DOCUMENTATIE

TEMA 3

CURCANU SAMUEL-ADRIAN

GRUPA:30227

# 

# CUPRINS

[1. Obiectivul temei 3](file:///C:\Users\Samuel-Adrian\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///C:\Users\Samuel-Adrian\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297886)

[3. Proiectare 3](file:///C:\Users\Samuel-Adrian\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297887)

4. Implementare…………………...……………………………………………………………………………..4

5. Rezultate………………………………………………………………………………………………………5

6. Concluzii……………………………………………………………………………………………………...5

7. Bibliografie………………………………………………………………………………………………...…5

**1.Obiectivul temei**

Luăm în considerare o aplicație de gestionare a comenzilor pentru procesarea comenzilor de la un depozit. Bazele de date relaționale ar trebui utilizate pentru a stoca produsele, clienții și comenzile. Aplicația ar trebui să fie proiectată conform modelului de arhitectură stratificată și ar trebui să utilizeze (minim) următoarele clase:

• Model classes - represent the data models of the application

• Business Logic classes - contain the application logic

• Presentation classes – GUI related classes

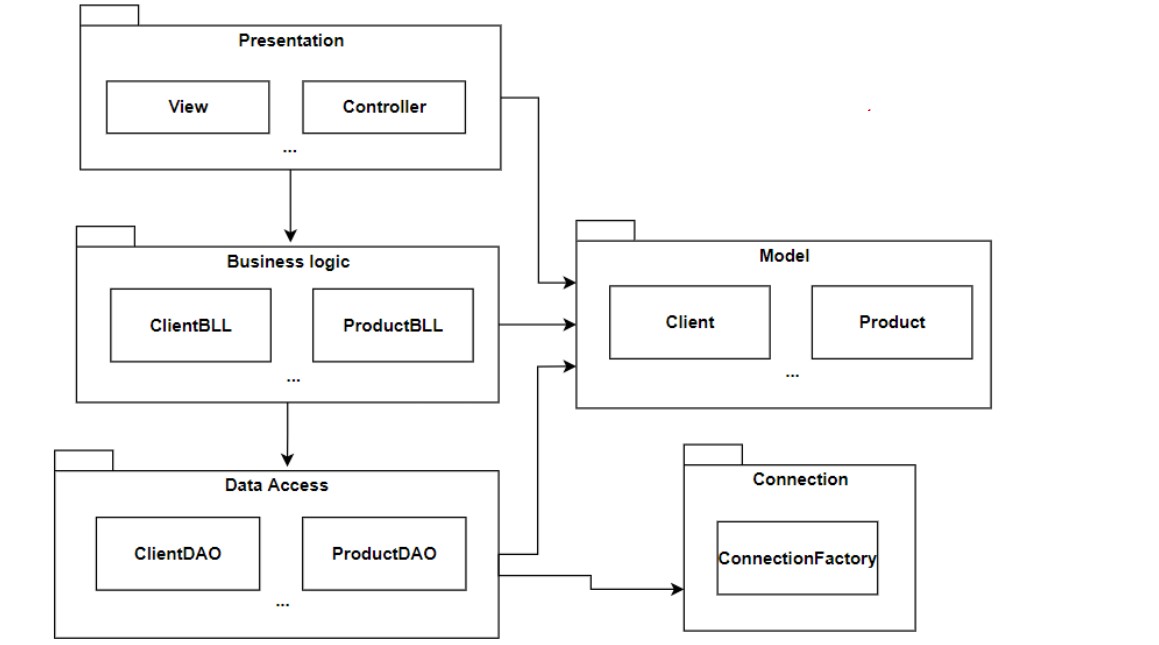
• Data access classes - classes that contain the access to the database"

**2.Analiza problemei, modelare, scenarii, cazuri de utilizare**

Pentru functionarea corecta a problemei vom folosi modelul pus in prezentarea problemei si vom face unele modificari ulterioare

**3.Proiectare**

Tema este creeata in stil OOP, variabilele sunt toate private putand fi accesate doar prin metode de get si set. Pentru o vizualizare cat mai buna clasele sunt in pachete diferite avand un nume corespunzator, pentru a ajuta utilizatorul sa gaseasca portiuni de cod dorite cat mai usor.



**4.Implementare**

Clasa Client din pachetul Model reprezintă un client cu informațiile personale asociate. Aceasta conține următoarele atribute private: id (de tip long), name (de tip String), address (de tip String) și number (de tip String), care stochează ID-ul, numele, adresa și numărul de telefon al clientului.

Clasa Orders din pachetul Model reprezintă o comandă cu detaliile sale.

Această clasă conține următoarele atribute private: Id (de tip long), idClient (de tip long), productId (de tip long) și quantity (de tip String). Acestea reprezintă ID-ul comenzii, ID-ul clientului asociat comenzii, ID-ul produsului asociat comenzii și cantitatea produsului comandat.

Clasa Product din pachetul Model reprezintă un produs împreună cu detaliile sale.

Această clasă conține următoarele atribute private: id (de tip long), name (de tip String), price (de tip String) și quantity (de tip String). Acestea reprezintă ID-ul produsului, numele produsului, prețul produsului și cantitatea disponibilă a acestuia.

Clasa AbstractDAO<T> din pachetul DataAccess reprezintă o clasă abstractă generică utilizată pentru a realiza operații de bază pe obiecte persistente într-o bază de date.

Această clasă folosește reflexia pentru a obține informații despre tipul generic T și să realizeze operații precum inserarea, actualizarea, ștergerea și obținerea obiectelor din baza de date.

Clasa AbstractDAO<T> are următoarele metode și funcționalități principale:

Constructorul clasei utilizează reflexia pentru a obține tipul actual al clasei și să îl stocheze în variabila type.

Metoda createSelectQuery construiește o interogare SQL pentru a obține un obiect din baza de date în funcție de o anumită coloană specificată.

Metoda findById obține un obiect din baza de date bazat pe ID-ul specificat.

Metoda createDeleteQuery construiește o interogare SQL pentru a șterge un obiect din baza de date.

Metoda insertIntoReflection construiește o interogare SQL de inserare utilizând reflexia pentru a obține proprietățile obiectului.

Metoda findAll obține toate obiectele din baza de date.

Metoda getTable returnează un obiect JTable cu datele din baza de date.

Metoda createObjects creează obiecte bazate pe rezultatele unei interogări SQL și le adaugă într-o listă.

Metoda insert inserează un obiect în baza de date.

Metoda delete șterge un obiect din baza de date.

Metoda update actualizează o coloană specificată a unui obiect în baza de date.

Clasa AbstractDAO<T> este o clasă abstractă generată pentru a oferi funcționalități comune de acces la date pentru alte clase DAO specifice fiecărui tip de obiect din aplicație.

Clasele ClientDAO ProductDAO si OrderDAO din pachetul DataAccess extind clasa AbstractDAO<Client> și oferă operații specifice de acces la date pentru modelul Client.

Constructorul fiercarei clase inițializează clasa de bază AbstractDAO.

Clasele ClientBLL, ProductBLL, OrderBLL din pachetul BusinessLogic oferă operații de logică de afaceri pentru modelul Client,Product si Orders.

Clasa ConnectionFactory din pachetul Connection este responsabilă de crearea și gestionarea conexiunii la baza de date.

Această clasă conține următoarele metode și funcționalități:

Se definește un obiect de tip Logger pentru a înregistra mesajele de tip log.

Se specifică informațiile necesare pentru conexiunea la baza de date: driverul JDBC, URL-ul bazei de date, numele de utilizator și parola.

Se definește un obiect singleInstance de tip ConnectionFactory pentru a asigura că există o singură instanță a clasei în timpul execuției.

Constructorul clasei încarcă driverul JDBC necesar pentru conexiunea la baza de date.

Metoda createConnection este responsabilă de crearea efectivă a conexiunii la baza de date. Aceasta utilizează informațiile de configurare (URL, utilizator, parolă) pentru a stabili conexiunea și returnează obiectul Connection.

Metoda getConnection este utilizată pentru a obține o instanță a conexiunii la baza de date. Aceasta returnează conexiunea creată prin apelul metodei createConnection.

Metodele close(Connection connection), close(Statement statement) și close(ResultSet resultSet) sunt utilizate pentru închiderea conexiunii, a declarației (statement) și a obiectului ResultSet, respectiv. Aceste metode verifică mai întâi dacă obiectul respectiv nu este nul și apoi încearcă închiderea acestuia. În cazul în care apare o excepție la închidere, se înregistrează un mesaj de avertizare în jurnalul de loguri.

Clasa ConnectionFactory oferă o modalitate convenabilă de a obține și a închide conexiunea la baza de date în cadrul aplicației. Aceasta folosește driverul JDBC și informațiile de configurare specificate pentru a crea conexiuni și asigură închiderea corectă a acestora pentru a evita scurgerile de resurse și erori în timpul execuției.

Clasa Reflexion din pachetul Start oferă metode de utilitate pentru operații de reflexie.

Această clasă conține următoarele metode și funcționalități:

Metoda retrieveProperties este utilizată pentru a obține proprietățile și valorile unui obiect. Prin parcurgerea câmpurilor declarate ale obiectului, se obțin valorile acestora și se construiește o string care reprezintă proprietățile și valorile obiectului. Câmpurile obiectului sunt accesibile folosind metoda getDeclaredFields(), iar apoi se setează accesibilitatea câmpurilor utilizând metoda setAccessible(true). Valorile câmpurilor sunt obținute folosind metoda get(object), iar numele și valorile câmpurilor sunt adăugate la stringul de câmpuri și stringul de valori, respectiv.

Metoda delete este utilizată pentru a genera o interogare de ștergere pentru un obiect. Prin parcurgerea câmpurilor declarate ale obiectului, se construiește o string care reprezintă interogarea de ștergere. Numele tabelei este obținut folosind metoda getSimpleName(), iar condițiile de ștergere sunt adăugate într-un StringJoiner pe baza valorilor câmpurilor obiectului. Condițiile sunt create în formatul "nume\_câmp = 'valoare'", iar numele câmpului și valoarea sunt obținute folosind metoda getName() și, respectiv, get(object).

Clasa Reflexion furnizează utilități pentru a obține proprietățile și valorile unui obiect și pentru a genera interogări de ștergere pe baza obiectelor. Aceste metode folosesc reflecția pentru a accesa câmpurile și valorile obiectelor și sunt utile în contextul manipulării și gestionării obiectelor dinamic.

Clasa PhoneNumber din pachetul Validators implementează interfața Validator pentru a valida numerele de telefon pentru obiectele de tip Client.

Acest cod reprezintă o interfață grafică realizată în limbajul de programare Java, folosind biblioteca Swing. Interfața conține trei butoane pentru a accesa tabelele cu clienți, produse și comenzi. Fiecare buton are asociată o acțiune care afișează tabela corespunzătoare.

Metoda showClientsTable() afișează tabela cu clienți, folosind clasele ClientDAO și ClientBLL pentru a obține datele din baza de date și pentru a le afișa într-un obiect JTable. Această metodă este apelată atunci când utilizatorul apasă butonul "Clients".

Metoda showProductsTable() afișează tabela cu produse, folosind clasele ProductDAO și ProductBLL pentru a obține datele și pentru a le afișa într-un obiect JTable. Această metodă este apelată atunci când utilizatorul apasă butonul "Product".

Metoda showOrdersTable() afișează tabela cu comenzi, folosind clasele OrderDAO și OrderBLL pentru a obține datele și pentru a le afișa într-un obiect JTable. Această metodă este apelată atunci când utilizatorul apasă butonul "Orders".

Există și alte metode auxiliare, cum ar fi createClientFrame(), createProductFrame() și createOrderFrame(), care creează ferestre de dialog pentru adăugarea de noi clienți, produse și comenzi, respectiv.

**5.Rezultate**

Se vor adauga client comenzi sau produse care se vor afisa in timp real pe interfata. De asemenea se pot modifica datele introduce de la client si produse.

**6.Concluzii**

Tema a fost una destul de complexa punand la testare cunostintele de java, de asemenea aceasta tema se mai poate dezvolta ulterior pentru a o face mult mai complexa, cum ar fi creearea de facturi pentru fiecare comanda in parte etc.

**7.Bibliografie**

1. FUNDAMENTAL PROGRAMMING TECHNIQUES LABORATORY GUIDE - <https://dsrl.eu/courses/pt/>

2. http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html