HANDS ON EXERCISE WEEK-2

1. Java DAO Class (BankDAO.java)

import java.sql.\*;

public class BankDAO {

private static final String URL = "jdbc:oracle:thin:@localhost:1521:xe";

private static final String USER = "your\_username";

private static final String PASS = "your\_password";

public String addCustomer(int id, String name, double balance) {

String sql = "{call add\_customer(?, ?, ?)}";

try (Connection conn = DriverManager.getConnection(URL, USER, PASS);

CallableStatement stmt = conn.prepareCall(sql)) {

stmt.setInt(1, id);

stmt.setString(2, name);

stmt.setDouble(3, balance);

stmt.execute();

return "Customer Added";

} catch (SQLException e) {

return "Error: " + e.getMessage();

}

}

public String deposit(int id, double amount) {

String sql = "{call deposit\_amount(?, ?)}";

try (Connection conn = DriverManager.getConnection(URL, USER, PASS);

CallableStatement stmt = conn.prepareCall(sql)) {

stmt.setInt(1, id);

stmt.setDouble(2, amount);

stmt.execute();

return "Deposit Successful";

} catch (SQLException e) {

return "Error: " + e.getMessage();

}

}

public String withdraw(int id, double amount) {

String sql = "{call withdraw\_amount(?, ?)}";

try (Connection conn = DriverManager.getConnection(URL, USER, PASS);

CallableStatement stmt = conn.prepareCall(sql)) {

stmt.setInt(1, id);

stmt.setDouble(2, amount);

stmt.execute();

return "Withdraw Successful";

} catch (SQLException e) {

return "Error: " + e.getMessage();

}

}

public double getBalance(int id) {

String sql = "{? = call get\_balance(?)}";

try (Connection conn = DriverManager.getConnection(URL, USER, PASS);

CallableStatement stmt = conn.prepareCall(sql)) {

stmt.registerOutParameter(1, Types.DOUBLE);

stmt.setInt(2, id);

stmt.execute();

return stmt.getDouble(1);

} catch (SQLException e) {

e.printStackTrace();

return -1;

}

}

}

1. Unit Testing with JUnit (BankDAOTest.java.

import org.junit.jupiter.api.Test;

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

public class BankDAOTest {

BankDAO dao = new BankDAO();

@Test

public void testAddCustomer() {

String result = dao.addCustomer(1, "Alice", 5000);

assertEquals("Customer Added", result);

}

@Test

public void testDeposit() {

dao.deposit(1, 1000);

double balance = dao.getBalance(1);

assertTrue(balance >= 6000);

}

@Test

public void testWithdrawWithSufficientBalance() {

String result = dao.withdraw(1, 500);

assertEquals("Withdraw Successful", result);

}

@Test

public void testWithdrawWithInsufficientBalance() {

String result = dao.withdraw(1, 100000); // huge amount

assertTrue(result.contains("Insufficient"));

}

@Test

public void testGetBalance() {

double balance = dao.getBalance(1);

assertTrue(balance >= 0);

}

}

3.PL/SQL: Bulk Update Procedur.

-- Table for inventory

CREATE TABLE products (

product\_id NUMBER PRIMARY KEY,

product\_name VARCHAR2(100),

quantity NUMBER

);

-- Procedure for bulk updating product quantities

CREATE OR REPLACE PROCEDURE update\_product\_quantities (

p\_product\_ids IN SYS.ODCINUMBERLIST,

p\_quantities IN SYS.ODCINUMBERLIST

) AS

BEGIN

FOR i IN 1..p\_product\_ids.COUNT LOOP

UPDATE products

SET quantity = quantity + p\_quantities(i)

WHERE product\_id = p\_product\_ids(i);

END LOOP;

END;

/