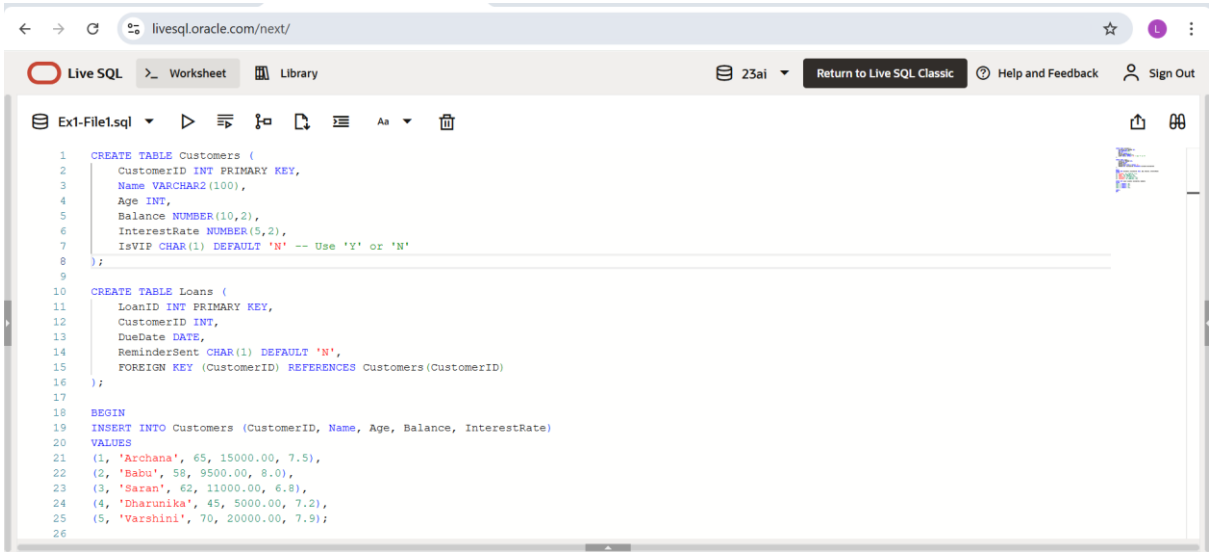
WEEK 2 - EXERCISES

PL/SQL PROGRAMMING

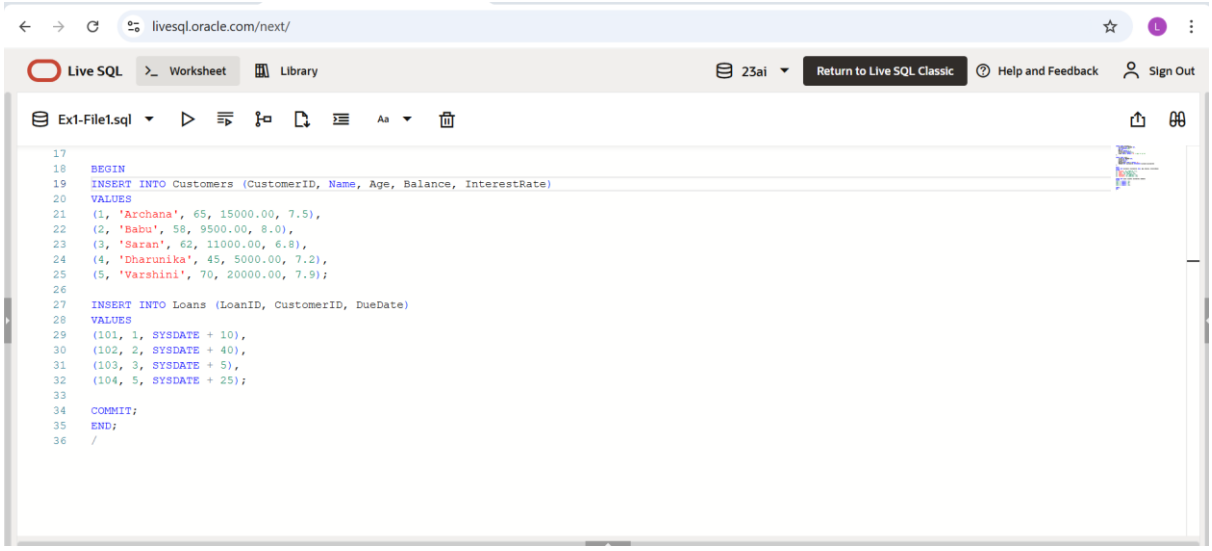
Exercise 1 : Control Structures




```sql
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR2(100),
    Age INT,
    Balance NUMBER(10,2),
    InterestRate NUMBER(5,2),
    IsVIP CHAR(1) DEFAULT 'N' -- Use 'Y' or 'N'
);

CREATE TABLE Loans (
    LoanID INT PRIMARY KEY,
    CustomerID INT,
    DueDate DATE,
    ReminderSent CHAR(1) DEFAULT 'N',
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

BEGIN
INSERT INTO Customers (CustomerID, Name, Age, Balance, InterestRate)
VALUES
(1, 'Archana', 65, 15000.00, 7.5),
(2, 'Babu', 58, 9500.00, 8.0),
(3, 'Saran', 62, 11000.00, 6.8),
(4, 'Dharunika', 45, 5000.00, 7.2),
(5, 'Varshini', 70, 20000.00, 7.9);
```



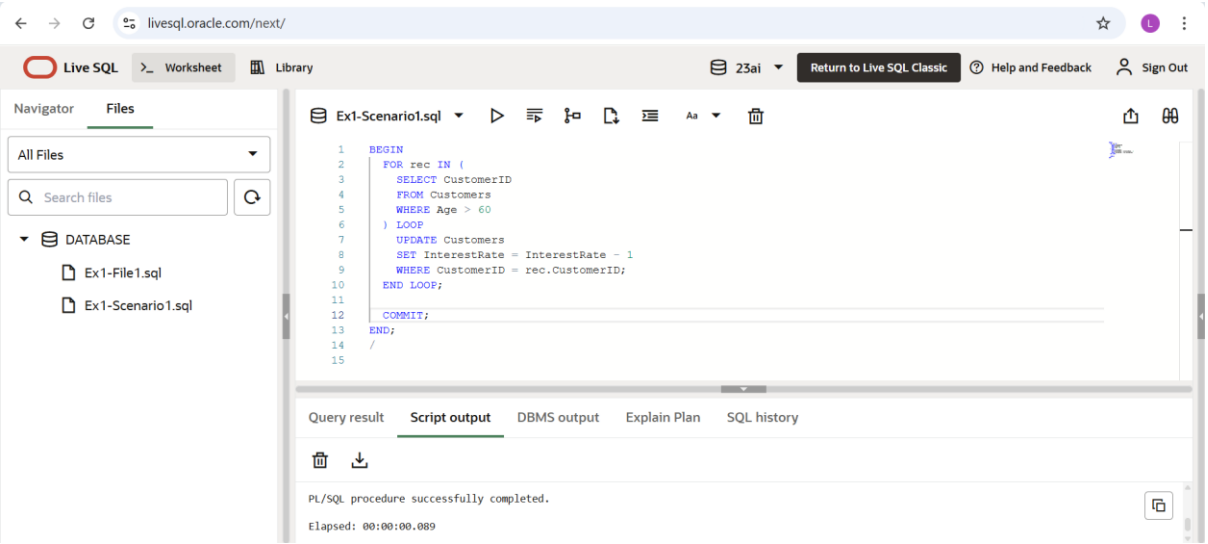

```sql

BEGIN
INSERT INTO Customers (CustomerID, Name, Age, Balance, InterestRate)
VALUES
(1, 'Archana', 65, 15000.00, 7.5),
(2, 'Babu', 58, 9500.00, 8.0),
(3, 'Saran', 62, 11000.00, 6.8),
(4, 'Dharunika', 45, 5000.00, 7.2),
(5, 'Varshini', 70, 20000.00, 7.9);

INSERT INTO Loans (LoanID, CustomerID, DueDate)
VALUES
(101, 1, SYSDATE + 10),
(102, 2, SYSDATE + 40),
(103, 3, SYSDATE + 5),
(104, 5, SYSDATE + 25);

COMMIT;
END;
/
```

Scenario 1 :

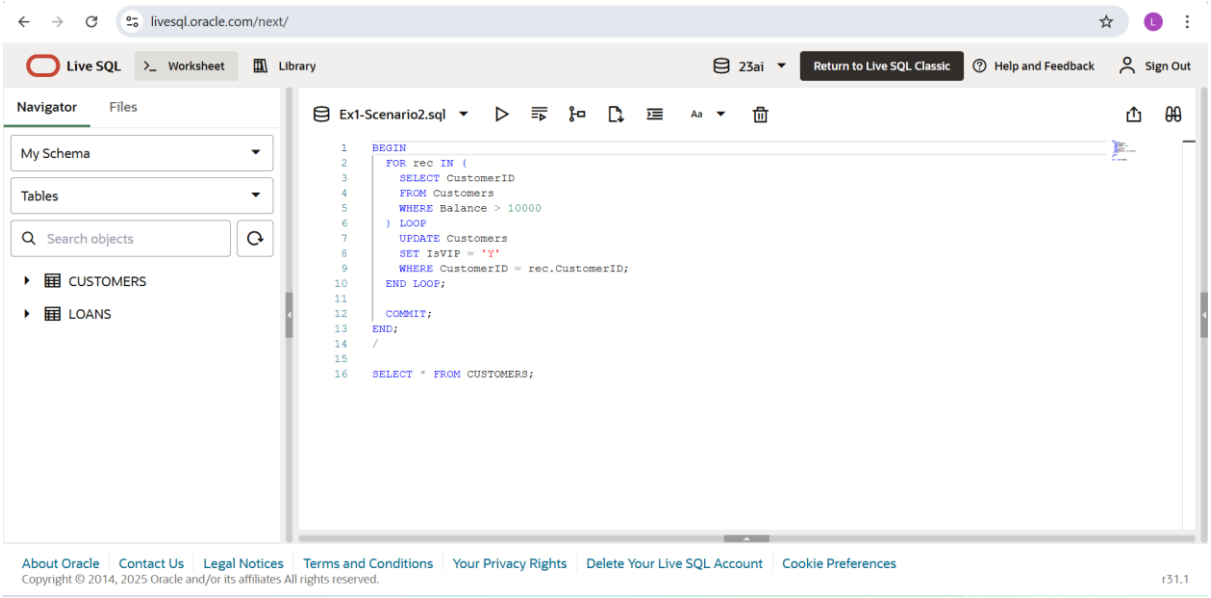


Query result | Script output | DBMS output | Explain Plan | SQL history

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 :



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

Download Execution time: 0.004 seconds

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

Scenario 3:

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-Scenario3.sql

```
BEGIN
  FOR rec IN (
    SELECT LoanID, CustomerID, DueDate
    FROM Loans
    WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30
      AND ReminderSent = 'N'
  ) LOOP
    -- Print reminder message
    DBMS_OUTPUT.PUT_LINE(
      'Reminder: Customer ' || rec.CustomerID ||
      ' has a loan due on ' || TO_CHAR(rec.DueDate, 'DD-MON-YYYY')
    );

    -- Mark reminder as sent
    UPDATE Loans
    SET ReminderSent = 'Y'
    WHERE LoanID = rec.LoanID;
  END LOOP;

  COMMIT;
END;
/
SELECT * from LOANS;
```

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

```
Reminder: Customer 1 has a loan due on 08-JUL-2025
Reminder: Customer 3 has a loan due on 03-JUL-2025
Reminder: Customer 5 has a loan due on 23-JUL-2025
```

Exercise 3 : Stored Procedures

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

PL-SQL_Ex3.sql

```
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

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

PL-SQL_Ex3-Scenario1.sql

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
  FOR rec IN (
    SELECT CustomerID, Balance
    FROM Customers2
    WHERE AccountType = 'Savings'
  ) LOOP
    UPDATE Customers2
    SET Balance = Balance + (Balance * 0.01)
    WHERE CustomerID = rec.CustomerID;
  END LOOP;

  COMMIT;
END;
/
EXECUTE ProcessMonthlyInterest;

SELECT * FROM CUSTOMERS2;
```

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

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

PL-SQL_Ex3-Scenario2.sql

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(
  p_Department IN VARCHAR2,
  p_BonusPercent IN NUMBER
) AS
BEGIN
  UPDATE Employees
  SET Salary = Salary + (Salary * p_BonusPercent / 100)
  WHERE Department = p_Department;

  COMMIT;
END;
/
EXECUTE UpdateEmployeeBonus('IT', 5);

SELECT * FROM EMPLOYEES;
```

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

r31.1

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

Download ▾ Execution time: 0.004 seconds

	EMPLOYEEID	NAME	DEPARTMENT	SALARY
1	101	Jagan	IT	63000
2	102	Saran	HR	55000
3	103	Manoj	Finance	70000
4	104	Naren	IT	65100
5	105	Lingesh	Finance	68000

Scenario 3 :

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

PL-SQL_Ex3-Scenario3.sql

```
CREATE OR REPLACE PROCEDURE TransferFunds(
  p_FromCustID IN NUMBER,
  p_ToCustID IN NUMBER,
  p_Amount IN NUMBER
) AS
  v_Balance NUMBER;
BEGIN
  -- Check balance of sender
  SELECT Balance INTO v_Balance
  FROM Customers2
  WHERE CustomerID = p_FromCustID;

  IF v_Balance < p_Amount THEN
    RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds for transfer');
  END IF;

  -- Deduct from source
  UPDATE Customers2
  SET Balance = Balance - p_Amount
  WHERE CustomerID = p_FromCustID;

  -- Add to target
  UPDATE Customers2
  SET Balance = Balance + p_Amount
  WHERE CustomerID = p_ToCustID;

```


r31.1

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

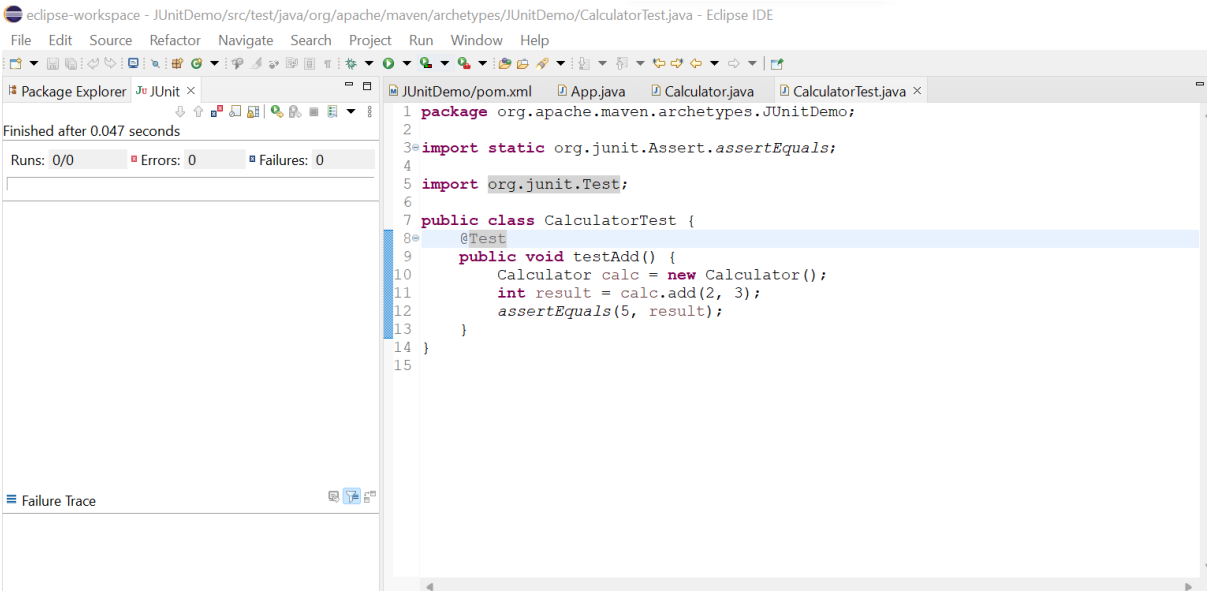
Download ▾ Execution time: 0.002 seconds

	CUSTOMERID	NAME	ACCOUNTTYPE	BALANCE	INTERESTRATE
1	1	Arjun	Savings	9100	1
2	2	Pranav	Current	6000	1
3	3	Arun	Savings	20200	1
4	4	Malini	Savings	8080	1
5	5	Priya	Current	12000	1

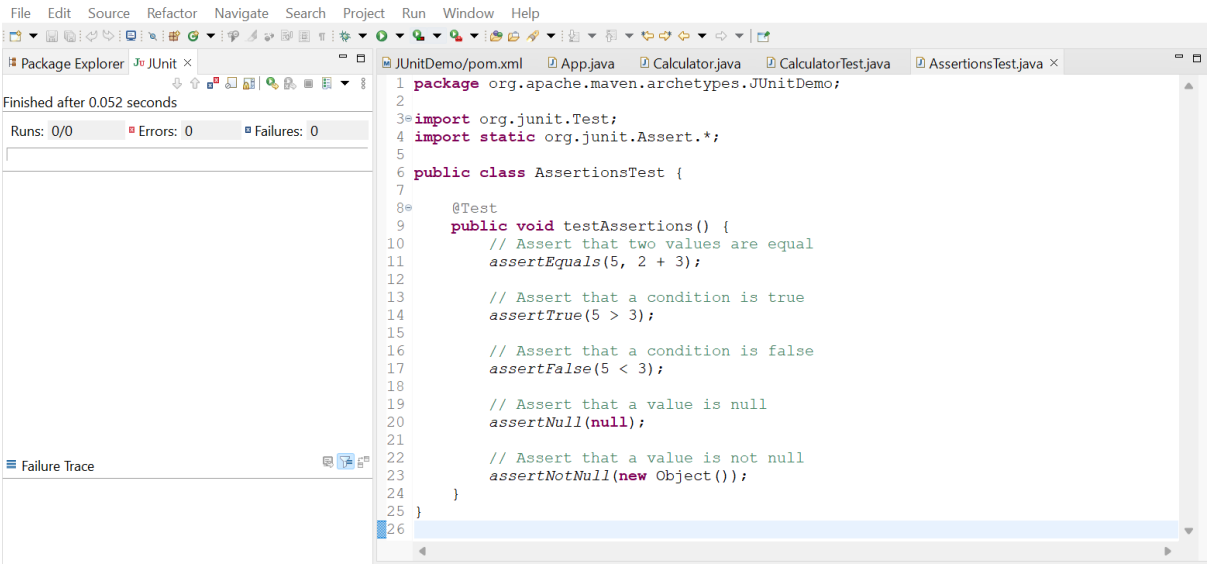
TDD USING JUNIT5 AND MOCKITO

JUnit basic testing exercises

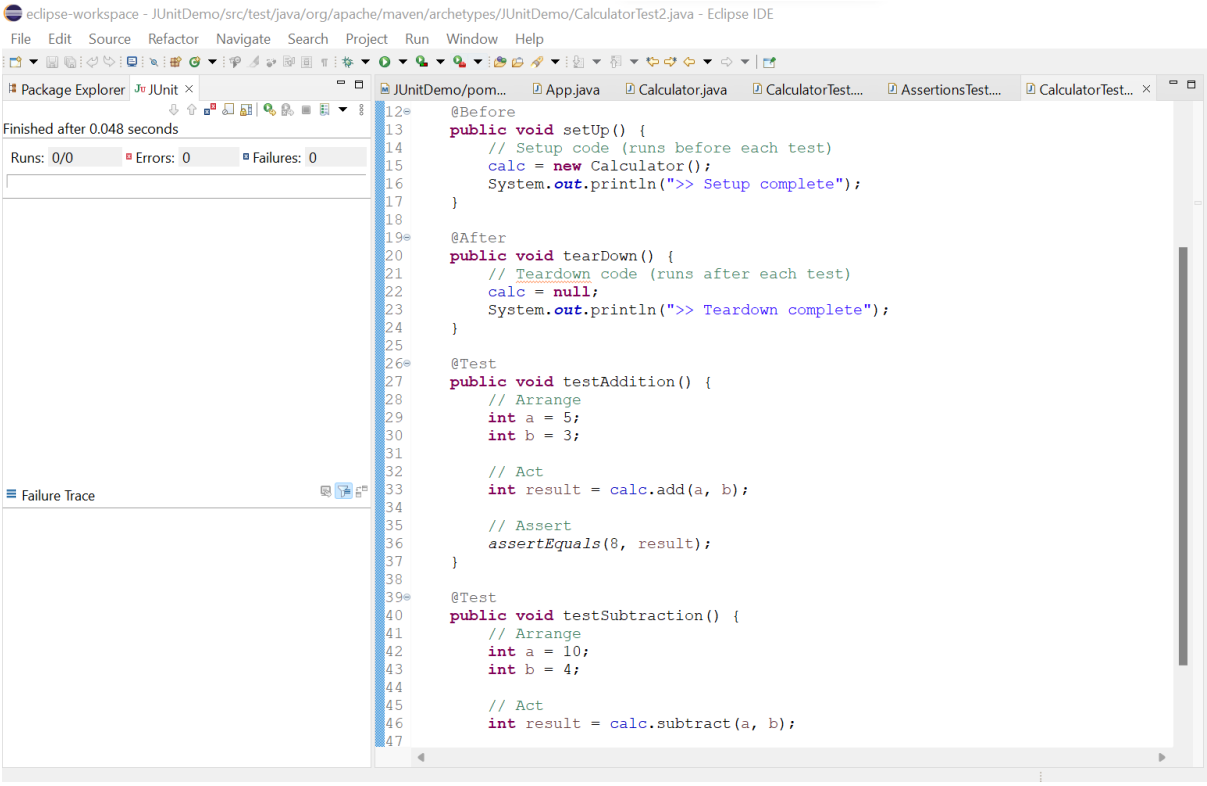
Excercise 1 : Setting up JUnit



Exercise 3 : Assertions in JUnit

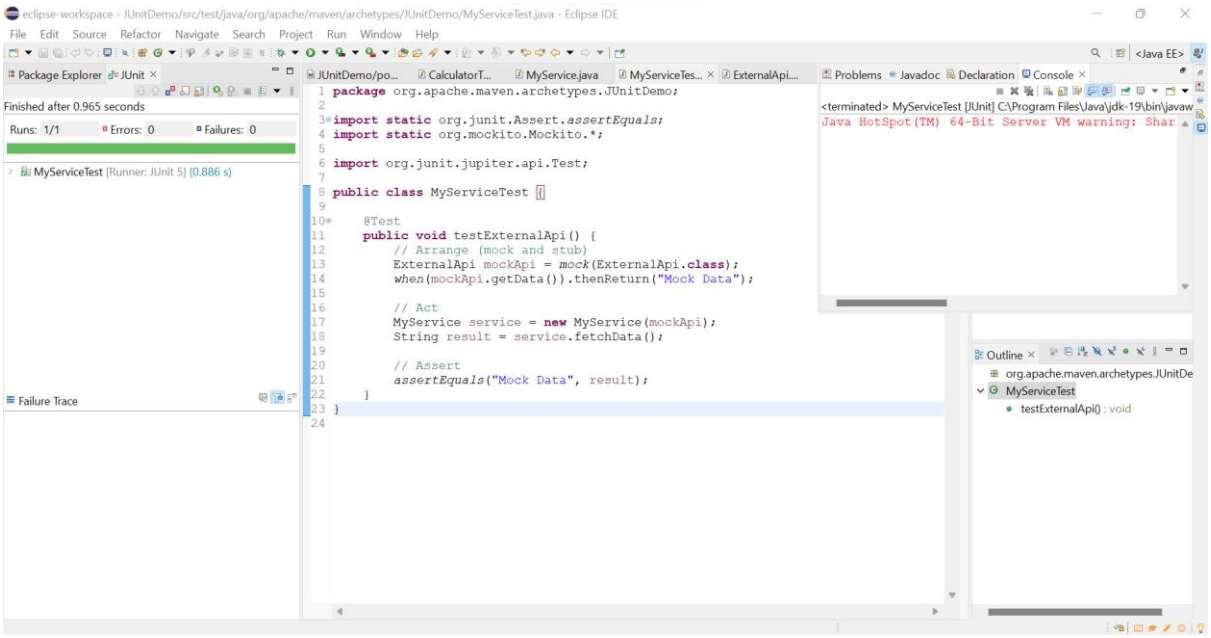


Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

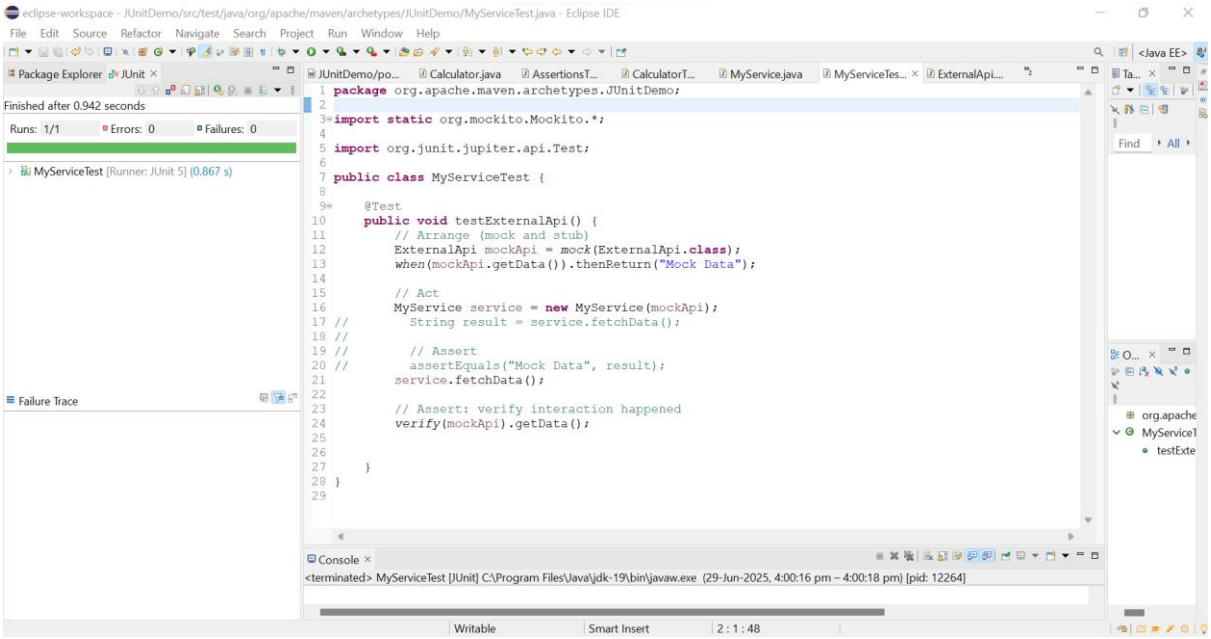


Mockito Exercises

Exercise 1 : Mocking and Stubbing



Excercise 2 : Verifying Interactions



<u>SL4J Logging Framework</u>

Exercise 1 : Logging Error Messages and Warning Levels