**Question-1:Control Structures**

**Solution:**

Sce1:

BEGIN

FOR cust\_rec IN (

SELECT cus\_id, age, loan\_id, interest\_rate

FROM customers

JOIN loans ON customers.cus\_id = loans.cus\_id

WHERE age > 60

)

LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 0.01

WHERE loan\_id = cust\_rec.loan\_id;

END LOOP;

COMMIT;

END;

Sce2:

BEGIN

FOR cust\_rec IN (

SELECT cus\_id, balance

FROM customers

WHERE balance > 10000

)

LOOP

UPDATE customers

SET isvip = 'TRUE'

WHERE cus\_id = cust\_rec.cus\_id;

END LOOP;

COMMIT;

END;

Sce3:

DECLARE

v\_due\_date loans.due\_date%TYPE;

v\_name customers.name%TYPE;

BEGIN

FOR loan\_rec IN (

SELECT c.name, l.due\_date

FROM loans l

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

WHERE l.due\_date <= SYSDATE + 30

)

LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || loan\_rec.name ||

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

END LOOP;

END;

**Output:**

Based on our data output is—

Reminder: Dear Alice, your loan is due on 15-JUL-2025.

Reminder: Dear Bob, your loan is due on 02-JUL-2025.

**Question-2: Stored Procedures**

**Solution:**

Sce1:

CREATE OR REPLACE PROCEDURE ProcessMonthly AS

BEGIN

FOR acc IN (

SELECT acco\_id, balance

FROM accounts

WHERE acco\_type = 'SAVINGS'

)

LOOP

UPDATE accounts

SET balance = balance + (balance \* 0.01)

WHERE acco\_id = acc.acco\_id;

END LOOP;

COMMIT;

END;

Sce2:

CREATE OR REPLACE PROCEDURE UpdateEmpBonus (

p\_dept\_id IN NUMBER,

p\_bonus\_pct IN NUMBER

) AS

BEGIN

UPDATE employees

SET salary = salary + (salary \* p\_bonus\_pct / 100)

WHERE department\_id = p\_dept\_id;

COMMIT;

END;

Sce3:

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

SELECT balance INTO v\_balance

FROM accounts

WHERE acco\_id = p\_from\_acco

FOR UPDATE;

IF v\_balance < p\_amount THEN

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

END IF;

UPDATE accounts

SET balance = balance - p\_amount

WHERE acco\_id = p\_from\_acco;

UPDATE accounts

SET balance = balance + p\_amount

WHERE acco\_id = p\_to\_acco;

COMMIT;

END;

**Question-3: Setting Up Junit**

**Solution:**

Step-1:

You can do this using an IDE like IntelliJ IDEA or Eclipse:

Open your IDE

Create a new Maven Project (or Java Project)

Choose a name like JUnitExample

Step-2:Junit Dependency pom.xml:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

Step-3:

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

int result = calc.add(2, 3);

assertEquals(5, result);

}

}

**Output:**

Running com.example.CalculatorTest

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

BUILD SUCCESS

**Question-4: Assertions in Junit**

**Solution:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionTest {

public void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 3);

assertFalse(5 < 3);

assertNull(null);

assertNotNull(new Object());

}

}

**Output:**

Maven

If all assertions pass:

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

If any fail:

java.lang.AssertionError: expected:<5> but was:<4>

**Question-5: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in Junit**

**Solution:**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int divide(int a, int b) {

return a / b;

}

}

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.");

}

public void testAdd() {

int a = 5, b = 3;

int result = calculator.add(a, b);

assertEquals(8, result);

}

public void testDivide() {

int a = 10, b = 2;

int result = calculator.divide(a, b);

assertEquals(5, result);

}

public void testDivideByZero() {

int a = 10, b = 0;

calculator.divide(a, b);

}

}

**Output:**

Setup: Calculator initialized.

Teardown: Calculator cleaned up.

Setup: Calculator initialized.

Teardown: Calculator cleaned up.

Setup: Calculator initialized.

Teardown: Calculator cleaned up.

**Question-6: Mocking and Stubbing**

**Solution:**

public interface ExternApi{

String getData();

}

public class MyService {

private ExternApi api;

public MyService(ExternApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

public void testExternApi() {

ExternApi mockApi = Mockito.mock(ExternApi.class);

when(mockApi.getData()).thenReturn("Mock Data");

MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Output:**

All tests passes:

Tests run: 1, Failures: 0

**Question-7: Verifying Interactions**

**Solution:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

public void testVerifyInteraction() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

}

}

**Output:**

Tests run: 1, Failures: 0

**Question-8: Logging Error Messages and Warning Levels**

**Solution:**

XML:

<dependencies>

<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>

</dependencies>

Main code:

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");

}

}

**Output:**

20xx-xx-xx xx:xx:xx ERROR LoggingExample - This is an error message

20xx-xx-xx xx:xx:xx WARN LoggingExample - This is a warning message