PL/SQL programming

PLSQL\_Exercises

Exercise 1: Control Structures

Creating the required tables

CREATE TABLE customers (

    customer\_id NUMBER PRIMARY KEY,

    name VARCHAR2(100),

    age NUMBER

);

CREATE TABLE loans (

    loan\_id NUMBER PRIMARY KEY,

    customer\_id NUMBER REFERENCES customers(customer\_id),

    interest\_rate NUMBER(5, 2)

);

SET SERVEROUTPUT ON;

DECLARE

CURSOR senior\_customers IS

SELECT c.customer\_id

FROM customers c

WHERE c.age > 60;

BEGIN

FOR cust IN senior\_customers LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = cust.customer\_id;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Interest rate discount applied to senior customers.');

EXCEPTION

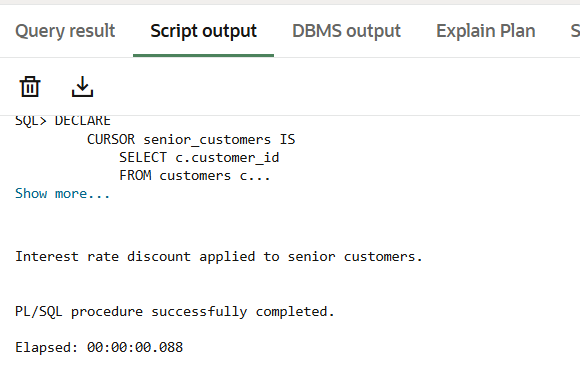
WHEN OTHERS THEN

ROLLBACK;

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

END;

Output:



Exercise 3: Stored Procedures

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    UPDATE accounts

    SET balance = balance + (balance \* 0.01)

    WHERE UPPER(account\_type) = 'SAVINGS';

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Monthly interest processed successfully.');

EXCEPTION

    WHEN OTHERS THEN

        ROLLBACK;

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

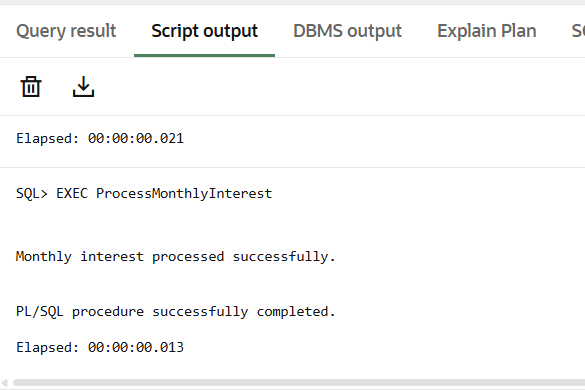
END;

/

SET SERVEROUTPUT ON;

EXEC ProcessMonthlyInterest;

Output:



|  |  |
| --- | --- |
| 1. JUnit\_Basic Testing Exercises | Exercise 1: Setting Up JUnit |
| 1. JUnit\_Basic Testing Exercises | Exercise 3: Assertions in JUnit |
| 1. JUnit\_Basic Testing Exercises | Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and |
| Teardown Methods in JUnit |

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

CALCULATOR.JAVA

package Calculator;

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

package Calculator;

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));

}

*@Test*

public void testSubtract() {

Calculator calc = new Calculator();

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

}

}

ASSERTIONSTEST.JAVA

package Calculator;

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);

*assertNull*(null);

*assertNotNull*(new Object());

}

}

ADVANCEDCALCULATORTEST.JAVA

package Calculator;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AdvancedCalculatorTest {

private Calculator calc;

*@Before*

public void setUp() {

calc = new Calculator();

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

}

*@After*

public void tearDown() {

System.***out***.println("Test finished.");

}

*@Test*

public void testAdd() {

int result = calc.add(10, 5);

*assertEquals*(15, result);

}

*@Test*

public void testSubtract() {

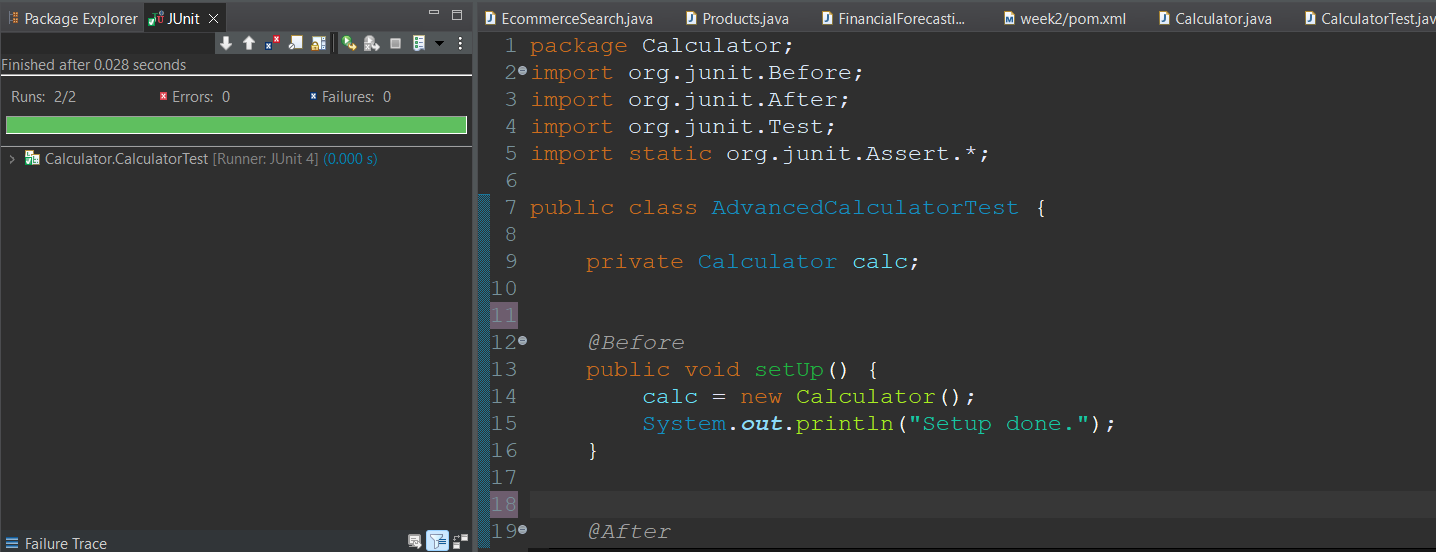
int result = calc.subtract(10, 4);

*assertEquals*(6, result);

}

}

OUTPUT:



|  |  |
| --- | --- |
| 3. Mockito exercises | Exercise 1: Mocking and Stubbing |
| 3. Mockito exercises | Exercise 2: Verifying Interactions |
| 6. SL4J Logging exercises | Exercise 1: Logging Error Messages and Warning Levels |

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.12.0</version>

<scope>test</scope>

</dependency>

</dependencies>

EXTERNALAPI.JAVA

package cts.mockitoex;

public interface ExternalApi {

String getData();

}

MYSERVICCE.JAVA

package cts.mockitoex;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

MYSERVICETEST.JAVA

package cts.mockitoex;

import static org.mockito.Mockito.\*;

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

import org.junit.Test;

public class MyServiceTest {

*@Test*

public void testExternalApi() {

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

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

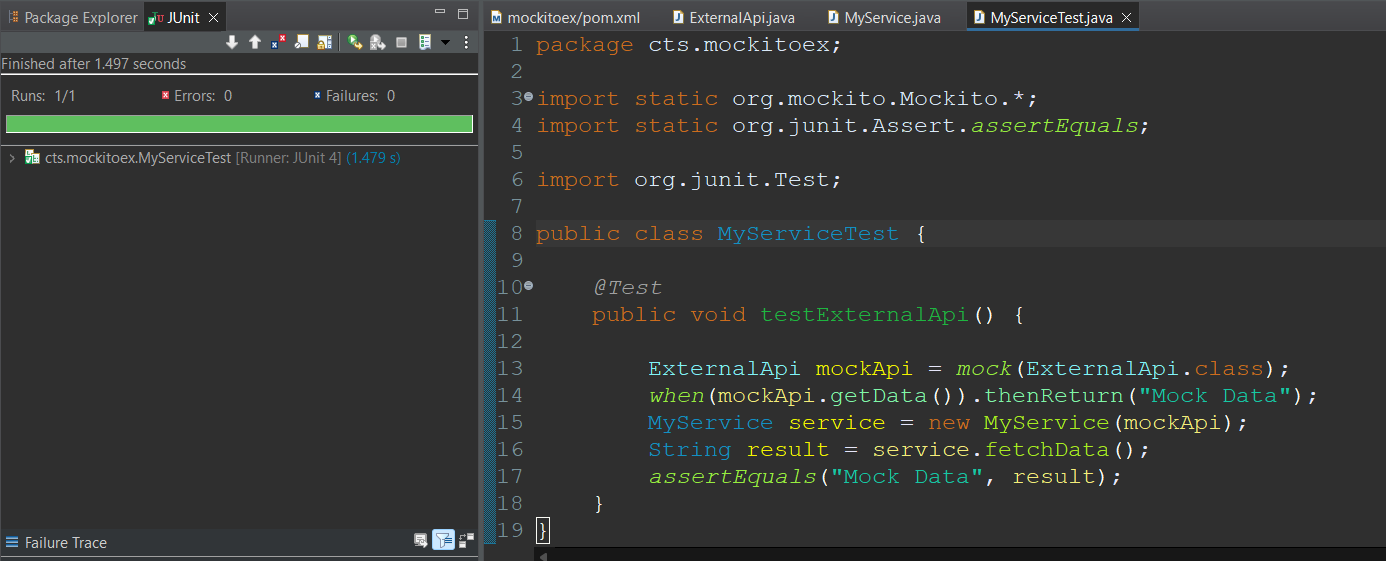
MyService service = new MyService(mockApi);

String result = service.fetchData();

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

}

}



EXERCISE 2 Exercise 2: Verifying Interactions

package cts.mockitoex;

import static org.mockito.Mockito.\*;

import static org.junit.Assert.\*;

import org.junit.Test;

public class MyServiceTest {

*@Test*

public void testVerifyInteraction() {

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

MyService service = new MyService(mockApi);

service.fetchData();

*verify*(mockApi).getData();

}

}

