**DOCUMENTATIE**

**TEMA 3**

**ORDER MANAGEMENT**

TEHNICI DE PROGRAMARE

UNIVERSITATEA TEHNICA DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICA SI CALCULATOARE

SPECIALIZAREA CALCULATOARE SI TEHNOLOGIA INFORMATIEI

*Piștănilă Ana-Maria*

*Grupa 30226*

CUPRINS

1. Obiectivul temei. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   1. Obiectivul principal. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   2. Obiective secundare. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
2. Analiza problemei. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.1 Modelare. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.2 Scenarii. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.3 Cazuri de utilizare. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1. Proiectare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.1 Decizii de proiectare. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.2 Diagrame UML. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.3 Structuri de date. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.4 Proiectare clase. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.5 Algoritmi. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1. Implementare. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
2. Rezultate. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
3. Concluzii. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
4. Bibliografie. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**1.OBIECTIVUL TEMEI**

* 1. *Obiectiv principal*

Obiectivul problemei a fost acela de a proiecta si implementa o aplicatie pentru managementul comenzilor unor clienti pentru un depozit , folosind baze de date relationale . Bazele de date relaționale sunt utilizate pentru a stoca produsele, clienții și comenzile .

Aplicația este structurata pe pachete, conform cerintei pentru a asigura protejarea accesului si controlul spatiului numelor.

Aplicația ar trebui să permită procesarea comenzilor dintr-un fișier text dat ca argument, sa efectueze operatiile corespunzatoare, sa salveze in baza de date si sa genereze rapoarte în format pdf .

* 1. *Obiective secundare*

Obiectivele secundare sunt:

* realizarea operatiilor de adaugare a unui client/produs/comanda, stergerea unui client, modificarea anumitor campuri din baza de date, crearea pdf-urilor si generarea tabelelor corespunzatoare fiecarui caz;
* folosirea corecta a conceptelor de programare OOP;
* alegerea structurilor de date;
* impartirea pe clase;
* scrirea unui cod care poate fi usor de inteles de catre oricine il citeste;
* dezvoltarea de use caseuri si scenarii;
* testare;

**2.ANALIZA PROBLEMEI**

Realizarea acestei probleme implica pe langa cunoasterea conceptelor OOP, cunoasterea notiunilor de baza pentru proiectarea unei baze de date.

O baza de date este un ansamblu structurat de date pentru a satisface simultan cerintele mai multor utilizatori intr-un mod selectiv si in timp util. Elementele dintr-o baza de date relationala sunt organizate ca un set de tabele, cu randuri si coloane. Tehnologia bazelor de date relationale ofera cel mai eficient si flexibil mod de a accesa informatiile structurate.

Baza de date relaționala defineste și manipulează informațiile scrise într-un limbaj de interogare structurat (SQL).

SQL este limbajul standard pentru tratarea bazelor de date relaționale. Poate fi utilizat pentru a insera, căuta și sterge înregistrările bazei de date.

In aplicatia mea, procesarea comenzilor este realizata cu ajutorul unui fisier text , primit ca argument .Citesc pe rand fiecare linie din fisierul de intrare si procesez fiecare comanda : ( de adaugare – inserez clientul sau produsul in baza de date , de stergere – sterg clientul sau produsul din baza de date , de adaugare a unei comenzi pentru un anumit client din baza de date, daca acea comanda a putut fi efectuata, de generare a rapoartelor – documente PDF cu evidenta clientilor , comenzilor , produselor dupa anumite comenzi ).

**3.PROIECTARE**

Structuri de date folosite:

-ArrayList- deoarece este o colectie simpla care poate stoca orice tip de obiect. Aceasta clasa ne permite sa construim un vector care sa creasca in marime, adaugand mereu elemente noi. In plus, avantajele majore ale acestei clase sunt utilizarea mai buna a memoriei si administrarea mai usoara a elementelor listei.

Proiectare clase

Proiectul este structurat in 4 pachete: *Access:*care contine clasele ConnectionDB, VerificareClient, VerificareProdus, VerificareComanda, *Business:*care contine clasa Validare, *Model:* care contine clasele Client, Order, Product si *Presentation*: care contine clasa Main.

Pachetul Access:

-Clasa ConnectionDB: In aceasta clasa este realizata conectarea la baze de date, respectiv inchiderea conexiunii. Accesul la baza de date se face prin intermediul unui driver specific tipului respectiv de baza de date. Acesta este responsabil cu accesul efectiv la datele stocate, fiind legatura între aplicatie si baza de date.O conexiune la o baza de date reprezinta un context prin care sunt trimise secvente SQL si primite rezultate. Am folosit clasa DriverManager pentru realizarea conexiunii, ce se ocupa cu înregistrarea driverelor ce vor fi folosite în aplicatie. Având în vedere faptul ca pot exista mai multe drivere înregistrate în memorie, se specifica prin intermediul unei adrese, DBURL ce driver sa fie folosit.Inchiderea conexiunii am realizat-o cu functia close() din clasa Connection.

-Clasa VerificareClient: In aceasta clasa am realizat 2 metode. Prima metoda ( addClient) are rolul de a adauga un client (nume, prenume, adresa) in baza de date, pe baza unei interogari SQL. Am folosit PreparedStatement pentru a transmite o instructiune SQL atunci cand obiectul este creat. Avantajul este ca aceasta declaratie este trimisa la SGBD imediat, unde este compilata. Apoi am folosit metoda executeUpdate() care întoarce un număr întreg având semnificaţia înregistrărilor afectate de expresia SQL(adica daca s-a putut face o inserare, o stergere sau o modificare a unui anumit camp-update). A doua metoda(deleteClient) are rolul de a sterge un anumit client in functie de un nume si un prenume dat ca si argument. Diferenta implementarii fata de metoda anterioara este ca interogarea SQL va fi diferita, iar argumentele vor fi sterse apeland clearParameters() care sterge imediat valorile parametrului curent.

-clasa VerificareProdus: In aceasta clasa am realizat 3 metode: Prima metoda (addProdus) care adauga un produs in baza de date(numele produsului, cantitatea si pretul). Implementarea este identica cu cea de la metoda addClient, singurele diferente constand in interogarea SQL si setarea parametrilor corespunzatori. A doua metoda(deleteProdus) sterge un anumit produs din baza de date dupa un nume introdus. Iar a treia metoda(updateProdus) modifica cantitatea unui anumit produs.

-clasa VerificareComanda: In aceasta clasa am realizat o singura metoda ( addComanda) care adauga o comanda( numele persoanei care a comandat, prenumele acestei persoane, numele produsului comandat, cantitatea comandata). Implementarea este la fel ca in cazurile precedente, insa cu setarea parametrilor corespunzatori .

Pachetul Business:

-clasa Validare: In aceasta clasa am implementat 3 metode asemanatoare. Prima metoda(adaugaClient) verifica daca datele preluate din fisier nu sunt vide, iar in cazul in care totul este in regula, apeleaza metoda addClient din VerificareClient pentru a adauga in baza de date respectivul client. A doua metoda(adaugaProdus) verifica acelasi lucru ca in cazul primei metode, insa pentru parametri corespunzatori clasei Produs. Daca totul este in regula, apeleaza metoda addProdus din VerificareProdus. A treia metoda(adaugaComanda) verifica acelasi lucru ca in cazul primei metode, insa pentru parametri corespunzatori clasei Order. Daca totul este in regula, apeleaza metoda addComanda din VerificareComanda.

Pachetul Model:

-clasa Client: In aceasta clasa am trei variabile: nume, prenume, adresa, un constructor care initializeaza aceste campuri si metode de get si set pentru acestea.

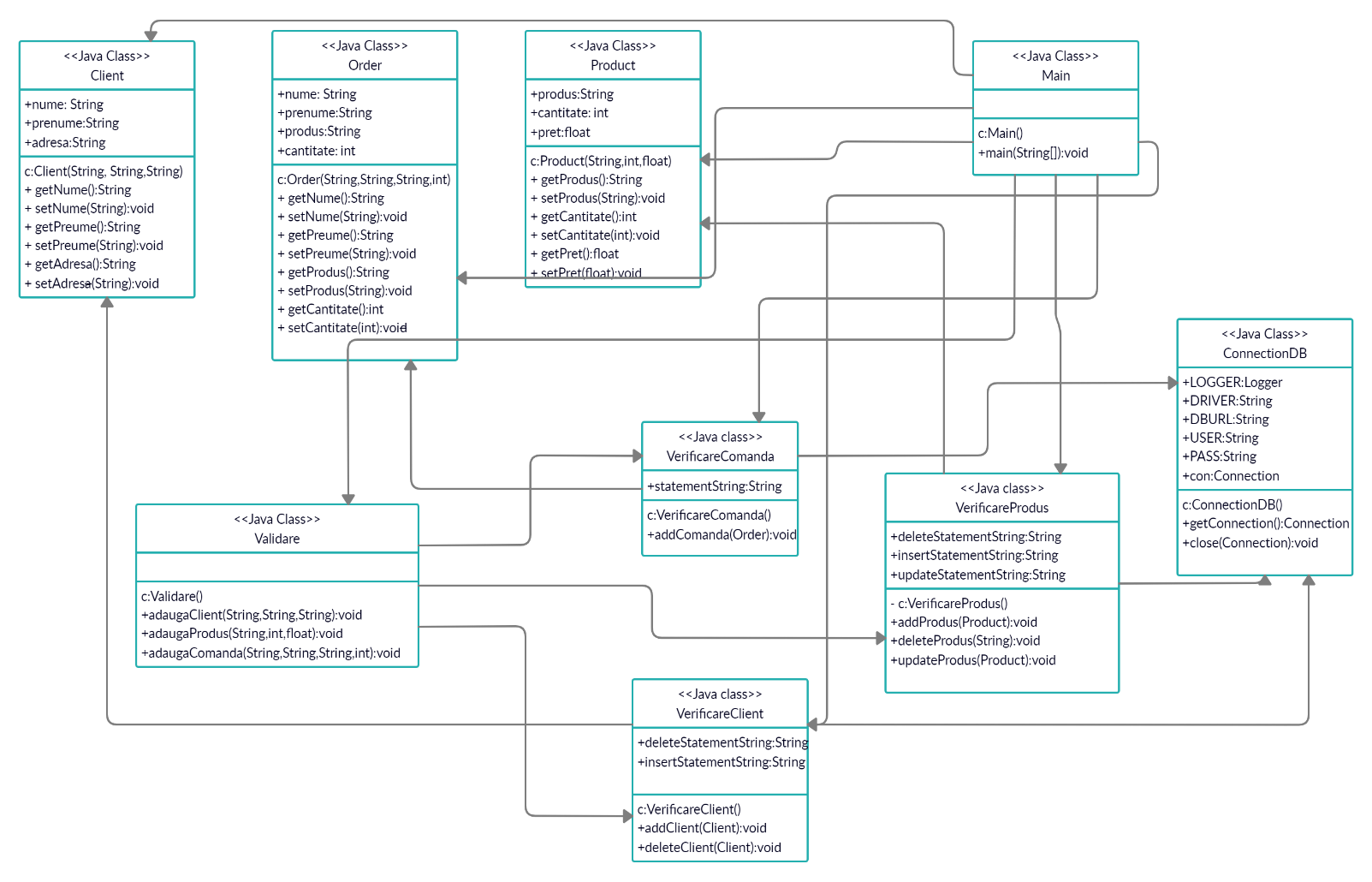
-clasa Product: In aceasta clasa am trei variabile: produs, cantitate, pret, un constructor care initializeaza aceste campuri si metode de get si set pentru acestea.

-clasa Order: In aceasta clasa am patru variabile: nume, prenume, produs,cantitate, un constructor care initializeaza aceste campuri si metode de get si set pentru acestea.

Pachetul Presentation:

-clasa Main: In aceasta clasa am declarat variabile pentru a putea prelua datele din fisier, un arrayList pentru fiecare clasa in care pastrez clientii, produsele respectiv comenzile, documentele PDF in care generez tabele pentru fiecare caz preluat din fisier. Intr-un while in care testez daca mai sunt linii in fisier, citesc pe rand fiecare linie si verific pe ce caz ma aflu. Aici apelez si metodele din celelalte clase pentru a putea ajunge la rezultatul dorit.

DIAGRAME UML

Diagramele sunt grafuri care prezintă simboluri ale elementelor de modelare (model element) aranjate astfel încât să ilustreze o anumită parte sau un anumit aspect al sistemului.Diagramele **UML** fac mult mai evidenta functionalitatea acestei aplicatii. Diagrama de clase foloseste clase si o interfata si mai multe relatii intre acestea. Ea descrie din punct de vedere structural sistemul si ajuta la intelegerea implementarii acestuia.

**4.IMPLEMENTARE**

Aplicatia citeste initial datele din fisierul text dat ca argument, iar apoi verifica pe ce caz ne aflam.

Daca avem Insert client, atunci adaugam in baza de date in tabelul client, clientul respectiv cu numele,prenumele si adresa preluate din fisier. Daca avem Delete client atunci stergem din baza de date acel client. Daca avem Report client atunci se creeaza un document PDF cu ajutorul librariilor din iText, in care se adauga cate un tabel pentru fiecare Report client intalnit (adica in fiecare tabel avem pasii executati pana la urmatorul Report Client intalnit). Daca avem Insert product inseram un produs in baza de date in tabelul Product. Daca avem Delete Product atunci stergem din baza de date respectivul produs, iar daca avem Report Product se creeaza un document PDF ca in cazul anterior.

Daca avem Order atunci avem cateva lucruri de verificat si calculat. Prima data se verifica daca acea comanda s-a putut efectua, daca avem in stoc atatea produse cate cere clientul respectiv. Daca s-a putut efectua, doar atunci se adauga in baza de date in tabelul Order comanda respectiva si se genereaza o chitanta cu acea comanda. In acelasi timp, pe baza cantitatii comandate de client, se calculeaza pretul corespunzator si se face update la cantitate pentru acel Produs. Daca acea comanda nu s-a putut efectua, atunci se genereaza un pdf in care se “anunta” ca acea comanda nu a avut succes. Daca avem Report Order atunci se genereaza un document PDF cu comenzile care s-au putut efectua si se afiseaza numele clientilor si comanda facuta.

**5.REZULTATE**

Setul de rezultate obtinut in urma testarii pune in evidenta faptul ca aplicatia este implementata corect, conform cerintei, iar algoritmul functioneaza conform asteptarilor.

Aplicatia ne ajuta sa ne dam seama cum functioneaza o baza de date, cum stocheaza anumite informatii, cum le modifica si cum le sterge in functie de comenzile pe care utilizatorul le da.

**6.CONCLUZII**

Din aceasta tema am invatat cum sa creez PDF-uri din limbajul Java, cum sa ma conectez la o baza de date si cum sa fac modificari pe acea baza de date in functie de anumite preferinte si decizii luate.

**7.BIBLIOGRAFIE**

* <http://stackoverflow.com/>
* <http://users.utcluj.ro/~igiosan/teaching_poo.html>
* <http://coned.utcluj.ro/~salomie/PT_Lic/4_Lab/Assignment_3/Assignment_3_Indications.pdf>
* <https://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html>
* <https://www.baeldung.com/java-pdf-creation>