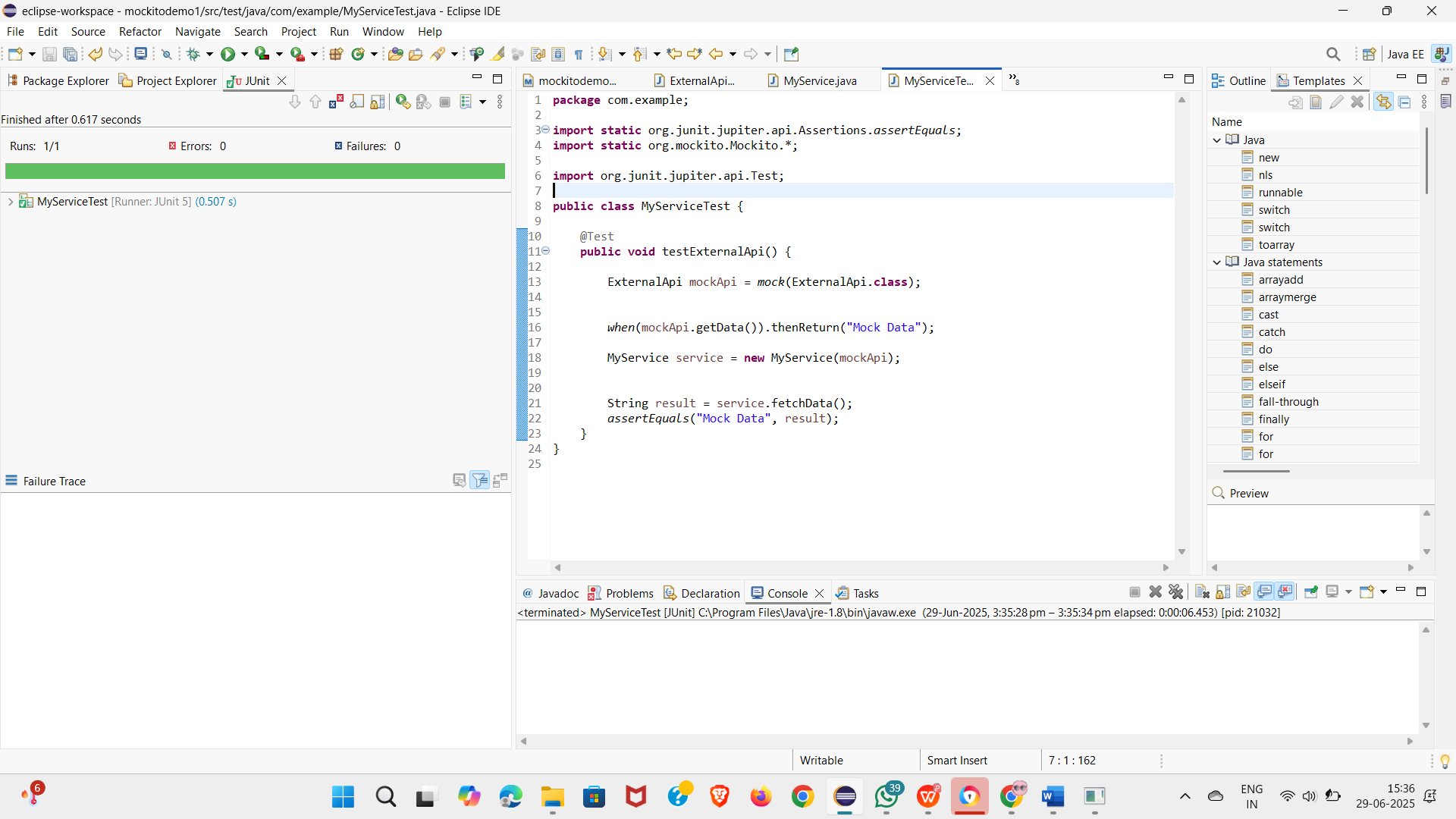
MyService service = new MyService(mockApi)

String result = service.fetchData();

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

}

}



**Exercise 2: Verifying Interactions**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.example;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

@Test

public void testVerifyInteraction()

ExternalApi mockApi = mock(ExternalApi.class);

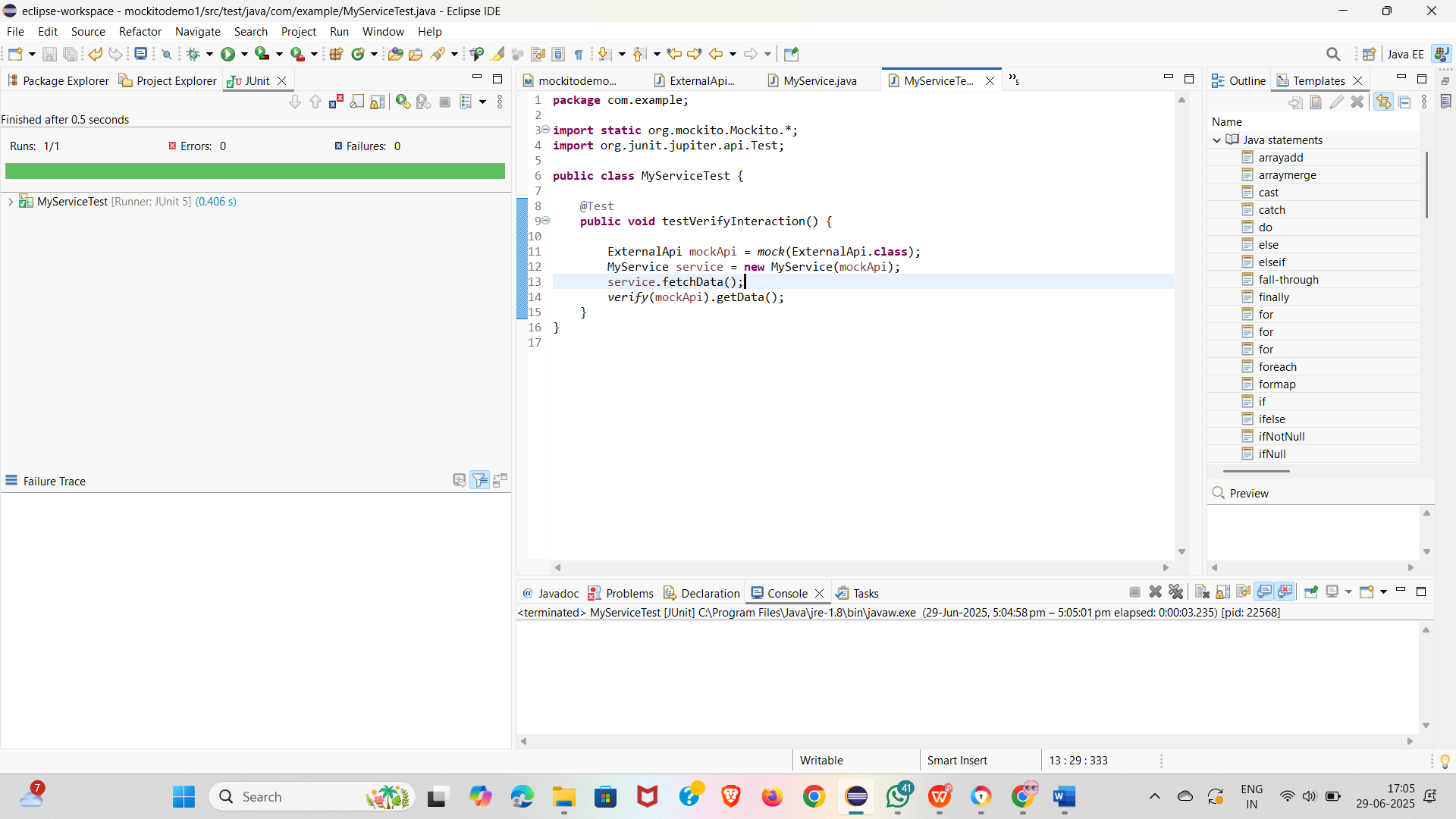
MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

}

}

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

**LoggingExample.java**

package com.example;

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");

}

}

