**PL/SQL**

**Exercise 1:**

CREATE TABLE Customers (

    CustomerID NUMBER PRIMARY KEY,

    Name VARCHAR2(50),

    Age NUMBER,

    Balance NUMBER(10,2),

    IsVIP CHAR(1) DEFAULT 'N'

);

CREATE TABLE Loans (

    LoanID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    InterestRate NUMBER(5,2),

    DueDate DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers VALUES (1, 'Kinshuk', 65, 15000, 'N');

INSERT INTO Customers VALUES (2, 'Sherlock', 45, 8000, 'N');

INSERT INTO Customers VALUES (3, 'Poirot', 70, 12000, 'N');

INSERT INTO Loans VALUES (101, 1, 7.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 6.5, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 8.0, SYSDATE + 5);

COMMIT;



**Question 1:**

BEGIN

    FOR rec IN (SELECT l.LoanID, l.InterestRate, c.Age FROM Loans l JOIN Customers c ON l.CustomerID = c.CustomerID) LOOP

        IF rec.Age > 60 THEN

            UPDATE Loans

            SET InterestRate = InterestRate - 1

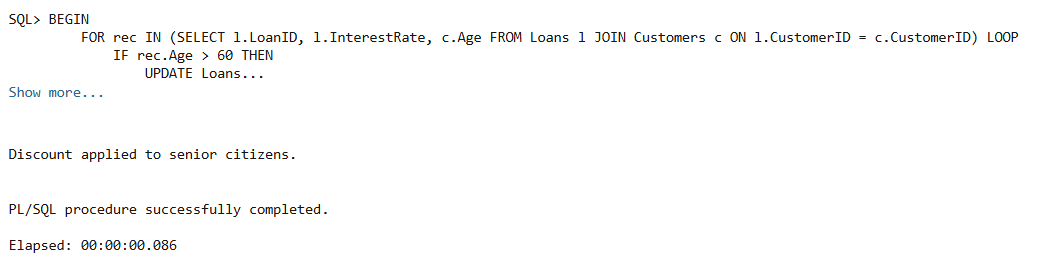
            WHERE LoanID = rec.LoanID;

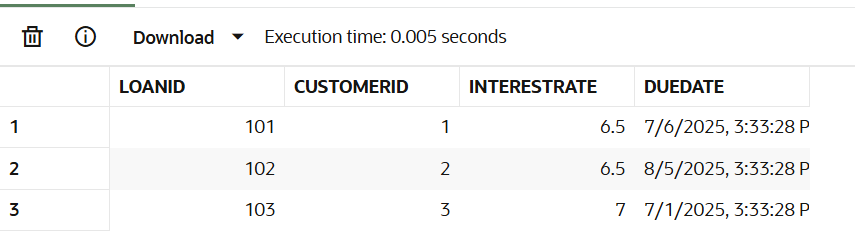
        END IF;

    END LOOP;

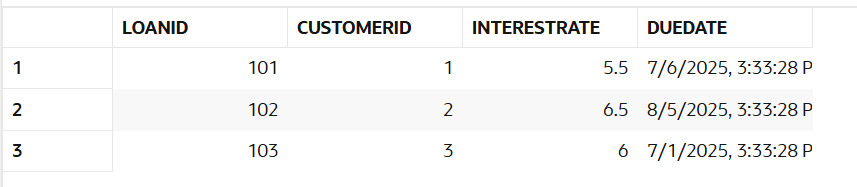
    DBMS\_OUTPUT.PUT\_LINE('Discount applied to senior citizens.');

END;



BEFORE:

AFTER:



Question 2

BEGIN

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

IF rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = rec.CustomerID;

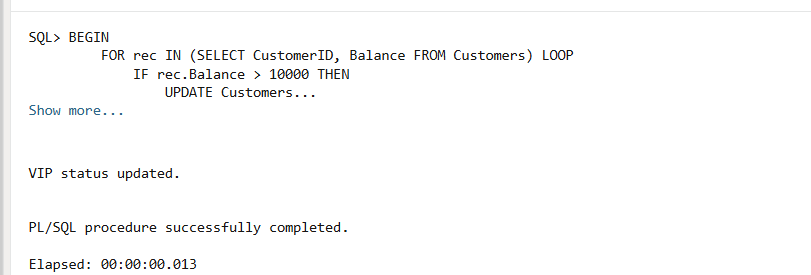
END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated.');

END;

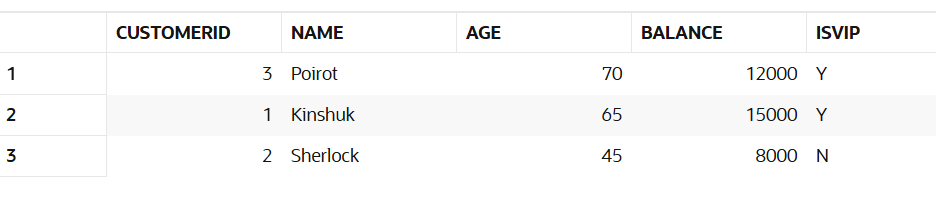
/



BEFORE



AFTER



Question 3

SET SERVEROUTPUT ON;

BEGIN

    FOR rec IN (

        SELECT c.Name, l.DueDate

        FROM Loans l JOIN Customers c ON l.CustomerID = c.CustomerID

        WHERE l.DueDate <= SYSDATE + 30

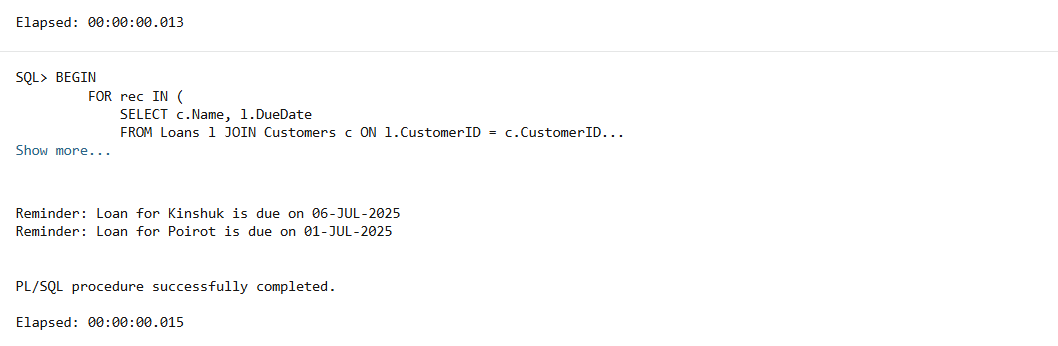
    ) LOOP

        DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan for ' || rec.Name || ' due on ' || TO\_CHAR(rec.DueDate, 'DD-MON-YYYY'));

    END LOOP;

END;

/



Exercise 3

CREATE TABLE SavingsAccounts (

    AccountID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    Balance NUMBER(10,2)

);

CREATE TABLE Employees (

    EmployeeID NUMBER PRIMARY KEY,

    Name VARCHAR2(50),

    Department VARCHAR2(30),

    Salary NUMBER(10,2)

);

\

Q1

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR rec IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP

UPDATE SavingsAccounts

SET Balance = Balance + (Balance \* 0.01)

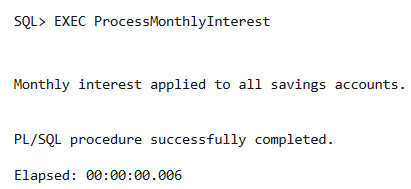
WHERE AccountID = rec.AccountID;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

END;

/



BEFORE



AFTER



Q2

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

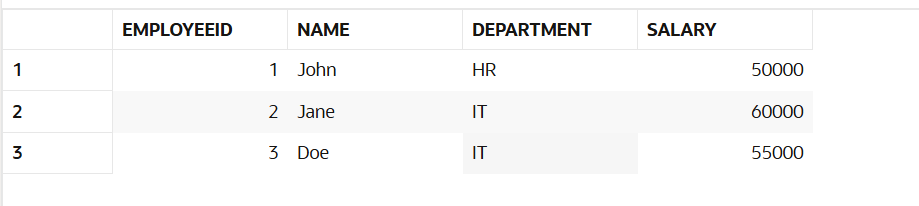
WHERE Department = dept;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department: ' || dept);

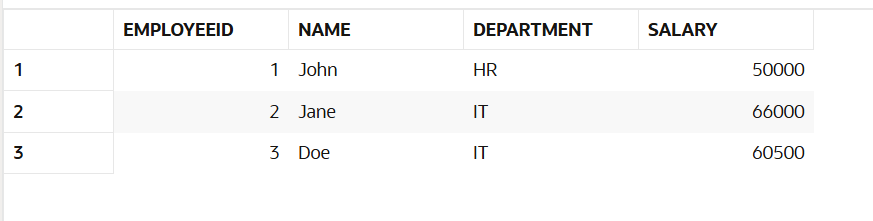
END;

/

BEFORE

****

**AFTER**

****

**Q3**

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_acct IN NUMBER,

to\_acct IN NUMBER,

amt IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Check balance

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

IF v\_balance < amt THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in source account.');

RETURN;

END IF;

-- Deduct from source

UPDATE Accounts SET Balance = Balance - amt WHERE AccountID = from\_acct;

-- Add to target

UPDATE Accounts SET Balance = Balance + amt WHERE AccountID = to\_acct;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: ' || amt || ' transferred.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Invalid account ID.');

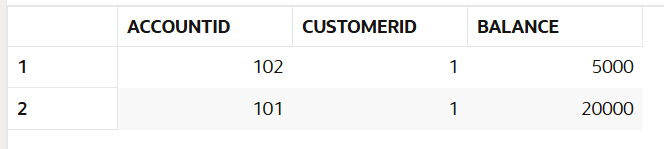
WHEN OTHERS THEN

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

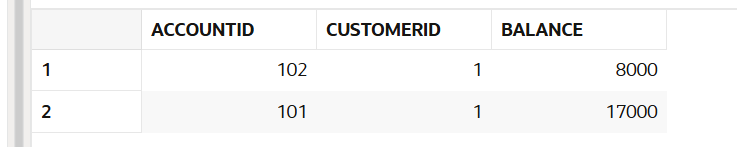
END;

/

**BEFORE**

****

**AFTER**

****

**TDD using JUnit5 and Mockito**

**Exercise 1**

**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 org.junit.jupiter.api.Test;

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

public class CalculatorTest {

    @Test

    void testAddition() {

        Calculator calc = new Calculator();

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

    }

    @Test

    void testSubtraction() {

        Calculator calc = new Calculator();

        assertEquals(1, calc.subtract(3, 2));

    }

    @Test

    void testNotEqual() {

        Calculator calc = new Calculator();

        assertNotEquals(10, calc.add(2, 3));

    }

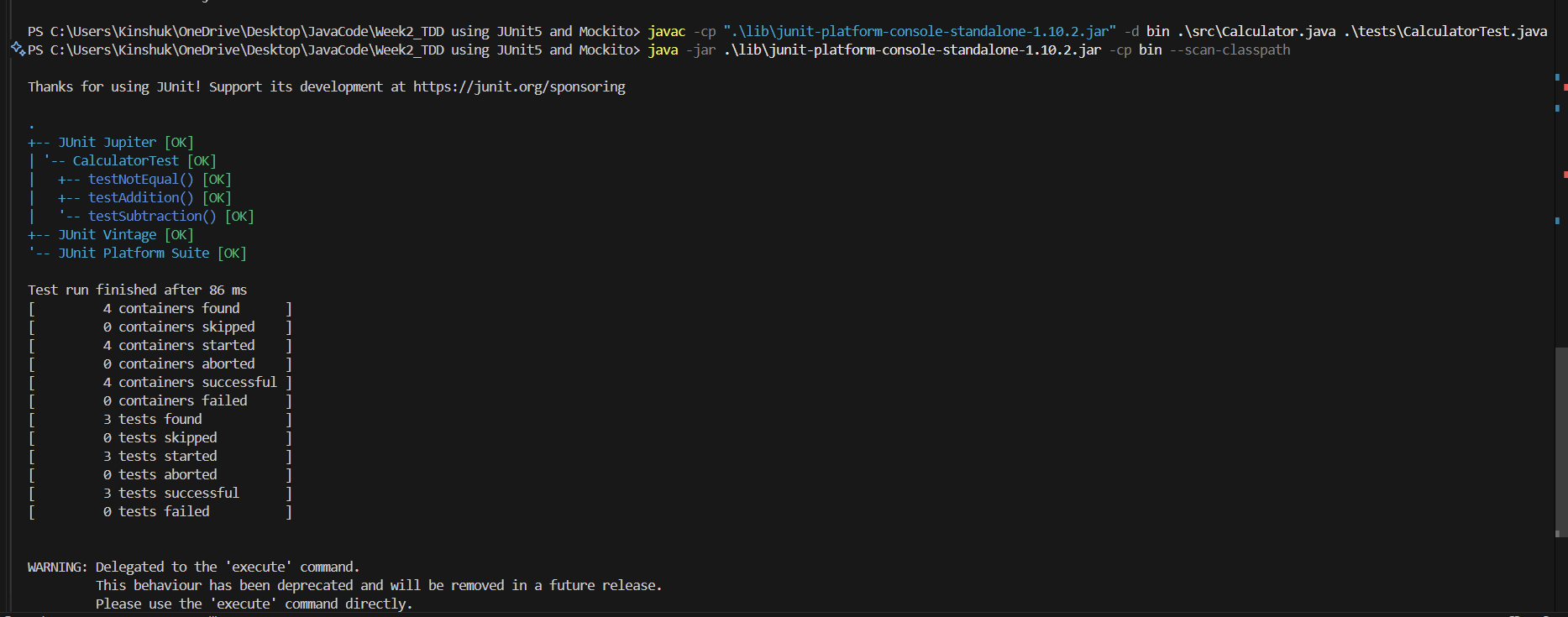
}

**COMMAND 1**

**>javac -cp ".\lib\junit-platform-console-standalone-1.10.2.jar" -d bin .\src\Calculator.java .\tests\CalculatorTest.java**

**>java -jar .\lib\junit-platform-console-standalone-1.10.2.jar -cp bin --scan-classpath**

**OUTPUT**

****

**Exercise 3**

**We put the AssertionsTest.java file under tests directory**

import org.junit.jupiter.api.Test;

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

public class AssertionsTest {

    @Test

    void testAssertions() {

        // Assert equals

        assertEquals(5, 2 + 3);

        // Assert true

        assertTrue(5 > 3);

        // Assert false

        assertFalse(5 < 3);

        // Assert null

        Object obj1 = null;

        assertNull(obj1);

        // Assert not null

        Object obj2 = new Object();

        assertNotNull(obj2);

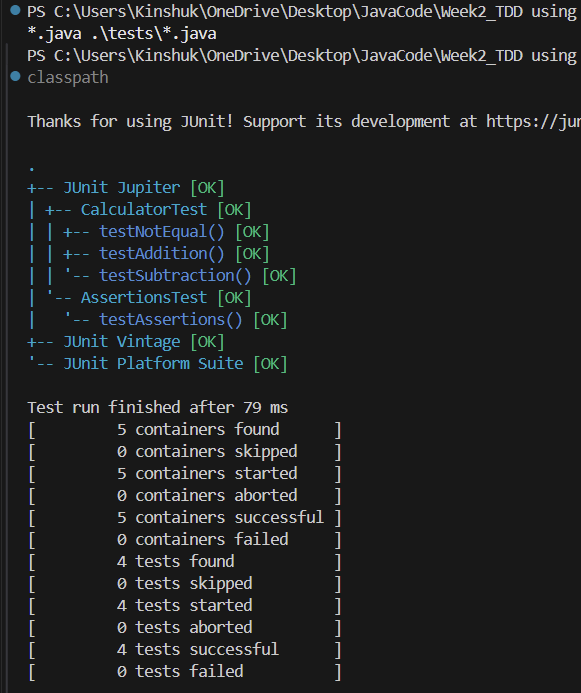
    }

}

**COMMANDS**

**>javac -cp ".\lib\junit-platform-console-standalone-1.10.2.jar" -d bin .\src\\*.java .\tests\\*.java**

**>java -jar .\lib\junit-platform-console-standalone-1.10.2.jar -cp bin --scan-classpath**

****

**Exercise 4**

**Code**

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

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

public class CalculatorWithSetupTest {

    private Calculator calculator;

    @BeforeEach

    void setUp() {

        // Setup phase: runs before each test

        calculator = new Calculator();

        System.out.println("Setting up Calculator instance...");

    }

    @AfterEach

    void tearDown() {

        // Teardown phase: runs after each test

        calculator = null;

        System.out.println("Tearing down Calculator instance...");

    }

    @Test

    void testAddition() {

        // Arrange

        int a = 10;

        int b = 5;

        // Act

        int result = calculator.add(a, b);

        // Assert

        assertEquals(15, result);

    }

    @Test

    void testSubtraction() {

        // Arrange

        int a = 10;

        int b = 4;

        // Act

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

        // Assert

        assertEquals(6, result);

    }

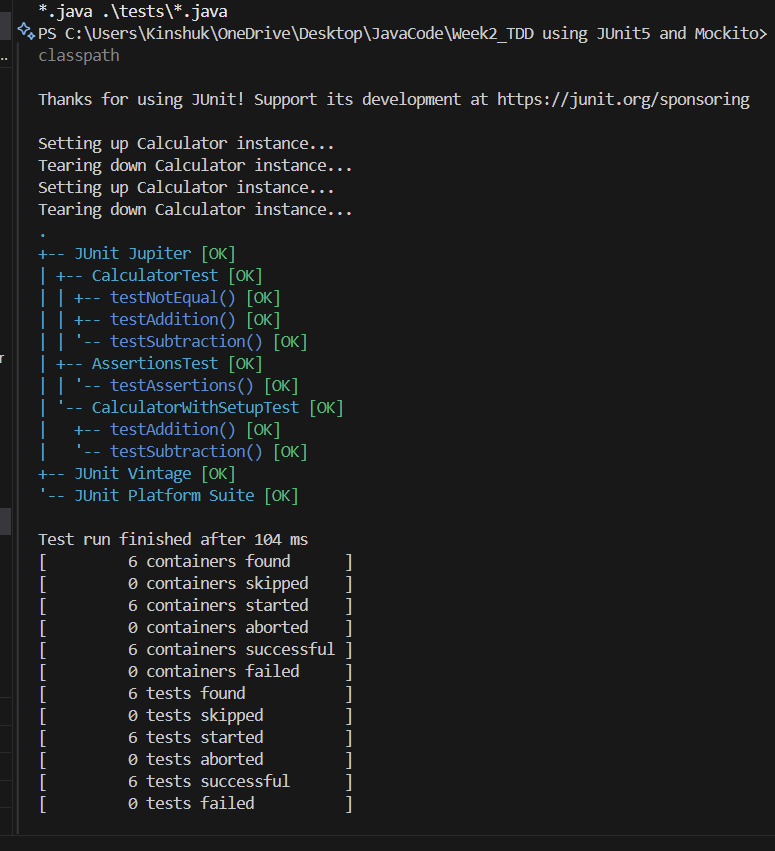
}

Command

> javac -cp ".\lib\junit-platform-console-standalone-1.10.2.jar" -d bin .\src\\*.java .\tests\\*.java

Command

> java -jar .\lib\junit-platform-console-standalone-1.10.2.jar -cp bin --scan-classpath



**MOCKITIO**

**Exercise 1 & 2:**

**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 org.junit.jupiter.api.Test;

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

import static org.mockito.Mockito.\*;

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);

    }

    @Test

    public void testVerifyInteraction() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.fetchData();

        verify(mockApi).getData();  // Verifies the method was called

    }

}

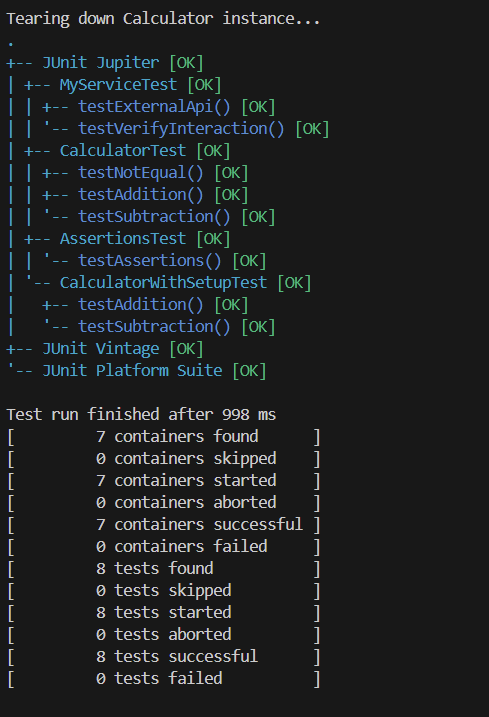
**COMMANDS:**

>javac -cp "lib/\*" -d bin src/\*.java tests/\*.java

>java "-Dnet.bytebuddy.experimental=true" -cp "lib/\*;bin" org.junit.platform.console.ConsoleLauncher --scan-classpath

We had to run experimental mode because our java 24 not supported by bytebuddies

**Exercise 1 & 2 OUTPUT:**



**SL4J Logging exercises**

Exercise 1:

Code:

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 from Kinshuk");

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

    }

}

COMMAND

javac -cp "lib/\*" -d bin src/LoggingExample.java

java -cp "lib/\*;bin" LoggingExample

OUTPUT

