<u>Digital Nurture – 4.0 JAVA FSE – Week 2 Assignments</u>

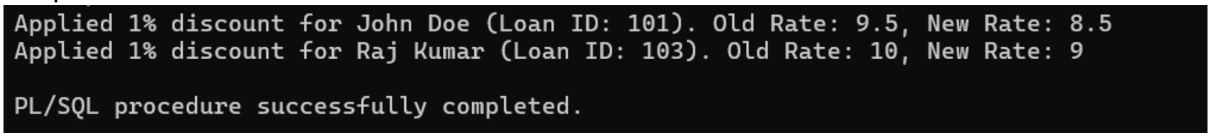
<u>File name:</u> PLSQL_Exercises
<u>Name:</u> Exercise 1: Control Structures

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

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

INSERT INTO Customers VALUES (1, 'John Doe', 65, 15000.00, 'FALSE');
INSERT INTO Customers VALUES (2, 'Jane Smith', 45, 8000.00, 'FALSE');
INSERT INTO Customers VALUES (3, 'Raj Kumar', 70, 10500.00, 'FALSE');
INSERT INTO Customers VALUES (4, 'Anita Rao', 35, 12000.00, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 10.5, SYSDATE + 10);
INSERT INTO Loans VALUES (102, 2, 9.0, SYSDATE + 45);
INSERT INTO Loans VALUES (103, 3, 11.0, SYSDATE + 20);
INSERT INTO Loans VALUES (104, 4, 8.5, SYSDATE + 5);
COMMIT;
```

Scenario 1

```
SET SERVEROUTPUT ON;

DECLARE
  v_old_rate Loans.InterestRate%TYPE;
BEGIN
  FOR rec IN (
    SELECT l.LoanID, l.InterestRate, c.Name
    FROM Loans l
    JOIN Customers c ON l.CustomerID = c.CustomerID
    WHERE c.Age > 60
  )
  LOOP
    v_old_rate := rec.InterestRate;

    UPDATE Loans
    SET InterestRate = InterestRate - 1
    WHERE LoanID = rec.LoanID;

    DBMS_OUTPUT.PUT_LINE('Applied 1% discount for ' || rec.Name ||
                         ' (Loan ID: ' || rec.LoanID || '). ' ||
                         'Old Rate: ' || v_old_rate ||
                         ', New Rate: ' || (v_old_rate - 1));
  END LOOP;
  COMMIT;
END;
/
```

Output:



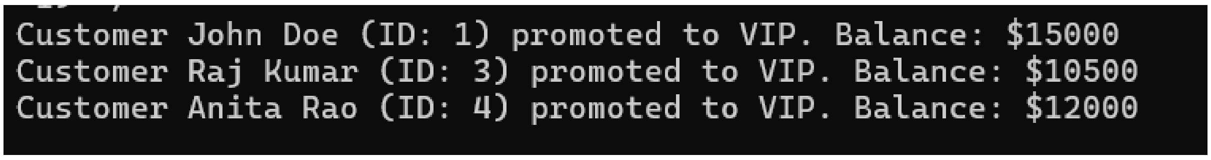
```
Applied 1% discount for John Doe (Loan ID: 101). Old Rate: 9.5, New Rate: 8.5
Applied 1% discount for Raj Kumar (Loan ID: 103). Old Rate: 10, New Rate: 9

PL/SQL procedure successfully completed.
```

Scenario 2

```
SET SERVEROUTPUT ON;

BEGIN
  FOR rec IN (
    SELECT CustomerID, Name, Balance
    FROM Customers
    WHERE Balance > 10000 AND IsVIP != 'TRUE'
  )
  LOOP
    UPDATE Customers
    SET IsVIP = 'TRUE'
    WHERE CustomerID = rec.CustomerID;

    DBMS_OUTPUT.PUT_LINE('Customer ' || rec.Name ||
                         ' (ID: ' || rec.CustomerID ||
                         ') promoted to VIP. Balance: $' || rec.Balance);
  END LOOP;

  COMMIT;
```

```
END;
/
```

Output:

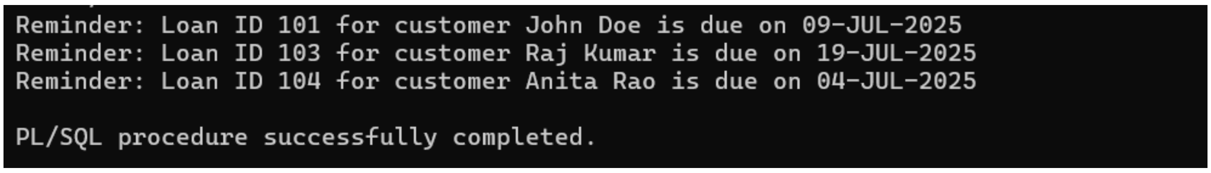


```
Customer John Doe (ID: 1) promoted to VIP. Balance: $15000
Customer Raj Kumar (ID: 3) promoted to VIP. Balance: $10500
Customer Anita Rao (ID: 4) promoted to VIP. Balance: $12000
```

Scenario 3:

```
SET SERVEROUTPUT ON;

DECLARE
  v_today      DATE := SYSDATE;
  v_due_limit  DATE := SYSDATE + 30;
BEGIN
  FOR rec IN (
    SELECT c.Name, l.LoanID, l.DueDate
    FROM Loans l
    JOIN Customers c ON l.CustomerID = c.CustomerID
    WHERE l.DueDate BETWEEN v_today AND v_due_limit
  )
  LOOP
    DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || rec.LoanID ||
                         ' for customer ' || rec.Name ||
                         ' is due on ' || TO_CHAR(rec.DueDate, 'DD-MON-YYYY'));
  END LOOP;
END;
/
```

Output:



```
Reminder: Loan ID 101 for customer John Doe is due on 09-JUL-2025
Reminder: Loan ID 103 for customer Raj Kumar is due on 19-JUL-2025
Reminder: Loan ID 104 for customer Anita Rao is due on 04-JUL-2025

PL/SQL procedure successfully completed.
```

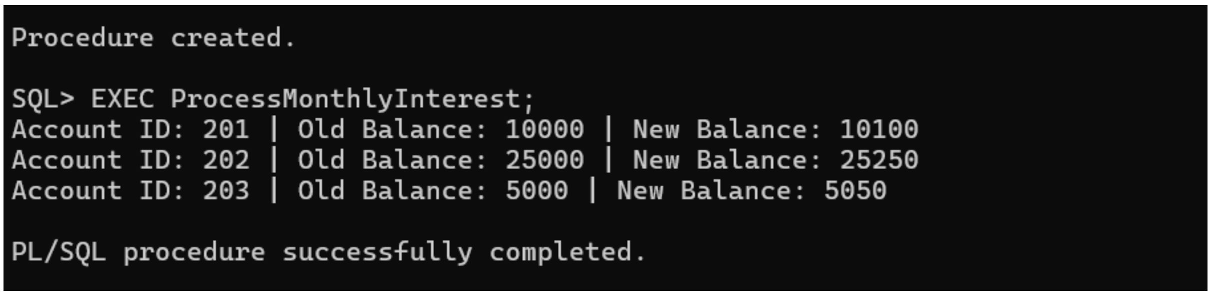
File name: PLSQL_Exercises

Name: Exercise 3: Stored Procedures

Scenario 1 (ProcessMonthlyInterest)

```
CREATE TABLE SavingsAccounts (
  AccountID   NUMBER PRIMARY KEY,
  CustomerID  NUMBER,
  Balance     NUMBER(12,2)
);

INSERT INTO SavingsAccounts VALUES (201, 1, 10000);
INSERT INTO SavingsAccounts VALUES (202, 2, 25000);
INSERT INTO SavingsAccounts VALUES (203, 3, 5000);
COMMIT;

SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
  v_new_balance SavingsAccounts.Balance%TYPE;
BEGIN
  FOR rec IN (
    SELECT AccountID, Balance FROM SavingsAccounts
  )
  LOOP
    v_new_balance := rec.Balance * 1.01;

    UPDATE SavingsAccounts
    SET Balance = v_new_balance
    WHERE AccountID = rec.AccountID;

    DBMS_OUTPUT.PUT_LINE('Account ID: ' || rec.AccountID ||
                         ' | Old Balance: ' || rec.Balance ||
                         ' | New Balance: ' || v_new_balance);
  END LOOP;

  COMMIT;
END;
/
EXEC ProcessMonthlyInterest;
```

Output:

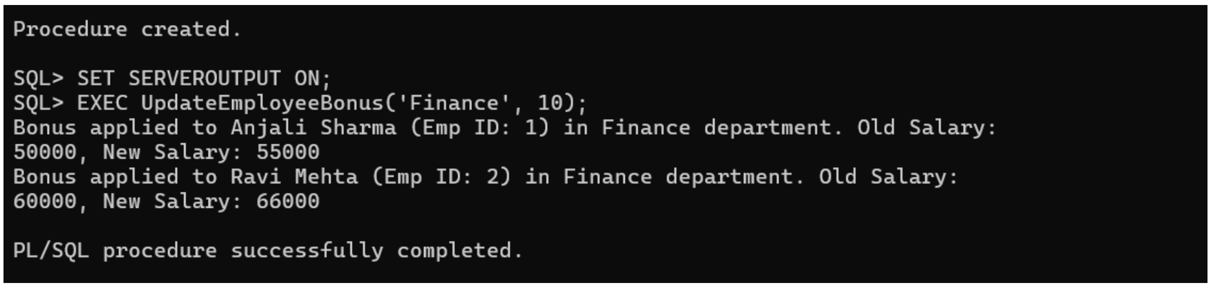


```
Procedure created.

SQL> EXEC ProcessMonthlyInterest;
Account ID: 201 | Old Balance: 10000 | New Balance: 10100
Account ID: 202 | Old Balance: 25000 | New Balance: 25250
Account ID: 203 | Old Balance: 5000 | New Balance: 5050

PL/SQL procedure successfully completed.
```

Scenario 2: Employee Bonus Update

```
CREATE TABLE Employees (
  EmpID       NUMBER PRIMARY KEY,
  Name        VARCHAR2(100),
  Department   VARCHAR2(50),
  Salary      NUMBER(10,2)
);

INSERT INTO Employees VALUES (1, 'Anjali Sharma', 'Finance', 50000);
INSERT INTO Employees VALUES (2, 'Ravi Mehta', 'Finance', 60000);
INSERT INTO Employees VALUES (3, 'Sunita Rao', 'HR', 45000);
INSERT INTO Employees VALUES (4, 'Arjun Das', 'IT', 70000);
COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
  p_dept IN VARCHAR2,
  p_bonus_pct IN NUMBER
) IS
  v_new_salary NUMBER;
BEGIN
  FOR rec IN (
    SELECT EmpID, Name, Salary FROM Employees WHERE Department = p_dept
  )
  LOOP
    v_new_salary := rec.Salary + (rec.Salary * (p_bonus_pct / 100));

    UPDATE Employees
    SET Salary = v_new_salary
    WHERE EmpID = rec.EmpID;

    DBMS_OUTPUT.PUT_LINE('Bonus applied to ' || rec.Name ||
                    ' (Emp ID: ' || rec.EmpID || ') in ' || p_dept ||
                    ' department. Old Salary: ' || rec.Salary ||
                    ', New Salary: ' || v_new_salary);
  END LOOP;

  COMMIT;
END;
/

SET SERVEROUTPUT ON;
EXEC UpdateEmployeeBonus('Finance', 10);
```

Output:

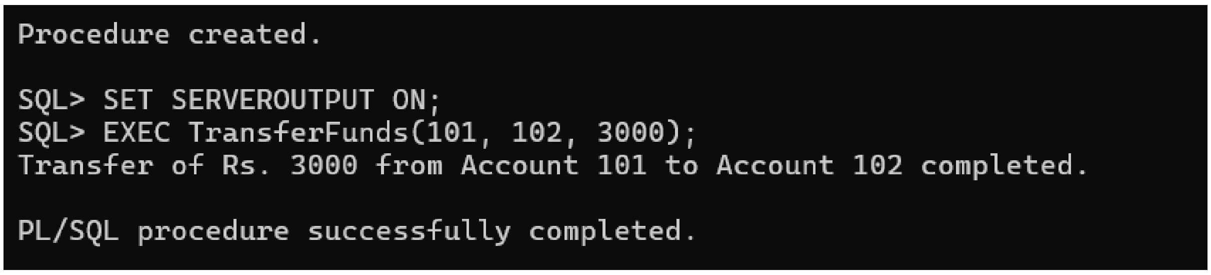


```
Procedure created.

SQL> SET SERVEROUTPUT ON;
SQL> EXEC UpdateEmployeeBonus('Finance', 10);
Bonus applied to Anjali Sharma (Emp ID: 1) in Finance department. Old Salary:
50000, New Salary: 55000
Bonus applied to Ravi Mehta (Emp ID: 2) in Finance department. Old Salary:
60000, New Salary: 66000

PL/SQL procedure successfully completed.
```

Scenario 3: Stored Procedure TransferFunds

```
CREATE TABLE Accounts (
  AccountID   NUMBER PRIMARY KEY,
  CustomerID  NUMBER,
  Balance     NUMBER(12,2)
);

INSERT INTO Accounts VALUES (101, 1, 15000);
INSERT INTO Accounts VALUES (102, 2, 5000);
INSERT INTO Accounts VALUES (103, 3, 3000);
COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds (
  p_from_acct IN NUMBER,
  p_to_acct   IN NUMBER,
  p_amount    IN NUMBER
) IS
  v_balance_from NUMBER;
BEGIN
  -- Lock the source row and check balance
  SELECT Balance INTO v_balance_from
  FROM Accounts
  WHERE AccountID = p_from_acct
  FOR UPDATE;

  IF v_balance_from < p_amount THEN
    DBMS_OUTPUT.PUT_LINE('Transfer failed: insufficient balance in Account ' ||
p_from_acct);
    RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance');
  ELSE
    -- Deduct from source account
    UPDATE Accounts
    SET Balance = Balance - p_amount
    WHERE AccountID = p_from_acct;

    -- Add to destination account
    UPDATE Accounts
    SET Balance = Balance + p_amount
    WHERE AccountID = p_to_acct;

    DBMS_OUTPUT.PUT_LINE('Transfer of Rs. ' || p_amount ||
                         ' from Account ' || p_from_acct ||
                         ' to Account ' || p_to_acct || ' completed.');
  END IF;

  COMMIT;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('One or both accounts not found.');
    ROLLBACK;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
    ROLLBACK;
END;
/
SET SERVEROUTPUT ON;
EXEC TransferFunds(101, 102, 3000);
```

Output:



```
Procedure created.

SQL> SET SERVEROUTPUT ON;
SQL> EXEC TransferFunds(101, 102, 3000);
Transfer of Rs. 3000 from Account 101 to Account 102 completed.

PL/SQL procedure successfully completed.
```

File name: 1. JUnit_Basic Testing Exercises
Name: Exercise 1: Setting Up Junit

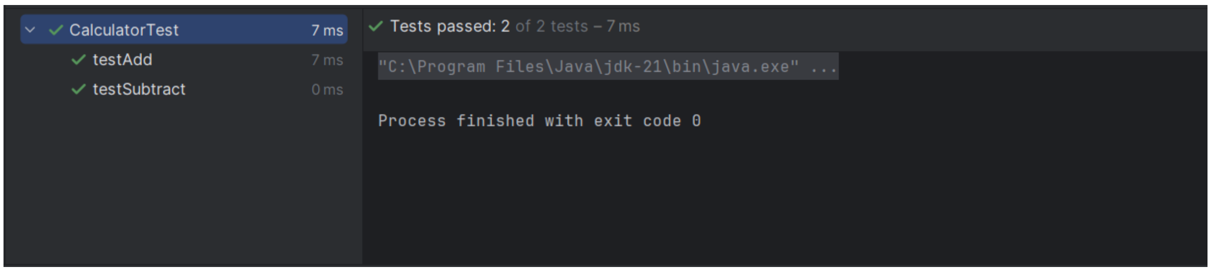
Calculator.java;

```
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;

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

public class CalculatorTest {

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

  @Test
  public void testSubtract() {
    Calculator c = new Calculator();
    assertEquals(5, c.subtract(10, 5));
  }
}
```

Output:



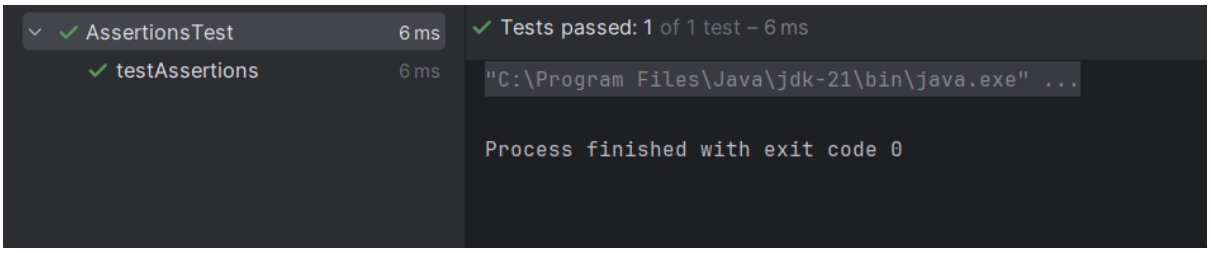
```
✓ CalculatorTest      7 ms    ✓ Tests passed: 2 of 2 tests – 7 ms
    ✓ testAdd         7 ms    "C:\Program Files\Java\jdk-21\bin\java.exe" ...
    ✓ testSubtract    0 ms
                              Process finished with exit code 0
```

File name: PLSQL_Exercises
Name: Exercise 3: Stored Procedures

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

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

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

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
      Object obj1 = null;
```

```
        assertNull(obj1);

            Object obj2 = new Object();
        assertNotNull(obj2);
    }
}
```

Output;



```
AssertionsTest          6 ms     Tests passed: 1 of 1 test – 6 ms
    testAssertions      6 ms     "C:\Program Files\Java\jdk-21\bin\java.exe" ...

                                 Process finished with exit code 0
```

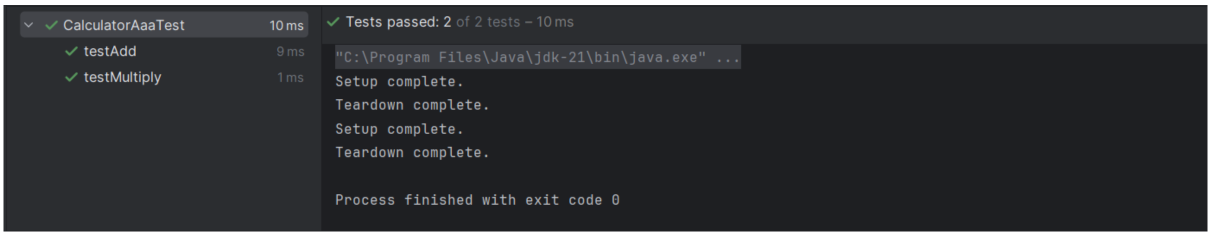
<u>File name:</u> PLSQL_Exercises
<u>Name:</u> Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

Calculator.java;

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

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

CalculatorAaaTest.java;

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

public class CalculatorAaaTest {

    private Calculator calculator;
    @Before
    public void setUp() {
        calculator = new Calculator();
        System.out.println("Setup complete.");
    }

    @After
    public void tearDown() {
        calculator = null;
        System.out.println("Teardown complete.");
    }

    @Test
    public void testAdd() {
        int a = 10;
        int b = 5;

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

        assertEquals(15, result);
    }

    @Test
    public void testMultiply() {
        int a = 4;
        int b = 3;

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



        assertEquals(12, result);
    }
}
```
Output;



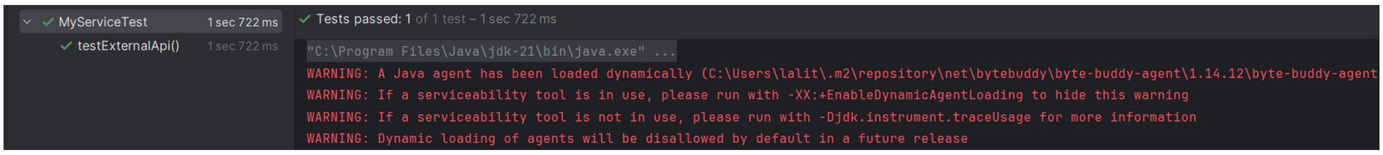
```
"C:\Program Files\Java\jdk-21\bin\java.exe" ...
Setup complete.
Teardown complete.
Setup complete.
Teardown complete.

Process finished with exit code 0
```

File name: Mockito_Exercises
Name: Exercise 1: Mocking and Stubbing

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

MyService.java;
public class MyService {
  private ExternalApi api;

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

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

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

public class MyServiceTest {

  @Test
  public void testExternalApi() {
    // Step 1: Create mock object
    ExternalApi mockApi = mock(ExternalApi.class);

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


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

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

Output;



```
MyServiceTest      1 sec 722 ms
  testExternalApi()  1 sec 722 ms
```

```
Tests passed: 1 of 1 test – 1 sec 722 ms
"C:\Program Files\Java\jdk-21\bin\java.exe" ...
WARNING: A Java agent has been loaded dynamically (C:\Users\lalit\.m2\repository\net\bytebuddy\byte-buddy-agent\1.14.12\byte-buddy-agent
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
```

<u>File name:</u> Mockito_Exercises

<u>Name:</u> Exercise 2: Verifying Interactions

ExternalApi.java;

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

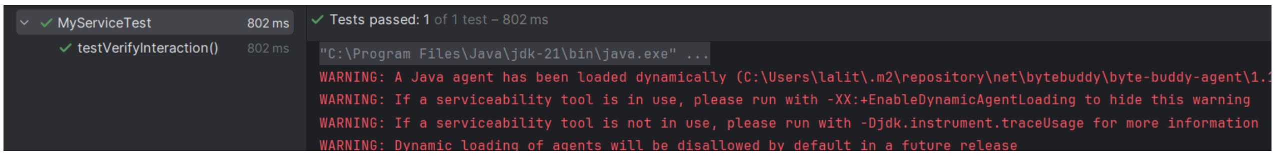
MyService.java;

```
public class MyService {
    private ExternalApi api;

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

    public String fetchData() {
        return api.getData();
    }
}
```

MyServiceTest.java;

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

public class MyServiceTest {
  @Test
  public void testVerifyInteraction() {
    // Step 1: Create mock object
    ExternalApi mockApi = mock(ExternalApi.class);

    // Step 2: Use service with mock
    MyService service = new MyService(mockApi);
    service.fetchData();

    // Step 3: Verify interaction
    verify(mockApi).getData();
  }
}
```

Output;



```
MyServiceTest  802 ms          Tests passed: 1 of 1 test – 802 ms
  testVerifyInteraction()  802 ms
"C:\Program Files\Java\jdk-21\bin\java.exe" ...
WARNING: A Java agent has been loaded dynamically (C:\Users\lalit\.m2\repository\net\bytebuddy\byte-buddy-agent\1.
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
```

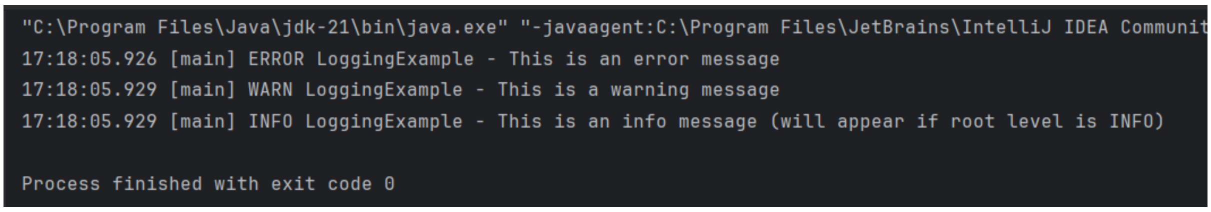
<u>File name:</u> 6. SL4J Logging exercises

<u>Name:</u> Exercise 1: Logging Error Messages and Warning Levels

LoggingExample.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");
    logger.info("This is an info message (will appear if root level is INFO)");
  }
}
```

Output;



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Communit
17:18:05.926 [main] ERROR LoggingExample - This is an error message
17:18:05.929 [main] WARN LoggingExample - This is a warning message
17:18:05.929 [main] INFO LoggingExample - This is an info message (will appear if root level is INFO)

Process finished with exit code 0
```