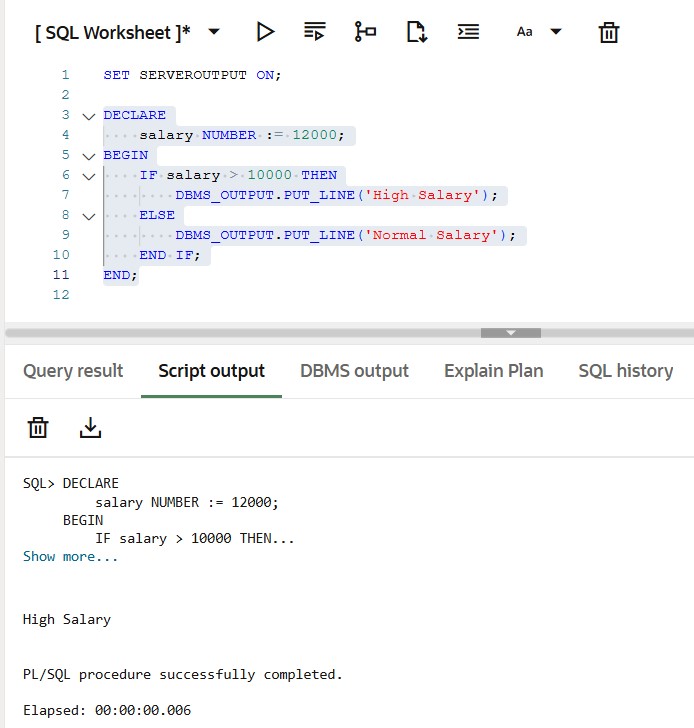
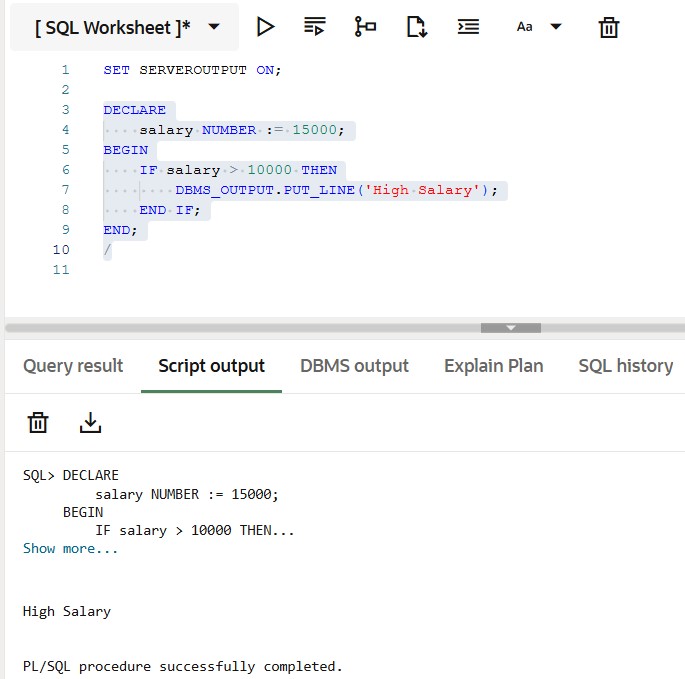
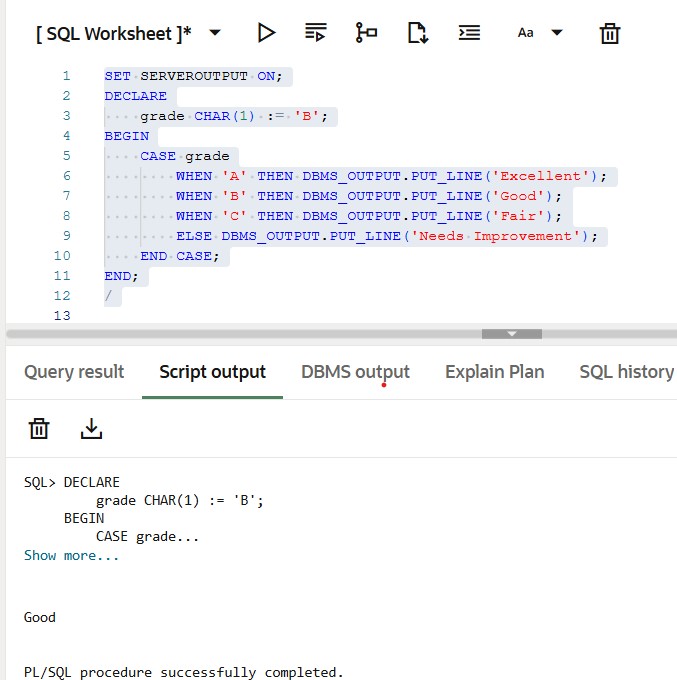
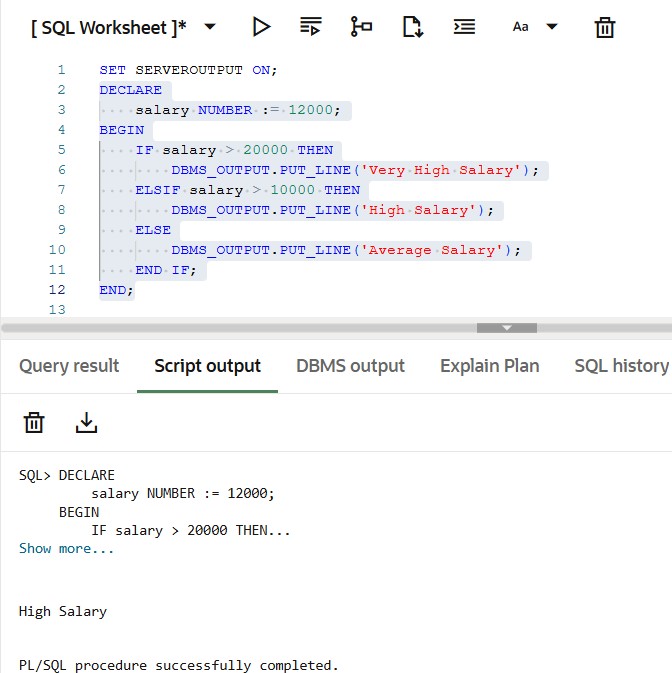
# PL/SQL

*EXERCISE 1:*

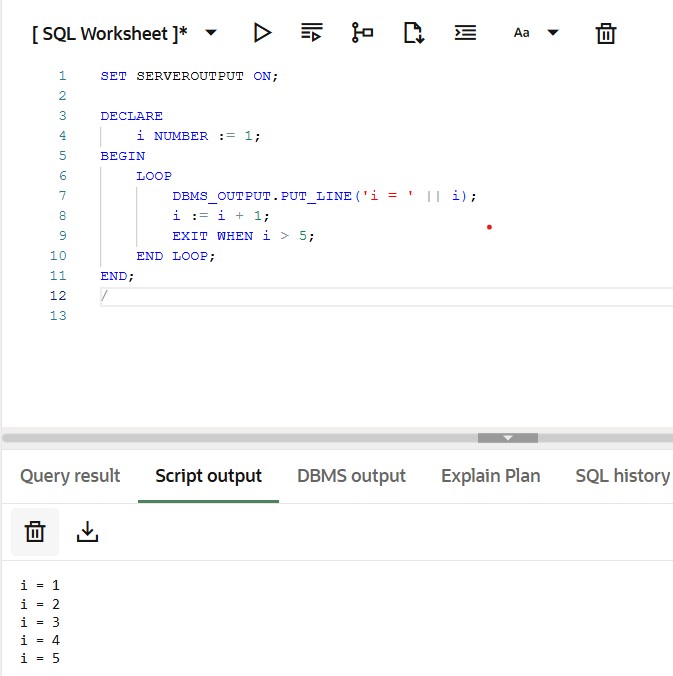
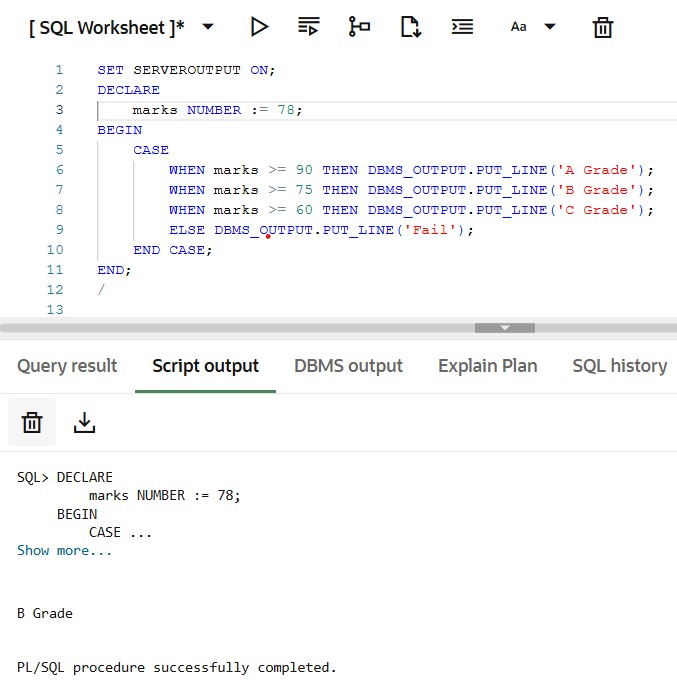
1. IF...THEN 2. IF...THEN...ELSE



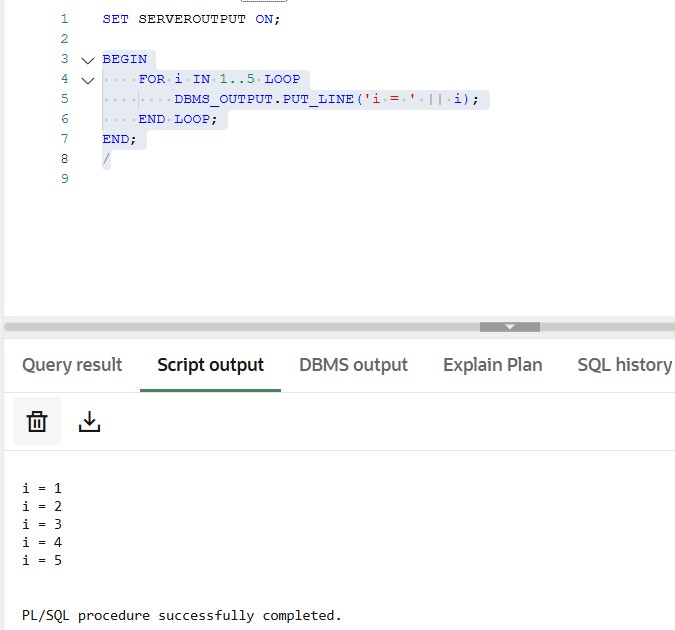
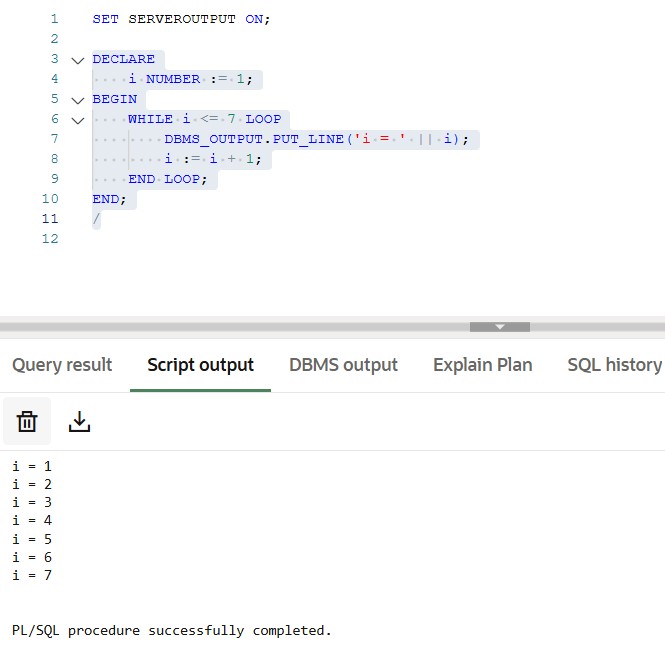
1. 3. IF...ELSIF...ELSE 4. Simple CASE Statement



5. Searched CASE Statement 6. LOOP (Basic loop)



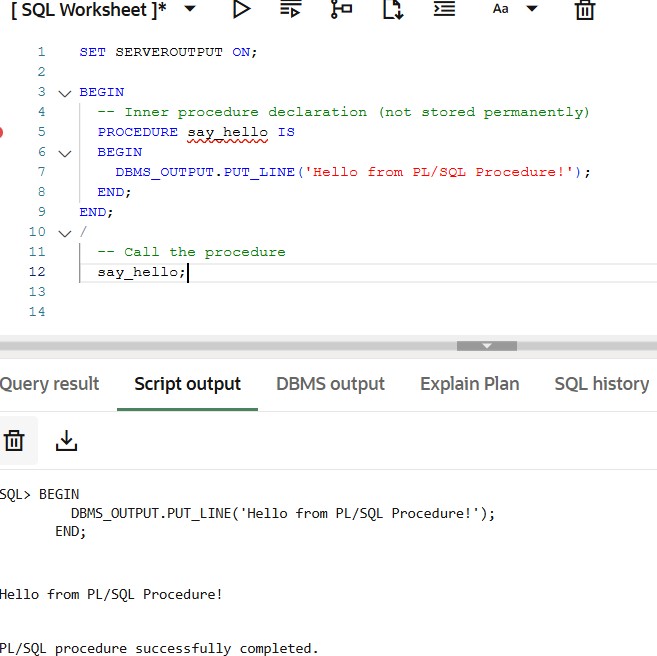
7. WHILE loop 8. FOR loop



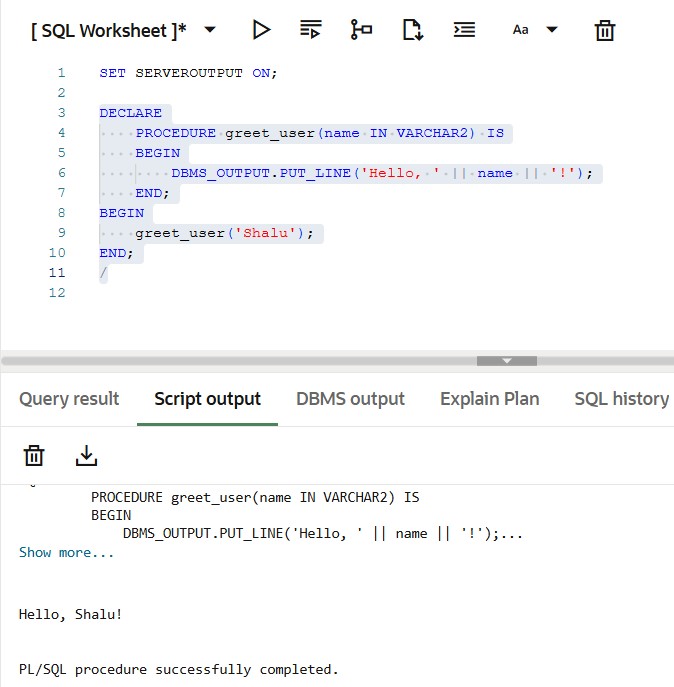
**EXERCISE 2:** *STORED PROCEDURE* **What is a Stored Procedure?**

A stored procedure is a named block of code that performs a task and is stored in the Oracle database. You can call it whenever needed.

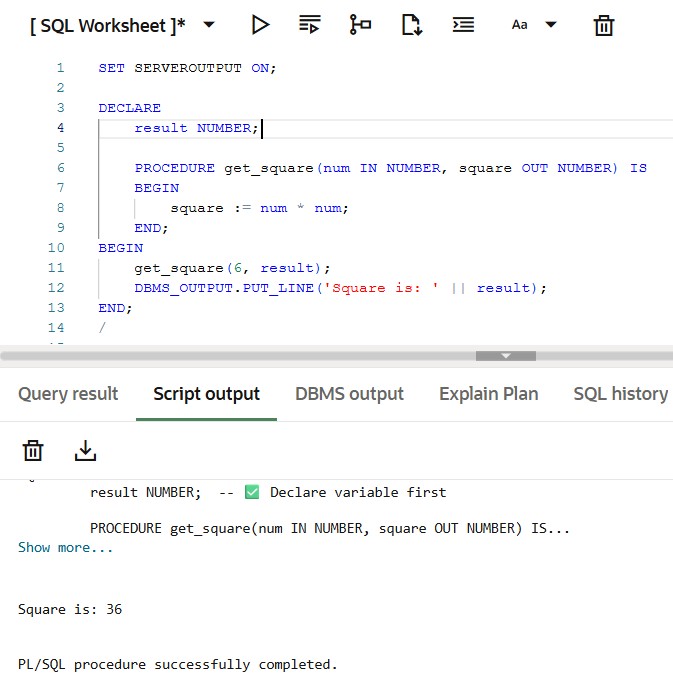
1. Simple Stored Procedure: Basic procedure with no parameters.



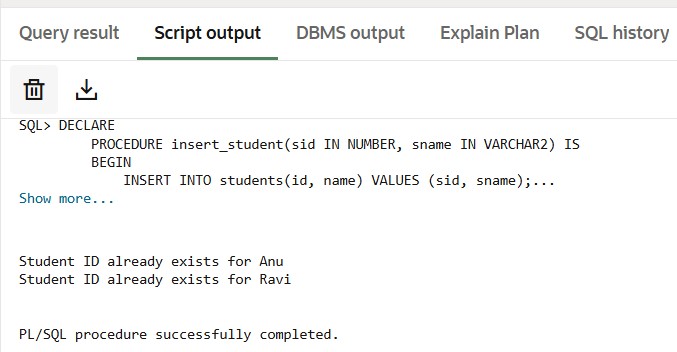
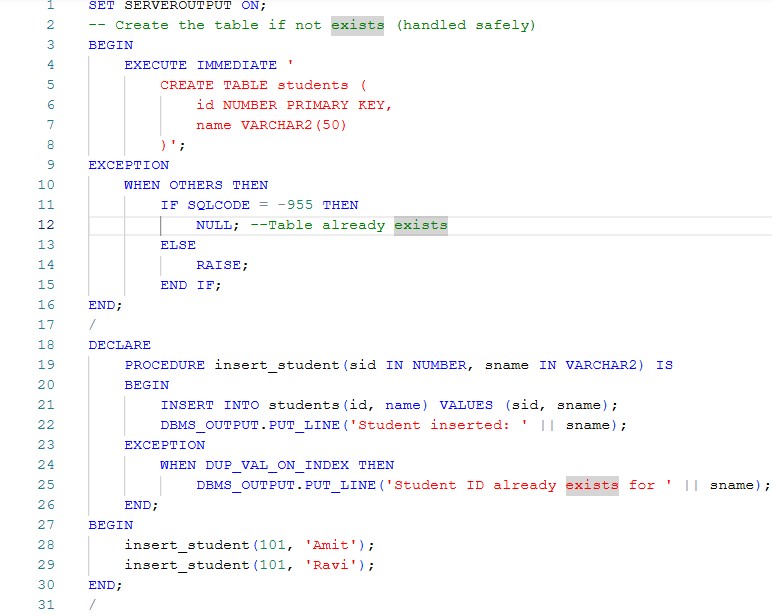
1. Procedure with IN Parameter: Learn how to pass input to a procedure



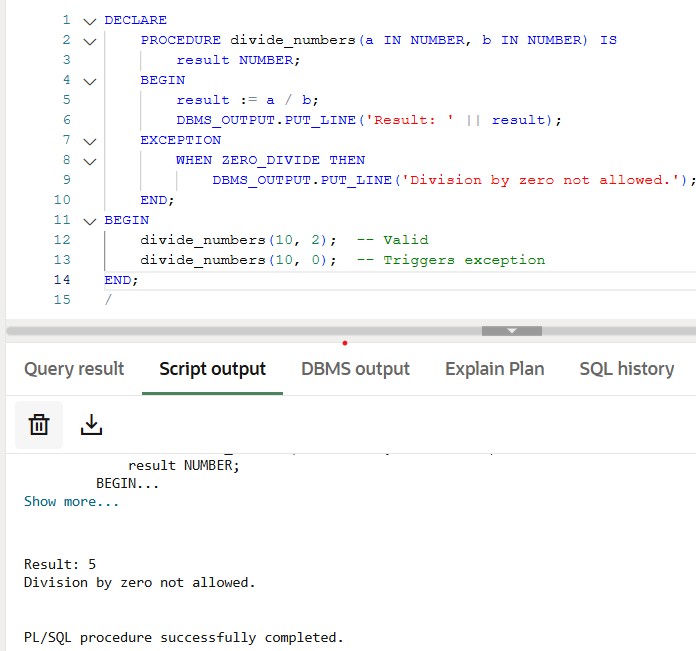
1. Procedure with IN and OUT Parameters: Learn how to return values from a procedure using OUT.



1. Procedure with DML (Insert into a table)

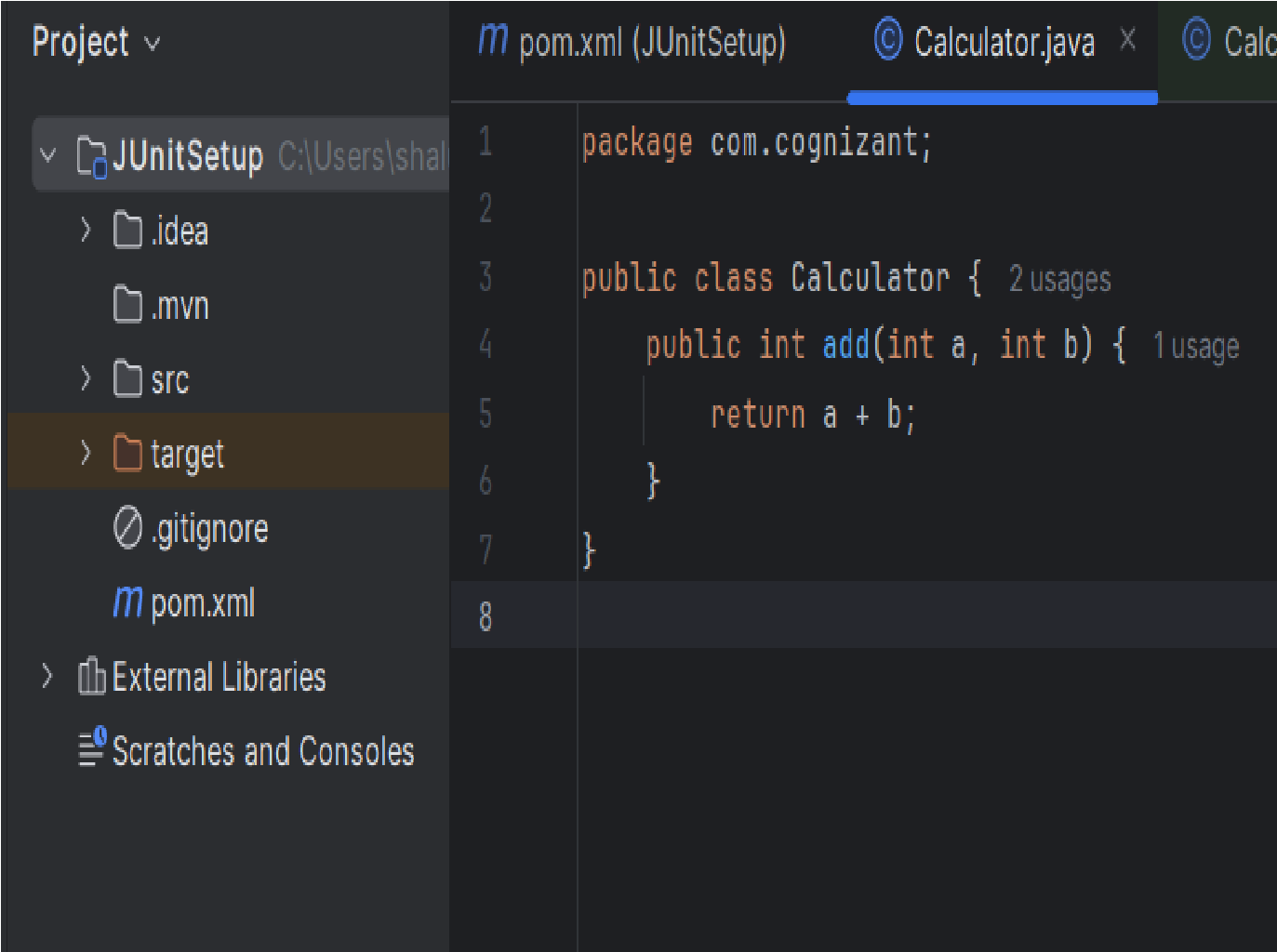


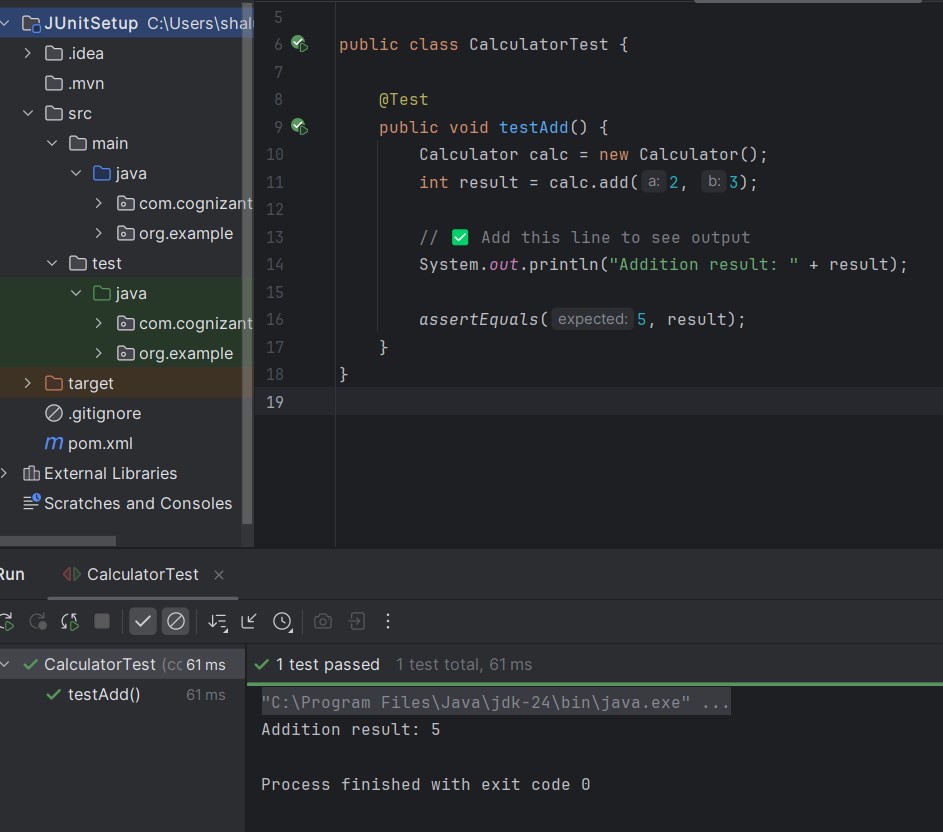
1. Procedure with exception handling



# TDD using JUnit5 and Mockito

***Exercise 1:*** Setting Up JUnit

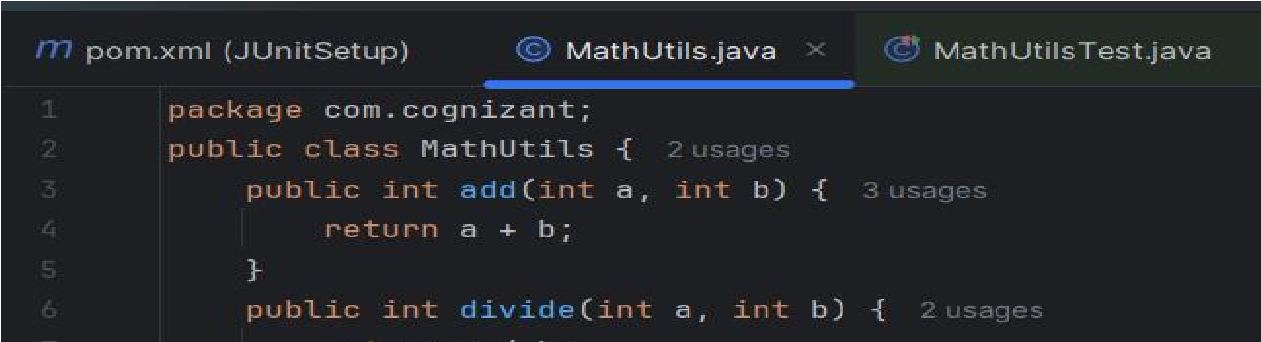


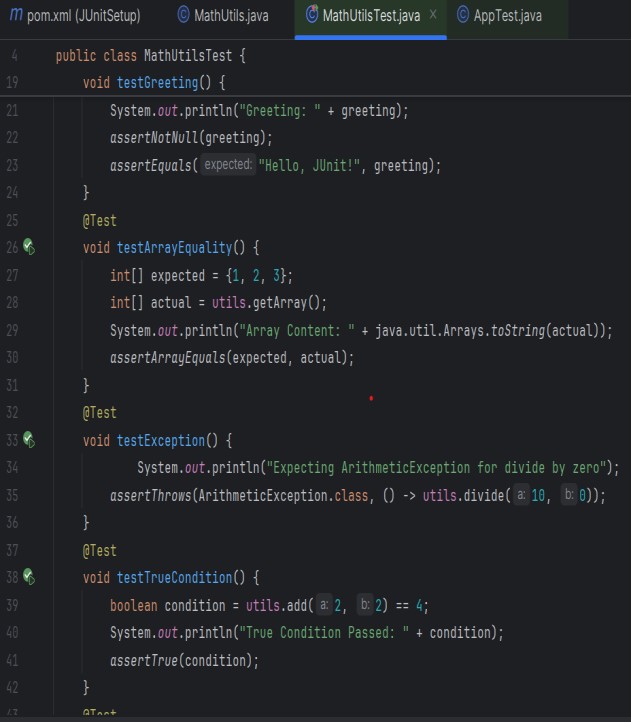
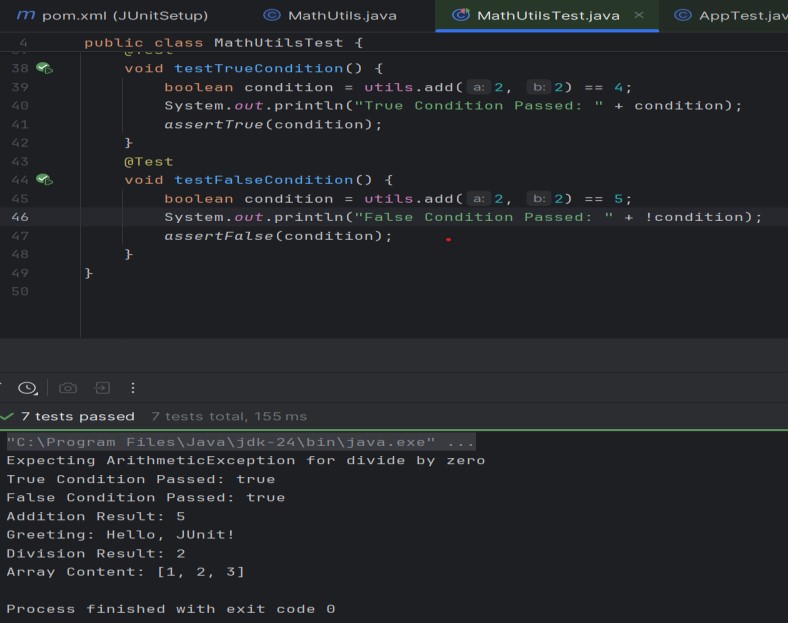
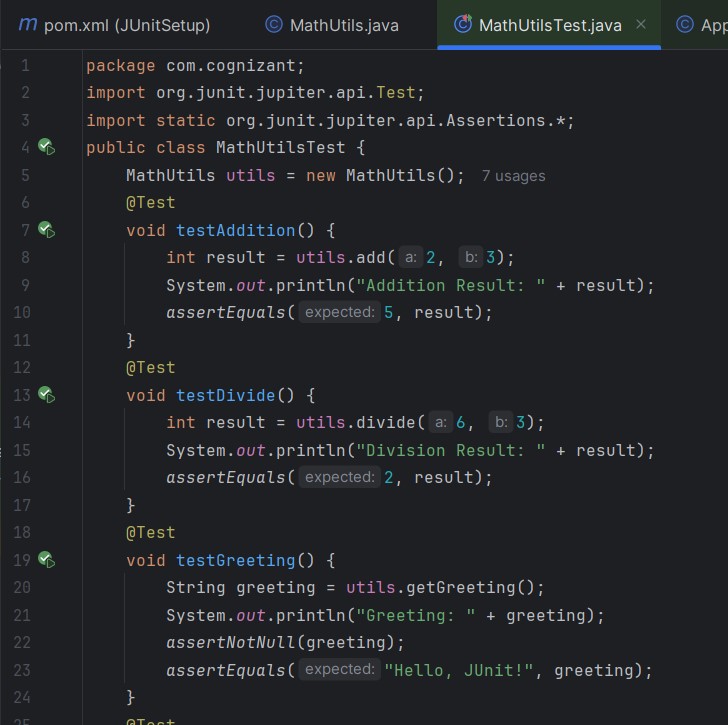


***Exercise 2:*** Assertions in JUnit

***Assertion, Use case***

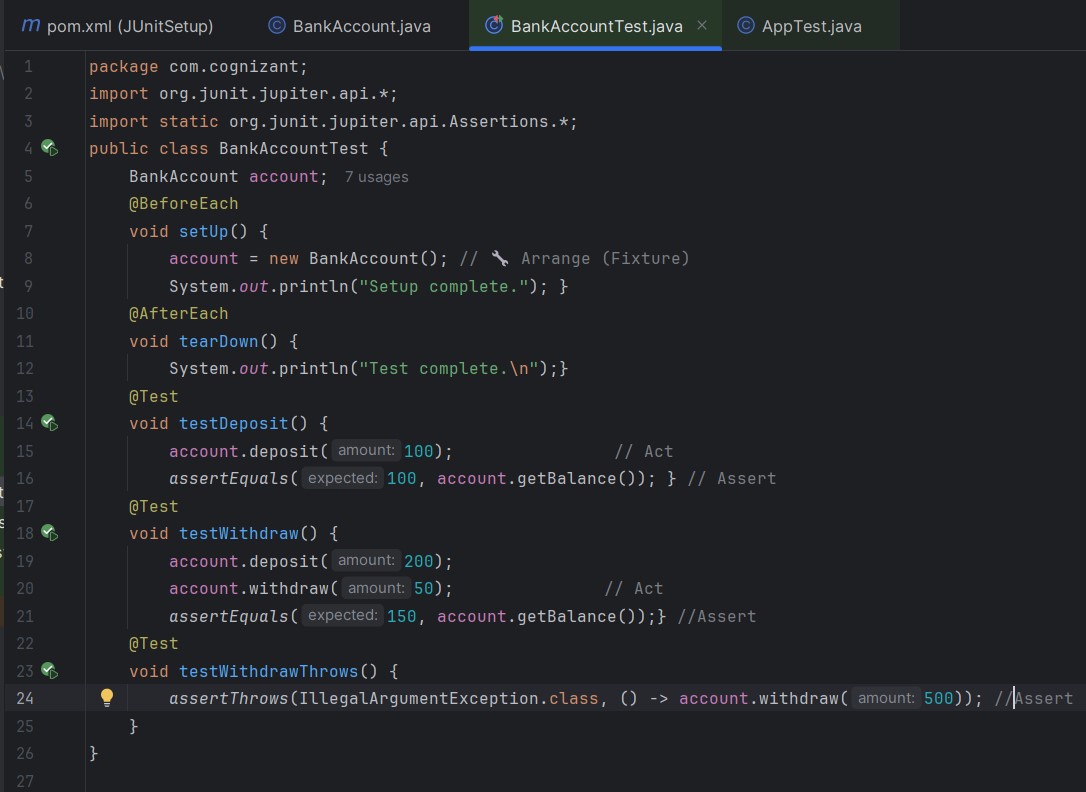
* AssertEquals: Compare expected and actual values
* AssertNotNull: ensure object is not null
* AssertArrayEquals: Compare arrays
* AssertThrows: Test if exception is thrown
* AssertTrue/False: Validate boolean conditions

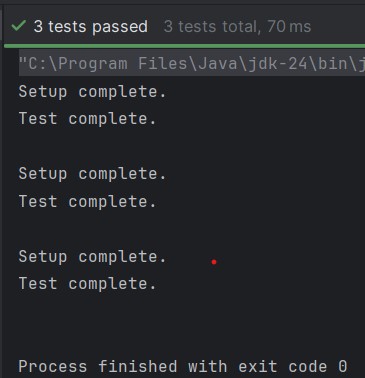




***Exercise 3***: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods

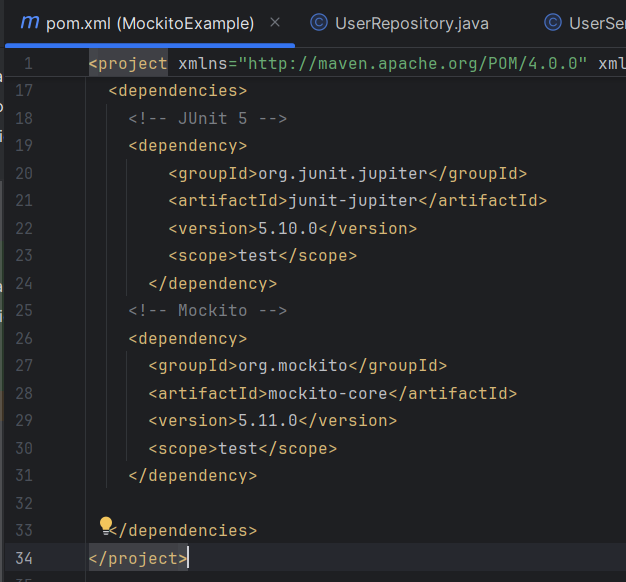
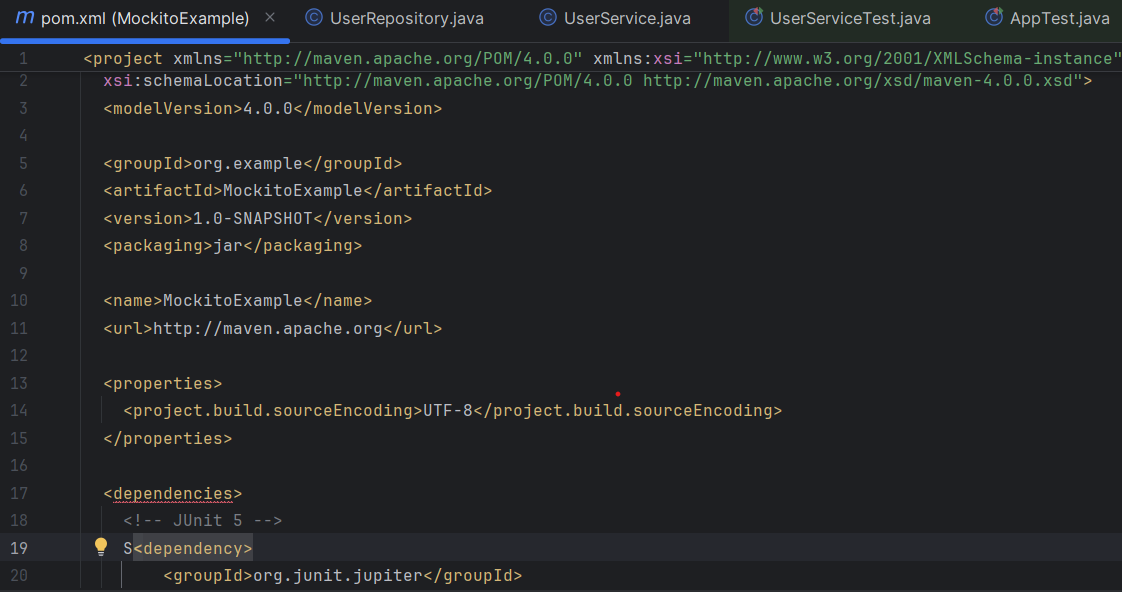
The **Arrange-Act-Assert (AAA)** Pattern helps organize each test case into three clear steps: first, you arrange or set up the necessary data and environment; second, you act by calling the method under test; and finally, you assert the expected outcome. A test fixture refers to reusable setup code, such as creating a Bank Account object before each test, which ensures consistency and avoids duplication. The **@BeforeEach** annotation is used to execute setup code before each test method runs, making sure every test starts with a clean state. Similarly, **@AfterEach** is used to define any cleanup or logging that should happen after each test completes. These practices together make your tests more readable, maintainable, and reliable.

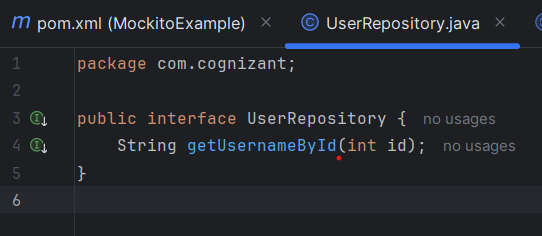


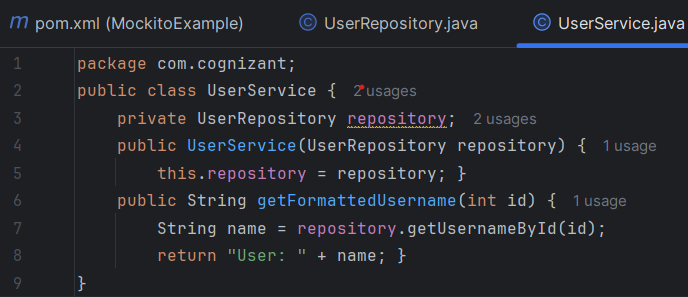


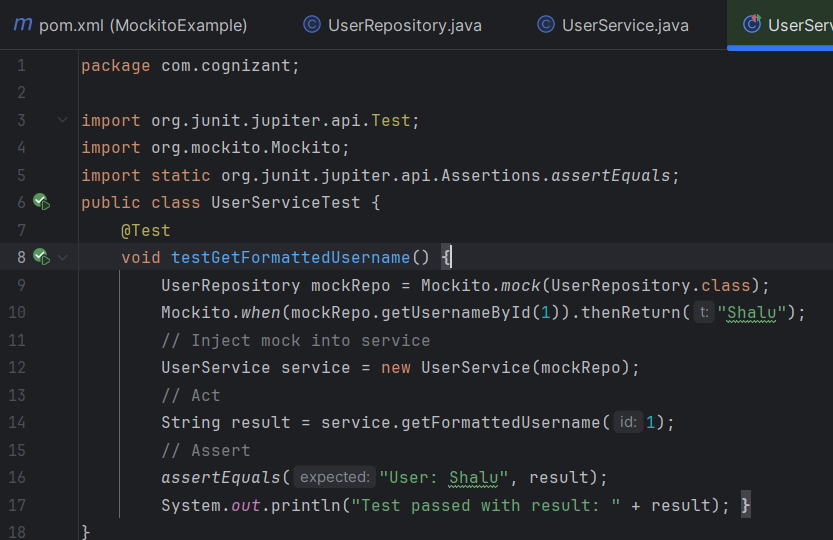
**Mockito exercises**

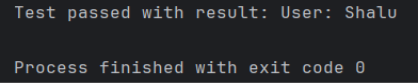
**Exercise 1: Mocking and Stubbing**







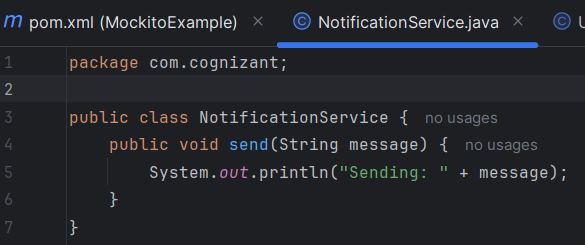


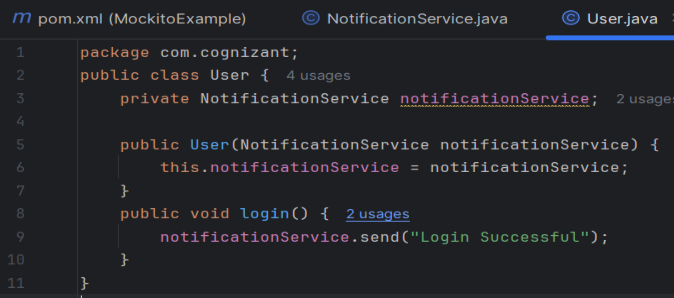


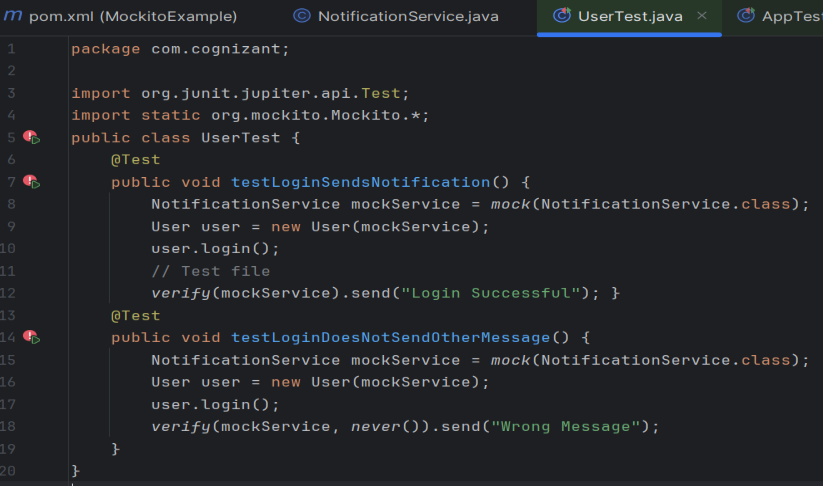
***Exercise 2:***   **Verifying Interactions**

In Mockito, verifying interactions helps ensure that specific methods were called (or not called) on mocked dependencies. This is crucial when testing behavior — especially in scenarios like sending notifications, updating databases, or logging. For instance, after a login() method is called, we may want to confirm that a notification service’s send() method was triggered exactly once. Mockito provides verify() for this purpose, and it also allows checking that no more or unwanted interactions occurred using methods like verifyNoMoreInteractions() or verifyNever().

**Example: Verifying Login Notification with Mockito**







***SL4J Logging exercises***

**Exercise 3: Logging Error Messages and Warning Levels**

