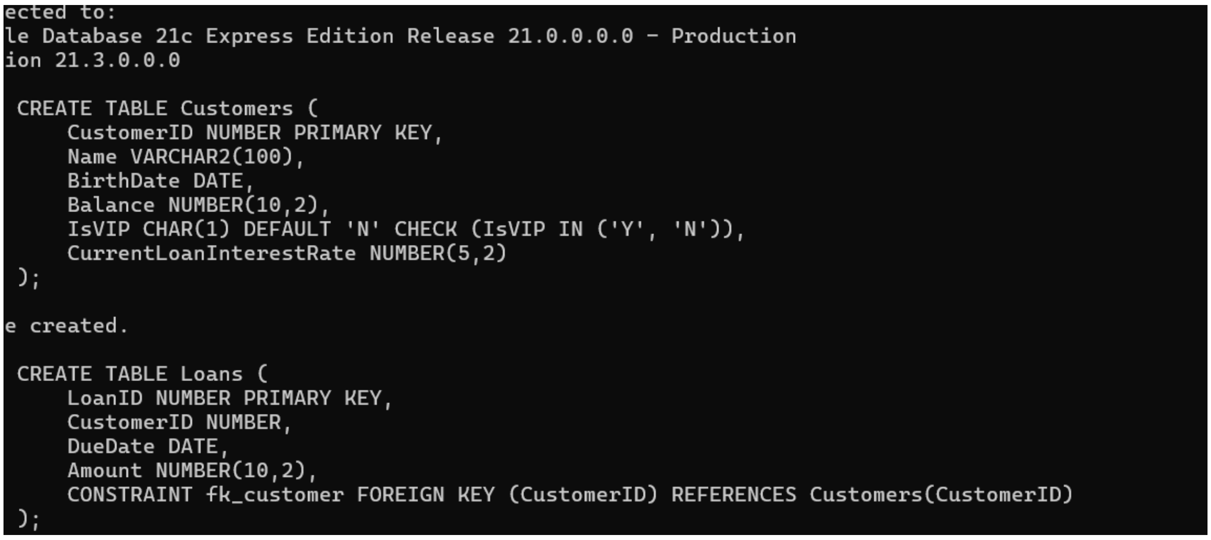
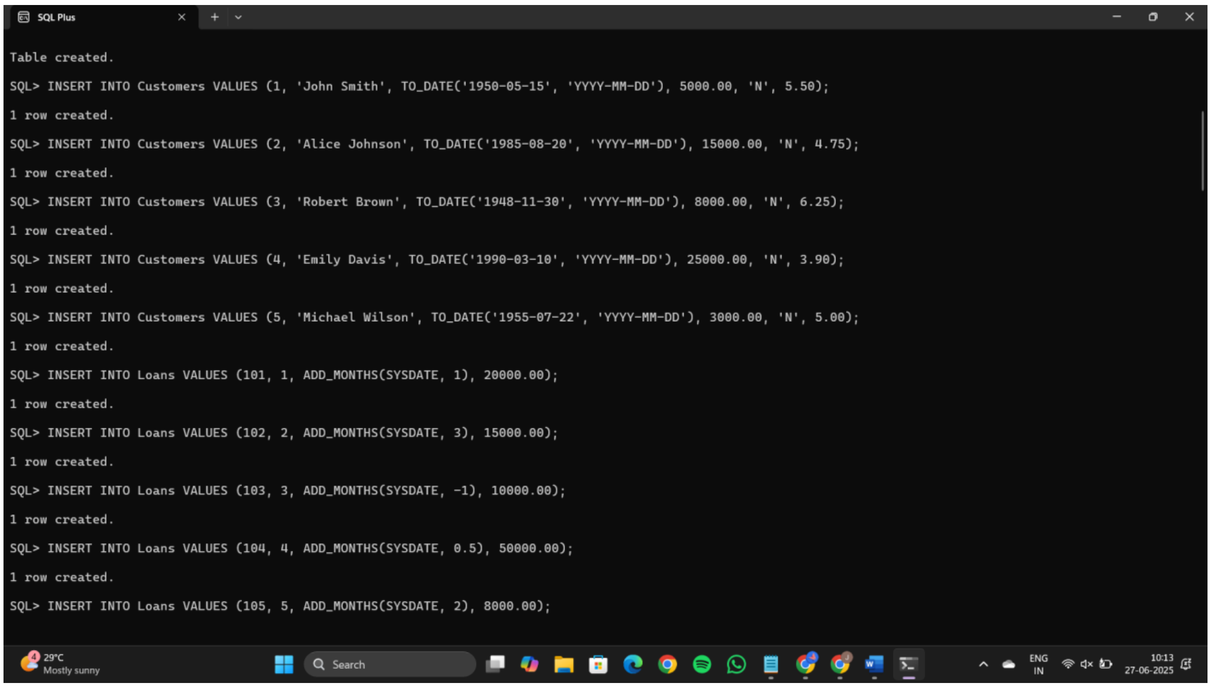
WEEK 2

1. <u>Exercise 1: Control Structures</u>

Table Creation:

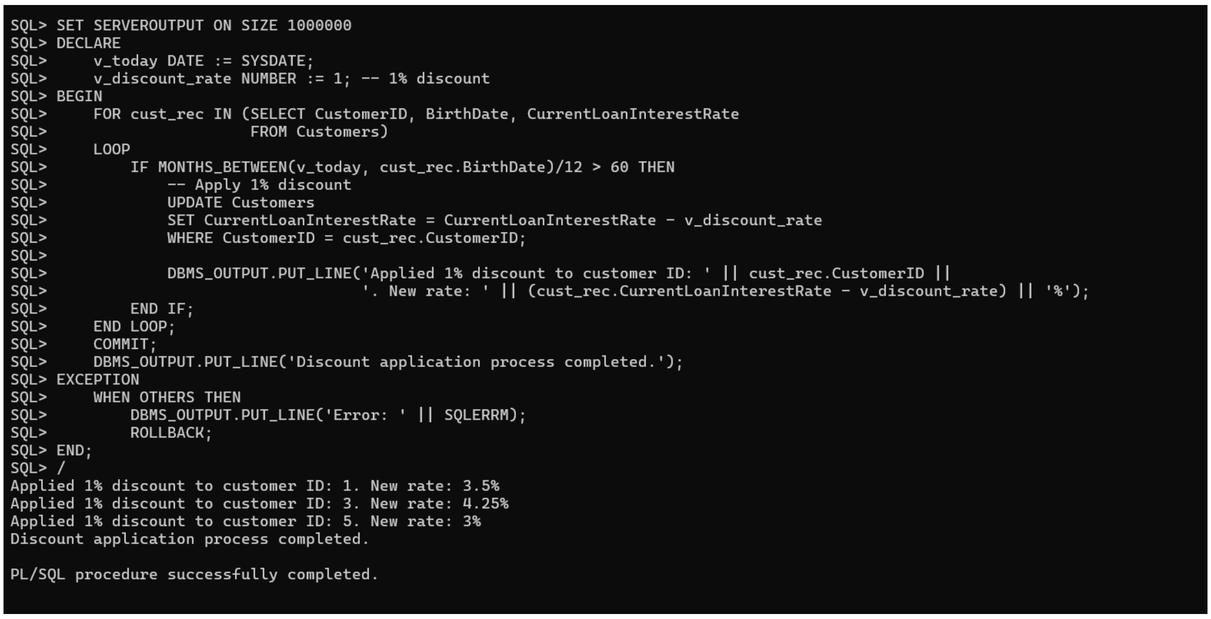


```
ected to:
le Database 21c Express Edition Release 21.0.0.0.0 - Production
ion 21.3.0.0.0

 CREATE TABLE Customers (
     CustomerID NUMBER PRIMARY KEY,
     Name VARCHAR2(100),
     BirthDate DATE,
     Balance NUMBER(10,2),
     IsVIP CHAR(1) DEFAULT 'N' CHECK (IsVIP IN ('Y', 'N')),
     CurrentLoanInterestRate NUMBER(5,2)
 );

e created.

 CREATE TABLE Loans (
     LoanID NUMBER PRIMARY KEY,
     CustomerID NUMBER,
     DueDate DATE,
     Amount NUMBER(10,2),
     CONSTRAINT fk_customer FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
 );
```



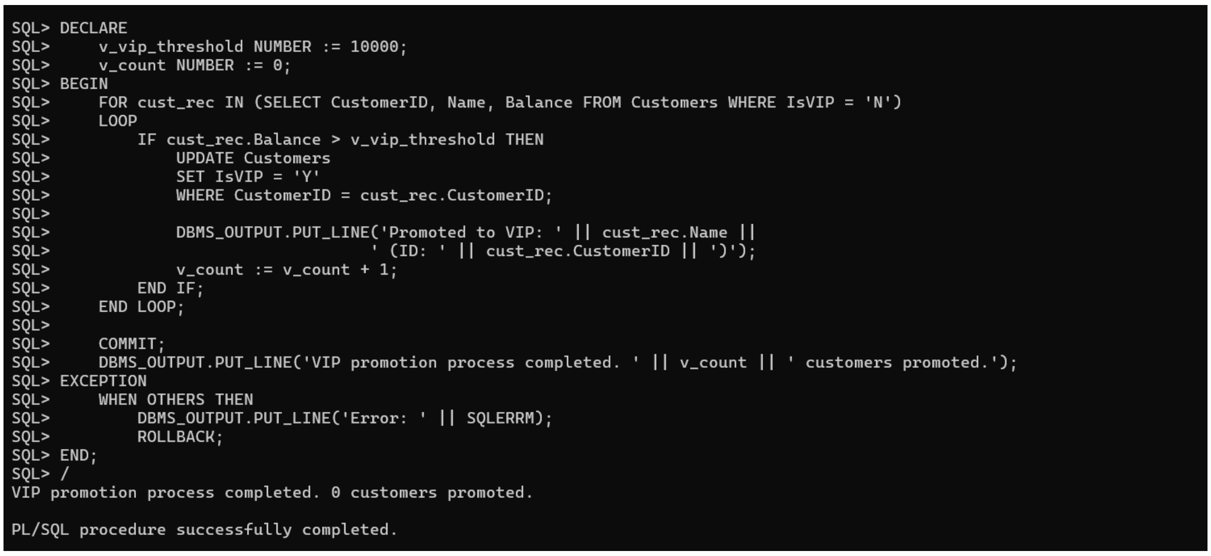
```
Table created.

SQL> INSERT INTO Customers VALUES (1, 'John Smith', TO_DATE('1950-05-15', 'YYYY-MM-DD'), 5000.00, 'N', 5.50);

1 row created.

SQL> INSERT INTO Customers VALUES (2, 'Alice Johnson', TO_DATE('1985-08-20', 'YYYY-MM-DD'), 15000.00, 'N', 4.75);

1 row created.

SQL> INSERT INTO Customers VALUES (3, 'Robert Brown', TO_DATE('1948-11-30', 'YYYY-MM-DD'), 8000.00, 'N', 6.25);

1 row created.

SQL> INSERT INTO Customers VALUES (4, 'Emily Davis', TO_DATE('1990-03-10', 'YYYY-MM-DD'), 25000.00, 'N', 3.90);

1 row created.

SQL> INSERT INTO Customers VALUES (5, 'Michael Wilson', TO_DATE('1955-07-22', 'YYYY-MM-DD'), 3000.00, 'N', 5.00);

1 row created.

SQL> INSERT INTO Loans VALUES (101, 1, ADD_MONTHS(SYSDATE, 1), 20000.00);

1 row created.

SQL> INSERT INTO Loans VALUES (102, 2, ADD_MONTHS(SYSDATE, 3), 15000.00);

1 row created.

SQL> INSERT INTO Loans VALUES (103, 3, ADD_MONTHS(SYSDATE, -1), 10000.00);

1 row created.

SQL> INSERT INTO Loans VALUES (104, 4, ADD_MONTHS(SYSDATE, 0.5), 50000.00);

1 row created.

SQL> INSERT INTO Loans VALUES (105, 5, ADD_MONTHS(SYSDATE, 2), 8000.00);
```

Scenario 1:



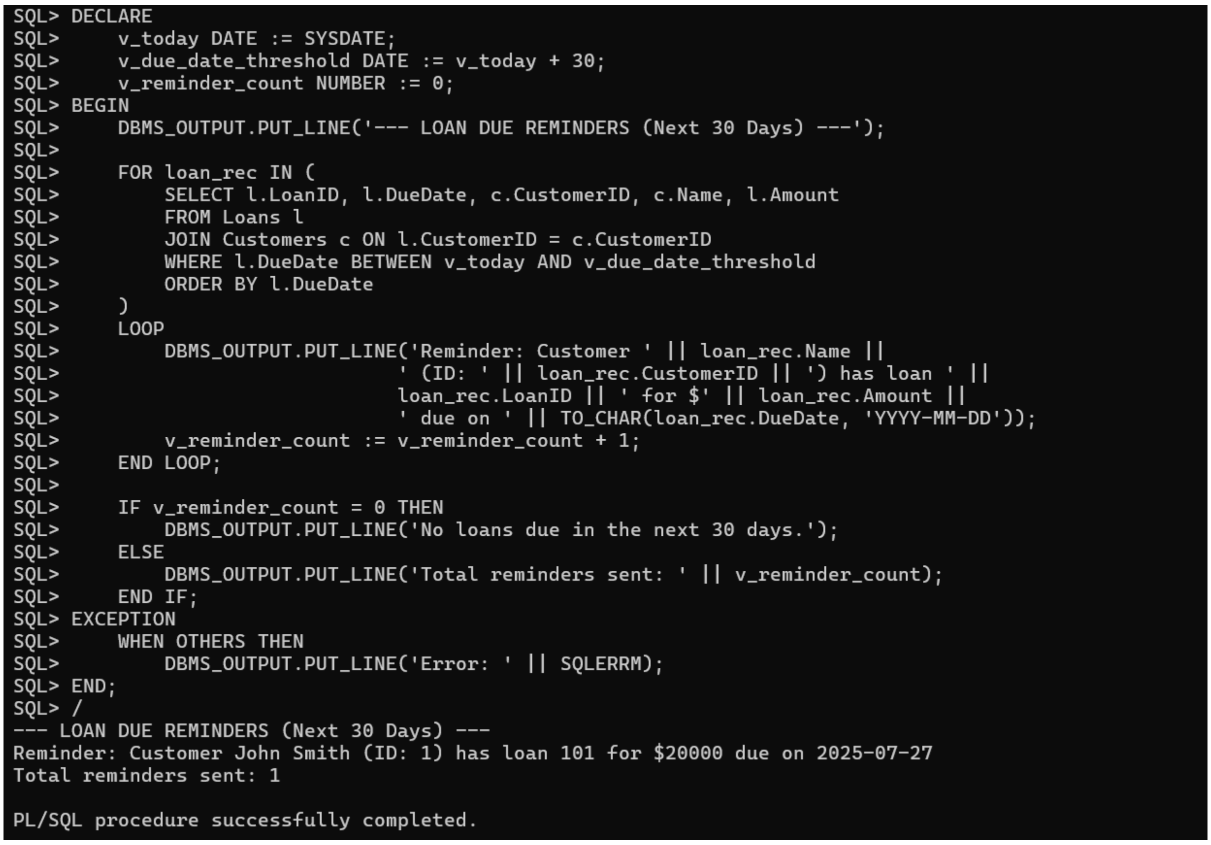
```
SQL> SET SERVEROUTPUT ON SIZE 1000000
SQL> DECLARE
SQL>     v_today DATE := SYSDATE;
SQL>     v_discount_rate NUMBER := 1; -- 1% discount
SQL> BEGIN
SQL>     FOR cust_rec IN (SELECT CustomerID, BirthDate, CurrentLoanInterestRate
SQL>                      FROM Customers)
SQL>     LOOP
SQL>         IF MONTHS_BETWEEN(v_today, cust_rec.BirthDate)/12 > 60 THEN
SQL>             -- Apply 1% discount
SQL>             UPDATE Customers
SQL>             SET CurrentLoanInterestRate = CurrentLoanInterestRate - v_discount_rate
SQL>             WHERE CustomerID = cust_rec.CustomerID;
SQL>
SQL>             DBMS_OUTPUT.PUT_LINE('Applied 1% discount to customer ID: ' || cust_rec.CustomerID ||
SQL>                                  '. New rate: ' || (cust_rec.CurrentLoanInterestRate - v_discount_rate) || '%');
SQL>         END IF;
SQL>     END LOOP;
SQL>     COMMIT;
SQL>     DBMS_OUTPUT.PUT_LINE('Discount application process completed.');
SQL> EXCEPTION
SQL>     WHEN OTHERS THEN
SQL>         DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
SQL>         ROLLBACK;
SQL> END;
SQL> /
Applied 1% discount to customer ID: 1. New rate: 3.5%
Applied 1% discount to customer ID: 3. New rate: 4.25%
Applied 1% discount to customer ID: 5. New rate: 3%
Discount application process completed.

PL/SQL procedure successfully completed.
```

Scenario 2:



```
SQL> DECLARE
SQL>     v_vip_threshold NUMBER := 10000;
SQL>     v_count NUMBER := 0;
SQL> BEGIN
SQL>     FOR cust_rec IN (SELECT CustomerID, Name, Balance FROM Customers WHERE IsVIP = 'N')
SQL>     LOOP
SQL>         IF cust_rec.Balance > v_vip_threshold THEN
SQL>             UPDATE Customers
SQL>             SET IsVIP = 'Y'
SQL>             WHERE CustomerID = cust_rec.CustomerID;
SQL>
SQL>             DBMS_OUTPUT.PUT_LINE('Promoted to VIP: ' || cust_rec.Name ||
SQL>                                  ' (ID: ' || cust_rec.CustomerID || ')');
SQL>             v_count := v_count + 1;
SQL>         END IF;
SQL>     END LOOP;
SQL>
SQL>     COMMIT;
SQL>     DBMS_OUTPUT.PUT_LINE('VIP promotion process completed. ' || v_count || ' customers promoted.');
SQL> EXCEPTION
SQL>     WHEN OTHERS THEN
SQL>         DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
SQL>         ROLLBACK;
SQL> END;
SQL> /
VIP promotion process completed. 0 customers promoted.

PL/SQL procedure successfully completed.
```

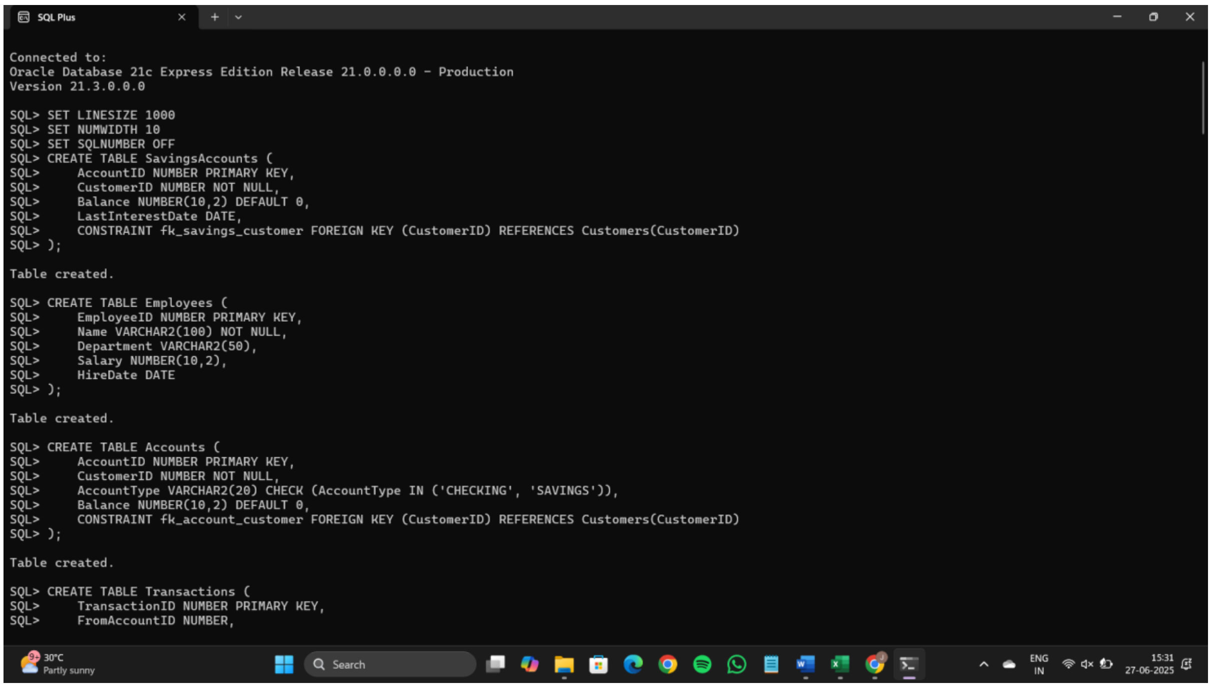
Scenario 3:



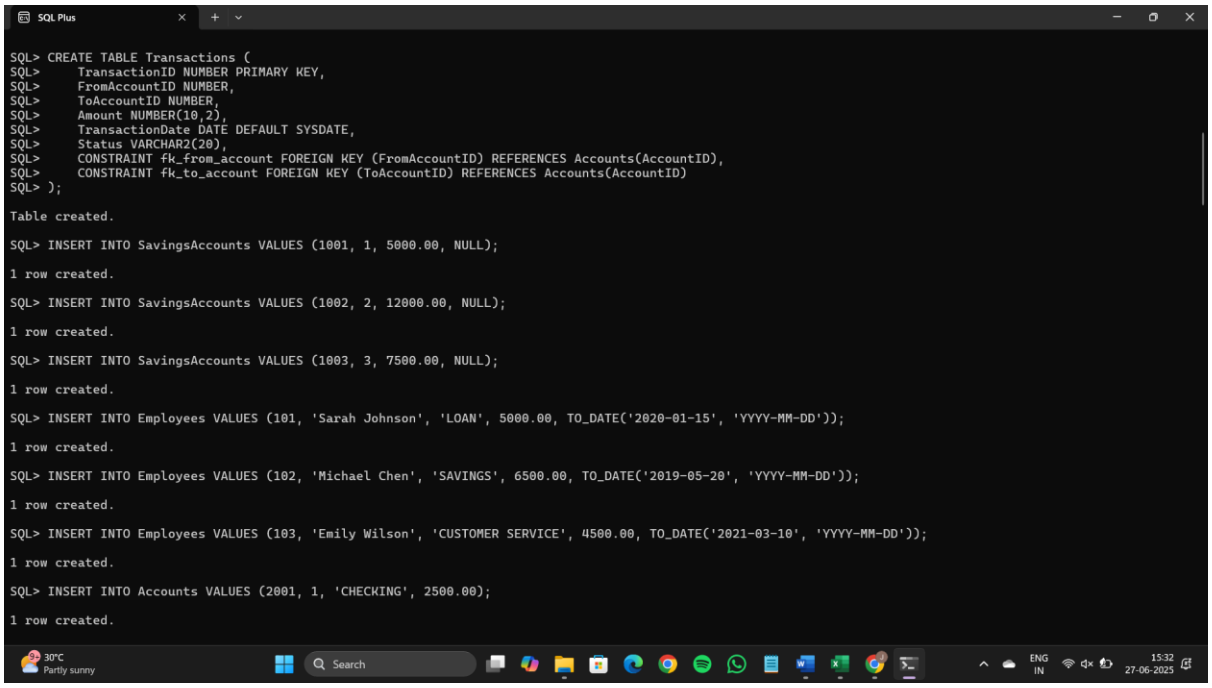
```
SQL> DECLARE
SQL>     v_today DATE := SYSDATE;
SQL>     v_due_date_threshold DATE := v_today + 30;
SQL>     v_reminder_count NUMBER := 0;
SQL> BEGIN
SQL>     DBMS_OUTPUT.PUT_LINE('--- LOAN DUE REMINDERS (Next 30 Days) ---');
SQL>
SQL>     FOR loan_rec IN (
SQL>         SELECT l.LoanID, l.DueDate, c.CustomerID, c.Name, l.Amount
SQL>         FROM Loans l
SQL>         JOIN Customers c ON l.CustomerID = c.CustomerID
SQL>         WHERE l.DueDate BETWEEN v_today AND v_due_date_threshold
SQL>         ORDER BY l.DueDate
SQL>     )
SQL>     LOOP
SQL>         DBMS_OUTPUT.PUT_LINE('Reminder: Customer ' || loan_rec.Name ||
SQL>                              ' (ID: ' || loan_rec.CustomerID || ') has loan ' ||
SQL>                              loan_rec.LoanID || ' for $' || loan_rec.Amount ||
SQL>                              ' due on ' || TO_CHAR(loan_rec.DueDate, 'YYYY-MM-DD'));
SQL>         v_reminder_count := v_reminder_count + 1;
SQL>     END LOOP;
SQL>
SQL>     IF v_reminder_count = 0 THEN
SQL>         DBMS_OUTPUT.PUT_LINE('No loans due in the next 30 days.');
SQL>     ELSE
SQL>         DBMS_OUTPUT.PUT_LINE('Total reminders sent: ' || v_reminder_count);
SQL>     END IF;
SQL> EXCEPTION
SQL>     WHEN OTHERS THEN
SQL>         DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
SQL> END;
SQL> /
--- LOAN DUE REMINDERS (Next 30 Days) ---
Reminder: Customer John Smith (ID: 1) has loan 101 for $20000 due on 2025-07-27
Total reminders sent: 1

PL/SQL procedure successfully completed.
```

2. Exercise 3: Stored Procedures

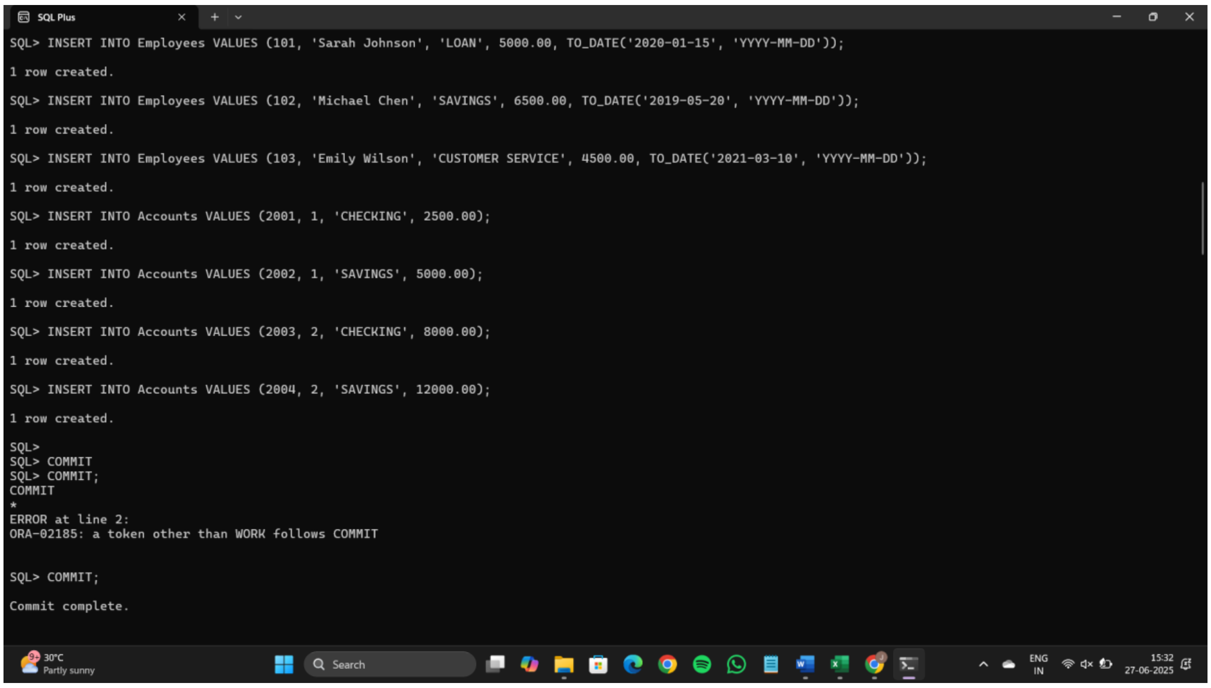
Table Creation:



```
Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> SET LINESIZE 1000
SQL> SET NUMWIDTH 10
SQL> SET SQLNUMBER OFF
SQL> CREATE TABLE SavingsAccounts (
SQL>     AccountID NUMBER PRIMARY KEY,
SQL>     CustomerID NUMBER NOT NULL,
SQL>     Balance NUMBER(10,2) DEFAULT 0,
SQL>     LastInterestDate DATE,
SQL>     CONSTRAINT fk_savings_customer FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
SQL> );

Table created.

SQL> CREATE TABLE Employees (
SQL>     EmployeeID NUMBER PRIMARY KEY,
SQL>     Name VARCHAR2(100) NOT NULL,
SQL>     Department VARCHAR2(50),
SQL>     Salary NUMBER(10,2),
SQL>     HireDate DATE
SQL> );

Table created.

SQL> CREATE TABLE Accounts (
SQL>     AccountID NUMBER PRIMARY KEY,
SQL>     CustomerID NUMBER NOT NULL,
SQL>     AccountType VARCHAR2(20) CHECK (AccountType IN ('CHECKING', 'SAVINGS')),
SQL>     Balance NUMBER(10,2) DEFAULT 0,
SQL>     CONSTRAINT fk_account_customer FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
SQL> );

Table created.

SQL> CREATE TABLE Transactions (
SQL>     TransactionID NUMBER PRIMARY KEY,
SQL>     FromAccountID NUMBER,
```

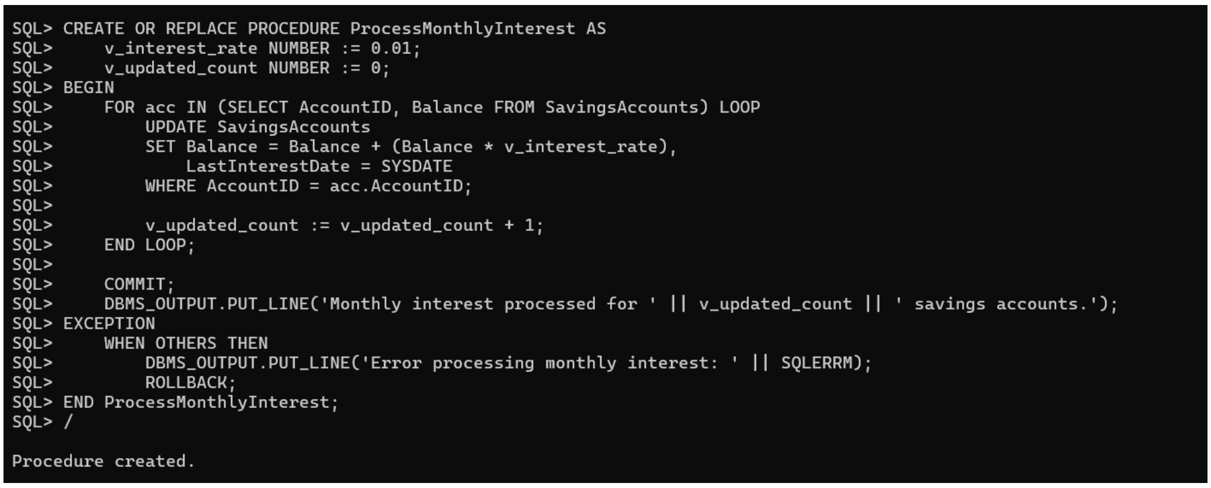


```
SQL> CREATE TABLE Transactions (
SQL>     TransactionID NUMBER PRIMARY KEY,
SQL>     FromAccountID NUMBER,
SQL>     ToAccountID NUMBER,
SQL>     Amount NUMBER(10,2),
SQL>     TransactionDate DATE DEFAULT SYSDATE,
SQL>     Status VARCHAR2(20),
SQL>     CONSTRAINT fk_from_account FOREIGN KEY (FromAccountID) REFERENCES Accounts(AccountID),
SQL>     CONSTRAINT fk_to_account FOREIGN KEY (ToAccountID) REFERENCES Accounts(AccountID)
SQL> );

Table created.

SQL> INSERT INTO SavingsAccounts VALUES (1001, 1, 5000.00, NULL);

1 row created.

SQL> INSERT INTO SavingsAccounts VALUES (1002, 2, 12000.00, NULL);

1 row created.

SQL> INSERT INTO SavingsAccounts VALUES (1003, 3, 7500.00, NULL);

1 row created.

SQL> INSERT INTO Employees VALUES (101, 'Sarah Johnson', 'LOAN', 5000.00, TO_DATE('2020-01-15', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO Employees VALUES (102, 'Michael Chen', 'SAVINGS', 6500.00, TO_DATE('2019-05-20', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO Employees VALUES (103, 'Emily Wilson', 'CUSTOMER SERVICE', 4500.00, TO_DATE('2021-03-10', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO Accounts VALUES (2001, 1, 'CHECKING', 2500.00);

1 row created.
```



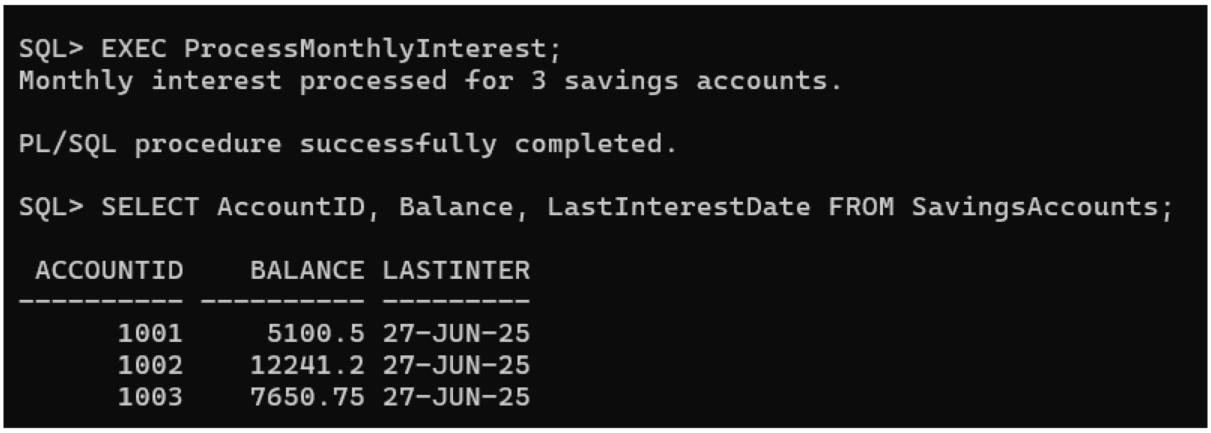
```
SQL> INSERT INTO Employees VALUES (101, 'Sarah Johnson', 'LOAN', 5000.00, TO_DATE('2020-01-15', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO Employees VALUES (102, 'Michael Chen', 'SAVINGS', 6500.00, TO_DATE('2019-05-20', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO Employees VALUES (103, 'Emily Wilson', 'CUSTOMER SERVICE', 4500.00, TO_DATE('2021-03-10', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO Accounts VALUES (2001, 1, 'CHECKING', 2500.00);

1 row created.

SQL> INSERT INTO Accounts VALUES (2002, 1, 'SAVINGS', 5000.00);

1 row created.

SQL> INSERT INTO Accounts VALUES (2003, 2, 'CHECKING', 8000.00);

1 row created.

SQL> INSERT INTO Accounts VALUES (2004, 2, 'SAVINGS', 12000.00);

1 row created.

SQL>
SQL> COMMIT
SQL> COMMIT;
COMMIT
*
ERROR at line 2:
ORA-02185: a token other than WORK follows COMMIT


SQL> COMMIT;

Commit complete.
```

Scenario 1:



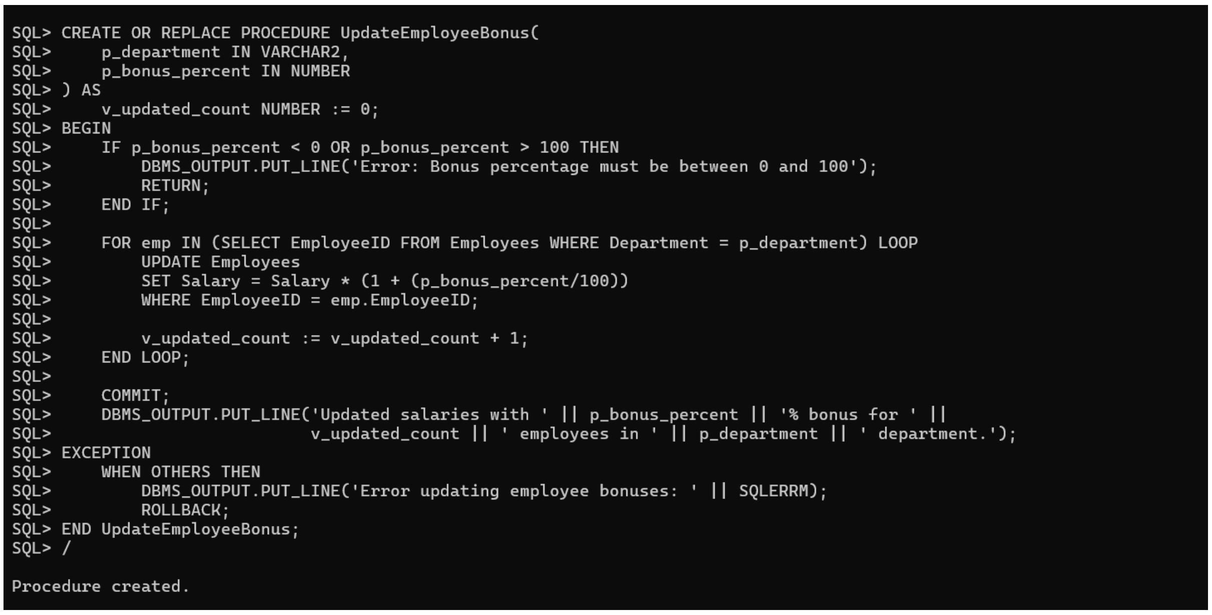
```
SQL> CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
SQL>     v_interest_rate NUMBER := 0.01;
SQL>     v_updated_count NUMBER := 0;
SQL> BEGIN
SQL>     FOR acc IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP
SQL>         UPDATE SavingsAccounts
SQL>         SET Balance = Balance + (Balance * v_interest_rate),
SQL>             LastInterestDate = SYSDATE
SQL>         WHERE AccountID = acc.AccountID;
SQL>
SQL>         v_updated_count := v_updated_count + 1;
SQL>     END LOOP;
SQL>
SQL>     COMMIT;
SQL>     DBMS_OUTPUT.PUT_LINE('Monthly interest processed for ' || v_updated_count || ' savings accounts.');
SQL> EXCEPTION
SQL>     WHEN OTHERS THEN
SQL>         DBMS_OUTPUT.PUT_LINE('Error processing monthly interest: ' || SQLERRM);
SQL>         ROLLBACK;
SQL> END ProcessMonthlyInterest;
SQL> /

Procedure created.
```

Output1:

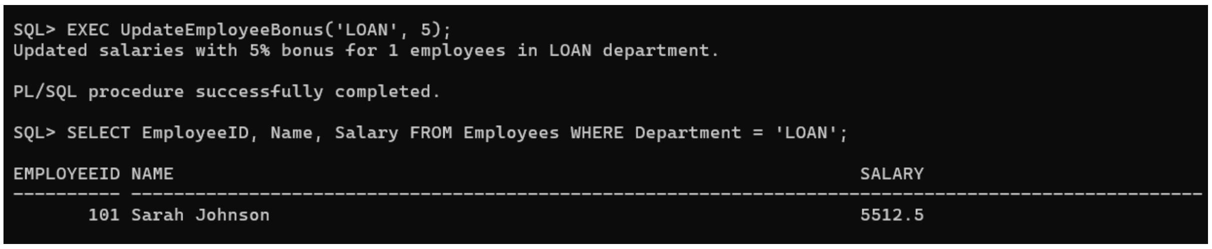


```
SQL> EXEC ProcessMonthlyInterest;
Monthly interest processed for 3 savings accounts.

PL/SQL procedure successfully completed.

SQL> SELECT AccountID, Balance, LastInterestDate FROM SavingsAccounts;

 ACCOUNTID    BALANCE LASTINTER
---------- ---------- ---------
      1001     5100.5 27-JUN-25
      1002    12241.2 27-JUN-25
      1003    7650.75 27-JUN-25
```

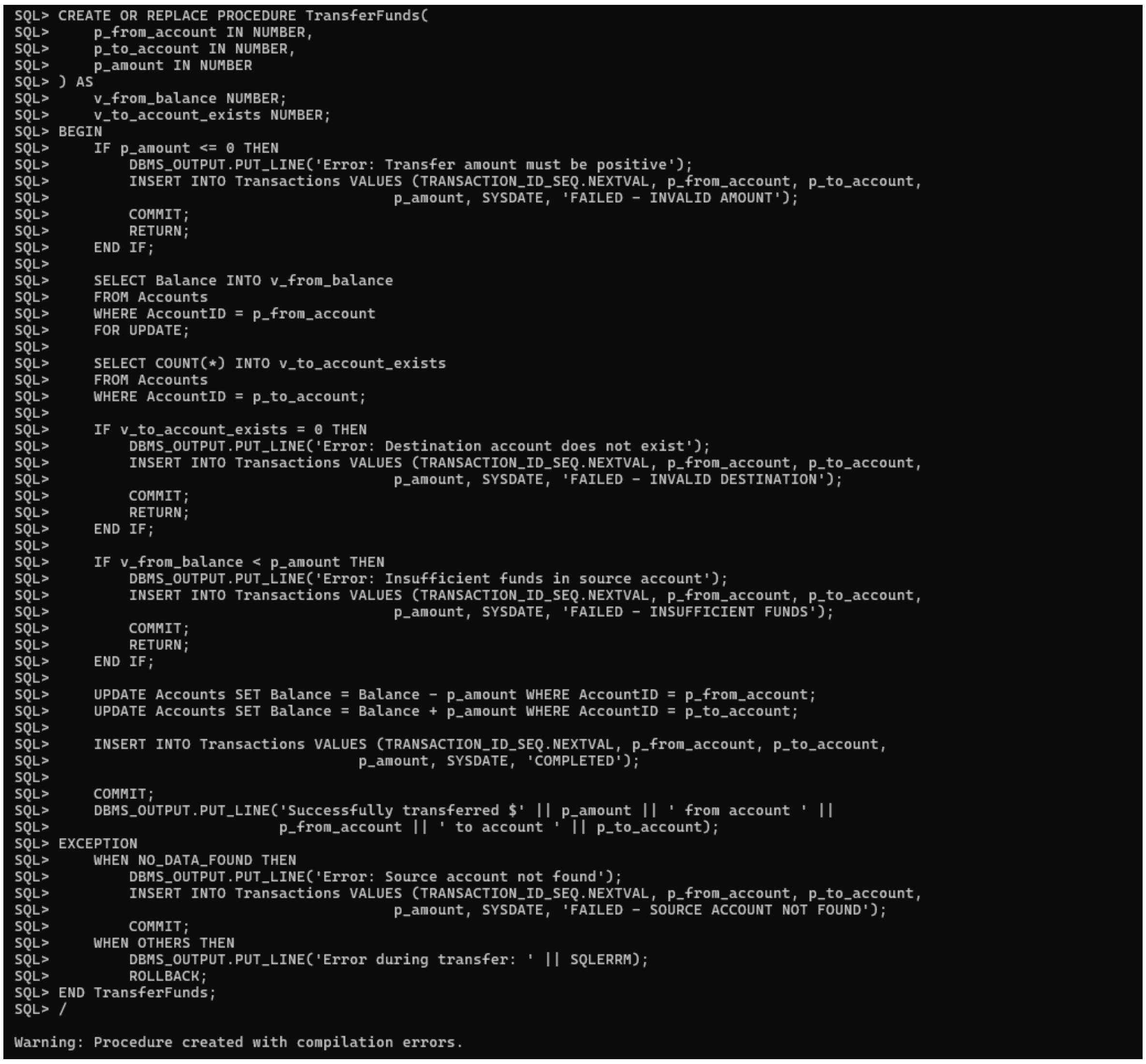
Scenario 2:


```
SQL> CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(
SQL>     p_department IN VARCHAR2,
SQL>     p_bonus_percent IN NUMBER
SQL> ) AS
SQL>     v_updated_count NUMBER := 0;
SQL> BEGIN
SQL>     IF p_bonus_percent < 0 OR p_bonus_percent > 100 THEN
SQL>         DBMS_OUTPUT.PUT_LINE('Error: Bonus percentage must be between 0 and 100');
SQL>         RETURN;
SQL>     END IF;
SQL>
SQL>     FOR emp IN (SELECT EmployeeID FROM Employees WHERE Department = p_department) LOOP
SQL>         UPDATE Employees
SQL>         SET Salary = Salary * (1 + (p_bonus_percent/100))
SQL>         WHERE EmployeeID = emp.EmployeeID;
SQL>
SQL>         v_updated_count := v_updated_count + 1;
SQL>     END LOOP;
SQL>
SQL>     COMMIT;
SQL>     DBMS_OUTPUT.PUT_LINE('Updated salaries with ' || p_bonus_percent || '% bonus for ' ||
SQL>                          v_updated_count || ' employees in ' || p_department || ' department.');
SQL> EXCEPTION
SQL>     WHEN OTHERS THEN
SQL>         DBMS_OUTPUT.PUT_LINE('Error updating employee bonuses: ' || SQLERRM);
SQL>         ROLLBACK;
SQL> END UpdateEmployeeBonus;
SQL> /

Procedure created.
```

Output2:

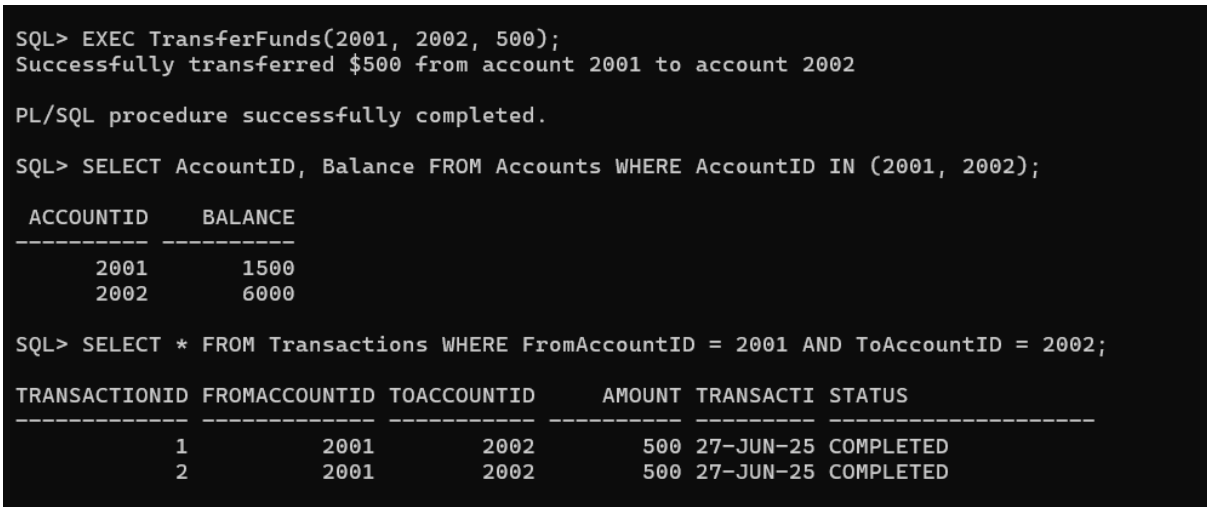

```
SQL> EXEC UpdateEmployeeBonus('LOAN', 5);
Updated salaries with 5% bonus for 1 employees in LOAN department.

PL/SQL procedure successfully completed.

SQL> SELECT EmployeeID, Name, Salary FROM Employees WHERE Department = 'LOAN';

EMPLOYEEID NAME                                                                   SALARY
---------- ----------------------------------------------------------------------------------------------------
       101 Sarah Johnson                                                          5512.5
```

Scenario 3:



```
SQL> CREATE OR REPLACE PROCEDURE TransferFunds(
SQL>     p_from_account IN NUMBER,
SQL>     p_to_account IN NUMBER,
SQL>     p_amount IN NUMBER
SQL> ) AS
SQL>     v_from_balance NUMBER;
SQL>     v_to_account_exists NUMBER;
SQL> BEGIN
SQL>     IF p_amount <= 0 THEN
SQL>         DBMS_OUTPUT.PUT_LINE('Error: Transfer amount must be positive');
SQL>         INSERT INTO Transactions VALUES (TRANSACTION_ID_SEQ.NEXTVAL, p_from_account, p_to_account,
SQL>                                       p_amount, SYSDATE, 'FAILED - INVALID AMOUNT');
SQL>         COMMIT;
SQL>         RETURN;
SQL>     END IF;
SQL>
SQL>     SELECT Balance INTO v_from_balance
SQL>     FROM Accounts
SQL>     WHERE AccountID = p_from_account
SQL>     FOR UPDATE;
SQL>
SQL>     SELECT COUNT(*) INTO v_to_account_exists
SQL>     FROM Accounts
SQL>     WHERE AccountID = p_to_account;
SQL>
SQL>     IF v_to_account_exists = 0 THEN
SQL>         DBMS_OUTPUT.PUT_LINE('Error: Destination account does not exist');
SQL>         INSERT INTO Transactions VALUES (TRANSACTION_ID_SEQ.NEXTVAL, p_from_account, p_to_account,
SQL>                                       p_amount, SYSDATE, 'FAILED - INVALID DESTINATION');
SQL>         COMMIT;
SQL>         RETURN;
SQL>     END IF;
SQL>
SQL>     IF v_from_balance < p_amount THEN
SQL>         DBMS_OUTPUT.PUT_LINE('Error: Insufficient funds in source account');
SQL>         INSERT INTO Transactions VALUES (TRANSACTION_ID_SEQ.NEXTVAL, p_from_account, p_to_account,
SQL>                                       p_amount, SYSDATE, 'FAILED - INSUFFICIENT FUNDS');
SQL>         COMMIT;
SQL>         RETURN;
SQL>     END IF;
SQL>
SQL>     UPDATE Accounts SET Balance = Balance - p_amount WHERE AccountID = p_from_account;
SQL>     UPDATE Accounts SET Balance = Balance + p_amount WHERE AccountID = p_to_account;
SQL>
SQL>     INSERT INTO Transactions VALUES (TRANSACTION_ID_SEQ.NEXTVAL, p_from_account, p_to_account,
SQL>                                   p_amount, SYSDATE, 'COMPLETED');
SQL>
SQL>     COMMIT;
SQL>     DBMS_OUTPUT.PUT_LINE('Successfully transferred $' || p_amount || ' from account ' ||
SQL>                          p_from_account || ' to account ' || p_to_account);
SQL> EXCEPTION
SQL>     WHEN NO_DATA_FOUND THEN
SQL>         DBMS_OUTPUT.PUT_LINE('Error: Source account not found');
SQL>         INSERT INTO Transactions VALUES (TRANSACTION_ID_SEQ.NEXTVAL, p_from_account, p_to_account,
SQL>                                       p_amount, SYSDATE, 'FAILED - SOURCE ACCOUNT NOT FOUND');
SQL>         COMMIT;
SQL>     WHEN OTHERS THEN
SQL>         DBMS_OUTPUT.PUT_LINE('Error during transfer: ' || SQLERRM);
SQL>         ROLLBACK;
SQL> END TransferFunds;
SQL> /

Warning: Procedure created with compilation errors.
```

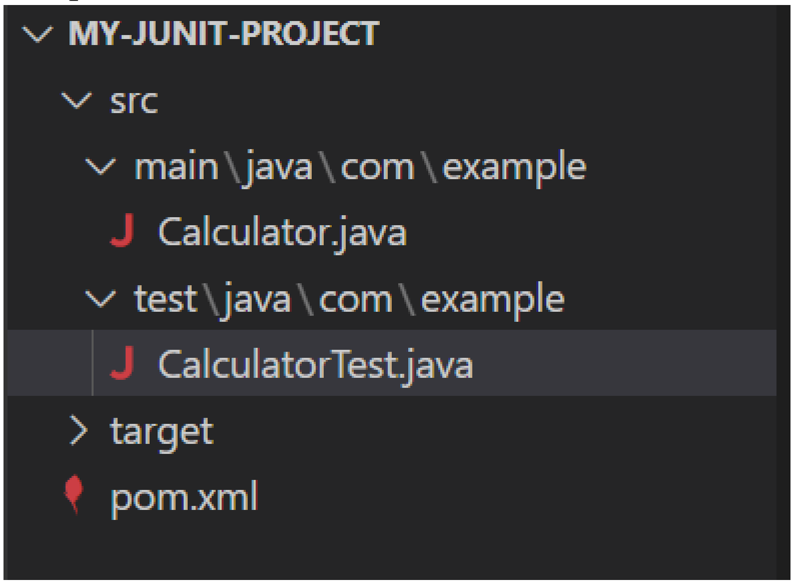
Output3:



```
SQL> EXEC TransferFunds(2001, 2002, 500);
Successfully transferred $500 from account 2001 to account 2002

PL/SQL procedure successfully completed.

SQL> SELECT AccountID, Balance FROM Accounts WHERE AccountID IN (2001, 2002);

 ACCOUNTID    BALANCE
---------- ----------
      2001       1500
      2002       6000

SQL> SELECT * FROM Transactions WHERE FromAccountID = 2001 AND ToAccountID = 2002;

TRANSACTIONID FROMACCOUNTID TOACCOUNTID     AMOUNT TRANSACTI STATUS
------------- ------------- ----------- ---------- --------- --------------------
            1          2001        2002        500 27-JUN-25 COMPLETED
            2          2001        2002        500 27-JUN-25 COMPLETED
```

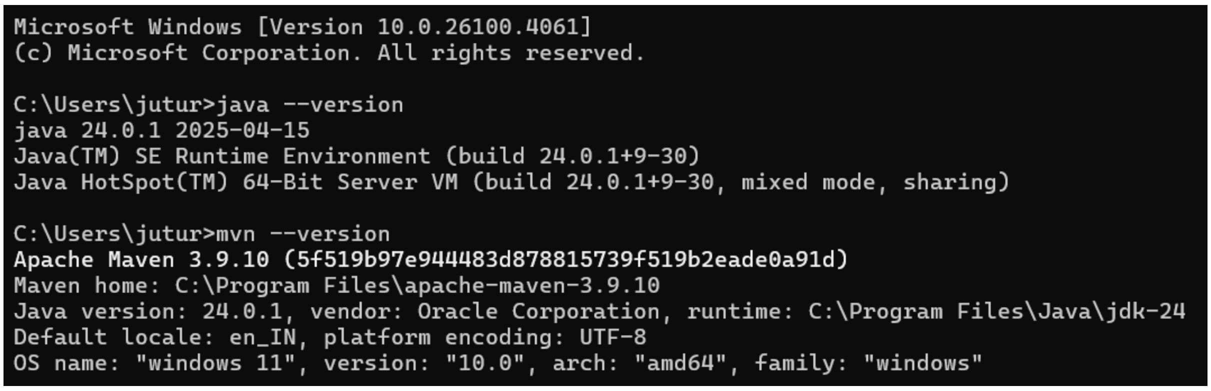
3. Exercise 1: Setting Up Junit

Setup:



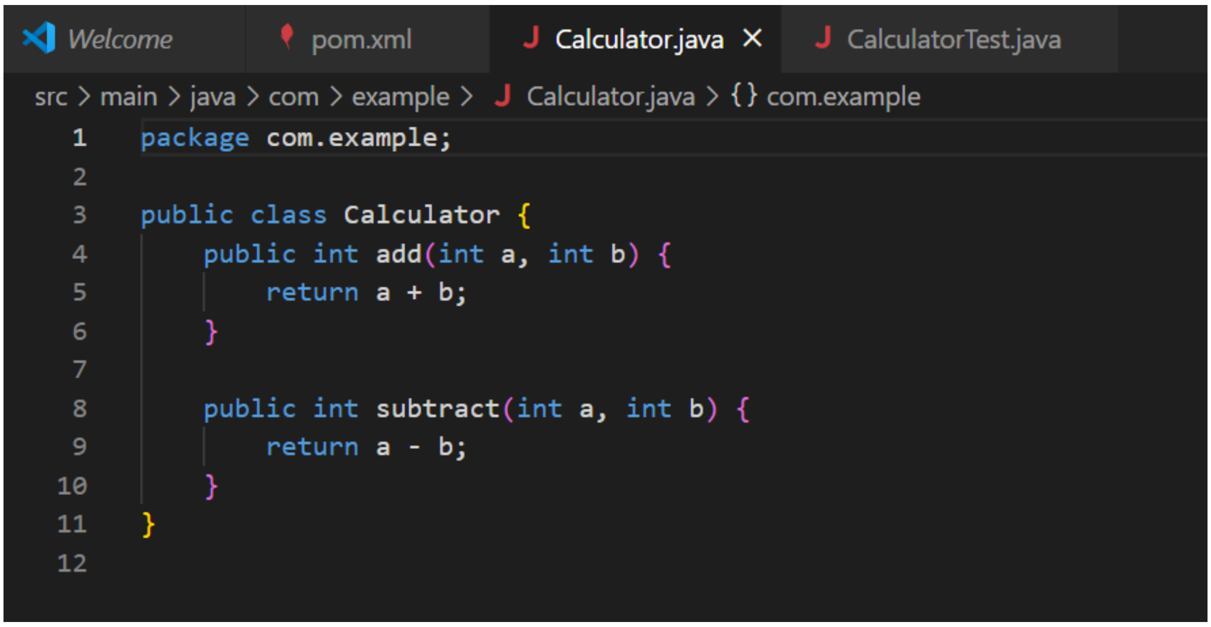
```
MY-JUNIT-PROJECT
  src
    main\java\com\example
      Calculator.java
    test\java\com\example
      CalculatorTest.java
  target
  pom.xml
```

Java and maven :



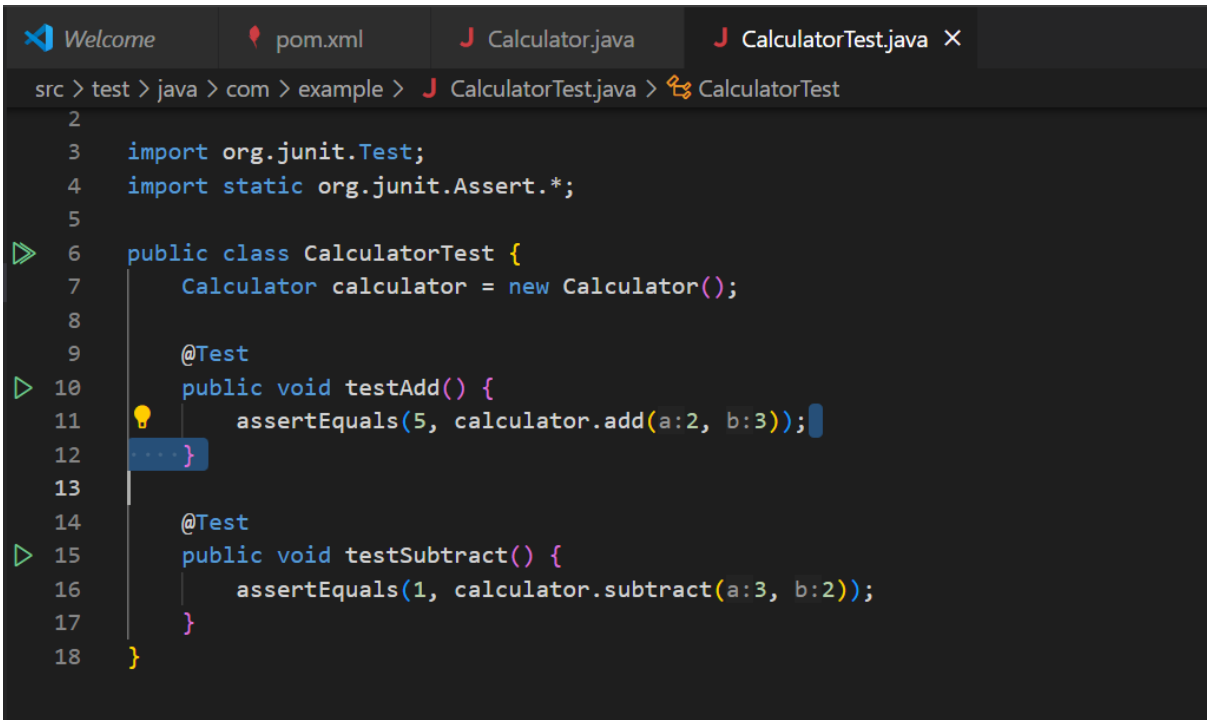
```
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jutur>java --version
java 24.0.1 2025-04-15
Java(TM) SE Runtime Environment (build 24.0.1+9-30)
Java HotSpot(TM) 64-Bit Server VM (build 24.0.1+9-30, mixed mode, sharing)

C:\Users\jutur>mvn --version
Apache Maven 3.9.10 (5f519b97e944483d878815739f519b2eade0a91d)
Maven home: C:\Program Files\apache-maven-3.9.10
Java version: 24.0.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-24
Default locale: en_IN, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
```

Calculator.java


Welcome
pom.xml
Calculator.java
CalculatorTest.java
src > main > java > com > example > Calculator.java > {} com.example

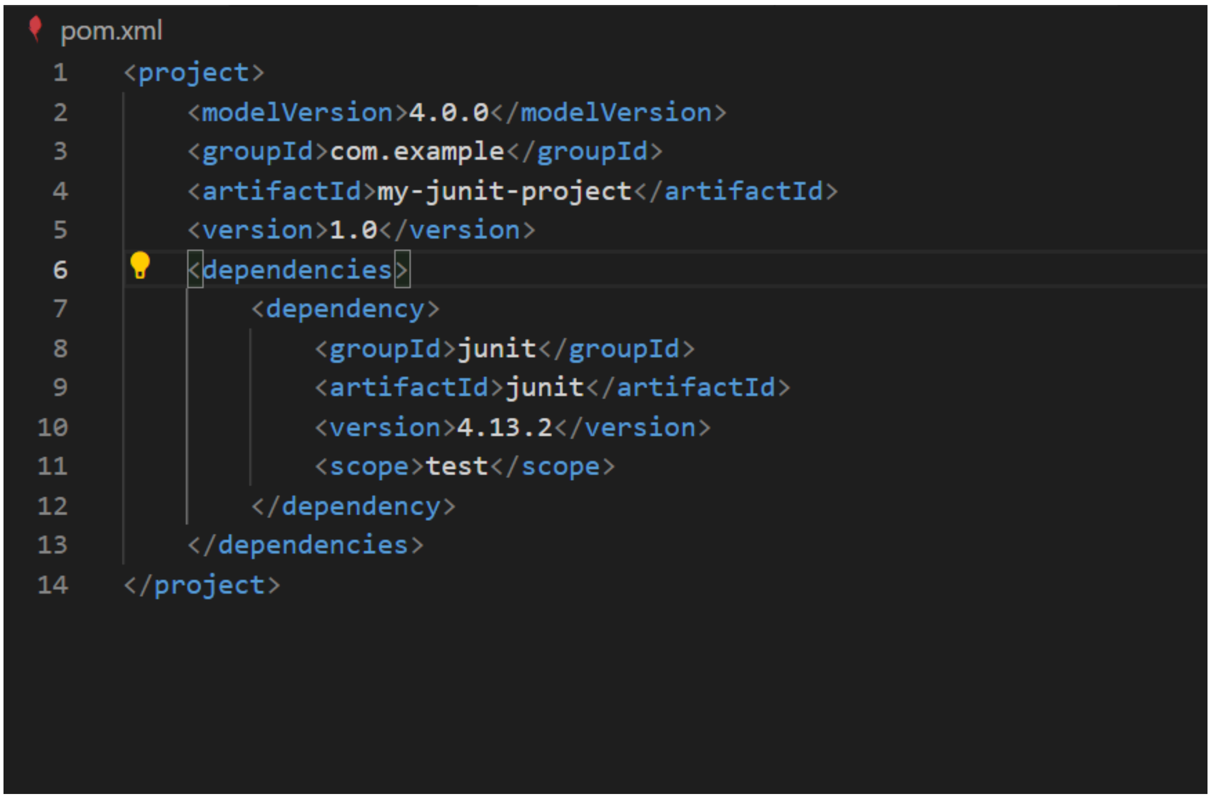
```java
package com.example;

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


Welcome
pom.xml
Calculator.java
CalculatorTest.java
src > test > java > com > example > CalculatorTest.java > CalculatorTest

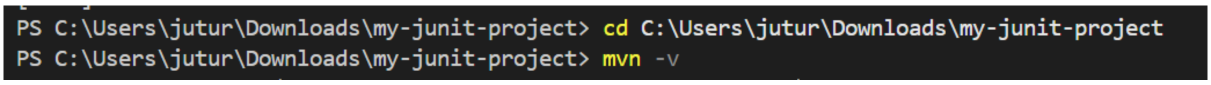
```java

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

public class CalculatorTest {
    Calculator calculator = new Calculator();

    @Test
    public void testAdd() {
        assertEquals(5, calculator.add(a:2, b:3));
    }

    @Test
    public void testSubtract() {
        assertEquals(1, calculator.subtract(a:3, b:2));
    }
}
```

pom.xml

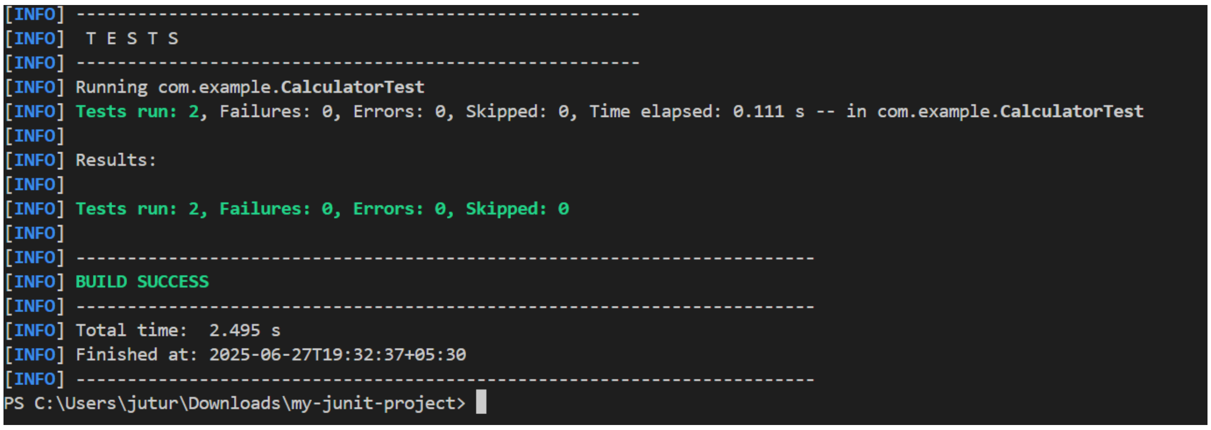


```
pom.xml
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>my-junit-project</artifactId>
    <version>1.0</version>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Output:



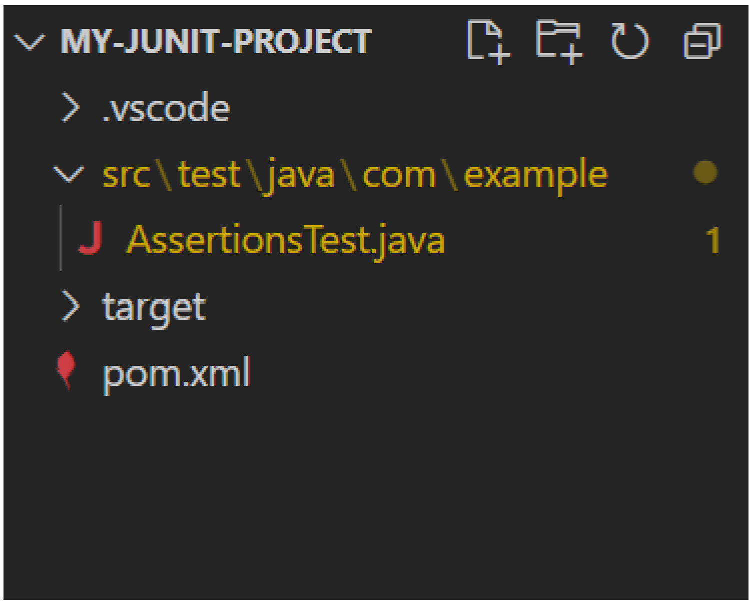
```
PS C:\Users\jutur\Downloads\my-junit-project> cd C:\Users\jutur\Downloads\my-junit-project
PS C:\Users\jutur\Downloads\my-junit-project> mvn -v
```



```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.CalculatorTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.111 s -- in com.example.CalculatorTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.495 s
[INFO] Finished at: 2025-06-27T19:32:37+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\jutur\Downloads\my-junit-project>
```

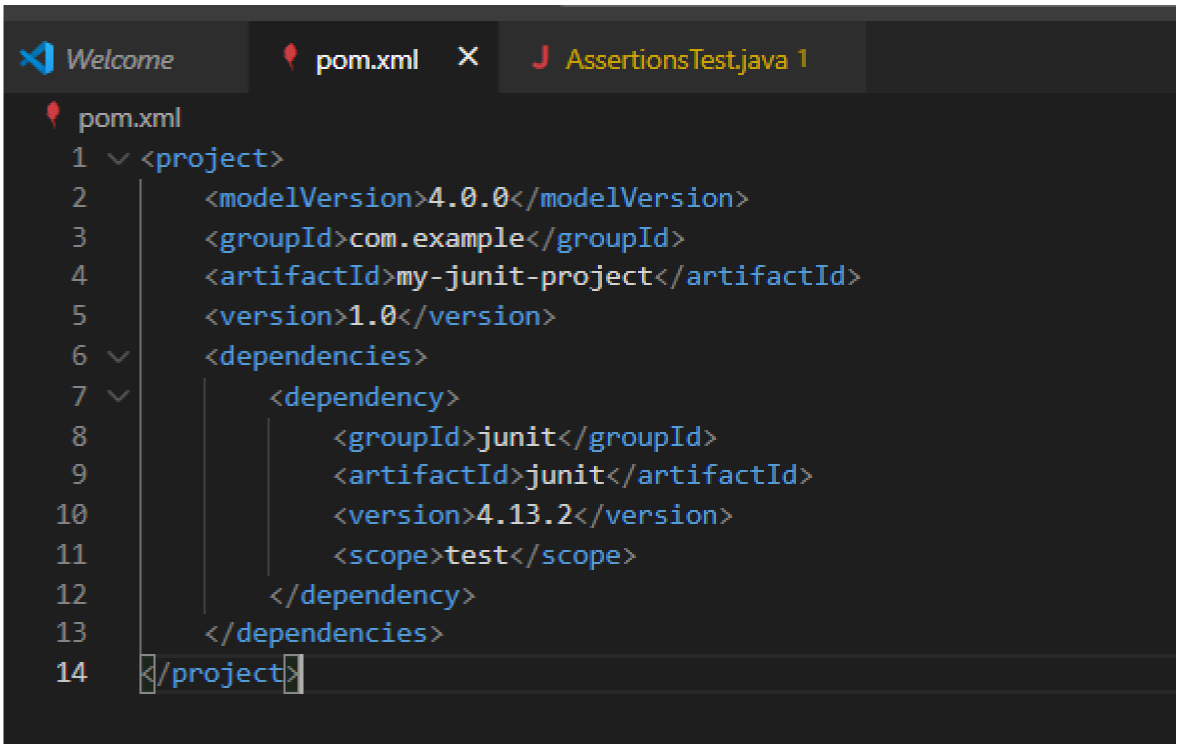
4. **Exercise 3: Assertions in Junit**

Setup:



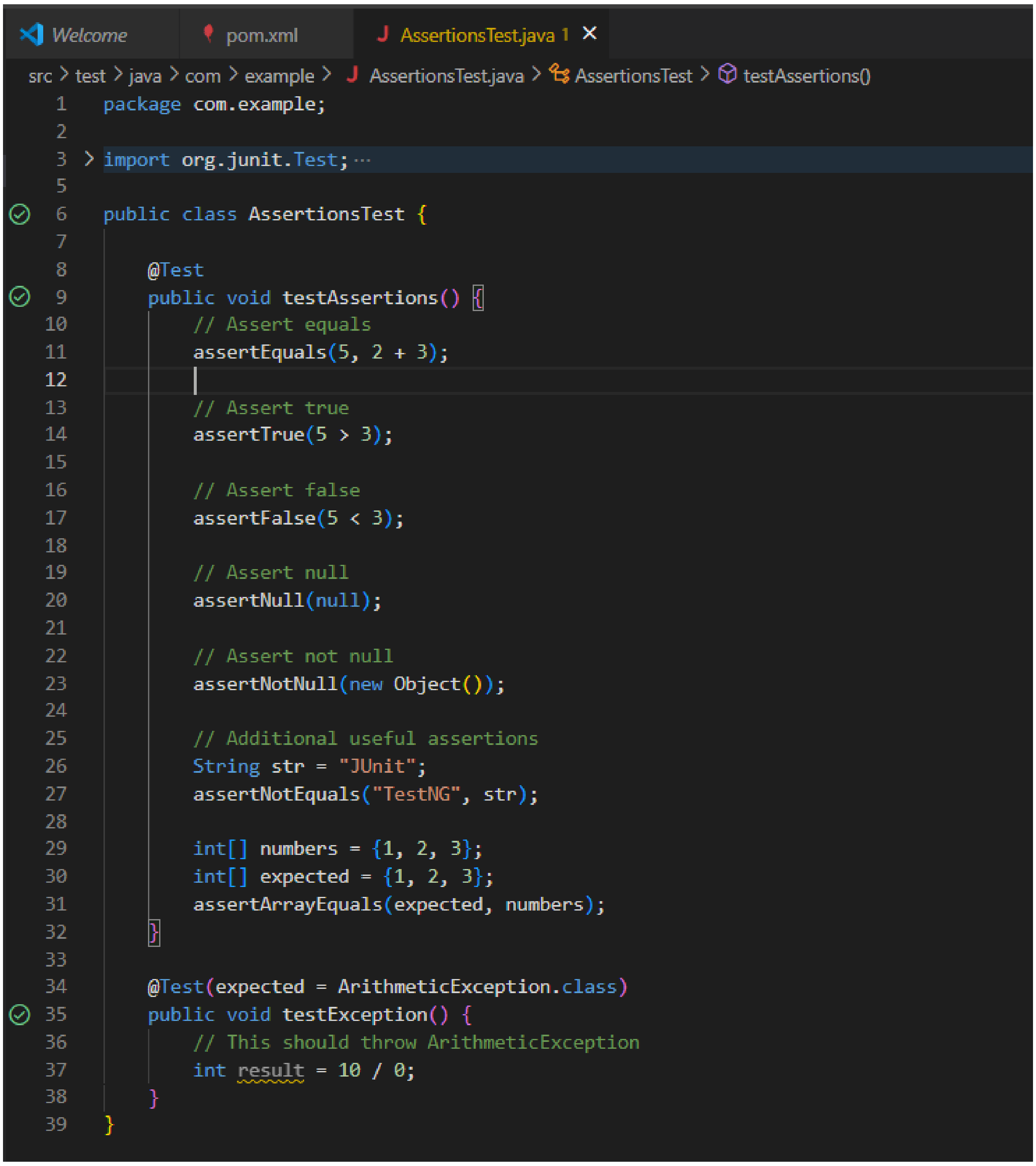
```
MY-JUNIT-PROJECT
  > .vscode
  v src\test\java\com\example
      AssertionsTest.java   1
  > target
    pom.xml
```

pom.xml



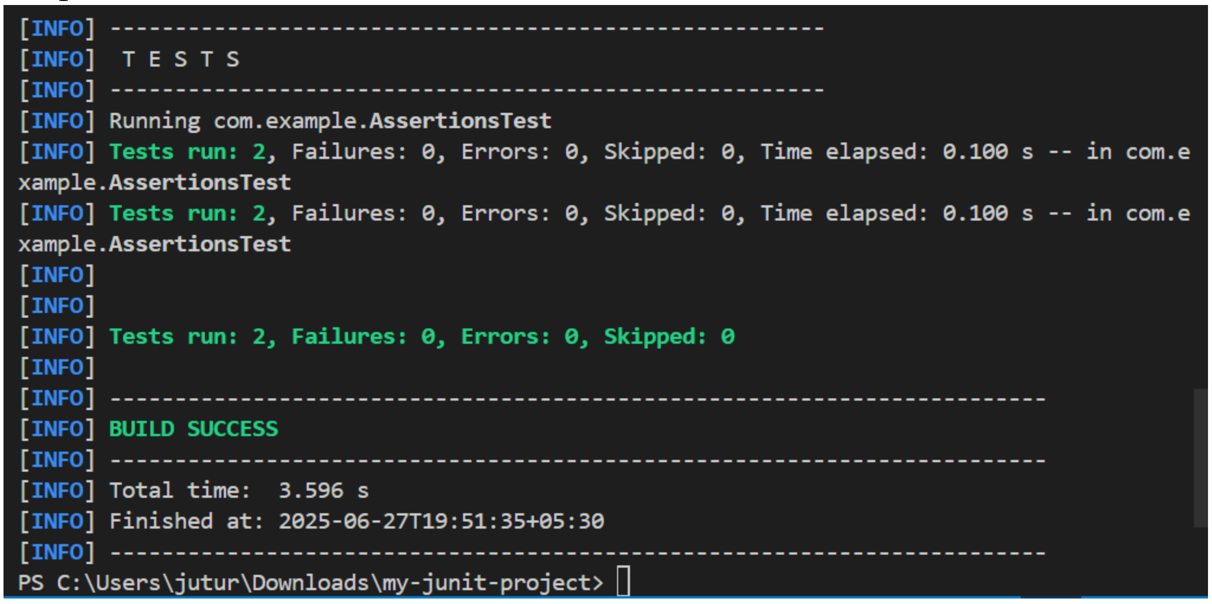
```xml
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>my-junit-project</artifactId>
    <version>1.0</version>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

AssertionsTest.java


Welcome
pom.xml
AssertionsTest.java 1
src > test > java > com > example > AssertionsTest.java > AssertionsTest > testAssertions()

```java
package com.example;

import org.junit.Test;···

public class AssertionsTest {

    @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());

        // Additional useful assertions
        String str = "JUnit";
        assertNotEquals("TestNG", str);

        int[] numbers = {1, 2, 3};
        int[] expected = {1, 2, 3};
        assertArrayEquals(expected, numbers);
    }

    @Test(expected = ArithmeticException.class)
    public void testException() {
        // This should throw ArithmeticException
        int result = 10 / 0;
    }
}
```

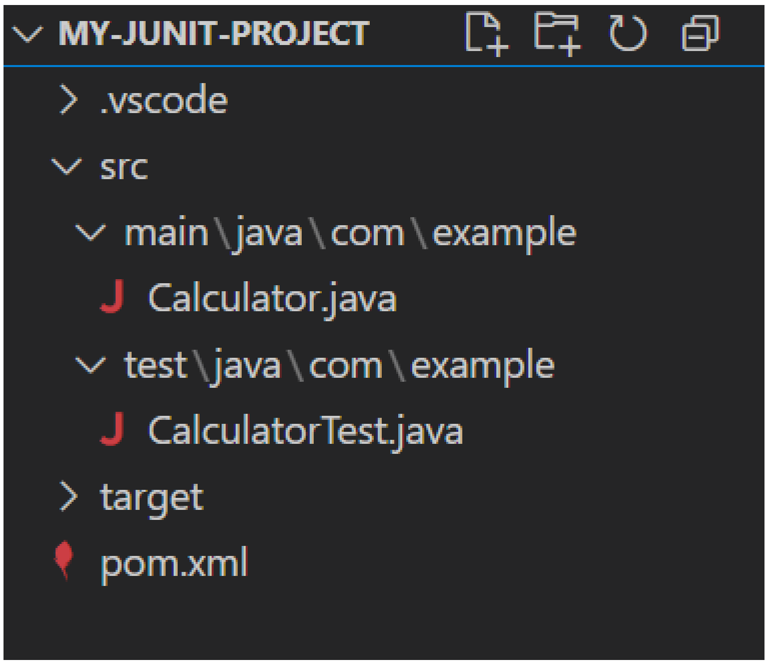
Output:



```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.AssertionsTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.100 s -- in com.e
xample.AssertionsTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.100 s -- in com.e
xample.AssertionsTest
[INFO]
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.596 s
[INFO] Finished at: 2025-06-27T19:51:35+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\jutur\Downloads\my-junit-project>
```

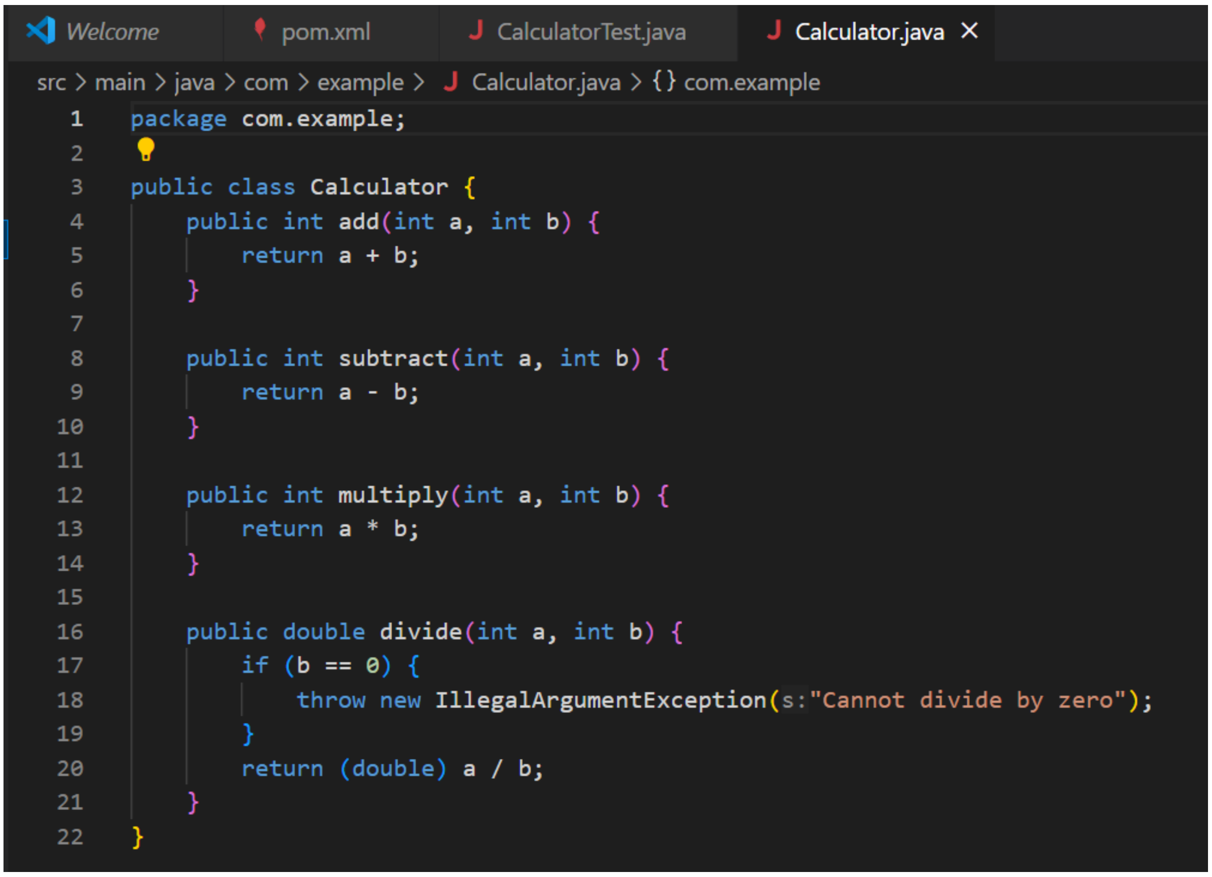
5. Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in Junit

Setup:



```
MY-JUNIT-PROJECT
  > .vscode
  v src
    v main\java\com\example
        Calculator.java
    v test\java\com\example
        CalculatorTest.java
  > target
    pom.xml
```

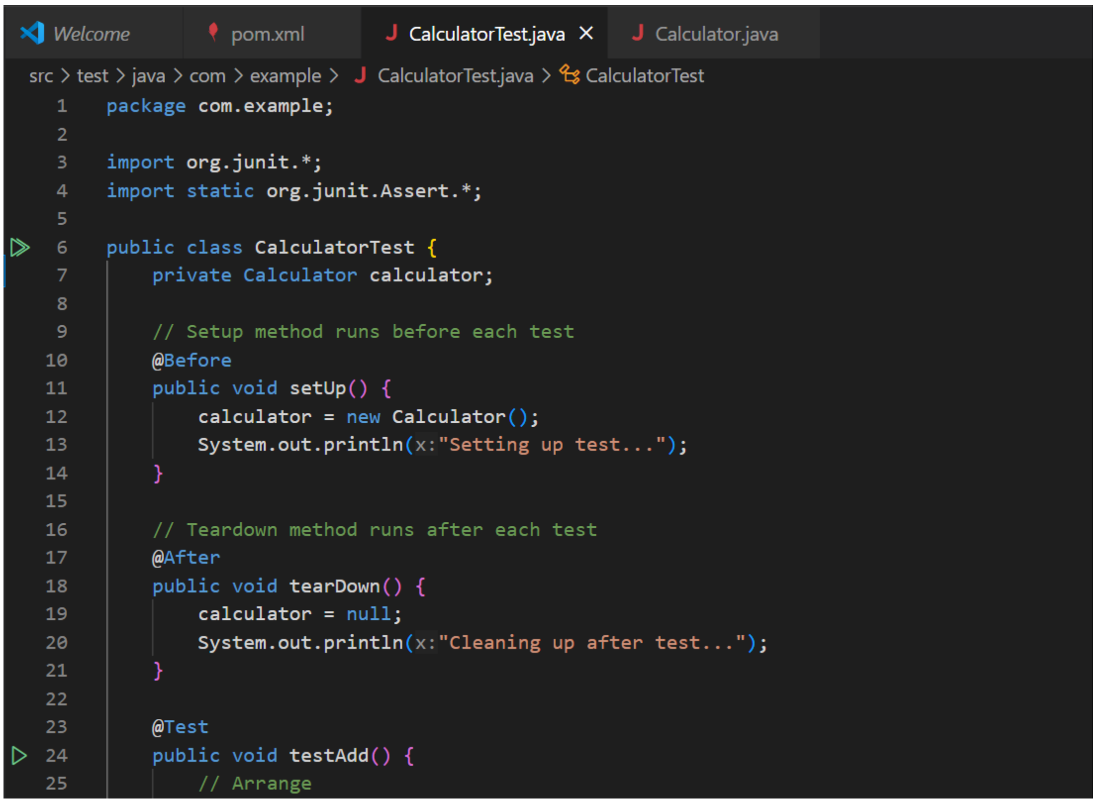
Calculator.java



Welcome
pom.xml
CalculatorTest.java
Calculator.java
src > main > java > com > example > Calculator.java > {} com.example

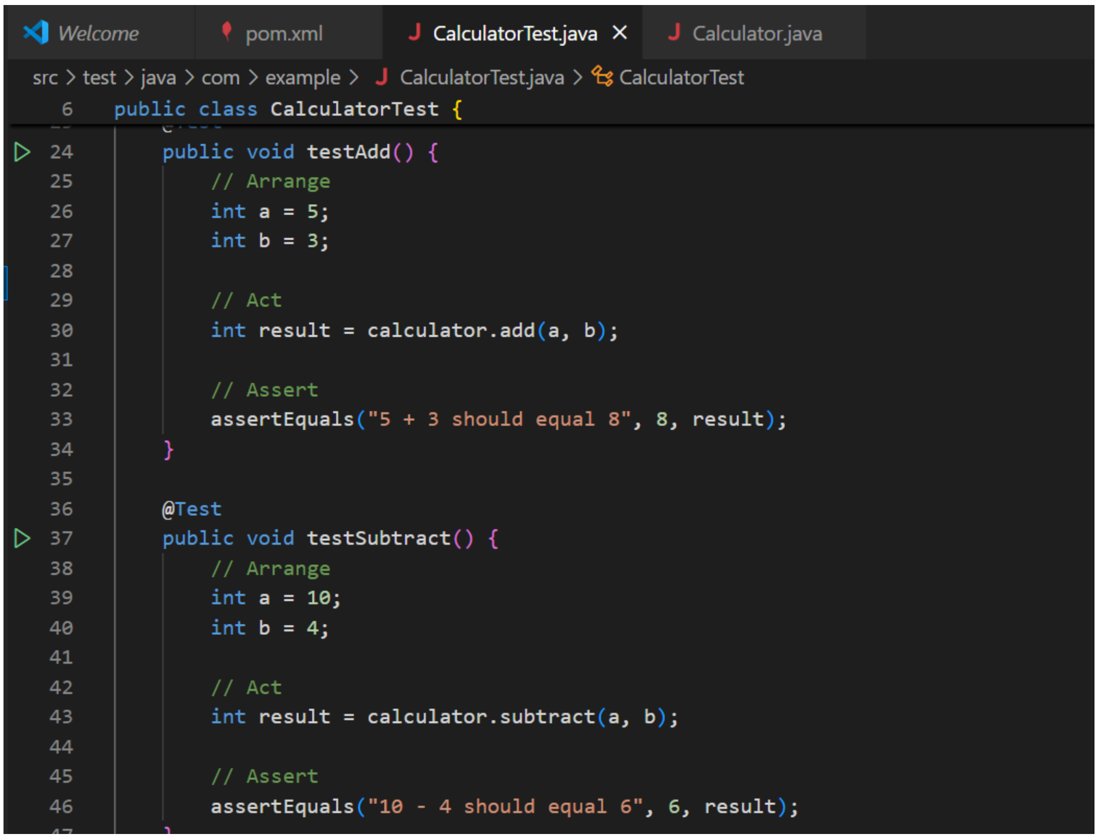
```java
package com.example;

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

    public int subtract(int a, int b) {
        return a - b;
    }

    public int multiply(int a, int b) {
        return a * b;
    }

    public double divide(int a, int b) {
        if (b == 0) {
            throw new IllegalArgumentException(s:"Cannot divide by zero");
        }
        return (double) a / b;
    }
}
```

CalculatorTest.java


Welcome | pom.xml | CalculatorTest.java | Calculator.java

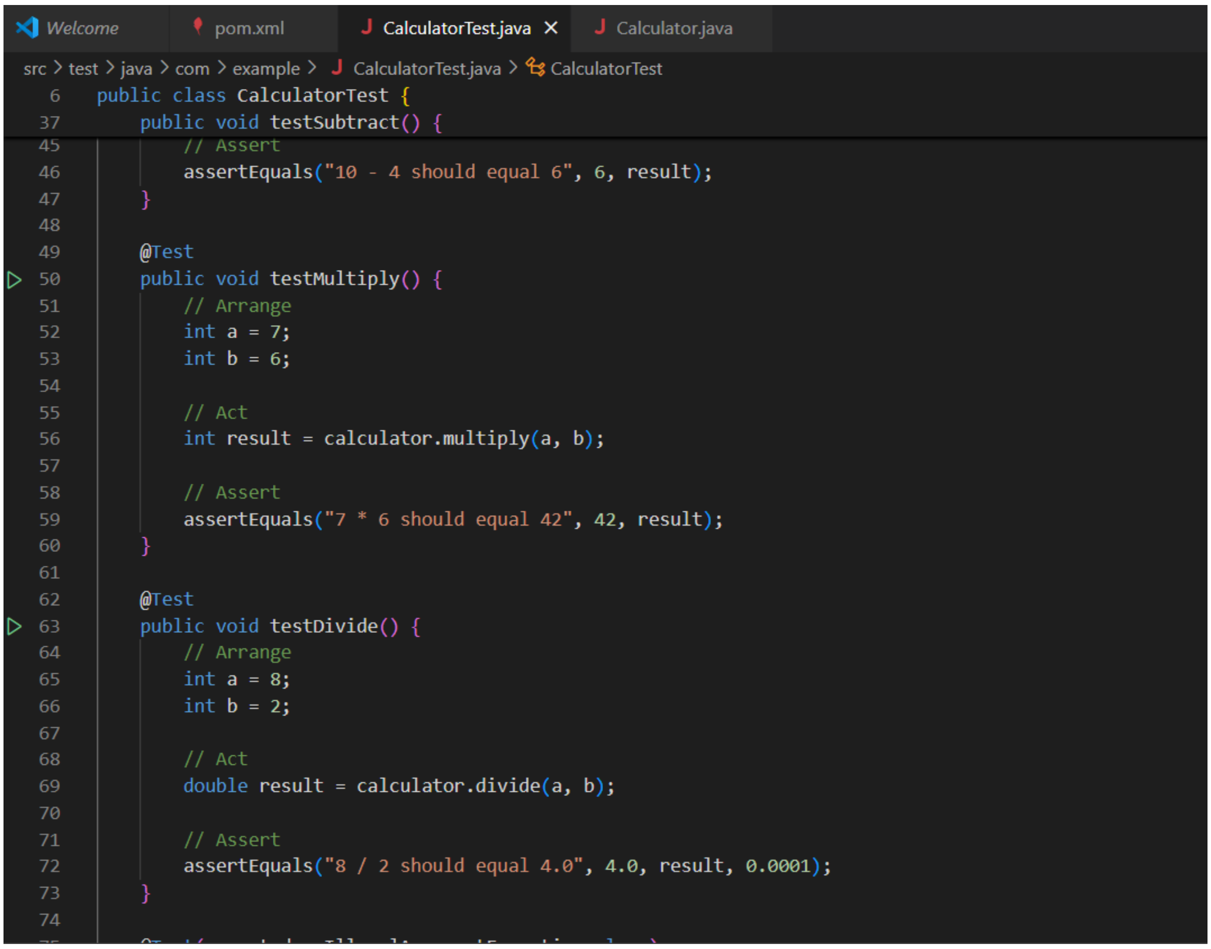
src > test > java > com > example > CalculatorTest.java > CalculatorTest

```java
package com.example;

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

public class CalculatorTest {
    private Calculator calculator;

    // Setup method runs before each test
    @Before
    public void setUp() {
        calculator = new Calculator();
        System.out.println(x:"Setting up test...");
    }

    // Teardown method runs after each test
    @After
    public void tearDown() {
        calculator = null;
        System.out.println(x:"Cleaning up after test...");
    }

    @Test
    public void testAdd() {
        // Arrange
```

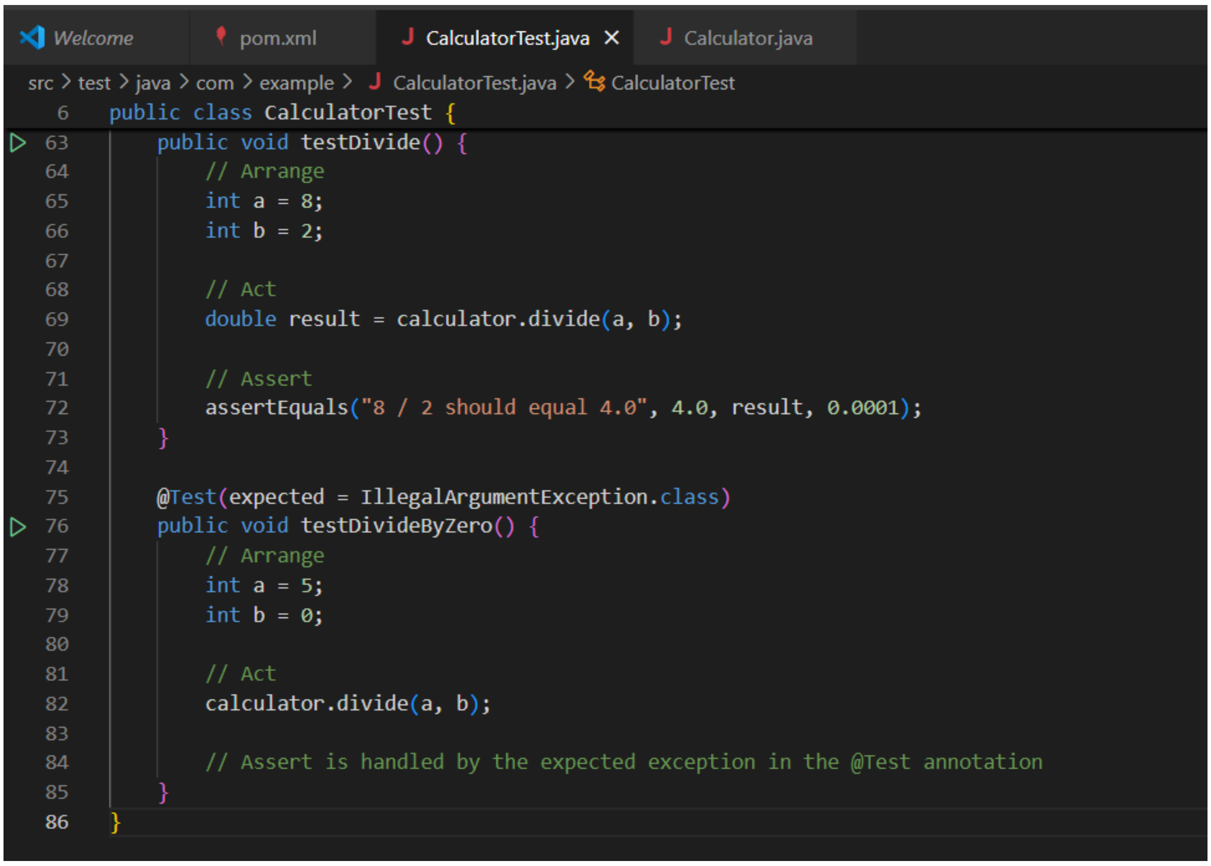

Welcome | pom.xml | CalculatorTest.java | Calculator.java

src > test > java > com > example > CalculatorTest.java > CalculatorTest

```java
public class CalculatorTest {
    public void testAdd() {
        // Arrange
        int a = 5;
        int b = 3;

        // Act
        int result = calculator.add(a, b);

        // Assert
        assertEquals("5 + 3 should equal 8", 8, result);
    }

    @Test
    public void testSubtract() {
        // Arrange
        int a = 10;
        int b = 4;

        // Act
        int result = calculator.subtract(a, b);

        // Assert
        assertEquals("10 - 4 should equal 6", 6, result);
```

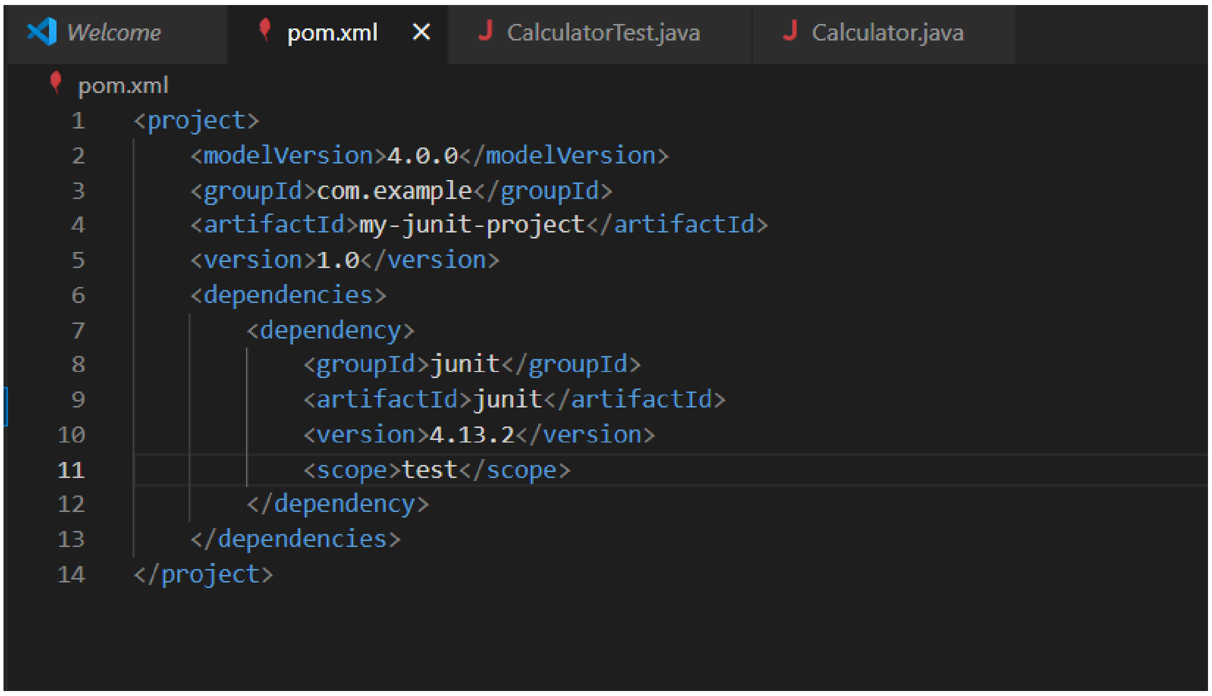

Welcome
pom.xml
CalculatorTest.java
Calculator.java
src > test > java > com > example > CalculatorTest.java > CalculatorTest

```java
public class CalculatorTest {
    public void testSubtract() {
        // Assert
        assertEquals("10 - 4 should equal 6", 6, result);
    }

    @Test
    public void testMultiply() {
        // Arrange
        int a = 7;
        int b = 6;

        // Act
        int result = calculator.multiply(a, b);

        // Assert
        assertEquals("7 * 6 should equal 42", 42, result);
    }

    @Test
    public void testDivide() {
        // Arrange
        int a = 8;
        int b = 2;

        // Act
        double result = calculator.divide(a, b);

        // Assert
        assertEquals("8 / 2 should equal 4.0", 4.0, result, 0.0001);
    }

```


Welcome
pom.xml
CalculatorTest.java
Calculator.java
src > test > java > com > example > CalculatorTest.java > CalculatorTest

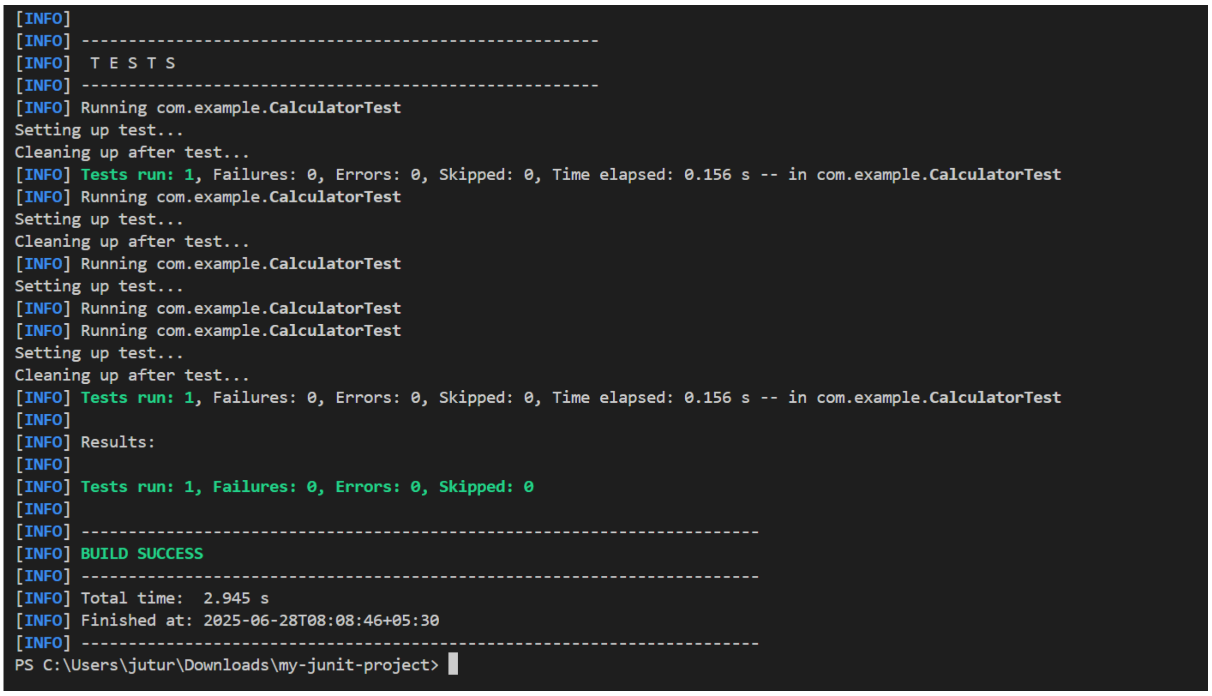
```java
public class CalculatorTest {
    public void testDivide() {
        // Arrange
        int a = 8;
        int b = 2;

        // Act
        double result = calculator.divide(a, b);

        // Assert
        assertEquals("8 / 2 should equal 4.0", 4.0, result, 0.0001);
    }

    @Test(expected = IllegalArgumentException.class)
    public void testDivideByZero() {
        // Arrange
        int a = 5;
        int b = 0;

        // Act
        calculator.divide(a, b);

        // Assert is handled by the expected exception in the @Test annotation
    }
}
```

Pom.xml



```xml
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>my-junit-project</artifactId>
    <version>1.0</version>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

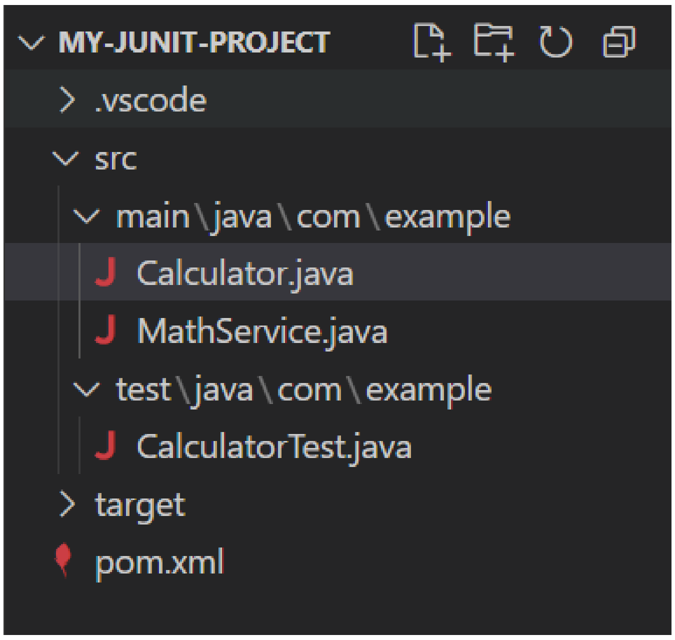
Output:



```
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.CalculatorTest
Setting up test...
Cleaning up after test...
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.156 s -- in com.example.CalculatorTest
[INFO] Running com.example.CalculatorTest
Setting up test...
Cleaning up after test...
[INFO] Running com.example.CalculatorTest
Setting up test...
[INFO] Running com.example.CalculatorTest
[INFO] Running com.example.CalculatorTest
Setting up test...
Cleaning up after test...
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.156 s -- in com.example.CalculatorTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.945 s
[INFO] Finished at: 2025-06-28T08:08:46+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\jutur\Downloads\my-junit-project>
```

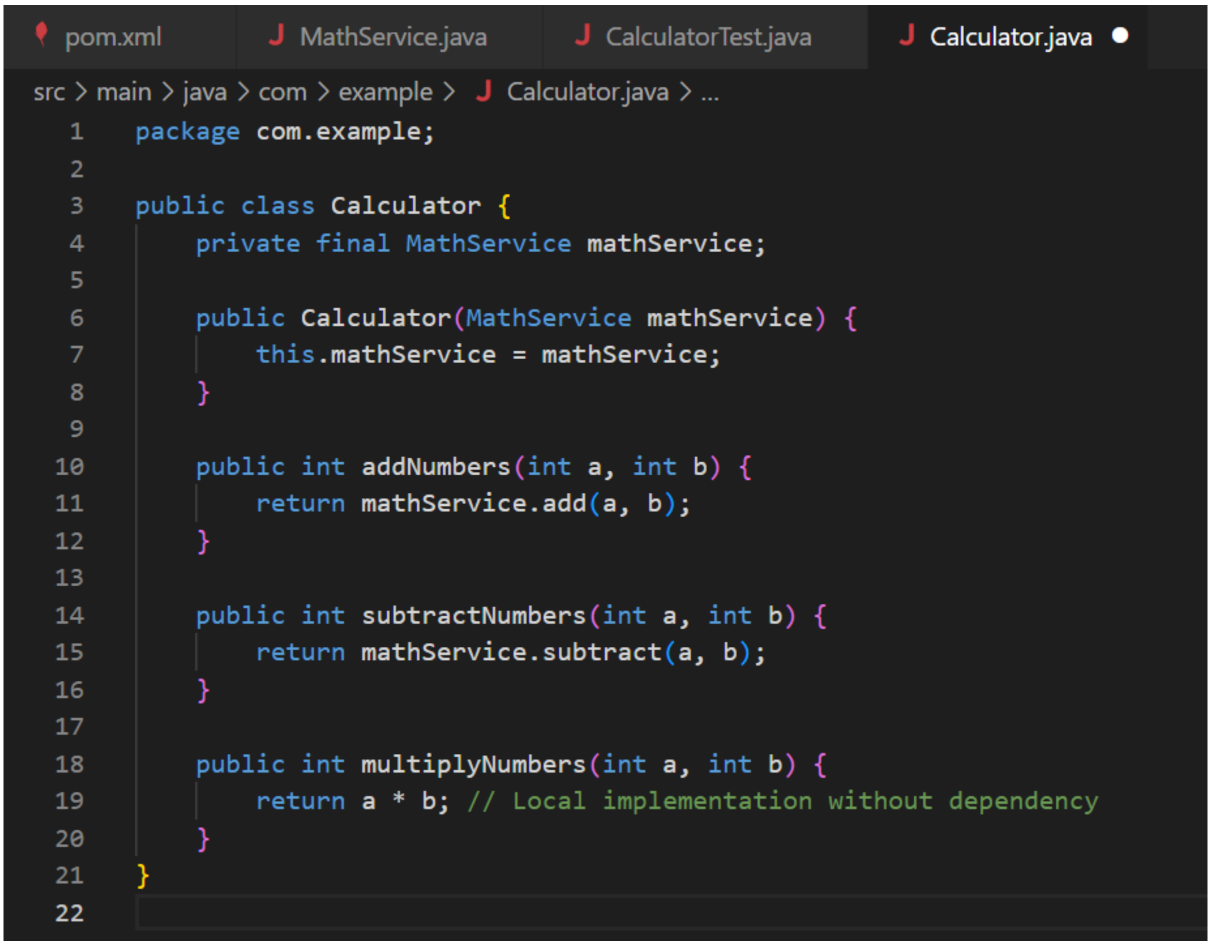
6. Exercise 1: Mocking and Stubbing

Setup:



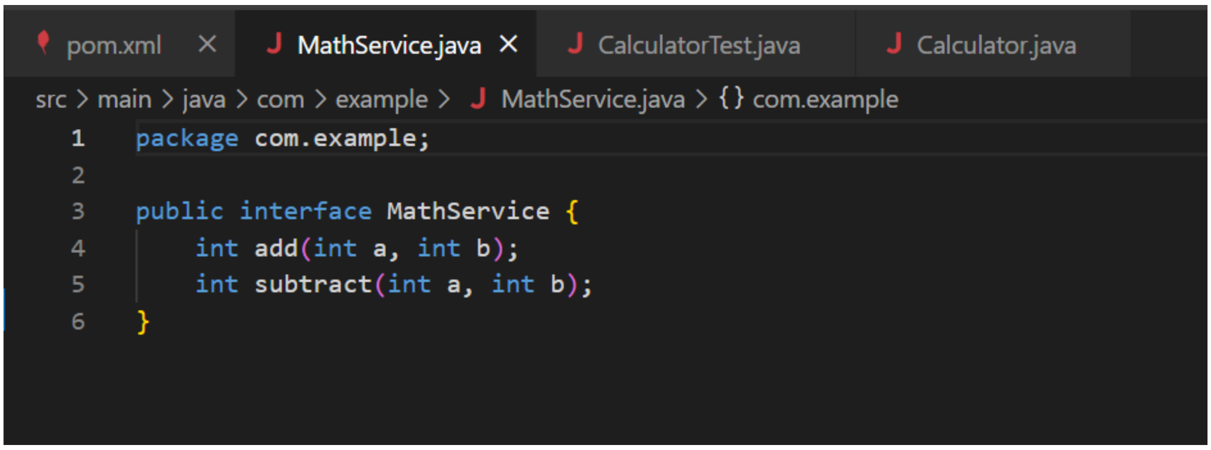
```
MY-JUNIT-PROJECT
  > .vscode
  v src
    v main\java\com\example
        Calculator.java
        MathService.java
    v test\java\com\example
        CalculatorTest.java
  > target
    pom.xml
```

Calculator.java



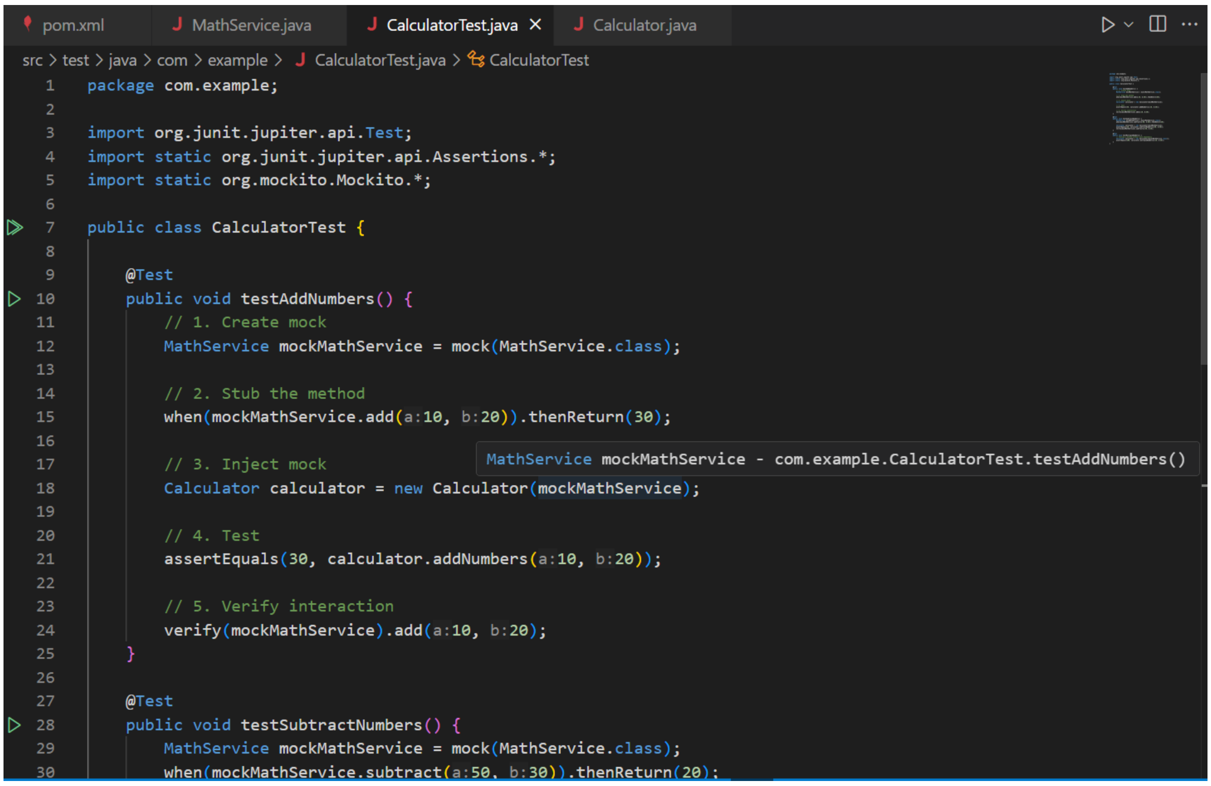
```java
package com.example;

public class Calculator {
    private final MathService mathService;

    public Calculator(MathService mathService) {
        this.mathService = mathService;
    }

    public int addNumbers(int a, int b) {
        return mathService.add(a, b);
    }

    public int subtractNumbers(int a, int b) {
        return mathService.subtract(a, b);
    }

    public int multiplyNumbers(int a, int b) {
        return a * b; // Local implementation without dependency
    }
}

```

MathService.java

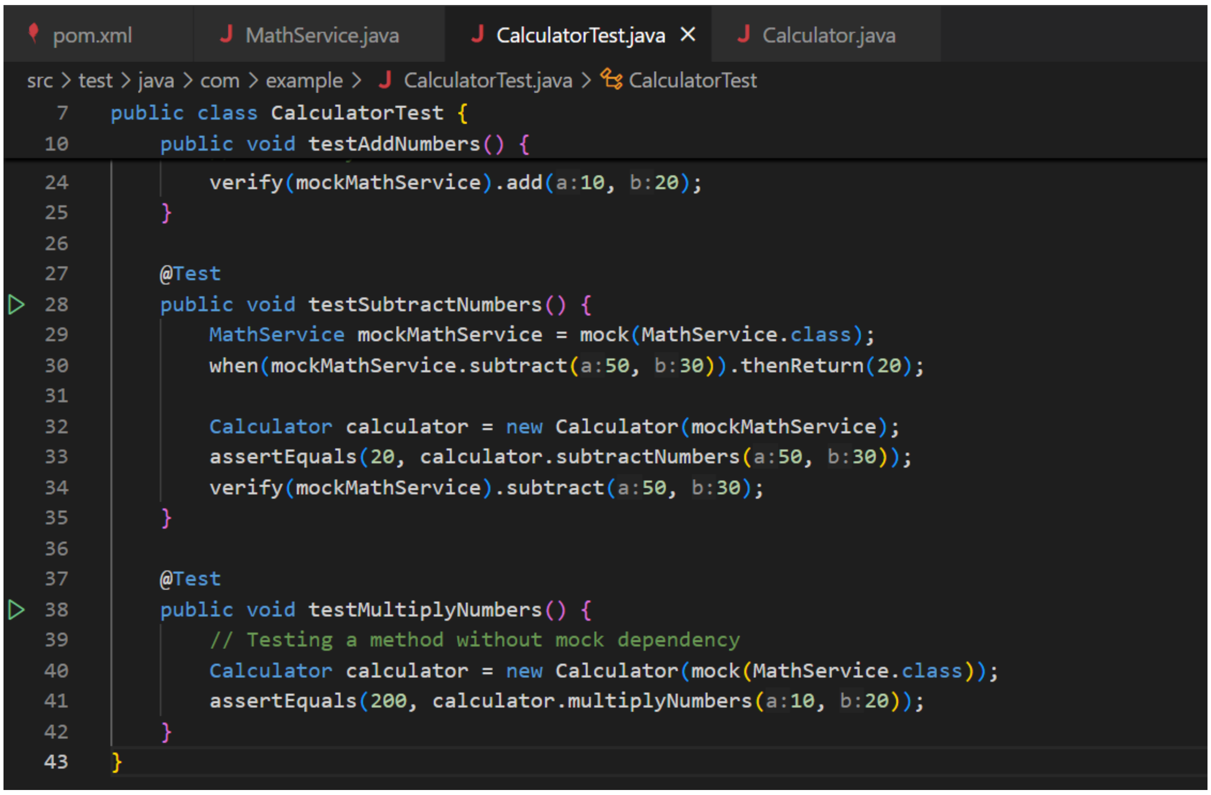

pom.xml
MathService.java
CalculatorTest.java
Calculator.java
src > main > java > com > example > MathService.java > {} com.example

```java
package com.example;

public interface MathService {
    int add(int a, int b);
    int subtract(int a, int b);
}
```

CalculatorTest.java

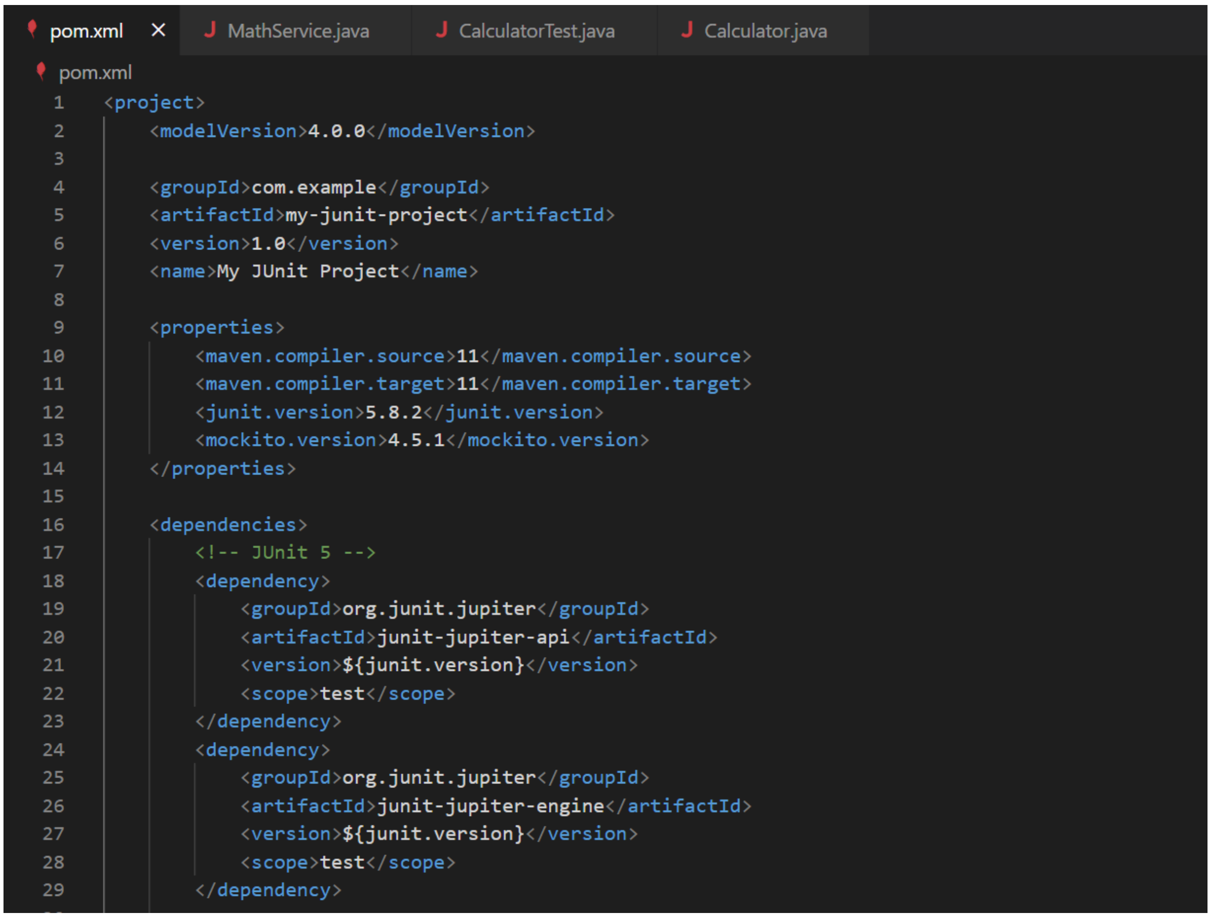

pom.xml
MathService.java
CalculatorTest.java
Calculator.java
src > test > java > com > example > CalculatorTest.java > CalculatorTest
MathService mockMathService - com.example.CalculatorTest.testAddNumbers()

```java
package com.example;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;

public class CalculatorTest {

    @Test
    public void testAddNumbers() {
        // 1. Create mock
        MathService mockMathService = mock(MathService.class);

        // 2. Stub the method
        when(mockMathService.add(a:10, b:20)).thenReturn(30);

        // 3. Inject mock
        Calculator calculator = new Calculator(mockMathService);

        // 4. Test
        assertEquals(30, calculator.addNumbers(a:10, b:20));

        // 5. Verify interaction
        verify(mockMathService).add(a:10, b:20);
    }

    @Test
    public void testSubtractNumbers() {
        MathService mockMathService = mock(MathService.class);
        when(mockMathService.subtract(a:50, b:30)).thenReturn(20);
```

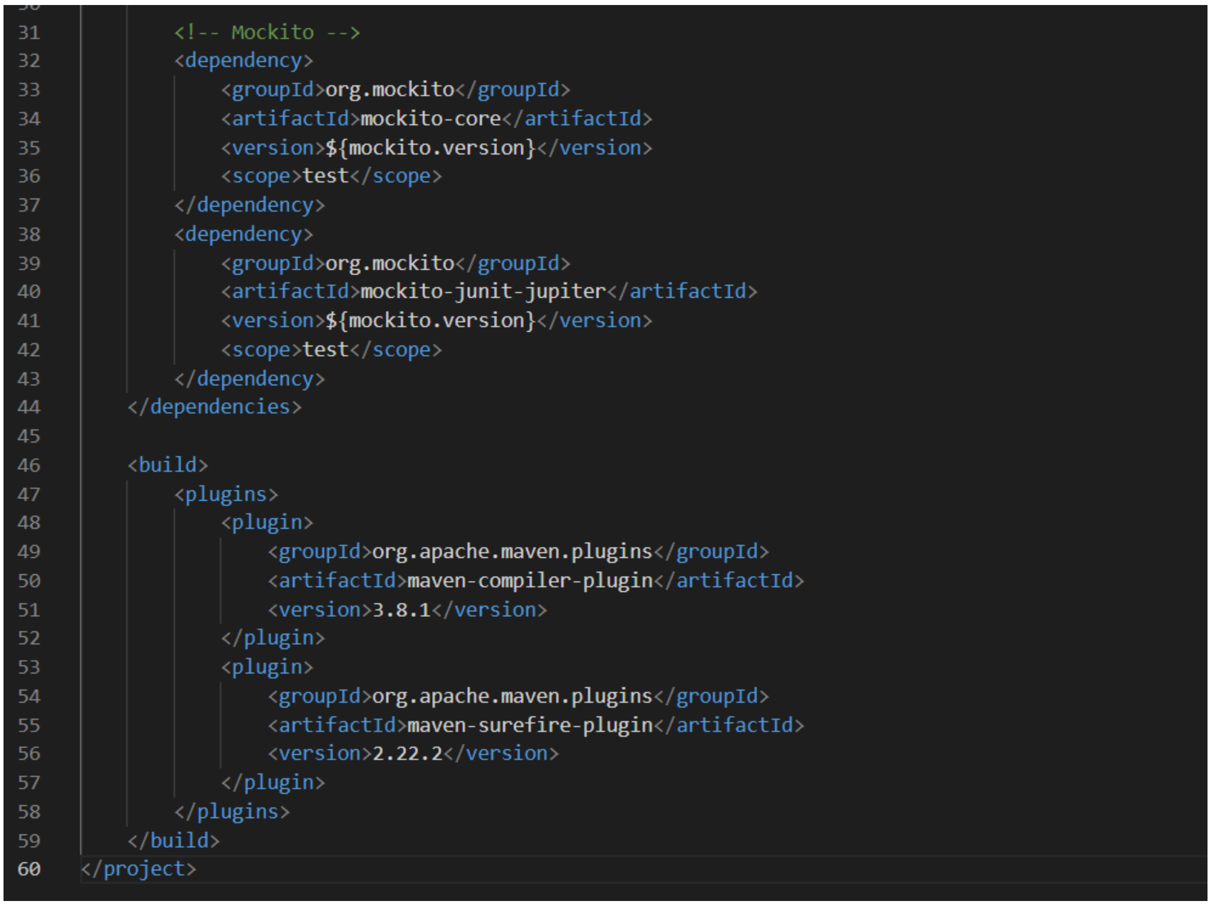

pom.xml
MathService.java
CalculatorTest.java
Calculator.java
src > test > java > com > example > CalculatorTest.java > CalculatorTest

```java
public class CalculatorTest {
    public void testAddNumbers() {
        verify(mockMathService).add(a:10, b:20);
    }

    @Test
    public void testSubtractNumbers() {
        MathService mockMathService = mock(MathService.class);
        when(mockMathService.subtract(a:50, b:30)).thenReturn(20);

        Calculator calculator = new Calculator(mockMathService);
        assertEquals(20, calculator.subtractNumbers(a:50, b:30));
        verify(mockMathService).subtract(a:50, b:30);
    }

    @Test
    public void testMultiplyNumbers() {
        // Testing a method without mock dependency
        Calculator calculator = new Calculator(mock(MathService.class));
        assertEquals(200, calculator.multiplyNumbers(a:10, b:20));
    }
}
```

pom.xml

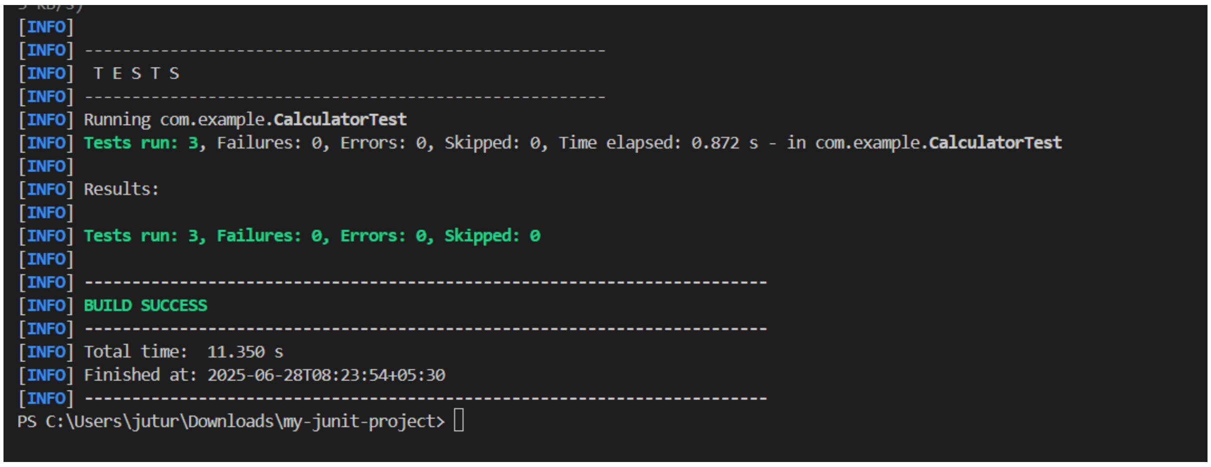


```xml
<project>
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>my-junit-project</artifactId>
    <version>1.0</version>
    <name>My JUnit Project</name>

    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
        <junit.version>5.8.2</junit.version>
        <mockito.version>4.5.1</mockito.version>
    </properties>

    <dependencies>
        <!-- JUnit 5 -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
```



```xml
        <!-- Mockito -->
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>${mockito.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-junit-jupiter</artifactId>
            <version>${mockito.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.22.2</version>
            </plugin>
        </plugins>
    </build>
</project>
```

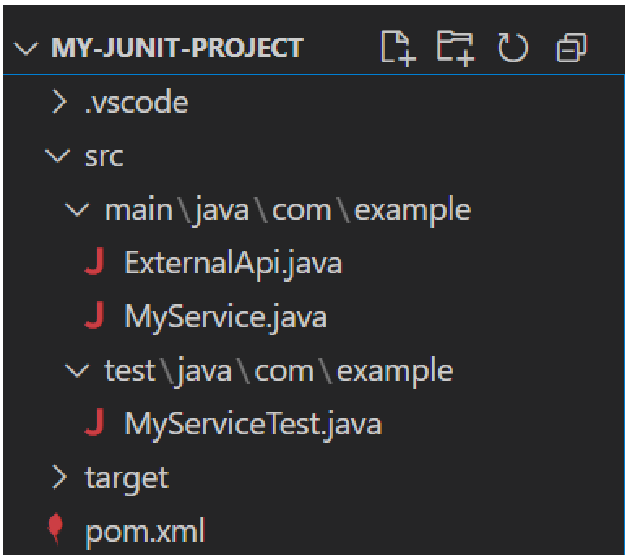
Output:



```
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.CalculatorTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.872 s - in com.example.CalculatorTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  11.350 s
[INFO] Finished at: 2025-06-28T08:23:54+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\jutur\Downloads\my-junit-project>
```

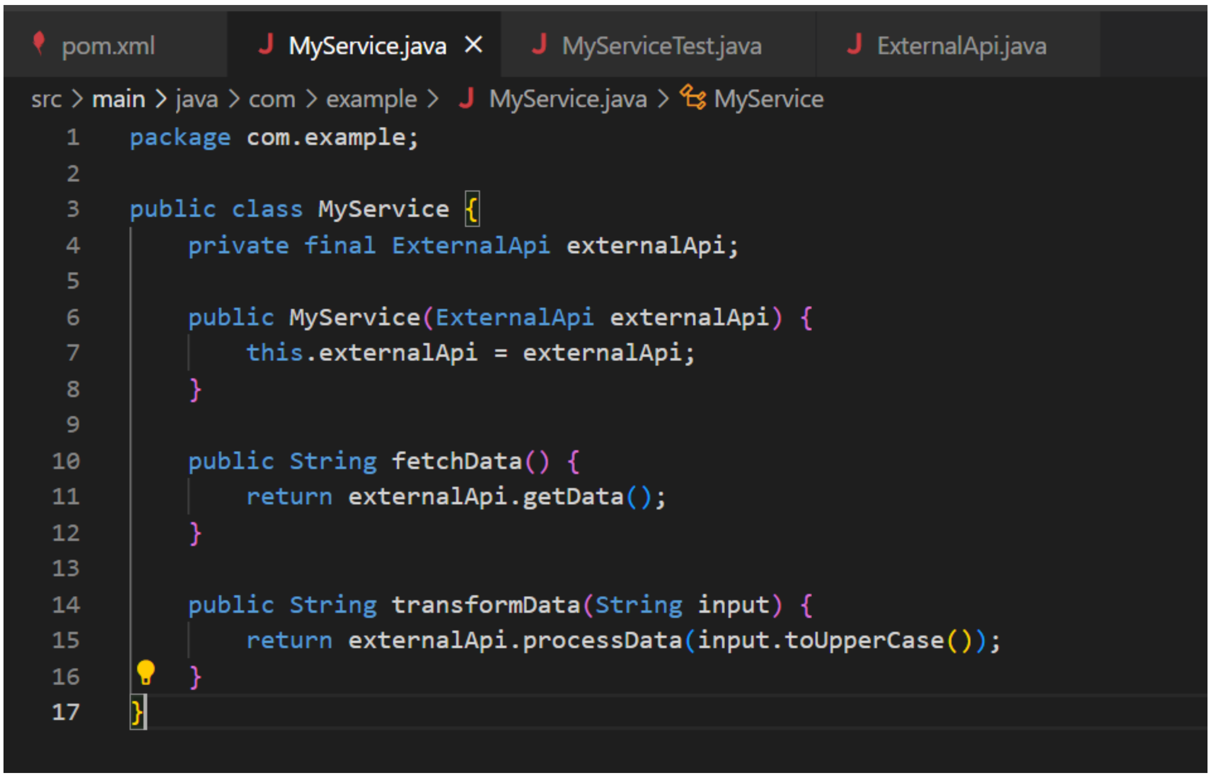
7. Exercise 2: Verifying Interactions

Setup:



```
MY-JUNIT-PROJECT
  > .vscode
  v src
    v main\java\com\example
      ExternalApi.java
      MyService.java
    v test\java\com\example
      MyServiceTest.java
  > target
  pom.xml
```

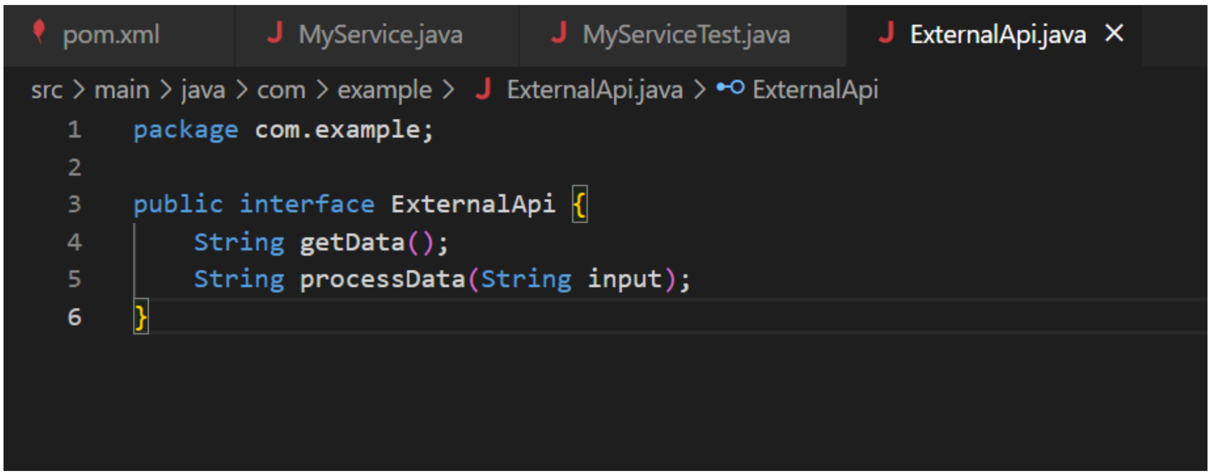
MyService.java



pom.xml
MyService.java
MyServiceTest.java
ExternalApi.java
src > main > java > com > example > MyService.java > MyService

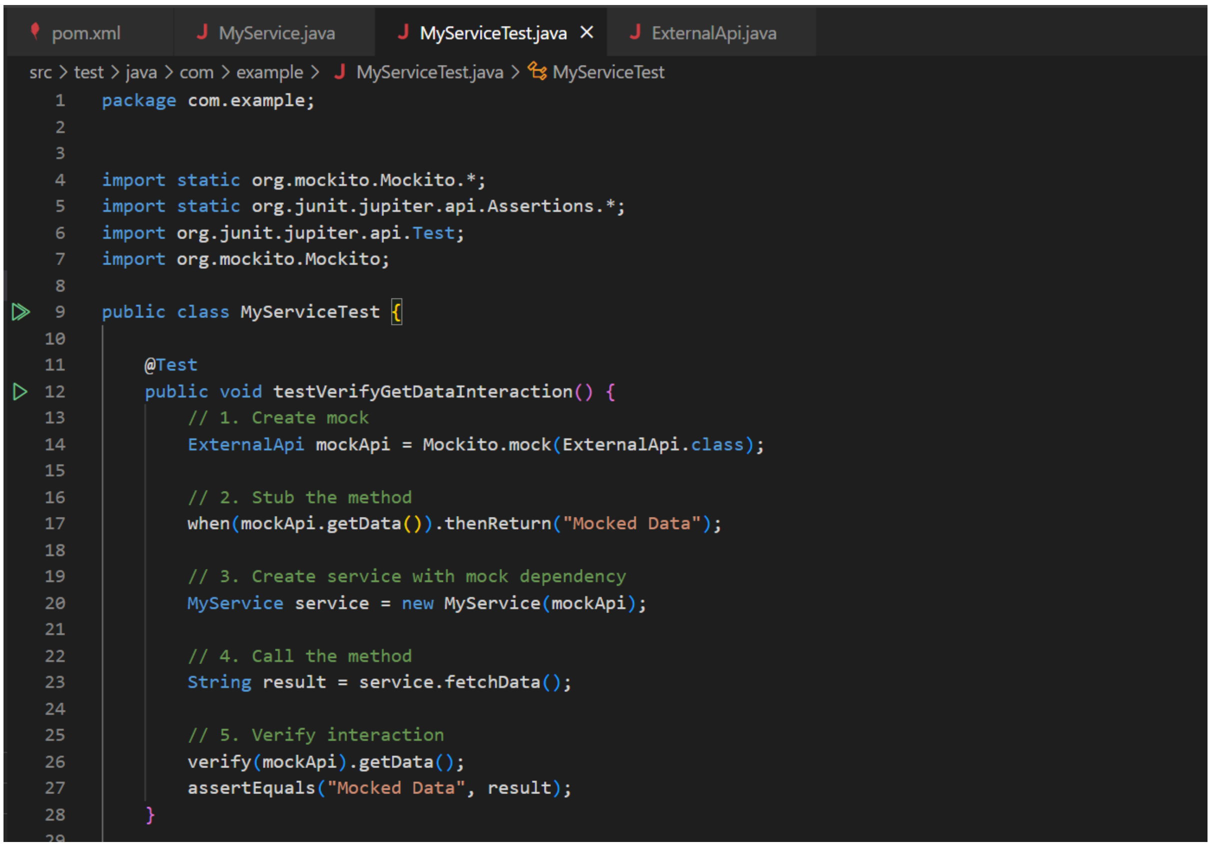
```java
package com.example;

public class MyService {
    private final ExternalApi externalApi;

    public MyService(ExternalApi externalApi) {
        this.externalApi = externalApi;
    }

    public String fetchData() {
        return externalApi.getData();
    }

    public String transformData(String input) {
        return externalApi.processData(input.toUpperCase());
    }
}
```

ExternalApi.java

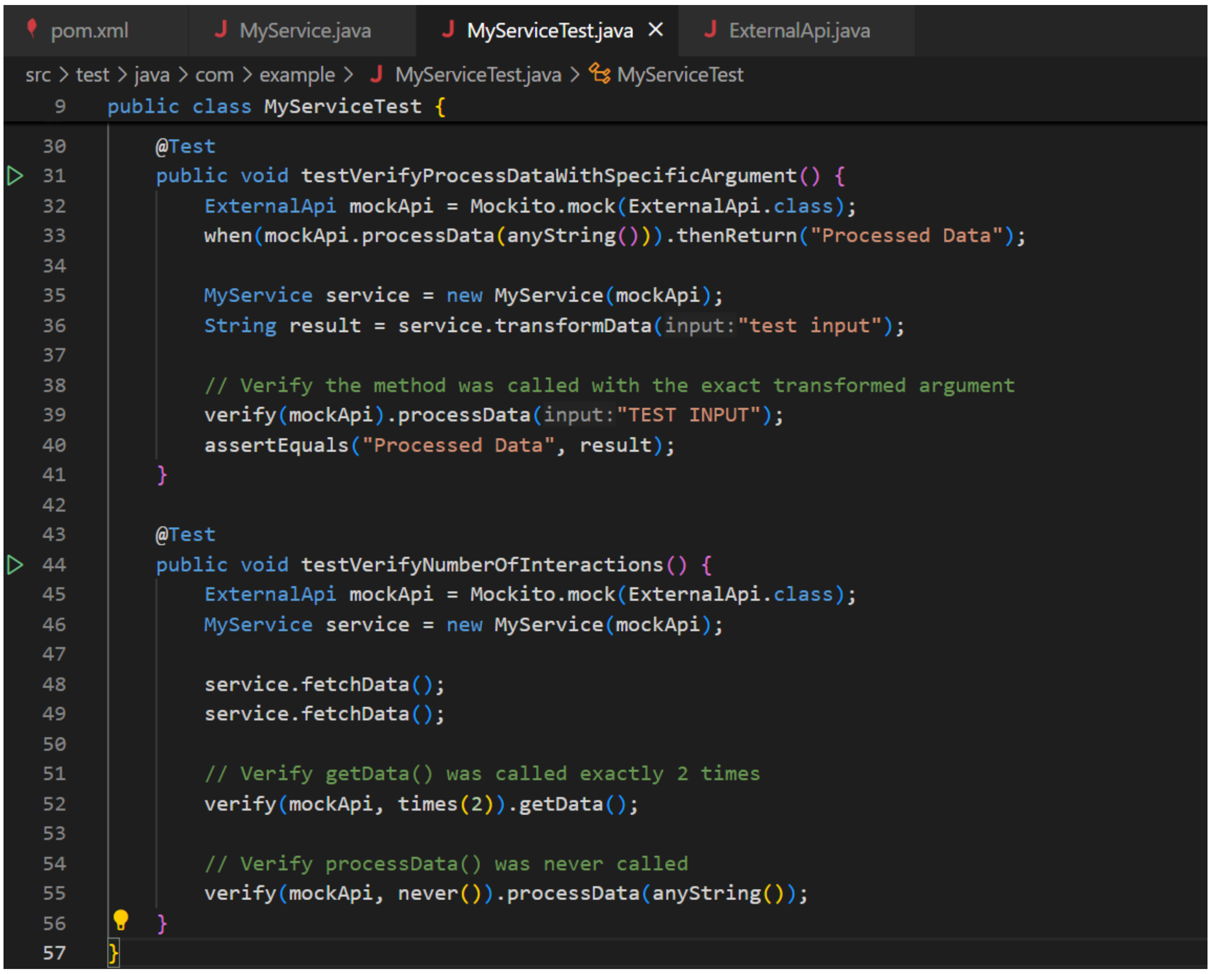

pom.xml
MyService.java
MyServiceTest.java
ExternalApi.java
src > main > java > com > example > ExternalApi.java > ExternalApi

```java
package com.example;

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

MyServiceTest.java

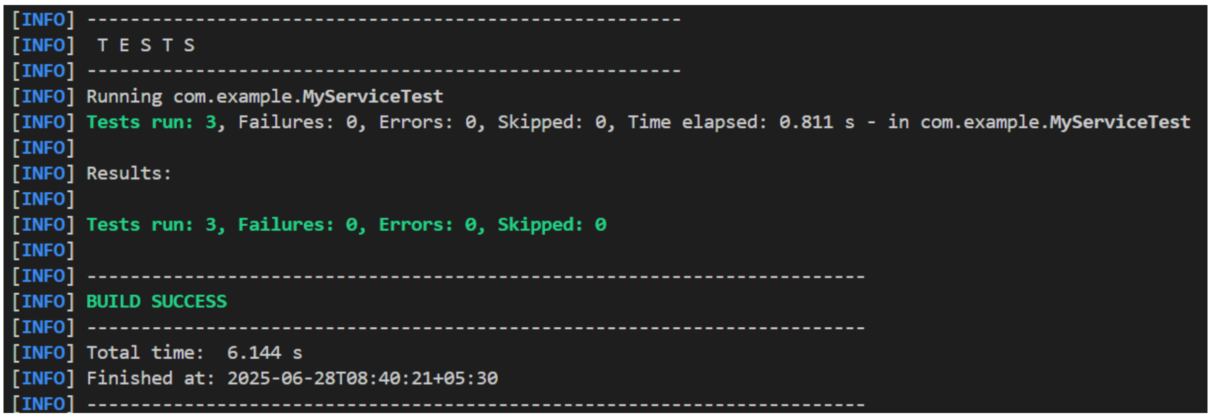

pom.xml
MyService.java
MyServiceTest.java
ExternalApi.java
src > test > java > com > example > MyServiceTest.java > MyServiceTest

```java
package com.example;


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 testVerifyGetDataInteraction() {
        // 1. Create mock
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);

        // 2. Stub the method
        when(mockApi.getData()).thenReturn("Mocked Data");

        // 3. Create service with mock dependency
        MyService service = new MyService(mockApi);

        // 4. Call the method
        String result = service.fetchData();

        // 5. Verify interaction
        verify(mockApi).getData();
        assertEquals("Mocked Data", result);
    }
```


pom.xml
MyService.java
MyServiceTest.java
ExternalApi.java
src > test > java > com > example > MyServiceTest.java > MyServiceTest

```java
public class MyServiceTest {

    @Test
    public void testVerifyProcessDataWithSpecificArgument() {
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);
        when(mockApi.processData(anyString())).thenReturn("Processed Data");

        MyService service = new MyService(mockApi);
        String result = service.transformData(input:"test input");

        // Verify the method was called with the exact transformed argument
        verify(mockApi).processData(input:"TEST INPUT");
        assertEquals("Processed Data", result);
    }

    @Test
    public void testVerifyNumberOfInteractions() {
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);
        MyService service = new MyService(mockApi);

        service.fetchData();
        service.fetchData();

        // Verify getData() was called exactly 2 times
        verify(mockApi, times(2)).getData();

        // Verify processData() was never called
        verify(mockApi, never()).processData(anyString());
    }
}
```

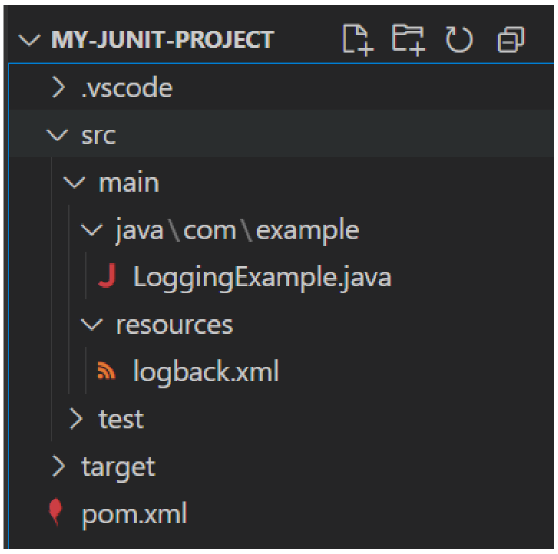
Output:



```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.MyServiceTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.811 s - in com.example.MyServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.144 s
[INFO] Finished at: 2025-06-28T08:40:21+05:30
[INFO] ------------------------------------------------------------------------
```

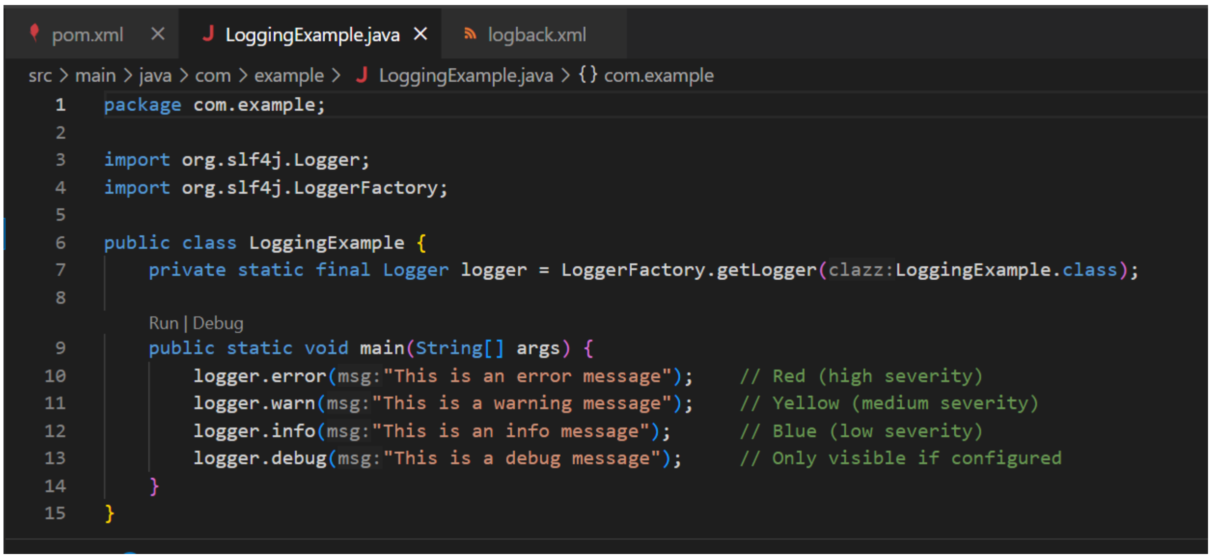
8. Exercise 1: Logging Error Messages and Warning Levels

Setup:



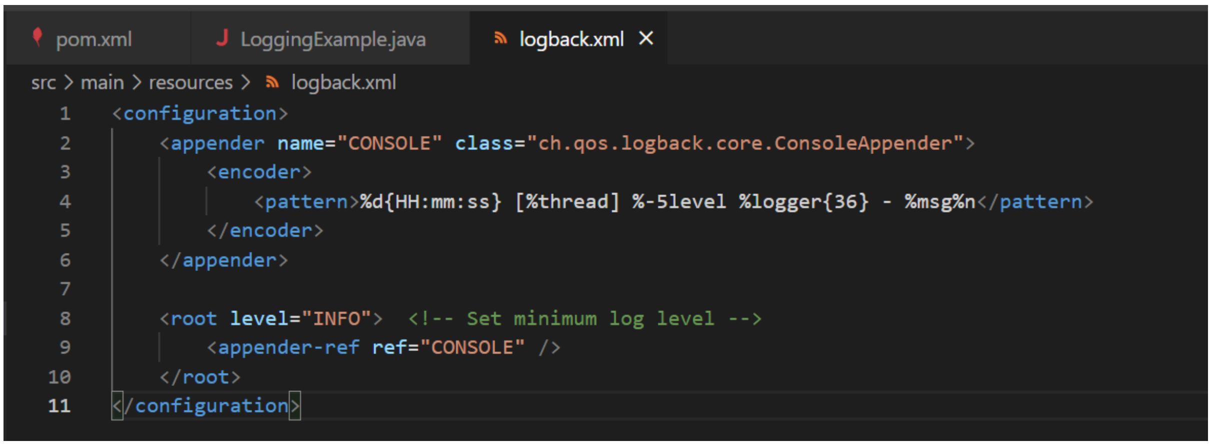
```
MY-JUNIT-PROJECT
> .vscode
v src
  v main
    v java\com\example
      LoggingExample.java
    v resources
      logback.xml
  > test
> target
pom.xml
```

LoggingExample.java


pom.xml
LoggingExample.java
logback.xml
src > main > java > com > example > LoggingExample.java > {} com.example
Run | Debug

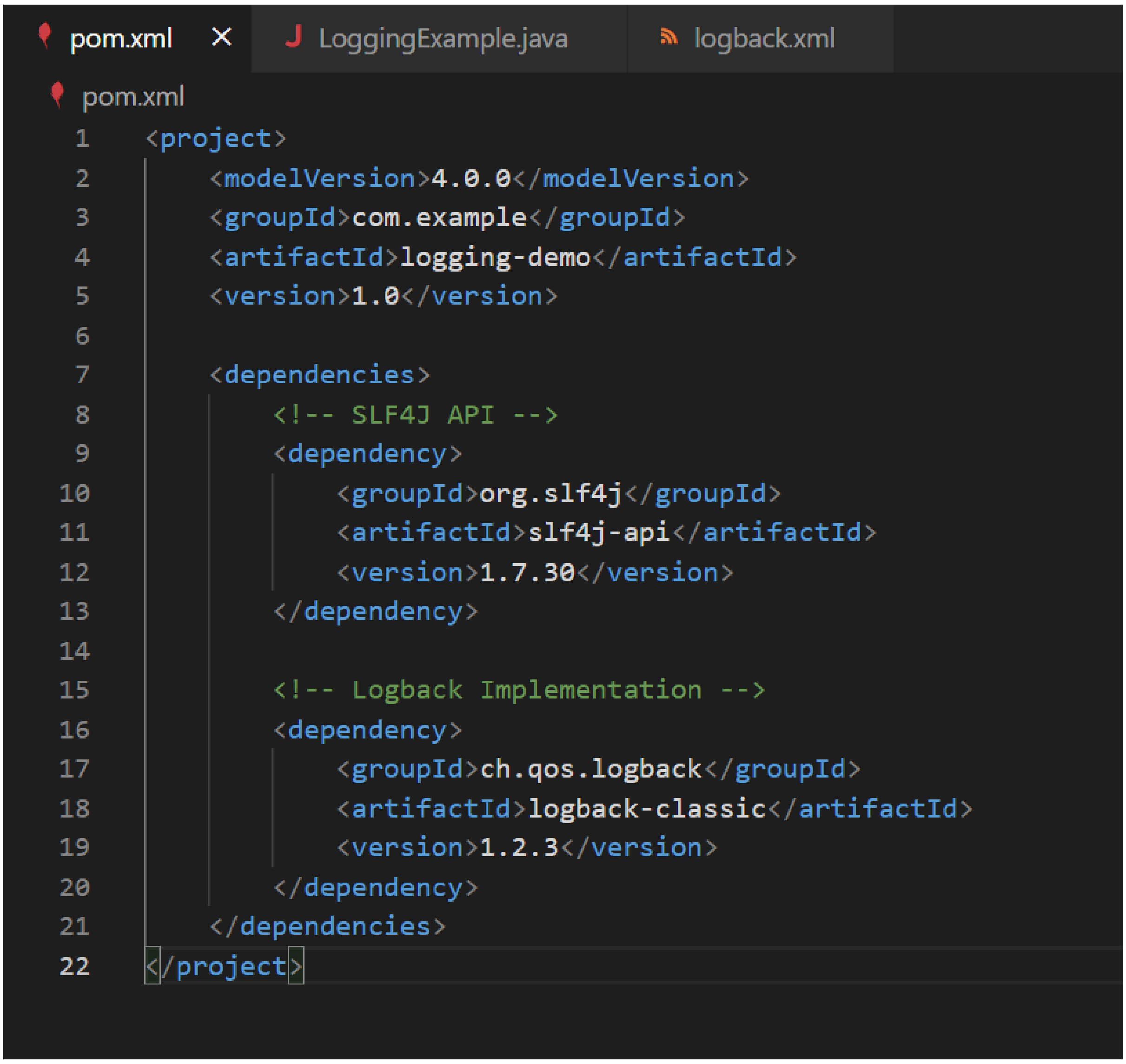
```java
package com.example;

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");   // Red (high severity)
        logger.warn(msg:"This is a warning message");   // Yellow (medium severity)
        logger.info(msg:"This is an info message");     // Blue (low severity)
        logger.debug(msg:"This is a debug message");    // Only visible if configured
    }
}
```

logback.xml



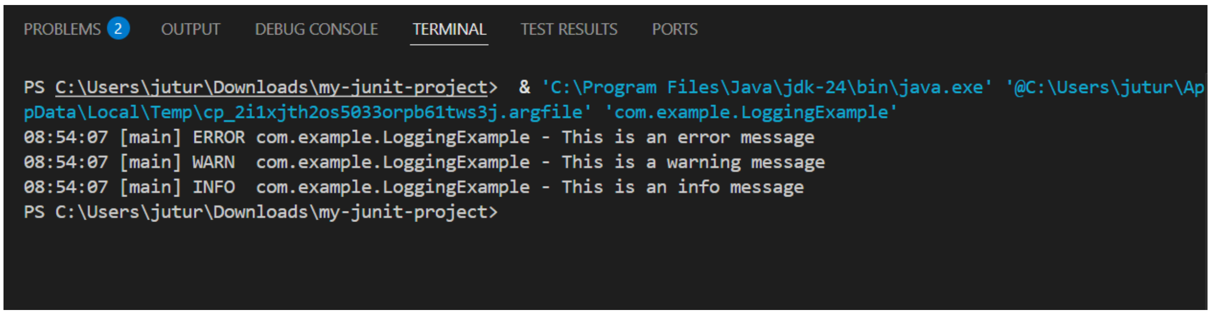
```xml
<configuration>
    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>

    <root level="INFO">  <!-- Set minimum log level -->
        <appender-ref ref="CONSOLE" />
    </root>
</configuration>
```

pom.xml



```xml
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>logging-demo</artifactId>
    <version>1.0</version>

    <dependencies>
        <!-- SLF4J API -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.7.30</version>
        </dependency>

        <!-- Logback Implementation -->
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
            <version>1.2.3</version>
        </dependency>
    </dependencies>
</project>
```

Output:



```
PS C:\Users\jutur\Downloads\my-junit-project>  & 'C:\Program Files\Java\jdk-24\bin\java.exe' '@C:\Users\jutur\Ap
pData\Local\Temp\cp_2i1xjth2os5033orpb61tws3j.argfile' 'com.example.LoggingExample'
08:54:07 [main] ERROR com.example.LoggingExample - This is an error message
08:54:07 [main] WARN  com.example.LoggingExample - This is a warning message
08:54:07 [main] INFO  com.example.LoggingExample - This is an info message
PS C:\Users\jutur\Downloads\my-junit-project>
```