**WEEK 02**

**1.Control Structures**

DECLARE

num NUMBER := 5;

i NUMBER := 1;

BEGIN

IF num > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Positive');

ELSIF num = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Zero');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Negative');

END IF;

CASE

WHEN num = 1 THEN DBMS\_OUTPUT.PUT\_LINE('One');

WHEN num = 5 THEN DBMS\_OUTPUT.PUT\_LINE('Five');

ELSE DBMS\_OUTPUT.PUT\_LINE('Other');

END CASE;

FOR i IN 1..3 LOOP

DBMS\_OUTPUT.PUT\_LINE('FOR loop i = ' || i);

END LOOP;

i := 1;

WHILE i <= 3 LOOP

DBMS\_OUTPUT.PUT\_LINE('WHILE loop i = ' || i);

i := i + 1;

END LOOP;

i := 1;

LOOP

DBMS\_OUTPUT.PUT\_LINE('LOOP i = ' || i);

i := i + 1;

EXIT WHEN i > 3;

END LOOP;

IF num = 5 THEN

GOTO jump;

END IF;

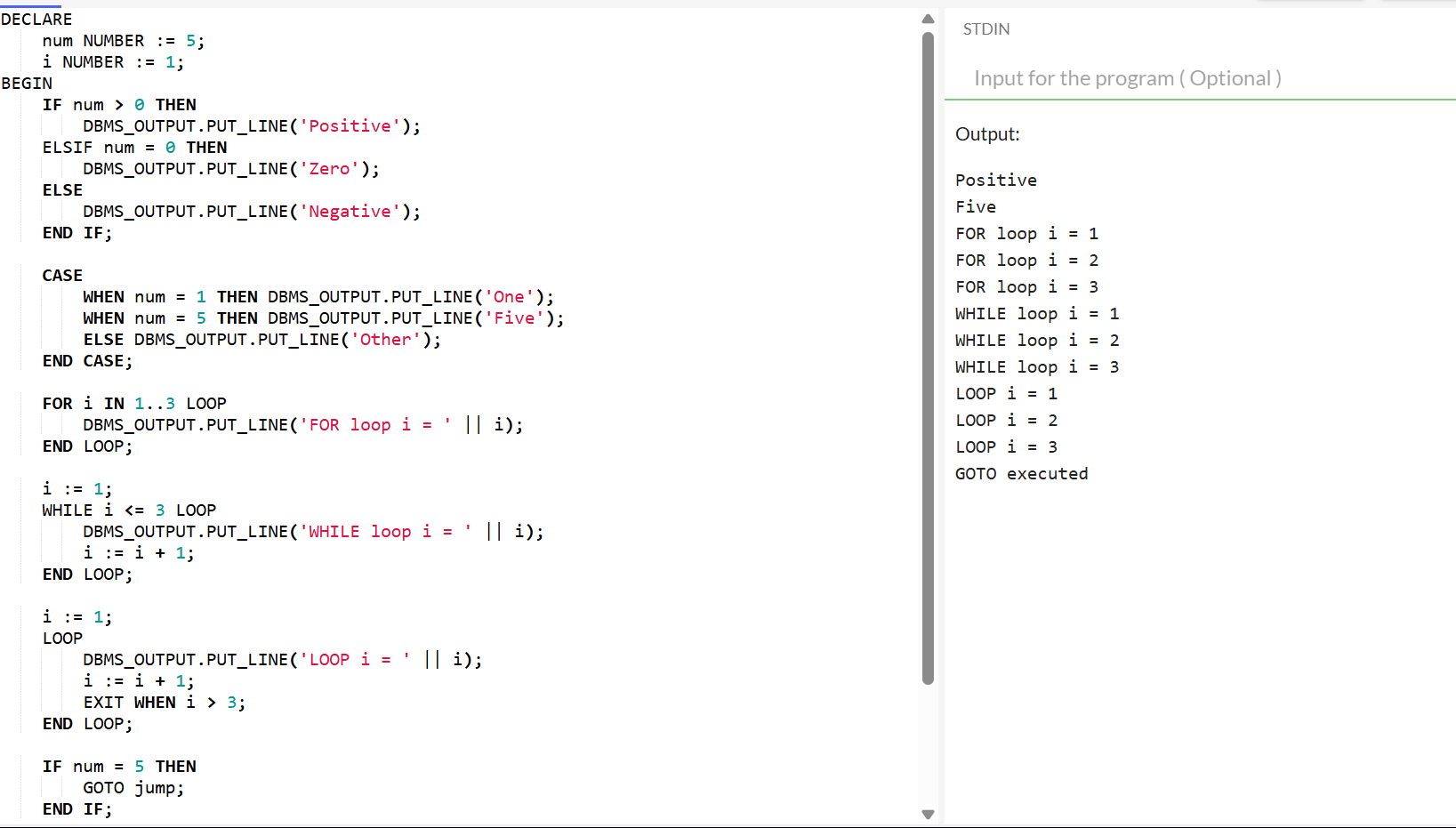
DBMS\_OUTPUT.PUT\_LINE('This line is skipped');

<<jump>>

DBMS\_OUTPUT.PUT\_LINE('GOTO executed');

END;

**OUTPUT :**



**2.Stored Procedures**

SET SERVEROUTPUT ON;

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE emp';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE emp (

empno NUMBER PRIMARY KEY,

ename VARCHAR2(100),

sal NUMBER

);

INSERT INTO emp VALUES (7369, 'SMITH', 3000);

INSERT INTO emp VALUES (7499, 'ALLEN', 3500);

COMMIT;

CREATE OR REPLACE PROCEDURE show\_employee\_details(p\_emp\_id IN NUMBER)

IS

v\_name VARCHAR2(100);

v\_salary NUMBER;

BEGIN

SELECT ename, sal INTO v\_name, v\_salary

FROM emp

WHERE empno = p\_emp\_id;

DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_name);

DBMS\_OUTPUT.PUT\_LINE('Salary: ' || v\_salary);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employee found for empno = ' || p\_emp\_id);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/

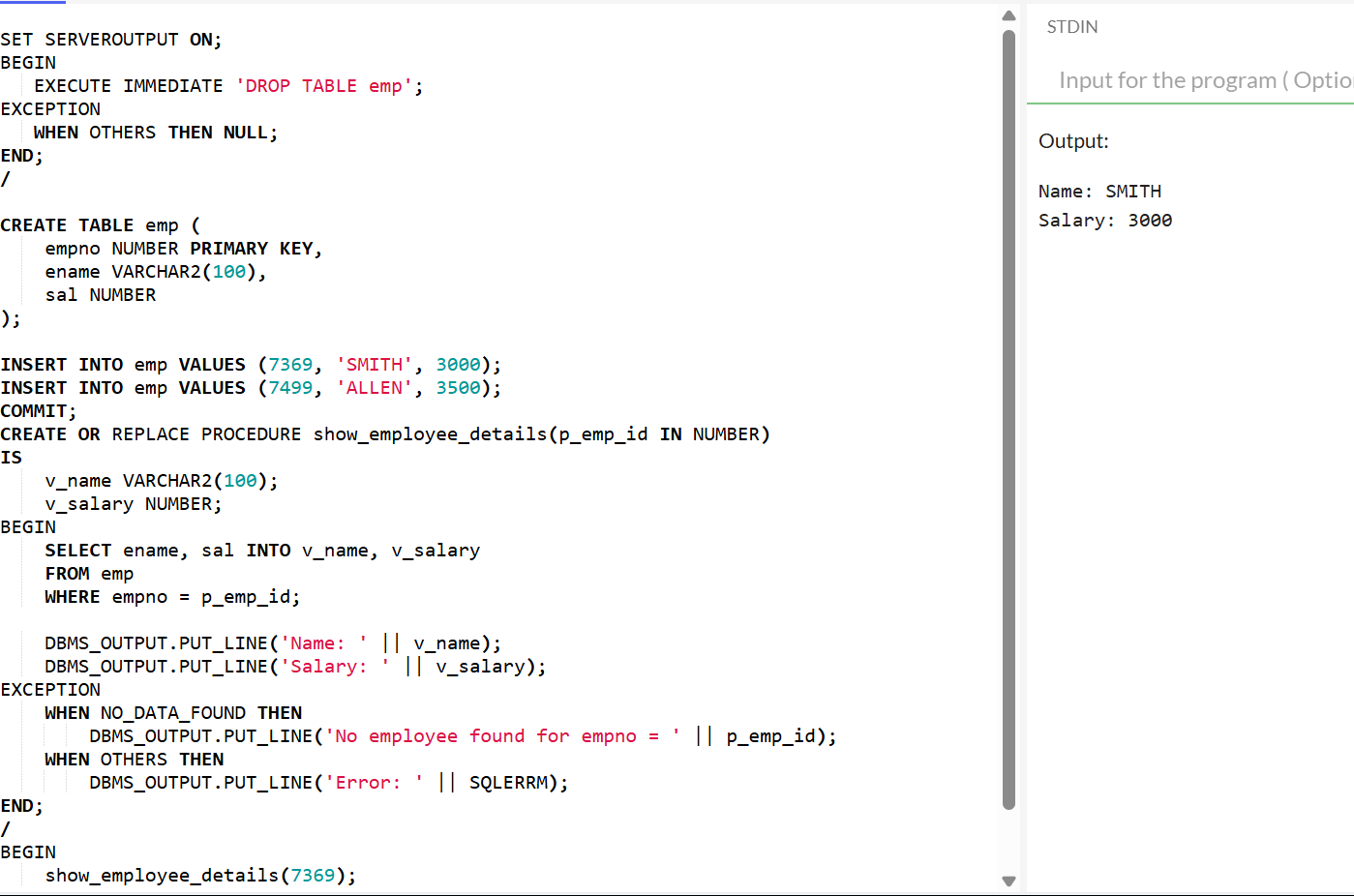
BEGIN

show\_employee\_details(7369);

END;

/

**OUTPUT :**

****

**3.Setting Up Junit**

**UserServiceTest.java**

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.junit.jupiter.MockitoExtension;

import static org.mockito.Mockito.\*;

import static org.junit.jupiter.api.Assertions.\*;

class UserServiceTest {

UserRepository userRepository;

UserService userService;

void testGetUserNameById() {

when(userRepository.findNameById(1)).thenReturn("Alice");

String result = userService.getUserNameById(1);

assertEquals("Alice", result);

verify(userRepository).findNameById(1);

}

}

**UserService.java**

public class UserService {

private final UserRepository repo;

public UserService(UserRepository repo) {

this.repo = repo;

}

public String getUserNameById(int id) {

return repo.findNameById(id);

}

}

**UserRepository.java**

public interface UserRepository {

String findNameById(int id);

}

**4.Assertions in Junit**

**OrderServiceTest.java**

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.junit.jupiter.MockitoExtension;

import java.util.Arrays;

import java.util.List;

import static org.mockito.Mockito.\*;

import static org.junit.jupiter.api.Assertions.\*;

class OrderServiceTest {

OrderRepository orderRepository;

OrderService orderService;

void testGetOrders() {

List<String> mockOrders = Arrays.asList("Item1", "Item2");

when(orderRepository.fetchOrders()).thenReturn(mockOrders);

List<String> orders = orderService.getOrders();

assertNotNull(orders);

assertEquals(2, orders.size());

assertTrue(orders.contains("Item1"));

assertFalse(orders.isEmpty());

assertLinesMatch(Arrays.asList("Item1", "Item2"), orders);

verify(orderRepository).fetchOrders();

}

}

**OrderService.java**

import java.util.List;

public class OrderService {

private final OrderRepository orderRepository;

public OrderService(OrderRepository orderRepository) {

this.orderRepository = orderRepository;

}

public List<String> getOrders() {

return orderRepository.fetchOrders();

}

}

**OrderRepository.java**

import java.util.List;

public interface OrderRepository {

List<String> fetchOrders();

}

**5. Arrange-Act-Assert (AAA) Pattern, Text Fixtures, Setup and Teardown Methods in Junit**

**PaymentServiceTest.java**

import org.junit.jupiter.api.\*;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.\*;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.\*;

class PaymentServiceTest {

PaymentGateway gateway;

PaymentService service;

PaymentRequest request;

void setUp() {

request = new PaymentRequest("user123", 100.0);

}

void tearDown() {

request = null;

}

void testSuccessfulPayment() {

when(gateway.charge("user123", 100.0)).thenReturn(true);

boolean result = service.process(request);

assertTrue(result);

verify(gateway).charge("user123", 100.0);

}

void testFailedPayment() {

when(gateway.charge("user123", 100.0)).thenReturn(false);

boolean result = service.process(request);

assertFalse(result);

verify(gateway).charge("user123", 100.0);

}

}

**PaymentService.java**

public class PaymentService {

private final PaymentGateway gateway;

public PaymentService(PaymentGateway gateway) {

this.gateway = gateway;

}

public boolean process(PaymentRequest request) {

return gateway.charge(request.getUserId(), request.getAmount());

}

}

**PaymentGateway.java**

public interface PaymentGateway {

boolean charge(String userId, double amount);

}

**PaymentRequest.java**

public class PaymentRequest {

private final String userId;

private final double amount;

public PaymentRequest(String userId, double amount) {

this.userId = userId;

this.amount = amount;

}

public String getUserId() {

return userId;

}

public double getAmount() {

return amount;

}

}

**6. Mocking and Stubbing**

**NotificationServiceTest.java**

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.junit.jupiter.MockitoExtension;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.\*;

class NotificationServiceTest {

EmailClient emailClient;

NotificationService notificationService;

void testSendNotificationSuccess() {

when(emailClient.sendEmail("test@example.com", "Hello")).thenReturn(true);

boolean result = notificationService.notify("test@example.com", "Hello");

assertTrue(result);

verify(emailClient).sendEmail("test@example.com", "Hello");

}

void testSendNotificationFailure() {

when(emailClient.sendEmail("fail@example.com", "Hi")).thenReturn(false);

boolean result = notificationService.notify("fail@example.com", "Hi");

assertFalse(result);

verify(emailClient).sendEmail("fail@example.com", "Hi");

}

}

**NotificationService.java**

public class NotificationService {

private final EmailClient emailClient;

public NotificationService(EmailClient emailClient) {

this.emailClient = emailClient;

}

public boolean notify(String recipient, String message) {

return emailClient.sendEmail(recipient, message);

}

}

**EmailClient.java**

public interface EmailClient {

boolean sendEmail(String recipient, String message);

}

**7. Verifying Interactions**

**AuditServiceTest.java**

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.junit.jupiter.MockitoExtension;

import static org.mockito.Mockito.\*;

class AuditServiceTest {

Logger logger;

AuditService auditService;

void testLogEvent() {

auditService.logEvent("LOGIN");

verify(logger).log("LOGIN");

}

void testLogEventCalledOnce() {

auditService.logEvent("LOGOUT");

verify(logger, times(1)).log("LOGOUT");

}

void testLogEventNeverCalledWithWrongInput() {

auditService.logEvent("UPDATE");

verify(logger, never()).log("DELETE");

}

}

**AuditService.java**

public class AuditService {

private final Logger logger;

public AuditService(Logger logger) {

this.logger = logger;

}

public void logEvent(String event) {

logger.log(event);

}

}

**Logger.java**

public interface Logger {

void log(String message);

}

**8. Logging Error Messages and Warning Levels**

**Add SLF4J Dependency**

**Using Maven:**

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>2.0.9</version>

</dependency>

<dependency>

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

<artifactId>logback-classic</artifactId>

<version>1.4.11</version>

</dependency>

**Using Java**

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

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

try {

int result = 10 / 0; // will throw ArithmeticException

} catch (ArithmeticException ex) {

logger.error("An exception occurred: {}", ex.getMessage(), ex);

}

}

}