**Week -2 HandsOn Solutions (Mandatory)**

**PL/SQL**

**Exercise:1 Control Structures**

Program

-- Step 1: Create tables only if not exists

BEGIN

EXECUTE IMMEDIATE '

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

age NUMBER,

balance NUMBER(10,2),

is\_vip CHAR(1) DEFAULT ''N''

)';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -955 THEN -- -955: object already exists

RAISE;

END IF;

END;

/

BEGIN

EXECUTE IMMEDIATE '

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

interest\_rate NUMBER(5,2),

due\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

)';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -955 THEN

RAISE;

END IF;

END;

/

-- Step 2: Upsert sample data (avoids duplicate primary key error)

MERGE INTO customers c

USING (SELECT 1 AS customer\_id, 'Alice' AS name, 65 AS age, 12000 AS balance, 'N' AS is\_vip FROM dual) src

ON (c.customer\_id = src.customer\_id)

WHEN NOT MATCHED THEN

INSERT (customer\_id, name, age, balance, is\_vip)

VALUES (src.customer\_id, src.name, src.age, src.balance, src.is\_vip);

MERGE INTO customers c

USING (SELECT 2 AS customer\_id, 'Bob' AS name, 45 AS age, 8000 AS balance, 'N' AS is\_vip FROM dual) src

ON (c.customer\_id = src.customer\_id)

WHEN NOT MATCHED THEN

INSERT (customer\_id, name, age, balance, is\_vip)

VALUES (src.customer\_id, src.name, src.age, src.balance, src.is\_vip);

MERGE INTO customers c

USING (SELECT 3 AS customer\_id, 'Charlie' AS name, 70 AS age, 5000 AS balance, 'N' AS is\_vip FROM dual) src

ON (c.customer\_id = src.customer\_id)

WHEN NOT MATCHED THEN

INSERT (customer\_id, name, age, balance, is\_vip)

VALUES (src.customer\_id, src.name, src.age, src.balance, src.is\_vip);

MERGE INTO loans l

USING (SELECT 101 AS loan\_id, 1 AS customer\_id, 7.5 AS interest\_rate, SYSDATE + 15 AS due\_date FROM dual) src

ON (l.loan\_id = src.loan\_id)

WHEN NOT MATCHED THEN

INSERT (loan\_id, customer\_id, interest\_rate, due\_date)

VALUES (src.loan\_id, src.customer\_id, src.interest\_rate, src.due\_date);

MERGE INTO loans l

USING (SELECT 102 AS loan\_id, 2 AS customer\_id, 6.0 AS interest\_rate, SYSDATE + 45 AS due\_date FROM dual) src

ON (l.loan\_id = src.loan\_id)

WHEN NOT MATCHED THEN

INSERT (loan\_id, customer\_id, interest\_rate, due\_date)

VALUES (src.loan\_id, src.customer\_id, src.interest\_rate, src.due\_date);

MERGE INTO loans l

USING (SELECT 103 AS loan\_id, 3 AS customer\_id, 8.0 AS interest\_rate, SYSDATE + 5 AS due\_date FROM dual) src

ON (l.loan\_id = src.loan\_id)

WHEN NOT MATCHED THEN

INSERT (loan\_id, customer\_id, interest\_rate, due\_date)

VALUES (src.loan\_id, src.customer\_id, src.interest\_rate, src.due\_date);

/

-- Step 3: Main Logic Block (All 3 Scenarios)

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Scenario 1: Interest Discount for Customers Above 60 ---');

FOR cust IN (

SELECT c.customer\_id, c.name, l.loan\_id

FROM customers c

JOIN loans l ON c.customer\_id = l.customer\_id

WHERE c.age > 60

) LOOP

UPDATE loans

SET interest\_rate = interest\_rate - (interest\_rate \* 0.01)

WHERE loan\_id = cust.loan\_id;

DBMS\_OUTPUT.PUT\_LINE('✔ Discount applied to ' || cust.name || ' for Loan ID: ' || cust.loan\_id);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(CHR(10) || '--- Scenario 2: Promote Customers to VIP ---');

FOR cust IN (

SELECT customer\_id, name, balance

FROM customers

WHERE balance > 10000 AND is\_vip = 'N'

) LOOP

UPDATE customers

SET is\_vip = 'Y'

WHERE customer\_id = cust.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('✔ Customer ' || cust.name || ' promoted to VIP. Balance: $' || cust.balance);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(CHR(10) || '--- Scenario 3: Loan Due Reminders ---');

FOR loan\_rec IN (

SELECT l.loan\_id, c.name, l.due\_date

FROM loans l

JOIN customers c ON l.customer\_id = c.customer\_id

WHERE l.due\_date <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('📢 Reminder: Loan ID ' || loan\_rec.loan\_id || ' for ' || loan\_rec.name ||

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

END LOOP;

END;

/

Output:

A screenshot of a computer

AI-generated content may be incorrect.

Exercise:3 Stored Procedures

-- Step 1: Create Tables (if not exists)

BEGIN

EXECUTE IMMEDIATE 'CREATE TABLE savings\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER(10,2)

)';

EXCEPTION

WHEN OTHERS THEN IF SQLCODE != -955 THEN RAISE; END IF;

END;

/

BEGIN

EXECUTE IMMEDIATE 'CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER(10,2)

)';

EXCEPTION

WHEN OTHERS THEN IF SQLCODE != -955 THEN RAISE; END IF;

END;

/

BEGIN

EXECUTE IMMEDIATE 'CREATE TABLE bank\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER(10,2)

)';

EXCEPTION

WHEN OTHERS THEN IF SQLCODE != -955 THEN RAISE; END IF;

END;

/

-- Step 2: Sample Data (MERGE avoids duplicate insert errors)

MERGE INTO savings\_accounts s

USING (SELECT 1 account\_id, 101 customer\_id, 1000 balance FROM dual) d

ON (s.account\_id = d.account\_id)

WHEN NOT MATCHED THEN INSERT (account\_id, customer\_id, balance)

VALUES (d.account\_id, d.customer\_id, d.balance);

MERGE INTO savings\_accounts s

USING (SELECT 2 account\_id, 102 customer\_id, 2000 balance FROM dual) d

ON (s.account\_id = d.account\_id)

WHEN NOT MATCHED THEN INSERT (account\_id, customer\_id, balance)

VALUES (d.account\_id, d.customer\_id, d.balance);

MERGE INTO employees e

USING (SELECT 1 employee\_id, 'John' name, 'IT' department, 50000 salary FROM dual) d

ON (e.employee\_id = d.employee\_id)

WHEN NOT MATCHED THEN INSERT (employee\_id, name, department, salary)

VALUES (d.employee\_id, d.name, d.department, d.salary);

MERGE INTO employees e

USING (SELECT 2 employee\_id, 'Jane' name, 'HR' department, 45000 salary FROM dual) d

ON (e.employee\_id = d.employee\_id)

WHEN NOT MATCHED THEN INSERT (employee\_id, name, department, salary)

VALUES (d.employee\_id, d.name, d.department, d.salary);

MERGE INTO bank\_accounts b

USING (SELECT 101 account\_id, 1 customer\_id, 3000 balance FROM dual) d

ON (b.account\_id = d.account\_id)

WHEN NOT MATCHED THEN INSERT (account\_id, customer\_id, balance)

VALUES (d.account\_id, d.customer\_id, d.balance);

MERGE INTO bank\_accounts b

USING (SELECT 102 account\_id, 1 customer\_id, 1500 balance FROM dual) d

ON (b.account\_id = d.account\_id)

WHEN NOT MATCHED THEN INSERT (account\_id, customer\_id, balance)

VALUES (d.account\_id, d.customer\_id, d.balance);

/

-- Step 3: Stored Procedure 1: ProcessMonthlyInterest

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (SELECT account\_id, balance FROM savings\_accounts) LOOP

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_id = acc.account\_id;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('✔ Monthly interest processed for all savings accounts.');

END;

/

-- Step 4: Stored Procedure 2: UpdateEmployeeBonus

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE employees

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

WHERE department = dept\_name;

DBMS\_OUTPUT.PUT\_LINE('✔ Bonus of ' || bonus\_percent || '% applied to department: ' || dept\_name);

END;

/

-- Step 5: Stored Procedure 3: TransferFunds

CREATE OR REPLACE PROCEDURE TransferFunds(

from\_account IN NUMBER,

to\_account IN NUMBER,

amount IN NUMBER

) AS

from\_balance NUMBER;

BEGIN

SELECT balance INTO from\_balance FROM bank\_accounts WHERE account\_id = from\_account FOR UPDATE;

IF from\_balance < amount THEN

RAISE\_APPLICATION\_ERROR(-20001, '❌ Insufficient funds in source account.');

END IF;

UPDATE bank\_accounts SET balance = balance - amount WHERE account\_id = from\_account;

UPDATE bank\_accounts SET balance = balance + amount WHERE account\_id = to\_account;

DBMS\_OUTPUT.PUT\_LINE('✔ Transferred ₹' || amount || ' from account ' || from\_account || ' to ' || to\_account);

END;

/

-- Step 6: Procedure Execution (Test Calls)

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Running Stored Procedures ---');

-- 1. Apply Monthly Interest

ProcessMonthlyInterest;

-- 2. Update Bonus for IT Dept

UpdateEmployeeBonus('IT', 10); -- 10% bonus

-- 3. Transfer ₹500 from account 101 to 102

TransferFunds(101, 102, 500);

END;

/

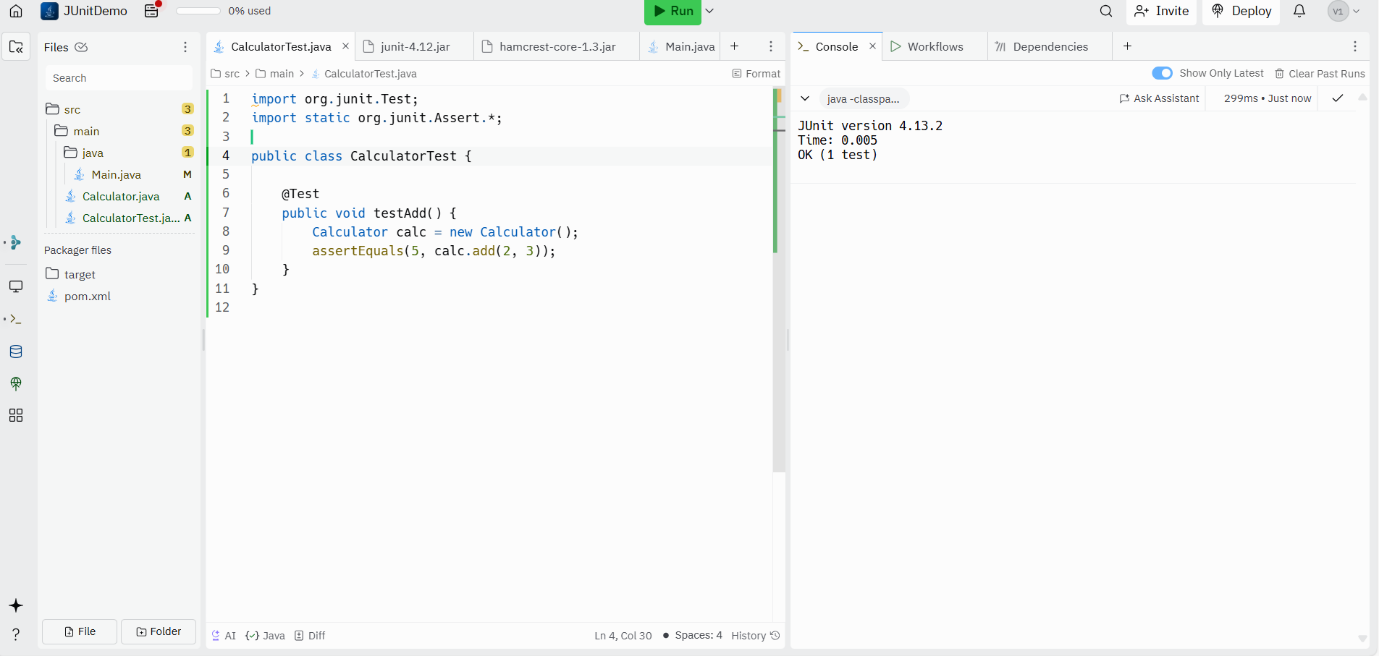
Output:

A screenshot of a computer screen

AI-generated content may be incorrect.

**Junit basic testing exercises**

Exercise:1 Setting up Junit

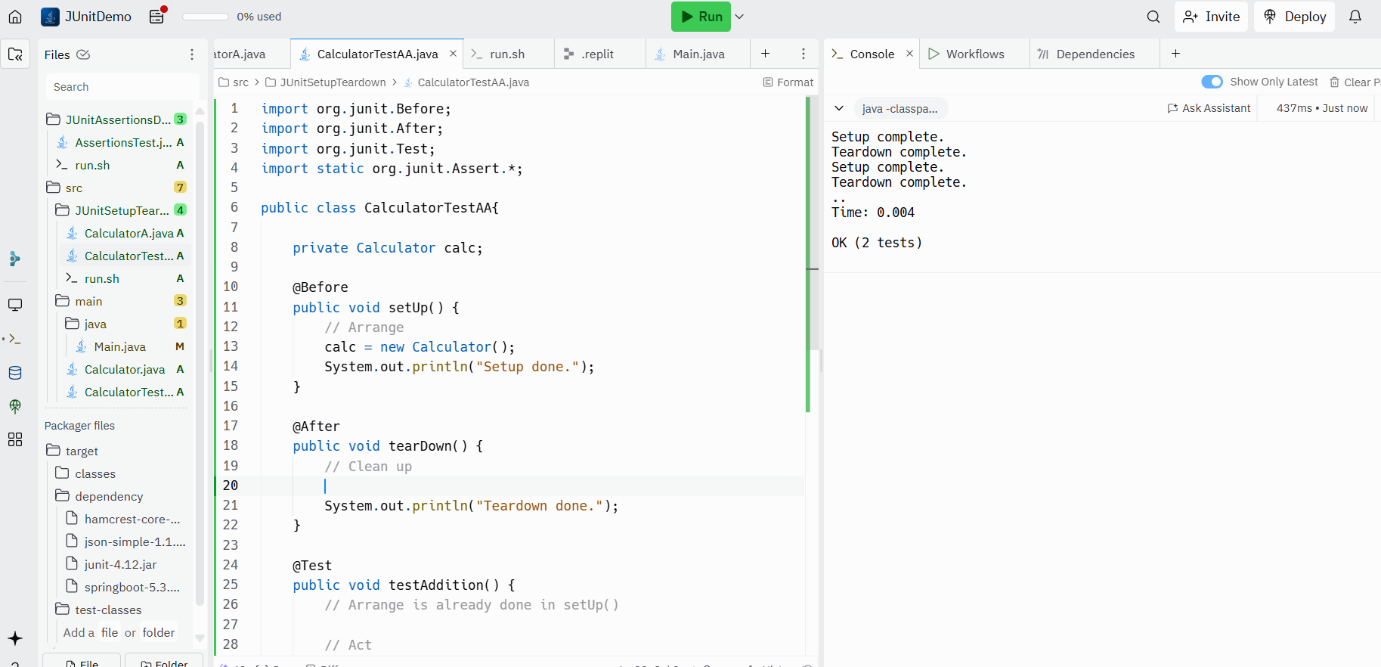


Exercise:3 Assertions in Junit

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



**Mockito Exercises Solutions**

A screenshot of a computer

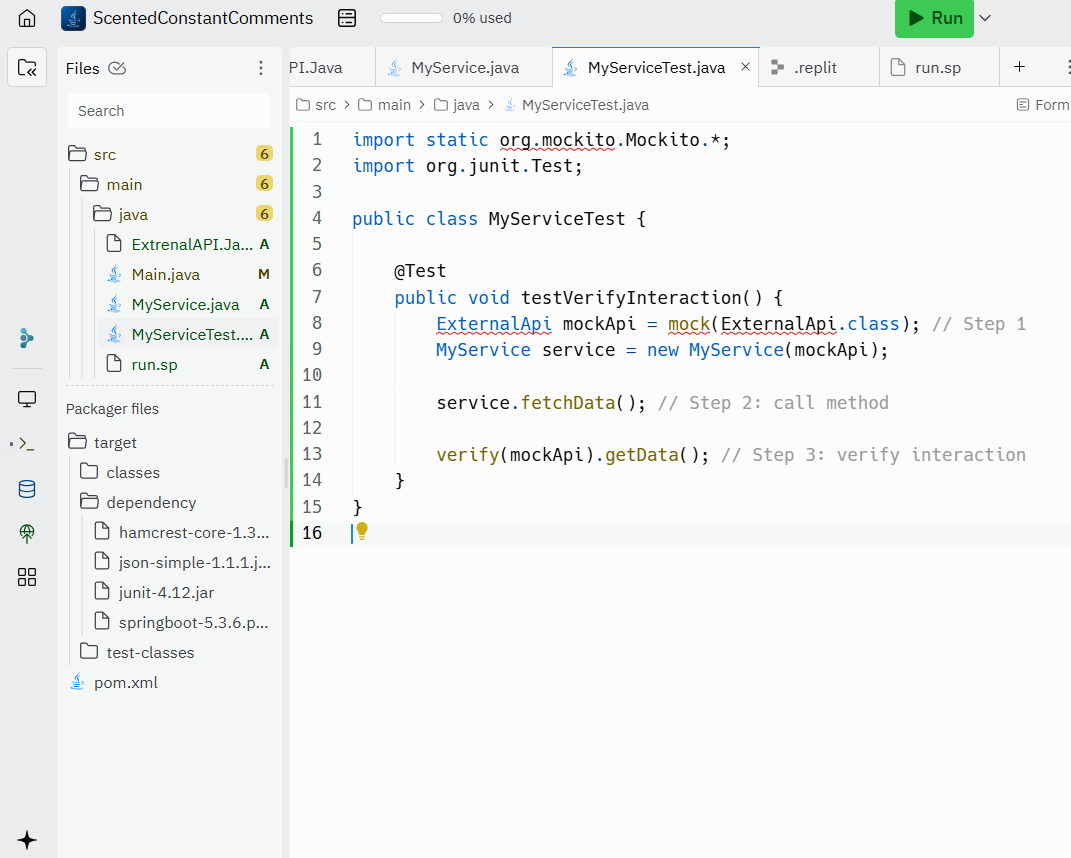
AI-generated content may be incorrect.Exercise:1 Mocking and Stubbing

Output

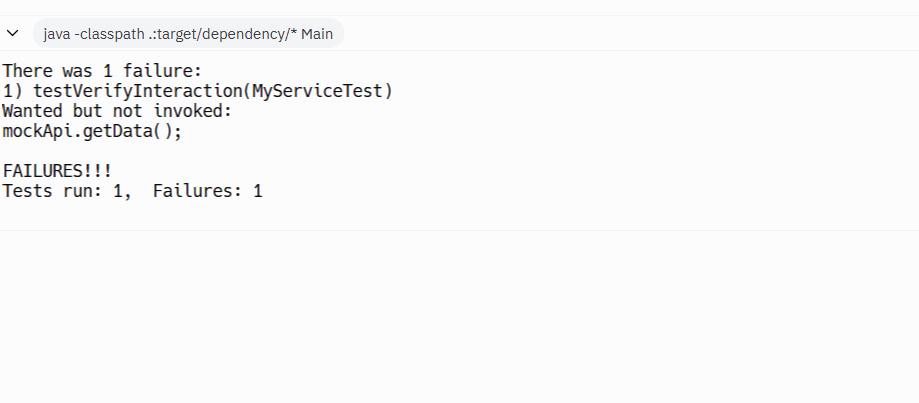
A screenshot of a computer

AI-generated content may be incorrect.

Exercise:2 Verifying Interactions



Output:



**Logging using SLF4J Solutions**

Exercise:1 Logging error messages and warning levels



A screenshot of a computer

AI-generated content may be incorrect.