**CTS – DIGITAL NUTURE – 4.0 - DEEPSKILLING PROGRAM**

**WEEK – 2**

**Exercise 2: Verifying Interactions**

Solution:

**package** com.cts.Mockito;

**public** **interface** ExternalAPI {

String getData();

}

**package** com.cts.Mockito;

**public** **class** MyService {

**private** ExternalAPI api;

**public** MyService(ExternalAPI api) {

**this**.api = api;

}

**public** String fetchData() {

**return** api.getData();

}

}

**package** com.cts.Mockito;

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

**import** org.mockito.Mockito;

**import** **static** org.mockito.Mockito.\*;

**public** **class** MyServiceTest {

@Test

**public** **void** testVerifyInteraction() {

// Step 1: Create mock object

ExternalAPI mockApi = Mockito.*mock*(ExternalAPI.**class**);

MyService service = **new** MyService(mockApi);

//2.Call the method

service.fetchData();

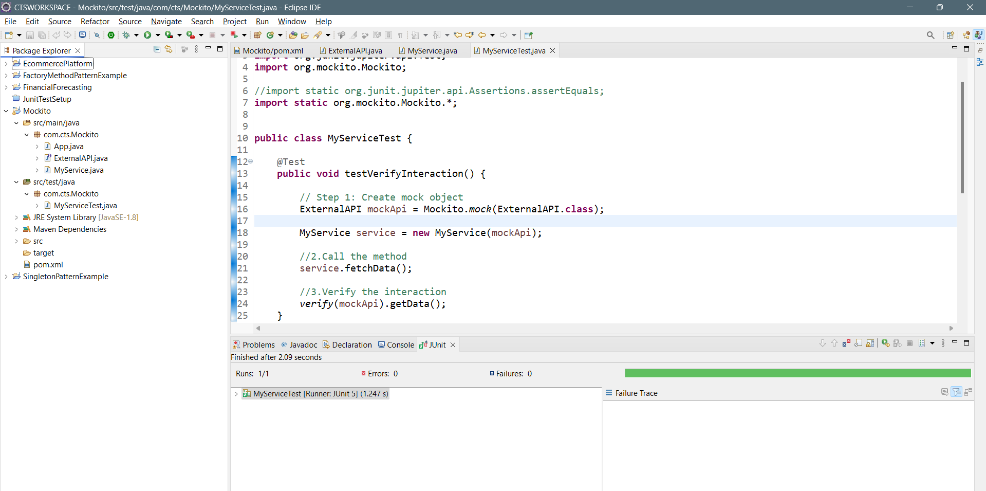
//3.Verify the interaction

*verify*(mockApi).getData();

}

}

Output:



**Exercise 3: Assertions in Junit**

**Solution Code:**

package com.cts.JunitTestSetup;

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

import org.junit.jupiter.api.Test;

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());

}

}

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 3: Stored Procedures**

CREATE TABLE customers (

customer\_id INTEGER PRIMARY KEY,

name VARCHAR(50),

age INTEGER,

balance NUMERIC(12,2),

isvip VARCHAR(5)

);

CREATE TABLE loans (

loan\_id INTEGER PRIMARY KEY,

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

interest\_rate NUMERIC(5,2),

due\_date DATE

);

CREATE TABLE employees (

employee\_id INTEGER PRIMARY KEY,

name VARCHAR(50),

department\_id INTEGER,

salary NUMERIC(10,2)

);

CREATE TABLE accounts (

account\_id INTEGER PRIMARY KEY,

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

account\_type VARCHAR(20),

balance NUMERIC(12,2)

);

INSERT INTO customers VALUES (1, 'Alice', 65, 15000.00, 'FALSE');

INSERT INTO customers VALUES (2, 'Bob', 45, 9500.00, 'FALSE');

INSERT INTO customers VALUES (3, 'Charlie', 70, 12000.00, 'FALSE');

INSERT INTO customers VALUES (4, 'Diana', 30, 10500.00, 'FALSE');

INSERT INTO loans VALUES (101, 1, 8.5, CURRENT\_DATE + 10);

INSERT INTO loans VALUES (102, 2, 7.9, CURRENT\_DATE + 40);

INSERT INTO loans VALUES (103, 3, 9.0, CURRENT\_DATE + 5);

INSERT INTO loans VALUES (104, 4, 7.5, CURRENT\_DATE - 3);

INSERT INTO loans VALUES (105, 1, 8.0, CURRENT\_DATE + 20);

INSERT INTO employees VALUES (201, 'Ravi', 10, 40000.00);

INSERT INTO employees VALUES (202, 'Neha', 10, 45000.00);

INSERT INTO employees VALUES (203, 'Anil', 20, 50000.00);

INSERT INTO accounts VALUES (1001, 1, 'SAVINGS', 5000.00);

INSERT INTO accounts VALUES (1002, 2, 'SAVINGS', 3000.00);

INSERT INTO accounts VALUES (1003, 3, 'CURRENT', 8000.00);

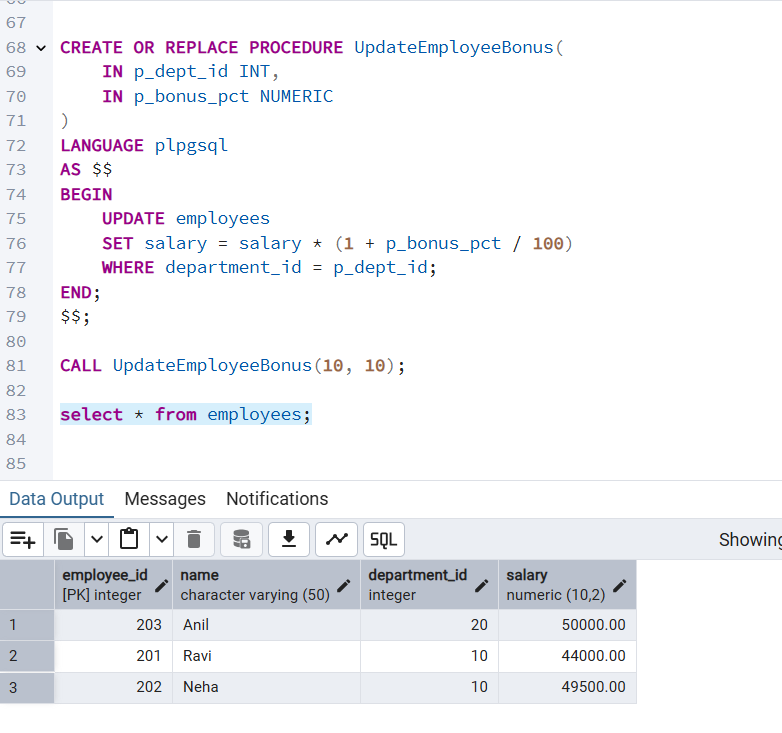
INSERT INTO accounts VALUES (1004, 4, 'SAVINGS', 2000.00);

**Scenario 1:**

A screenshot of a computer

Description automatically generated

**Scenario 2:**



**Scenario 3:**

CREATE OR REPLACE PROCEDURE TransferFunds(

IN p\_from\_acct INT,

IN p\_to\_acct INT,

IN p\_amount NUMERIC

)

LANGUAGE plpgsql

AS $$

DECLARE

v\_balance NUMERIC;

BEGIN

IF p\_amount <= 0 THEN

RAISE EXCEPTION 'Amount must be greater than zero.';

END IF;

SELECT balance INTO v\_balance FROM accounts WHERE account\_id = p\_from\_acct FOR UPDATE;

IF v\_balance < p\_amount THEN

RAISE EXCEPTION 'Insufficient balance in source account.';

END IF;

UPDATE accounts SET balance = balance - p\_amount WHERE account\_id = p\_from\_acct;

UPDATE accounts SET balance = balance + p\_amount WHERE account\_id = p\_to\_acct;

END;

$$;

CALL TransferFunds(1001, 1002, 1000);

select \* from accounts;

A screenshot of a computer

AI-generated content may be incorrect.

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

**Calculator.java**

**package** com.cts.JunitTestSetup;

**public** **class** Calculator

{

**public** **int** add(**int** a, **int** b) {

**return** a + b;

}

**public** **int** subtract(**int** a, **int** b) {

**return** a - b;

}

**public** **int** multiply(**int** a, **int** b) {

**return** a \* b;

}

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

**if** (b == 0) {

**throw** **new** ArithmeticException("Cannot divide by zero.");

}

**return** a / b;

}

**public** **boolean** isEven(**int** number) {

**return** number % 2 == 0;

}

**public** **int** square(**int** number) {

**return** number \* number;

}

}

**CalculatorTest.java**

**package** com.cts.JunitTestSetup;

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

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

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

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

**public** **class** CalculatorTest {

**private** Calculator calculator;

@BeforeEach

**void** setUp() {

calculator = **new** Calculator();

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

}

@AfterEach

**void** tearDown() {

calculator = **null**;

System.***out***.println("Teardown completed");

}

@Test

**void** testAddition() {

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

System.***out***.println("Addition Result: " + result);

*assertEquals*(15, result);

}

@Test

**void** testSubtraction() {

*assertEquals*(5, calculator.subtract(10, 5));

}

@Test

**void** testMultiplication() {

*assertEquals*(12, calculator.multiply(4, 3));

}

@Test

**void** testDivision() {

*assertEquals*(5, calculator.divide(20, 4));

}

@Test

**void** testDivisionByZero() {

*assertThrows*(ArithmeticException.**class**, () -> calculator.divide(10, 0));

}

@Test

**void** testIsEvenTrue() {

*assertTrue*(calculator.isEven(8));

}

@Test

**void** testIsEvenFalse() {

*assertFalse*(calculator.isEven(7));

}

@Test

**void** testSquare() {

*assertEquals*(36, calculator.square(6));

}

}

Output:

A screenshot of a computer

AI-generated content may be incorrect.

