# Week2

# JUnit Testing

**Exercise 1: Setting Up Junit**

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>junit\_learning</groupId>

<artifactId>JUnit</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

package learning\_junit5;

public class Calculator

{

int add(int number1,int number2)

{

return number1+number2;

}

}

package learning\_junit5;

import static org.junit.Assert.\*;

import org.junit.Test;

public class CalculatorTest {

@Test

public void addtest() {

Calculator calc = new Calculator();

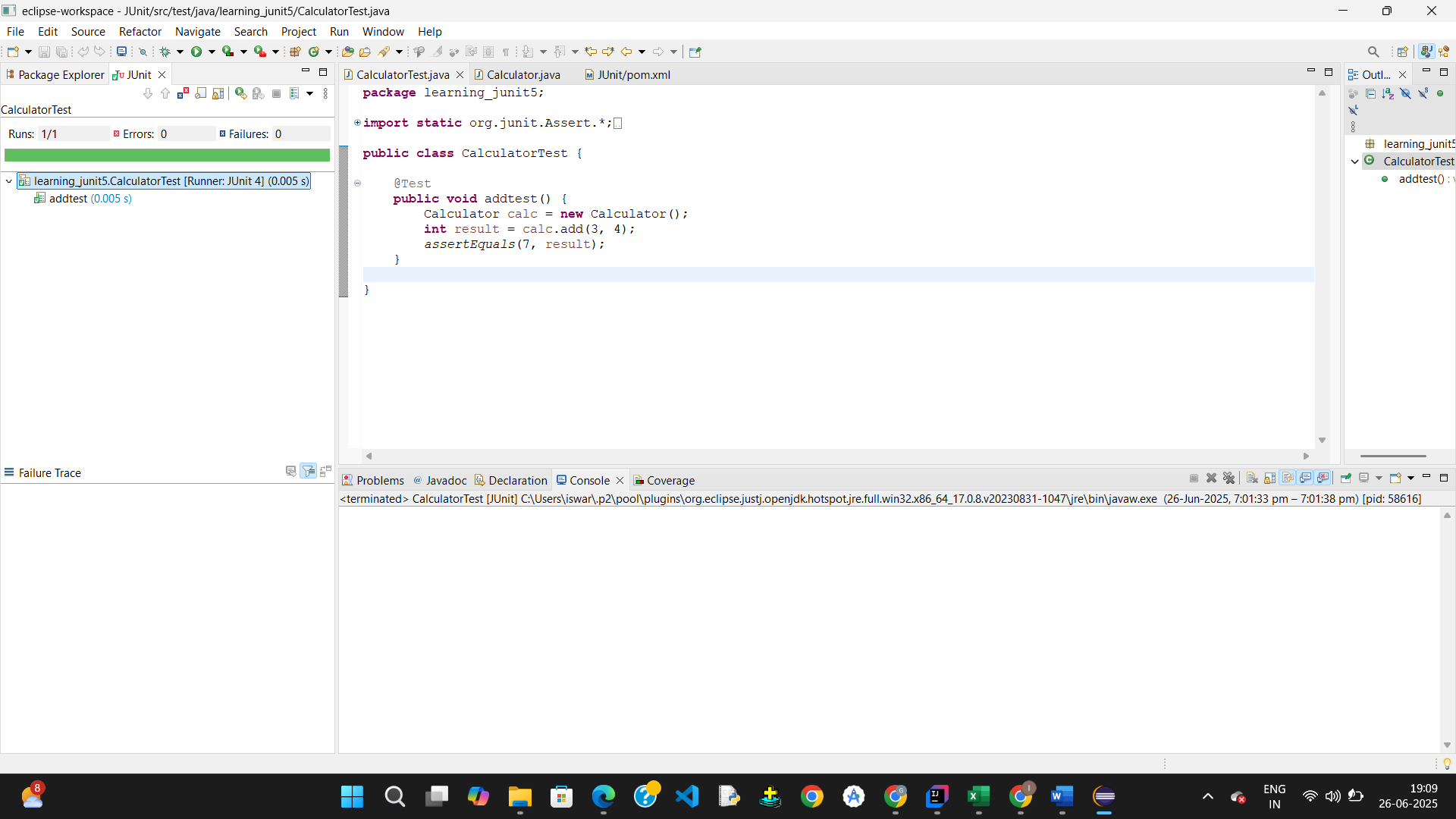
int result = calc.add(3, 4);

assertEquals(7, result);

}

}

**Output:**

****

**Exercise 3: Assertions in Junit**

package learning\_junit5;

public class AssertionsExamples {

public int add(int a, int b) {

return a + b;

}

public boolean isPositive(int num) {

return num > 0;

}

public String getWelcomeMessage() {

return "Welcome";

}

public Object getNullObject() {

return null;

}

public int[] getSampleArray() {

return new int[] {1, 2, 3};

}

}

package learning\_junit5;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AssertionsExamplesTest {

AssertionsExamples ae = new AssertionsExamples();

@Test

public void testAddition() {

assertEquals("Addition failed", 10, ae.add(4, 6));

}

@Test

public void testIsPositive() {

assertTrue("Expected number to be positive", ae.isPositive(5));

assertFalse("Expected number to be negative", ae.isPositive(-3));

}

@Test

public void testWelcomeMessage() {

assertEquals("Message doesn't match", "Welcome", ae.getWelcomeMessage());

}

@Test

public void testNullObject() {

assertNull("Expected null", ae.getNullObject());

}

@Test

public void testArrayEquality() {

int[] expected = {1, 2, 3};

assertArrayEquals("Arrays are not equal", expected, ae.getSampleArray());

}

@Test

public void testNotNull() {

assertNotNull("Should not be null", ae.getWelcomeMessage());

}

@Test

public void testSameObject() {

String msg = ae.getWelcomeMessage();

String ref = msg;

assertSame("Should be same reference", msg, ref);

}

@Test

public void testNotSameObject() {

String s1 = new String("hello");

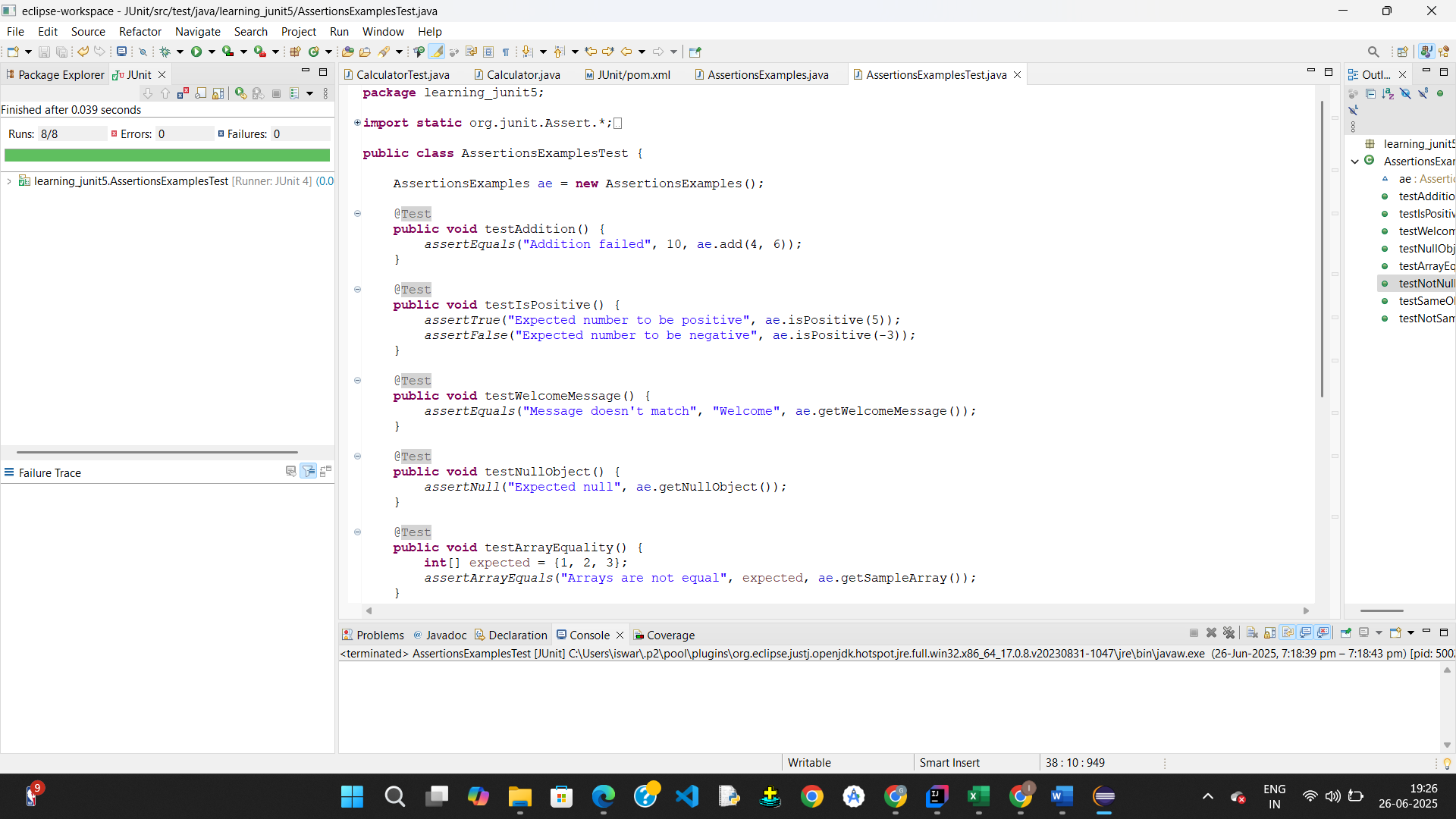
String s2 = new String("hello");

assertNotSame("Should not be same reference", s1, s2);

}

}

**Output:**

****

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

package learning\_junit5;

public class AAApattern {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

public boolean isEven(int num) {

return num % 2 == 0;

}

}

**package** learning\_junit5;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

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

**public** **class** AAApatternTest {

**private** AAApattern ap;

// Setup method - runs before each test

@Before

**public** **void** setUp() {

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

ap = **new** AAApattern(); // Arrange

}

// Teardown method - runs after each test

@After

**public** **void** tearDown() {

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

ap = **null**;

}

@Test

**public** **void** testAddition() {

// Act

**int** result = ap.add(10, 5);

// Assert

*assertEquals*("Addition should return correct result", 15, result);

}

@Test

**public** **void** testSubtraction() {

// Act

**int** result = ap.subtract(10, 4);

// Assert

*assertEquals*("Subtraction should return correct result", 6, result);

}

@Test

**public** **void** testIsEvenTrue() {

// Act

**boolean** isEven = ap.isEven(8);

// Assert

*assertTrue*("8 should be even", isEven);

}

@Test

**public** **void** testIsEvenFalse() {

// Act

**boolean** isEven = ap.isEven(7);

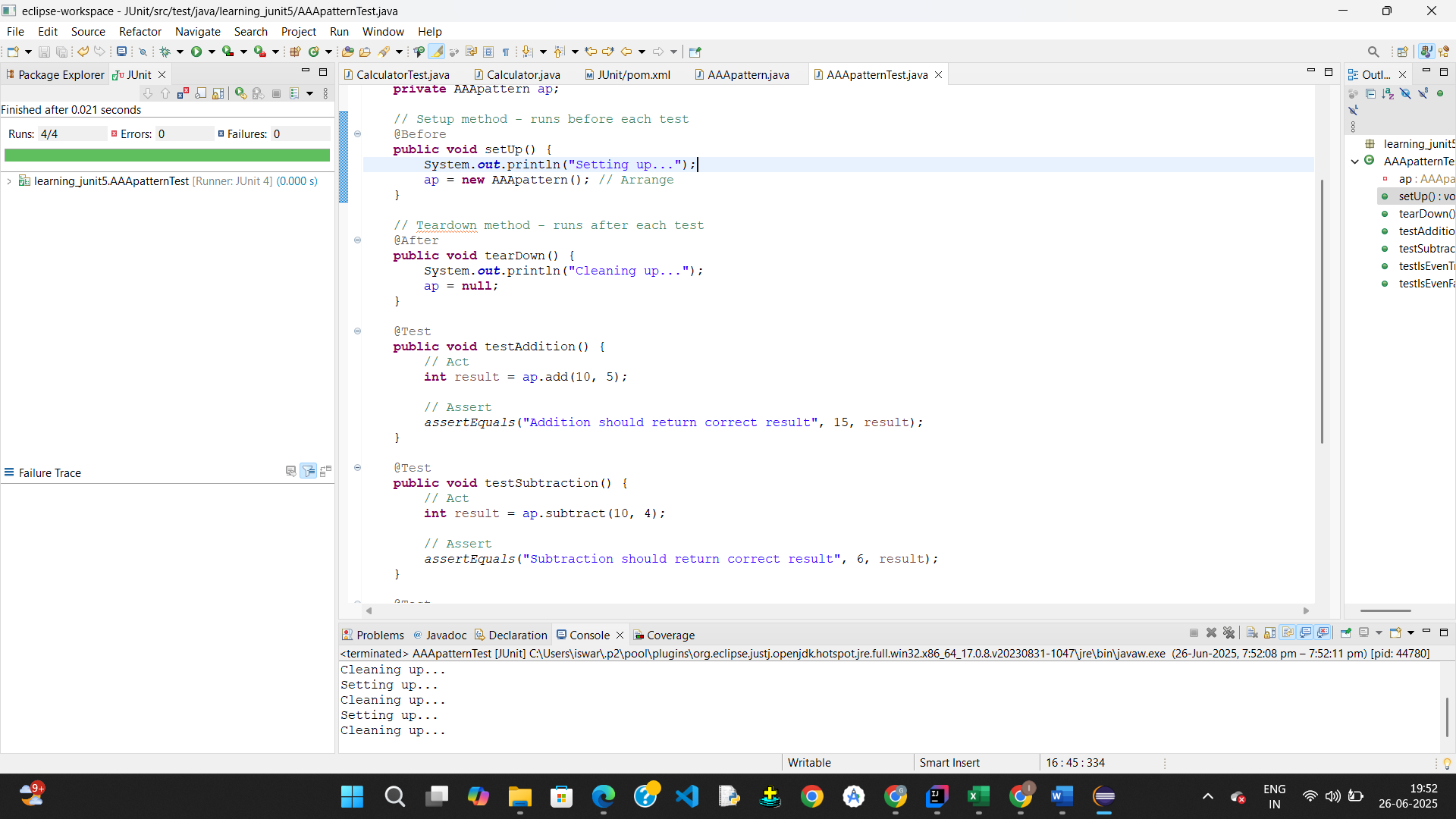
// Assert

*assertFalse*("7 should not be even", isEven);

}

}

Output:



# Week2

# Mockito

**Exercise 1: Mocking and Stubbing**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>junit\_learning</groupId>

<artifactId>JUnit</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>3.12.4</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**Interface**

**package** learning\_junit5;

**public** **interface** WeatherApiClient {

String getWeather(String city);

}

**WeatherService.java**

package learning\_junit5;

public class WeatherService {

private WeatherApiClient apiClient;

public WeatherService(WeatherApiClient apiClient) {

this.apiClient = apiClient;

}

public String fetchWeatherMessage(String city) {

String weather = apiClient.getWeather(city);

return "Weather in " + city + " is: " + weather;

}

}

package learning\_junit5;

import static org.junit.Assert.\*;

import org.junit.Test;

import org.junit.Test;

import static org.junit.Assert.\*;

import static org.mockito.Mockito.\*;

public class WeatherServiceTest {

@Test

public void testFetchWeatherMessage() {

// Step 1: Create mock object for WeatherApiClient

WeatherApiClient mockApi = mock(WeatherApiClient.class);

// Step 2: Stub the getWeather() method

when(mockApi.getWeather("Delhi")).thenReturn("Sunny");

// Step 3: Use mock object in WeatherService

WeatherService service = new WeatherService(mockApi);

// Act

String message = service.fetchWeatherMessage("Delhi");

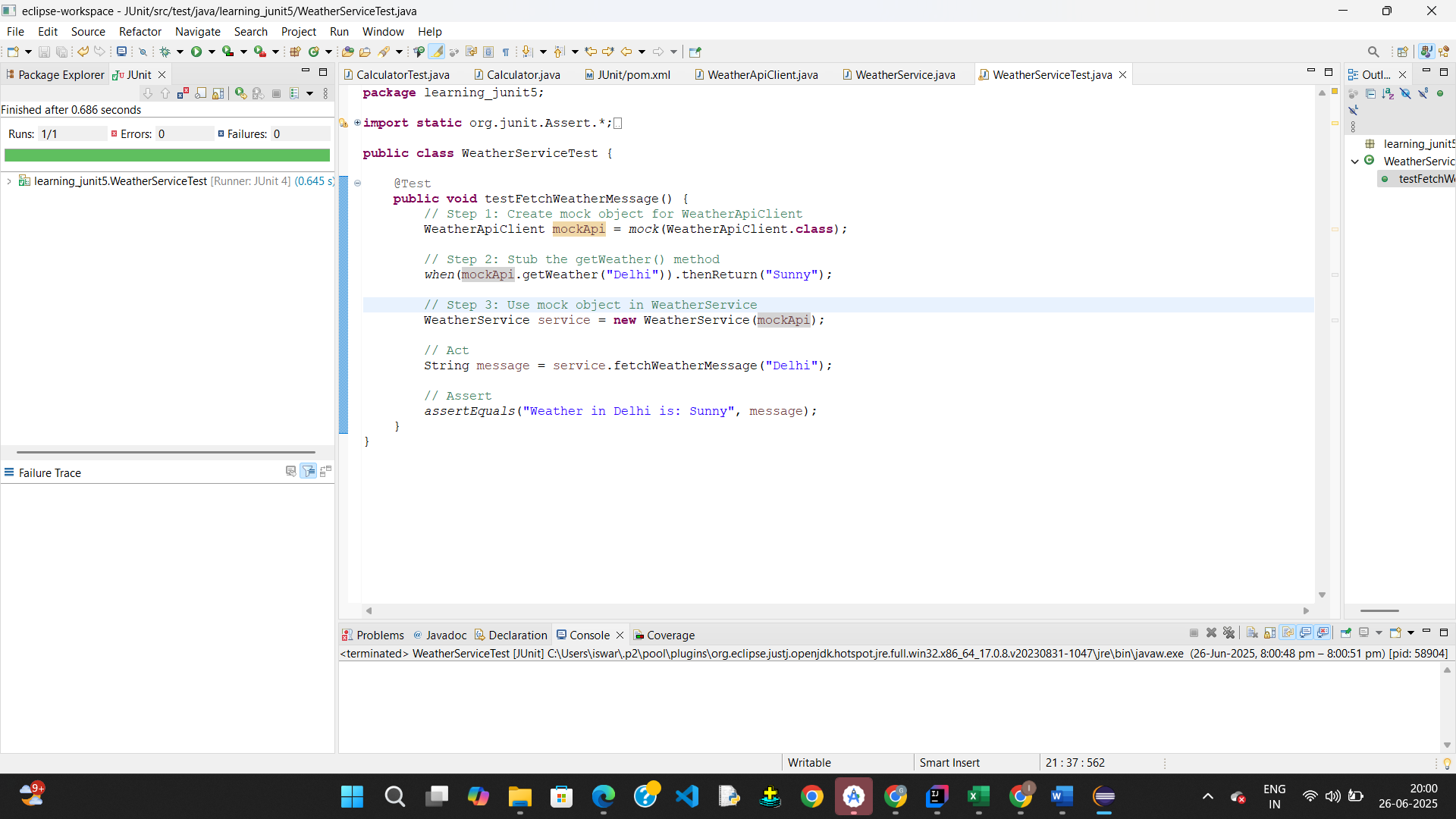
// Assert

assertEquals("Weather in Delhi is: Sunny", message);

}

}

Output:



**Exercise 2: Verifying Interactions**

**Interface**

package learning\_junit5;

public interface EmailSender {

void sendEmail(String to, String subject, String body);

}

**NotificationService.java**

**package** learning\_junit5;

**public** **class** NotificationService {

**private** EmailSender emailSender;

**public** NotificationService(EmailSender emailSender) {

**this**.emailSender = emailSender;

}

**public** **void** sendWelcomeEmail(String userEmail) {

emailSender.sendEmail(userEmail, "Welcome", "Thank you for joining!");

}

}

**NotificationServiceTest.java**

package learning\_junit5;

import static org.junit.Assert.\*;

import org.junit.Test;

import org.junit.Test;

import static org.mockito.Mockito.\*;

public class NotificationServiceTest {

@Test

public void testSendWelcomeEmail() {

// Step 1: Create mock object

EmailSender mockEmailSender = mock(EmailSender.class);

// Step 2: Call method with specific argument

NotificationService service = new NotificationService(mockEmailSender);

service.sendWelcomeEmail("test@example.com");

// Step 3: Verify the interaction

verify(mockEmailSender).sendEmail(

eq("test@example.com"),

eq("Welcome"),

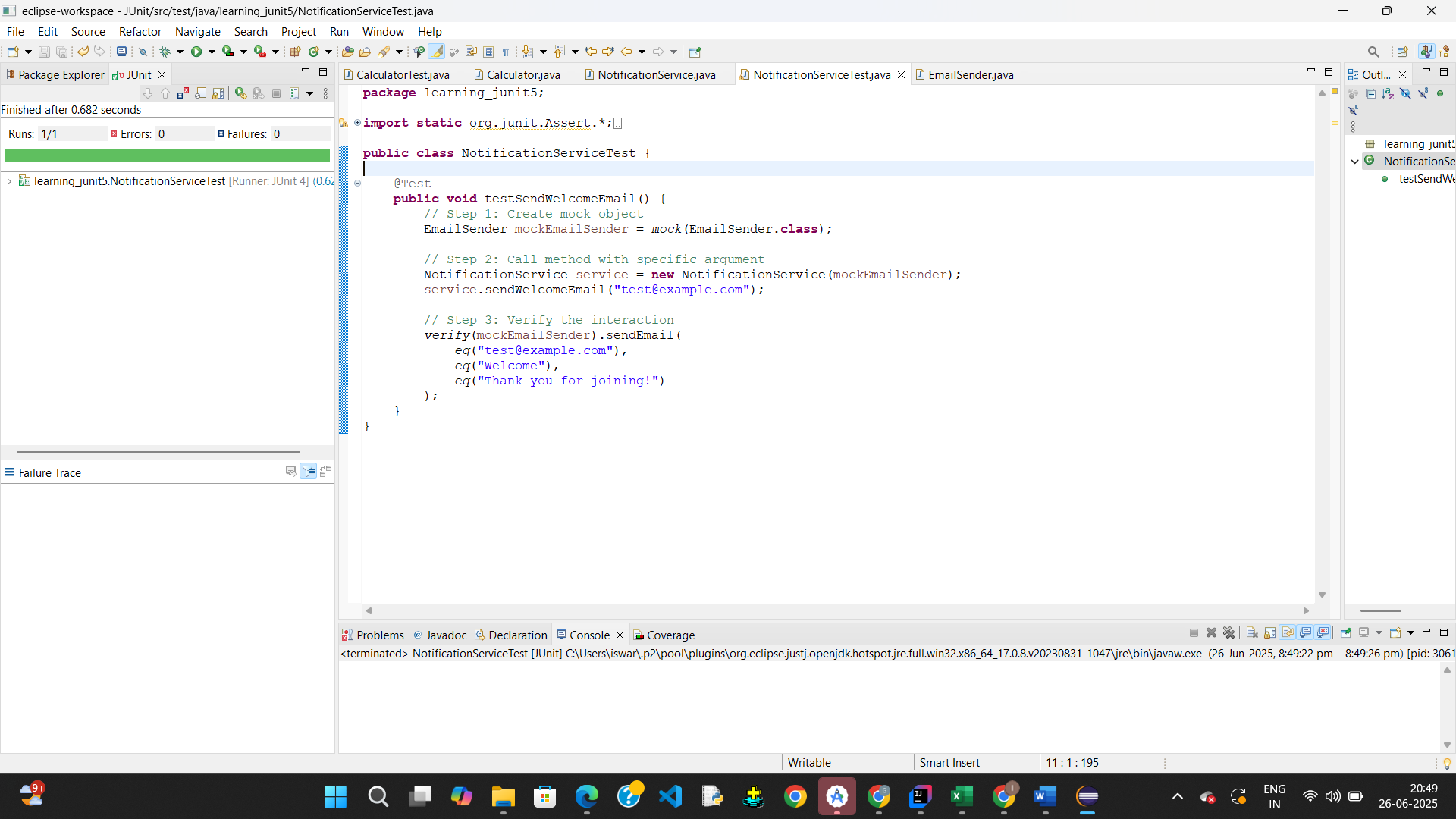
eq("Thank you for joining!")

);

}

}

**Output:**

****

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

**Poom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>junit\_learning</groupId>

<artifactId>JUnit</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>3.12.4</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

</project>

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.info("Application started");

try {

int result = divide(10, 0);

logger.info("Result is: {}", result);

} catch (ArithmeticException ex) {

logger.error("Error occurred while dividing: {}", ex.getMessage());

}

logger.warn("This is a warning about resource usage");

logger.info("Application finished");

}

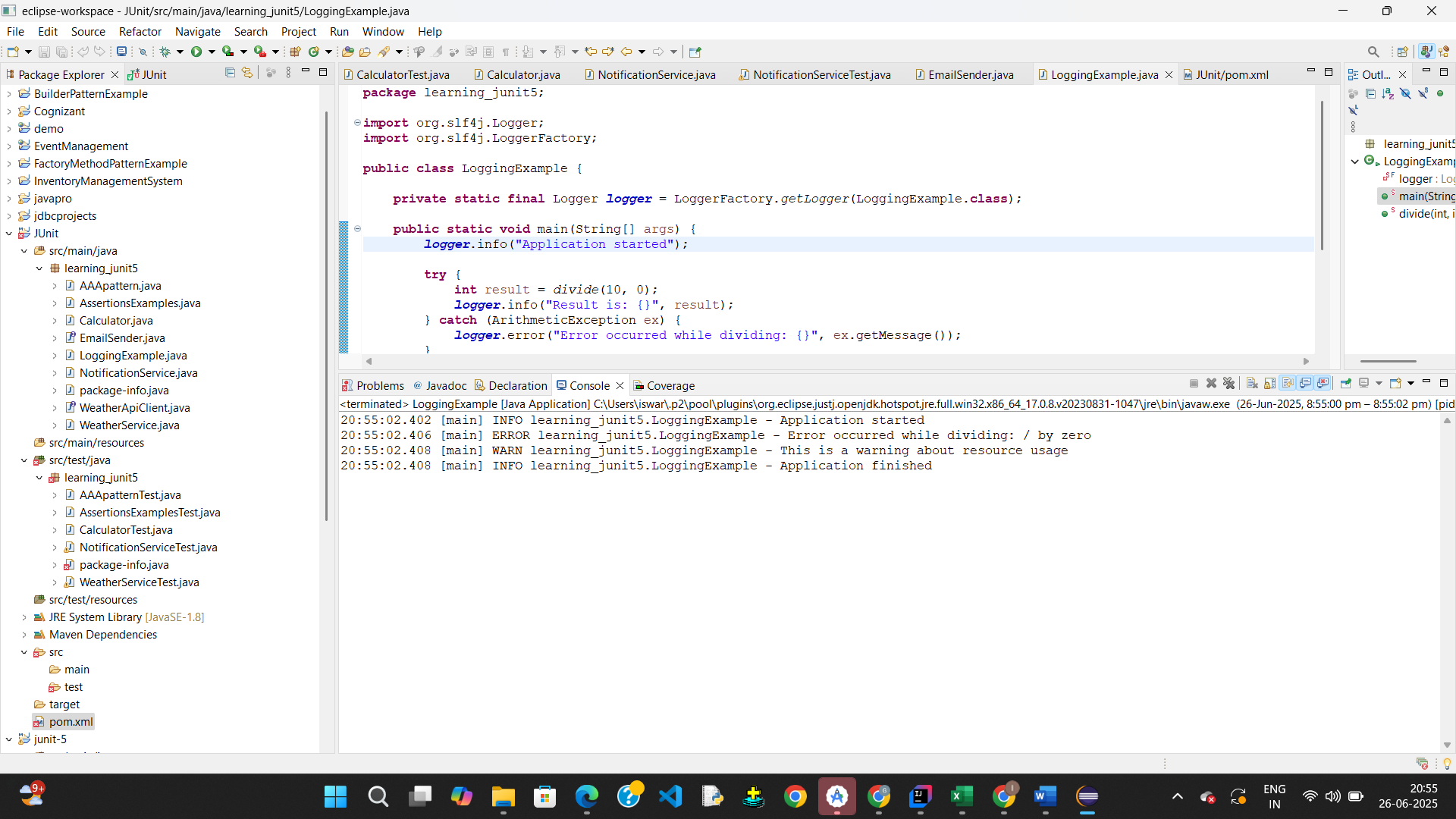
public static int divide(int a, int b) {

return a / b;

}

}

Output:



# Week-2

# PL/SQL

**Exercise 1: Control Structures**

**Scenario-1**

BEGIN

FOR rec IN (

SELECT l.LoanID, l.InterestRate, c.DOB

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

)

LOOP

IF MONTHS\_BETWEEN(SYSDATE, rec.DOB) / 12 > 60 THEN

UPDATE Loans SET InterestRate = rec.InterestRate - 1 WHERE LoanID = rec.LoanID;

DBMS\_OUTPUT.PUT\_LINE('Updated LoanID: ' || rec.LoanID || ' | New Interest Rate: ' || (rec.InterestRate - 1));

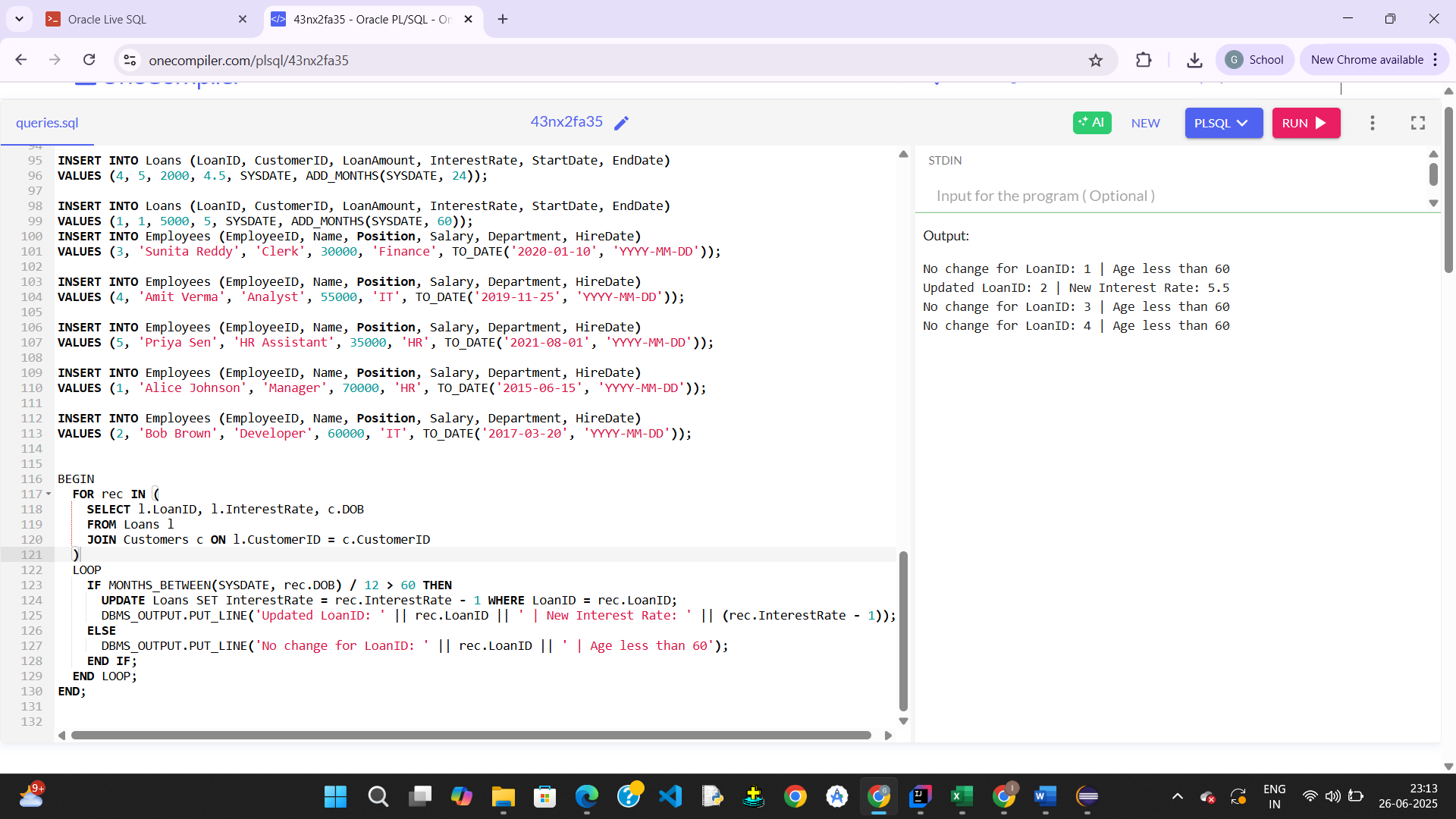
ELSE

DBMS\_OUTPUT.PUT\_LINE('No change for LoanID: ' || rec.LoanID || ' | Age less than 60');

END IF;

END LOOP;

END;



**Scenario-2**

BEGIN

FOR rec IN (SELECT CustomerID, Balance FROM Customers)

LOOP

IF rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer ' || rec.CustomerID || ' promoted to VIP.');

ELSE

UPDATE Customers

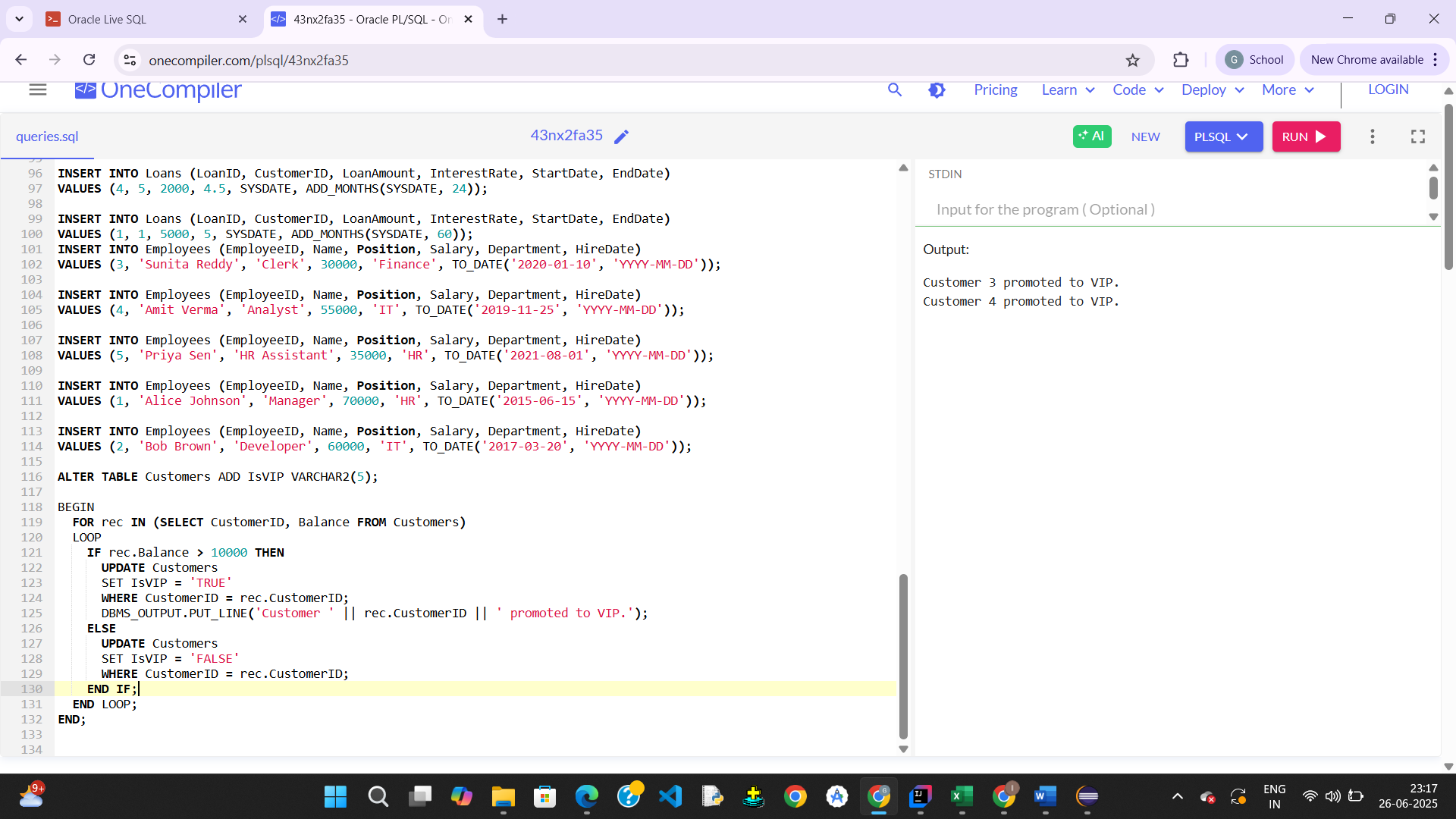
SET IsVIP = 'FALSE'

WHERE CustomerID = rec.CustomerID;

END IF;

END LOOP;

END;



**Scenario-3**

BEGIN

FOR rec IN (

SELECT l.LoanID, c.Name, l.EndDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

)

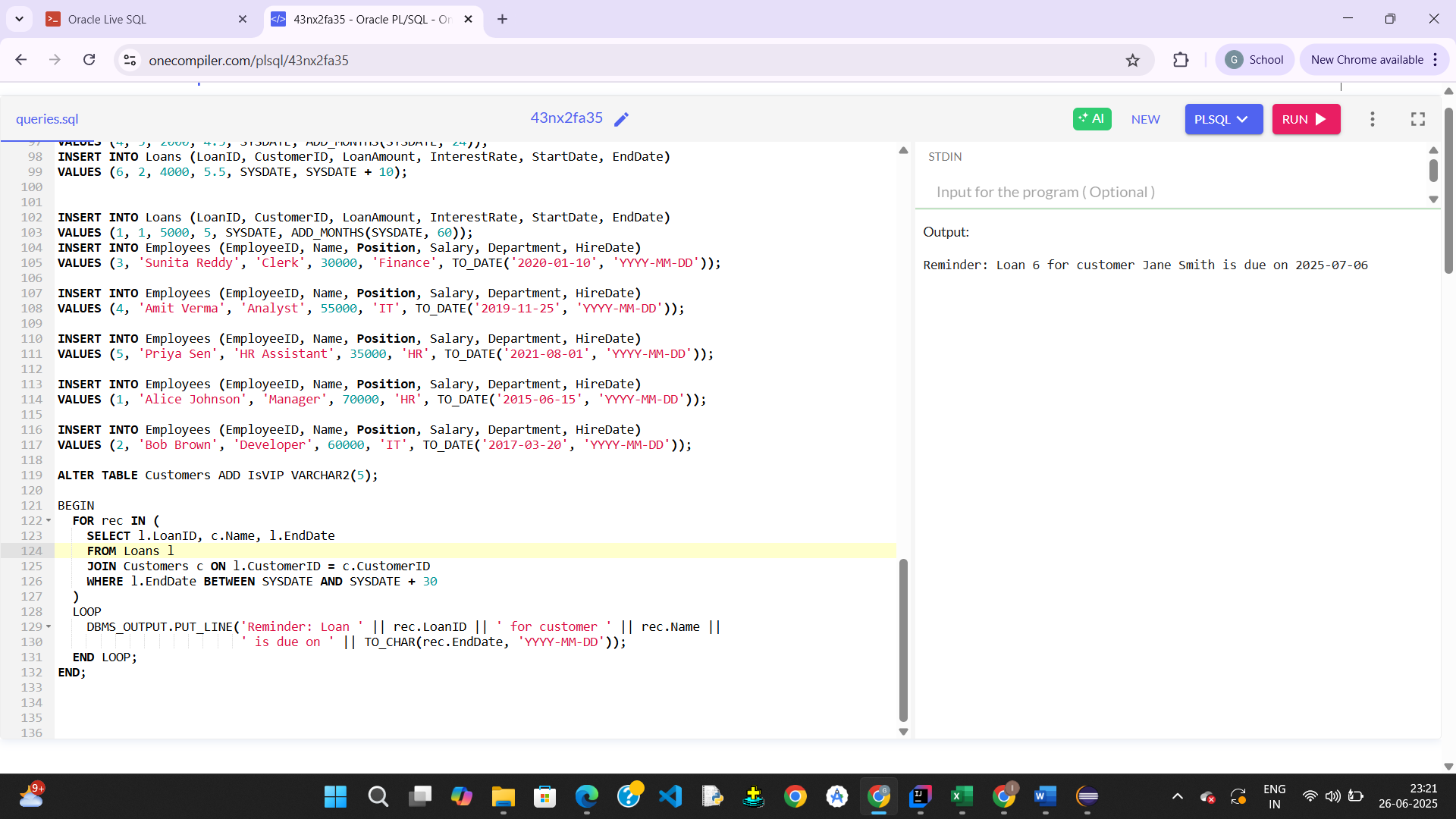
LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ' || rec.LoanID || ' for customer ' || rec.Name ||

' is due on ' || TO\_CHAR(rec.EndDate, 'YYYY-MM-DD'));

END LOOP;

END;



**Exercise 3: Stored Procedures**

**Scenario-1**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR rec IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

)

LOOP

UPDATE Accounts

SET Balance = rec.Balance + (rec.Balance \* 0.01),

LastModified = SYSDATE

WHERE AccountID = rec.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest added to AccountID: ' || rec.AccountID);

END LOOP;

END;

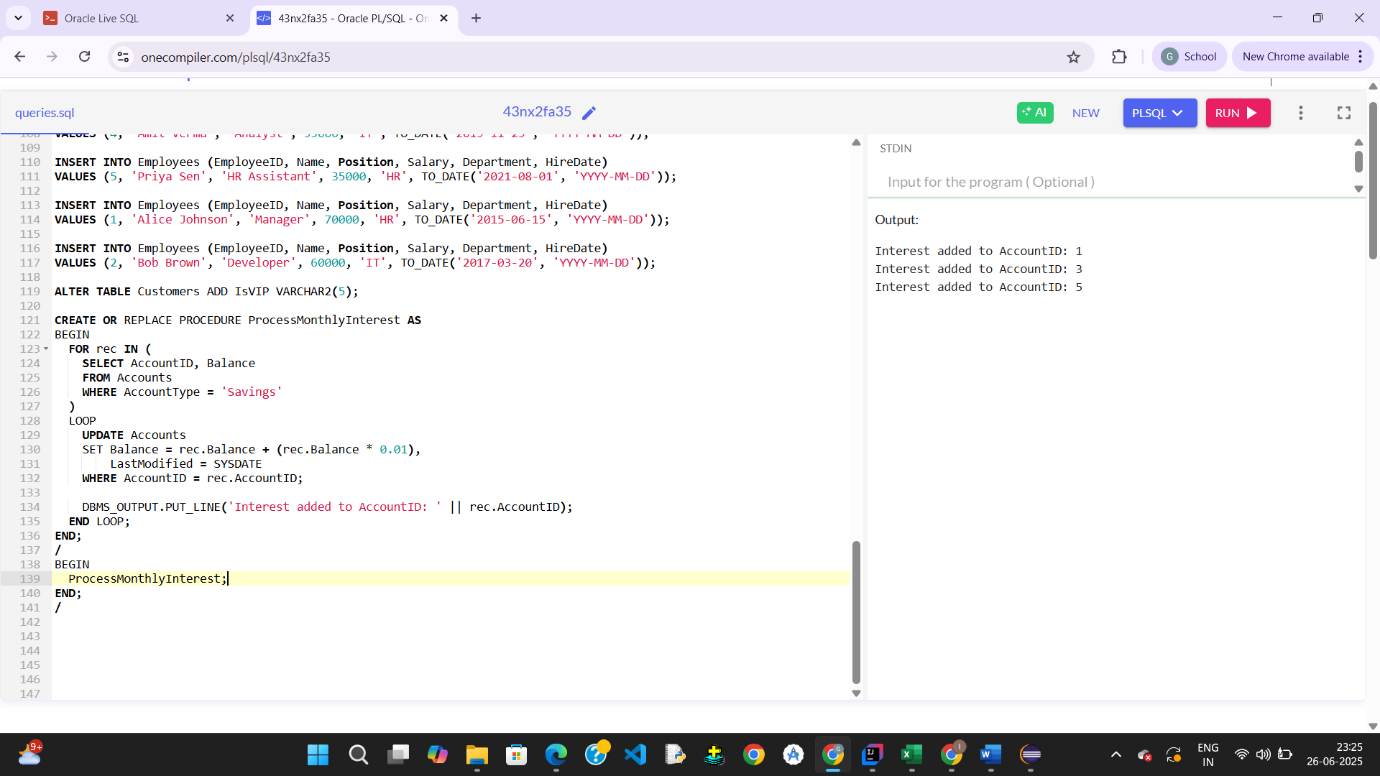
/

BEGIN

ProcessMonthlyInterest;

END;

/



**Scenario-2**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

deptName IN VARCHAR2,

bonusPercent IN NUMBER

) AS

BEGIN

FOR rec IN (

SELECT EmployeeID, Salary

FROM Employees

WHERE Department = deptName

)

LOOP

UPDATE Employees

SET Salary = rec.Salary + (rec.Salary \* bonusPercent / 100)

WHERE EmployeeID = rec.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Bonus updated for EmployeeID: ' || rec.EmployeeID);

END LOOP;

END;

/

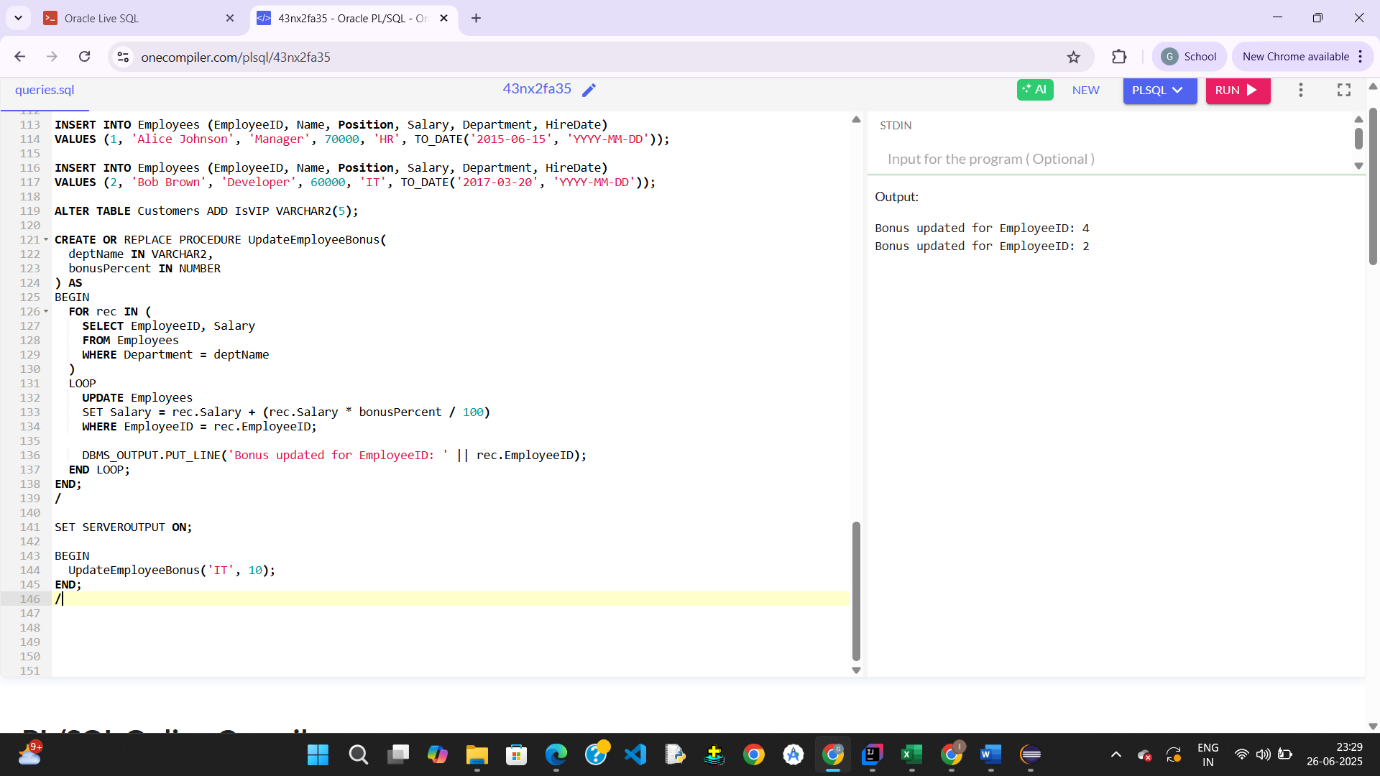
SET SERVEROUTPUT ON;

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

/



**Scenario-3**

CREATE OR REPLACE PROCEDURE TransferFunds(

fromAccount IN NUMBER,

toAccount IN NUMBER,

amount IN NUMBER

) AS

insufficient\_balance EXCEPTION;

from\_balance NUMBER;

BEGIN

SELECT Balance INTO from\_balance FROM Accounts WHERE AccountID = fromAccount;

IF from\_balance < amount THEN

RAISE insufficient\_balance;

END IF;

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - amount,

LastModified = SYSDATE

WHERE AccountID = fromAccount;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + amount,

LastModified = SYSDATE

WHERE AccountID = toAccount;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: ' || amount ||

' transferred from Account ' || fromAccount ||

' to Account ' || toAccount);

EXCEPTION

WHEN insufficient\_balance THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient balance in Account ' || fromAccount);

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: One or both accounts not found.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Unexpected error: ' || SQLERRM);

END;

/

set serveroutput on;

BEGIN

TransferFunds(1, 2, 100);

END;

/

