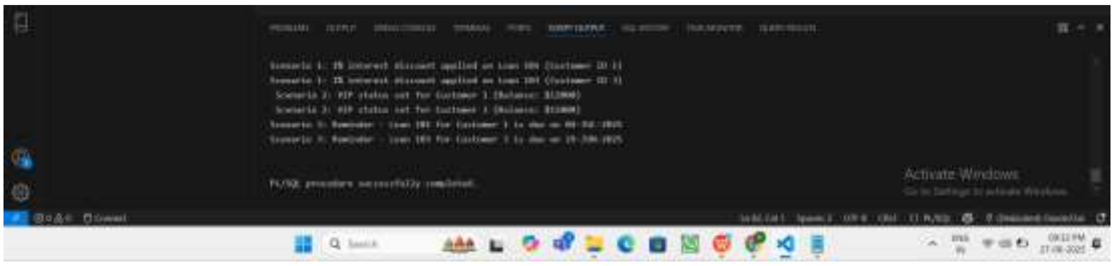
PL_SQL_Exercise

Excerise 1:Control Structures

**CODE:**

```
SET SERVEROUTPUT ON;

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE loans';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE customers';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE customers (

  cust_id   NUMBER PRIMARY KEY,

  age       NUMBER,

  balance   NUMBER,

  vip_flag  VARCHAR2(5)

);

CREATE TABLE loans (
  loan_id   NUMBER PRIMARY KEY,
  cust_id   NUMBER,
  int_rate  NUMBER,
  due_on    DATE,
  FOREIGN KEY (cust_id) REFERENCES customers(cust_id)
);
INSERT INTO customers VALUES (1, 65, 12000, 'FALSE');
INSERT INTO customers VALUES (2, 45, 8000, 'FALSE');
INSERT INTO customers VALUES (3, 70, 15000, 'FALSE');

INSERT INTO loans VALUES (101, 1, 10, TO_DATE('04-JUL-2025','DD-MON-YYYY'));
INSERT INTO loans VALUES (102, 2, 9,  TO_DATE('01-SEP-2025','DD-MON-YYYY'));
INSERT INTO loans VALUES (103, 3, 8,  TO_DATE('29-JUN-2025','DD-MON-YYYY'));
COMMIT;
BEGIN
  FOR loan_rec IN (
    SELECT l.loan_id, l.cust_id, l.int_rate
    FROM loans l
    JOIN customers c ON l.cust_id = c.cust_id
    WHERE c.age > 60
  )
  LOOP
    UPDATE loans
    SET int_rate = int_rate - 1
    WHERE loan_id = loan_rec.loan_id;

    DBMS_OUTPUT.PUT_LINE(
      'Scenario 1: 1% interest discount applied on Loan ' || loan_rec.loan_id ||
      ' (Customer ID ' || loan_rec.cust_id || ')'
    );
  END LOOP;
  FOR cust_rec IN (
    SELECT cust_id, balance FROM customers
    WHERE balance > 10000
  )
  LOOP
    UPDATE customers
    SET vip_flag = 'TRUE'
    WHERE cust_id = cust_rec.cust_id;

    DBMS_OUTPUT.PUT_LINE(
      ' Scenario 2: VIP status set for Customer ' || cust_rec.cust_id ||
      ' (Balance: $' || cust_rec.balance || ')'
    );
  END LOOP;
  FOR due_rec IN (
    SELECT loan_id, cust_id, due_on
    FROM loans
    WHERE due_on BETWEEN SYSDATE AND SYSDATE + 30
  )
  LOOP
    DBMS_OUTPUT.PUT_LINE(
      'Scenario 3: Reminder - Loan ' || due_rec.loan_id ||
      ' for Customer ' || due_rec.cust_id ||
      ' is due on ' || TO_CHAR(due_rec.due_on, 'DD-MON-YYYY')
    );
  END LOOP;
  COMMIT;
END;
/
```

**OUTPUT:**



```
Scenario 1: 1% interest discount applied on Loan 101 (Customer ID 1)

Scenario 1: 1% interest discount applied on Loan 103 (Customer ID 3)

 Scenario 2: VIP status set for Customer 1 (Balance: $12000)

 Scenario 2: VIP status set for Customer 3 (Balance: $15000)

Scenario 3: Reminder - Loan 101 for Customer 1 is due on 04-JUL-2025

Scenario 3: Reminder - Loan 103 for Customer 3 is due on 29-JUN-2025


PL/SQL procedure successfully completed.
```

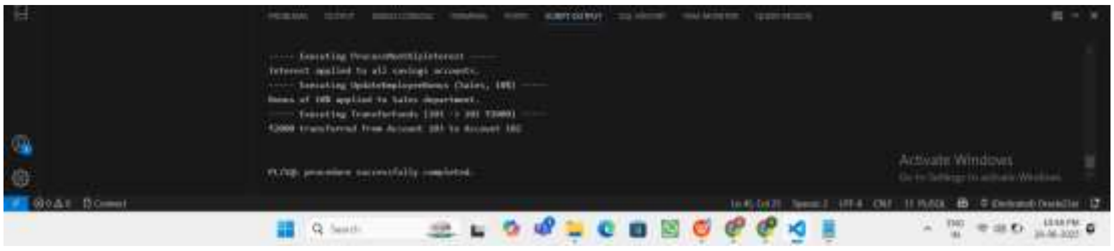
Exercise:3 Stored Procedures

**CODE:**

```
SET SERVEROUTPUT ON;

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE accounts';

EXCEPTION WHEN OTHERS THEN NULL;
```

```
END;
/
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE employees';
EXCEPTION WHEN OTHERS THEN NULL;
END;
/
CREATE TABLE accounts (
  account_id    NUMBER PRIMARY KEY,
  customer_id   NUMBER,
  balance       NUMBER,
  account_type  VARCHAR2(20)
);
CREATE TABLE employees (
  emp_id     NUMBER PRIMARY KEY,
  name       VARCHAR2(50),
  department VARCHAR2(50),
  salary     NUMBER
);
INSERT INTO accounts VALUES (101, 1, 10000, 'SAVINGS');
INSERT INTO accounts VALUES (102, 2, 15000, 'CURRENT');
INSERT INTO accounts VALUES (103, 3, 20000, 'SAVINGS');
INSERT INTO employees VALUES (1, 'Ravi',   'Sales',    40000);
INSERT INTO employees VALUES (2, 'Sneha',  'Finance',  45000);
INSERT INTO employees VALUES (3, 'Ajith',  'Sales',    42000);
COMMIT;
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
  UPDATE accounts
  SET balance = balance + (balance * 0.01)
  WHERE UPPER(account_type) = 'SAVINGS';

  DBMS_OUTPUT.PUT_LINE('Interest applied to all savings accounts.');
  COMMIT;
END;
/
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
  p_dept       IN VARCHAR2,
  p_bonus_pct  IN NUMBER
) IS
BEGIN
  UPDATE employees
  SET salary = salary + (salary * p_bonus_pct / 100)
  WHERE LOWER(department) = LOWER(p_dept);
  DBMS_OUTPUT.PUT_LINE('Bonus of ' || p_bonus_pct || '% applied to ' ||
p_dept || ' department.');
  COMMIT;
END;
/
CREATE OR REPLACE PROCEDURE TransferFunds (
  p_from_account IN NUMBER,
  p_to_account   IN NUMBER,
  p_amount       IN NUMBER
) IS
  v_balance NUMBER;
BEGIN
  SELECT balance INTO v_balance
  FROM accounts
  WHERE account_id = p_from_account;
  IF v_balance < p_amount THEN
    RAISE_APPLICATION_ERROR(-20001, 'Not enough balance in source
account.');
  END IF;
  UPDATE accounts
  SET balance = balance - p_amount
  WHERE account_id = p_from_account;
  UPDATE accounts
  SET balance = balance + p_amount
  WHERE account_id = p_to_account;
  DBMS_OUTPUT.PUT_LINE('₹' || p_amount || ' transferred from Account ' ||
p_from_account || ' to Account ' || p_to_account);
  COMMIT;
END;
/
BEGIN
  DBMS_OUTPUT.PUT_LINE('----- Executing ProcessMonthlyInterest -----');
  ProcessMonthlyInterest;

  DBMS_OUTPUT.PUT_LINE('----- Executing UpdateEmployeeBonus (Sales,
10%) -----');
  UpdateEmployeeBonus('Sales', 10);
  DBMS_OUTPUT.PUT_LINE('----- Executing TransferFunds (103 -> 102
₹2000) -----');
  TransferFunds(103, 102, 2000);
END;
```

**OUTPUT**



```
----- Executing ProcessMonthlyInterest -----
Interest applied to all savings accounts.
----- Executing UpdateEmployeeBonus (Sales, 10%) -----
Bonus of 10% applied to Sales department.
----- Executing TransferFunds (101 -> 102 $2000) -----
$2000 transferred from Account 101 to Account 102.

PL/SQL procedure successfully completed.
```

Junit Testing Exercise

Exercise 1: Setting Up Junit

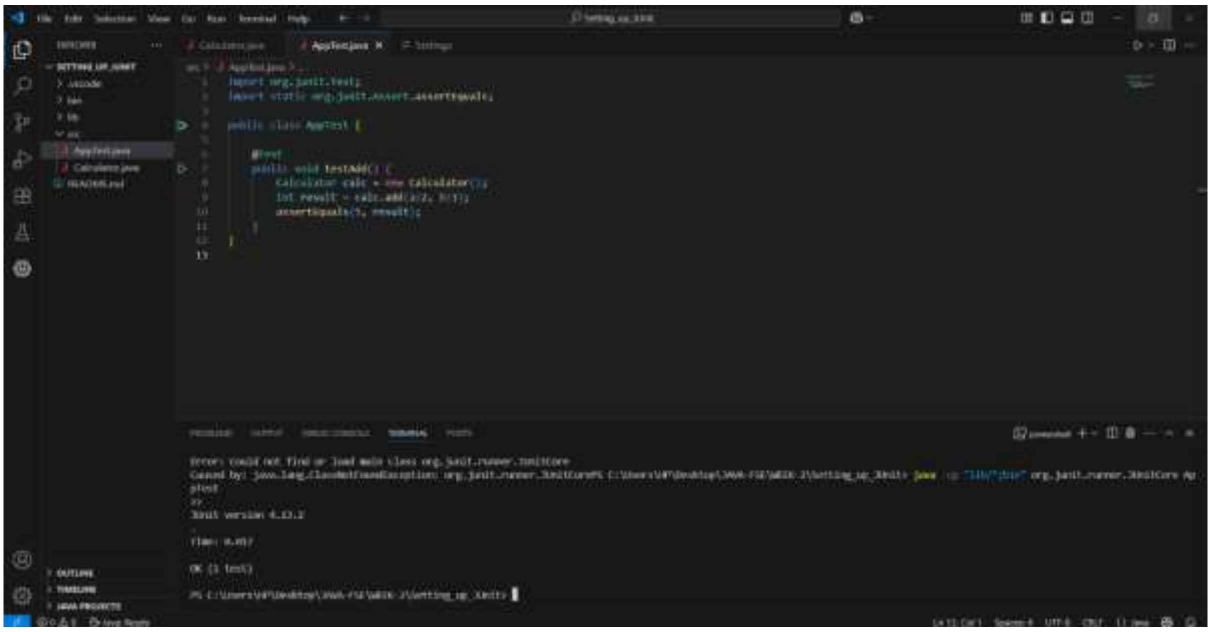
Calculator.java

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

AppTest.java

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class AppTest {
    @Test
    public void testAdd() {
        Calculator calc = new Calculator();
        int result = calc.add(2, 3);
        assertEquals(5, result);
    }
}
```

OUTPUT:



```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class AppTest {

    @Test
    public void testAdd() {
        Calculator calc = new Calculator();
        int result = calc.add(a:2, b:3);
        assertEquals(5, result);
    }
}
```

```
Error: Could not find or load main class org.junit.runner.JUnitCore
Caused by: java.lang.ClassNotFoundException: org.junit.runner.JUnitCorePS C:\Users\HP\Desktop\JAVA-FSE\WEEK-2\setting_up_JUnit> java -cp "lib/*;bin" org.junit.runner.JUnitCore Ap
pTest
>>
JUnit version 4.13.2
.
Time: 0.057

OK (1 test)

PS C:\Users\HP\Desktop\JAVA-FSE\WEEK-2\setting_up_JUnit> 
```

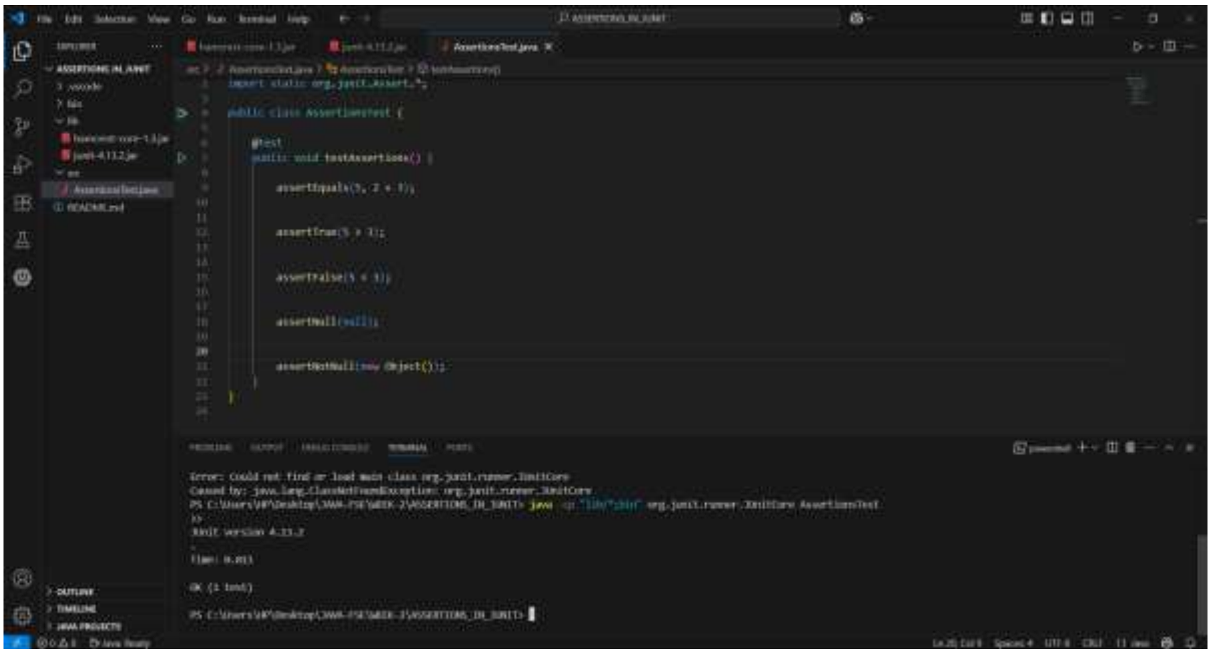
EXERCISE:3

ASSERTIONS IN JUNIT

AssertionsTest.java

```
import org.junit.Test;

import static org.junit.Assert.*;

public class AssertionsTest {

    @Test

    public void testAssertions() {

        assertEquals(5, 2 + 3);

        assertTrue(5 > 3);

        assertFalse(5 < 3);

        assertNull(null);

        assertNotNull(new Object());

    }

}
```

OUTPUT:



```java
import static org.junit.Assert.*;

public class AssertionsTest {

    @Test
    public void testAssertions() {

        assertEquals(5, 2 + 3);

        assertTrue(5 > 3);

        assertFalse(5 < 3);

        assertNull(null);

        assertNotNull(new Object());
    }
}
```

```
Error: Could not find or load main class org.junit.runner.JUnitCore
Caused by: java.lang.ClassNotFoundException: org.junit.runner.JUnitCore
PS C:\Users\HP\Desktop\JAVA-FSE\WEEK-2\ASSERTIONS_IN_JUNIT> java -cp "lib/*;bin" org.junit.runner.JUnitCore AssertionsTest
>>
JUnit version 4.13.2
.
Time: 0.011

OK (1 test)

PS C:\Users\HP\Desktop\JAVA-FSE\WEEK-2\ASSERTIONS_IN_JUNIT>
```

EXERCISE:4 AAA_PATTERN

Calculator.java

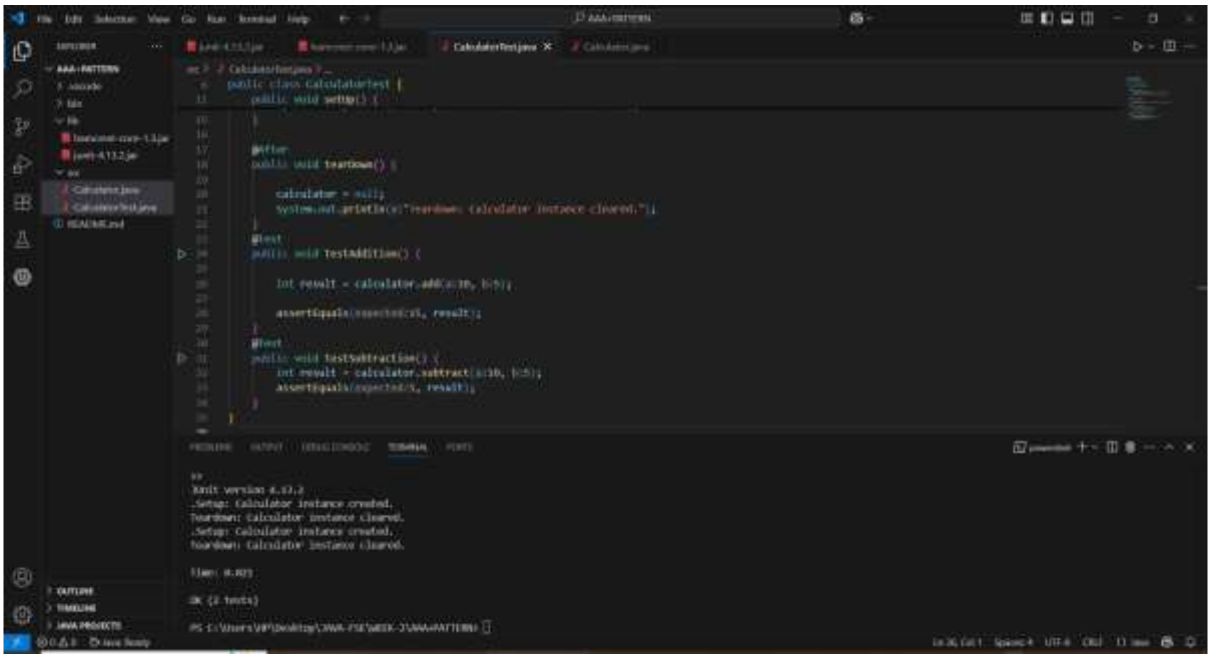
```java
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
    public int subtract(int a, int b) {
        return a - b;
    }
}
```

CalculatorTest.java

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

```
public class CalculatorTest {
    private Calculator calculator;
    @Before
    public void setUp() {
        calculator = new Calculator();
        System.out.println("Setup: Calculator instance created.");
    }
    @After
    public void tearDown() {
        calculator = null;
        System.out.println("Teardown: Calculator instance cleared.");
    }
    @Test
    public void testAddition() {
        int result = calculator.add(10, 5);
        assertEquals(15, result);
    }
    @Test
    public void testSubtraction() {
        int result = calculator.subtract(10, 5);
        assertEquals(5, result);
    }
}
```

OUTPUT:



```
public class CalculatorTest {
    public void setUp() {
    }

    @After
    public void tearDown() {

        calculator = null;
        System.out.println(x:"Teardown: Calculator instance cleared.");
    }
    @Test
    public void testAddition() {

        int result = calculator.add(a:10, b:5);

        assertEquals(expected:15, result);
    }
    @Test
    public void testSubtraction() {
        int result = calculator.subtract(a:10, b:5);
        assertEquals(expected:5, result);
    }
}
```

```
JUnit version 4.13.2
.Setup: Calculator instance created.
Teardown: Calculator instance cleared.
.Setup: Calculator instance created.
Teardown: Calculator instance cleared.

Time: 0.023

OK (2 tests)

PS C:\Users\HP\Desktop\JAVA-FSE\WEEK-2\AAA-PATTERN>
```

Mockito Hands-On Exercises

Exercise 1: Mocking and Stubbing

ExternalApi.java

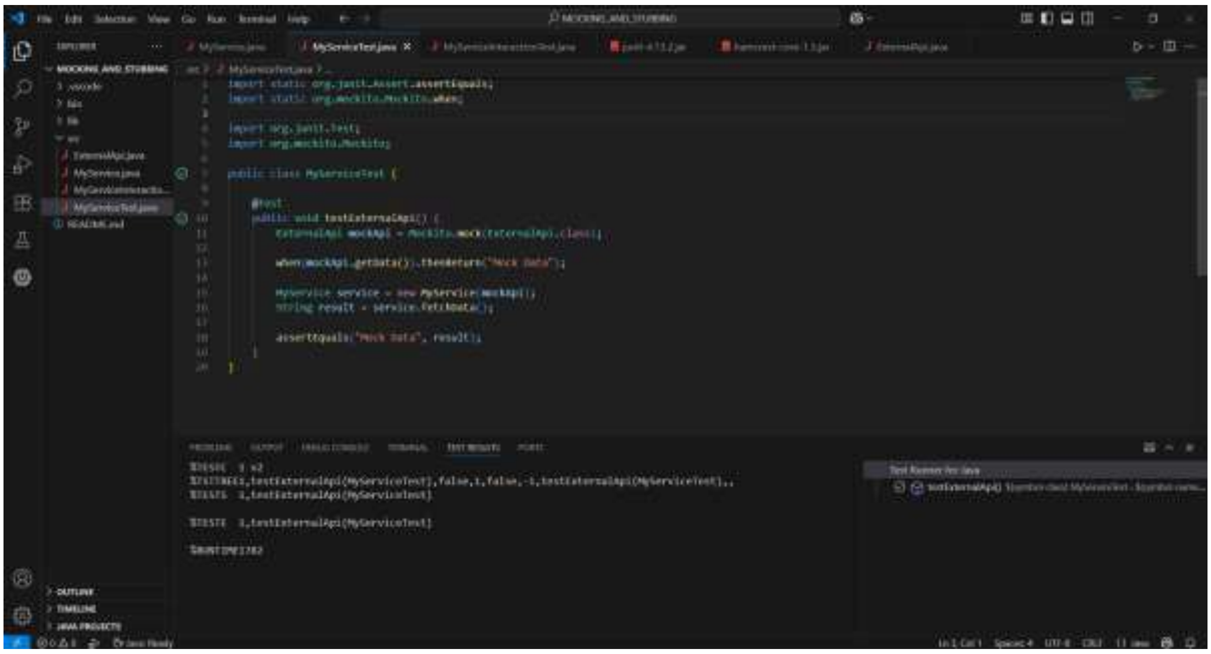
```
public interface ExternalApi {
    String getData();
}
```

MyService.java

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

```
}
MyServiceInteractionTest.java
import static org.mockito.Mockito.*;
import org.junit.Test;
import org.mockito.Mockito;
public class MyServiceInteractionTest {
    @Test
    public void testVerifyInteraction() {
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);
        MyService service = new MyService(mockApi);
        service.fetchData();
        verify(mockApi).getData();
    }
}
MyServiceTest.java
import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.when;
import org.junit.Test;
import org.mockito.Mockito;
public class MyServiceTest {
    @Test
    public void testExternalApi() {
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);

        when(mockApi.getData()).thenReturn("Mock Data");
        MyService service = new MyService(mockApi);
        String result = service.fetchData();
        assertEquals("Mock Data", result);
    }
}
```

OUTPUT:



```
import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.when;

import org.junit.Test;
import org.mockito.Mockito;

public class MyServiceTest {

    @Test
    public void testExternalApi() {
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);

        when(mockApi.getData()).thenReturn("Mock Data");

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

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

```
%TESTC  1 v2
%TSTTREE1,testExternalApi(MyServiceTest),false,1,false,-1,testExternalApi(MyServiceTest),,
%TESTS  1,testExternalApi(MyServiceTest)

%TESTE  1,testExternalApi(MyServiceTest)

%RUNTIME1782
```

Exercise 2: Verifying Interactions

ExternalApi.java

```
public interface ExternalApi {
    void getData();
}
```

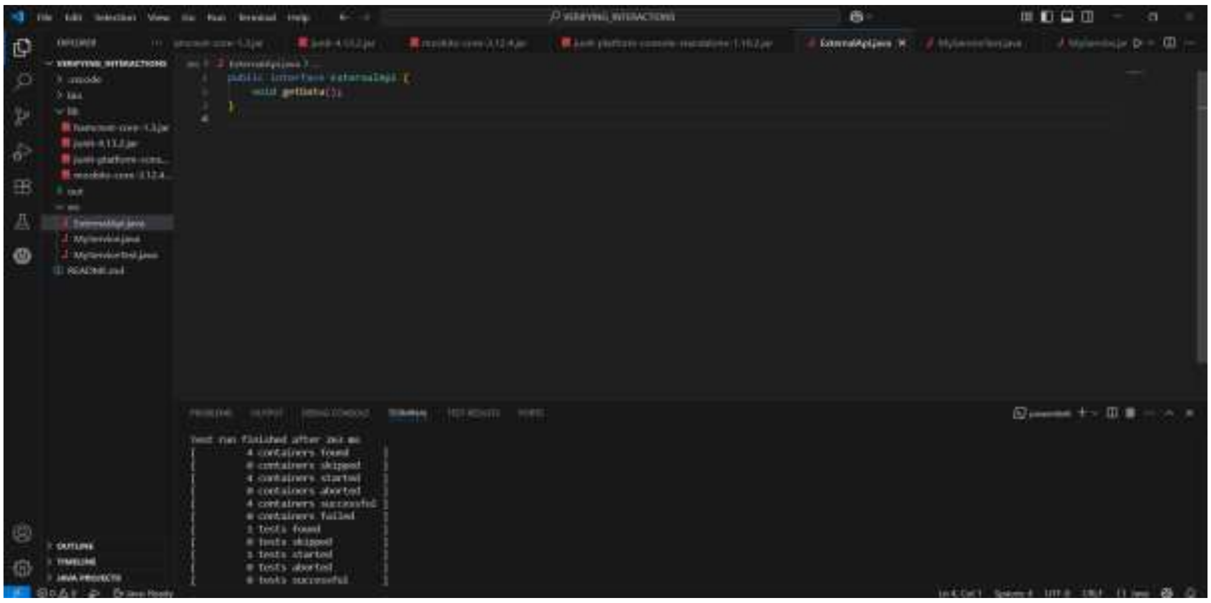
MyService.java

```
public class MyService {
    private final ExternalApi api;
  public MyService(ExternalApi api) {
        this.api = api;
    }
  public void fetchData() {
        api.getData();
    }
}
```

MyServiceTest.java

```
import static org.mockito.Mockito.*;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
public class MyServiceTest {
    @Test
    public void testVerifyInteraction() {
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);
        MyService service = new MyService(mockApi);
        service.fetchData();
        verify(mockApi).getData();
    }
}
```

OUTPUT:



```
public interface ExternalApi {
    void getData();
}
```

```
Test run finished after 261 ms
[         4 containers found      ]
[         0 containers skipped    ]
[         4 containers started    ]
[         0 containers aborted    ]
[         4 containers successful ]
[         0 containers failed     ]
[         1 tests found           ]
[         0 tests skipped         ]
[         1 tests started         ]
[         0 tests aborted         ]
[         0 tests successful      ]
```

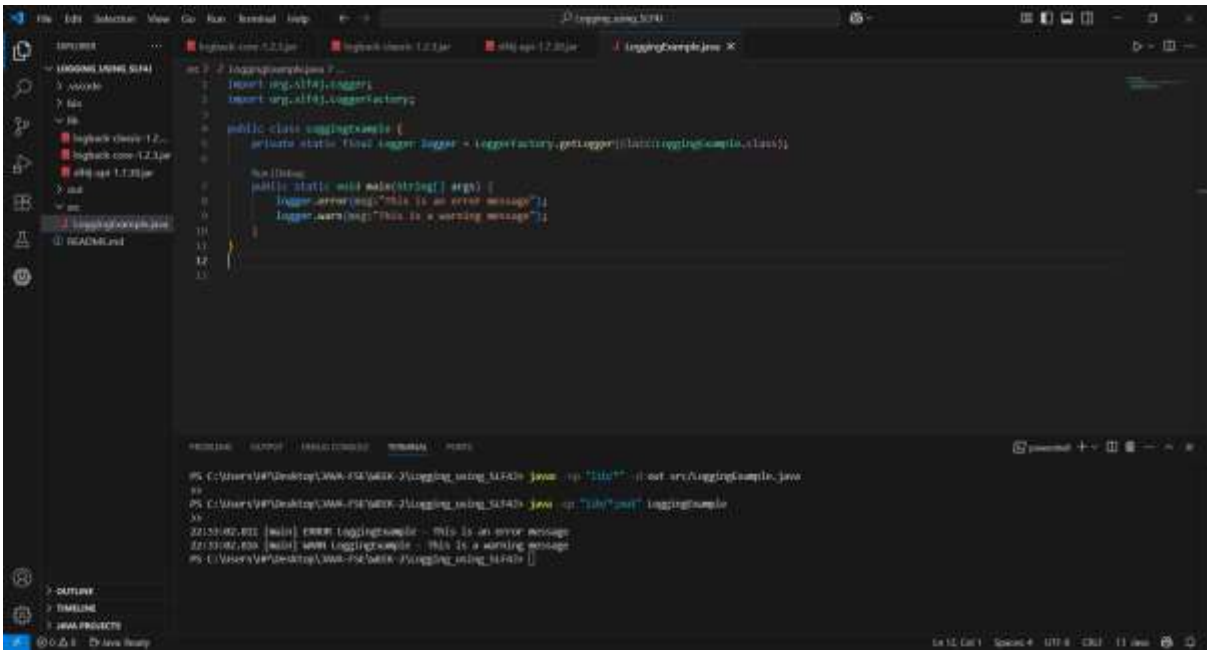
Logging using SLF4J

Exercise 1: Logging Error Messages and Warning Levels

LoggingExample.java

```java
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");

    }

}
```

OUTPUT:



```java
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class LoggingExample {
    private static final Logger logger = LoggerFactory.getLogger(clazz:LoggingExample.class);

    public static void main(String[] args) {
        logger.error(msg:"This is an error message");
        logger.warn(msg:"This is a warning message");
    }
}
```

```
PS C:\Users\HP\Desktop\JAVA-FSE\WEEK-2\Logging_using_SLF4J> javac -cp "lib/*" -d out src/LoggingExample.java
>>
PS C:\Users\HP\Desktop\JAVA-FSE\WEEK-2\Logging_using_SLF4J> java -cp "lib/*;out" LoggingExample
>>
22:33:02.831 [main] ERROR LoggingExample - This is an error message
22:33:02.836 [main] WARN LoggingExample - This is a warning message
PS C:\Users\HP\Desktop\JAVA-FSE\WEEK-2\Logging_using_SLF4J> 
```