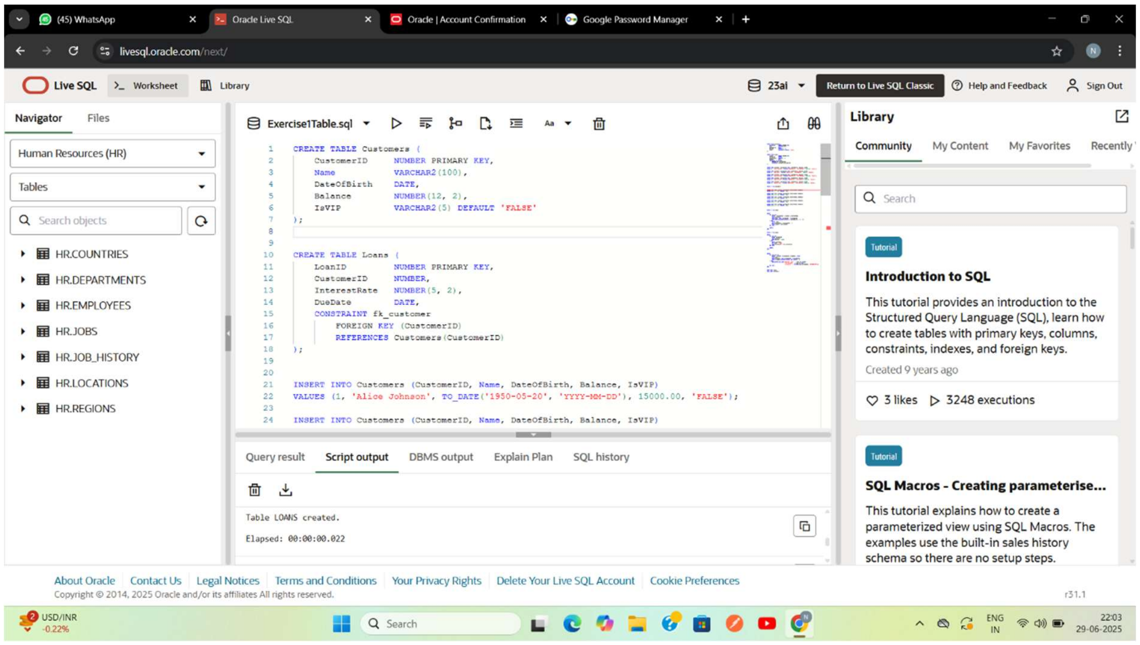
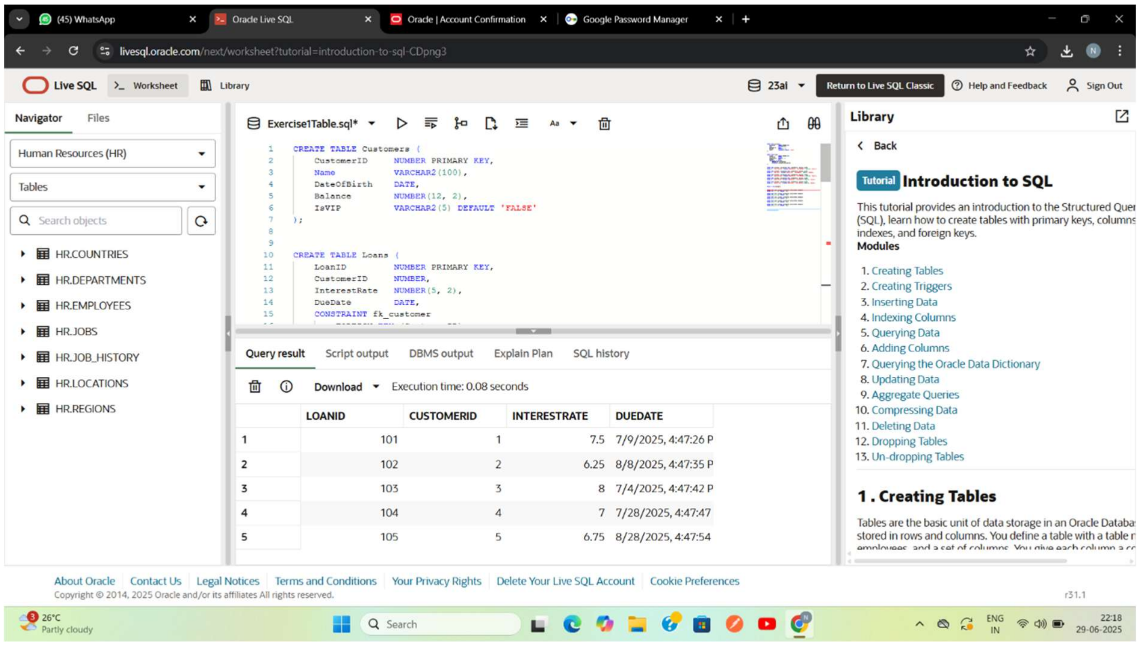
PL/SQL PROGRAMMING:

Exercise 1:Control Structures



```
CREATE TABLE Customers (
    CustomerID      NUMBER PRIMARY KEY,
    Name            VARCHAR2(100),
    DateOfBirth     DATE,
    Balance         NUMBER(12, 2),
    IsVIP           VARCHAR2(5) DEFAULT 'FALSE'
);


CREATE TABLE Loans (
    LoanID          NUMBER PRIMARY KEY,
    CustomerID      NUMBER,
    InterestRate    NUMBER(5, 2),
    DueDate         DATE,
    CONSTRAINT fk_customer
        FOREIGN KEY (CustomerID)
        REFERENCES Customers(CustomerID)
);


INSERT INTO Customers (CustomerID, Name, DateOfBirth, Balance, IsVIP)
VALUES (1, 'Alice Johnson', TO_DATE('1950-05-20', 'YYYY-MM-DD'), 15000.00, 'FALSE');

INSERT INTO Customers (CustomerID, Name, DateOfBirth, Balance, IsVIP)
```

```
Table LOANS created.

Elapsed: 00:00:00.022
```

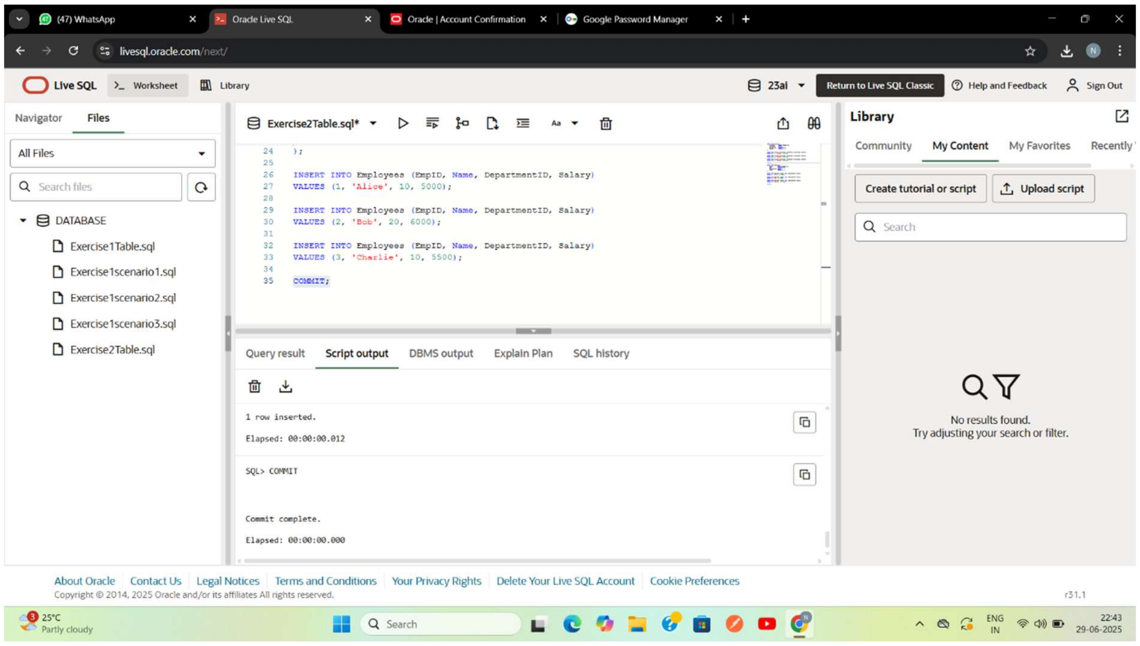


```
CREATE TABLE Customers (
    CustomerID      NUMBER PRIMARY KEY,
    Name            VARCHAR2(100),
    DateOfBirth     DATE,
    Balance         NUMBER(12, 2),
    IsVIP           VARCHAR2(5) DEFAULT 'FALSE'
);


CREATE TABLE Loans (
    LoanID          NUMBER PRIMARY KEY,
    CustomerID      NUMBER,
    InterestRate    NUMBER(5, 2),
    DueDate         DATE,
    CONSTRAINT fk_customer
```

Execution time: 0.08 seconds

	LOANID	CUSTOMERID	INTERESTRATE	DUEDATE
1	101	1	7.5	7/9/2025, 4:47:26 P
2	102	2	6.25	8/8/2025, 4:47:35 P
3	103	3	8	7/4/2025, 4:47:42 P
4	104	4	7	7/28/2025, 4:47:47
5	105	5	6.75	8/28/2025, 4:47:54

Exercise 2:Stored procedures

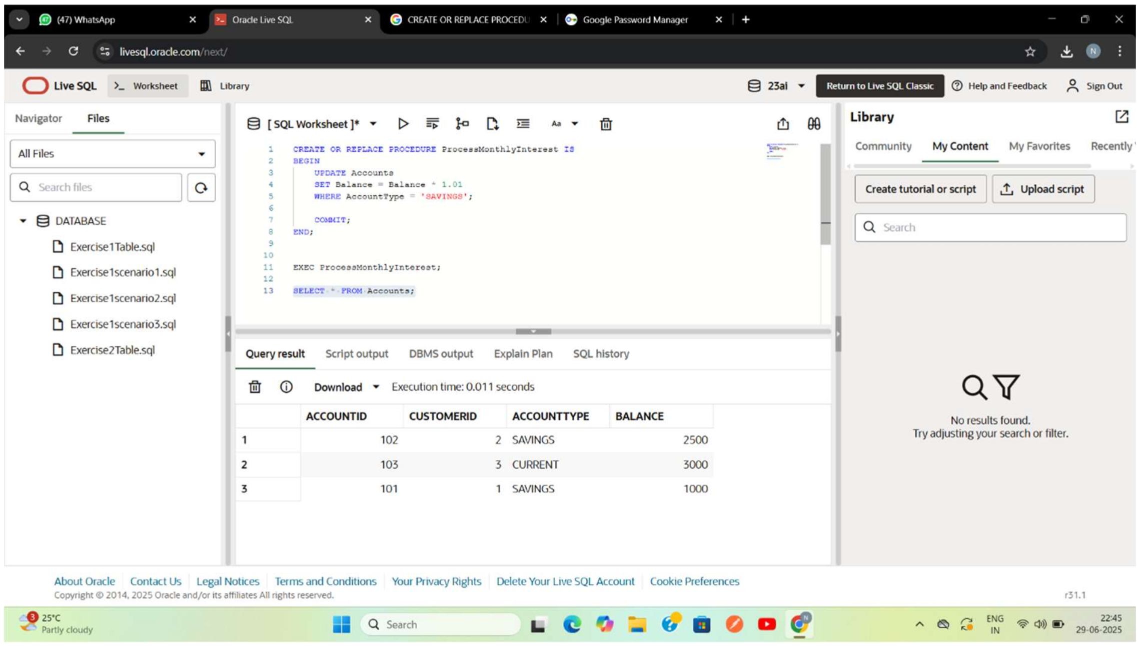


Exercise2Table.sql*

```
);

INSERT INTO Employees (EmpID, Name, DepartmentID, Salary)
VALUES (1, 'Alice', 10, 5000);

INSERT INTO Employees (EmpID, Name, DepartmentID, Salary)
VALUES (2, 'Bob', 20, 6000);

INSERT INTO Employees (EmpID, Name, DepartmentID, Salary)
VALUES (3, 'Charlie', 10, 5500);

COMMIT;
```

Script output

```
1 row inserted.

Elapsed: 00:00:00.012

SQL> COMMIT


Commit complete.

Elapsed: 00:00:00.000
```



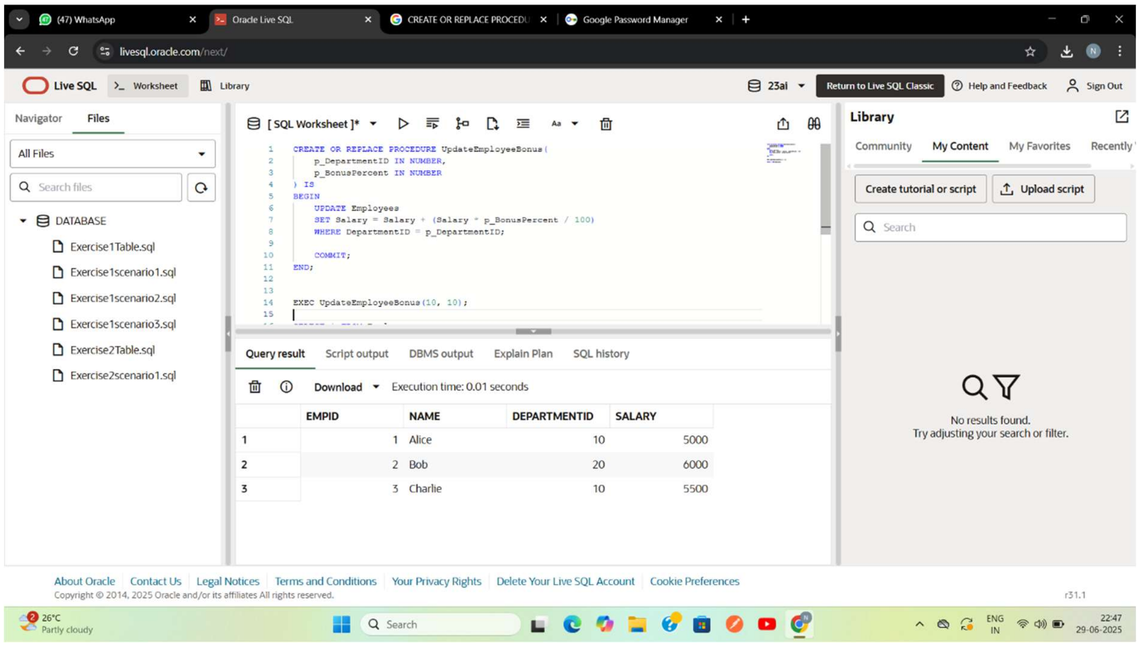
[SQL Worksheet]*

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
    UPDATE Accounts
    SET Balance = Balance * 1.01
    WHERE AccountType = 'SAVINGS';

    COMMIT;
END;


EXEC ProcessMonthlyInterest;

SELECT * FROM Accounts;
```

Query result

Execution time: 0.011 seconds

	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE
1	102	2	SAVINGS	2500
2	103	3	CURRENT	3000
3	101	1	SAVINGS	1000



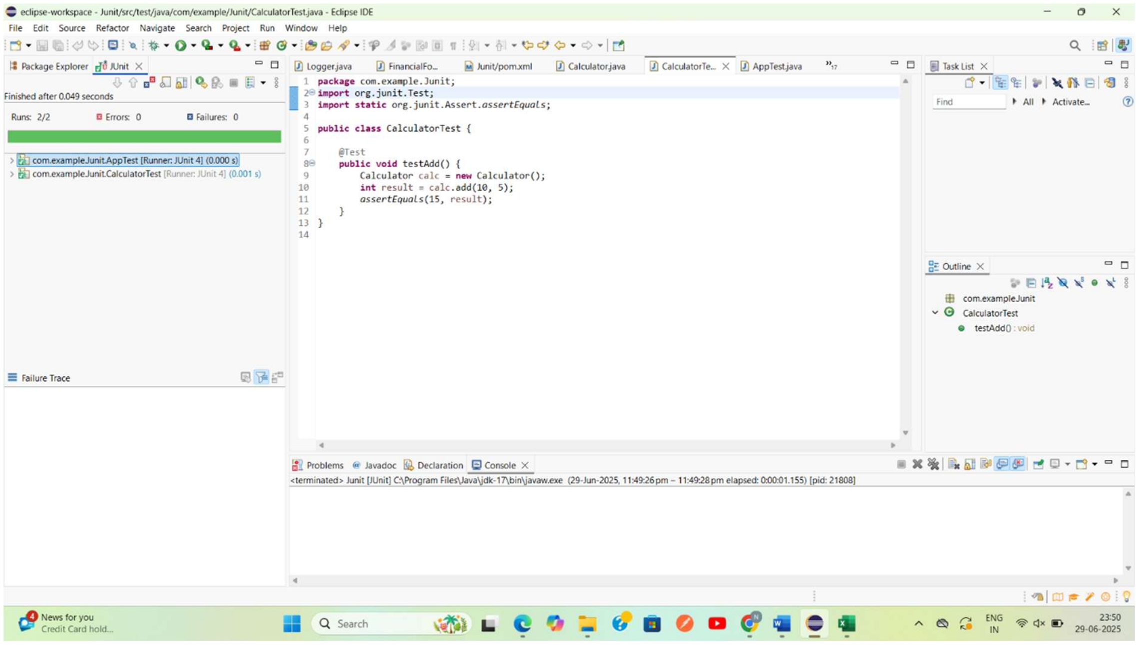
```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(
    p_DepartmentID IN NUMBER,
    p_BonusPercent IN NUMBER
) IS
BEGIN
    UPDATE Employees
    SET Salary = Salary + (Salary * p_BonusPercent / 100)
    WHERE DepartmentID = p_DepartmentID;

    COMMIT;
END;


EXEC UpdateEmployeeBonus(10, 10);
```

Execution time: 0.01 seconds

	EMPID	NAME	DEPARTMENTID	SALARY
1	1	Alice	10	5000
2	2	Bob	20	6000
3	3	Charlie	10	5500

TDD USING JUNIT5 AND MOCKITO

Exercise 1:Setting up Junit



```java
package com.example.Junit;
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class CalculatorTest {

    @Test
    public void testAdd() {
        Calculator calc = new Calculator();
        int result = calc.add(10, 5);
        assertEquals(15, result);
    }
}
```

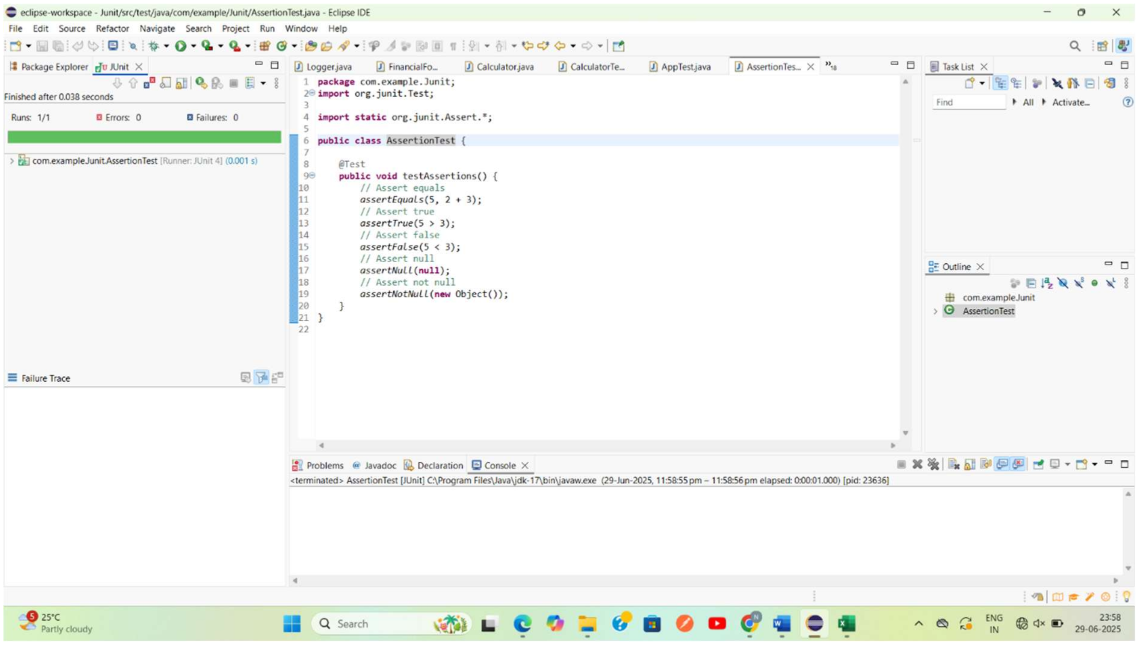
Finished after 0.049 seconds

Runs: 2/2 Errors: 0 Failures: 0

com.example.Junit.AppTest [Runner: JUnit 4] (0.000 s)
com.example.Junit.CalculatorTest [Runner: JUnit 4] (0.001 s)

```
<terminated> Junit [JUnit] C:\Program Files\Java\jdk-17\bin\javaw.exe (29-Jun-2025, 11:49:26 pm – 11:49:28 pm elapsed: 0:00:01.155) [pid: 21808]
```

Exercise 3:Assertions in Junit



```
package com.example.Junit;
import org.junit.Test;

import static org.junit.Assert.*;

public class AssertionTest {

    @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());
    }
}
```

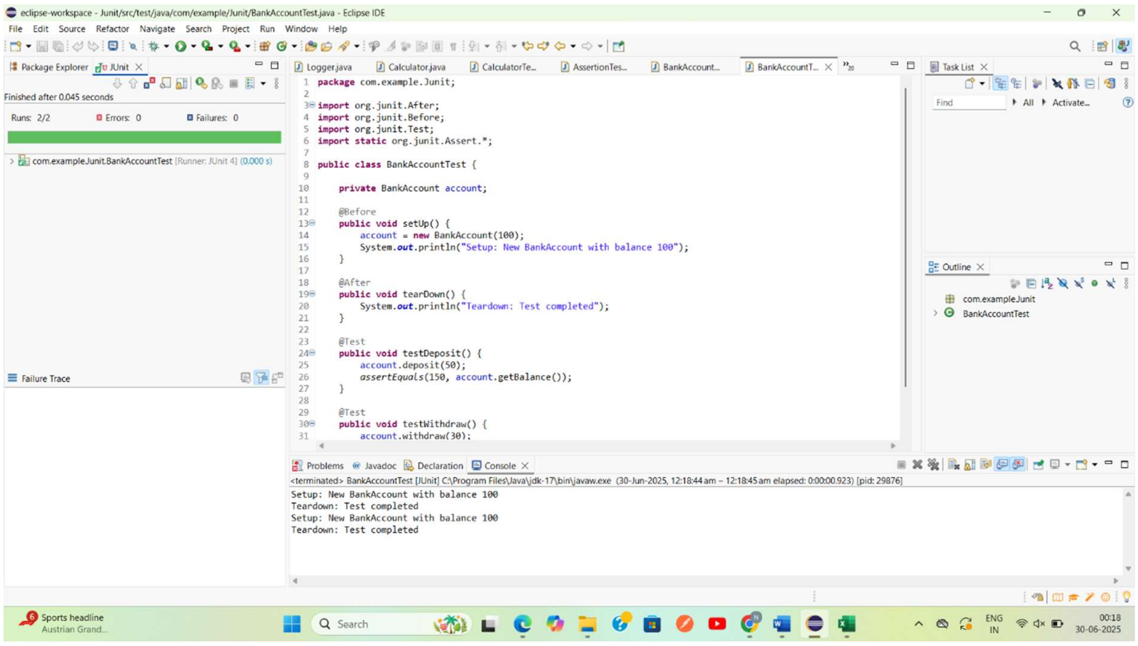
Finished after 0.038 seconds

Runs: 1/1 Errors: 0 Failures: 0

com.example.Junit.AssertionTest [Runner: JUnit 4] (0.001 s)

```
<terminated> AssertionTest [JUnit] C:\Program Files\Java\jdk-17\bin\javaw.exe (29-Jun-2025, 11:58:55 pm – 11:58:56 pm elapsed: 0:00:01.000) [pid: 23636]
```

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



```
package com.example.Junit;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

public class BankAccountTest {

    private BankAccount account;

    @Before
    public void setUp() {
        account = new BankAccount(100);
        System.out.println("Setup: New BankAccount with balance 100");
    }

    @After
    public void tearDown() {
        System.out.println("Teardown: Test completed");
    }

    @Test
    public void testDeposit() {
        account.deposit(50);
        assertEquals(150, account.getBalance());
    }

    @Test
    public void testWithdraw() {
        account.withdraw(30);
```

Finished after 0.045 seconds

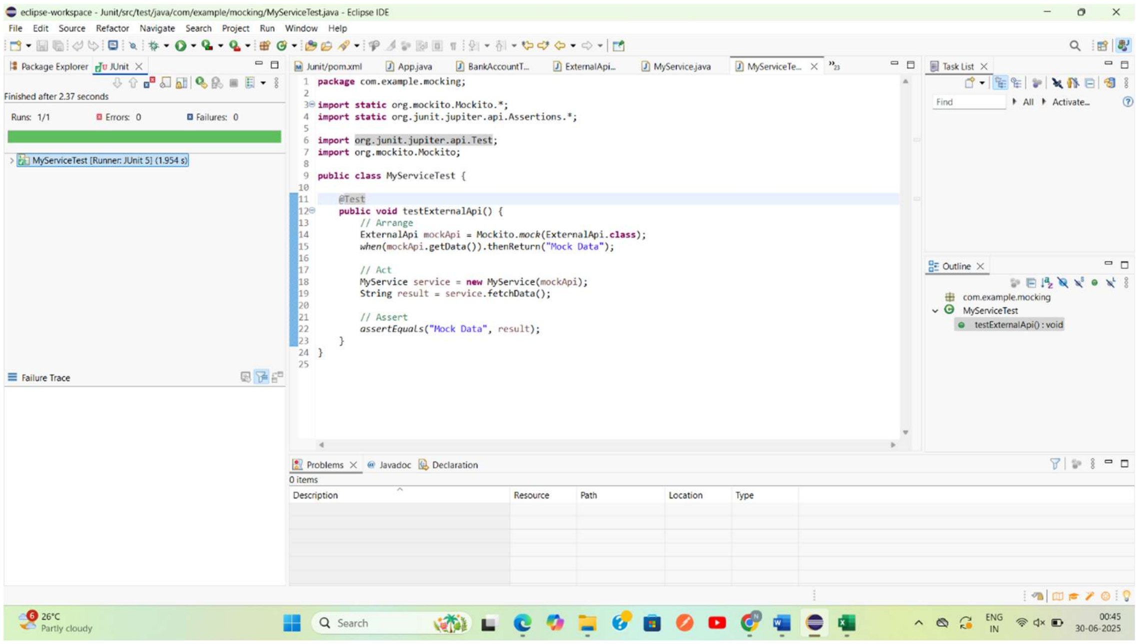
Runs: 2/2 Errors: 0 Failures: 0

com.example.Junit.BankAccountTest [Runner: JUnit 4] (0.000 s)

```
<terminated> BankAccountTest [JUnit] C:\Program Files\Java\jdk-17\bin\javaw.exe (30-Jun-2025, 12:18:44 am – 12:18:45 am elapsed: 0:00:00.923) [pid: 29876]
Setup: New BankAccount with balance 100
Teardown: Test completed
Setup: New BankAccount with balance 100
Teardown: Test completed
```

MOCKITO EXERCISES:

Exercise 1:MOCKING AND STUBBING



eclipse-workspace - Junit/src/test/java/com/example/mocking/MyServiceTest.java - Eclipse IDE

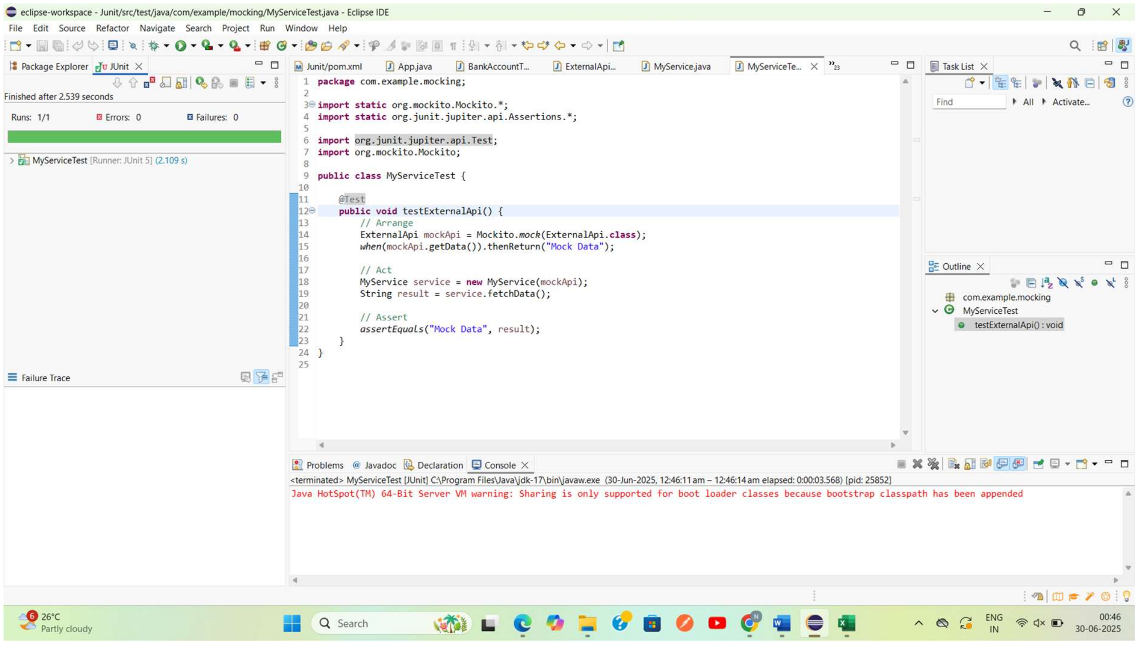
Finished after 2.37 seconds

Runs: 1/1 Errors: 0 Failures: 0

MyServiceTest [Runner: JUnit 5] (1.954 s)

```java
package com.example.mocking;

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() {
        // Arrange
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);
        when(mockApi.getData()).thenReturn("Mock Data");

        // Act
        MyService service = new MyService(mockApi);
        String result = service.fetchData();

        // Assert
        assertEquals("Mock Data", result);
    }
}
```

Problems: 0 items



eclipse-workspace - Junit/src/test/java/com/example/mocking/MyServiceTest.java - Eclipse IDE

Finished after 2.539 seconds

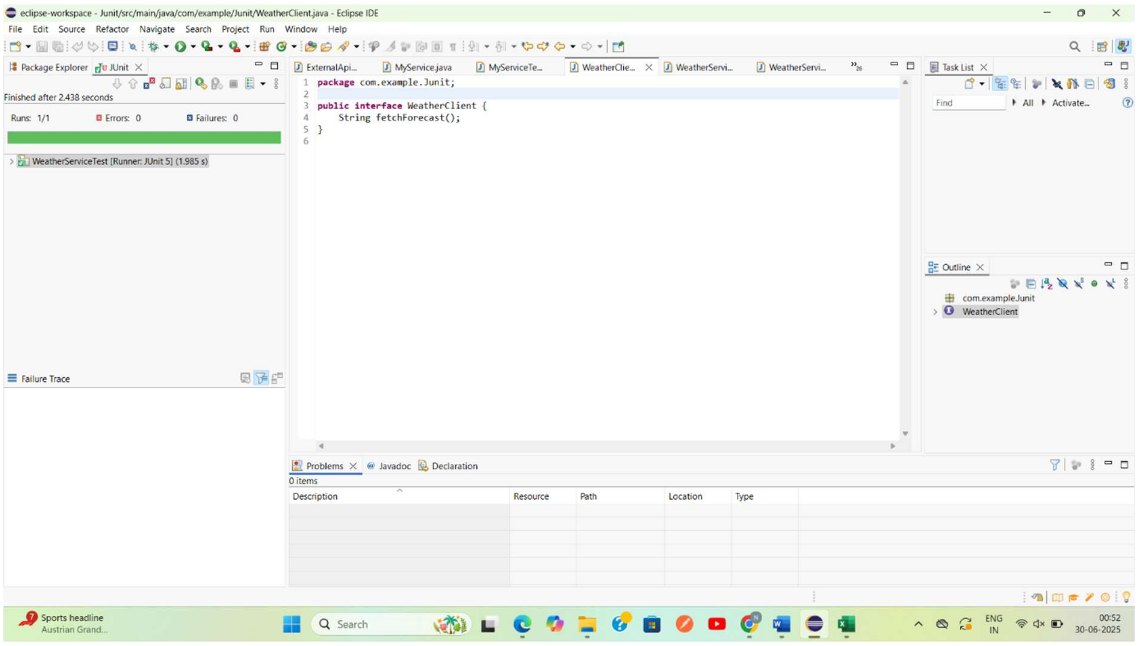
Runs: 1/1 Errors: 0 Failures: 0

MyServiceTest [Runner: JUnit 5] (2.109 s)

```java
package com.example.mocking;

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() {
        // Arrange
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);
        when(mockApi.getData()).thenReturn("Mock Data");

        // Act
        MyService service = new MyService(mockApi);
        String result = service.fetchData();

        // Assert
        assertEquals("Mock Data", result);
    }
}
```

```
<terminated> MyServiceTest [JUnit] C:\Program Files\Java\jdk-17\bin\javaw.exe  (30-Jun-2025, 12:46:11 am – 12:46:14 am elapsed: 0:00:03.568) [pid: 25852]
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
```

Exercise 2:Veryfying Interactions



eclipse-workspace - Junit/src/main/java/com/example/Junit/WeatherClient.java - Eclipse IDE

Finished after 2.438 seconds

Runs: 1/1 Errors: 0 Failures: 0

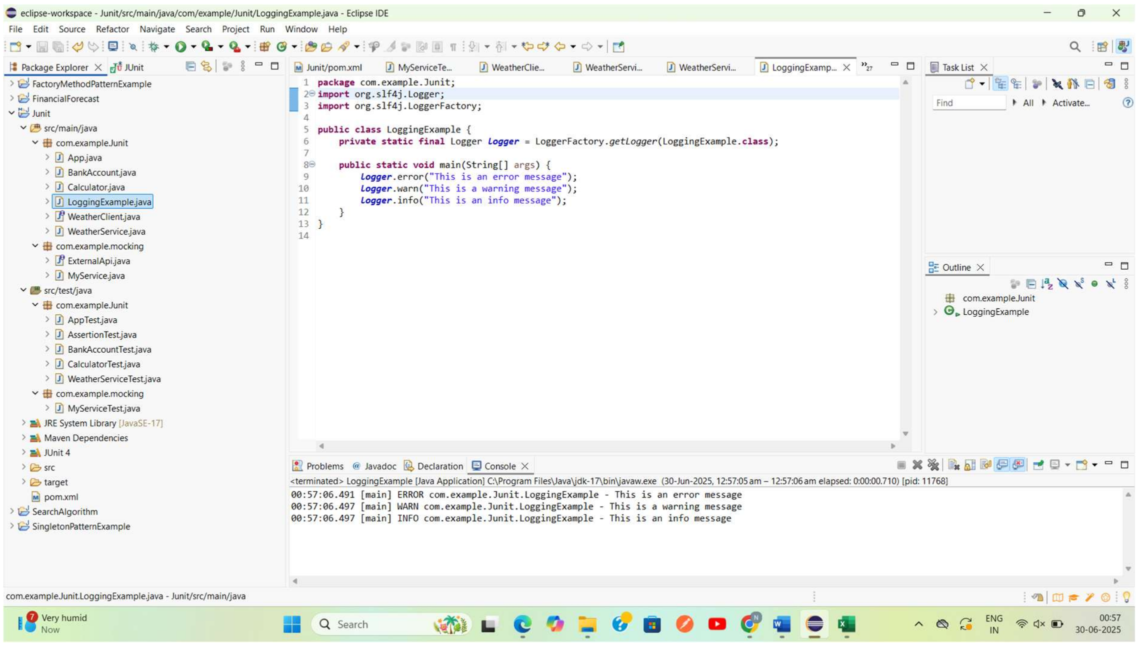
WeatherServiceTest [Runner: JUnit 5] (1.985 s)

```java
package com.example.Junit;

public interface WeatherClient {
    String fetchForecast();
}
```

Problems: 0 items

SL4J LOGGING EXERCISES:

Exercise 1:Logging error messages with warning Levels



eclipse-workspace - Junit/src/main/java/com/example/Junit/LoggingExample.java - Eclipse IDE

```java
package com.example.Junit;
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");
        Logger.info("This is an info message");
    }
}
```

```
<terminated> LoggingExample [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (30-Jun-2025, 12:57:05 am – 12:57:06 am elapsed: 0:00:00.710) [pid: 11768]
00:57:06.491 [main] ERROR com.example.Junit.LoggingExample - This is an error message
00:57:06.497 [main] WARN com.example.Junit.LoggingExample - This is a warning message
00:57:06.497 [main] INFO com.example.Junit.LoggingExample - This is an info message
```