MINISTERUL EDUCAŢIEI ŞI CERCETĂRII ŞTIINŢIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

ASSIGNMENT 3

ORDERS MANAGEMENT

NUME STUDENT: Boar Daniel-Ioan
GRUPA: 30223

Cuprins

1. Obiectivul temei

Obiectivul aceste teme este proiectarea si implementarea unui program ce va fi folosit in procesarea unor comenzi. Comenzile vor veni de la anumiti clienti, care doresc sa cumpere anumite cantitati de produse din magazin. Produsele se gasesc in stocul magazinului intr-o cantitate suficienta sau insuficienta, astfel incat un client nu poate comanda mai multe bucati dintr-un produs daca nu sunt destule. Comenzile se definesc printr-un anumit client (id) care le pleaseaza, produsul ales (id) si cantitatea dorita. Clientii au urmatoarele atribute: id si nume.

Aplicatia se foloseste de o baza de date MySQL in care am stochez fiecare client, produs si comanda.

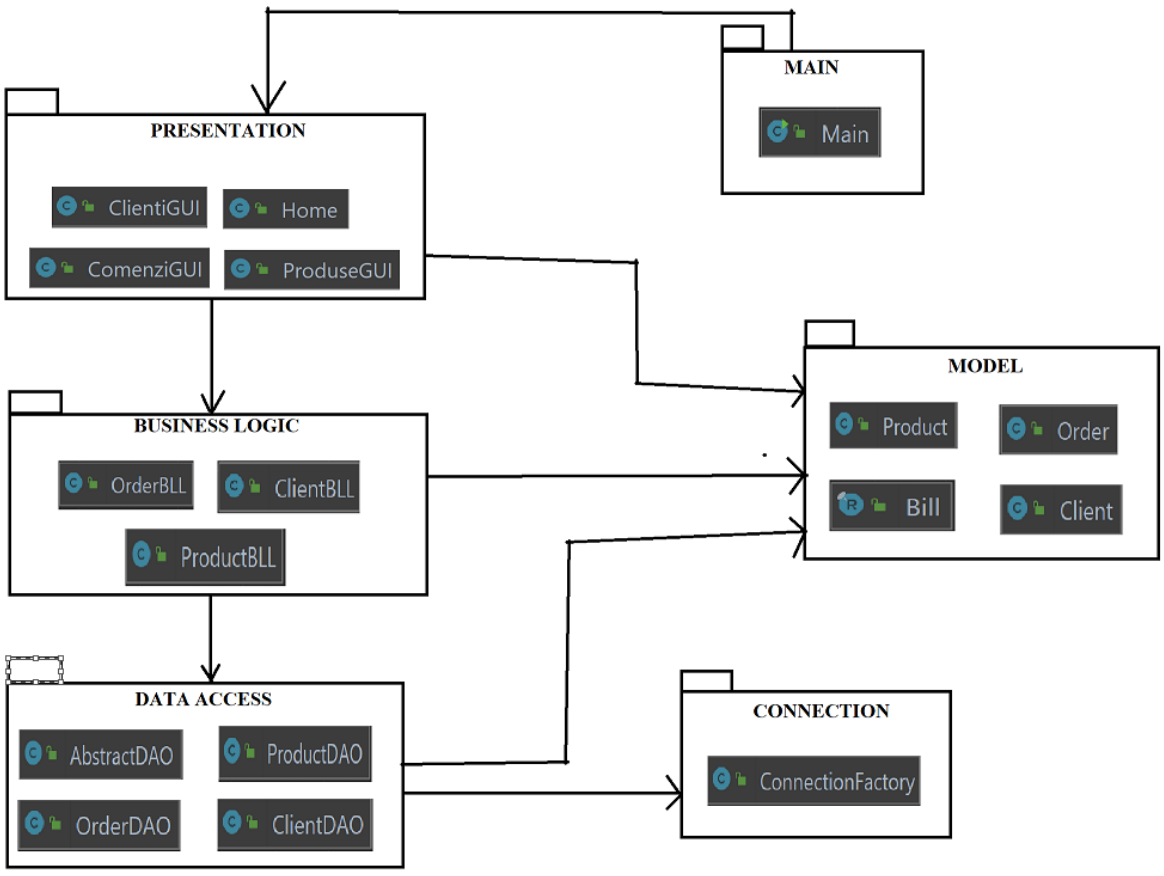
2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Pentru realizarea aplicatiei am folosit 17 clase. Aceste clase se impart in mai multe pachete care ne ajuta la intelegerea problemei: BusinessLogic, Connection, DataAcces, Model, Presentation si Main. In pachetul **BusinessLogic** avem logica aplicatiei, si alte apelari ale metodelor; **Connection** avem conexiunea dintre programul nostru Java si baza de date MySQL, **DataAcces** avem clasele DAO in care avem implementate operatiile din tabele; **Model** avem o clasa pentru fiecare tabel din baza de date in care avem atributele lor; **Presentation** este interfata user friendly.

Am folosit tehnica reflection pentru a putea extrage datele dintr-un tabel anume in functie de ce coloane are. In clasa AbstractDAO am folosit aceasta tehnica si mi-a fost de folos in a afisa in timp real continutul bazei de date (in functie de tabel) in interfata grafica.

3. Proiectare

DIAGRAMA UML :



4. Implementare

Dupa cum se poate vedea si in diagrama UML a claselor, aplicatia dispune de 17 clae. Aceste clase impreuna cu metodele si functionalitatile lor vor fi descrise mai jos.

a) MODEL

- **Clasa Client**

In aceasta clasa sunt atributele ce se regasesc si in tabela Client: id si numele clientului. Avem metode de getId si de getNume care ne returneaza id-ul, respectiv numele clientului. Tot aici mai avem si un constructor ce ne initializeaza aceste atribute.

```
public int getId()
public String getNume()
public Client(int id, String nume)
```

- **Clasa Product**

Aceasta clasa are atributele tabelei de produse: id, nume, cantitate si pretul produsului. In aceasta clasa am implementat un constructor care initializeaza atributele, si 4 metode care imi returneaza fiecare fie id-ul, numele, cantitatea sau pretul unui produs.

```
public Product(int id, String nume, int cantintate, int pret)
public int getId()
public String getNume()
public int getCantintate()
public int getPret()
```

- **Clasa Order**

In aceasta clasa avem atributele unei comenzi din baza de date: id-ul clientului, id-ul produsului, cantitatea si pretul total al comenzii. Ca mai sus, avem un constructor de initializare si metode ce ne returneaza id-ul clientului si a produsului, cantitatea si pretul comenzii.

- **Clasa BILL**

Clasa Bill este o clasa de tipul record care face legatura cu tabela „Log” din baza de date si este o clasa care ne genereza comanda pentru fiecare client in parte si ne retine numele clientului, numele produsului, cantitatea, pretul comenzii si data si ora la care s-a plasat comanda.

b) DataAccess

- **Clasa ClientDAO**

Aceasta clasa in mare parte face operatiile cu tabela de clienti. Avem o metoda „getclientbyID” care ne returneaza numele unui client si primeste ca input id-ul clientului

```
public static String getclientbyID(int id) throws SQLException
```

Mai departe avem o metode care ne insereaza, sterge si editeaza un client in baza noastra de date.

- **Clasa ProductDAO**

In acesta clasa avem la fel operatii cu tabela produs, cum ar fi: sa-mi returneze un produs cu toate atributele sale (metoda getproductbyId), metode de inserare, stergere si editare a unui produs in tabel din MySQL. Pe langa aceste metode mai avem 3 metode care ne returneaza numele, cantitatea si pretul unui produs din tabela produs din baza de date. Ultima metoda este o metoda care ne seteaza cantitatea unui produs si o folosesc atunci cand decrementez numarul de produse cand fac o comanda.

- **Clasa OrderDAO**

Aceasta clasa are o singura metoda care implementeaza operatia de inserare a unei comenzi in baza noastra de date, comanda care primeste id-ul de client, id-ul produsului, cantitatea unui produs si pretul total calculat.

Tot aici creez un obiect de tipul „BILL" si adaug comanda in tabela „Log" din baza de date cu atributele sale (nume_client, nume_produs, cantitate, pret, data).

- **Clasa AbstractDAO**

In aceasta clasa ne folosim de o metoda reflection care ne extrage datele dintr-un tabel anume din baza noastra de date primit ca parametru de intrare si ni le insereaza intr-un TableModel care va fi returnat de aceasta metoda.

```
public static DefaultTableModel getTableModel(String table)
```

c) BusinessLogic

Din acest pachet fac parte clasele ClientBLL, ProductBLL si OrderBLL care nu fac altceva decat sa apeleze metodele din ClientDAO, ProductDAO si OrderDAO, unele din ele verificand in plus daca exista clientul sau produsul cu id-ul respectiv.

d) Presentation

In acest pachet avem clasele care implementeaza interfetele grafice si fiecare in parte cu butoanele ei, care fac anumite operatii : insert, delete, update, back sau comanda (butonul acesta imi plaseaza comanda si imi adauga comanda in baza de date). In interiorul interfetelor de clienti si de produse avem afisata tabela client sau tabela product in functie de interfata la care suntem. Aceasta tabela este sub forma de TableModel si se actualizeaza mereu cand facem o operatie in tabela.

e) Connection

In pachetul Connection avem o singura clasa „ConnectionFactory" care face conexiunea dintre baza noastra de date si programul nostru Java.

f) Main

La fel, in pachetul Main avem doar clasa Main unde pornesc aplicatia noastra utilizand interfata „Home".

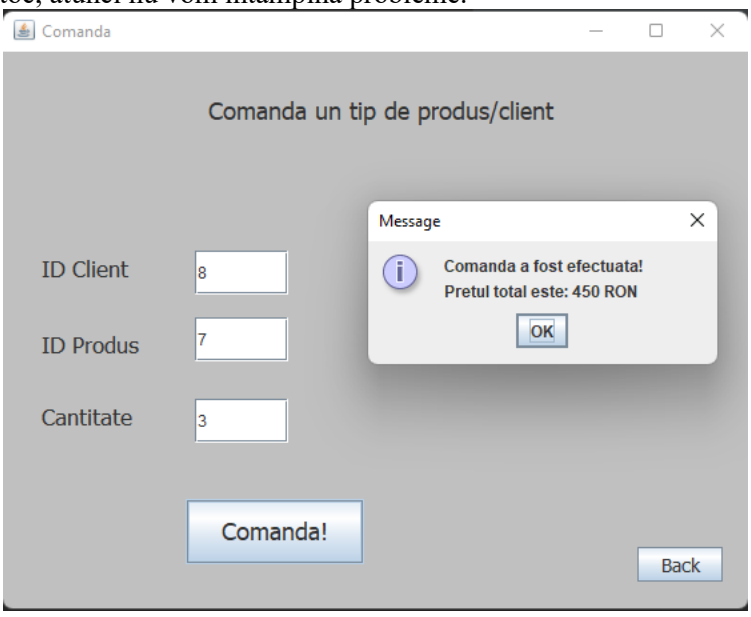
5. Rezultate

In momentul in care rulam aplicatia, ea arata astfel:

Dupa cum se poate vedea, aplicatia asteapta de la user sa se decida ce doreste sa faca: daca apasa pe butonul „Clienti” el poate intra in interfata de clienti, poate vedea tabela de clienti si poate adauga, sterge sau sa editeze un client. Daca cumva user-ul decide sa apese pe butonul „Produse” el poate intra in interfata de produse, vede tabela de produse din baza de date si poate adauga, sterge sau sa editeze un produs.

Intr-un final ajungem la punctul de plecare al acestei teme, plasarea unei comenzi. Daca user-ul apasa butonul „Plaseaza o comanda”, se va deshide interfata de comenzi. Aici, clientul poate sa introduca in textFielduri id-ul clientului, id-ul produsului si cantitatea dorita pentru a fi cumparata. Dupa ce se decide ce doreste sa cumpere, user-ul apasa pe butonul „Comanda” si se va afisa un pop-up cu mesajul „Comanda a fost efectuata!” pretul total al comenzii. Daca cumva nu exista destule produse in stoc se va afisa un mesaj „Stoc depasit!”.

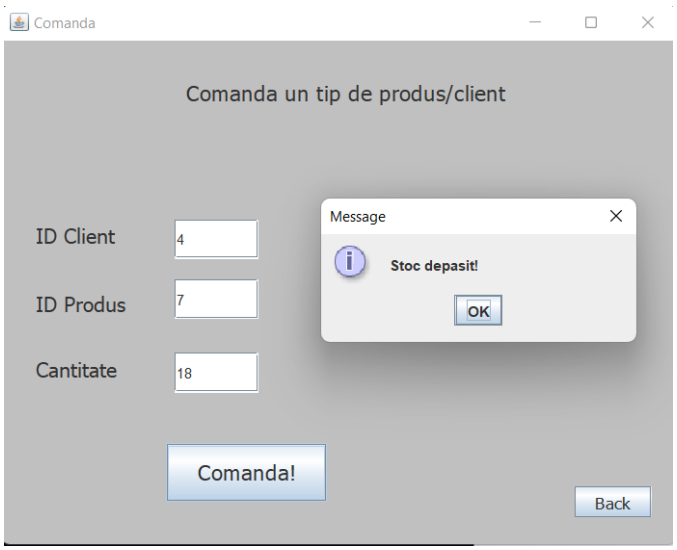
Haideti sa testam programul nostru pentru un exemplu dat de noi. Sa presupunem ca Dani vrea sa cumpere 3 produse de tipul „Tricou Nike”. Vom introduce id-ul lui Dani, id-ul produsului „Tricou Nike” si cantitatea de 3. Stim ca in tabela, pretul unui tricou este de 150 RON si sunt peste 3 tricouri in stoc, atunci nu vom intampina probleme.



Dupa ce am facut comanda, in tabela de produse, numarul de tricouri ar trebui decrementat cu 3, iar in tabela de order si log ar trebui sa apara comanda facuta.

	nume_cl... ▲ 1	nume_produs	cantitate	pret	data
1	Dani	Tricou Nike	3	450	Thu May 25 19:20:20 EEST 2023

Haideti sa dam si un contra-exemplul. Dorim ca alt client sa comande 18 de tricouri, stiind ca in stoc mai sunt 17. Ar trebui sa ne afiseze mesajul „Stoc depasit!” si sa nu putem sa facem comanda



6. Concluzii

Consider ca si aceasta tema a adus un bonus in dezvoltarea modului de lucru in Java si mai ales mi-am reamintit cum sa folosesc baza de date MySQL si sa o folosesc impreuna cu Java. Nu au fost chestii grele de implementat in aceasta tema, singurul lucru a fost ca au fost multe lucruri mici de implementat.

Ca dezvoltari ulterioare as putea adauga mai multe produse, fiecare cu imaginea sa, si sa existe in aplicatie si un admin, dar si un user care sa se poata conecta la interfata cu un user si o parola.

7. Bibliografie

[1] https://stackoverflow.com/questions/
[2] https://www.oracle.com/technical-resources/articles/java/javareflection.html
[3] https://www.baeldung.com/javadoc