WEEK -2 ASSIGNMENT (DIGITAL-NURTURE-4.0 JAVA FSE)  
Module 3: PL/SQL programming  
Module 4: C: JUnit, Mockito and SL4J  
  
  
**NAME: JEYAKUMAR R**

**SUPERSET ID: 6396631  
  
Exercise 1: Control Structures**

Scenario 1: Apply 1% Interest Discount for Senior Citizens (Age > 60)

DECLARE

CURSOR cur\_senior\_customers IS

SELECT customer\_id, loan\_id, interest\_rate

FROM Customers C

JOIN Loans L ON C.customer\_id = L.customer\_id

WHERE C.age > 60;

BEGIN

FOR rec IN cur\_senior\_customers LOOP

UPDATE Loans

SET interest\_rate = interest\_rate - 1

WHERE loan\_id = rec.loan\_id;

END LOOP;

COMMIT;

END;

/

**Scenario 2: Promote Customers to VIP Status (Balance > $10,000)**

BEGIN

FOR rec IN (SELECT customer\_id FROM Customers WHERE balance > 10000) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE customer\_id = rec.customer\_id;

END LOOP;

COMMIT;

END;

/

Scenario 3: Send Loan Due Reminders (Next 30 Days)  
DECLARE

CURSOR cur\_due\_loans IS

SELECT C.customer\_id, C.name, L.loan\_id, 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;

BEGIN

FOR rec IN cur\_due\_loans LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Customer ' || rec.name ||

' (ID: ' || rec.customer\_id ||

') has a loan (ID: ' || rec.loan\_id ||

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

END LOOP;

END;

/**Exercise 2: Stored Procedures**

Scenario 1: Process Monthly Interest for Savings Accounts

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_type = 'Savings';

COMMIT;

END;

/

Scenario 2: Update Employee Bonus Based on Department and Bonus %

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department\_id IN NUMBER,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE department\_id = p\_department\_id;

COMMIT;

END;

/

Scenario 3: Transfer Funds Between Customer Accounts

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_source\_account\_id IN NUMBER,

p\_target\_account\_id IN NUMBER,

p\_amount IN NUMBER

) IS

v\_source\_balance NUMBER;

BEGIN

-- Check source account balance

SELECT balance INTO v\_source\_balance

FROM Accounts

WHERE account\_id = p\_source\_account\_id

FOR UPDATE;

IF v\_source\_balance < p\_amount THEN

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

END IF;

-- Deduct from source

UPDATE Accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_source\_account\_id;

-- Add to target

UPDATE Accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_target\_account\_id;

COMMIT;

END;

/

**Exercise 3: Setting Up JUnit**

CODE:

**Calculator.java:**

package example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

**CalculatorTest.java:**

package 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(5,3);

assertEquals(8, result);

}

}

**Pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>settingupJUnit</artifactId>

<version>1.0-SNAPSHOT</version>

<name>settingupJUnit</name>

<description>A project for setting up JUnit</description>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0</version>

</plugin>

</plugins>

</build>

</project>

**Exercise 4: Assertions in JUnit**

CODE:

**AssertionsTest.java:**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

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());

}

}

**Pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>JUnitDemo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 4 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

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

CODE:

**CalculatorFixtureTest.java:**

<project xmlns=”http://maven.apache.org/POM/4.0.0”

xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”

xsi:schemaLocation=”http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd”>

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>JunitDemo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!—JUnit 4

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**Exercise 6: Mocking and Stubbing**

CODE:

**ExternalApi:**

package com.example;

public interface ExternalApi {

String getData();

}

**MyService:**

package com.example;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest**:

package com.example;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import static org.mockito.Mockito.when;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class MyServiceTest {

@Test

public void testExternalApi() {

// Step 1: Create mock object

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

// Step 2: Stub method

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

// Step 3: Inject mock into service

MyService service = new MyService(mockApi);

// Step 4: Assert result

String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>MockingandStubbing</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.9.1</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.8.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Ensure JUnit 5 runs correctly -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

<configuration>

<useModulePath>false</useModulePath>

</configuration>

</plugin>

<!-- Set compiler to Java 17 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>17</source>

<target>17</target>

</configuration>

</plugin>

</plugins>

</build>

</project>  
  
**Exercise 7: Verifying Interactions**

CODE:

**ExternalApi:**

package com.example;

public interface ExternalApi {

void getData();

}

MyService.java:

package com.example;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public void fetchData() {

// Call the API method

api.getData();

}

}

**MyServiceTest.java:**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

// Step 1: Create a mock object

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Inject mock into service

MyService service = new MyService(mockApi);

// Step 3: Call method that uses the mock

service.fetchData();

// Step 4: Verify interaction

verify(mockApi).getData(); // Verifies it was called once

}

}

**Pom.xml:**

<project>

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>MockingAndStubbing</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Set Java version -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>17</source>

<target>17</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Exercise 8: Logging Error Messages and Warning Levels**

CODE:

**LoggingExample.java:**

package com.example;

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");

logger.info("This is an info message");

logger.debug("This is a debug message");

logger.trace("This is a trace message");

}

}

**Pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>SLF4JLoggingExample</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- SLF4J API -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<!-- Logback Classic (SLF4J binding) -->

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

</project>

**Logback.xml:**

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<root level="debug">

<appender-ref ref="STDOUT"/>

</root>

</configuration>