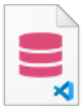
PLSQL programming Exercises

Exercise 1: Control Structures

Sample_Schema.sql

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.



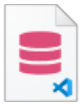
Scenario_1.sql

OUTPUT:

```
1% discount applied for Jvethesh
1% discount applied for Divya
```

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.



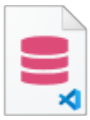
Scenario_2.sql

OUTPUT:

```
Jvethesh is promoted to VIP.
Ravi is promoted to VIP.
```

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.



Scenario_3.sql

OUTPUT:

```
Reminder: Loan ID 101 is due for Jvethesh on 05-JUL-2025
Reminder: Loan ID 103 is due for Ravi on 30-JUN-2025
```

Exercise 3: Stored Procedures

Sample_Schema.sql

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.

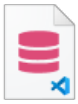
Scenario_1.sql

OUTPUT:

```
Monthly interest processed for all savings accounts.
```

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.



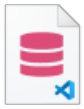
Scenario_2.sql

OUTPUT:

```
Bonus applied to employees in Sales department.
```

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.



Scenario_3.sql

OUTPUT:

```
₹500 transferred from account 101 to account 102.
```

JUnit Testing Exercises

Exercise 1: Setting Up JUnit

Scenario:

You need to set up JUnit in your Java project to start writing unit tests.

Steps:

1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).

2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml:

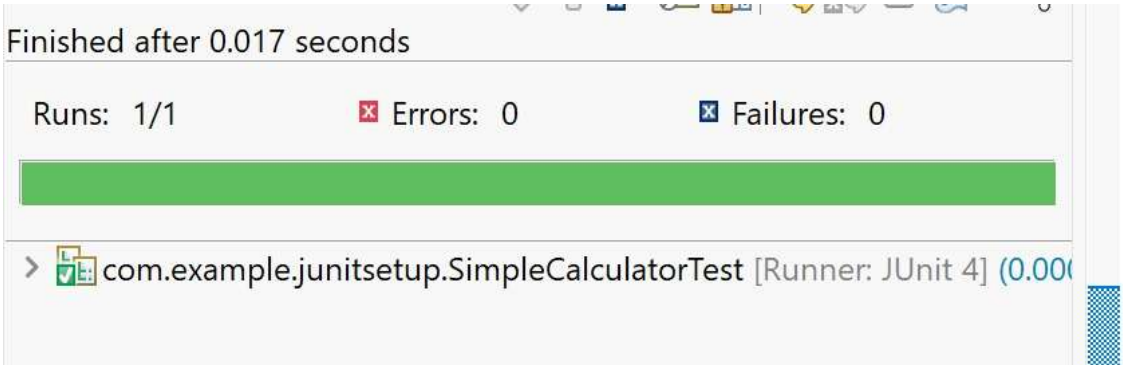
```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13.2</version>
  <scope>test</scope>
</dependency>
```

3. Create a new test class in your project.

SimpleCalculator.java

SimpleCalculatorTest.java

OUTPUT:



Exercise 3: Assertions in JUnit Scenario:

You need to use different assertions in JUnit to validate your test results.

Steps:

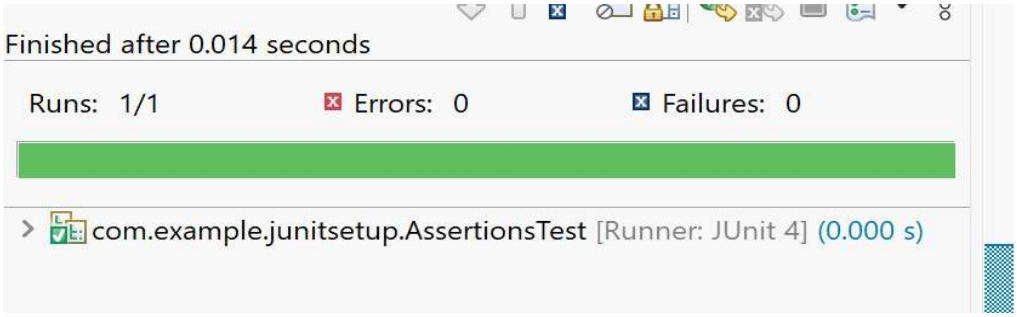
1. Write tests using various JUnit assertions.

Solution Code:

```
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());
 }
 }
```

AssertionsTest.java

OUTPUT:



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

Scenario: You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

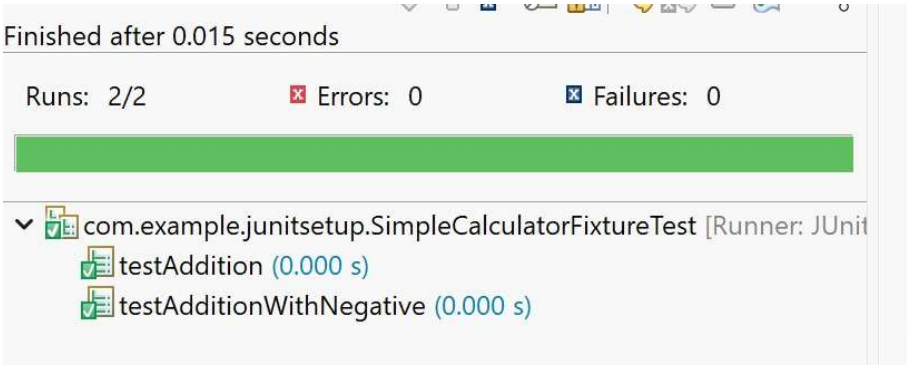
Steps:

1. Write tests using the AAA pattern.

2. Use @Before and @After annotations for setup and teardown methods.

SimpleCalculatorFixtureTest.java

OUTPUT:



Mockito Hands-On Exercises

Exercise 1: Mocking and Stubbing

Scenario:

You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.

Steps:

1. Create a mock object for the external API.

2. Stub the methods to return predefined values.

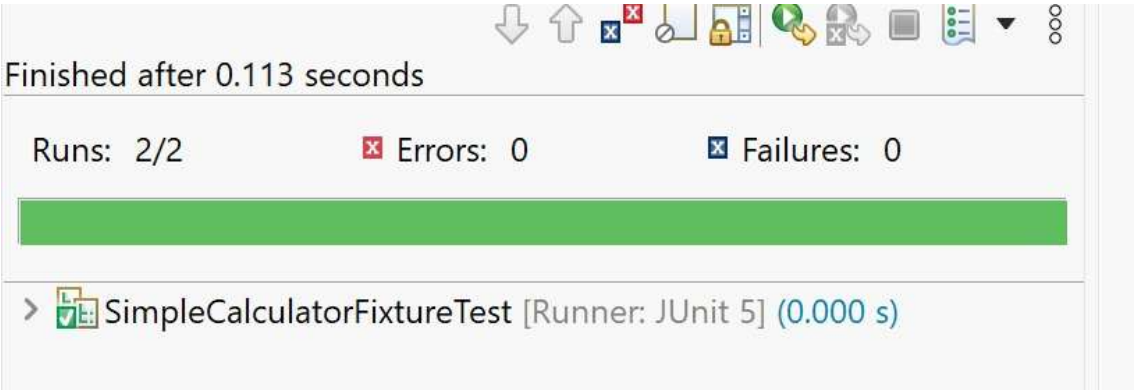
3. Write a test case that uses the mock object.

ExternalApi.java

SimpleCalculatorFixtureTest.java

MyServiceTest.java

OUTPUT:



Exercise 2: Verifying Interactions

Scenario:

You need to ensure that a method is called with specific arguments.

Steps:

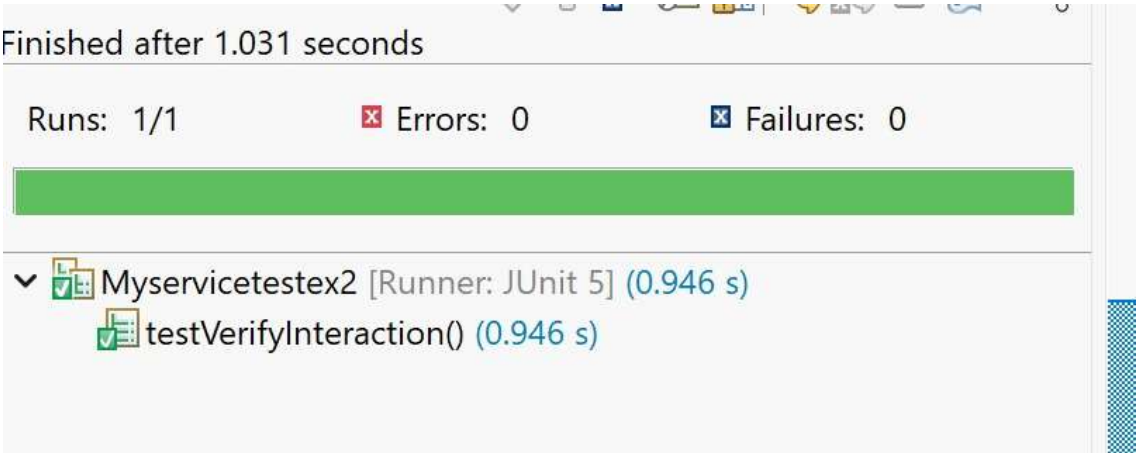
1. Create a mock object.

2. Call the method with specific arguments.

3. Verify the interaction.

Myservicetestex2.jav
a

OUTPUT:



Logging using SLF4J

Exercise 1: Logging Error Messages and Warning Levels

Task: Write a Java application that demonstrates logging error messages and warning levels using SLF4J.

1. Add SLF4J and Logback dependencies to your `pom.xml` file:

```
<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>
```

2. Create a Java class that uses SLF4J for logging:

LoggingExample.java

OUTPUT: