WEEK-2 HANDS ON

PL/SQL

Exercise 1: Control Structures

- Create required tables “customers” and “loans” with required fields. And insert data into the tables. Commit the tables.
- Initially set the isvip columns to False.

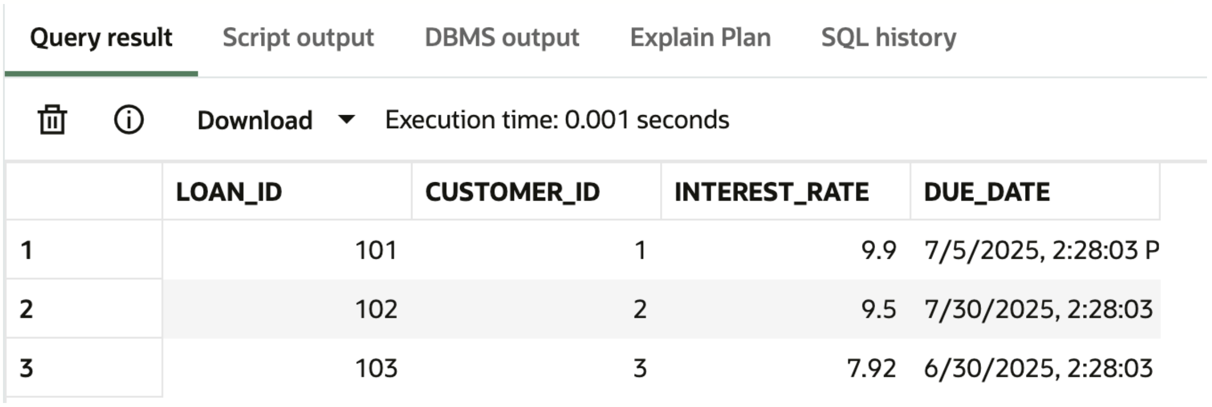
```
Create table customers (
   customer_id number primary key,
   customer_name varchar(100),
   age number,
   balance number,
   isvip varchar(5)
);

insert into customers values (1, 'ABC', 67, 15000, 'FALSE');
insert into  customers values (2, 'DEF', 45, 9000, 'FALSE');
insert into  customers values (3, 'ghi', 71, 12000, 'FALSE');

Create table loans (
   loan_id number primary key,
   customer_id number,
   interest_rate number,
   due_date date,
   foreign key (customer_id) references customers(customer_id)
);

insert into  loans values (101, 1, 10.0, SYSDATE+10 );
insert into  loans values (102, 2, 9.5, SYSDATE+35 );
insert into  loans values (103, 3, 8.0, SYSDATE+5 );

commit;
```

Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.

- **Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

```
begin

 for n in (select customer_id from customers where age > 60)

loop
```

```
update loans set interest_rate = interest_rate - (interest_rate*0.01)

where customer_id = n.customer_id;

end loop;

  commit;

end;

/

Select * from loans;
```



Query result | Script output | DBMS output | Explain Plan | SQL history

Download Execution time: 0.001 seconds

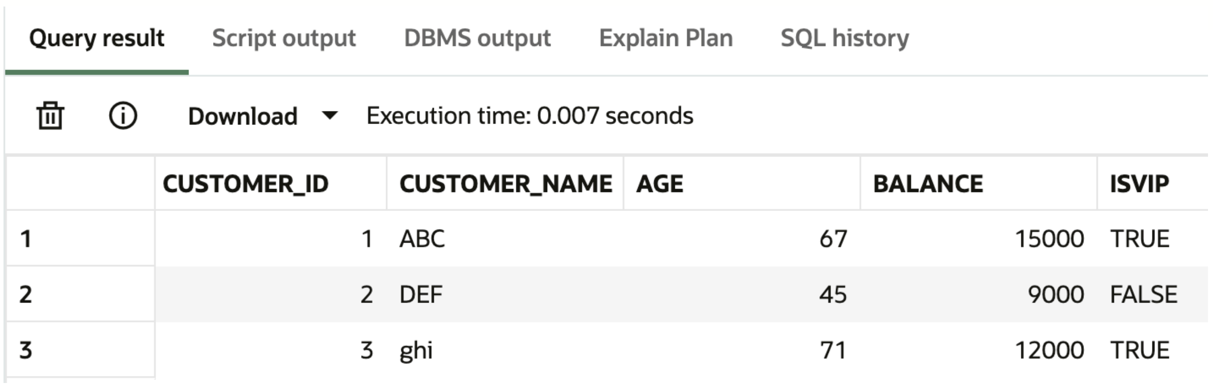
	LOAN_ID	CUSTOMER_ID	INTEREST_RATE	DUE_DATE
1	101	1	9.9	7/5/2025, 2:28:03 P
2	102	2	9.5	7/30/2025, 2:28:03
3	103	3	7.92	6/30/2025, 2:28:03

Scenario 2: A customer can be promoted to VIP status based on their balance.

- **Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

```
begin

 for r in ( select customer_id  from customers where balance > 10000)

loop

update customers set isvip = 'TRUE where customer_id = r.customer_id;

end loop;

commit;

end;

/
```

Output:



	CUSTOMER_ID	CUSTOMER_NAME	AGE	BALANCE	ISVIP
1	1	ABC	67	15000	TRUE
2	2	DEF	45	9000	FALSE
3	3	ghi	71	12000	TRUE

Scenario 3: The bank wants to send reminders to customers whose loans are due within the next 30 days.

- **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

begin

 for r in (select loan_id, due_date, customer_id from loans where due_date between sysdate and sysdate + 30)

loop

declare

customer_name customers.customer_name%type;

 begin

 select customer_name into customer_name from customers where customer_id = r.customer_id;

 dbms_outline.put_line('Reminder: Loan ID ' || r.loan_id || ' is due on ' || to_char(r.due_date, 'DD-MON-YYYY') || ' for customer ' || customer_name);

  end;

 end loop;

end;

/

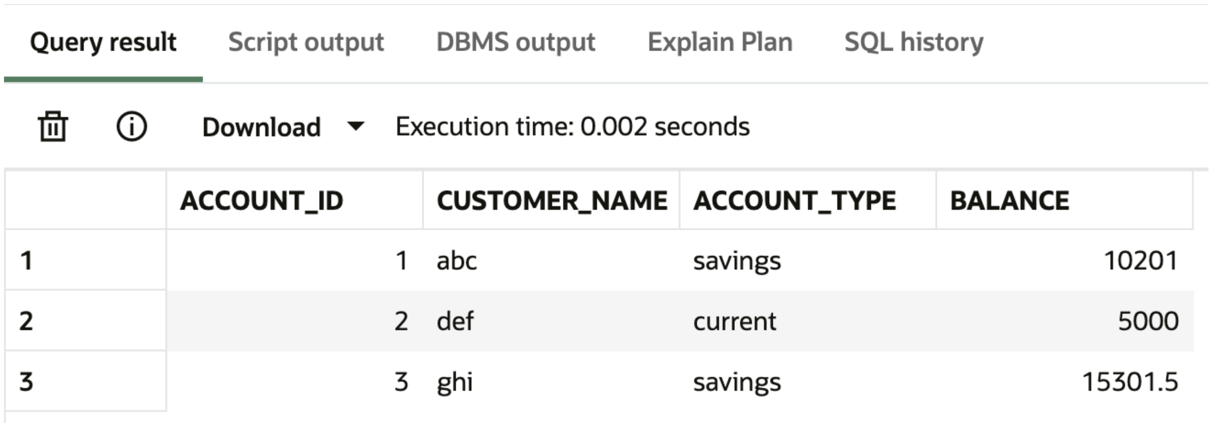
```
Reminder: Loan ID 101 is due on 05-JUL-2025 for customer ABC
Reminder: Loan ID 103 is due on 30-JUN-2025 for customer ghi


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.014
```

Exercise 3: Stored Procedures

Scenario 1: The bank needs to process monthly interest for all savings accounts.

- **Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

```
create table accounts ( account_id number primary key,  customer_name varchar2(100),  account_type varchar2(20),  balance number);

insert into accounts values (1, 'abc', 'savings', 10000);

insert into accounts values (2, 'def', 'current', 5000);

insert into accounts values (3, 'ghi', 'savings', 15000);

create or replace procedure ProcessMonthlyInterest is

begin

 update accounts set balance = balance + (balance * 0.01) where account_type = 'savings';

 commit;

end;

/

Exec ProcessMonthlyInterest;

Select * from accounts;
```

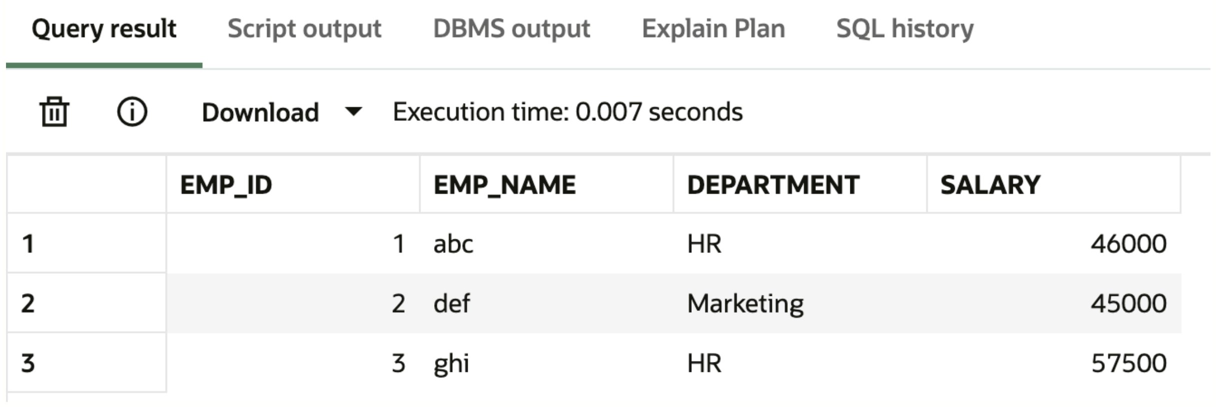


	ACCOUNT_ID	CUSTOMER_NAME	ACCOUNT_TYPE	BALANCE
1	1	abc	savings	10201
2	2	def	current	5000
3	3	ghi	savings	15301.5

Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.

- **Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

```
create table employees ( emp_id number primary key, emp_name varchar2(100)  department
varchar2(50), salary number);

insert into employees values (1, 'abc', 'HR', 40000);

insert into employees values (2, 'def', 'Marketing', 45000);

insert into employees values (3, 'ghi', 'HR', 50000);

create or replace procedure UpdateEmployeeBonus is

begin

 update employees set salary = salary + (salary * 15 / 100)where department = 'HR';

commit;

end;

/

exec UpdateEmployeeBonus;

select * from employees;
```



Execution time: 0.007 seconds

	EMP_ID	EMP_NAME	DEPARTMENT	SALARY
1	1	abc	HR	46000
2	2	def	Marketing	45000
3	3	ghi	HR	57500

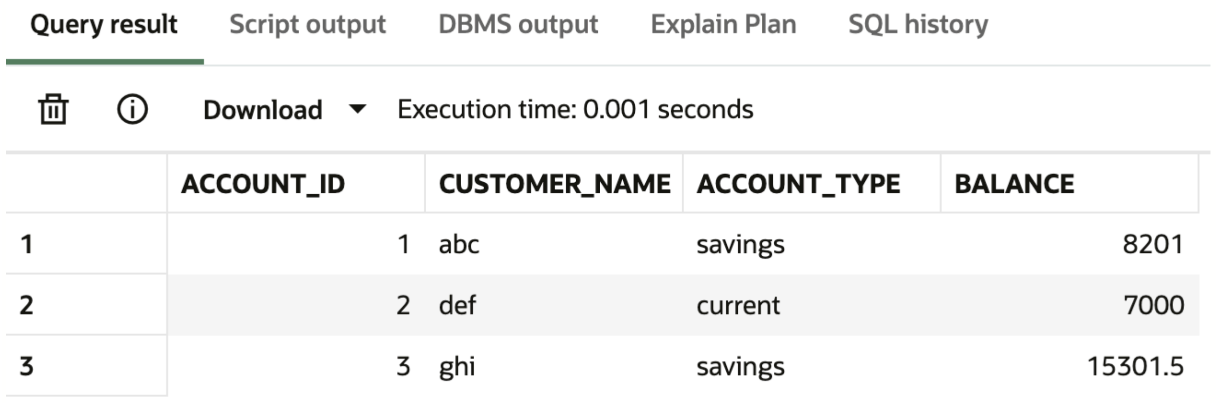
Scenario 3: Customers should be able to transfer funds between their accounts.

- **Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

We make use of the accounts table for this procedure.

```
create or replace procedure TransferFunds(from_acc in number, to_acc in number, amount in number ) is
from_balance number;
insufficient_balance exception;
```

```
begin
   select balance into from_balance from accounts where account_id=from_acc;
   if from_balance < amount then
   raise insufficient_balance;
   end if;
   update accounts set balance= balance - amount where account_id=from_acc;
   update accounts set balance= balance + amount where account_id=to_acc;
   commit;
exception
   when insufficient_balance then
   dbms_output.put_line('Balance is not sufficient.');
end;
/

exec TransferFunds(1,2,2000);

select * from accounts;
```



Execution time: 0.001 seconds

	ACCOUNT_ID	CUSTOMER_NAME	ACCOUNT_TYPE	BALANCE
1	1	abc	savings	8201
2	2	def	current	7000
3	3	ghi	savings	15301.5

JUnit Basic Testing Exercises

Exercise 1: Setting Up JUnit

- Created a java project named “Week2” in VS Code IDE using Maven Framework.
- Added junit dependency in the pom.xml file of the project

pom.xml

```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>demo</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

Exercise 3: Assertions in JUnit

- Created a java class named “calculator.java” inside com.example folder in VS Code that has can perform four basic arithmetic operations.
- Created a test class named “CalculatorTest.java” inside the test folder in VS Code to test Calculator functionalities.

Calculator.java

```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 int divide(int a, int b) throws Exception{
        if (b==0){
            throw new Exception("Cannot Divide by Zero");
        }
        return a/b;
    }
}
```

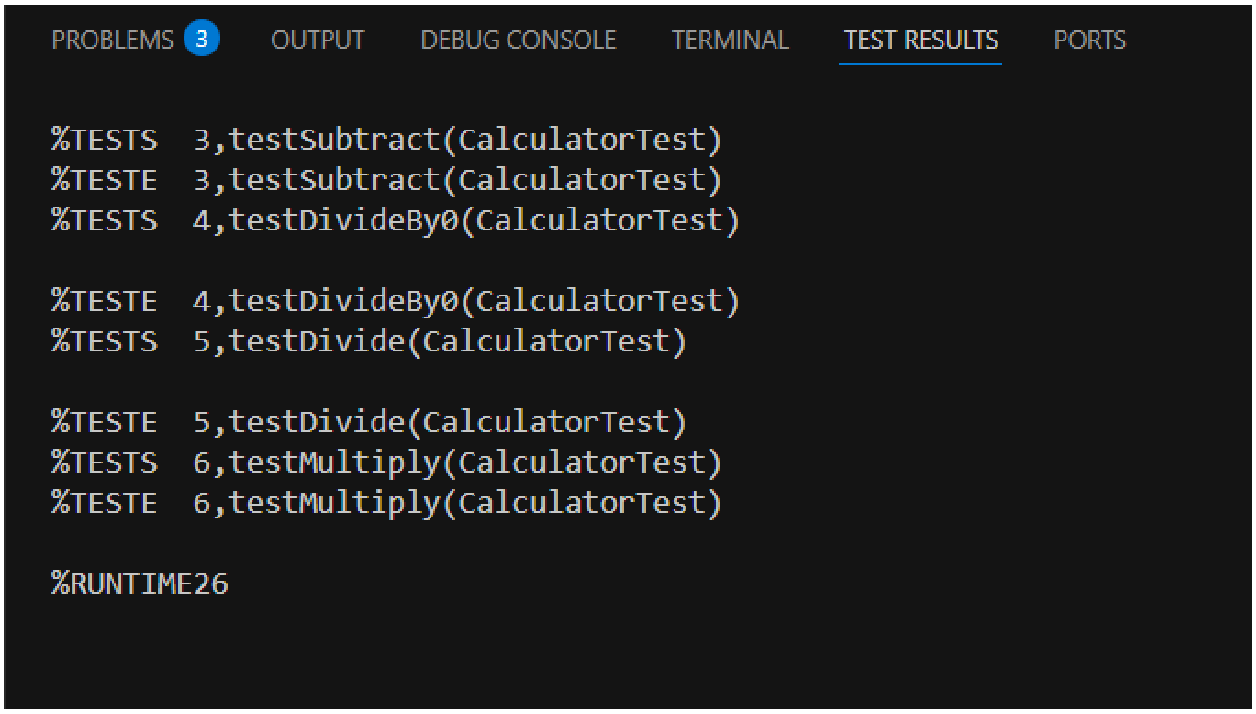
CalculatorTest.java

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

import org.junit.Assert;

import org.junit.Test;

import com.example.Calculator;



public class CalculatorTest {

    @Test

    public void testAdd(){

        Calculator calculator = new Calculator();

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

        Assert.assertEquals(result, 5);

    }



    @Test
```

```
public void testSubtract(){

    Calculator calculator = new Calculator();

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

    Assert.assertEquals(result, 1);

}


@Test

public void testMultiply(){

    Calculator calculator = new Calculator();

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

    Assert.assertEquals(result, 6);

}


@Test

public void testDivide() throws Exception{

    Calculator calculator = new Calculator();

    int result = calculator.divide(10, 5);

    Assert.assertEquals(result, 2);

}


@Test

public void testDivideBy0() throws Exception{
        Calculator calculator = new Calculator();

        Exception exception = assertThrows(Exception.class, ()->{

            calculator.divide(10, 0);

        });

        Assert.assertEquals(exception.getMessage(), "Cannot Divide by Zero");

    }

}
```

Output:

All the test cases are passed



PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS

```
%TESTS  3,testSubtract(CalculatorTest)
%TESTE  3,testSubtract(CalculatorTest)
%TESTS  4,testDivideBy0(CalculatorTest)

%TESTE  4,testDivideBy0(CalculatorTest)
%TESTS  5,testDivide(CalculatorTest)

%TESTE  5,testDivide(CalculatorTest)
%TESTS  6,testMultiply(CalculatorTest)
%TESTE  6,testMultiply(CalculatorTest)

%RUNTIME26
```

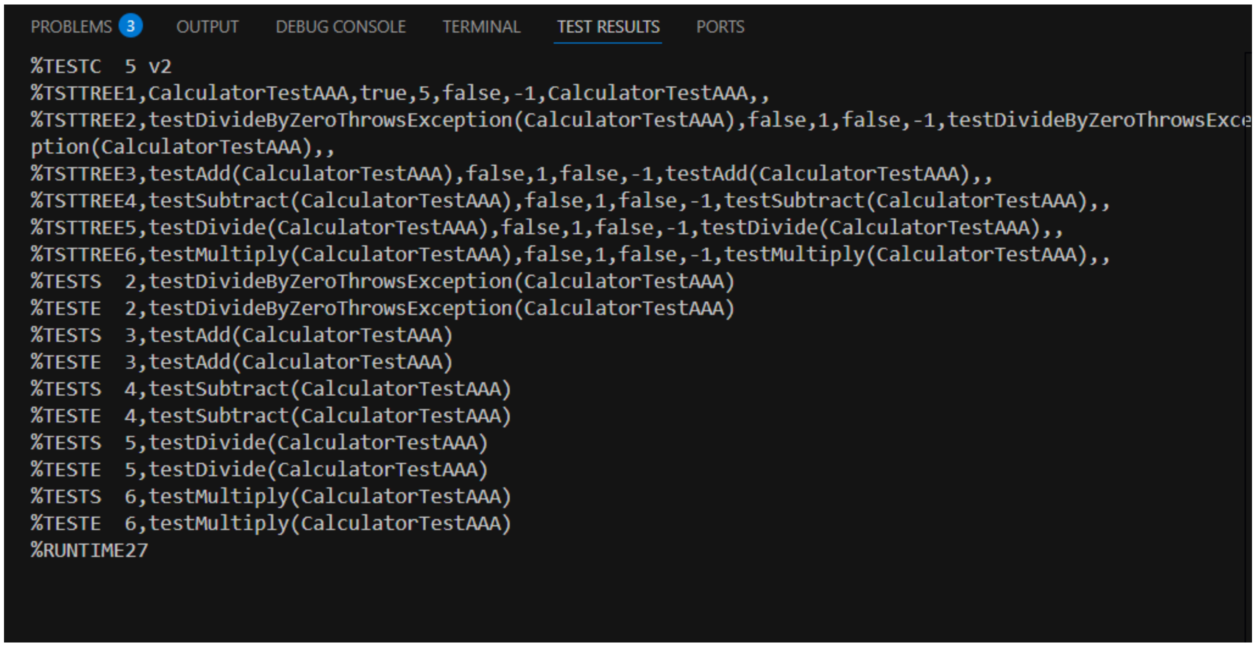
Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

- Created a test class “CalculatorTestAAA.java” to test in Arrange-Act-Assert (AAA) Pattern.

CalculatorTestAAA.java

```java
import com.example.Calculator;

import org.junit.Before;
import org.junit.After;
import org.junit.Test;

import static org.junit.Assert.*;

public class CalculatorTestAAA {

    private Calculator calculator;

    @Before
    public void setUp() {
        // Setup before each test
        calculator = new Calculator();
        System.out.println("Setup completed");
    }

    @After
    public void tearDown() {
        // Cleanup after each test
        calculator = null;
        System.out.println("Teardown completed");
    }

    @Test
    public void testAdd() {
        // Arrange
        int a = 5;
        int b = 3;

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

        // Assert
        assertEquals(8, result);
    }

    @Test
    public void testSubtract(){
        // Arrange
        int a = 5;
        int b = 3;

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

        // Assert
        assertEquals(2, result);
    }

    @Test
    public void testMultiply(){
        // Arrange
        int a = 5;
        int b = 3;

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

        // Assert
        assertEquals(15, result);
    }

    @Test
    public void testDivide() throws Exception{
        // Arrange
        int a = 10;
        int b = 5;

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

        // Assert
        assertEquals(2, result);
    }

    @Test
    public void testDivideByZeroThrowsException() {
        // Arrange
        int a = 10;
        int b = 0;

        // Act & Assert
        Exception exception = assertThrows(Exception.class, () -> {
            calculator.divide(a, b);
        });

        assertEquals("Cannot Divide by Zero", exception.getMessage());
    }
}
```

Output:


PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS

```
%TESTC  5 v2
%TSTTREE1,CalculatorTestAAA,true,5,false,-1,CalculatorTestAAA,,
%TSTTREE2,testDivideByZeroThrowsException(CalculatorTestAAA),false,1,false,-1,testDivideByZeroThrowsExce
ption(CalculatorTestAAA),,
%TSTTREE3,testAdd(CalculatorTestAAA),false,1,false,-1,testAdd(CalculatorTestAAA),,
%TSTTREE4,testSubtract(CalculatorTestAAA),false,1,false,-1,testSubtract(CalculatorTestAAA),,
%TSTTREE5,testDivide(CalculatorTestAAA),false,1,false,-1,testDivide(CalculatorTestAAA),,
%TSTTREE6,testMultiply(CalculatorTestAAA),false,1,false,-1,testMultiply(CalculatorTestAAA),,
%TESTS  2,testDivideByZeroThrowsException(CalculatorTestAAA)
%TESTE  2,testDivideByZeroThrowsException(CalculatorTestAAA)
%TESTS  3,testAdd(CalculatorTestAAA)
%TESTE  3,testAdd(CalculatorTestAAA)
%TESTS  4,testSubtract(CalculatorTestAAA)
%TESTE  4,testSubtract(CalculatorTestAAA)
%TESTS  5,testDivide(CalculatorTestAAA)
%TESTE  5,testDivide(CalculatorTestAAA)
%TESTS  6,testMultiply(CalculatorTestAAA)
%TESTE  6,testMultiply(CalculatorTestAAA)
%RUNTIME27
```

Mockito exercises

Exercise 1: Mocking and Stubbing

- Created a java class “UserService.java” and an interface “EmailService.java” the UserService uses the EmailService functionality.
- Created a java class “UserServiceTest.java” to test using Mockito.
- Followed the three step process of Mocking, Stubbing and Testing.

EmailService.java

```java
package com.example;
public interface EmailService {
    boolean sendEmail(String to, String subject, String body);
}
```

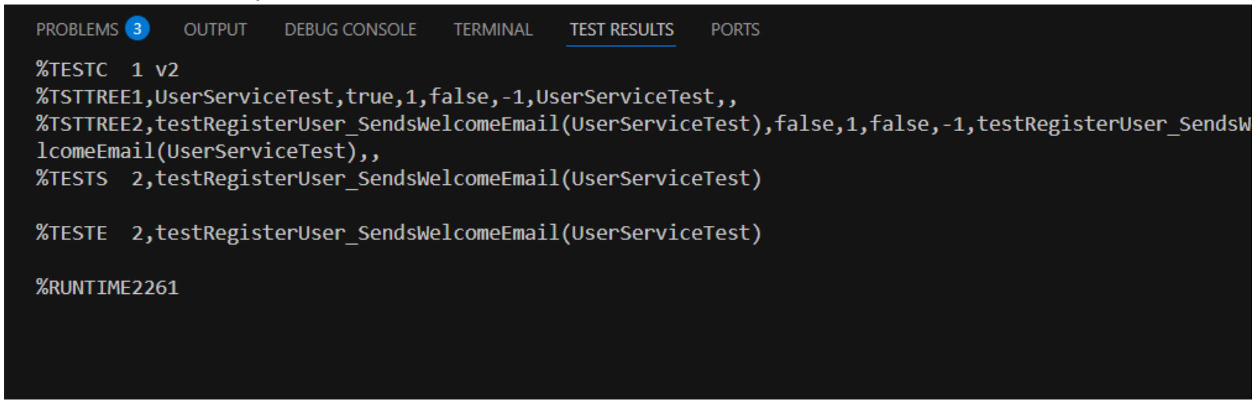
UserService.java

```java
package com.example;

public class UserService {
    private EmailService emailService;

    public UserService(EmailService emailService) {
        this.emailService = emailService;
    }

    public boolean registerUser(String email) {
        // do some user registration logic (omitted)
        String subject = "Welcome!";
        String body = "Thanks for registering.";
        return emailService.sendEmail(email, subject, body);
    }
}
```

UserServiceTest.java

```
// UserServiceTest.java

import org.junit.Test;
import static org.junit.Assert.assertTrue;
import static org.mockito.Mockito.*;

import com.example.EmailService;
import com.example.UserService;

public class UserServiceTest {

    @Test
    public void testRegisterUser_SendsWelcomeEmail() {
        // ✅ Mocking
        EmailService mockEmailService = mock(EmailService.class);

        // ✅ Stubbing
        when(mockEmailService.sendEmail(anyString(), anyString(),
anyString()))
            .thenReturn(true);

        UserService userService = new UserService(mockEmailService);

        // ✅ Act
        boolean result = userService.registerUser("user@example.com");

        // ✅ Assert
        assertTrue(result);
    }
}
```

Output:
All the test cases are passed with no errors.



```
%TESTC  1 v2
%TSTTREE1,UserServiceTest,true,1,false,-1,UserServiceTest,,
%TSTTREE2,testRegisterUser_SendsWelcomeEmail(UserServiceTest),false,1,false,-1,testRegisterUser_SendsW
lcomeEmail(UserServiceTest),,
%TESTS  2,testRegisterUser_SendsWelcomeEmail(UserServiceTest)

%TESTE  2,testRegisterUser_SendsWelcomeEmail(UserServiceTest)

%RUNTIME2261
```

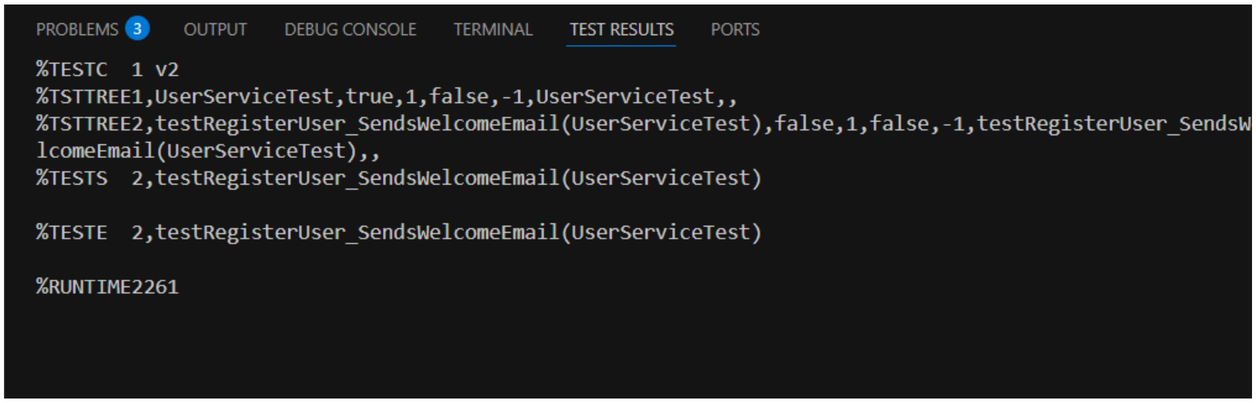
Exercise 2: Verifying Interactions

- Added the interaction verification logic to the “UserServiceTest.java” class

UserServiceTest.java

```java
// UserServiceTest.java

import org.junit.Test;
import static org.junit.Assert.assertTrue;
import static org.mockito.Mockito.*;

import com.example.EmailService;
import com.example.UserService;

public class UserServiceTest {

    @Test
    public void testRegisterUser_SendsWelcomeEmail() {
        // ✅ Mocking
        EmailService mockEmailService = mock(EmailService.class);

        // ✅ Stubbing
        when(mockEmailService.sendEmail(anyString(), anyString(),
anyString()))
            .thenReturn(true);

        UserService userService = new UserService(mockEmailService);

        // ✅ Act
        boolean result = userService.registerUser("user@example.com");

        // ✅ Assert
        assertTrue(result);

        // ✅ Verifying interactions
        verify(mockEmailService).sendEmail(
            eq("user@example.com"),
            eq("Welcome!"),
            eq("Thanks for registering.")
        );
    }
}
```

Output:
All the test cases and verification logic are passed as there are no exceptions in the testing.



```
%TESTC  1 v2
%TSTTREE1,UserServiceTest,true,1,false,-1,UserServiceTest,,
%TSTTREE2,testRegisterUser_SendsWelcomeEmail(UserServiceTest),false,1,false,-1,testRegisterUser_SendsW
lcomeEmail(UserServiceTest),,
%TESTS  2,testRegisterUser_SendsWelcomeEmail(UserServiceTest)

%TESTE  2,testRegisterUser_SendsWelcomeEmail(UserServiceTest)

%RUNTIME2261
```

Logging using SLF4J

Exercise 1: Logging Error Messages and Warning Levels

- Added “org.slf4j” and “ch.qos.logback” dependencies in the pom.xml file.
- Created a java class “LoggingExample.java” inside com.example folder in VS Code.

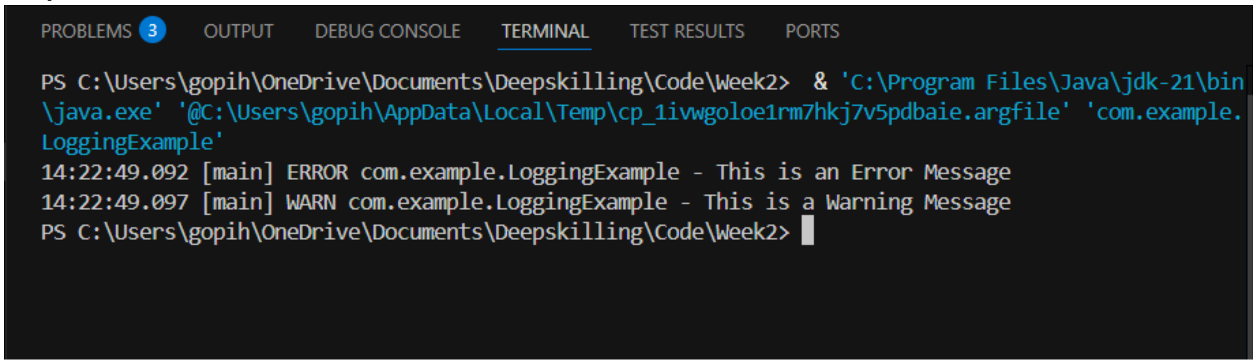
pom.xml

```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>demo</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>5.11.0</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>

</project>
```

LoggingExample.java

```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");;
    }
}
```

Output



```
PS C:\Users\gopih\OneDrive\Documents\Deepskilling\Code\Week2>  & 'C:\Program Files\Java\jdk-21\bin
\java.exe' '@C:\Users\gopih\AppData\Local\Temp\cp_1ivwgoloe1rm7hkj7v5pdbaie.argfile' 'com.example.
LoggingExample'
14:22:49.092 [main] ERROR com.example.LoggingExample - This is an Error Message
14:22:49.097 [main] WARN com.example.LoggingExample - This is a Warning Message
PS C:\Users\gopih\OneDrive\Documents\Deepskilling\Code\Week2>
```