**PL/SQL**

**Exercise 1: Control Structures**

**1.Create Tables**

CREATE TABLE customers (

customer\_id NUMBER,

name VARCHAR2(100),

age NUMBER,

balance NUMBER,

is\_vip VARCHAR2(5) DEFAULT 'FALSE'

);

CREATE TABLE loans (

loan\_id NUMBER,

customer\_id NUMBER,

interest\_rate NUMBER,

due\_date DATE

);

**2.Insert Sample Data**

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

INSERT INTO customers VALUES (2, 'Bob', 45, 8000, 'FALSE');

INSERT INTO loans VALUES (101, 1, 8.5, SYSDATE + 20); -- Alice's loan, due soon

INSERT INTO loans VALUES (102, 2, 9.0, SYSDATE + 40); -- Bob's loan, due later

COMMIT;

**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.

**Code:**

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 - (interest\_rate \* 0.01)

WHERE loan\_id = cust.loan\_id;

END LOOP;

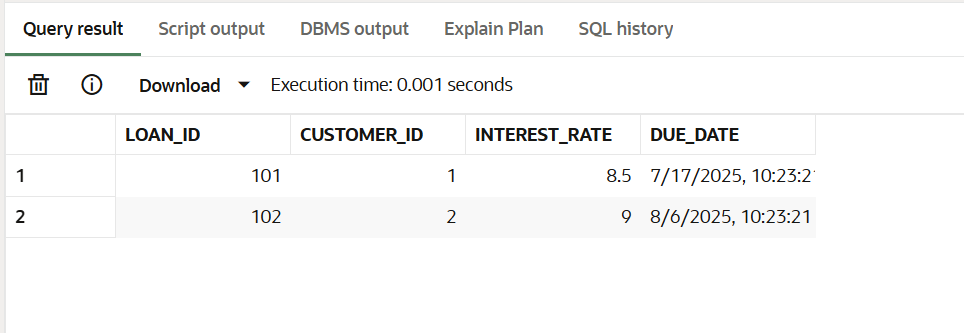
COMMIT;

END;

**Check output:**

SELECT \* FROM loans;

**Output:**

****

**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.

**Code:**

BEGIN

FOR cust IN (

SELECT customer\_id

FROM customers

WHERE balance > 10000

)

LOOP

UPDATE customers

SET is\_vip = 'TRUE'

WHERE customer\_id = cust.customer\_id;

END LOOP;

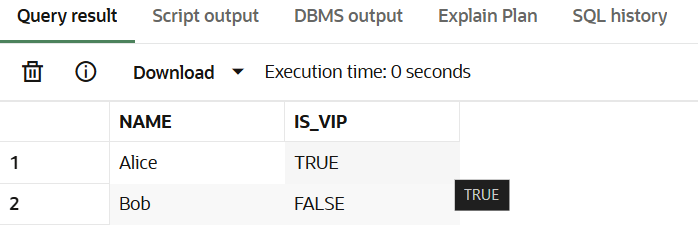
COMMIT;

END;

**Check output:**

SELECT name, is\_vip FROM customers;

**Output:**

****

**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

FOR l IN (

SELECT c.name, l.loan\_id, l.due\_date

FROM loans l

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

WHERE l.due\_date <= SYSDATE + 30

)

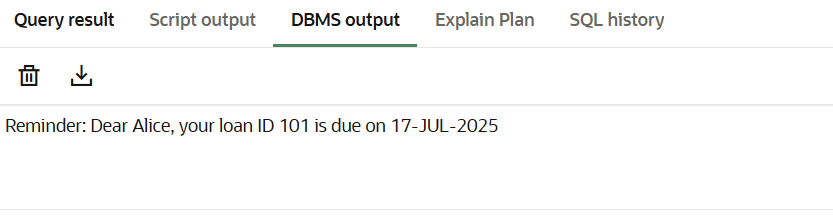
LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || l.name || ', your loan ID ' || l.loan\_id ||

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

END LOOP;

END;  
**Output:**



**Exercise 3: Stored Procedures**

**1.Setup Tables and Sample Data**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

AccountType VARCHAR2(20),

Balance NUMBER(10,2)

);

INSERT INTO Accounts VALUES (1, 'Savings', 10000);

INSERT INTO Accounts VALUES (2, 'Savings', 20000);

INSERT INTO Accounts VALUES (3, 'Current', 15000);

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Department VARCHAR2(30),

Salary NUMBER(10,2)

);

INSERT INTO Employees VALUES (101, 'Alice', 'HR', 50000);

INSERT INTO Employees VALUES (102, 'Bob', 'IT', 60000);

INSERT INTO Employees VALUES (103, 'Carol', 'HR', 55000);

COMMIT;

**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.

**Code:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountType = 'Savings';

END;

**Execution Block:**

BEGIN

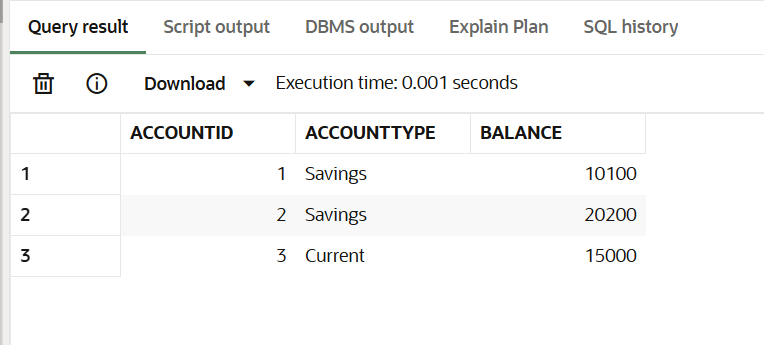
ProcessMonthlyInterest;

END;

**Check Output:**

SELECT \* FROM Accounts;

**Output:**

****

**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.

**Code:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept IN VARCHAR2,

bonus\_pct IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_pct / 100)

WHERE Department = dept;

END;

**Execution Block:**

BEGIN

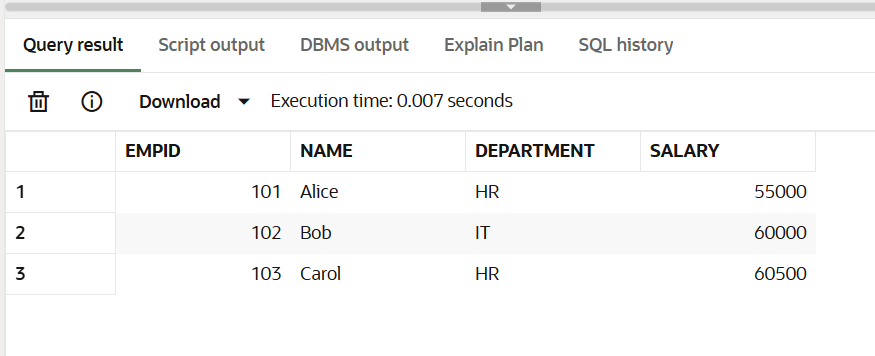
UpdateEmployeeBonus('HR', 10);

END;

**Check Output:**

SELECT \* FROM Employees;

**Output:**



**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:**

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account IN NUMBER,

to\_account IN NUMBER,

amount IN NUMBER

) IS

insufficient\_balance EXCEPTION;

BEGIN

DECLARE

bal NUMBER;

BEGIN

SELECT Balance INTO bal FROM Accounts WHERE AccountID = from\_account;

IF bal < amount THEN

RAISE insufficient\_balance;

END IF;

END;

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = from\_account;

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = to\_account;

EXCEPTION

WHEN insufficient\_balance THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds for transfer.');

END;

**Execution Block:**

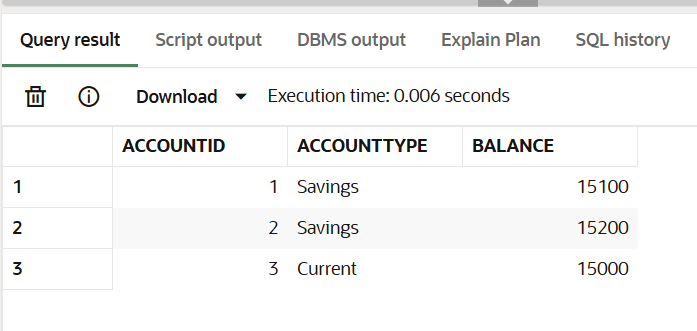
BEGIN

TransferFunds(2, 1, 5000);

END;

**Check Output:**

SELECT \* FROM Accounts;  
  
**Output:**



**JUnit Testing Exercises**

**Exercise 1: Setting Up Junit**

**Code:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class HelloWorldTest {

public void testMessage() {

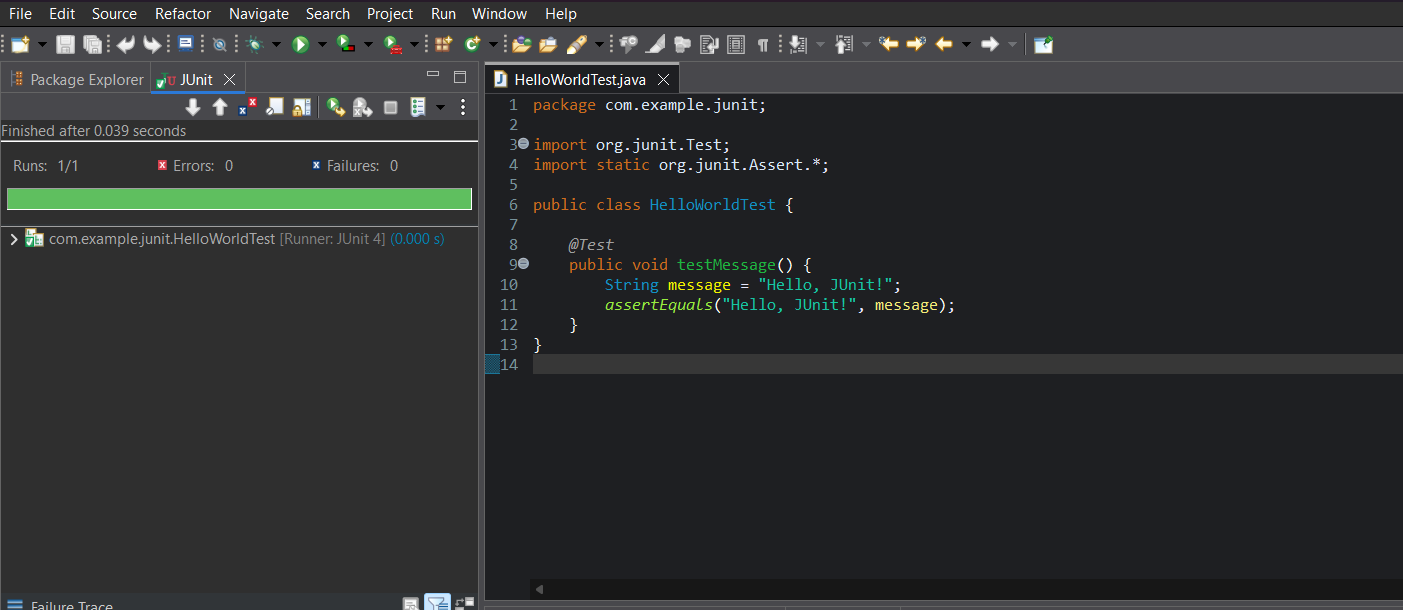
String message = "Hello, JUnit!";

assertEquals("Hello, JUnit!", message);

}

}

**Output:**

****

**Exercise 3: Assertions in Junit**

**Code:**

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

import org.junit.jupiter.api.Test;

public class AssertionsTest {

public void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 3);

assertFalse(5 < 3);

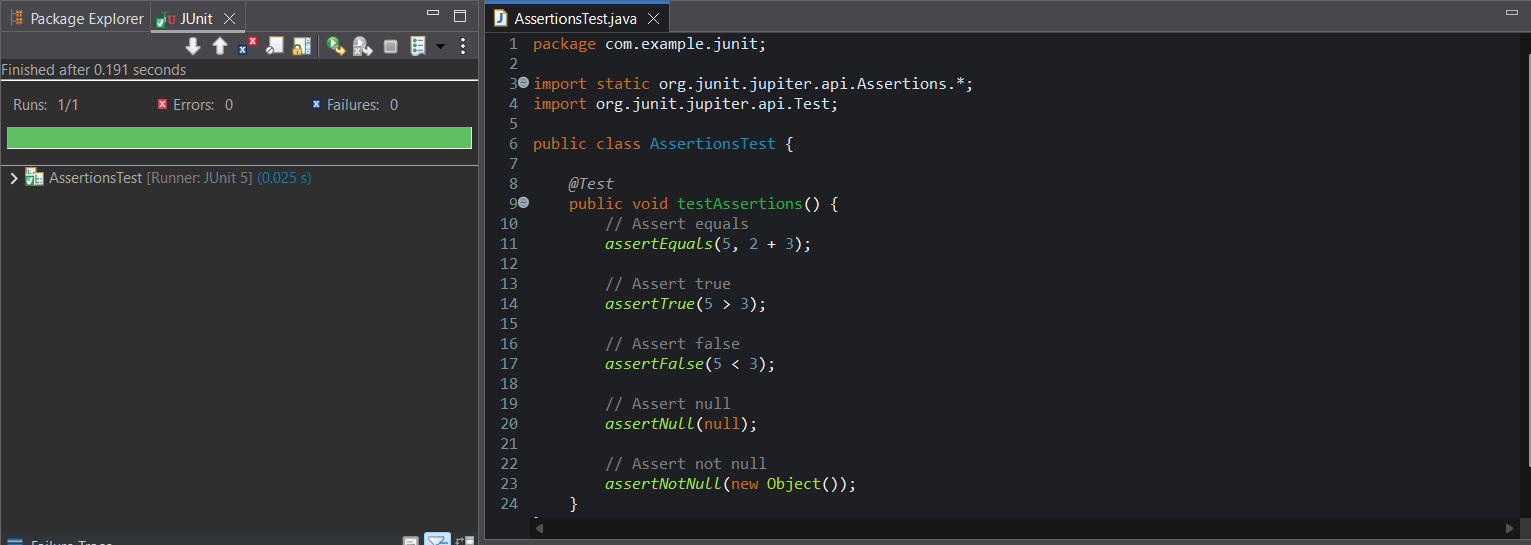
assertNull(null);

assertNotNull(new Object());

}

}

**Output:**

****

**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in Junit**

**Code:**

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

public void setUp() {

calculator = new Calculator();

System.out.println("Setup: Calculator instance created");

}

public void tearDown() {

calculator = null;

System.out.println("Teardown: Calculator instance cleared");

}

public void testAddition() {

int a = 5;

int b = 3;

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

assertEquals(8, result);

}

public void testSubtraction() {

int a = 10;

int b = 4;

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

assertEquals(6, result);

}

}

class Calculator {

public int add(int a, int b) {

return a + b;

}

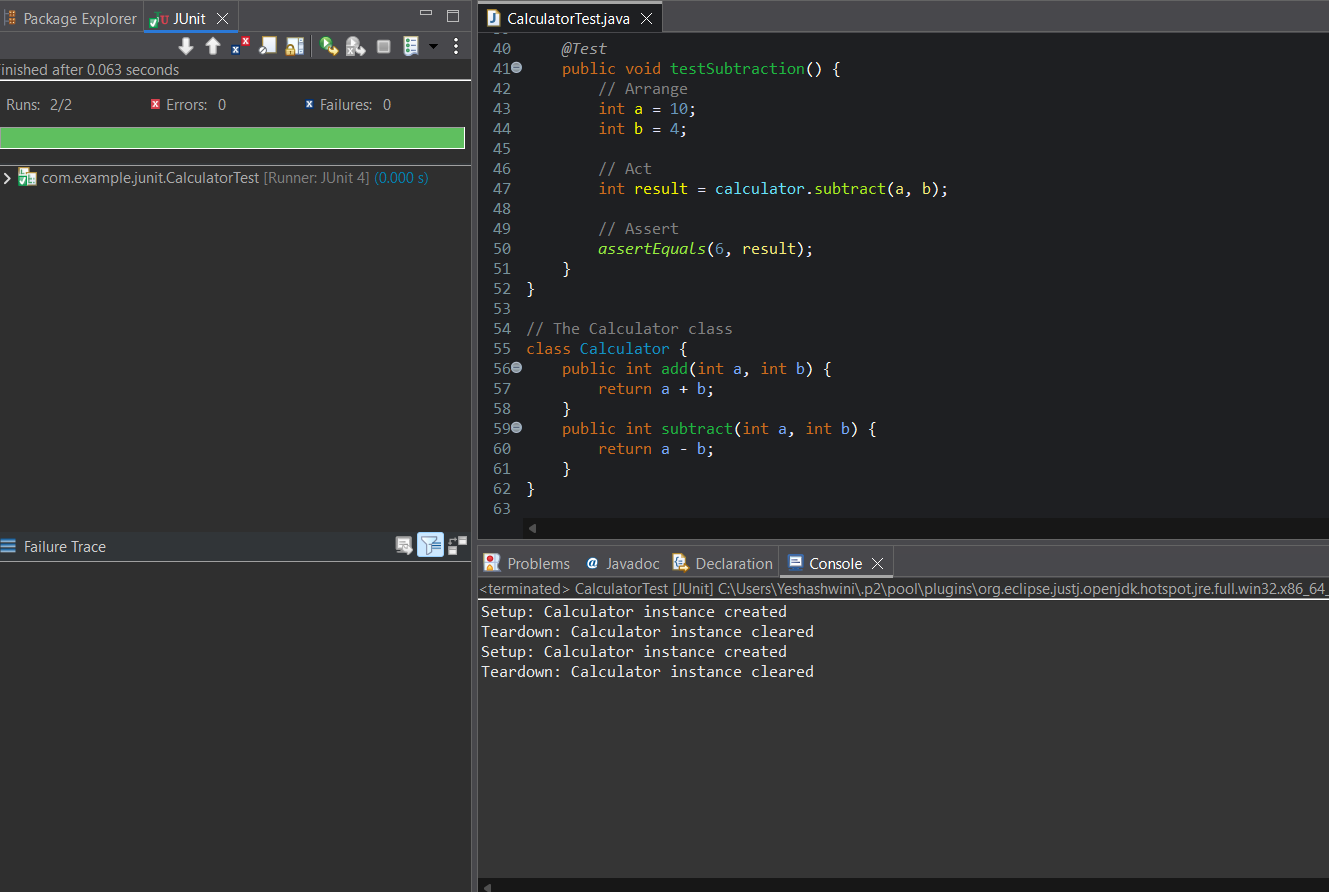
public int subtract(int a, int b) {

return a - b;

}

}

**Output:**

****

**Mockito Hands-On Exercises**

**Exercise 1: Mocking and Stubbing**

**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>mockito-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

<junit.version>5.10.0</junit.version>

<mockito.version>5.11.0</mockito.version>

</properties>

<dependencies>

<!-- JUnit 5 -->

<dependency>

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

<artifactId>junit-jupiter-api</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>junit-jupiter-engine</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>${mockito.version}</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Required to run JUnit 5 tests -->

<plugin>

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

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

<version>2.22.2</version>

<configuration>

<includes>

<include>\*\*/\*Test.java</include>

</includes>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Code:**

**1.ExternalApi class:**

public interface ExternalApi {

String getData();

}

**2.** **MyService class:**

public class MyService {

private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

**3.** **MyServiceTest class:**

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 testExternalApi() {

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

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

MyService service = new MyService(mockApi);

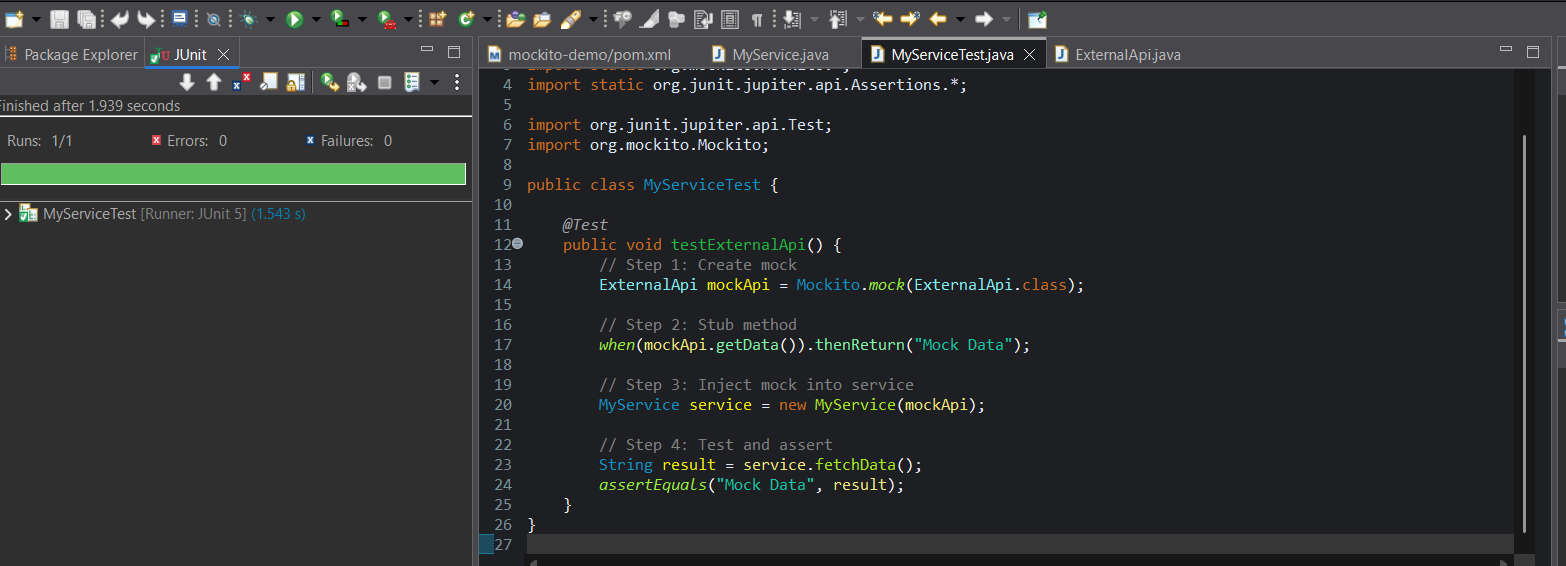
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Output:**

****

**Exercise 2: Verifying Interactions**

**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>mockito-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

<junit.version>5.10.0</junit.version>

<mockito.version>5.11.0</mockito.version>

</properties>

<dependencies>

<!-- JUnit 5 -->

<dependency>

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

<artifactId>junit-jupiter-api</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>junit-jupiter-engine</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>${mockito.version}</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>2.22.2</version>

<configuration>

<includes>

<include>\*\*/\*Test.java</include>

</includes>

</configuration>

</plugin>

</plugins>

</build>

</project>

\

**Code:**

**1.ExternalApi class:**

package com.example;

public interface ExternalApi {

String getData();

}

**2.** **MyService class:**

package com.example;

public class MyService {

private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

**3.** **MyServiceTest class:**

package com.example;

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);

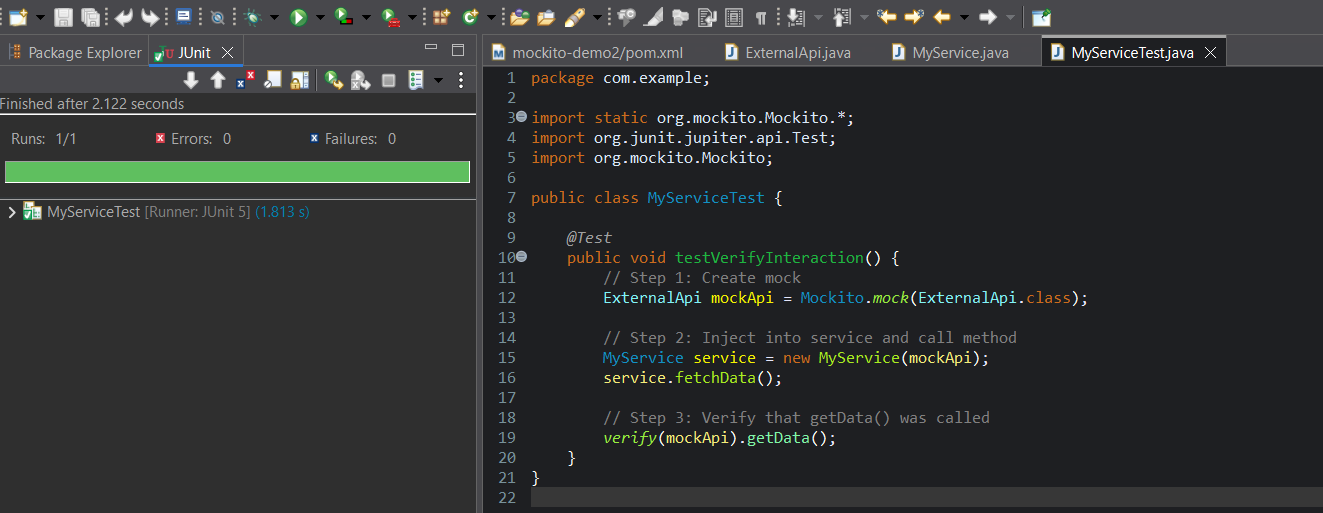
service.fetchData();

verify(mockApi).getData();

}

}

**Output:**

****

**Logging using SLF4J**

**Exercise 1: Logging Error Messages and Warning Levels**

**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");

}

}

**Required Maven Dependencies (in pom.xml):**

<dependencies>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

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

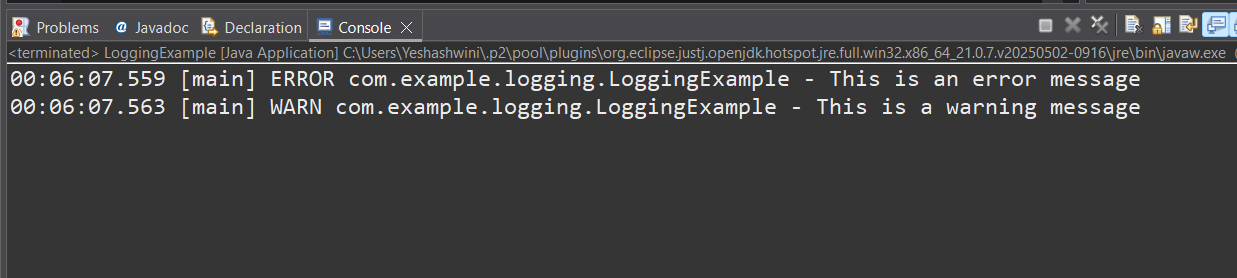
<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

**Output:**

****