Order Management

Assigment 3



**Profesor Coordonator:**  **Student:**

Dr.Pop Cristina Bianca Vaduva Calin Liviu

**Grupa**: 30228 **An:** 2

Cuprins

**1. Obiectivul Temei** ...............................................................................................

**2. Analiza Problemei** .............................................................................................

2.1 Diagrama Use-Case .......................................................................................

2.2 Cazuri de utilizare .........................................................................................

**3. Proiectare** ..........................................................................................................

3.1 Diagrama UML ...............................................................................................

3.2 Cazuri de utilizare ..........................................................................................

**4. Implementare** ....................................................................................................

4.1 Clase ..............................................................................................................

4.2 Interfata Grafica ............................................................................................

**5. Rezultate** ............................................................................................................

**6. Concluzii** .............................................................................................................

**7 Documentatie** ......................................................................................................

1.Obiectivul temei

Scopul aceste teme este atat proiectarea cat si implementarea unei aplicatii de tipul Odrder Management pentru procesarea comenziilor clientilor unui magazin sau depozit . Aplicatia trebuie sa foloseasca un mediu virual de stocare a clientiilor, comenziilor cat is a produselor intr-o baza de date. Totodata aceasta tema trebuie sa implementeze o interfata grafica care sa faca mai usoara gestionarea comenziilor .

Obiectivele secundare a acestei teme sunt :

* Analiza problemei ( va fi prezentata in capitolul 2 ) -- care cere prezentarea cerintelor

functionale si cerintelor de utilizare.

* Proiectare ( va fi prezentata in capitolul 3 ) – contine proiectarea orientata pe obiect a aplicatiei, diagramele UML, de clase, de obiecte, interfetele si algoritmi folositi .
* Implementare ( va fi prezentata in capitolul 4 ) – contine descrierea claselor , metodelor folosite, si descrierea implementarii interfetei grafice.
* Rezultate ( prezentata in capitolul 5 )

2. Analiza Problemei

Aplicatia ar trebuii sa ofere posibilitatea utilizatoriilor de a insera, sterge si modifica datele dintr-o baza de date, sa tina evidenta clientiilor, produselor si comenziilor realizate de catre acestia, dat totodata sa si poata genera o factura a comenzii plasate de catre un client.

**Aplicatia trebuie sa permita utilizatorului :**

- Sa permita utilizatorului sa vizualizeze tablourile populate de date prin una sau mai multe fereste:

- Fereastra pentru Client permite : - introducerea a noi clienti in baza de date

- realizarea operatiei de update a unui client

- stergerea unui client din baza de date

- vizualizarea clientiilor prin intermediul unui tabel

- Fereasta pentru Produse permite : - introducerea unui nou produs in baza de date

- realizarea operatiei de actualizare a unui produs

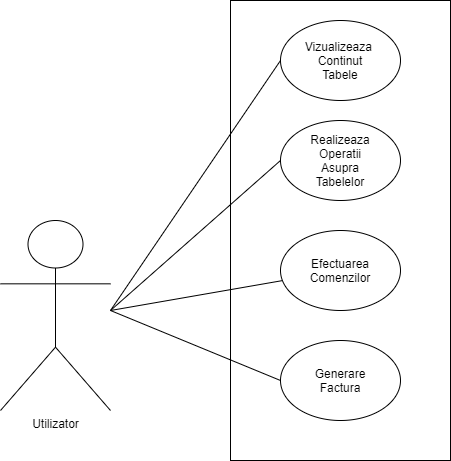
- stergerea unui produs dupa id

- vizualizarea tuturor clientilor prin intermediul unui tabel.

- Fereastra pentru Comenzi(Orders) permite:

- introducerea unei comenzi in baza de date

- listarea tuturor comenziilor din baza de date prin intermediul uni tabel.

**Diagrama Use-Case a aplicatiei**:

**Descriere diagrama Use-Case:**

Actorul principal este userul.

**Cele 4 cazuri de utilizare a aplicatiei :**

1. Utilizatorul vizualizeaza continutul tabelelor Client , Products , Orders din baza de date.

2. Utilizotorul realizeaza operatii asupra continutului tabelelor, operatii de inserare , stergere, sau modificare a datelor.

3. Utilizatorul creeaza o comanda noua cu id-ul unui client si id-ul unui produs existent si prin introducerea unei cantitati dorite pentru comanda respectiva .

4. In urma creeari unei comenzi se genereaza o factura a comenzii respective cu numele clientului , numele produsului cat si cantitatea cumparata.

**Descrierea cazurilor de utilizare :**

**Scenariul de succes :**

1) Se porneste aplicatia si se vizualizeaza continutul tabelei prin apasarea butonului “ListAll” din fiecare tabel ( Client, Product, Orders) din cele 3 interfete grafice.

2) Se introduce un Client nou prin popularea casutelor text din interfata grafica corespunzatoare acestuia iar apoi sa apasa pe butonul “INSERT” pentru crearea unui nou client

in baza de date .

3) Se apasa pe butonul ListAll pentru actualizarea in tabelul din interfata grafica a inserarii unui nou client facuta la punctul 2)

4) Daca se mai doreste un nou client sau inserarea unui produs se reia de la pasul 2)

5) Pentru creearea unei comenzi se populeaza casutele text din interfata Orders cu id-ul a Clientului si cu id-ul produsului cat si cantitatea pe care o doresti .

6) Se apasa butonul “ORDER” prin care se va face o noua comanda, dar in cazul in care cantitatea dorita este mai mare decat cantitatea in stock atunci comanda se va anula si nu se va introduce in baza de date nici o comanda si nici nu se va creea facutura.

7) Se apasa butonul “LIST” pentru afisarea in tabel a noii comenzi in cazul in care a fost valida.

**Scenariu alternativ:**

Utilizatorul poate sa listeze apoi sa stearga un Client sau un Produs din baza de date daca in locul butonului de “INSERT” este apasat butonul de “DELETE” doar dupa introducerea in casuta text “ID=” id-ul clientului sau produsului pe care acesta doreste sa il stearga. Utilizatorul trebuie sa aiba atentie deoarece un client sau produs o data sters nu se mai poate recupera.

In cazul in care ulilizatorul introduce un id pentru oricare tabela si acesta exista deja in baza de date, inserarea nu va avea loc iar aplicatia va functiona ca si cum nu a fost inserat nimic.

In cazul in care se face o comanda si cantitatea selectata pe comanda este mai mare decat cantitatea in stoc a produsului selectat , atunci comanda nu va fi luata in considerare si factura nu se va genera, dar aplicatia va continua fara sa se intrerupa .

Alte erori se pot semnala in cazul problemelor legate de legatura cu baza de date sau a tebelelor declarate in baza de date.

3. Proiectare

Aplicatia este realizata folosind modelul arhitectural stratificat, pe Layere. Acesta contine pachetele : - **model**, contine clasele care modeleaza datele aplicatiei.

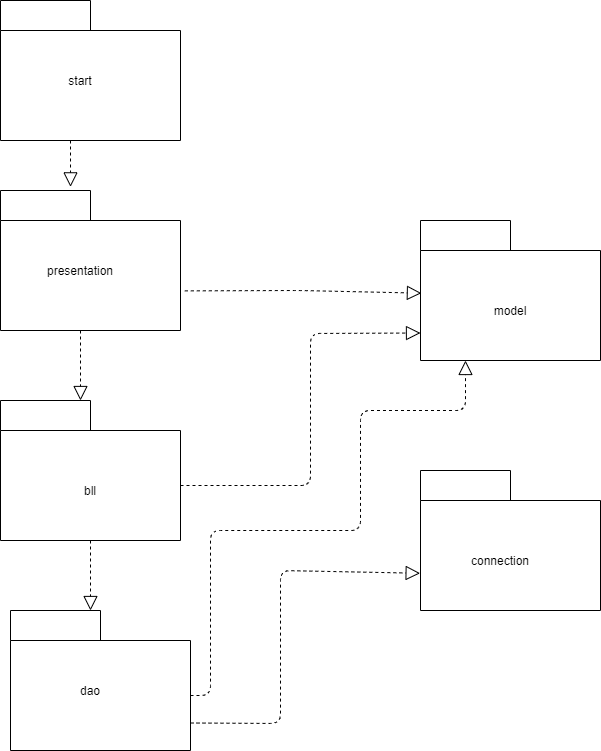
- **bll** (business logic layer), cuprinde clasele care implementeaza functionalitatea aplicatiei

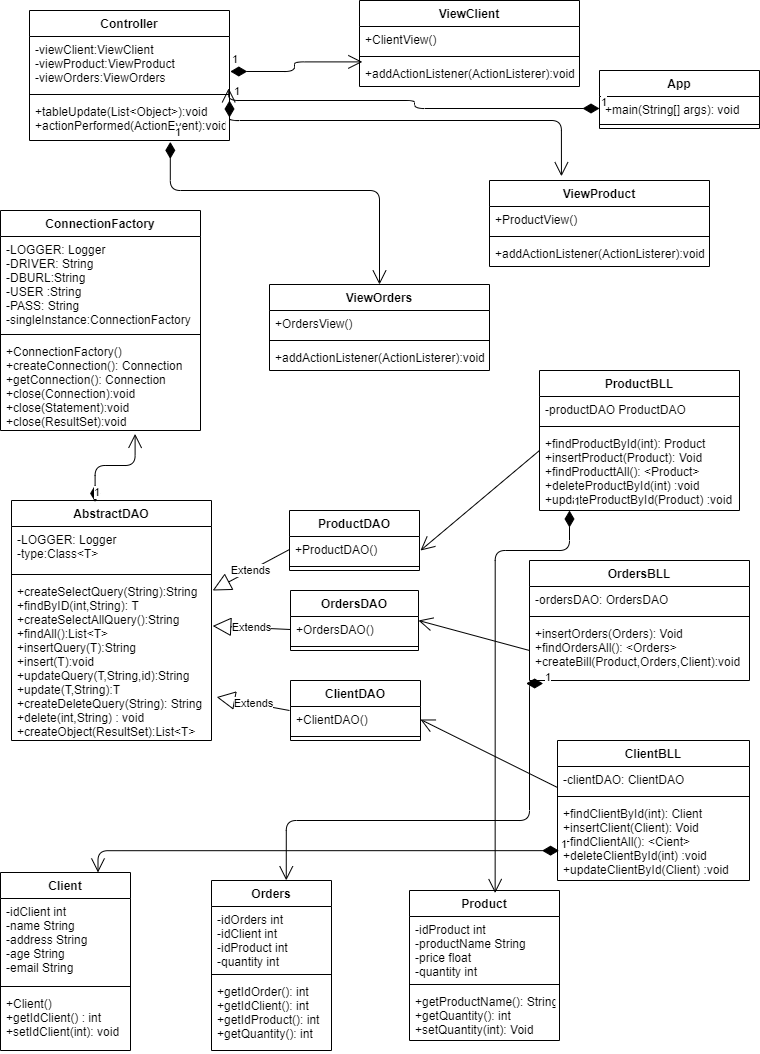
- **dao** (data acces object), contine clasele cu care implementam operatiile pe datele din baza de date.

-**presentation**, contine clasele care implementeaza interfata grafica .

-**connection**, contine clasa care face conexiunea aplicatiei la baza de date.

**Diagramele UML a aplicatiei :**

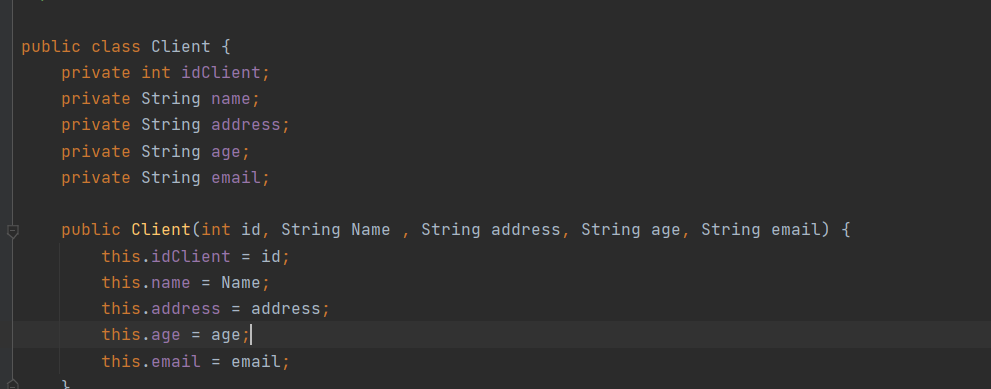
a) **Diagrama de pachete**:

b) **Diagrama de clase:**

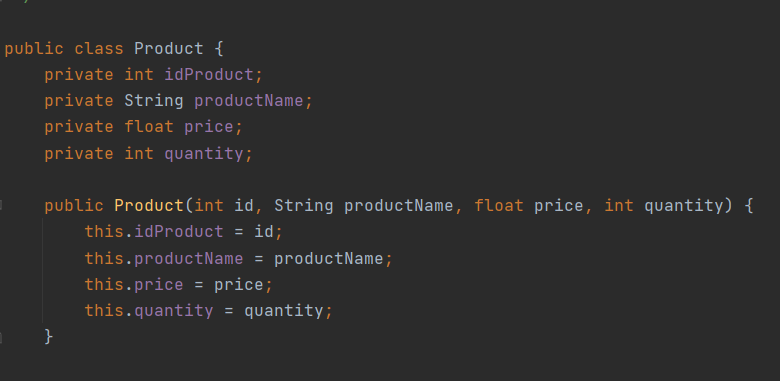
4. Implementare

**4.1 Clase :**

a) Clasa **Client**, aceasta defineste tabelul clientilor, contiinand toate atributele acestuia: idClient, name, address, email, age cat si settere si gettere pentru fiecare atribut in parte.

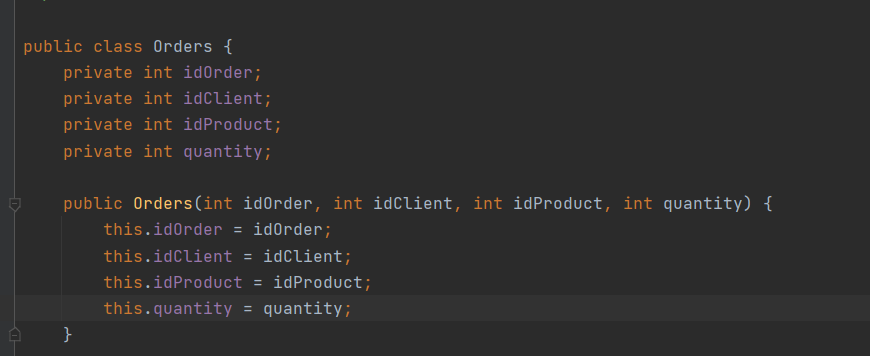


b) **Clasa Product**, aceasta clasa defineste tabelul produselor, atributele acesteia sunt :

idProduct , productName, price, quantity cat si setterele si getterele pentru a se putea face operatii cu acestea. Este atasata mai jos o imagine cu codul propriu zis a clasei prouct cu toate atributele acesteia.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

c) **Clasa Orders** : este o clasa de model ce contine atributele tabelei Orders din baza de date, acestea sunt idOrders, idClient, idProduct, quantity si contine si gettere si settere ale acestora pentru efectuarea operatiilor . Este atatasa mai jos o poza cu implementarea acestei clase



. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

d)Clasa **AbstracDAO** : este o clasa abstracta si generica, fiind super clasa lui **ClientDao, ProductDAO** cat si a lui **OrdersDAO,** clasa care structureaza operatiile cu baza de date pentru fiecare tabela.

Contine urmatoarele metode principale : -- *findById(),* care acceseaza query-ul trimis de catre metoda createSelectQuery tot din aceasta clasa, gaseste clientul, produsul sau comanda in functie de un id dat ca si parametru

-- *findALL(),* este metoda acceseaza query-ul trimis de metoda createSelectALLQuery din aceasi clasa , si reutrneaza o lista de obiecte de tipul obiectului care apelat metoda, astfel in tabel se va afisa totalitatea Clientiilor sau a Produselor sau a Comenziilor

-- *insert(T t),* este metoda prin intermediul careia inseram elemente in baza de date , aceasta primeste ca parametru un obiect de tip T si nu returneaza nimic. Metoda acceseaza un string de query dintr-o metoda din interiorul aceleasi clase si insereaza o noua line in interiorul tabelului.

*-- update(T t, String[] fields , String id )* este metoda de actualizare a unui element din baza de date. Are ca parametrii obiectul T, sirul de stringuri numit fields si id-ul cautat ca sa modificam elementul de la id inserat.

-- *delete( int id, String field)* este metoda de stergere a unui element din tabel. Aceasta contine parametrii id care este id-ul elementului care va fi sters si field care este un string de recunoastere in ce tabel va fi sters ( Client , Orders, Products) .

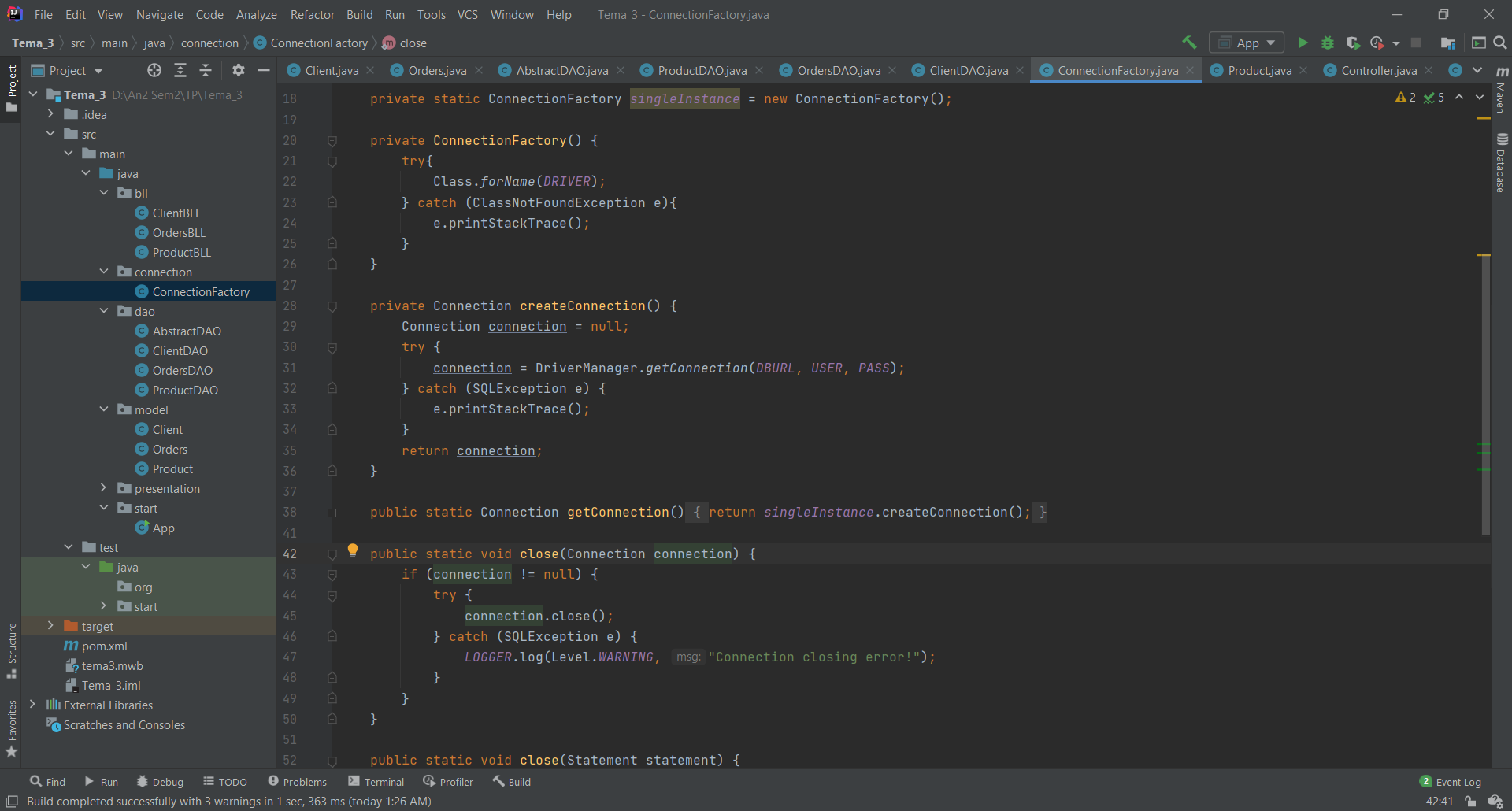
-- *createObjects ( ResultSet resultSet )* este metoda care face posibila folosirea tehnici de reflection prin intermediul careia facem operatii pe baza de date.

e) Clasa **ClientDAO** mosteneste clasa AbstractDAO si nu are metode proprii deoarece toate operatiile sunt facute in clasa parinte si ea nu mai are nevoie de alte particularitati diferite de restu metodelor

f) Clasa **ProductDAO** mosteneste clasa AbstractDAO si nu are metode decat cele mostenite de la clasa parinte .

g) Clasa **OrdersDAO** mosteneste clasa AbstractDAO si la fel ca si clasele ProductDAO si ClientDAO nu are alte metode decat cele mostenite de la clasa parinte .

h)Clasa **ConnectionFactory** este clasa principala ce face accesul la baza de date posibil. Aceasta face conexiunea la baza de date prin intermediul metodei ConectionFactory , CreateConnection , si GetConnection . In poza de mai jos sunt afisate codurile propiu zise ale acestor metode



. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

i) Clasa **ClientBLL** face parte din pachetul bll care contine clasele de business logic si prin intermediul careia se fac operatiile pe tabele. Aceste functionalitati pe care le au aceste clase sunt implementate folosind clasele din pachetul dao.

j) Clasa **ProductBLL** tot din pachetul bll contine metode precum findProductById care creaza un obiect de tipul Product si apeleaza metode din productDAO prin intermediul acestuia pentru modelarea interfetei grafice

k) Clasa **OrdersBLL** pe langa metoda de inserare si verificarea corecta a inserari in tabelul Orders contine si generarea de factura in urma inserarii in tabela orders prin metoda denumita CreateBill( Product product , Orders orders , Client client) aceasta creaza un fisier text in care pune datele comenzii tocmai inserate .

**4.2** **Interfata Grafica**

Interfata grafica se afla in pachetul presentation si contine urmatoarele clase:

a) Clasa **ViewClient** : Aceasta clasa este fereastra pentru operatiile pe tabela client. Este tot ceea ce depinde de aspectul ferestii cat si definirea butoanelor si a casutelor text pentru fiecare coloana a tabelului . Tot in aceasta clasa este implementata si tabelul de tip JTable pentru afisarea in aplicatie a tabelului Client din baza de date dar contine si metodele de get si set a casutelor text din aplicatie.

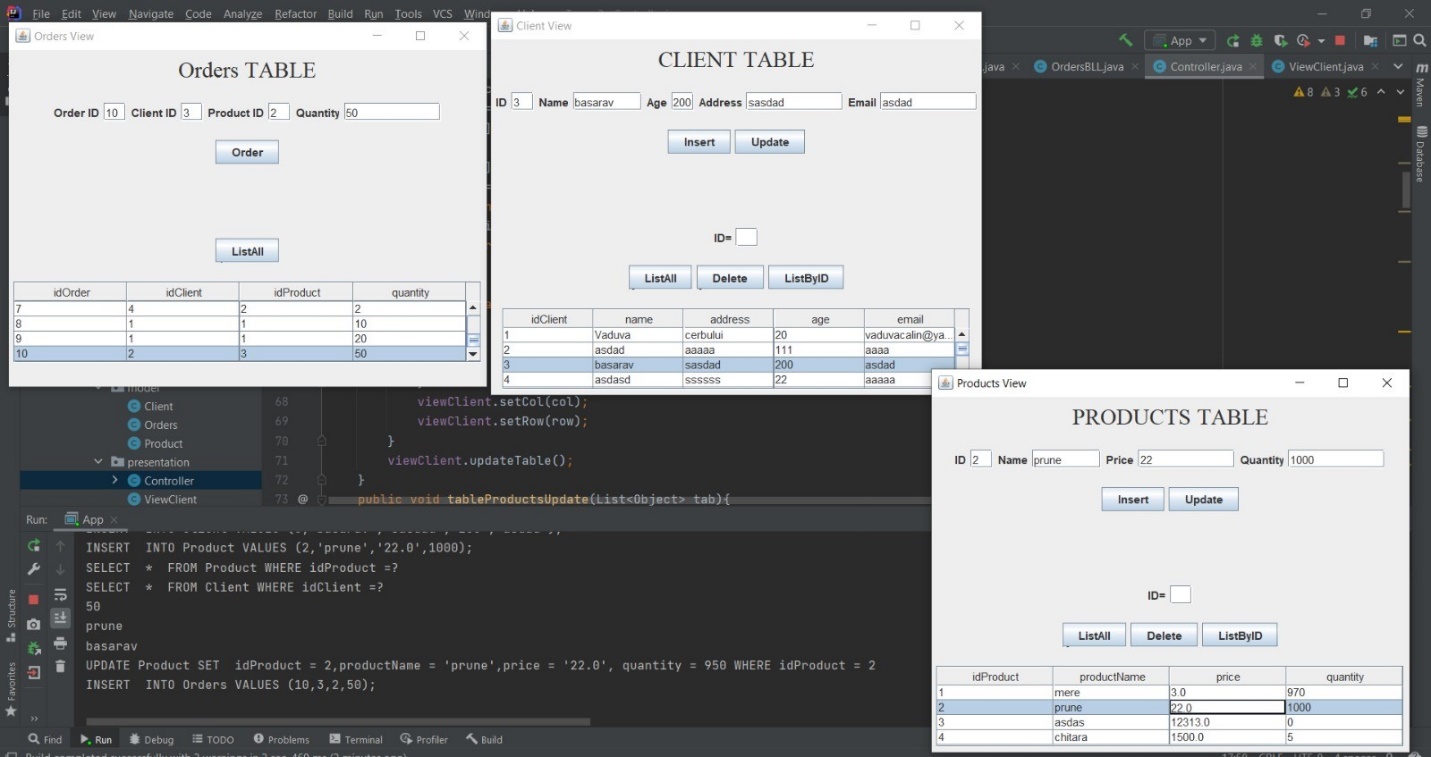
b) Clasa **ViewProduct** : Aceasta clasa este foarte asemanatoare cu cea a clientului, dar casutele text cat si butoanele sunt denumite diferit, unele implementand alte functionalitati iar tabelul care este afisat in interfata este tabelul din baza de date cu numele Product

c) Clasa **ViewOrders** se aseamana cu celelalte 2 din acelasi pachet doar ca numarul de butoane care implementeaza actiuni este mai mic. Aceasta interfata poate sa implementeze doar listarea si inserarea a anumitor Orders din si in Tabelul cu acelasi nume .

d) Clasa **Controller** din acelasi pachet cu clasele de View face conexiunea cu utilizatorul si aplicatia, dupa cum ii poarta si numele prin intermediul acestuia utilizatorul poate sa insereze in Views date si ca butoanele din interfata sa implementeze anumite comenzi precum listare , stergere, actualizare sau inserare de linii intr-un tabel selectat.

5. Rezultate

In urma implementarii claselor, pentru testarea si verificarea aplicatiei s-au introdus cate un produs mai apoi un client, pentru verificarea corectitudinii s-au si listat iar numai m-ai apoi s-a creat o comanda cu ajutorul clientului si a produsului creat iar pentru fiecare a fost afisata mai jos cate o imagine .



6. Concluzie

In concluzie pot spune ca in urma realizarii acestui proiect denumit Order Management mi-au fost imbogatite , chiar innoite cunostintele legate de programarea orientata pe obiecte in limbajul de programare java. Iar tot prin intermediul acestui proiect am invatat cum sa extrag , sa inserez sa sterg date din si intr-un tabel dintr-o baza de date, altfel spus am invatat cum sa lucrez cu baze de date prin intermediul liblajului de programare java .

7 Documentatie

-materiale de curs si laborator.

-https://docs.oracle.com/javase/tutorial/reflect/member/field.html

-https://www.baeldung.com/java-reflection

-https://www.logicbig.com/tutorials/java-swing/generate-jtable-model.html

-https://www.javatpoint.com/java-field-setaccessible-method

-https://www.javatpoint.com/example-to-connect-to-the-mysql-database