**Exercise 1: Control Structures**

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

DECLARE

    CURSOR cur\_customers IS

        SELECT l.customer\_id, l.interest\_rate,

               c.first\_name, c.last\_name, c.balance,c.AGE

        FROM loans l

        JOIN customers c ON l.customer\_id = c.customer\_id

        WHERE c.age > 60;

BEGIN

    FOR rec IN cur\_customers LOOP

        UPDATE loans

        SET interest\_rate = rec.interest\_rate - 1

        WHERE customer\_id = rec.customer\_id;

        DBMS\_OUTPUT.PUT\_LINE('Applied 1% discount to Customer ID: ' || rec.customer\_id ||

                             ', Name: ' || rec.first\_name || ' ' || rec.last\_name ||

                             ', Balance: $' || rec.balance|| ', Age:' || rec.AGE);

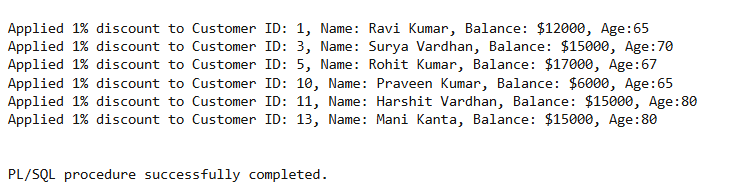
    END LOOP;

    COMMIT;

END;

/

**Output:**

****

**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.

**DECLARE**

  CURSOR cur\_balance IS

        SELECT c.customer\_id, c.balance, l.loan\_id, l.due\_date, l.interest\_rate

        FROM customers c

        JOIN loans l ON c.customer\_id = l.customer\_id

        WHERE c.balance > 10000;

BEGIN

    FOR rec IN cur\_balance LOOP

        UPDATE customers

        SET isVIP = 'TRUE'

        WHERE customer\_id = rec.customer\_id;

        DBMS\_OUTPUT.PUT\_LINE('Promoted to VIP: Customer ID ' || rec.customer\_id ||

                             ', Interest Rate: ' || rec.interest\_rate ||

                             ', Due Date: ' || TO\_CHAR(rec.due\_date, 'DD-MON-YYYY'));

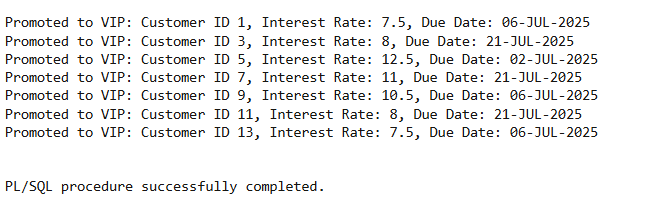
    END LOOP;

    COMMIT;

END;

/

**Output:**



**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.

DECLARE

    CURSOR due\_soon\_loans IS

        SELECT l.loan\_id, l.customer\_id, l.due\_date,

               c.first\_name, c.last\_name

        FROM loans l

        JOIN customers c ON l.customer\_id = c.customer\_id

        WHERE l.due\_date BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

    FOR rec IN due\_soon\_loans LOOP

        DBMS\_OUTPUT.PUT\_LINE(' Reminder: Loan ID ' || rec.loan\_id ||

                             ' for customer ' || rec.first\_name || ' ' || rec.last\_name ||

                             ' (Customer ID: ' || rec.customer\_id || ') is due on ' ||

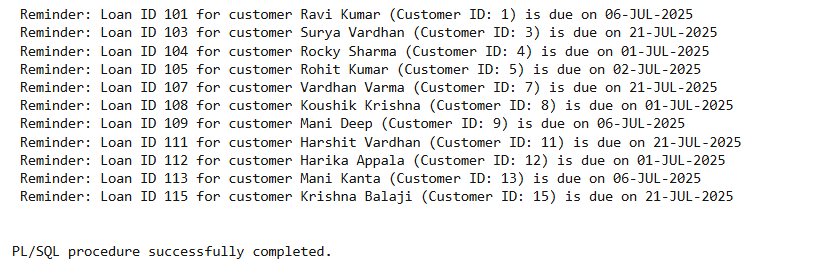
                             TO\_CHAR(rec.due\_date, 'DD-MON-YYYY'));

    END LOOP;

END;

/

**Output:**

****

**Junit\_Basic Testing Exercise:**

**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>

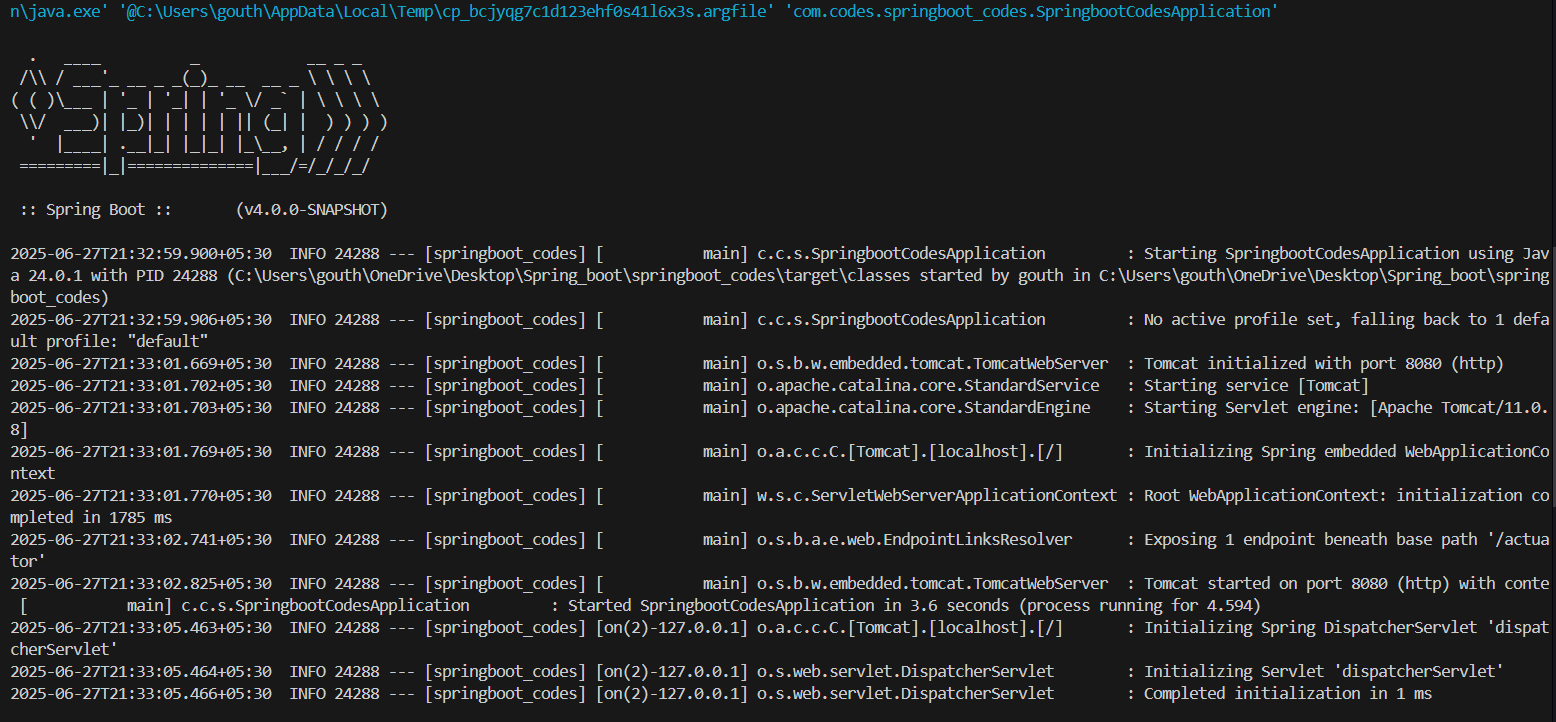
<version>4.13.2</version>

<scope>test</scope>

</dependency>

3. Create a new test class in your project.

**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:

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);

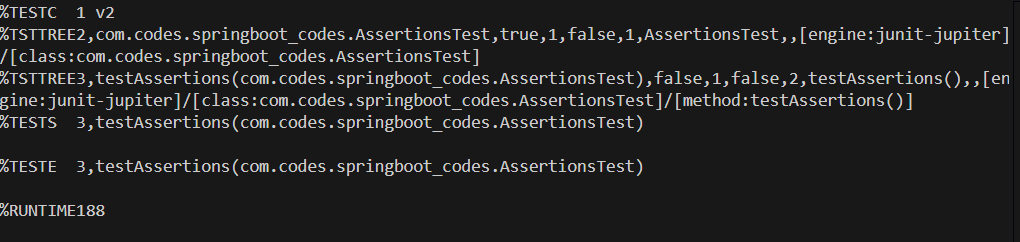
// Assert not null

assertNotNull(new Object());

}

}

**Output;**

****

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

Scenario:

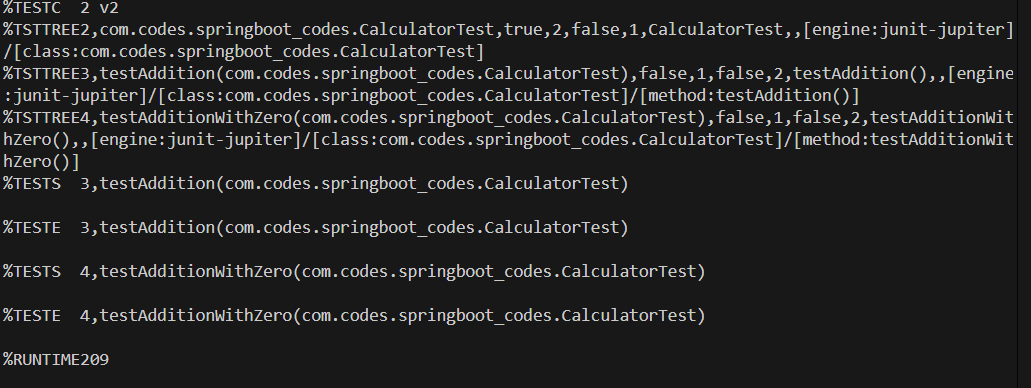
You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

Steps:

1. Write tests using the AAA pattern.

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

**Output:**

****

**Mockito 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.

Solution Code:

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class); when(mockApi.getData()).thenReturn("Mock Data");

MyService service = new MyService(mockApi);

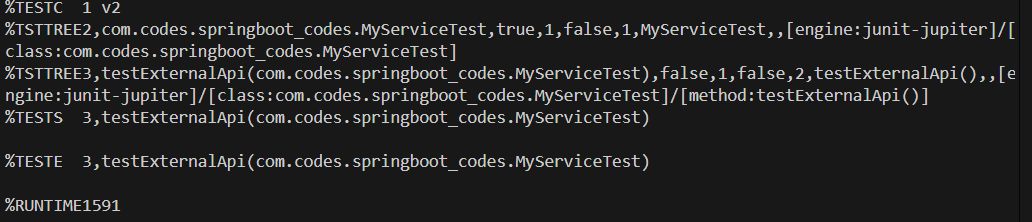
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Output:**

****

**Exercise 2: Verifying Interactions**

**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.

Solution Code:

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

MyService service = new MyService(mockApi);

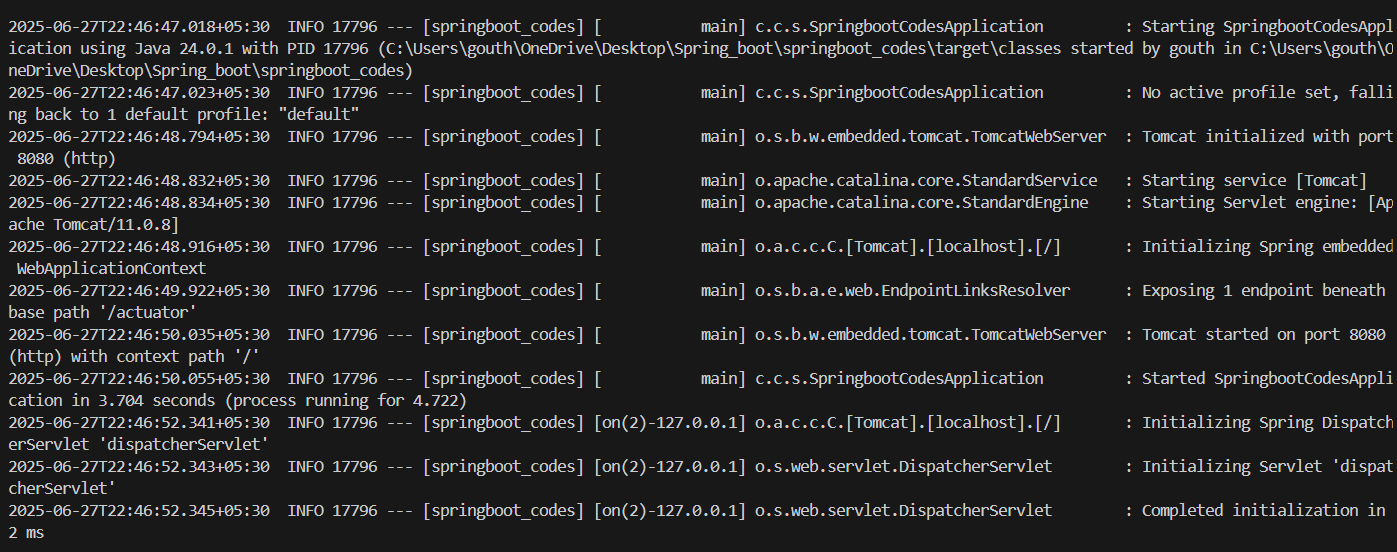
service.fetchData();

verify(mockApi).getData();

}

}

**Output:**

****

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

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

<dependency>

<groupld>org.slf4j</groupld>

<artifactld>slf4j-api</artifactid>

<version>1.7.30</version>

</dependency>

<dependency>

<groupld>ch.qos.logback</groupId>

<artifactld>logback-classic</artifactld>

<version>1.2.3</version>

</dependency> //Create java class

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) {

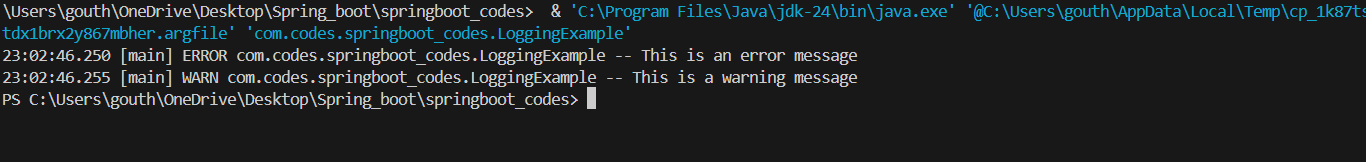
logger.error("This is an error message");

logger.warn("This is a warning message");

}

}

**Output:**

****