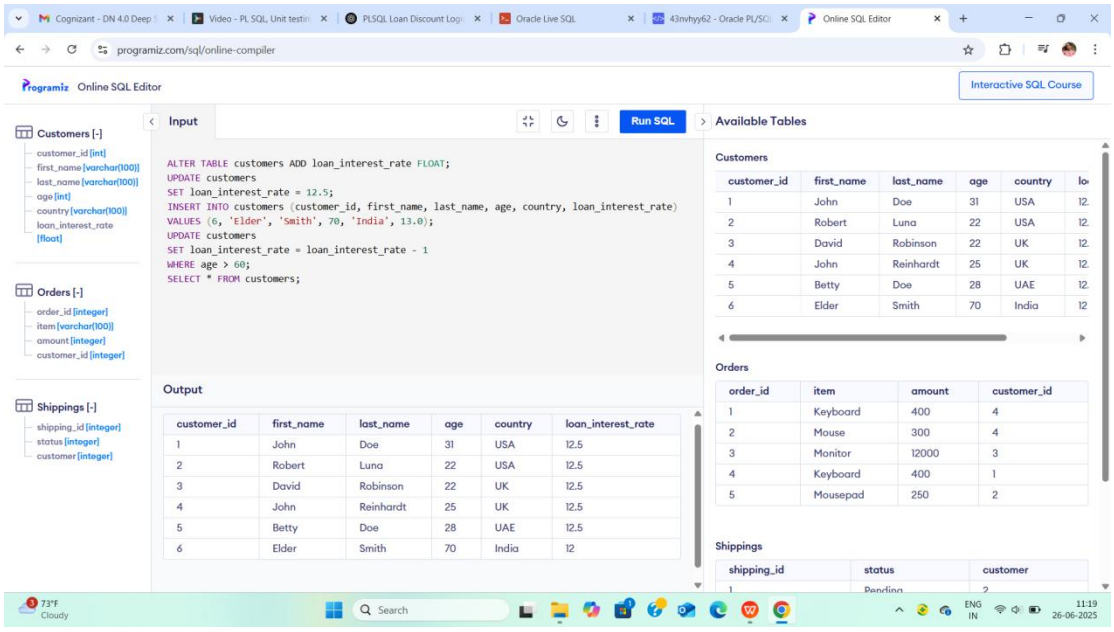
WEEK 2 HANDS ON

Exercise 1:

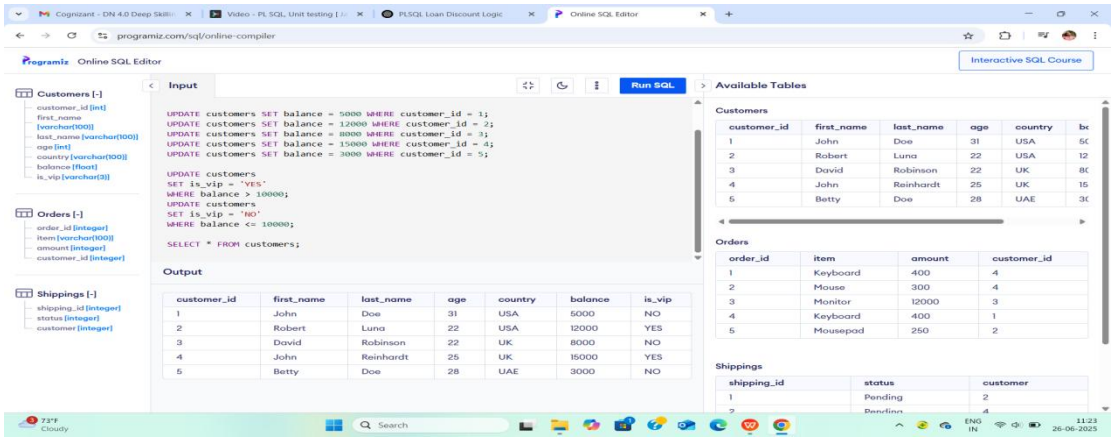
Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.

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.



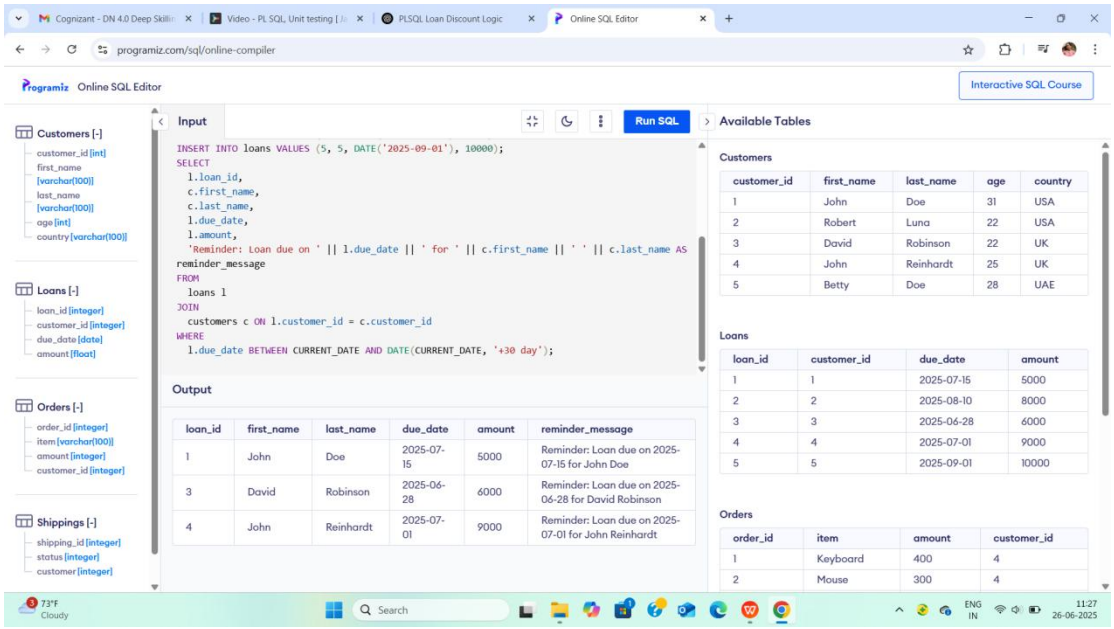
Scenario 2: A customer can be promoted to VIP status based on their balance.

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.

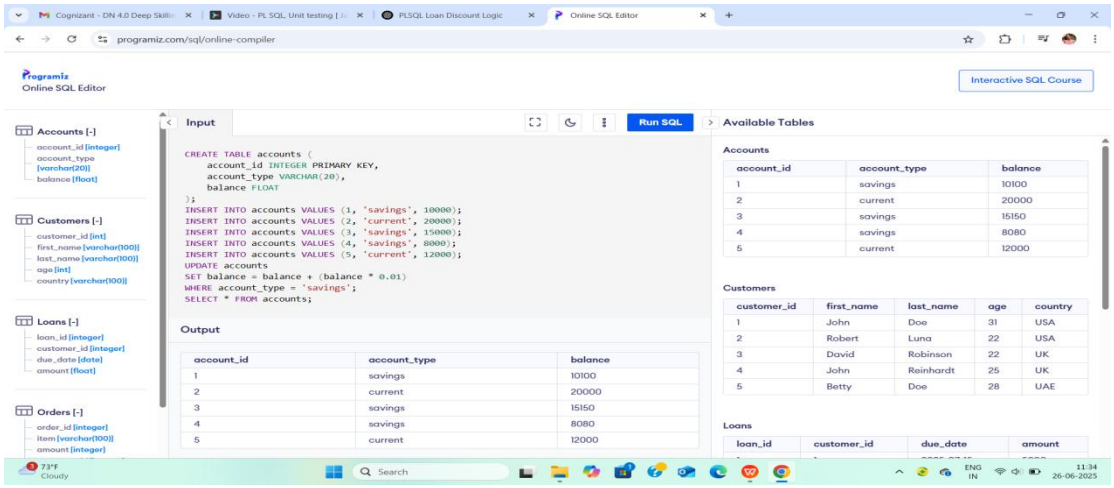
Question: Write a PL/SQL block that fetches all loans due in the next 30 days and prints reminder message for each customer.



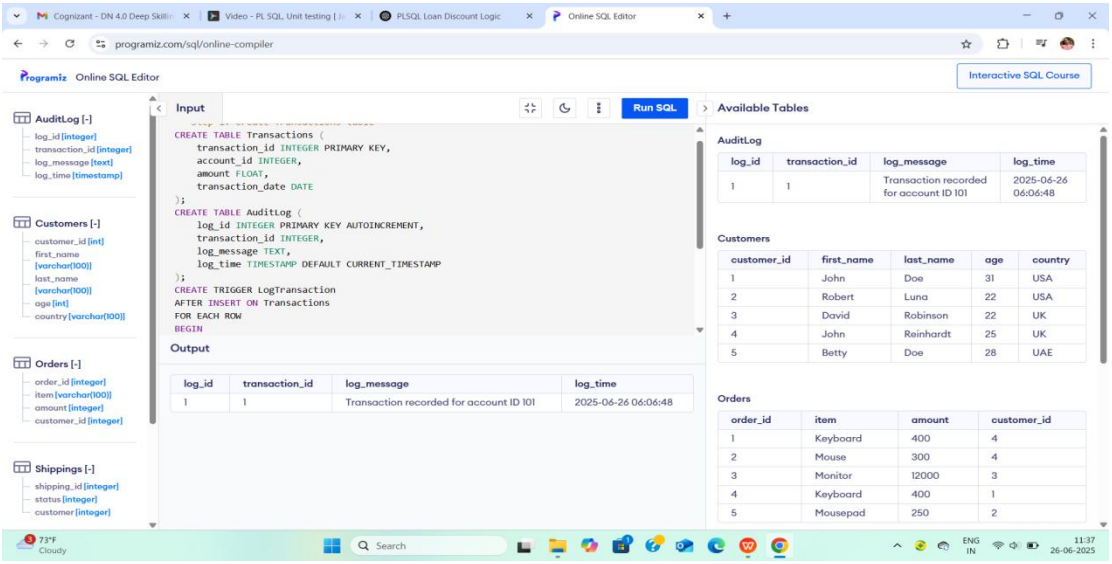
Exercise 2: Stored Procedures

Scenario 1: The bank needs to process monthly interest for all savings accounts.

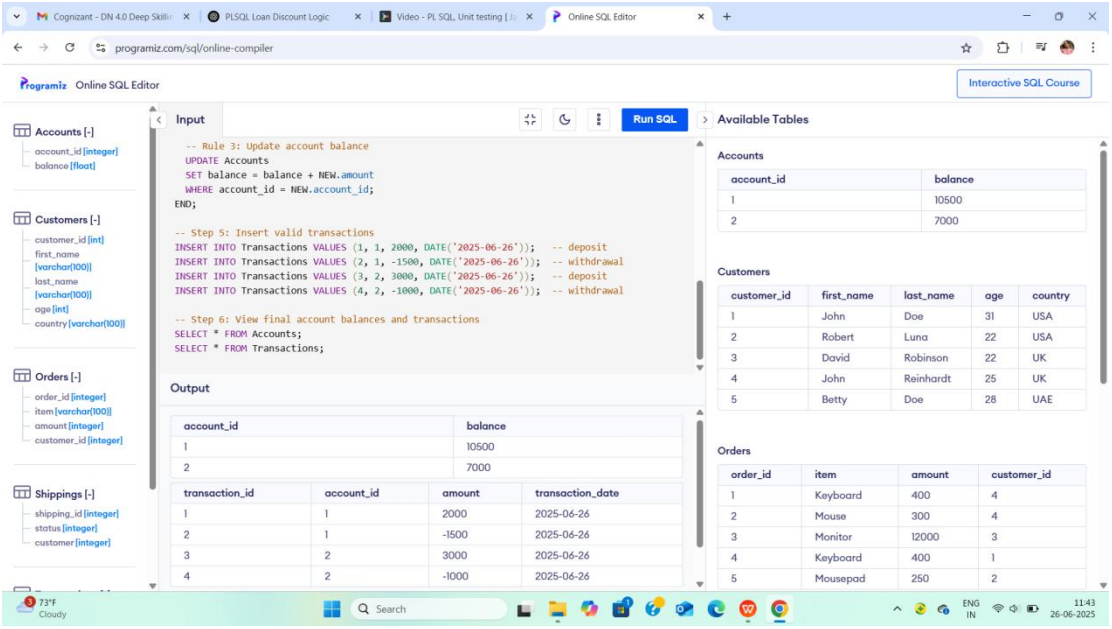
Question: Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.



Scenario 2: Maintain an audit log for all transactions.
Question: Write a trigger **LogTransaction** that inserts a record into an AuditLog table whenever a transaction is inserted into the Transactions table.



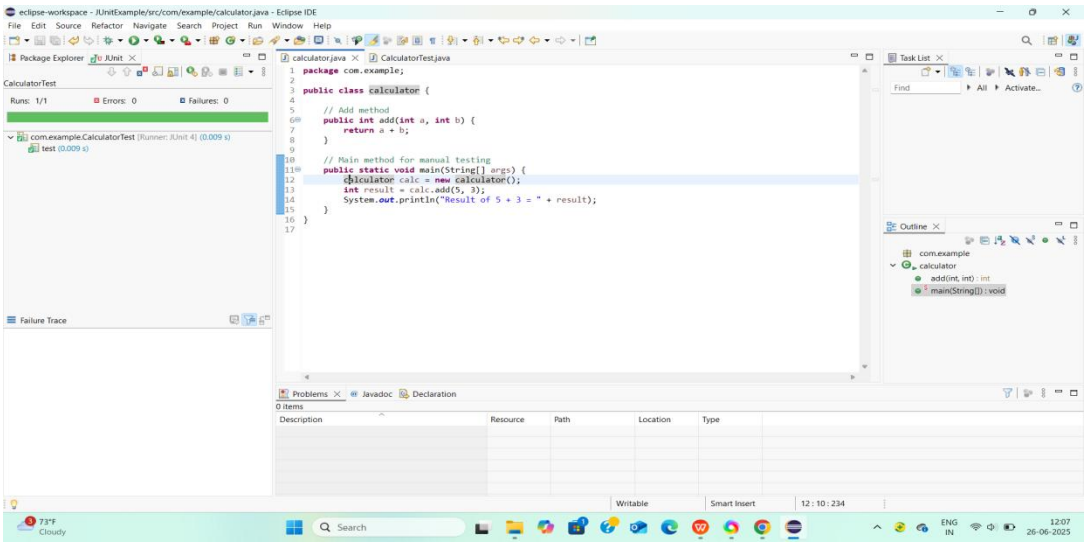
Scenario 3: Enforce business rules on deposits and withdrawals.
Question: Write a trigger **CheckTransactionRules** that ensures withdrawals do not exceed the balance and deposits are positive before inserting a record into the Transactions table.

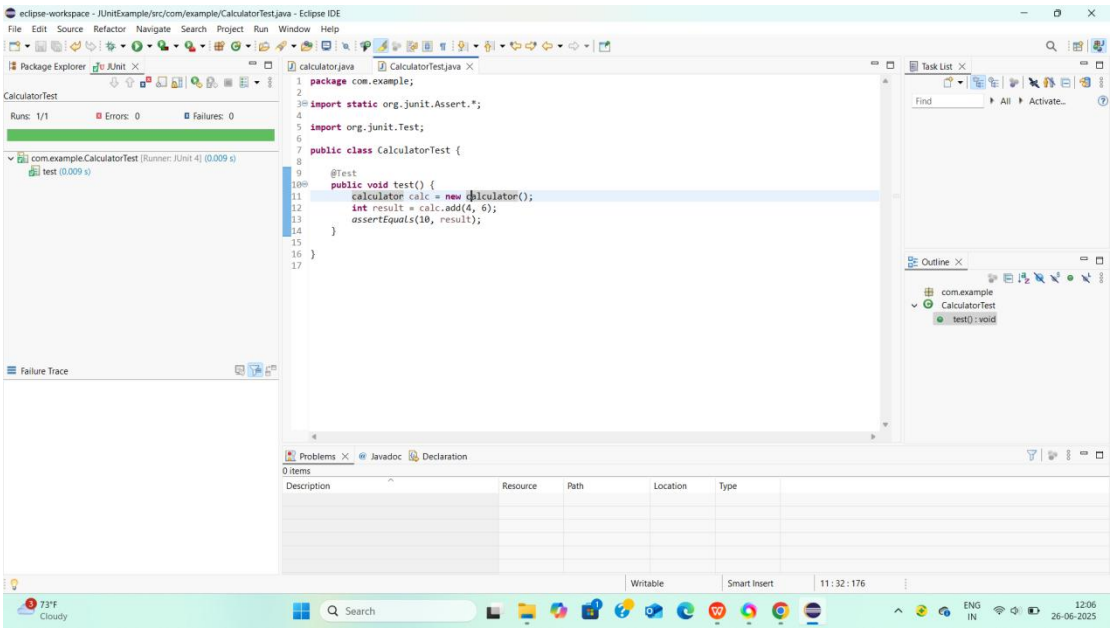


JUNIT EXERCISES

Exercise 1: Setting Up JUnit Scenario:

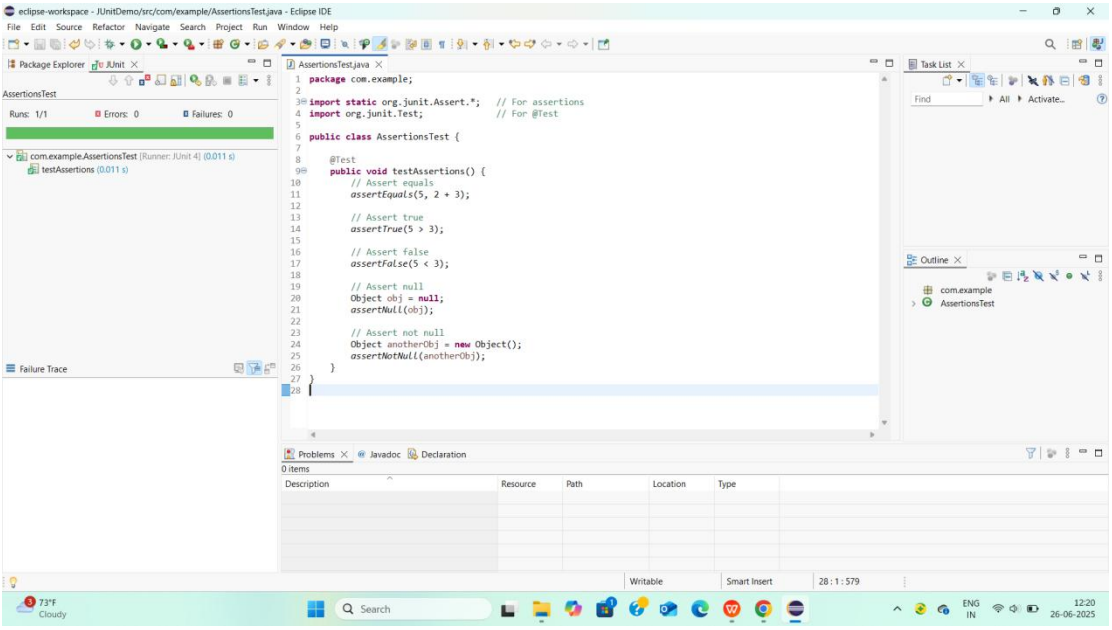
You need to set up JUnit in your Java project to start writing unit tests.
Steps: 1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).
2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml: junit junit 4.13.2 test
3. Create a new test class in your project.





Exercise 3: Assertions in JUnit

Scenario:You need to use different assertions in JUnit to validate your test results.

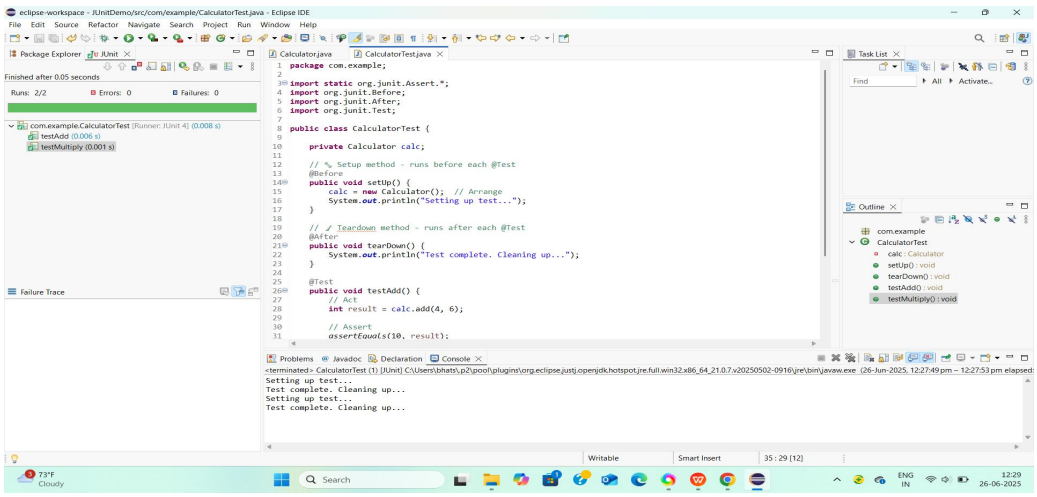


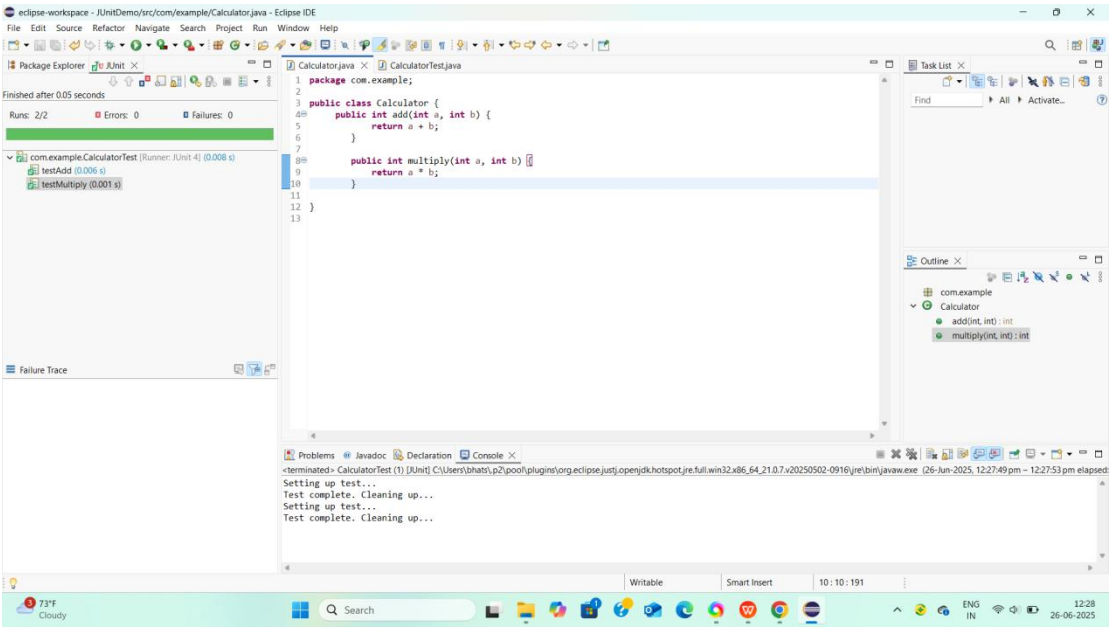
Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

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.

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

3. 



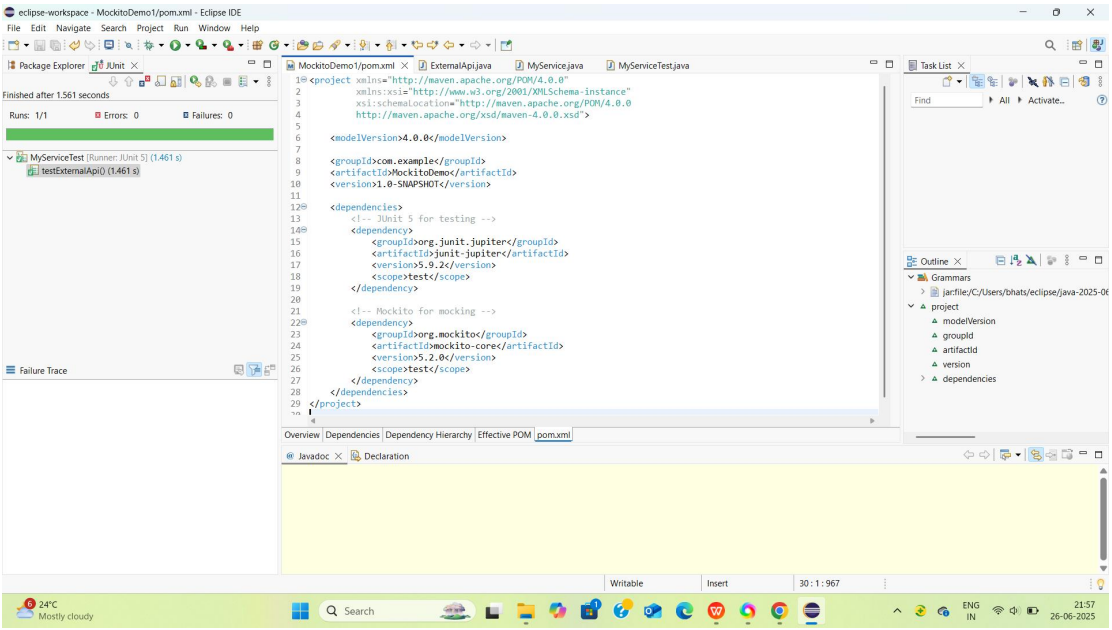
Mockito Hands-On Exercises

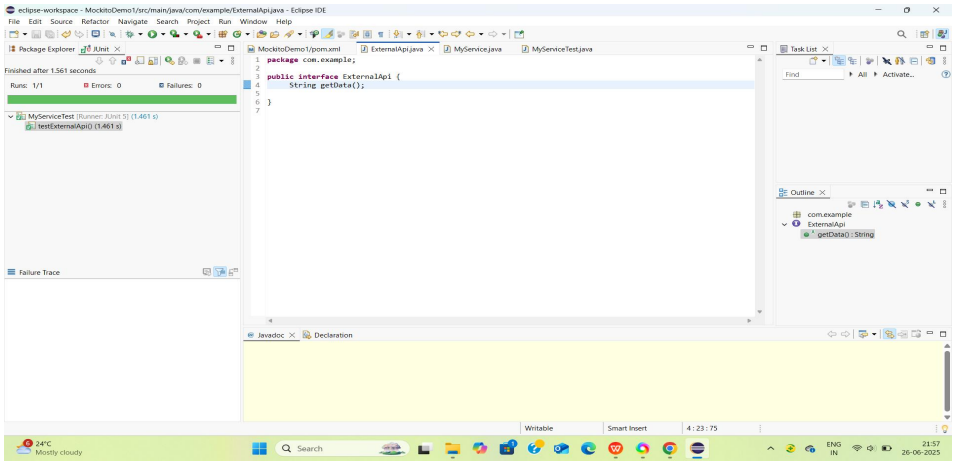
Exercise 1: Mocking and Stubbing

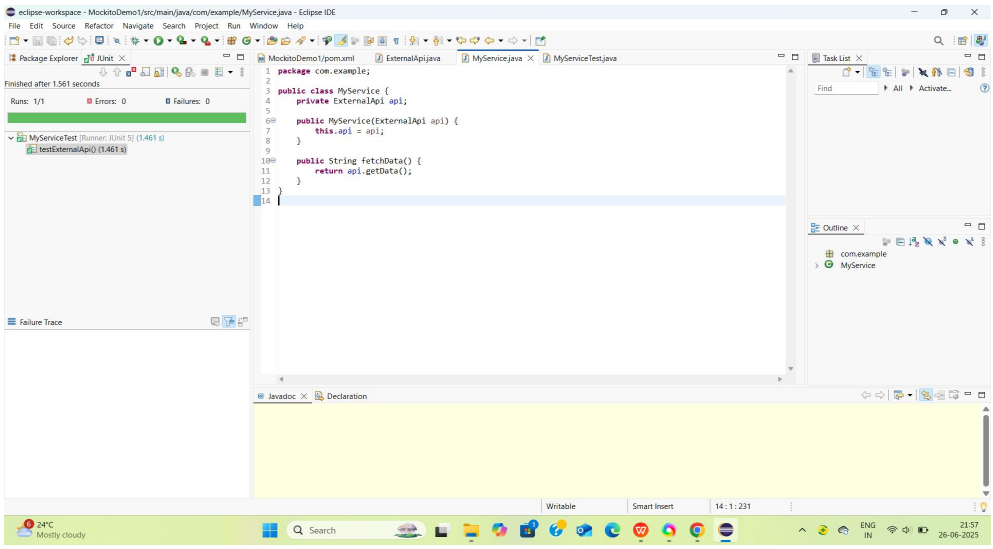
Scenario: You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.

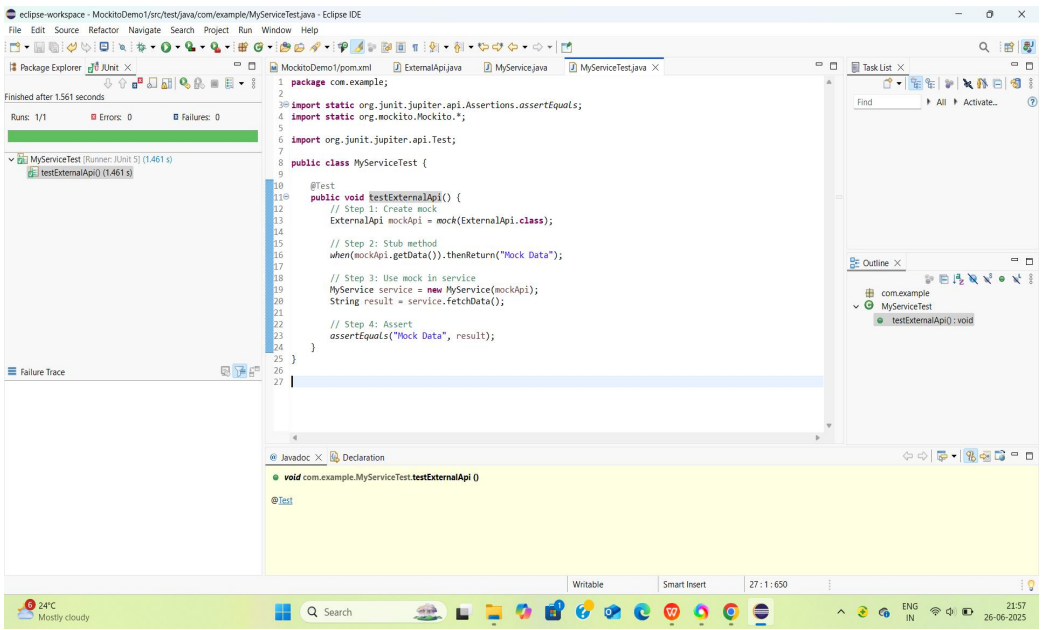
Steps: 1. Create a mock object for the external API.

2. Stub the methods to return predefined values.



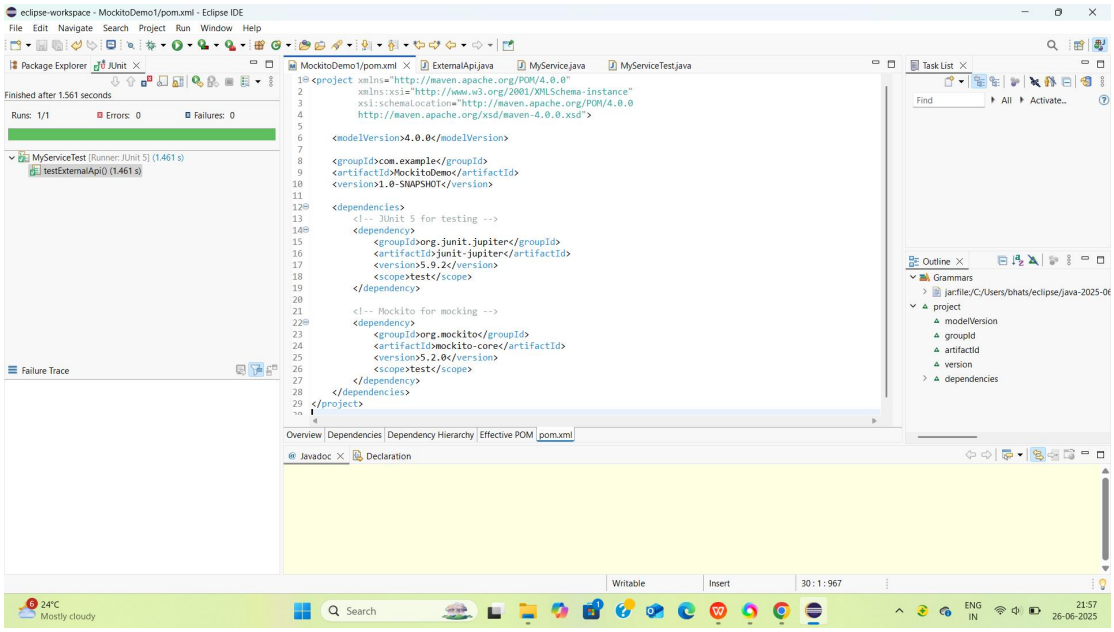

eclipse-workspace - MockitoDemo1/src/main/java/com/example/ExternalApi.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 1.561 seconds
Runs: 1/1 Errors: 0 Failures: 0
MyServiceTest [Runner: JUnit 5] (1.461 s)
testExternalApi() (1.461 s)
Failure Trace
MockitoDemo1/pom.xml ExternalApi.java MyService.java MyServiceTest.java
1 package com.example;
2
3 public interface ExternalApi {
4 String getData();
5
6 }
7
Task List
Find All Activate...
Outline
com.example
ExternalApi
getData() : String
Javadoc Declaration
Writable Smart Insert 4 : 23 : 75
24°C Mostly cloudy
Search
ENG IN 21:57 26-06-2025

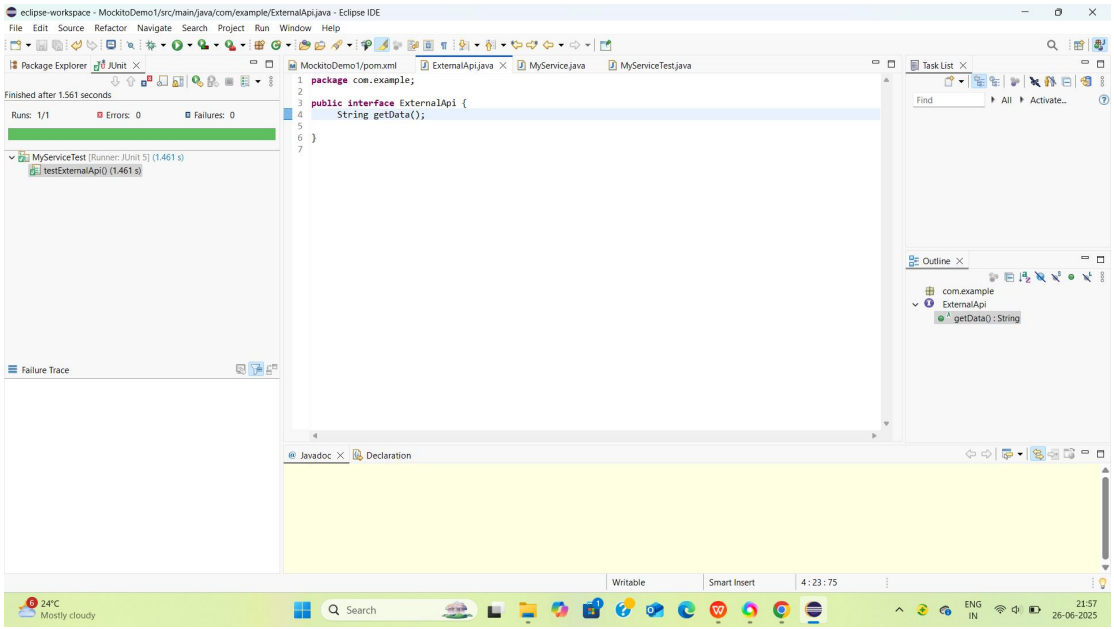

eclipse-workspace - MockitoDemo1/src/main/java/com/example/MyService.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 1.561 seconds
Runs: 1/1 Errors: 0 Failures: 0
MyServiceTest [Runner: JUnit 5] (1.461 s)
testExternalApi() (1.461 s)
Failure Trace
MockitoDemo1/pom.xml ExternalApi.java MyService.java MyServiceTest.java
1 package com.example;
2
3 public class MyService {
4 private ExternalApi api;
5
6 public MyService(ExternalApi api) {
7 this.api = api;
8 }
9
10 public String fetchData() {
11 return api.getData();
12 }
13 }
14
Task List
Find All Activate...
Outline
com.example
MyService
Javadoc Declaration
Writable Smart Insert 14 : 1 : 231
24°C Mostly cloudy
Search
ENG IN 21:57 26-06-2025

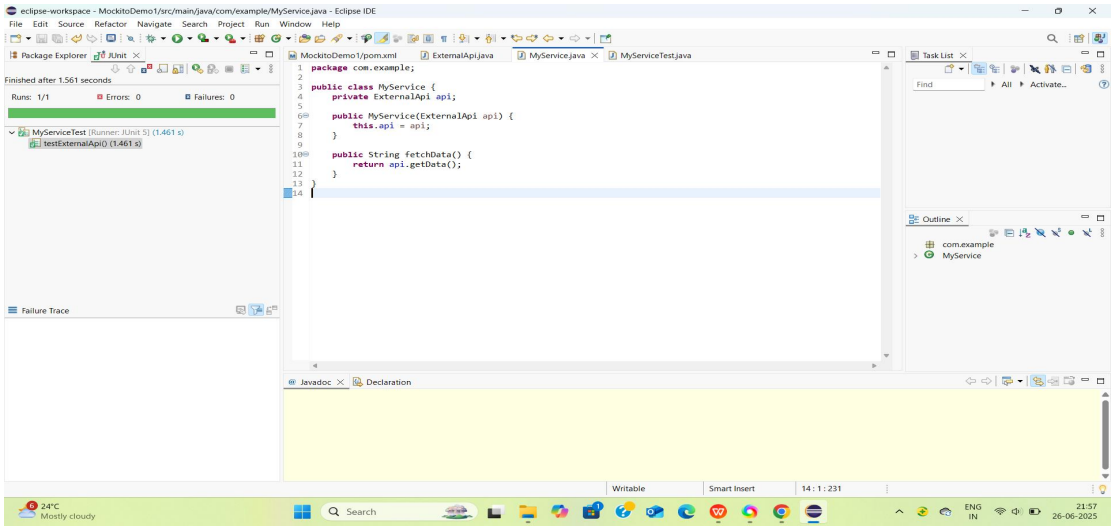

eclipse-workspace - MockitoDemo1/src/test/java/com/example/MyServiceTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 1.561 seconds
Runs: 1/1 Errors: 0 Failures: 0
MyServiceTest [Runner: JUnit 5] (1.461 s)
testExternalApi() (1.461 s)
Failure Trace
MockitoDemo1/pom.xml ExternalApi.java MyService.java MyServiceTest.java
1 package com.example;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4 import static org.mockito.Mockito.*;
5
6 import org.junit.jupiter.api.Test;
7
8 public class MyServiceTest {
9
10 @Test
11 public void testExternalApi() {
12 // Step 1: Create mock
13 ExternalApi mockApi = mock(ExternalApi.class);
14
15 // Step 2: Stub method
16 when(mockApi.getData()).thenReturn("Mock Data");
17
18 // Step 3: Use mock in service
19 MyService service = new MyService(mockApi);
20 String result = service.fetchData();
21
22 // Step 4: Assert
23 assertEquals("Mock Data", result);
24 }
25 }
26
27
Task List
Find All Activate...
Outline
com.example
MyServiceTest
testExternalApi() : void
Javadoc Declaration
void com.example.MyServiceTest.testExternalApi ()
@Test
Writable Smart Insert 27 : 1 : 650
24°C Mostly cloudy
Search
ENG IN 21:57 26-06-2025

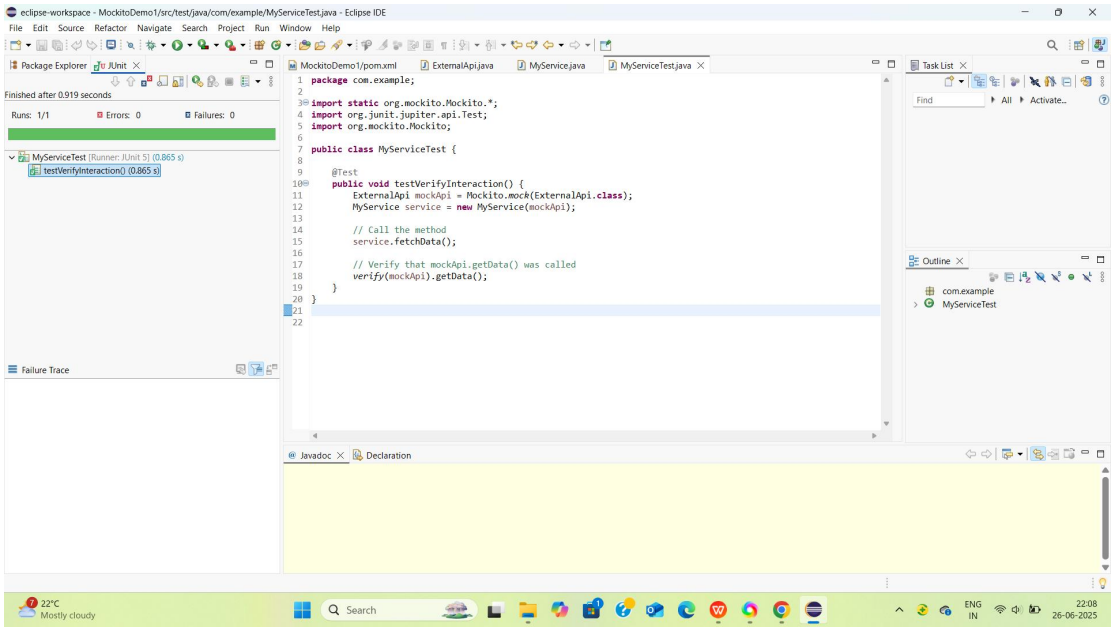
Exercise 2: Verifying Interactions

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.






eclipse-workspace - MockitoDemo1/src/main/java/com/example/MyService.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 1.561 seconds
Runs: 1/1
Errors: 0
Failures: 0
MyServiceTest [Runner: JUnit 5] (1.461 s)
testExternalApi() (1.461 s)
Failure Trace
MockitoDemo1/pom.xml
ExternalApi.java
MyService.java
MyServiceTest.java
1 package com.example;
3 public class MyService {
4 private ExternalApi api;
6 public MyService(ExternalApi api) {
7 this.api = api;
8 }
10 public String fetchData() {
11 return api.getData();
12 }
13 }
Task List
Find
All
Activate...
Outline
com.example
MyService
Javadoc
Declaration
Writable
Smart Insert
14 : 1 : 231
24°C
Mostly cloudy
Search
ENG
IN
21:57
26-06-2025


eclipse-workspace - MockitoDemo1/src/test/java/com/example/MyServiceTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.919 seconds
Runs: 1/1
Errors: 0
Failures: 0
MyServiceTest [Runner: JUnit 5] (0.865 s)
testVerifyInteraction() (0.865 s)
Failure Trace
MockitoDemo1/pom.xml
ExternalApi.java
MyService.java
MyServiceTest.java
1 package com.example;
3 import static org.mockito.Mockito.*;
4 import org.junit.jupiter.api.Test;
5 import org.mockito.Mockito;
7 public class MyServiceTest {
9 @Test
10 public void testVerifyInteraction() {
11 ExternalApi mockApi = Mockito.mock(ExternalApi.class);
12 MyService service = new MyService(mockApi);
14 // Call the method
15 service.fetchData();
17 // Verify that mockApi.getData() was called
18 verify(mockApi).getData();
19 }
20 }
Task List
Find
All
Activate...
Outline
com.example
MyServiceTest
Javadoc
Declaration
22°C
Mostly cloudy
Search
ENG
IN
22:08
26-06-2025

Logging using SLF4J

Exercise 1: Logging Error Messages and Warning Levels

Task: Write a Java application that demonstrates logging error messages and warning levels using SLF4J.