**WEEK 2:**

**Module 3 - PL/SQL Programming**

**Exercise 1: Control Structures**

**Task: Write a PL/SQL block to check if a number is even or odd.(Using online oracle live sql)**

**Sql Query:**

DECLARE

num NUMBER := 7;

BEGIN

IF MOD(num, 2) = 0 THEN

DBMS\_OUTPUT.PUT\_LINE(num || ' is even.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(num || ' is odd.');

END IF;

END;

/

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 3: Stored Procedures**

**Task: Create a stored procedure to calculate factorial of a number.**

**Sql Query:**

SET SERVEROUTPUT ON;

DECLARE

   n NUMBER := 5;

   i NUMBER := 1;

   res NUMBER := 1;

BEGIN

   WHILE i <= n LOOP

      res := res \* i;

      i := i + 1;

   END LOOP;

   DBMS\_OUTPUT.PUT\_LINE('Factorial of ' || n || ' is: ' || res);

END;

/

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Module 4 – Test driven development and Logging framework**

**Exercise 1: Setting Up JUnit**

**Add Junit jar files:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Code:**

package com.module4;

import org.junit.jupiter.api.Test;

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

public class CalculatorTest {

@Test

void testAddition() {

Calculator calc = new Calculator();

*assertEquals*(5, calc.add(2, 3));

}

}

class Calculator {

public int add(int a, int b) {

return a + b;}}

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 3: Assertions in Junit**

**Code:**

package com.module4;

import org.junit.jupiter.api.Test;

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

public class StringTest {

@Test

void testStringOperations() {

String str = "JUnit Testing";

*assertNotNull*(str);

*assertTrue*(str.contains("Test"));

*assertEquals*("JUnit Testing", str);

*assertFalse*(str.isEmpty());

}

}

**OUTPUT:**

**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**

**Code:**

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

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

public class BankAccountTest {

BankAccount account;

@BeforeEach

void setUp() {

account = new BankAccount(1000);

}

@Test

void testWithdraw() {

// Arrange done in setUp

// Act

account.withdraw(200);

// Assert

assertEquals(800, account.getBalance());

}

@AfterEach

void tearDown() {

account = null;

}

}

class BankAccount {

private int balance;

public BankAccount(int balance) {

this.balance = balance;

}

public void withdraw(int amount) {

balance -= amount;

}

public int getBalance() {

return balance;

}

}

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 1: Mocking and Stubbing**

**Add Libraries:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Code:**

package com.module4;

import static org.mockito.Mockito.\*;

import java.util.List;

import org.junit.jupiter.api.Test;

public class MockitoMockTest {

@Test

void testMocking() {

List<String> mockedList = *mock*(List.class);

*when*(mockedList.get(0)).thenReturn("Hello Mockito");

System.*out*.println(mockedList.get(0));

System.*out*.println(mockedList.get(999));

}}

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 2: Verifying Interactions**

**Code:**

package com.module4;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import java.util.List;

public class MockitoVerifyTest {

@Test

void testVerifyInteraction() {

List<String> mockedList = *mock*(List.class);

mockedList.add("first");

mockedList.add("second");

mockedList.clear();

*verify*(mockedList).add("first");

*verify*(mockedList).add("second");

*verify*(mockedList).clear();

}

}

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**