# **Cognizant\_Week-2 HandsOn**

**Exercise 1: Control Structures**

**Code:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE,

IsVIP VARCHAR2(5)

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

-- Customers

INSERT INTO Customers VALUES (1, 'Ramesh', TO\_DATE('1950-03-10','YYYY-MM-DD'), 12000, SYSDATE, NULL);

INSERT INTO Customers VALUES (2, 'Mohan', TO\_DATE('1990-07-22','YYYY-MM-DD'), 5400, SYSDATE, NULL);

INSERT INTO Customers VALUES (3, 'Suresh', TO\_DATE('1962-11-02','YYYY-MM-DD'), 8900, SYSDATE, NULL);

-- Accounts

INSERT INTO Accounts VALUES (101, 1, 'Savings', 8000, SYSDATE);

INSERT INTO Accounts VALUES (102, 1, 'Checking', 4000, SYSDATE);

INSERT INTO Accounts VALUES (103, 2, 'Savings', 5400, SYSDATE);

INSERT INTO Accounts VALUES (104, 3, 'Checking', 8900, SYSDATE);

-- Transactions

INSERT INTO Transactions VALUES (1001, 101, TO\_DATE('2025-06-01','YYYY-MM-DD'), 500, 'Credit');

INSERT INTO Transactions VALUES (1002, 101, TO\_DATE('2025-06-05','YYYY-MM-DD'), 200, 'Debit');

INSERT INTO Transactions VALUES (1003, 102, TO\_DATE('2025-06-10','YYYY-MM-DD'), 1000, 'Credit');

INSERT INTO Transactions VALUES (1004, 103, TO\_DATE('2025-06-11','YYYY-MM-DD'), 300, 'Debit');

INSERT INTO Transactions VALUES (1005, 104, TO\_DATE('2025-06-15','YYYY-MM-DD'), 1500, 'Credit');

-- Loans

INSERT INTO Loans VALUES (201, 1, 25000, 5.5, TO\_DATE('2020-01-01','YYYY-MM-DD'), TO\_DATE('2025-07-15','YYYY-MM-DD'));

INSERT INTO Loans VALUES (202, 2, 15000, 6.0, TO\_DATE('2023-06-15','YYYY-MM-DD'), TO\_DATE('2028-06-15','YYYY-MM-DD'));

INSERT INTO Loans VALUES (203, 3, 30000, 4.5, TO\_DATE('2022-09-01','YYYY-MM-DD'), TO\_DATE('2025-07-10','YYYY-MM-DD'));

-- Employees

INSERT INTO Employees VALUES (501, 'Deepak', 'Manager', 75000, 'Finance', TO\_DATE('2015-05-20','YYYY-MM-DD'));

INSERT INTO Employees VALUES (502, 'Kavita', 'Clerk', 40000, 'Customer Service', TO\_DATE('2019-08-12','YYYY-MM-DD'));

INSERT INTO Employees VALUES (503, 'Amit', 'Loan Officer', 60000, 'Loans', TO\_DATE('2018-03-30','YYYY-MM-DD'));

INSERT INTO Employees VALUES (504, 'Anjali', 'Teller', 35000, 'Operations', TO\_DATE('2021-02-10','YYYY-MM-DD'));

**Scenario 1:**

BEGIN

FOR cust\_rec IN (

SELECT c.CustomerID, TRUNC(MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12) AS Age

FROM Customers c

) LOOP

IF cust\_rec.Age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust\_rec.CustomerID;

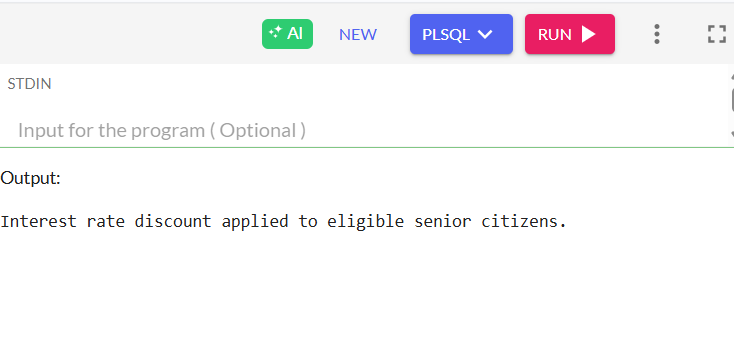
END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Interest rate discount applied to eligible senior citizens.');

END;

**Output:**



**Scenario 2:**

BEGIN

FOR vip\_rec IN (SELECT CustomerID, Balance FROM Customers) LOOP

IF vip\_rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = vip\_rec.CustomerID;

ELSE

UPDATE Customers

SET IsVIP = 'FALSE'

WHERE CustomerID = vip\_rec.CustomerID;

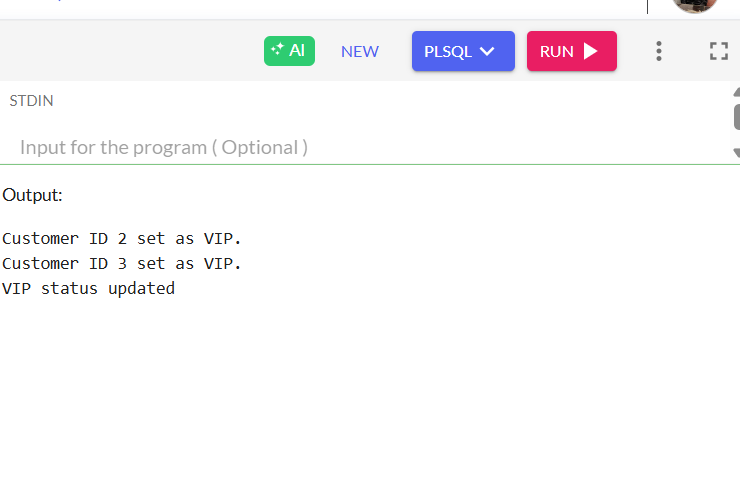
DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || vip\_rec.CustomerID || ' set as VIP.');

END IF;

END LOOP;

END;

**Output:**



**Scenario 3:**

BEGIN

FOR loan\_rec IN (

SELECT l.LoanID, l.CustomerID, c.Name, l.EndDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

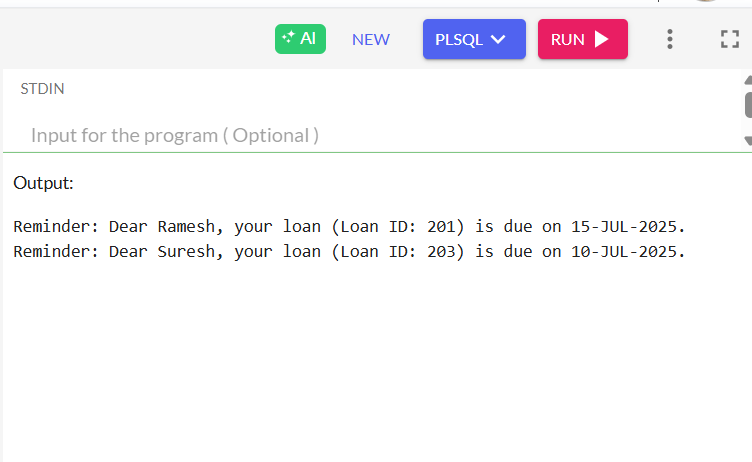
) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || loan\_rec.Name || ', your loan (Loan ID: ' || loan\_rec.LoanID || ') is due on ' || TO\_CHAR(loan\_rec.EndDate, 'DD-MON-YYYY') || '.');

END LOOP;

END;

**Output:**



**Exercise 3: Stored Procedures**

**Code:**

**Scenario 1:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

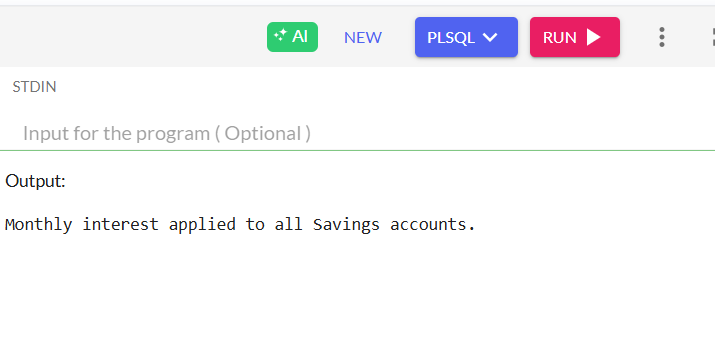
WHERE AccountType = 'Savings';

DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all Savings accounts.');

END;

EXEC ProcessMonthlyInterest;

**Output:**



**Scenario 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE Employees

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

WHERE Department = dept\_name;

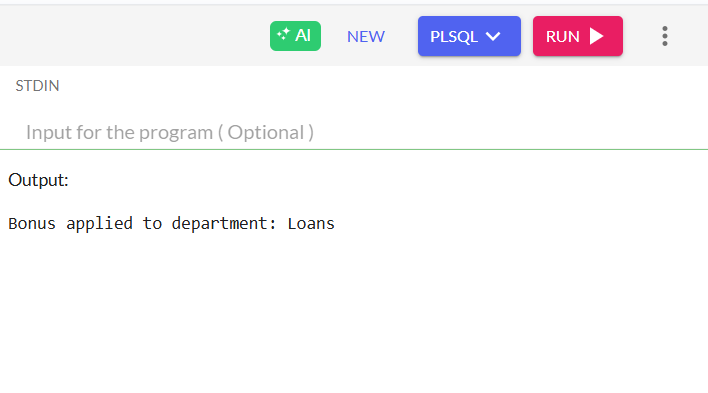
DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department: ' || dept\_name);

END;

/

EXEC UpdateEmployeeBonus('Loans', 10);

**Output:**



**Scenario 3:**

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

amount IN NUMBER

) AS

from\_balance NUMBER;

BEGIN

SELECT Balance INTO from\_balance

FROM Accounts

WHERE AccountID = from\_account\_id;

IF from\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: insufficient balance.');

RETURN;

END IF;

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = from\_account\_id;

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = to\_account\_id;

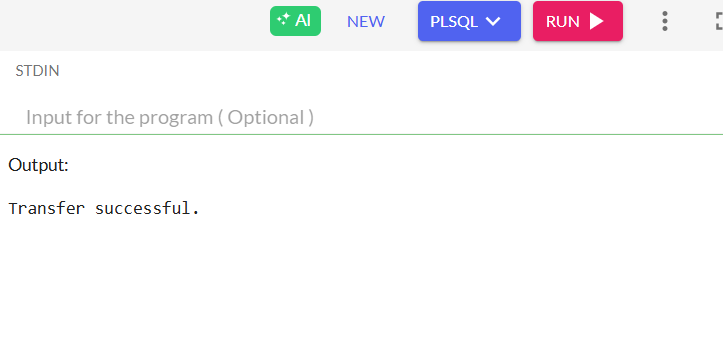
DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

END;

/

EXEC TransferFunds(101, 102, 500);

**Output:**



**Exercise 1: Setting Up Junit**

**Code:**

package com.example;  
  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class SampleTest {  
  
 @Test  
 public void testAddition() {  
 *assertEquals*(4, 2 + 2);  
 }  
}

**Pom.xml Code:**

<?xml version="1.0" encoding="UTF-8"?>  
<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>  
  
 <properties>  
 <maven.compiler.source>21</maven.compiler.source>  
 <maven.compiler.target>21</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <!-- JUnit 4.13.2 -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- Hamcrest for assertions -->  
 <dependency>  
 <groupId>org.hamcrest</groupId>  
 <artifactId>hamcrest</artifactId>  
 <version>2.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</project>

Output:



**Exercise 3: Assertions in Junit**

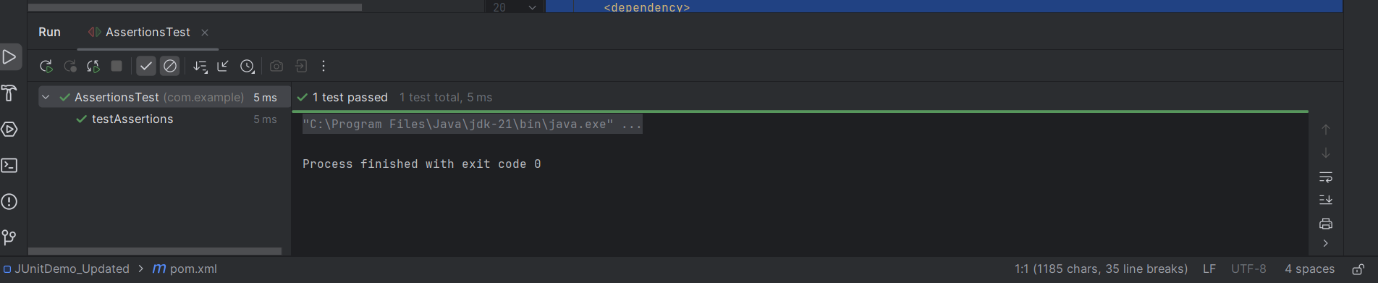
**Code:**

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 Code**:

<?xml version="1.0" encoding="UTF-8"?>  
<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>  
  
 <properties>  
 <maven.compiler.source>21</maven.compiler.source>  
 <maven.compiler.target>21</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <!-- JUnit 4.13.2 -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- Hamcrest for assertions -->  
 <dependency>  
 <groupId>org.hamcrest</groupId>  
 <artifactId>hamcrest</artifactId>  
 <version>2.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</project>

**Output:**



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

**Code:**

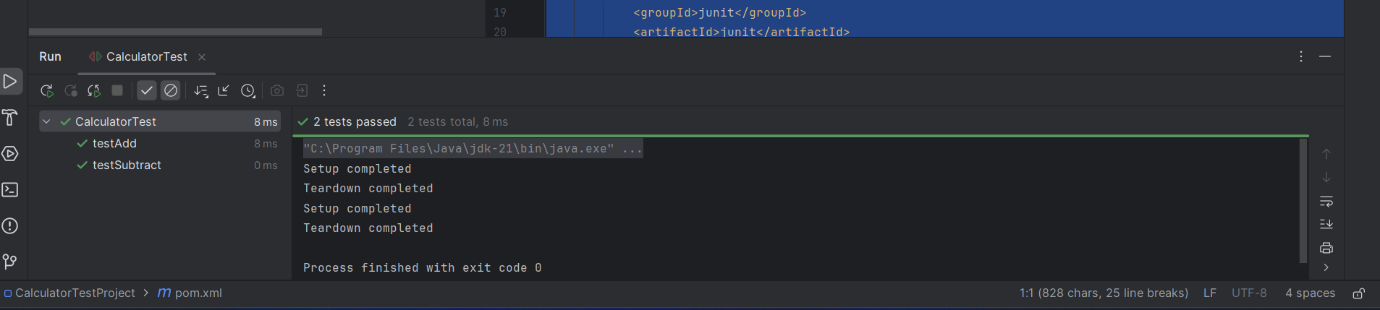
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
 public int subtract(int a, int b) {  
 return a - b;  
 }  
}

import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import static org.junit.Assert.*assertEquals*;  
  
public class CalculatorTest {  
  
 private Calculator calculator;  
  
 @Before  
 public void setUp() {  
 calculator = new Calculator();  
 System.*out*.println("Setup completed");  
 }  
  
 @After  
 public void tearDown() {  
 calculator = null;  
 System.*out*.println("Teardown completed");  
 }  
  
 @Test  
 public void testAdd() {  
 int a = 5;  
 int b = 3;  
 int result = calculator.add(a, b);  
 *assertEquals*(8, result);  
 }  
  
 @Test  
 public void testSubtract() {  
 int a = 10;  
 int b = 4;  
 int result = calculator.subtract(a, b);  
 *assertEquals*(6, result);  
 }  
}

**Pom.xml Code**:

<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>CalculatorTestProject</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
 <!-- JUnit 4 dependency -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</project>

**Output:**



**Exercise 1: Mocking and Stubbing**

**Code:**

public class MyService {  
 private final ExternalApi externalApi;  
  
 public MyService(ExternalApi externalApi) {  
 this.externalApi = externalApi;  
 }  
  
 public String fetchData() {  
 return externalApi.getData();  
 }  
}

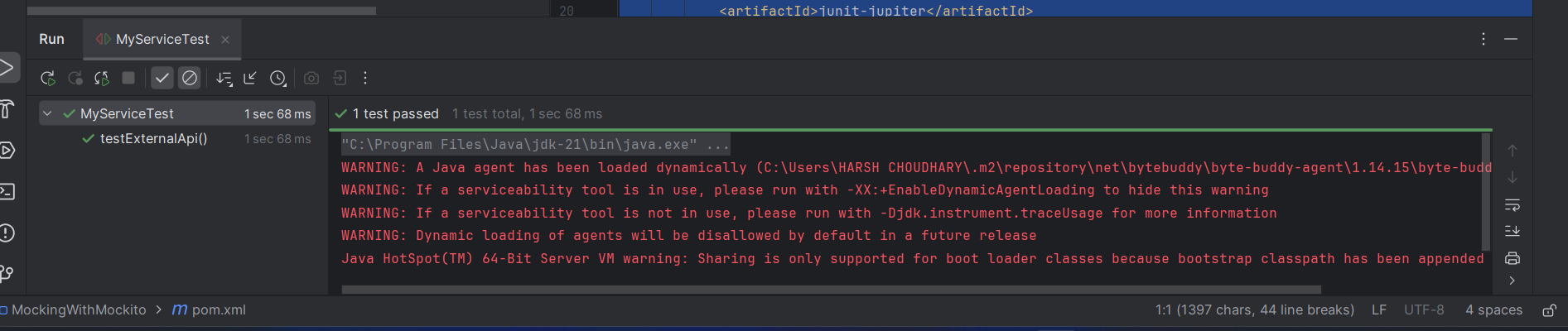
public interface ExternalApi {  
 String getData();  
}

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 {  
  
 @Test  
 public void testExternalApi() {  
 // Step 1: Create a mock object for ExternalApi  
 ExternalApi mockApi = Mockito.*mock*(ExternalApi.class);  
  
 // Step 2: Stub getData() to return a predefined value  
 *when*(mockApi.getData()).thenReturn("Mock Data");  
  
 // Step 3: Inject mock into service  
 MyService service = new MyService(mockApi);  
  
 // Act  
 String result = service.fetchData();  
  
 // Assert  
 *assertEquals*("Mock Data", result);  
 }  
}

**Pom.xml Code**:

<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>MockingWithMockito</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>  
  
 <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>5.12.0</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Maven Surefire Plugin for running tests -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>3.2.5</version>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Output:**



**Exercise 2: Verifying Interactions**

**Code:**

public class MyService {  
 private final ExternalApi externalApi;  
  
 public MyService(ExternalApi externalApi) {  
 this.externalApi = externalApi;  
 }  
  
 public String fetchData() {  
 return externalApi.getData();  
 }  
}

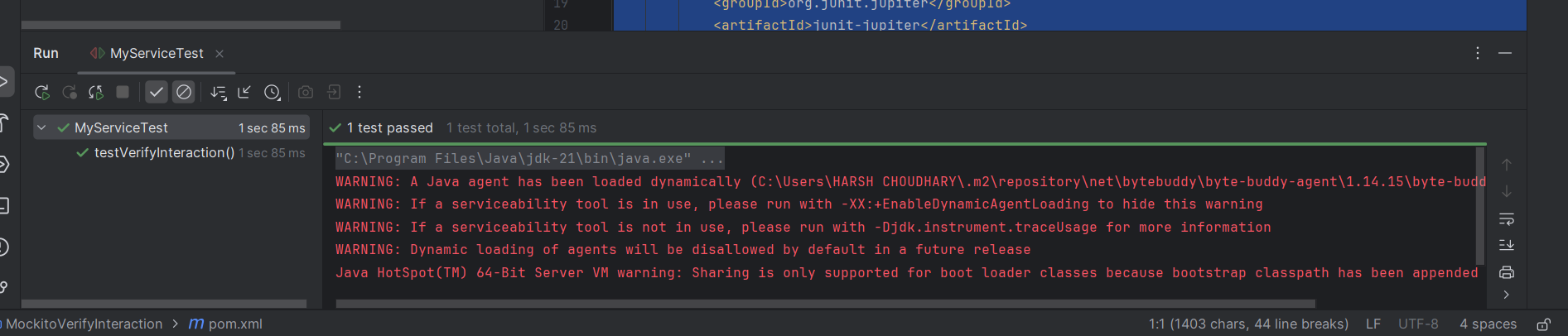
public interface ExternalApi {  
 String getData();  
}

import static org.mockito.Mockito.\*;  
import org.junit.jupiter.api.Test;  
import org.mockito.Mockito;  
  
public class MyServiceTest {  
  
 @Test  
 public void testVerifyInteraction() {  
 // Step 1: Create a mock object  
 ExternalApi mockApi = Mockito.*mock*(ExternalApi.class);  
  
 // Step 2: Inject mock into service and call method  
 MyService service = new MyService(mockApi);  
 service.fetchData();  
  
 // Step 3: Verify the interaction  
 *verify*(mockApi).getData();  
 }  
}

**Pom.xml Code**:

<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>MockitoVerifyInteraction</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>  
  
 <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>5.12.0</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Maven Surefire Plugin for running tests -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>3.2.5</version>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Output:**



**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");  
 }  
}

**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>

**Pom.xml Code**:

<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>  
  
 <properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
 <!-- SLF4J API -->  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.30</version>  
 </dependency>  
  
 <!-- Logback Classic -->  
 <dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-classic</artifactId>  
 <version>1.2.3</version>  
 </dependency>  
 </dependencies>  
</project>

**Output**:

