**PL/SQL EXERCISE:**

**Exercise 1: Control Structures**

-- Create Customers table

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Age NUMBER,

Balance NUMBER(10, 2),

IsVIP VARCHAR2(5) DEFAULT 'FALSE'

);

-- Create Loans table

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES Customers(CustomerID),

InterestRate NUMBER(5, 2),

DueDate DATE

);

-- Customers

INSERT INTO Customers VALUES (1, 'Anita Sharma', 65, 8500, 'FALSE');

INSERT INTO Customers VALUES (2, 'Rahul Mehta', 45, 12000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Priya Verma', 70, 3000, 'FALSE');

INSERT INTO Customers VALUES (4, 'Deepak Rao', 61, 15000, 'FALSE');

-- Loans

INSERT INTO Loans VALUES (101, 1, 6.5, SYSDATE + 15);

INSERT INTO Loans VALUES (102, 2, 7.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 8.0, SYSDATE + 10);

INSERT INTO Loans VALUES (104, 4, 6.8, SYSDATE + 5);

COMMIT;

BEGIN

FOR rec IN (SELECT CustomerID FROM Customers WHERE Age > 60) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1.0

WHERE CustomerID = rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Interest rate discounted for Customer ' || rec.CustomerID);

END LOOP;

END;

/

BEGIN

FOR rec IN (SELECT CustomerID FROM Customers WHERE Balance > 10000) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('VIP status granted to Customer ' || rec.CustomerID);

END LOOP;

END;

/

BEGIN

FOR rec IN (

SELECT l.LoanID, c.Name, l.DueDate

FROM Loans l

JOIN Customers c ON c.CustomerID = l.CustomerID

WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ' || rec.LoanID || ' for ' || rec.Name ||

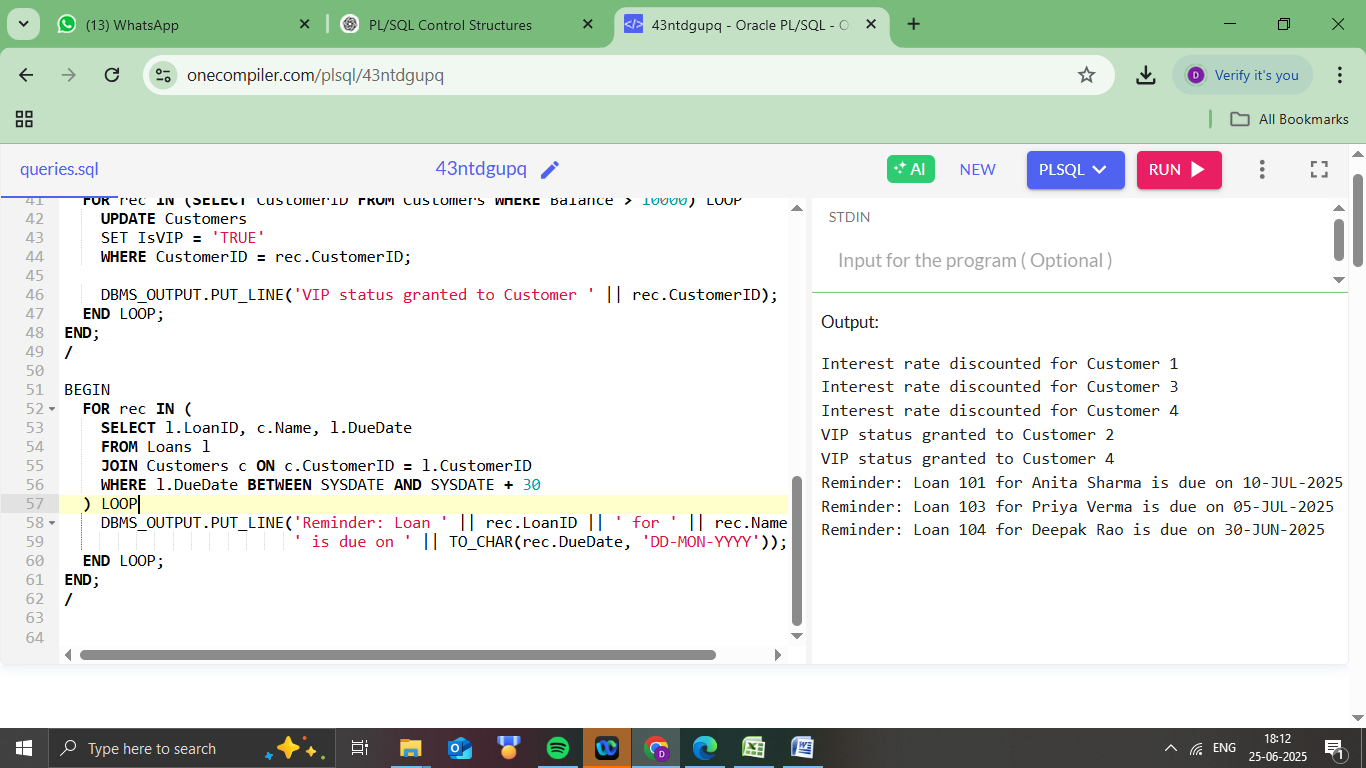
' is due on ' || TO\_CHAR(rec.DueDate, 'DD-MON-YYYY'));

END LOOP;

END;

/

**OUTPUT:**

****

**Exercise 3: Stored Procedures**

-- Enable DBMS output

SET SERVEROUTPUT ON;

-- Create SavingsAccounts table

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(100),

Balance NUMBER(12,2)

);

-- Create Employees table

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Department VARCHAR2(50),

Salary NUMBER(10,2)

);

-- Create Accounts table for fund transfers

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(100),

Balance NUMBER(12,2)

);

-- Insert into SavingsAccounts

INSERT INTO SavingsAccounts VALUES (1, 'Anita Sharma', 10000);

INSERT INTO SavingsAccounts VALUES (2, 'Rahul Mehta', 25000);

-- Insert into Employees

INSERT INTO Employees VALUES (1, 'Suresh Kumar', 'Finance', 60000);

INSERT INTO Employees VALUES (2, 'Meena Iyer', 'HR', 55000);

INSERT INTO Employees VALUES (3, 'Ravi Das', 'Finance', 62000);

-- Insert into Accounts

INSERT INTO Accounts VALUES (101, 'Anita Sharma', 8000);

INSERT INTO Accounts VALUES (102, 'Rahul Mehta', 12000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR rec IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP

UPDATE SavingsAccounts

SET Balance = Balance + (rec.Balance \* 0.01)

WHERE AccountID = rec.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest added for Account ID ' || rec.AccountID);

END LOOP;

END;

/

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

DeptName IN VARCHAR2,

BonusPct IN NUMBER

) IS

BEGIN

FOR rec IN (SELECT EmpID, Salary FROM Employees WHERE Department = DeptName) LOOP

UPDATE Employees

SET Salary = Salary + (rec.Salary \* BonusPct / 100)

WHERE EmpID = rec.EmpID;

DBMS\_OUTPUT.PUT\_LINE('Bonus updated for Employee ID ' || rec.EmpID);

END LOOP;

END;

/

CREATE OR REPLACE PROCEDURE TransferFunds (

FromAcct IN NUMBER,

ToAcct IN NUMBER,

Amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = FromAcct;

IF v\_balance < Amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance to transfer.');

ELSE

UPDATE Accounts

SET Balance = Balance - Amount

WHERE AccountID = FromAcct;

UPDATE Accounts

SET Balance = Balance + Amount

WHERE AccountID = ToAcct;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || Amount || ' from Account ' || FromAcct || ' to Account ' || ToAcct);

END IF;

END;

/

-- Call monthly interest procedure

BEGIN

ProcessMonthlyInterest;

END;

/

-- Call bonus update for Finance department with 10% bonus

BEGIN

UpdateEmployeeBonus('Finance', 10);

END;

/

-- Call fund transfer (transfer 3000 from 101 to 102)

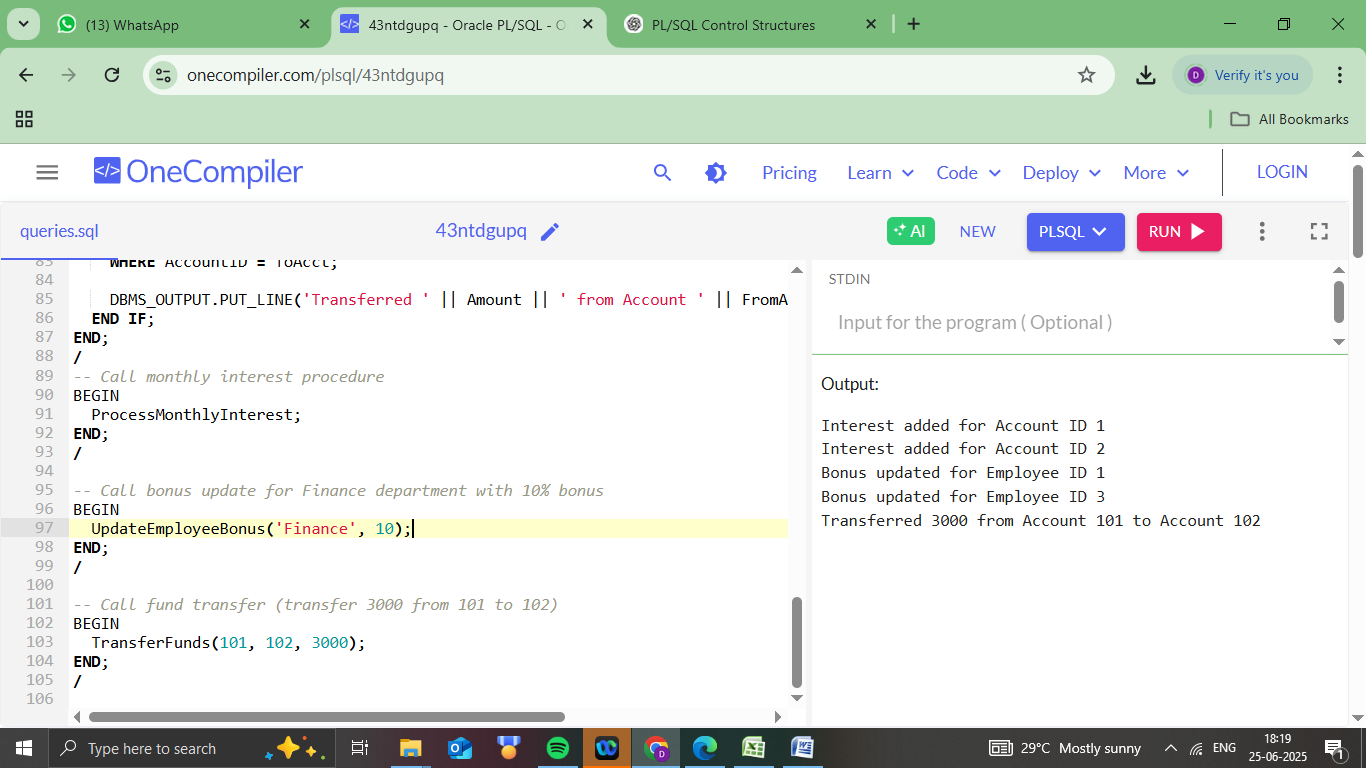
BEGIN

TransferFunds(101, 102, 3000);

END;

/

**OUTPUT:**

****

**JUnit\_Basic Testing Exercises:**

**Exercise 1: Setting Up JUnit**

**Calculator.java (in src/main/java)**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorTest.java (in src/test/java)**

import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.Test;

public class CalculatorTest {

@Test

void testAdd() {

Calculator calculator = new Calculator();

int result = calculator.add(2, 3);

assertEquals(5, result);

}

}

**pom.xml**

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

</dependencies>

**Exercise 3: Assertions in JUnit**

**AssertionsTest.java**

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

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);

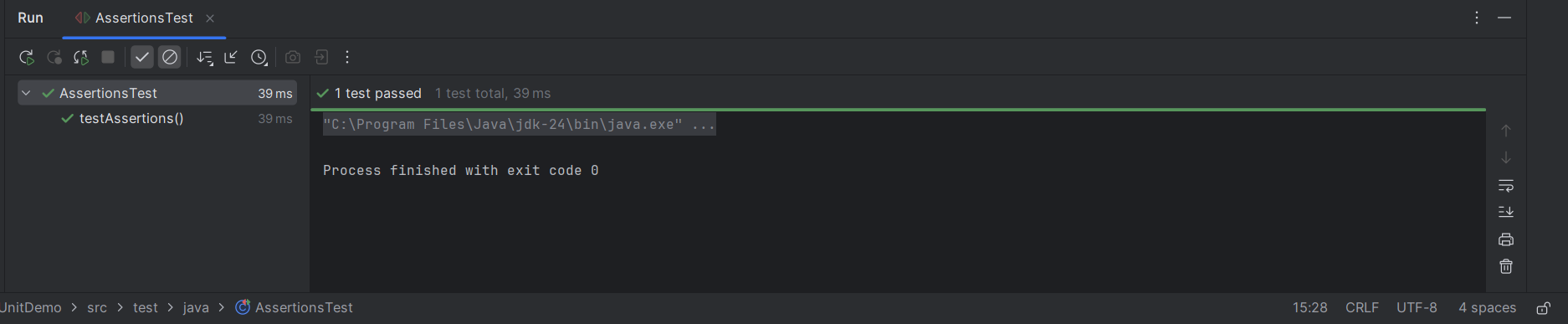
// Assert not null

assertNotNull(new Object());

}

}

Output:



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

**Scenario: Testing a Calculator with AAA Pattern and Setup/Teardown**

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 (JUnit 5 test with AAA + Setup/Teardown)**

import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.\*;

public class CalculatorTest {

private Calculator calculator;

@BeforeEach

void setUp() {

// Setup: runs before each test

calculator = new Calculator();

System.out.println("Setup complete");

}

@AfterEach

void tearDown() {

// Teardown: runs after each test

System.out.println("Teardown complete");

}

@Test

void testAddition() {

int a = 5;

int b = 3;

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

assertEquals(8, result);

} @Test

void testSubtraction() {

int a = 10;

int b = 4;

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

assertEquals(6, result);

}

}

**3. Mockito exercises**

**Exercise 1: Mocking and Stubbing**

Pom.xml:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>

  <artifactId>mockitomockingandstubbing</artifactId>

  <version>1.0-SNAPSHOT</version>

  <name>mockitomockingandstubbing</name>

  <!-- FIXME change it to the project's website -->

  <url>http://www.example.com</url>

  <properties>

    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <maven.compiler.source>1.8</maven.compiler.source>

    <maven.compiler.target>1.8</maven.compiler.target>

  </properties>

  <dependencies>

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>4.11</version>

      <scope>test</scope>

    </dependency>

     <dependency>

        <groupId>org.junit.jupiter</groupId>

        <artifactId>junit-jupiter</artifactId>

        <version>5.10.0</version>

        <scope>test</scope>

    </dependency>

    <dependency>

        <groupId>org.mockito</groupId>

        <artifactId>mockito-core</artifactId>

        <version>5.7.0</version>

        <scope>test</scope>

    </dependency>

  </dependencies>

**ExternalApi.java**

package com.example;

public interface ExternalApi {

    String getData();

}

**MyService.java**

package com.example;

public class MyService {

    private ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

**MyServiceTest.java**

package com.example;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

    @Test

    public void testExternalApi() {

        ExternalApi mockApi = mock(ExternalApi.class);

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

        MyService service = new MyService(mockApi);

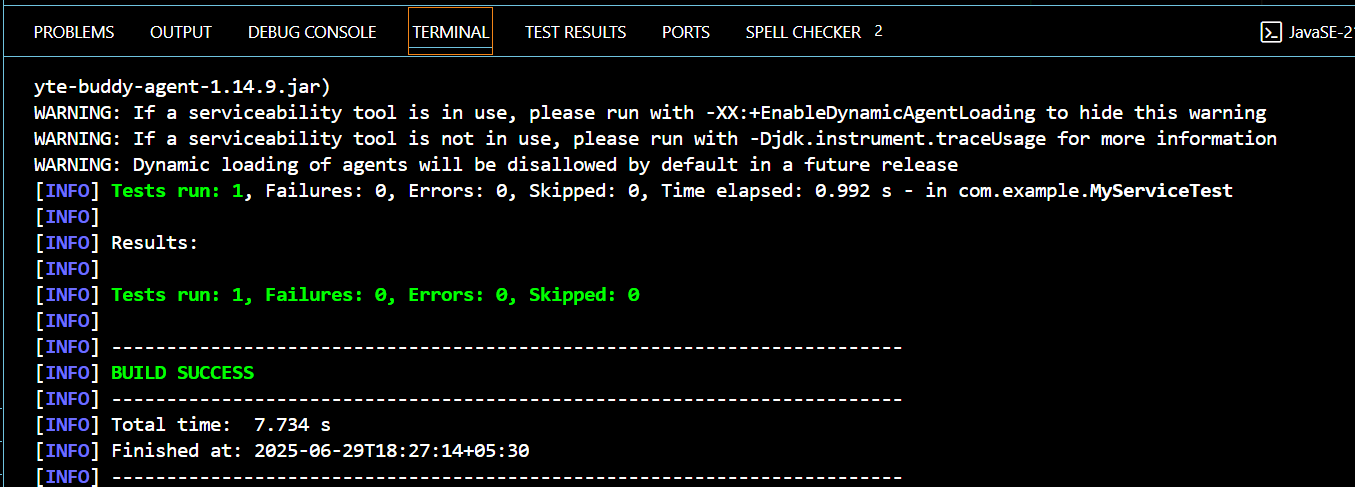
        String result = service.fetchData();

        assertEquals("Mock Data", result);

    }

}

**Output:**

****

**Exercise 2: Verifying Interactions**

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>

  <artifactId>verifyinginteractions</artifactId>

  <version>1.0-SNAPSHOT</version>

  <name>verifyinginteractions</name>

  <!-- FIXME change it to the project's website -->

  <url>http://www.example.com</url>

  <properties>

    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <maven.compiler.source>1.8</maven.compiler.source>

    <maven.compiler.target>1.8</maven.compiler.target>

  </properties>

  <dependencies>

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>4.11</version>

      <scope>test</scope>

    </dependency>

    <dependency>

        <groupId>org.junit.jupiter</groupId>

        <artifactId>junit-jupiter</artifactId>

        <version>5.10.0</version>

        <scope>test</scope>

    </dependency>

    <dependency>

        <groupId>org.mockito</groupId>

        <artifactId>mockito-core</artifactId>

        <version>5.7.0</version>

        <scope>test</scope>

    </dependency>

  </dependencies>

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

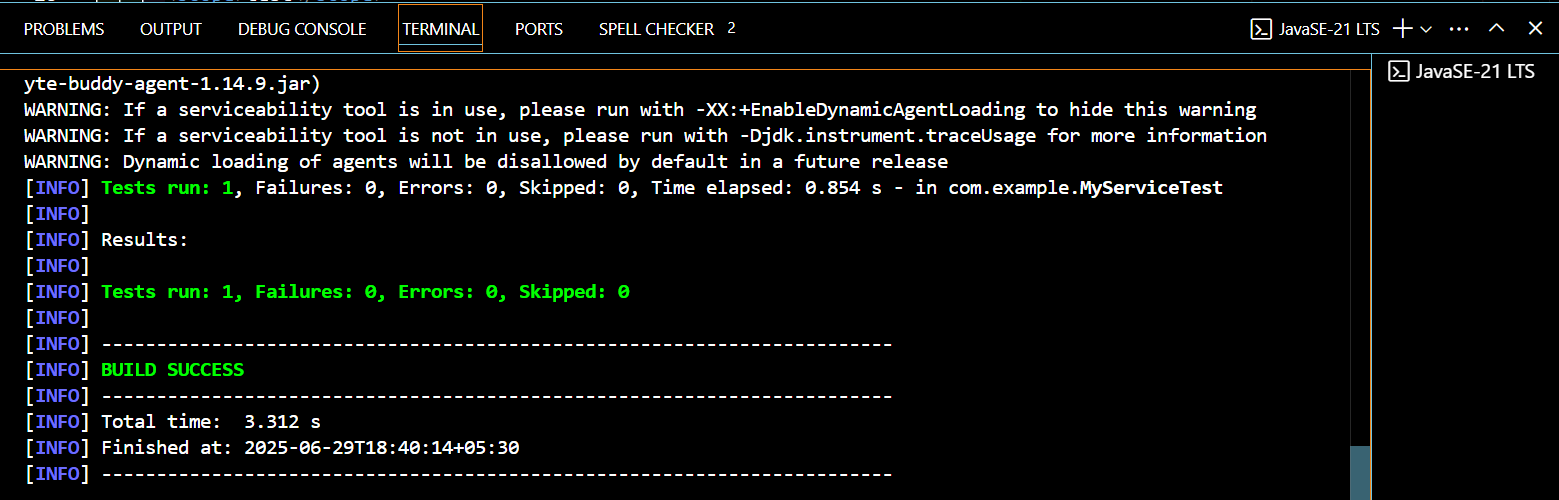
service.fetchData();

verify(mockApi).getData();

}

}

**Output:**



**6. SL4J Logging exercises**

**Exercise 1: Logging Error Messages and Warning Levels**

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>

  <artifactId>logging</artifactId>

  <version>1.0-SNAPSHOT</version>

  <name>logging</name>

  <!-- FIXME change it to the project's website -->

  <url>http://www.example.com</url>

  <properties>

    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <maven.compiler.source>1.8</maven.compiler.source>

    <maven.compiler.target>1.8</maven.compiler.target>

  </properties>

  <dependencies>

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>4.11</version>

      <scope>test</scope>

    </dependency>

    <dependency>

    <groupId>org.slf4j</groupId>

    <artifactId>slf4j-api</artifactId>

    <version>1.7.30</version>

</dependency>

<dependency>

    <groupId>ch.qos.logback</groupId>

    <artifactId>logback-classic</artifactId>

    <version>1.2.3</version>

</dependency>

  </dependencies>

**LoggingExample.java**

package com.example;

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");

}

}