CHALLA VENKATA ANIRUDH

PL/SQL

EXERCISE 1: CONTROL STRUCTURES

-

SCENARIO 1: THE BANK WANTS TO APPLY A DISCOUNT TO LOAN INTEREST RATES FOR CUSTOMERS ABOVE 60 YEARS OLD.

- QUESTION: WRITE A PL/SQL BLOCK THAT LOOPS THROUGH ALL CUSTOMERS, CHECKS THEIR AGE, AND IF THEY ARE ABOVE 60, APPLY A 1% DISCOUNT TO THEIR CURRENT LOAN INTEREST RATES.

SCENARIO 2: A CUSTOMER CAN BE PROMOTED TO VIP STATUS BASED ON THEIR BALANCE.

- QUESTION: WRITE A PL/SQL BLOCK THAT ITERATES THROUGH ALL CUSTOMERS AND SETS A FLAG ISVIP TO TRUE FOR THOSE WITH A BALANCE OVER $10,000.

SCENARIO 3: THE BANK WANTS TO SEND REMINDERS TO CUSTOMERS WHOSE LOANS ARE DUE WITHIN THE NEXT 30 DAYS.

- QUESTION: WRITE A PL/SQL BLOCK THAT FETCHES ALL LOANS DUE IN THE NEXT 30 DAYS AND PRINTS A REMINDER MESSAGE FOR EACH CUSTOMER.

SCENARIO 1:-

```
BEGIN
  FOR cust IN (
    SELECT customer_id
    FROM customers
    WHERE age > 60
  ) LOOP
    UPDATE loans
    SET interest_rate = interest_rate - 1
    WHERE customer_id = cust.customer_id;
  END LOOP;

  COMMIT;
END;
```

SCENARIO 2:-

```
BEGIN
   FOR cust IN (
      SELECT customer_id
      FROM customers
      WHERE balance > 10000
   ) LOOP
      UPDATE customers
      SET isvip = 'TRUE'
      WHERE customer_id = cust.customer_id;
   END LOOP;

   COMMIT;
END;
```

SCENARIO 3:-

```
BEGIN
  FOR loan_rec IN (
    SELECT c.customer_id, c.name, l.due_date
    FROM customers c
    JOIN loans l ON c.customer_id = l.customer_id
    WHERE l.due_date BETWEEN SYSDATE AND SYSDATE + 30
  ) LOOP
    DBMS_OUTPUT.PUT_LINE(
      'Reminder: Loan for customer ' || loan_rec.name ||
      ' (ID: ' || loan_rec.customer_id || ') is due on ' || TO_CHAR(loan_rec.due_date, 'DD-MON-YYYY')
    );
  END LOOP;
END;
```

EXERCISE 3: STORED PROCEDURES

-

SCENARIO 1: THE BANK NEEDS TO PROCESS MONTHLY INTEREST FOR ALL SAVINGS ACCOUNTS.

- QUESTION: WRITE A STORED PROCEDURE PROCESSMONTHLYINTEREST THAT CALCULATES AND UPDATES THE BALANCE OF ALL SAVINGS ACCOUNTS BY APPLYING AN INTEREST RATE OF 1% TO THE CURRENT BALANCE.

SCENARIO 2: THE BANK WANTS TO IMPLEMENT A BONUS SCHEME FOR EMPLOYEES BASED ON THEIR PERFORMANCE.

- QUESTION: WRITE A STORED PROCEDURE UPDATEEMPLOYEEBONUS THAT UPDATES THE SALARY OF EMPLOYEES IN A GIVEN DEPARTMENT BY ADDING A BONUS PERCENTAGE PASSED AS A PARAMETER.

SCENARIO 3: CUSTOMERS SHOULD BE ABLE TO TRANSFER FUNDS BETWEEN THEIR ACCOUNTS.

- QUESTION: WRITE A STORED PROCEDURE TRANSFERFUNDS THAT TRANSFERS A SPECIFIED AMOUNT FROM ONE ACCOUNT TO ANOTHER, CHECKING THAT THE SOURCE ACCOUNT HAS SUFFICIENT BALANCE BEFORE MAKING THE TRANSFER.

SCENARIO 1:-

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
  UPDATE accounts
  SET balance = balance + (balance * 0.01)
  WHERE account_type = 'SAVINGS';

  COMMIT;
END;
```

SCENARIO 2:-

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
  dept_id IN NUMBER,
  bonus_percent IN NUMBER
) IS
BEGIN
  UPDATE employees
  SET salary = salary + (salary * bonus_percent / 100)
  WHERE department_id = dept_id;

  COMMIT;
END;
```

SCENARIO 3:-

```
CREATE OR REPLACE PROCEDURE TransferFunds (
  from_account IN NUMBER,
  to_account IN NUMBER,
  amount IN NUMBER
) IS
  insufficient_funds EXCEPTION;
BEGIN
  -- Check balance
  DECLARE
    current_balance NUMBER;
  BEGIN
    SELECT balance INTO current_balance FROM accounts WHERE account_id = from_account FOR UPDATE;

    IF current_balance < amount THEN
      RAISE insufficient_funds;
    END IF;

    -- Debit from source account
    UPDATE accounts
    SET balance = balance - amount
    WHERE account_id = from_account;
```

```
    -- Credit to destination account
    UPDATE accounts
    SET balance = balance + amount
    WHERE account_id = to_account;

    COMMIT;
  END;

EXCEPTION
  WHEN insufficient_funds THEN
    DBMS_OUTPUT.PUT_LINE('Transfer failed: Insufficient balance.');
    ROLLBACK;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
    ROLLBACK;
END;
```

JUNIT5

EXERCISE 1: SETTING UP JUNIT

SCENARIO:

YOU NEED TO SET UP JUNIT IN YOUR JAVA PROJECT TO START WRITING UNIT TESTS.

STEPS:

1. CREATE A NEW JAVA PROJECT IN YOUR IDE (E.G., INTELLIJ IDEA, ECLIPSE).
2. ADD JUNIT DEPENDENCY TO YOUR PROJECT.
3. CREATE A NEW TEST CLASS IN YOUR PROJECT

EXERCISE 1: SETTING UP JUNIT

SCENARIO:

YOU NEED TO SET UP JUNIT IN YOUR JAVA PROJECT TO START WRITING UNIT TESTS.

STEPS:

1. CREATE A NEW JAVA PROJECT IN YOUR IDE (E.G., INTELLIJ IDEA, ECLIPSE).

2. ADD JUNIT DEPENDENCY TO YOUR PROJECT.

3. CREATE A NEW TEST CLASS IN YOUR PROJECT

EXERCISE 3: ASSERTIONS IN JUNIT

SCENARIO:

YOU NEED TO USE DIFFERENT ASSERTIONS IN JUNIT TO VALIDATE YOUR TEST RESULTS.

STEPS:

1. WRITE TESTS USING VARIOUS JUNIT ASSERTIONS.

EXERCISE 4: ARRANGE-ACT-ASSERT (AAA) PATTERN, TEST FIXTURES, SETUP AND TEARDOWN METHODS IN JUNIT

SCENARIO:

YOU NEED TO ORGANIZE YOUR TESTS USING THE ARRANGE-ACT-ASSERT (AAA) PATTERN AND USE SETUP AND TEARDOWN METHODS.

STEPS:

1. WRITE TESTS USING THE AAA PATTERN.

2. USE @BEFORE AND @AFTER ANNOTATIONS FOR SETUP AND TEARDOWN METHODS.

TOTAL CODE:-

JUnit > src > J Calculator.java > ...

```java
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }

    public int subtract(int a, int b) {
        return a - b;
    }
}

```

JUnit > src > J CalculatorTest.java > ...

```java
import org.junit.Before;
import org.junit.After;
import org.junit.Test;
import static org.junit.Assert.*;

public class CalculatorTest {

    private Calculator calc;

    @Before
    public void setUp() {
        calc = new Calculator();
    }

    @After
    public void tearDown() {
        calc = null;
    }

    @Test
    public void testAdd() {
        assertEquals(5, calc.add(a:2, b:3));
    }

    @Test
    public void testSubtract() {
        assertEquals(1, calc.subtract(a:4, b:3));
    }
}

```

MOCKITO

EXERCISE 1: MOCKING AND STUBBING

SCENARIO:

YOU NEED TO TEST A SERVICE THAT DEPENDS ON AN EXTERNAL API. USE MOCKITO TO MOCK THE EXTERNAL API AND STUB ITS METHODS.

STEPS:

1. CREATE A MOCK OBJECT FOR THE EXTERNAL API.
2. STUB THE METHODS TO RETURN PREDEFINED VALUES.
3. WRITE A TEST CASE THAT USES THE MOCK OBJECT.

EXERCISE 2: VERIFYING INTERACTIONS

SCENARIO:

YOU NEED TO ENSURE THAT A METHOD IS CALLED WITH SPECIFIC ARGUMENTS.

STEPS:

1. CREATE A MOCK OBJECT.
2. CALL THE METHOD WITH SPECIFIC ARGUMENTS.
3. VERIFY THE INTERACTION.

TOTAL CODE:-

Mockito > src > ExternalApi.java > ...

```java
public interface ExternalApi {
    String getData();
}

```

Mockito > src > MyService.java > ...

```java
public class MyService {
    private ExternalApi api;

    public MyService(ExternalApi api) {
        this.api = api;
    }

    public String fetchData() {
        return api.getData();
    }
}

```

Mockito > src > MyServiceTest.java > Java > MyServiceTest > testExternalApi()

```java
import static org.junit.Assert.*;
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();

        System.out.println("Result: " + result);
        assertEquals("Mock Data", result);
    }
}
```

LOGGING USING SLF4J

EXERCISE 1: LOGGING ERROR MESSAGES AND WARNING LEVELS

TASK:

WRITE A JAVA APPLICATION THAT DEMONSTRATES LOGGING ERROR MESSAGES AND WARNING LEVELS USING SLF4J.

CODE:

ExternalApi.java U | MyService.java U | Logging.java 9+, U

SL4J > src > Logging.java > ...

```java
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Logging {
    private static final Logger logger = LoggerFactory.getLogger(Logging.class);

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        logger.error("This is an error message");
        logger.warn("This is a warning message");
        logger.info("This is an info message");
    }
}

```

```
PS C:\Users\aniru\Engineering Concepts> cd SL4J
PS C:\Users\aniru\Engineering Concepts\SL4J> javac -cp "lib/*" -d bin src/Logging.java
>>
PS C:\Users\aniru\Engineering Concepts\SL4J> java -cp "lib/*;bin" Logging
>>
23:02:10.434 [main] ERROR Logging - This is an error message
23:02:10.436 [main] WARN Logging - This is a warning message
23:02:10.437 [main] INFO Logging - This is an info message
PS C:\Users\aniru\Engineering Concepts\SL4J>
```