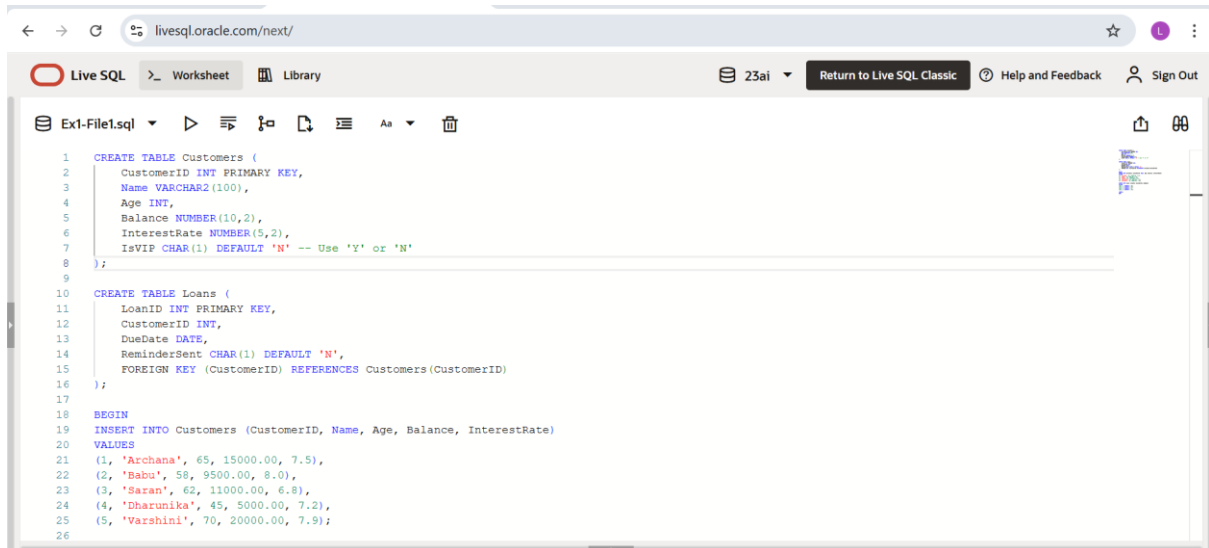


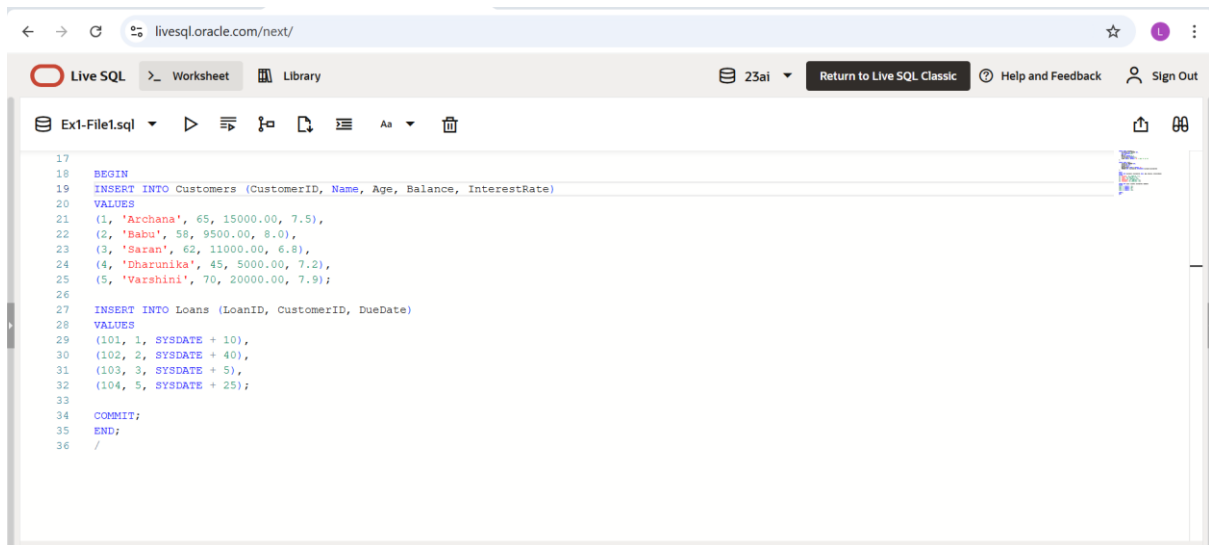
WEEK 2 - EXERCISES

PL/SQL PROGRAMMING

Exercise 1 : Control Structures



```
1 CREATE TABLE Customers (  
2   CustomerID INT PRIMARY KEY,  
3   Name VARCHAR2(100),  
4   Age INT,  
5   Balance NUMBER(10,2),  
6   InterestRate NUMBER(5,2),  
7   IsVIP CHAR(1) DEFAULT 'N' -- Use 'Y' or 'N'  
8 );  
9  
10 CREATE TABLE Loans (  
11   LoanID INT PRIMARY KEY,  
12   CustomerID INT,  
13   DueDate DATE,  
14   ReminderSent CHAR(1) DEFAULT 'N',  
15   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  
16 );  
17  
18 BEGIN  
19   INSERT INTO Customers (CustomerID, Name, Age, Balance, InterestRate)  
20   VALUES  
21   (1, 'Archana', 65, 15000.00, 7.5),  
22   (2, 'Babu', 58, 9500.00, 8.0),  
23   (3, 'Saran', 62, 11000.00, 6.8),  
24   (4, 'Dharunika', 45, 5000.00, 7.2),  
25   (5, 'Varshini', 70, 20000.00, 7.9);  
26
```



```
17  
18 BEGIN  
19   INSERT INTO Customers (CustomerID, Name, Age, Balance, InterestRate)  
20   VALUES  
21   (1, 'Archana', 65, 15000.00, 7.5),  
22   (2, 'Babu', 58, 9500.00, 8.0),  
23   (3, 'Saran', 62, 11000.00, 6.8),  
24   (4, 'Dharunika', 45, 5000.00, 7.2),  
25   (5, 'Varshini', 70, 20000.00, 7.9);  
26  
27   INSERT INTO Loans (LoanID, CustomerID, DueDate)  
28   VALUES  
29   (101, 1, SYSDATE + 10),  
30   (102, 2, SYSDATE + 40),  
31   (103, 3, SYSDATE + 5),  
32   (104, 5, SYSDATE + 25);  
33  
34   COMMIT;  
35 END;  
36 /
```

Scenario 1 :

livesql.oracle.com/next/

Live SQL Worksheet Library 23ai Return to Live SQL Classic Help and Feedback Sign Out

Navigator Files

All Files

Search files

DATABASE

- Ex1-File1.sql
- Ex1-Scenario1.sql

Ex1-Scenario1.sql

```

1 BEGIN
2   FOR rec IN (
3     SELECT CustomerID
4     FROM Customers
5     WHERE Age > 60
6   ) LOOP
7     UPDATE Customers
8     SET InterestRate = InterestRate - 1
9     WHERE CustomerID = rec.CustomerID;
10  END LOOP;
11
12  COMMIT;
13 END;
14 /
15

```

Query result Script output DBMS output Explain Plan SQL history

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.089

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.081 seconds

	CUSTOMERID	NAME	AGE	BALANCE	INTERESTRATE	ISVIP
1	1	Archana	65	15000	6.5	N
2	2	Babu	58	9500	8	N
3	3	Saran	62	11000	5.8	N
4	4	Dharunika	45	5000	7.2	N
5	5	Varshini	70	20000	6.9	N

Scenario 2 :

livesql.oracle.com/next/

Live SQL Worksheet Library 23ai Return to Live SQL Classic Help and Feedback Sign Out

Navigator Files

My Schema

Tables

Search objects

CUSTOMERS

LOANS

Ex1-Scenario2.sql

```

1 BEGIN
2   FOR rec IN (
3     SELECT CustomerID
4     FROM Customers
5     WHERE Balance > 10000
6   ) LOOP
7     UPDATE Customers
8     SET IsVIP = 'Y'
9     WHERE CustomerID = rec.CustomerID;
10  END LOOP;
11
12  COMMIT;
13 END;
14 /
15
16 SELECT * FROM CUSTOMERS;

```

About Oracle Contact Us Legal Notices Terms and Conditions Your Privacy Rights Delete Your Live SQL Account Cookie Preferences

Copyright © 2014, 2025 Oracle and/or its affiliates All rights reserved. r31.1

Exercise 3 : Stored Procedures

Live SQL

Worksheet

Library

23ai

Return to Live SQL Classic

Help and Feedback

Sign Out

Navigator

Files

All Files

Search files

DATABASE

PL-SQL_Ex3.sql

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

CREATE TABLE Customers2 (
CustomerID NUMBER PRIMARY KEY,
Name VARCHAR2(100),
AccountType VARCHAR2(20), -- e.g., 'Savings', 'Current'
Balance NUMBER(10, 2),
InterestRate NUMBER(5,2)
);

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

INSERT INTO Customers2 VALUES (1, 'Arjun', 'Savings', 10000.00, 1.0);
INSERT INTO Customers2 VALUES (2, 'Pranav', 'Current', 5000.00, 1.0);
INSERT INTO Customers2 VALUES (3, 'Arun', 'Savings', 20000.00, 1.0);
INSERT INTO Customers2 VALUES (4, 'Malini', 'Savings', 8000.00, 1.0);
INSERT INTO Customers2 VALUES (5, 'Priya', 'Current', 12000.00, 1.0);

INSERT INTO Employees VALUES (101, 'Jagan', 'IT', 60000);
INSERT INTO Employees VALUES (102, 'Saran', 'HR', 55000);
INSERT INTO Employees VALUES (103, 'Manoj', 'Finance', 70000);
INSERT INTO Employees VALUES (104, 'Naren', 'IT', 62000);
INSERT INTO Employees VALUES (105, 'Lingesh', 'Finance', 68000);

About Oracle

Contact Us

Legal Notices

Terms and Conditions

Your Privacy Rights

Delete Your Live SQL Account

Cookie Preferences

Copyright © 2014, 2025 Oracle and/or its affiliates All rights reserved. r31.1

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.008 seconds

	CUSTOMERID	NAME	ACCOUNTTYPE	BALANCE	INTERESTRATE
1	1	Arjun	Savings	10000	1
2	2	Pranav	Current	5000	1
3	3	Arun	Savings	20000	1
4	4	Malini	Savings	8000	1
5	5	Priya	Current	12000	1

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.001 seconds

	EMPLOYEEID	NAME	DEPARTMENT	SALARY
1	101	Jagan	IT	60000
2	102	Saran	HR	55000
3	103	Manoj	Finance	70000
4	104	Naren	IT	62000
5	105	Lingesh	Finance	68000

Scenario 1:

livesql.oracle.com/next/

Live SQL Worksheet Library 23ai Return to Live SQL Classic Help and Feedback Sign Out

Navigator Files

All Files

Search files

DATABASE

- Ex1-File1.sql
- Ex1-Scenario1.sql
- Ex1-Scenario2.sql
- Ex1-Scenario3.sql
- PL-SQL_Ex3-Scenario1.sql
- PL-SQL_Ex3.sql

```
1 CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
2 BEGIN
3   FOR rec IN (
4     SELECT CustomerID, Balance
5     FROM Customers2
6     WHERE AccountType = 'Savings'
7   ) LOOP
8     UPDATE Customers2
9     SET Balance = Balance + (Balance * 0.01)
10    WHERE CustomerID = rec.CustomerID;
11  END LOOP;
12
13  COMMIT;
14 END;
15 /
16 EXECUTE ProcessMonthlyInterest;
17
18 SELECT * FROM CUSTOMERS2;
```

About Oracle Contact Us Legal Notices Terms and Conditions Your Privacy Rights Delete Your Live SQL Account Cookie Preferences

Copyright © 2014, 2025 Oracle and/or its affiliates All rights reserved. r31.1

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.001 seconds

	CUSTOMERID	NAME	ACCOUNTTYPE	BALANCE	INTERESTRATE
1	1	Arjun	Savings	10100	1
2	2	Pranav	Current	5000	1
3	3	Arun	Savings	20200	1
4	4	Malini	Savings	8080	1
5	5	Priya	Current	12000	1

Scenario 2:

livesql.oracle.com/next/

Live SQL Worksheet Library 23ai Return to Live SQL Classic Help and Feedback Sign Out

Navigator Files

All Files

Search files

DATABASE

- Ex1-File1.sql
- Ex1-Scenario1.sql
- Ex1-Scenario2.sql
- Ex1-Scenario3.sql
- PL-SQL_Ex3-Scenario1.sql
- PL-SQL_Ex3-Scenario2.sql
- PL-SQL_Ex3.sql

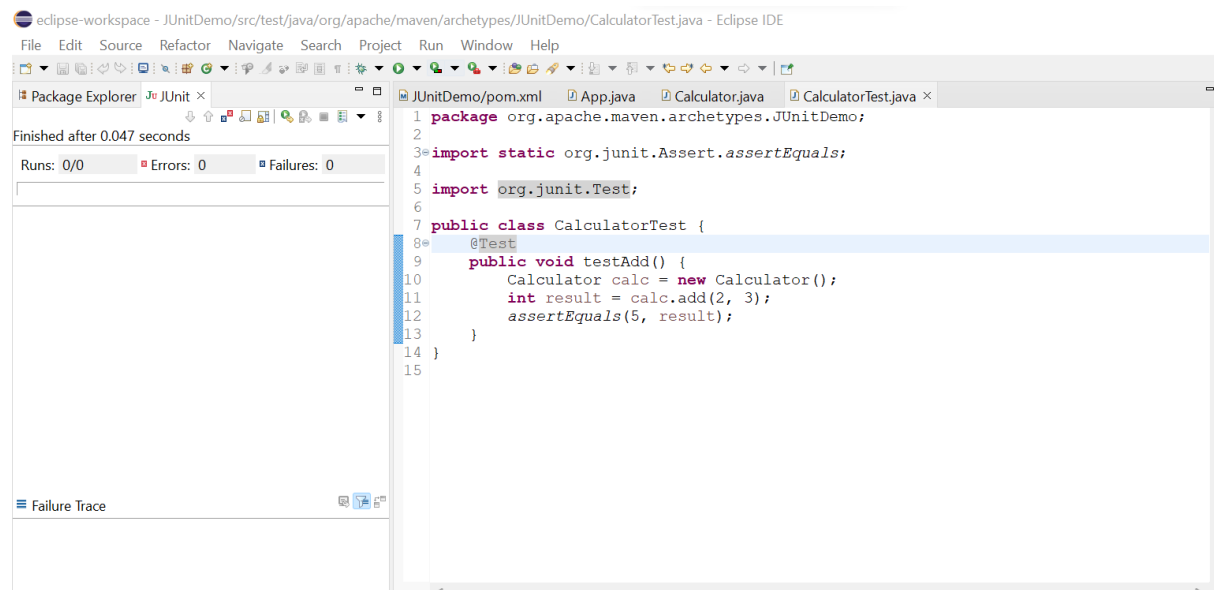
```
1 CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(
2   p_Department IN VARCHAR2,
3   p_BonusPercent IN NUMBER
4 ) AS
5 BEGIN
6   UPDATE Employees
7   SET Salary = Salary + (Salary * p_BonusPercent / 100)
8   WHERE Department = p_Department;
9
10  COMMIT;
11 END;
12 /
13 EXECUTE UpdateEmployeeBonus('IT', 5);
14
15 SELECT * FROM EMPLOYEES;
```

About Oracle Contact Us Legal Notices Terms and Conditions Your Privacy Rights Delete Your Live SQL Account Cookie Preferences

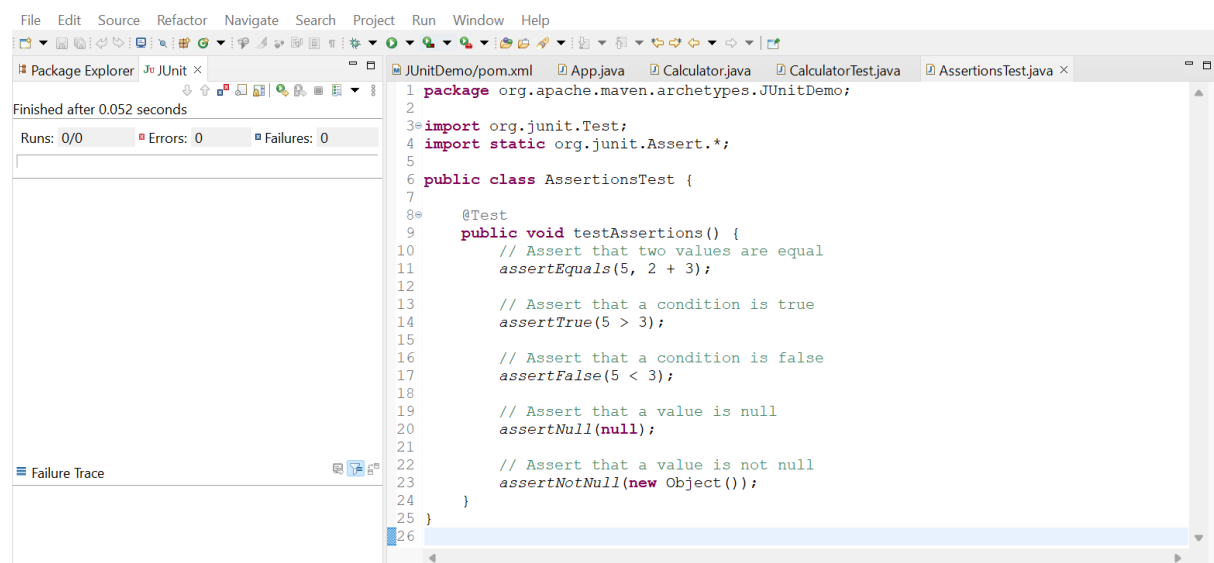
Copyright © 2014, 2025 Oracle and/or its affiliates All rights reserved. r31.1

JUnit basic testing exercises

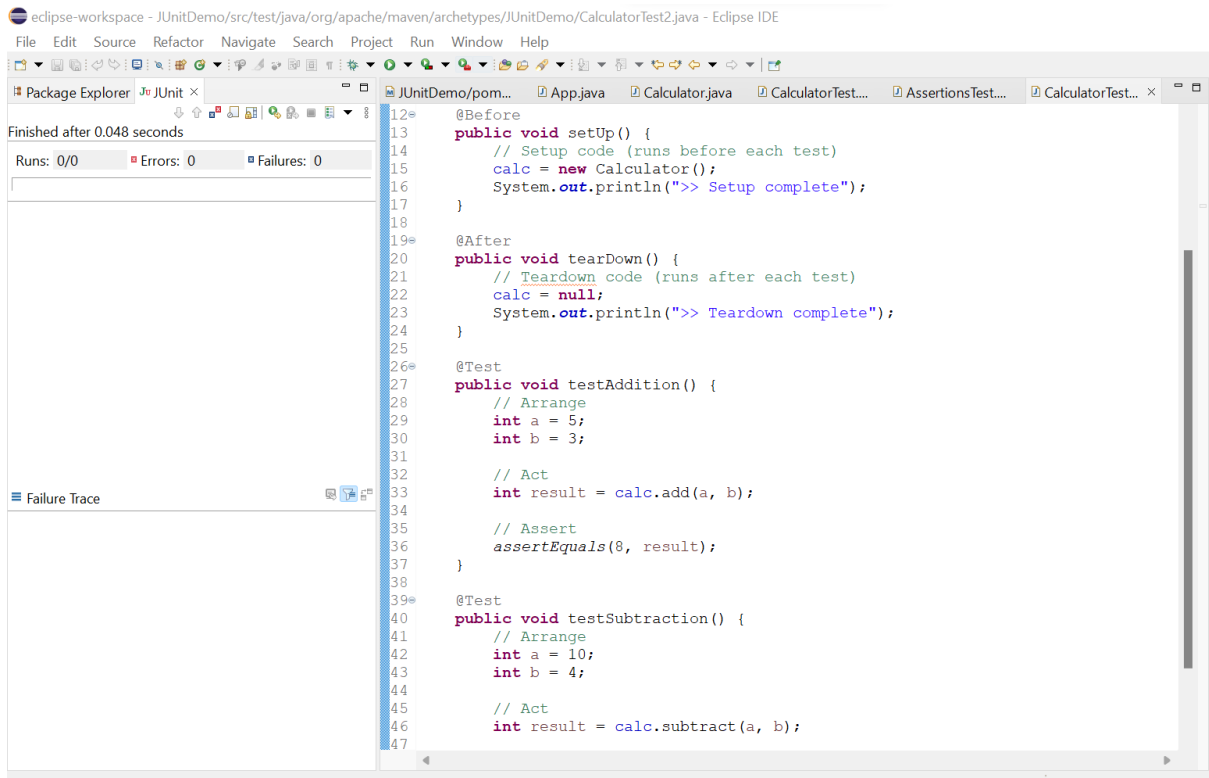
Exercise 1 : Setting up JUnit



Exercise 3 : Assertions in JUnit



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

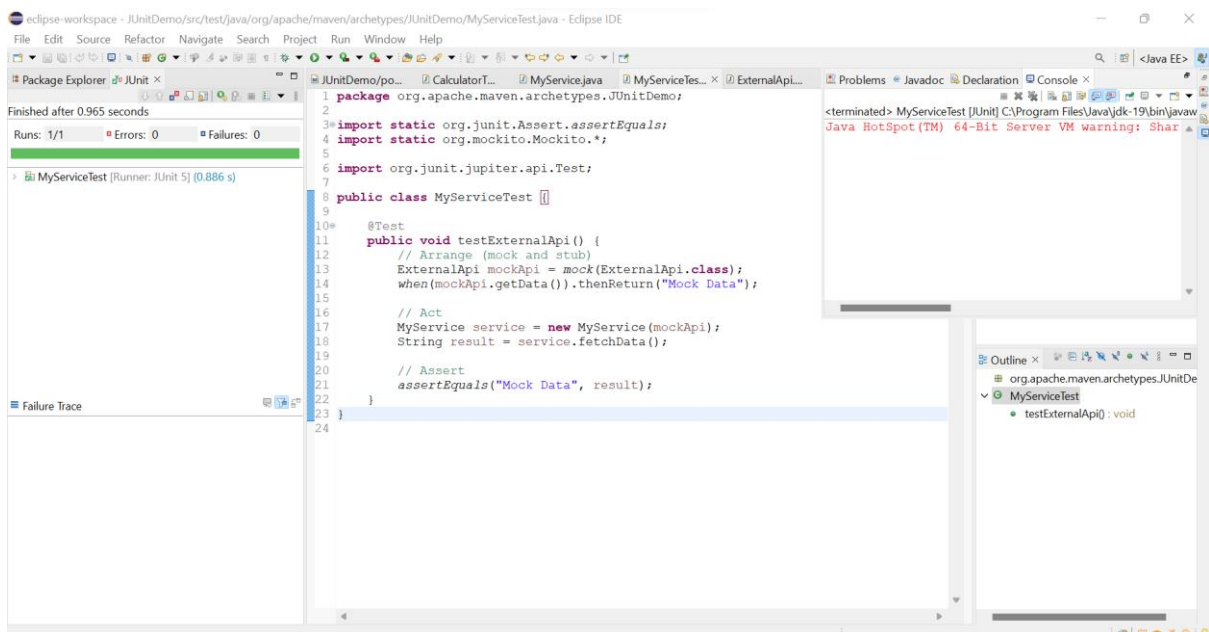


The screenshot shows the Eclipse IDE with the file `JUnitDemo/src/test/java/org/apache/maven/archetypes/JUnitDemo/CalculatorTest2.java` open. The code implements the AAA pattern with `@Before`, `@After`, and `@Test` annotations. The `setUp()` method initializes a `Calculator` instance, and the `tearDown()` method sets it to `null`. Two test methods, `testAddition()` and `testSubtraction()`, are shown, each with arrange, act, and assert steps. The left sidebar shows the Package Explorer and a console output indicating the test finished after 0.048 seconds with 0 runs, 0 errors, and 0 failures.

```
12=
13  @Before
14  public void setUp() {
15      // Setup code (runs before each test)
16      calc = new Calculator();
17      System.out.println(">> Setup complete");
18  }
19=
20  @After
21  public void tearDown() {
22      // Teardown code (runs after each test)
23      calc = null;
24      System.out.println(">> Teardown complete");
25  }
26=
27  @Test
28  public void testAddition() {
29      // Arrange
30      int a = 5;
31      int b = 3;
32
33      // Act
34      int result = calc.add(a, b);
35
36      // Assert
37      assertEquals(8, result);
38  }
39=
40  @Test
41  public void testSubtraction() {
42      // Arrange
43      int a = 10;
44      int b = 4;
45
46      // Act
47      int result = calc.subtract(a, b);
48  }
```

Mockito Exercises

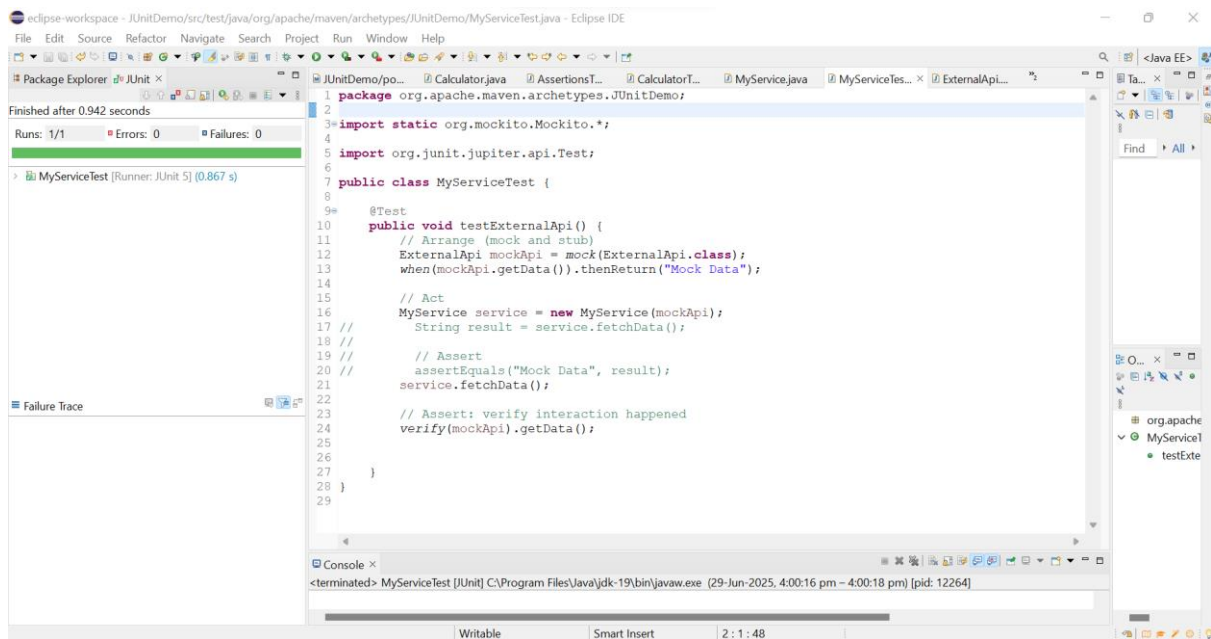
Exercise 1 : Mocking and Stubbing



The screenshot shows the Eclipse IDE with the file `JUnitDemo/src/test/java/org/apache/maven/archetypes/JUnitDemo/MyServiceTest.java` open. The code uses Mockito for mocking and stubbing. It imports `org.junit.Assert.assertEquals`, `org.mockito.Mockito`, and `org.junit.jupiter.api.Test`. The `testExternalApi()` method uses `mock()` to create a `MockApi` and `when()` to stub the `getData()` method to return "Mock Data". The `act` step calls `service.fetchData()`, and the `assert` step verifies the result. The left sidebar shows the Package Explorer and a console output indicating the test finished after 0.965 seconds with 1/1 runs, 0 errors, and 0 failures. The right sidebar shows the Outline view with the `testExternalApi()` method listed.

```
1 package org.apache.maven.archetypes.JUnitDemo;
2
3 import static org.junit.Assert.assertEquals;
4 import static org.mockito.Mockito.*;
5
6 import org.junit.jupiter.api.Test;
7
8 public class MyServiceTest {}
9
10 @Test
11 public void testExternalApi() {
12     // Arrange (mock and stub)
13     ExternalApi mockApi = mock(ExternalApi.class);
14     when(mockApi.getData()).thenReturn("Mock Data");
15
16     // Act
17     MyService service = new MyService(mockApi);
18     String result = service.fetchData();
19
20     // Assert
21     assertEquals("Mock Data", result);
22 }
23
24 }
```


Exercise 2 : Verifying Interactions



SL4J Logging Framework

Exercise 1 : Logging Error Messages and Warning Levels

