**PL/SQL PROBLEMS**

**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.

**CODE**

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE loans';

  EXECUTE IMMEDIATE 'DROP TABLE customers';

EXCEPTION

  WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE customers (

  cust\_id NUMBER PRIMARY KEY,

  name VARCHAR2(50),

  age NUMBER,

  balance NUMBER,

  interest NUMBER,

  is\_vip VARCHAR2(5)

);

INSERT INTO customers VALUES (1, 'Arun', 65, 15000, 10, 'FALSE');

INSERT INTO customers VALUES (2, 'Priya', 58, 9000, 12, 'FALSE');

INSERT INTO customers VALUES (3, 'Chandra', 70, 11000, 11, 'FALSE');

INSERT INTO customers VALUES (4, 'Vignesh', 45, 5000, 13, 'FALSE');

INSERT INTO customers VALUES (5, 'Latha', 61, 10500, 9, 'FALSE');

COMMIT;

CREATE TABLE loans (

  loan\_id NUMBER PRIMARY KEY,

  cust\_id NUMBER,

  due\_date DATE

);

INSERT INTO loans VALUES (101, 1, SYSDATE + 10);

INSERT INTO loans VALUES (102, 2, SYSDATE + 40);

INSERT INTO loans VALUES (103, 3, SYSDATE + 5);

INSERT INTO loans VALUES (104, 4, SYSDATE + 25);

INSERT INTO loans VALUES (105, 5, SYSDATE + 90);

COMMIT;

BEGIN

  FOR i IN (SELECT cust\_id FROM customers WHERE age > 60) LOOP

    UPDATE customers

    SET interest = interest - 1

    WHERE cust\_id = i.cust\_id;

  END LOOP;

  COMMIT;

END;

/

SELECT cust\_id, name, age, interest FROM customers;

BEGIN

  FOR i IN (SELECT cust\_id FROM customers WHERE balance > 10000) LOOP

    UPDATE customers

    SET is\_vip = 'TRUE'

    WHERE cust\_id = i.cust\_id;

  END LOOP;

  COMMIT;

END;

/

SELECT cust\_id, name, balance, is\_vip FROM customers;

BEGIN

  FOR i IN (

    SELECT cust\_id, due\_date

    FROM loans

    WHERE due\_date <= SYSDATE + 30

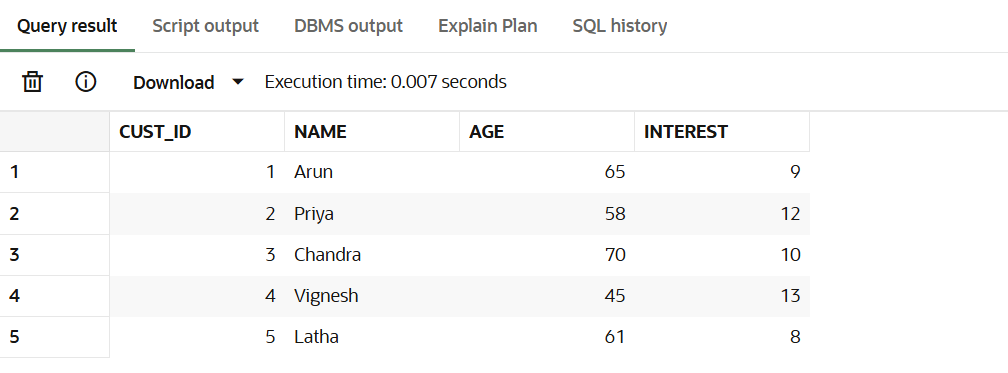
  ) LOOP

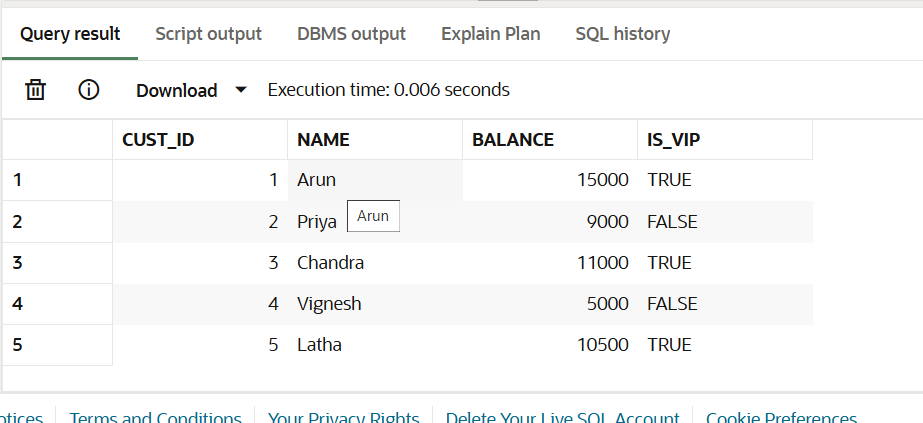
    DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan for customer ' || i.cust\_id || ' is due on ' || i.due\_date);

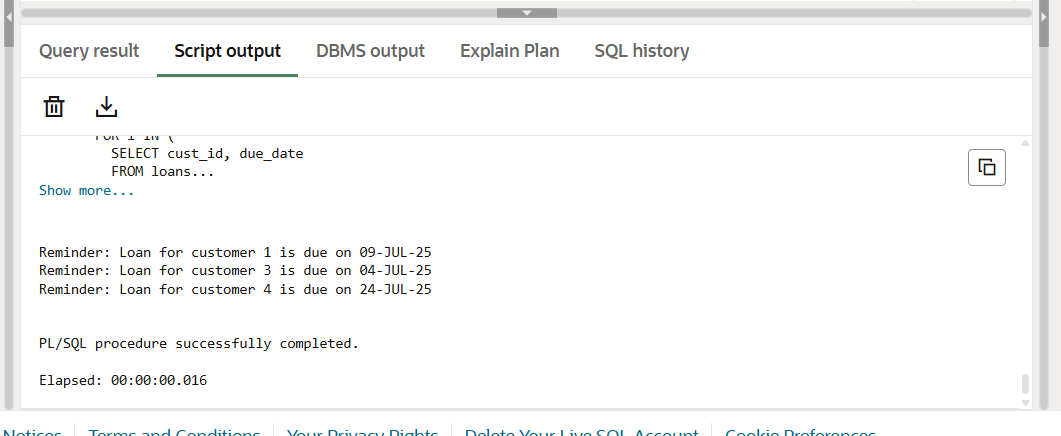
  END LOOP;

END;

/

**OUTPUT** 

****



**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.

**CODE**

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE accounts';

  EXECUTE IMMEDIATE 'DROP TABLE employees';

EXCEPTION

  WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE accounts (

  acc\_id NUMBER PRIMARY KEY,

  name VARCHAR2(50),

  type VARCHAR2(20),

  balance NUMBER

);

INSERT INTO accounts VALUES (1, 'Arun', 'savings', 10000);

INSERT INTO accounts VALUES (2, 'Kavitha', 'savings', 20000);

INSERT INTO accounts VALUES (3, 'Ravi', 'current', 15000);

INSERT INTO accounts VALUES (4, 'Meena', 'savings', 5000);

INSERT INTO accounts VALUES (5, 'Vignesh', 'current', 25000);

COMMIT;

CREATE TABLE employees (

  emp\_id NUMBER PRIMARY KEY,

  name VARCHAR2(50),

  dept VARCHAR2(20),

  salary NUMBER

);

INSERT INTO employees VALUES (101, 'Kumar', 'HR', 30000);

INSERT INTO employees VALUES (102, 'Devi', 'Sales', 35000);

INSERT INTO employees VALUES (103, 'Sundar', 'HR', 32000);

INSERT INTO employees VALUES (104, 'Renu', 'IT', 40000);

INSERT INTO employees VALUES (105, 'Raja', 'Sales', 37000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

  FOR i IN (SELECT acc\_id FROM accounts WHERE type = 'savings') LOOP

    UPDATE accounts

    SET balance = balance + (balance \* 0.01)

    WHERE acc\_id = i.acc\_id;

  END LOOP;

  COMMIT;

END;

/

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(p\_dept VARCHAR2, p\_bonus NUMBER) IS

BEGIN

  FOR i IN (SELECT emp\_id FROM employees WHERE dept = p\_dept) LOOP

    UPDATE employees

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

    WHERE emp\_id = i.emp\_id;

  END LOOP;

  COMMIT;

END;

/

CREATE OR REPLACE PROCEDURE TransferFunds(p\_from NUMBER, p\_to NUMBER, p\_amount NUMBER) IS

  bal NUMBER;

BEGIN

  SELECT balance INTO bal FROM accounts WHERE acc\_id = p\_from;

  IF bal >= p\_amount THEN

    UPDATE accounts SET balance = balance - p\_amount WHERE acc\_id = p\_from;

    UPDATE accounts SET balance = balance + p\_amount WHERE acc\_id = p\_to;

    COMMIT;

  END IF;

END;

/

BEGIN

  ProcessMonthlyInterest;

END;

/

BEGIN

  UpdateEmployeeBonus('Sales', 10);

END;

/

BEGIN

  TransferFunds(1, 2, 3000);

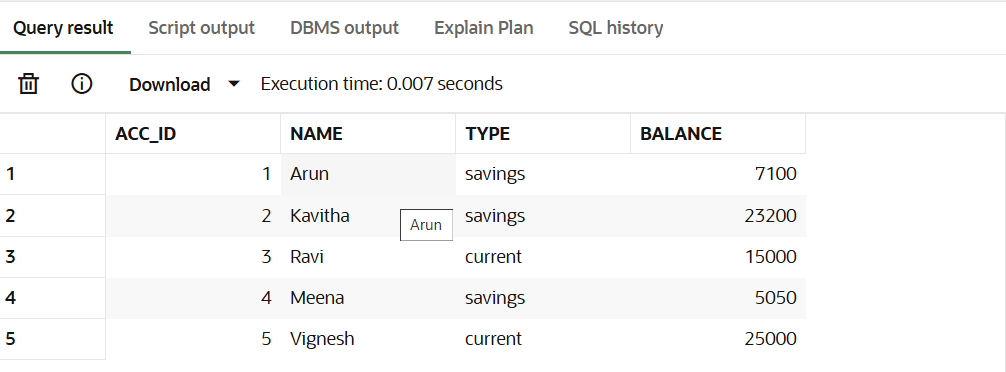
END;

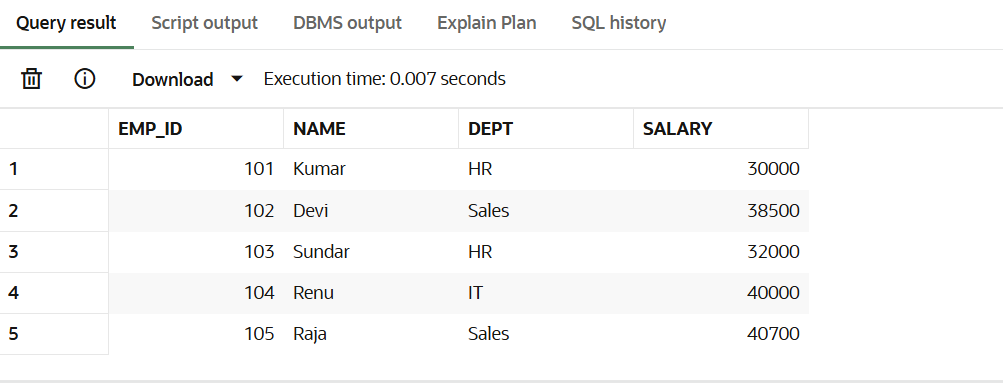
/

SELECT \* FROM accounts;

SELECT \* FROM employees;

**OUTPUT**

****

****

**JUNIT BASIC TESTING EXERCISES**

**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. If you are using Maven, add the following to your

pom.xml:

<dependency>

<groupId>junit</groupld>

<artifactld>junit</artifactld>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

1. Create a new test class in your project.

**CODE**

**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>JunitExample</artifactId>

<version>1.0</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorTest.java**

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator c = new Calculator();

int result = c.add(5, 3);

assertEquals(8, result);

}

}

**Output**

PS C:\Users\prithivi\Desktop\MyJUnitProject> mvn test

[INFO] Scanning for projects…

[INFO]

[INFO] ----------------< com.example:MyJUnitProject >----------------

[INFO] Building MyJUnitProject 1.0-SNAPSHOT

[INFO] --------------------------------[ jar ]-------------------------------

[INFO]

[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ MyJUnitProject ---

[INFO] Using ‘UTF-8’ encoding to copy filtered resources.

[INFO] skip non existing resourceDirectory C:\Users\prithivi\Desktop\MyJUnitProject\src\main\resources

[INFO]

[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ MyJUnitProject ---

[INFO] Changes detected – recompiling the module!

[INFO] Compiling 1 source file to C:\Users\prithivi\Desktop\MyJUnitProject\target\classes

[INFO]

[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ MyJUnitProject ---

[INFO] Using ‘UTF-8’ encoding to copy filtered resources.

[INFO] skip non existing resourceDirectory C:\Users\prithivi\Desktop\MyJUnitProject\src\test\resources

[INFO]

[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ MyJUnitProject ---

[INFO] Changes detected – recompiling the module!

[INFO] Compiling 1 source file to C:\Users\prithivi\Desktop\MyJUnitProject\target\test-classes

[INFO]

[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ MyJUnitProject ---

T E S T S

Running CalculatorTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 sec

Results :

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

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 2.354 s

[INFO] Finished at: 2025-06-29T19:31:07+05:30

[INFO] Final Memory: 15M/55M

[INFO] ------------------------------------------------------------------------

**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()):

**CODE**

**AssertionsTest.java**

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 3);

assertFalse(5 < 3);

assertNull(null);

assertNotNull(new Object());

}

}

**output:**

PS C:\Users\prithivi\Desktop\MyJUnitProject> mvn test

[INFO] Scanning for projects…

[INFO]

[INFO] -------------------< com.example:MyJUnitProject >-------------------

[INFO] Building MyJUnitProject 1.0-SNAPSHOT

[INFO] --------------------------------[ jar ]-----------------------------

[INFO]

[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ MyJUnitProject ---

[INFO] Using ‘UTF-8’ encoding to copy filtered resources.

[INFO] skip non existing resourceDirectory C:\Users\prithivi\Desktop\MyJUnitProject\src\main\resources

[INFO]

[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ MyJUnitProject ---

[INFO] Changes detected – recompiling the module!

[INFO] Compiling 0 source files to C:\Users\prithivi\Desktop\MyJUnitProject\target\classes

[INFO]

[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ MyJUnitProject ---

[INFO] Using ‘UTF-8’ encoding to copy filtered resources.

[INFO] skip non existing resourceDirectory C:\Users\prithivi\Desktop\MyJUnitProject\src\test\resources

[INFO]

[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ MyJUnitProject ---

[INFO] Changes detected – recompiling the module!

[INFO] Compiling 1 source file to C:\Users\prithivi\Desktop\MyJUnitProject\target\test-classes

[INFO]

[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ MyJUnitProject ---

T E S T S

Running AssertionsTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.006 sec

Results :

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

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 2.201 s

[INFO] Finished at: 2025-06-29T21:04:32+05:30

[INFO] Final Memory: 13M/55M

[INFO] ------------------------------------------------------------------------

**MOCKITO EXERCISES**

**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.

1. Use @Before and @After annotations for setup and teardown methods.

**Code**

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}

}

**CalculatorTest.java**

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

Calculator calc;

@Before

public void setUp() {

calc = new Calculator();

System.out.println("Setup complete");

}

@After

public void tearDown() {

System.out.println("Teardown complete");

}

@Test

public void testAddition() {

int a = 5;

int b = 3;

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

assertEquals(8, result);

}

@Test

public void testMultiplication() {

int x = 4;

int y = 6;

int result = calc.multiply(x, y);

assertEquals(24, result);

}

}

**Output**

PS C:\Users\prithivi\Desktop\MyJUnitProject> mvn test

[INFO] Scanning for projects...

[INFO]

[INFO] -------------------< com.example:MyJUnitProject >--------------------

[INFO] Building MyJUnitProject 1.0-SNAPSHOT

[INFO] --------------------------------[ jar ]---------------------------------

[INFO]

[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ MyJUnitProject ---

[INFO] Using 'UTF-8' encoding to copy filtered resources.

[INFO] skip non existing resourceDirectory C:\Users\prithivi\Desktop\MyJUnitProject\src\main\resources

[INFO]

[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ MyJUnitProject ---

[INFO] Changes detected - recompiling the module!

[INFO] Compiling 1 source file to C:\Users\prithivi\Desktop\MyJUnitProject\target\classes

[INFO]

[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ MyJUnitProject ---

[INFO] Using 'UTF-8' encoding to copy filtered resources.

[INFO] skip non existing resourceDirectory C:\Users\prithivi\Desktop\MyJUnitProject\src\test\resources

[INFO]

[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ MyJUnitProject ---

[INFO] Changes detected - recompiling the module!

[INFO] Compiling 1 source file to C:\Users\prithivi\Desktop\MyJUnitProject\target\test-classes

[INFO]

[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ MyJUnitProject ---

-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running CalculatorTest

Setup complete

Teardown complete

Setup complete

Teardown complete

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 sec

Results :

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 1.428 s

[INFO] Finished at: 2025-06-29T18:45:53+05:30

[INFO] ------------------------------------------------------------------------

**Exercise 1: Mocking and Stubbing**

Scenario:

You need to test a service that depends on an external API. Use Moc

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);

}

}

**CODE**

**ExternalApi.java**

public interface ExternalApi {

String getData();

}

**MyService.java**

public class MyService {

ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

import org.junit.jupiter.api.Test;

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

import static org.mockito.Mockito.\*;

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();

assertEquals("Mock Data", result);

}

}

**Pom.xml**

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

**Output**

-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running MyServiceTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 sec

Results :

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

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 1.234 s

[INFO] Finished at: 2025-06-29T16:05:47+05:30

[INFO] ------------------------------------------------------------------------

**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();

}

**CODE**

public interface ExternalApi {

String getData();

}

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

}

}

**Pom.xml**

<dependencies>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.12.0</version>

<scope>test</scope>

</dependency>

</dependencies>

**Output**

-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running MyServiceTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.010 sec

Results :

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

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 1.154 s

[INFO] Finished at: 2025-06-29T17:33:10+05:30

[INFO] ------------------------------------------------------------------------

**SL4J LOGGING EXERCISES**

**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>

<groupld>org.slf4j</groupld>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupld>ch.qos.logback</groupld>

<artifactld>logback-classic</artifactld>

<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");

}

**CODE**

**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>LoggingDemo</artifactId>

<version>1.0-SNAPSHOT</version>

<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>

</project>

**LoggingExample.java**

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**

12:10:22.013 [main] ERROR LoggingExample - This is an error message

12:10:22.016 [main] WARN LoggingExample - This is a warning message