PL/SQL

Exercise 1: Control Structures

Customers Table:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1	CUSTOMER_ID	NUMBER	No	(null)	1
2	NAME	VARCHAR2(100 BYTE)	Yes	(null)	2
3	AGE	NUMBER	Yes	(null)	3
4	BALANCE	NUMBER(10,2)	Yes	(null)	4
5	IS_VIP	CHAR(1 BYTE)	Yes	'N'	5

Loans Table:

	LOAN_ID	CUSTOMER_ID	INTEREST_RATE	DUE_DATE
1	101	1	8.5	08/07/25
2	102	2	7	07/08/25
3	103	3	9.5	03/07/25
4	104	4	6	18/07/25
5	105	5	8	27/07/25

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.

Program:

```
SET SERVEROUTPUT ON;

BEGIN
   FOR cust IN (
      SELECT c.customer_id, l.loan_id, l.interest_rate
      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 - 1
      WHERE loan_id = cust.loan_id;

      DBMS_OUTPUT.PUT_LINE('1% discount applied for Customer ID ' || cust.customer_id);
   END LOOP;

   COMMIT;
END;
/
```

Output:

```
1% discount applied for Customer ID 2
1% discount applied for Customer ID 3
1% discount applied for Customer ID 5


PL/SQL procedure successfully completed.
```

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.

Program:

```
SET SERVEROUTPUT ON;

BEGIN
   FOR cust IN (
      SELECT customer_id, balance FROM customers
      WHERE balance > 10000
   ) LOOP
      UPDATE customers
      SET is_vip = 'Y'
      WHERE customer_id = cust.customer_id;
      DBMS_OUTPUT.PUT_LINE('Customer ID ' || cust.customer_id || ' promoted to VIP.');
   END LOOP;

   COMMIT;
END;
/
```

Output:

```
Customer ID 2 promoted to VIP.
Customer ID 4 promoted to VIP.
Customer ID 5 promoted to VIP.


PL/SQL procedure successfully completed.
```

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.

Program:

```
SET SERVEROUTPUT ON;

BEGIN
   FOR loan_rec IN (
      SELECT c.name, l.loan_id, l.due_date
      FROM loans l
      JOIN customers c ON l.customer_id = c.customer_id
      WHERE l.due_date BETWEEN SYSDATE AND 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:

```
Reminder: Loan ID 101 for Ravi Kumar is due on 08-Jul-2025
Reminder: Loan ID 103 for John Thomas is due on 03-Jul-2025
Reminder: Loan ID 104 for Priya Shah is due on 18-Jul-2025
Reminder: Loan ID 105 for Sundar Rajan is due on 27-Jul-2025


PL/SQL procedure successfully completed.
```

Exercise 3: Stored Procedures

Savings Account Table:

	ACCOUNT_ID	CUSTOMER_NAME	BALANCE
1	1	Ravi Kumar	10000
2	2	Anita Mehta	8500
3	3	John Thomas	15000
4	4	Priya Shah	5000

Employee Table:

	EMP_ID	EMP_NAME	SALARY	DEPT_ID
1	101	Vikram Desai	40000	10
2	102	Nisha Verma	45000	20
3	103	Sundar Rajan	38000	10
4	104	Deepa Reddy	50000	30

Customer account Table:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ACCOUNT_ID	NUMBER	No	(null)	1	(null)
2	CUSTOMER_NAME	VARCHAR2(100 BYTE)	Yes	(null)	2	(null)
3	BALANCE	NUMBER(12,2)	Yes	(null)	3	(null)

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.

Program:

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
BEGIN
   FOR acc IN (SELECT account_id, balance FROM savings_accounts) LOOP
      UPDATE savings_accounts
      SET balance = balance * 1.01
      WHERE account_id = acc.account_id;
   END LOOP;

   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Interest applied to all savings accounts.');
END;
/
```

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.

Program:

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
   p_dept_id IN NUMBER,
   p_bonus_percent IN NUMBER
) AS
BEGIN
   UPDATE employees_bonus
   SET salary = salary + (salary * p_bonus_percent / 100)
   WHERE dept_id = p_dept_id;

   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Bonus applied to department ' || p_dept_id);
END;
/
```

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

Program:

```
CREATE OR REPLACE PROCEDURE TransferFunds (
   p_from_acc IN NUMBER,
   p_to_acc   IN NUMBER,
   p_amount   IN NUMBER
) AS
   v_balance NUMBER;
BEGIN
   SELECT balance INTO v_balance FROM customer_accounts WHERE account_id = p_from_acc;

   IF v_balance < p_amount THEN
      RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance.');
   END IF;

   UPDATE customer_accounts SET balance = balance - p_amount WHERE account_id = p_from_acc;
   UPDATE customer_accounts SET balance = balance + p_amount WHERE account_id = p_to_acc;

   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Transferred ' || p_amount || ' from ' || p_from_acc || ' to ' ||
p_to_acc);
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Account not found.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Transfer failed: ' || SQLERRM);
      ROLLBACK;
```

END;
/

Output:

Savings Account Table:

	ACCOUNT_ID	CUSTOMER_NAME	BALANCE
1	1	Ravi Kumar	10100
2	2	Anita Mehta	8585
3	3	John Thomas	15150
4	4	Priya Shah	5050

Employee Table:

	EMP_ID	EMP_NAME	SALARY	DEPT_ID
1	101	Vikram Desai	44000	10
2	102	Nisha Verma	45000	20
3	103	Sundar Rajan	41800	10
4	104	Deepa Reddy	50000	30

Customer account Table:

	ACCOUNT_ID	CUSTOMER_NAME	BALANCE
1	201	Rahul Sen	10000
2	202	Kiran Das	5000
3	203	Meena Joshi	7000

Test driven development and Logging framework

JUnit Testing Exercises

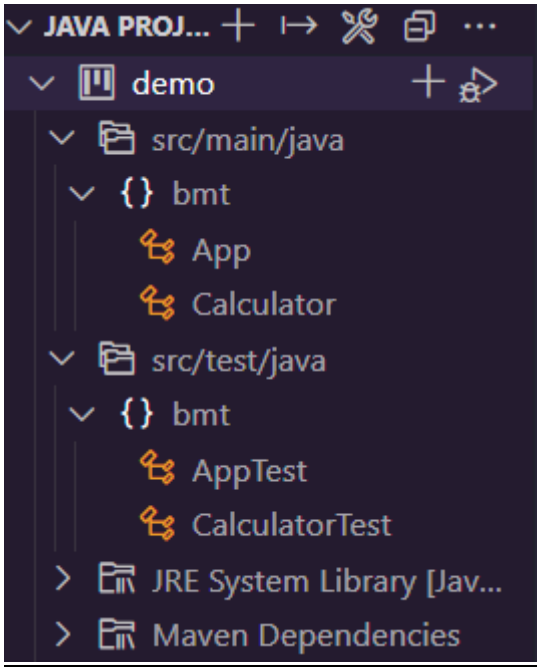
Exercise 1: Setting Up JUnit Scenario:

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. If you are using Maven, add the following to your pom.xml:

```
<dependency>
  <groupId>junit</groupid>
  <artifactid>junit</artifactid >
  <version>4.13.2</ version >
  <scope>test</scope>
</dependency>
```

3. Create a new test class in your project.

Output:



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.
Solution Code:

```
public 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());
   }
 }
```

Output:

```
Running bmt.AssertionsTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.077 s - in bmt.AssertionsTest

Results:

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```

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 method

Programs:
Calculator.java

```
package bmt;

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

```
    }
}
```

CalculatorTest.java

```
package bmt;

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

public class CalculatorTest {
    private Calculator calculator;
    @Before
    public void setUp() {
        calculator = new Calculator();
        System.out.println("Setup: Calculator initialized");
    }
    @After
    public void tearDown() {
        calculator = null;
        System.out.println("Teardown: Calculator cleaned up");
    }
    @Test
    public void testAddition() {
        int a = 10;
        int b = 5;
        int result = calculator.add(a, b);
        assertEquals(15, result);
    }
    @Test
    public void testAdditionWithZero() {
        int a = 0;
        int b = 5;
        int result = calculator.add(a, b);
        assertEquals(5, result);
    }
}
```

Output:

```
[INFO] Running bmt.CalculatorTest
Setup: Calculator initialized
Teardown: Calculator cleaned up
Setup: Calculator initialized
Teardown: Calculator cleaned up
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 s - in bmt.CalculatorTest
```

Mockito exercises

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.

Solution Code:

```
import static org.mockito.Mockito.*;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
public class MyServiceTest {
  @Test
  public void testExternalApi() {
    ExternalApi mockApi = Mockito.mock(ExternalApi.class);
    when(mockApi.getData()).thenReturn("Mock Data");
    MyService service = new MyService(mockApi);
    String result = service.fetchData();
    assertEquals("Mock Data", result);
  }
}
```

Program:

pom.xml setup:

```
</dependency>
<dependency>
  <groupId>org.mockito</groupId>
  <artifactId>mockito-core</artifactId>
  <version>5.11.0</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter</artifactId>
  <version>5.10.2</version>
  <scope>test</scope>
</dependency>
```

Output:

```
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.MyServiceTest
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.217 s - in com.example.MyServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```

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.

Solution Code:

```
import static org.mockito.Mockito.*;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
public class MyServiceTest {
  @Test
  public void testVerifyInteraction() {
  ExternalApi mockApi = Mockito.mock(ExternalApi.class);
  MyService service = new MyService(mockApi);
  service.fetchData();
  verify(mockApi).getData();
  }
}
```

Program:

pom.xml setup:

```
</dependency>
<dependency>
  <groupId>org.mockito</groupId>
  <artifactId>mockito-core</artifactId>
  <version>5.11.0</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter</artifactId>
  <version>5.10.2</version>
  <scope>test</scope>
</dependency>
```

Output:

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.MyServiceTest
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.725 s - in com.example.MyServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
```

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.

Step-by-Step Solution:

1. Add SLF4J and Logback dependencies to your `pom.xml` file:

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.7.30</version>
</dependency>
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>1.2.3</version>
</dependency>
```

2. Create a Java class that uses SLF4J for logging:

```
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");
  logger.warn("This is a warning message");
  }
}
```

Program:

```
package com.example;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class LoggerTester {
    private static final Logger logger = LoggerFactory.getLogger(App.class);
    public static void main(String[] args) {
        logger.info("SLF4J Logging started.");
        logger.warn("This is a warning");
        logger.error("This is an error");
        logger.debug("Debug messages show if enabled in config");
    }
}
```

Output:

```
[main] INFO com.example.App - SLF4J Logging started.
[main] WARN com.example.App - This is a warning
[main] ERROR com.example.App - This is an error
```