DOCUMENTAȚIE

ORDERS MANAGEMENET TEMA 3

NUME STUDENT: Filip Raul

GRUPA: 302210

**CUPRINS**

1. [Obiectivul temei 3](#_bookmark0)
2. [Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_bookmark1)
3. Proiectare 5
4. Implementare 8
5. Rezultate 14
6. Concluzii 16
7. Bibliografie 16

# Obiectivul temei

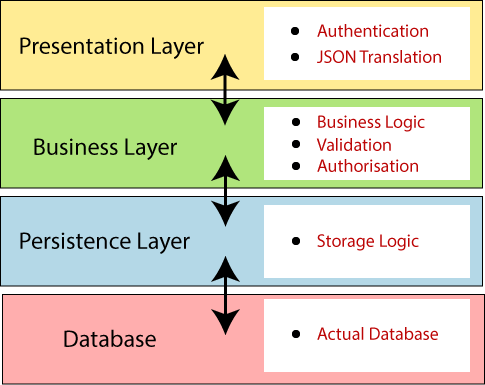
Obiectivul acestei teme este implementarea și proiectarea unei aplicatii pentru procesarea comenzilor unor produse plasate de clienti. Bazele de date relationale sunt folosite pentru a stoca datele despre clienti, produse si comenzi. Pentru realizarea obiectivului principal, trebuie să trecem printr-o serie de obiective secundare, după cum urmează:

* Analiza problemei, modelare, scenarii, cazuri de utilizare. În acest capitol se vor prezenta cerințele funcționale și cazurile de utilizare, precum și diagrama USE-CASE. (capitolul 2).
* Proiectarea. În acest capitol se va descrie diagrama UML de clase, structurile de date folosite (capitolul 3).
* Implementarea. În acest capitol se vor descrie metodele precum și atributele din fiecare clasă (capitolul 4).

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Pentru implementarea acestui proiect, s-a folosit o arhitectura de tipul Layered architecture. Acest tip de arhitectura presupune organizarea proiectului in 4 nivele:

* + PRESENTATION LAYER – reprezentat de pachetul presentation din acest proiect, contine toate clasele responsabile cu interfata grafica care intra in interactiune cu utilizatorul.
  + BUSSINES LAYER – reprezentat de pachetul bll din acest proiect, contine toate clasele responsabile cu logica aplicatiei. Acest nivel se ocupa de obiecte pentru a indeplini un posibil scenariu de utilizare cerut de catre utilizator.
  + DATABASE LAYER – reprezentat de pachetul dao si connection din acest proiect, contine clasele care definesc interogarile necesare si face conexiunea cu baza de date.
  + PERSISTENCE LAYER – contine clasele responsabile cu realizarea operatiilor tehnice, cum ar fi preluarea datelor din baza de date si prelucrarea acestora.



## BAZE DE DATE RELATIONALE

O bază de date relațională este un tip de bază de date care stochează și oferă acces la punctele de date care sunt legate între ele. [Bazele de date](https://www.oracle.com/ro/database/what-is-database/) relaționale se bazează pe modelul relațional, o modalitate intuitivă și simplă de reprezentare a datelor în tabele. Într-o bază de date relațională, fiecare rând din tabel este o înregistrare cu un cod unic numit cheie. Coloanele din tabel au atribute ale datelor și fiecare înregistrare are, de obicei, o valoare pentru fiecare atribut, facilitând stabilirea relațiilor dintre punctele de date.

O diagramă USE-CASE este o reprezentare grafică a posibilelor interacțiuni ale utilizatorului cu un sistem. O astfel de diagramă prezintă relația dintre utilizator și diversele posibilități de utilizare (reprezentate prin elipse sau cercuri). Utilizatorii sunt reprezentați prin omuleți. Aceste diagrame sunt înțelese mai ușor decât diagramele de clasă.

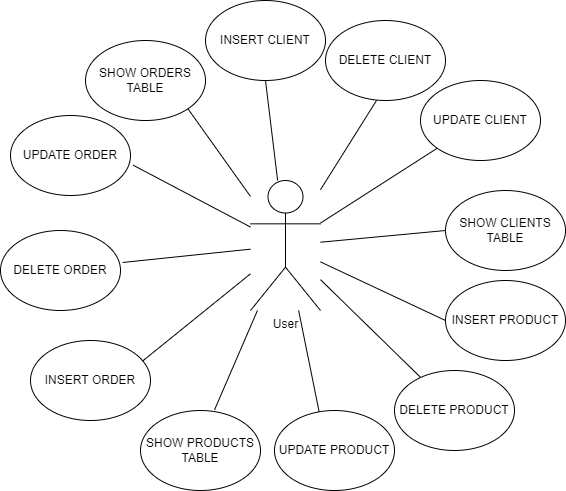
## REFLECTION

Reflection este o caracteristică a limbajului de programare Java. Permite unui program Java care se execută să examineze sau să „introspecteze” asupra lui însuși și să manipuleze proprietățile interne ale programului. De exemplu, este posibil ca o clasă Java să obțină numele tuturor membrilor săi și să le afișeze.

Capacitatea de a examina și de a manipula o clasă Java din interior poate să nu sune prea mult, dar în alte limbaje de programare această caracteristică pur și simplu nu există. De exemplu, nu există nicio

modalitate într-un program Pascal, C sau C++ de a obține informații despre funcțiile definite în acel program.

Exemplu USE-CASE:

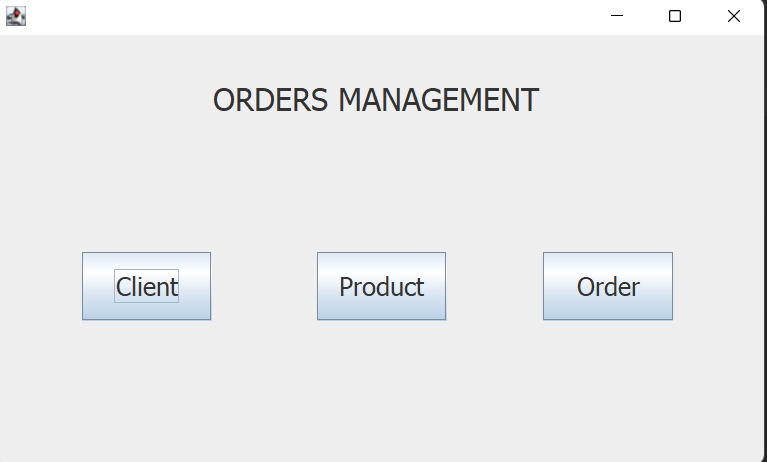


1. Utilizatorul apasa unul din cele 3 butoane aferente celor 3 tabele/ clase.
2. Utilizatorul selecteaza operatia ce doreste a fi executata: insert, delete, update, show table, back.
3. Daca utilizatorul a ales optiunea de inserare, se va deschide o noua fereastra unde trebuie introduse datele in campuri.
4. Dupa introducere se apasa butonul INSERT, si apoi butonul BACK.
5. Pentru a vedea inregistrarea ce tocmai a fost inserata, se apasa butonul SHOW TABLE.
6. Utilizatorul primește rezultatul intr-un tabel unde se pot observa si inserarile anterioare.

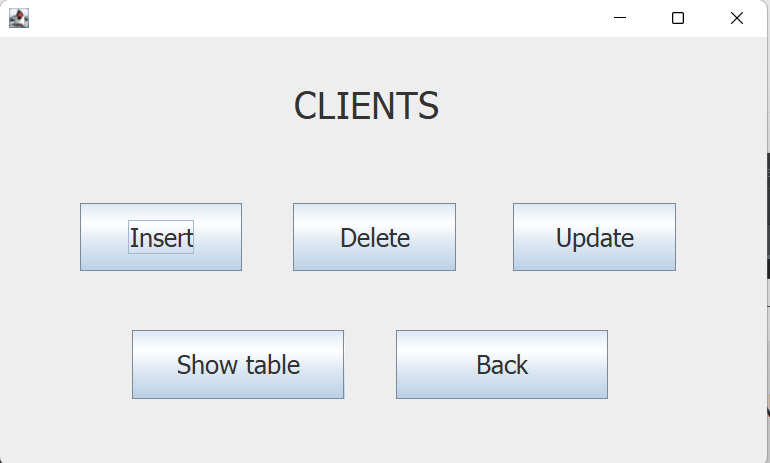
# Proiectare

Structurile de date folosite sunt: ArrayList<>.

În momentul rulării aplicației, pe ecran va apărea interfața grafică cu cele 3 butoane de CLIENT, PRODUCT si ORDER. In functie de ce doreste utilizatorul sa faca, poate sa selecteze oricare din cele 3 optiuni .

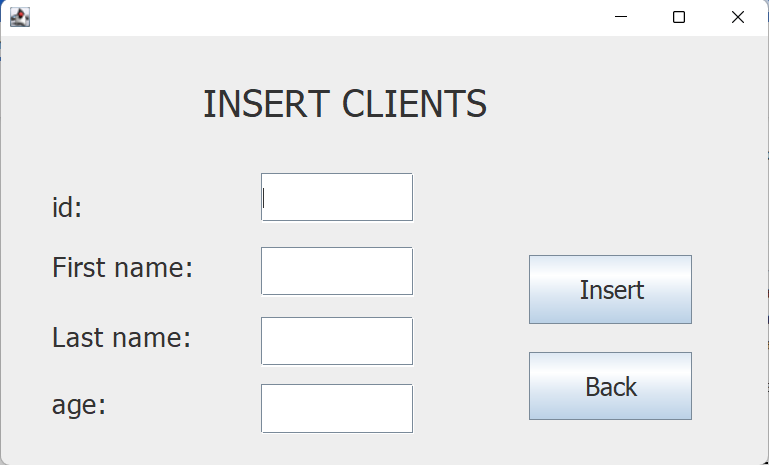


Daca se selecteaza optiunea Client, pe ecran se va deschide un alt frame.



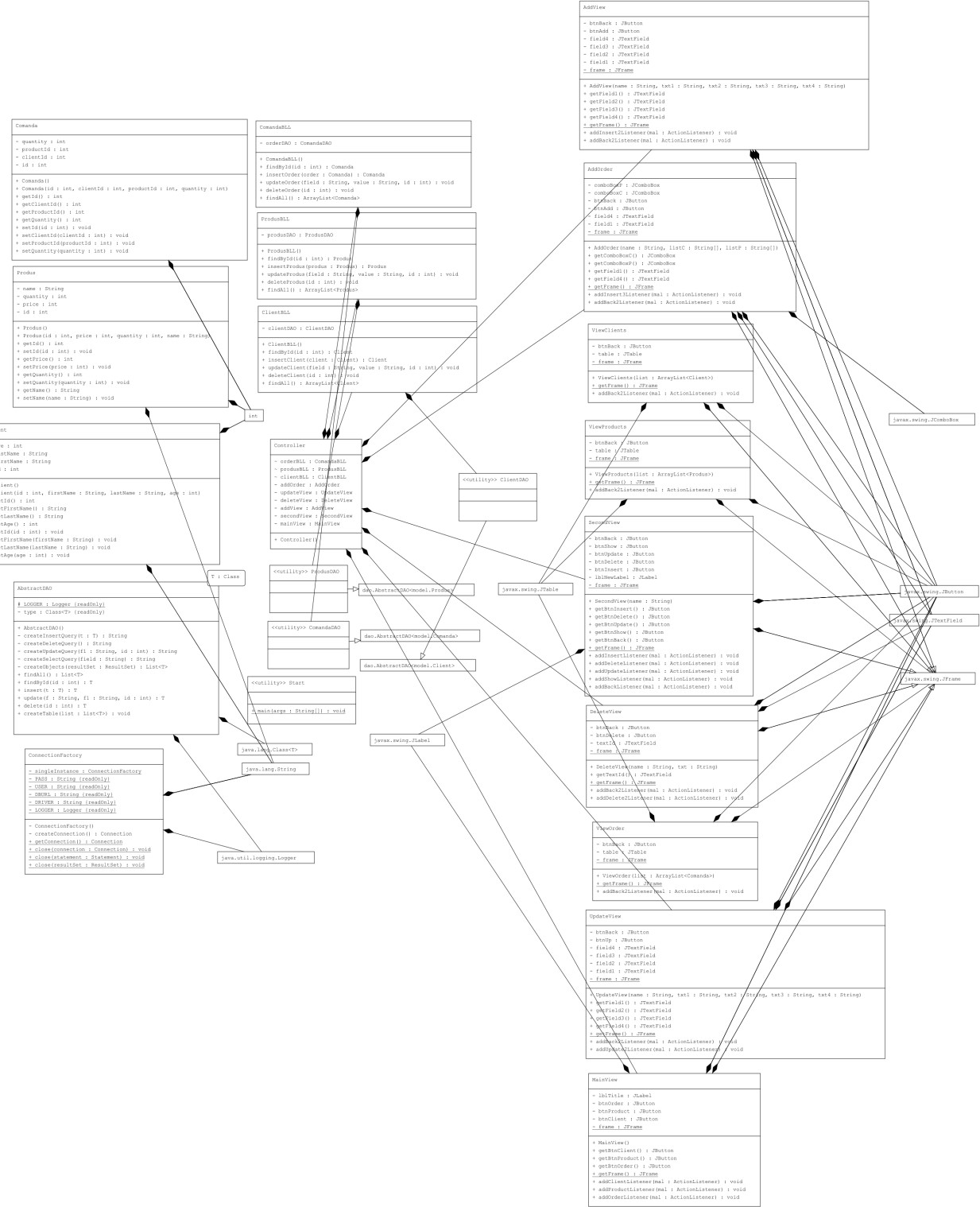
Utilizatorul poate sa insereze, stearga, editeze si sa vada continutul tabelei, dar poate sa mearga si inapoi in fereastra anterioara.

Daca se alege optiunea de INSERT, se va deschide o noua fereastra ce contine campurile ce trebuiesc inserate de catre utilizator.



Prin intermediul interfeței grafice se realizează legătura cu utilizatorul. Interfața grafică/ GUI (Graphical User Interface) este o interfață cu utilizatorul bazată pe un sistem de afișaj ce utilizează elemente grafice. Pentru a putea prezenta toate informațiile și acțiunile disponibile, un GUI oferă pictograme și indicatori vizuali, în contrast cu interfețele bazate pe text, care oferă doar nume de comenzi (care trebuie tastate) sau navigația text.

Diagrama UML este formată din 21 de clase si 2 interfete:

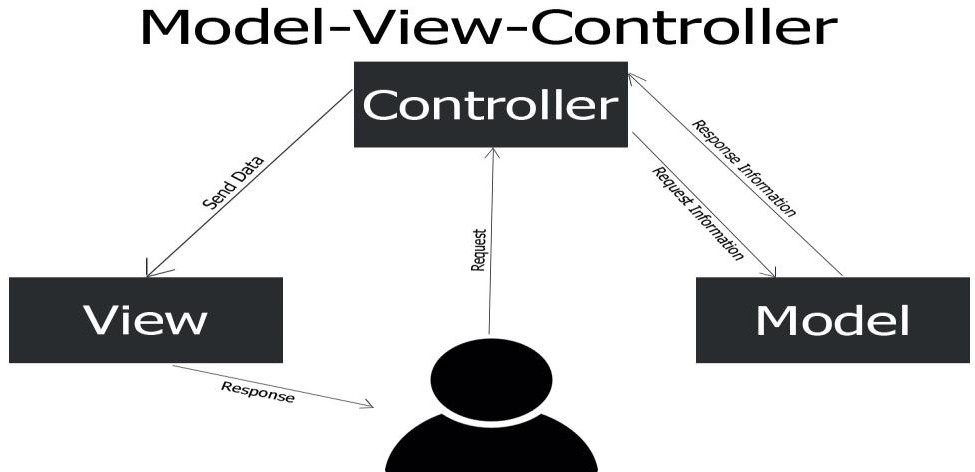


Putem observa că în diagramă se aplică regulile de vizibilitate pentru atribute și pentru metodele din clase. Fiecare modificator de accees este reprezentat printr-un simbol:

* Public (+): vizibil oriunde în proiect.
* Private (-): vizibil doar în clasa din care apaține.
* Protected (#): vizibil în clasele moștenite și în pachet.
* Package (~): vizibil în pachet.

Model-view-controller (MVC) este un model arhitectural folosit în ingineria software. Cu ajutorul controller-ului, al model-ului sau al view-ului putem manipula următoarele elemente: datele.

* Controller-ul este elementul cu care putem controla accesul la aplicația noastră. Pot fi fișiere, script- uri sau programe, în general orice tip de informație permisă de interfață.
* Model-ul manipulează operațiunile logice și de utilizare de informație trimisă dinainte de către rangul său superior) pentru a rezulta de o formă ușor de înțeles.
* View-ului îi corespunde reprezentarea grafică, sau mai bine zis, exprimarea ultimei forme a datelor: interfața grafică ce interacționează cu utilizatorul final. Rolul său este de a evidenția informația obținută până ce ea ajunge la controlor.



# Implementare

În acest proiect am implementat 22 de clase.

1. Clasa ClientBLL este clasa unde am implementat metodele care ajuta la adaugarea unui nou client, stergerea unui client deja existent, editarea unui client si afisarea tuturor clientilor din baza de date. Aceasta clasa face parte din pachetul bll.
2. Clasa ProdusBLL este clasa unde am implementat metodele care ajuta la adaugarea unui nou produs, stergerea unui produs deja existent, editarea unui produs si afisarea tuturor produselor din baza de date. Aceasta clasa face parte din pachetul bll.
3. Clasa ComandaBLL este clasa unde am implementat metodele care ajuta la adaugarea unui noi comenzi, stergerea unui comenzi deja existente, editarea unei comenzi si afisarea tuturor comenzilor din baza de date. Aceasta clasa face parte din pachetul bll.
4. Clasa Client este clasa care are atributele aferente unui client si contine gettere si settere pentru acestea. Aceasta clasa face parte din pachetul model.
5. Clasa Produs este clasa care are atributele aferente unui produs si contine gettere si settere pentru acestea. Aceasta clasa face parte din pachetul model.
6. Clasa Comanda este clasa care are atributele aferente unei comenzi si contine gettere si settere pentru acestea. Aceasta clasa face parte din pachetul model.
7. Clasa AbstractDAO este clasa unde am implementat interogarile aferente fiecarei operatii care se poate efectua pe tabele: inserare, stergere, editare, afisare, cautare. Aceasta clasa face parte din pachetul dao.
8. Clasa ClientDAO este clasa care mosteneste clasa AbstractDAO, precum si metodele acesteia. Aceasta clasa face parte din pachetul dao.
9. Clasa ProdusDAO este clasa care mosteneste clasa AbstractDAO, precum si metodele acesteia. Aceasta clasa face parte din pachetul dao.
10. Clasa ComandaDAO este clasa care mosteneste clasa AbstractDAO, precum si metodele acesteia. Aceasta clasa face parte din pachetul dao.
11. Clasa ConnectionFactory este clasa unde se face conexiunea cu baza de date. Aici avem implementate metodele pentru inchiderea conexiunii. Aceasta clasa face parte din pachetul connection.
12. Clasa Controller este cea in care am facut legatura cu clasele ClientBLL, ProdusBLL, ComandaBLL si clasele pentru view-uri. Aici am implementat si comportamentul interfetei grafice (de exemplu: ce se intampla cand se apasa un buton sau cand se introduce un client sau cand se sterge un produs). Aceasta clasa contine 8 atribute: mainView, secondView, addView, deleteView, updateView, clientBLL, produsBLL, orderBLL. De fiecare data cand se apasa un buton, se va deschide un nou frame, iar cu butonul BACK se poate reveni in frame-ul anterior. Aceasta clasa face parte din pachetul presentation.
13. Clasa MainView este clasa care reprezinta prima GUI care intra in contact cu utilizatorul. Este formata din 5 atribute ( un frame, 3 butoane, un label): frame, btnClient, btnProduct, btnOrder, lblTitle. Aceasta clasa mosteneste clasa JFrame. Tot in acesta clasa am adaugat listeneri pentru cele 3 butoane. Aceasta clasa face parte din pachetul presentation.
14. Clasa SecondView este clasa care reprezinta a doua GUI care intra in contact cu utilizatorul. Este formata din 7 atribute ( frame, btnInsert, btnDelete, btnUpdate, btnShow, btnBack, lblNewLabel ) reprezentate de un frame, 5 butoane si un label. Aceasta clasa este flexibila deoarece reprezinta interfata grafica principala cu operatiile ce se pot efectua pe cele 3 tabele: Produs, Client, Comanda. Este mostenita clasa JFrame. Aceasta clasa face parte din pachetul presentation.
15. Clasa AddView este clasa care reprezinta GUI care intra in contact cu user-ul atunci cand in secondView este apasat buctonul de INSERT. Aceasta clasa contine 7 atribute: frame, field1, field2, field3, field4, btnAdd, btnBack. Cele 4 field-uri reprezinta valorile introduse de utilizator ce urmeaza a fi adaugate in baza de date. Butonul de BACK ne duce in interfata precedenta, adica in SecondView, iar butonul de INSERT ne realizeaza actiunea de inserare in baza de date. Aceasta clasa este si ea flexibila, in sensul ca este interfata grafica pentru cele 3 clase cu cele 3 tabele pe care se realizeaza operatia de inserare. Este mostenita clasa JFrame. Aceasta clasa face parte din pachetul presentation.
16. Clasa AddOrder este clasa care reprezinta GUI care intra in contact cu user-ul atunci cand in secondView-ul comenzii este apasat butonul de INSERT. Aceasta clasa contine 7 atribute: frame, field1, field4, btnAdd, btnBack, comboBoxC, comboBoxP. Cele 2 field-uri trebuie introduse de catre utilizator, iar din cele doua comboBox-uri se selecteaza id-urile aferente produsului si clientului deja existenti in baza de date. Butonul de BACK ne duce in interfata precedenta, adica in SecondView, iar butonul de INSERT ne realizeaza actiunea de inserare in baza de date.
17. Clasa DeleteView este clasa care reprezinta GUI care intra in contact cu user-ul atunci cand in SecondView este apasat butonul de DELETE. Aceasta clasa contine 4 atribute: frame, textId, btnDelete, btnBack. Field-ul existent reprezinta id-ul corespunzator clientului/ produsului/ comenzii care se doreste sa fie stearsa. Butonul de BACK ne duce in interfata precedenta, adica in SecondView, iar butonul de DELETE ne realizeaza actiunea de stergere din baza de date. Aceasta clasa este si ea flexibila, in sensul ca este interfata grafica pentru cele 3 clase cu cele 3 tabele pe care

se realizeaza operatia de stergere. Este mostenita clasa JFrame. Aceasta clasa face parte din pachetul presentation.

1. Clasa UpdateView se comporta asemanator cu clasa InsertView, doar ca aceasta cauta in tabel elementul cu id-ul introdus de catre utilizator, iar daca este gasit, modifica valorile introduse. In caz ca se doresc a fi modificate doar cateva valori, se vor introduce doar acelea. Butonul de BACK ne duce in interfata precedenta, adica in SecondView, iar butonul de UPDATE ne realizeaza actiunea de editare a bazei de date. Aceasta clasa este si ea flexibila, in sensul ca este interfata grafica pentru cele 3 clase cu cele 3 tabele pe care se realizeaza operatia de editare. Este mostenita clasa JFrame. Aceasta clasa face parte din pachetul presentation.
2. Clasa ViewClients este clasa ce contine 3 atribute: frame, table, btnBack. In acesta clasa am creat un Jtable pentru clienti pe care l-am populat cu elementele din baza de date. Butonul de BACK ne duce in interfata precedenta, adica in SecondView. Este mostenita clasa JFrame. Aceasta clasa face parte din pachetul presentation.
3. Clasa ViewProducts este clasa ce contine 3 atribute: frame, table, btnBack. In acesta clasa am creat un Jtable pentru produse pe care l-am populat cu elementele din baza de date. Butonul de BACK ne duce in interfata precedenta, adica in SecondView. Este mostenita clasa JFrame.
4. Clasa ViewOrder este clasa ce contine 3 atribute: frame, table, btnBack. In acesta clasa am creat un Jtable pentru comenzi pe care l-am populat cu elementele din baza de date. Butonul de BACK ne duce in interfata precedenta, adica in SecondView. Este mostenita clasa JFrame.
5. Clasa Start contine metoda main in care ne-am creat fisierul „bill.txt” si in care ne-am creat un obiect de tipul Controller-ului.

# Rezultate

In urma plasarii comenzilor se calculeaza o factura aferenta acelei comenzi.

# Concluzii

În urma implementării acestei aplicații numite ORDERS MANAGEMENT am reușit să imi aprofundez cunostintele legate de programarea orientata pe obiect, precum si cele de SQL si am reusit sa invat cum functioneaza o aplicatie de tipul layered architecture. Am invatat de asemenea cum se conecteaza o baza de date la o aplicatie.

Printre îmbunătățirile pe care le-aș aduce eu proiectului meu, ar fi adaugarea unei clase Feedback unde clientii care au plasat minim 2 comenzi, sa poata sa lase un feedback produsului pe care acestia l-au cumparat. O alta imbunatatire ar fi crearea unui cos de cumparaturi, astfel incat clientul sa poata sa comande produse diferite intr-o singura comanda.

# Bibliografie

* 1. 1. <https://www.oracle.com/technical-resources/articles/java/javareflection.html>
  2. <https://www.jetbrains.com/help/idea/working-with-code-documentation.html#view-javadocs>
  3. https://[www.oracle.com/ro/database/what-is-a-relational-database/](http://www.oracle.com/ro/database/what-is-a-relational-database/)