**Cognizant - DN 4.0 Deep Skilling Stage**

**Mandatory Hands-On Week 2 Solutions**

1. **PL/SQl**

Exercise 1: Control Structures

**Scenario 1: Apply a 1% discount to loan interest rates for customers above 60 years old**

BEGIN

FOR customer\_rec IN (

SELECT CustomerID, InterestRate

FROM Customers

WHERE Age > 60

) LOOP

UPDATE Customers

SET InterestRate = InterestRate - 1

WHERE CustomerID = customer\_rec.CustomerID;

END LOOP;

COMMIT;

END;

**Scenario 2: Promote customers to VIP status based on balance over $10,000**

BEGIN

FOR cust IN (

SELECT CustomerID

FROM Customers

WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

END LOOP;

COMMIT;

END;

**Scenario 3: Send reminders to customers with loans due in the next 30 days**

BEGIN

FOR loan\_rec IN (

SELECT CustomerID, LoanID, DueDate

FROM Loans

WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Loan ID ' || loan\_rec.LoanID ||

' for Customer ID ' || loan\_rec.CustomerID ||

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

);

END LOOP;

END;

Exercise 2: Stored Procedures

**Scenario 1: ProcessMonthlyInterest – Apply 1% Interest to All Savings Accounts**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR r IN (

SELECT AccountID

FROM Accounts

WHERE AccountType = 'SAVINGS'

) LOOP

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountID = r.AccountID;

END LOOP;

COMMIT;

END;

/

**Scenario 2: UpdateEmployeeBonus – Add Bonus % to All Employees in a Department**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept\_id IN NUMBER,

p\_bonus\_pct IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary \* (1 + p\_bonus\_pct / 100)

WHERE DeptID = p\_dept\_id;

COMMIT;

END;

/

**Scenario 3: TransferFunds – Transfer Amount Between Two Accounts with Balance Check**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_acct IN NUMBER,

p\_to\_acct IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_from\_acct

FOR UPDATE;

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance');

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_acct;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_acct;

COMMIT;

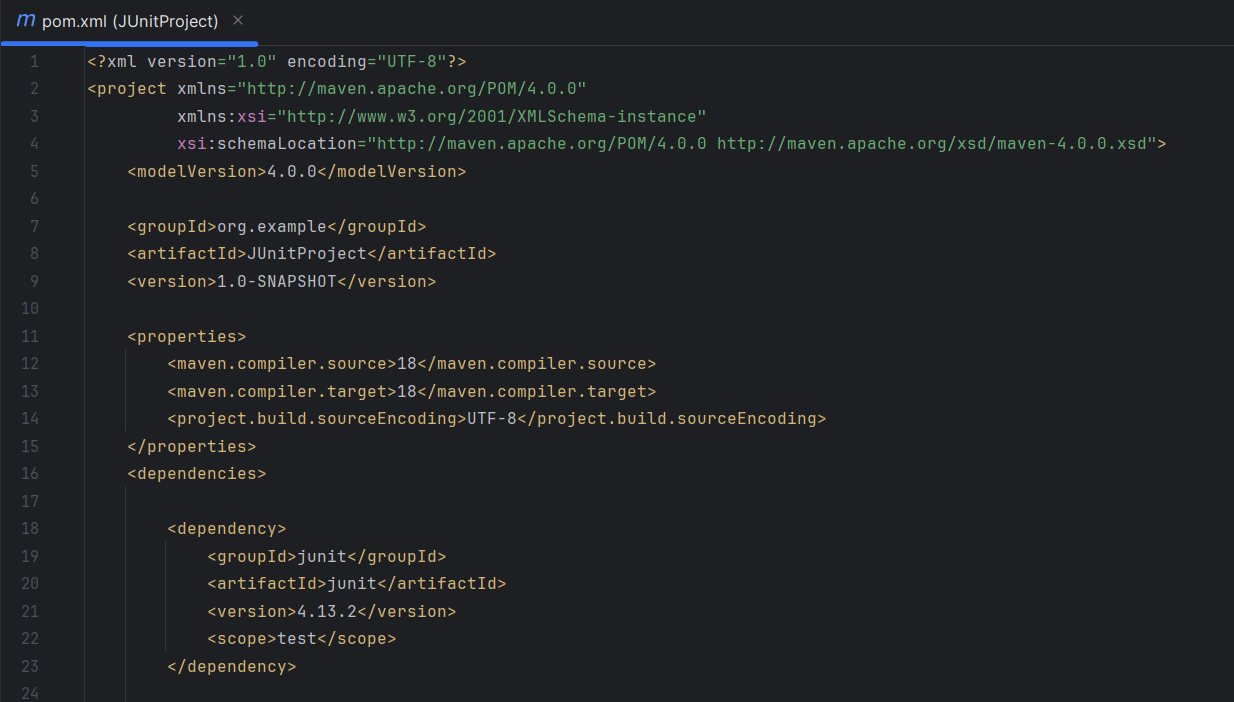
END;

/

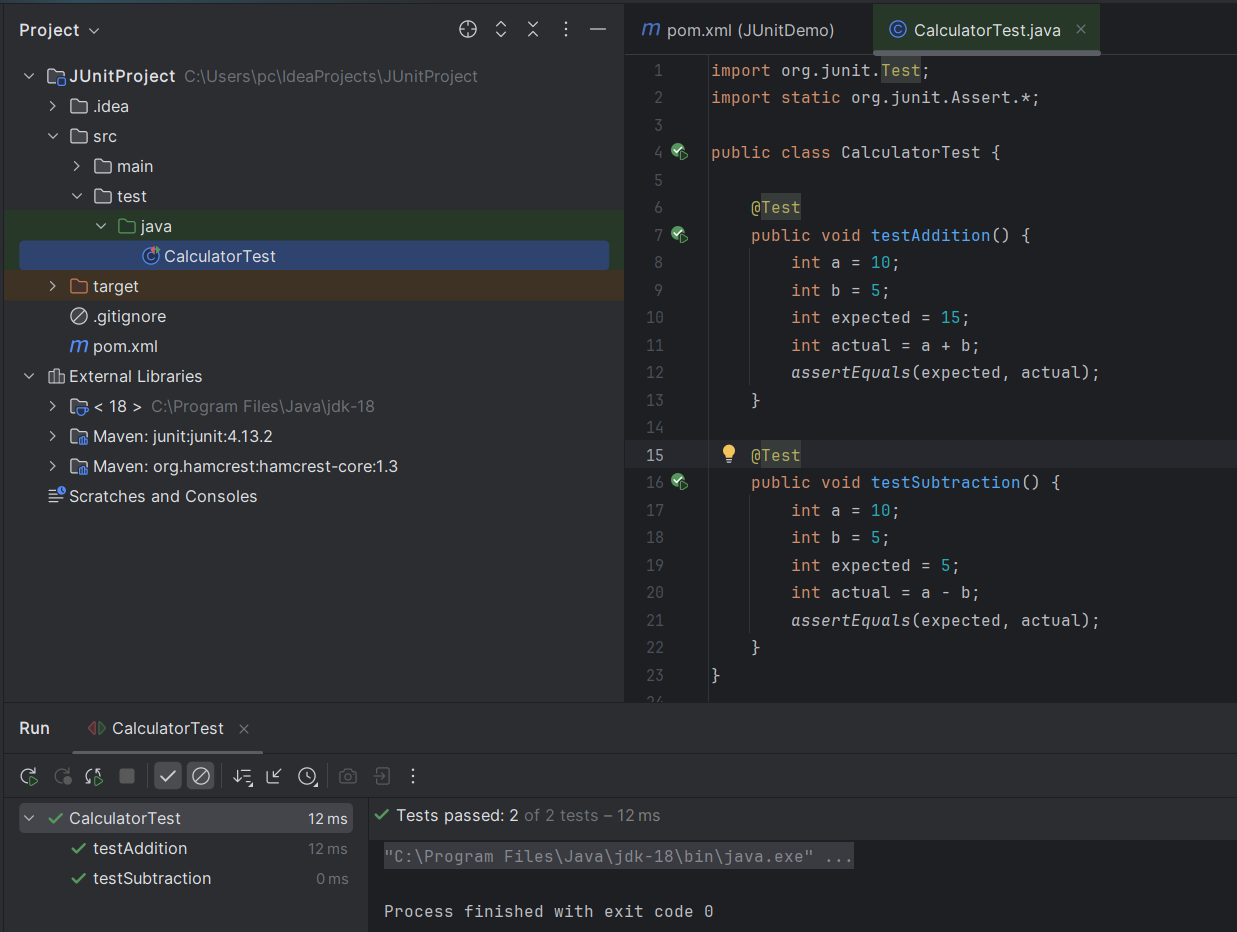
1. **JUnit\_Basic Testing Exercises**

Exercise 1: Setting Up JUnit

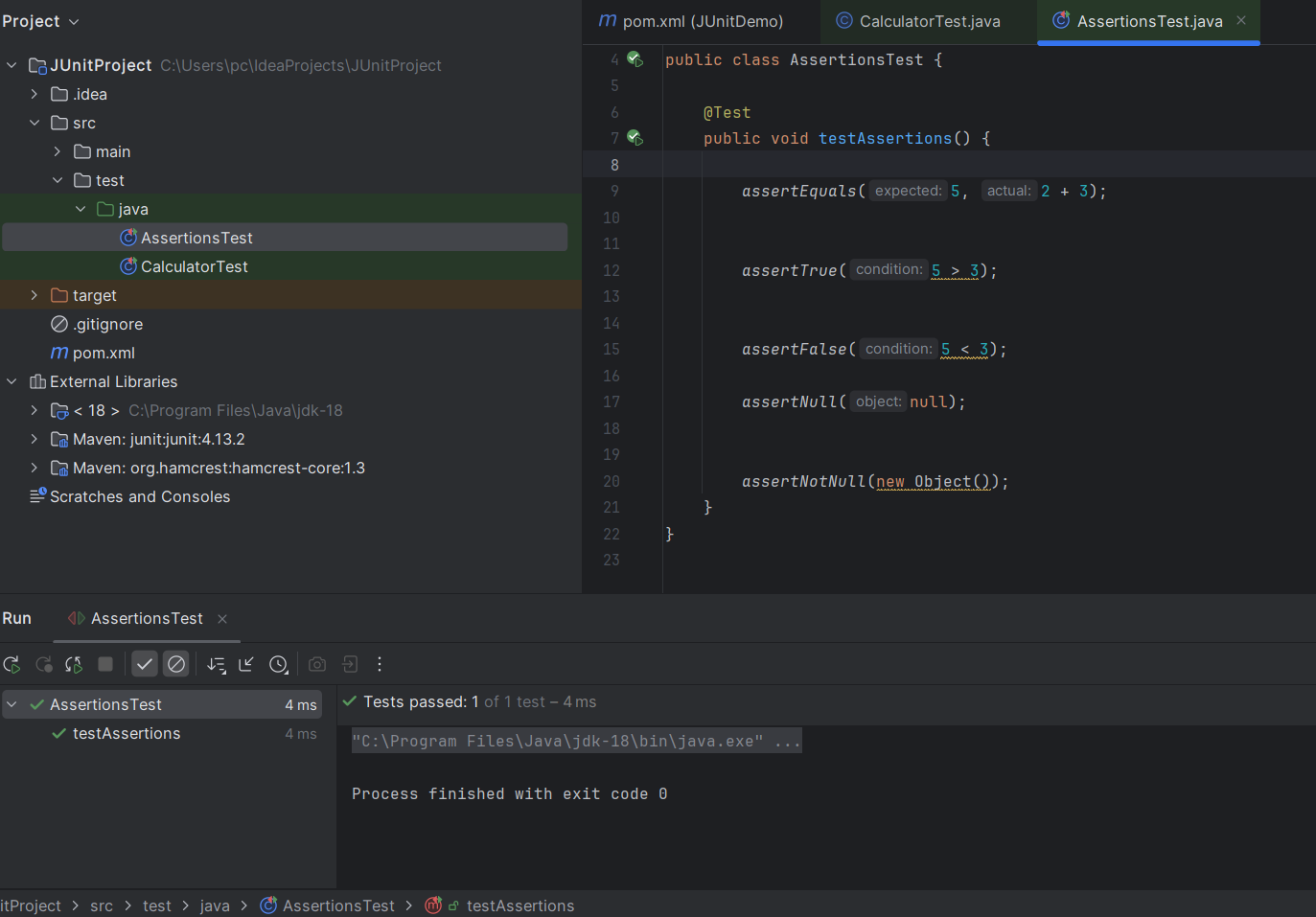
Adding dependencies in pom.xml:

****

Making a Test class to test it:

****

Exercise 2: Assertions in JUnit

****

## 

## 

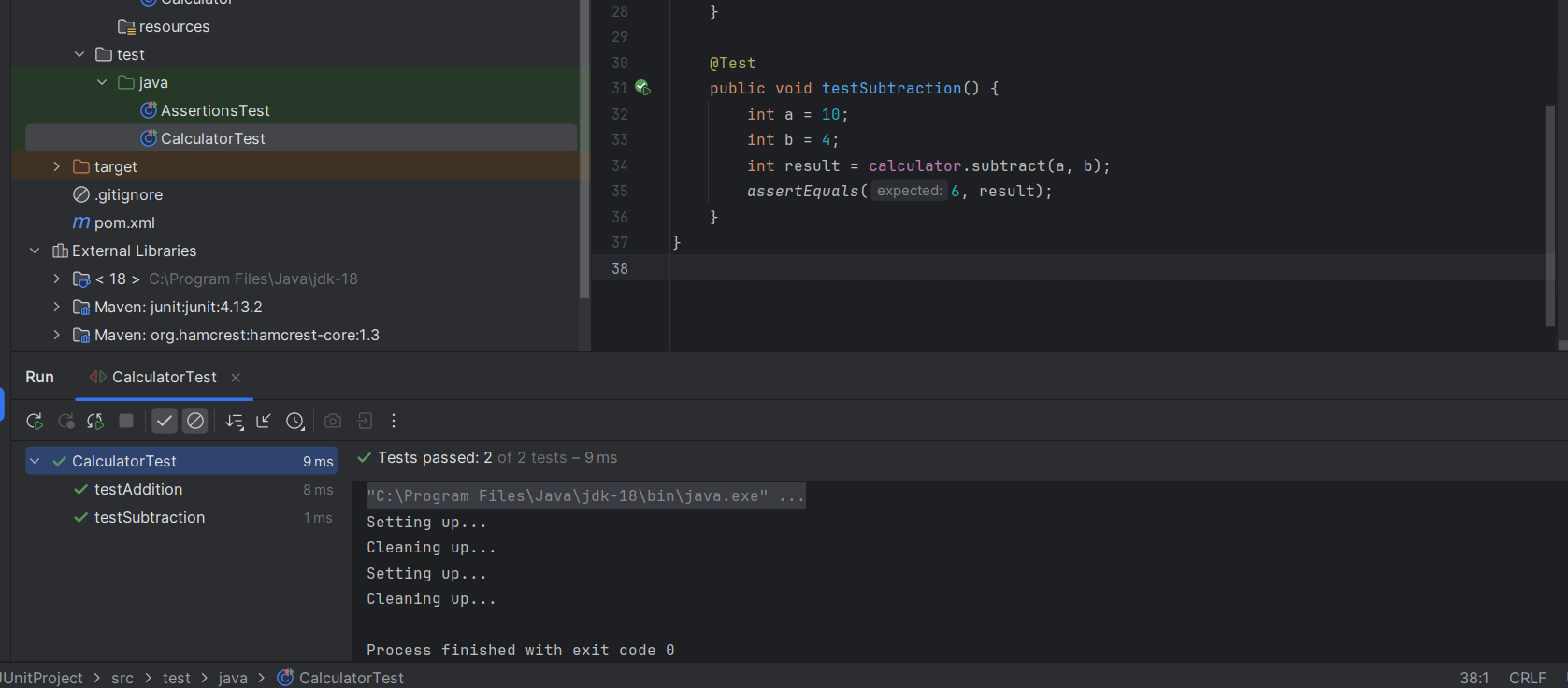
## 

## 

## Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and

## Teardown Methods in JUnit

## 



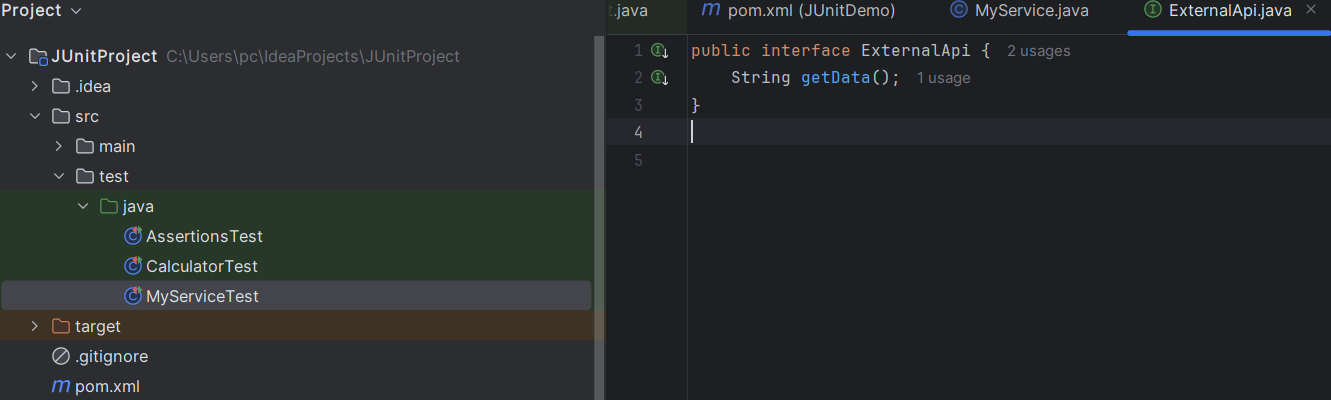
**3. Mockito exercises**

**Exercise 1: Mocking and Stubbing**

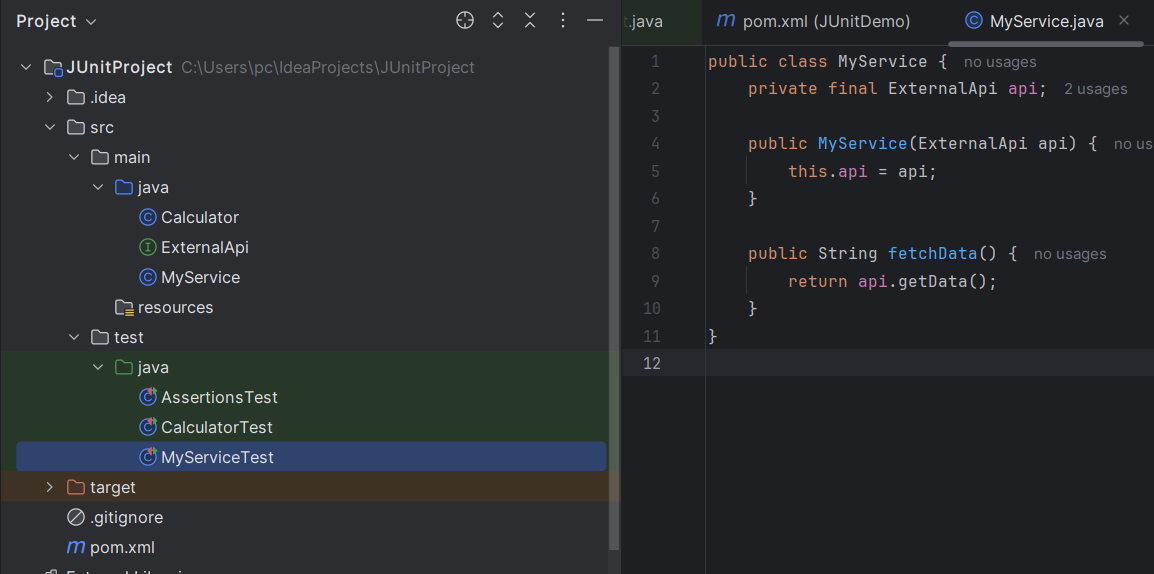
Add dependency in pom.xml



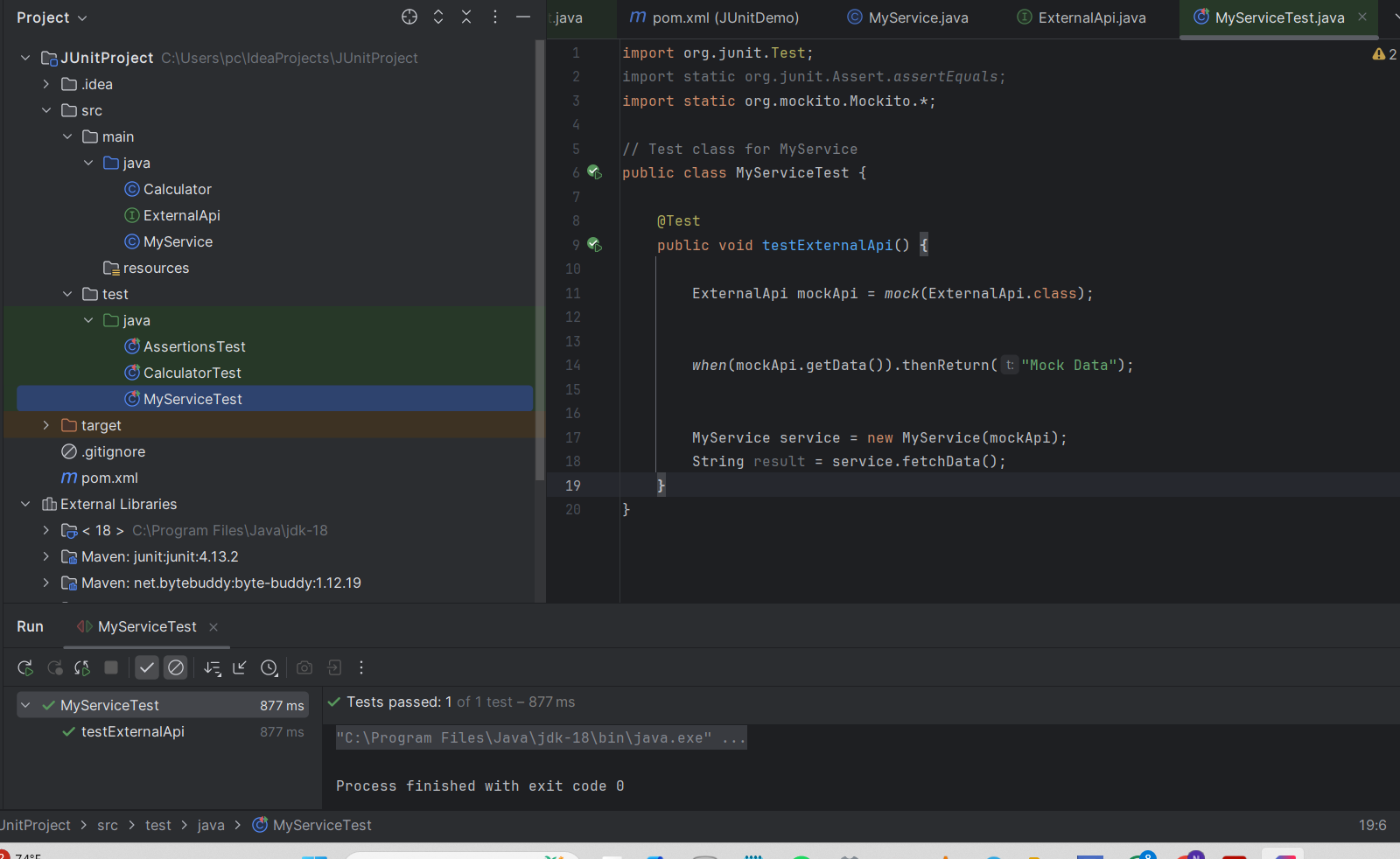
ExternalApi.java in src/main/java



MyService.java in src/main/java



MyServiceTest.java



**4.Logging using SLF4J**

Exercise 1: Logging Error Messages and Warning Levels

Add dependencies:

