****

**TEHNICI DE PROGRAMARE**

**ASSIGNMENT 3**

**ORDER MANAGEMENT**

**Chelemen Stefana-Bianca**

**GRUPA 30221**

**1.Obiectivul temei**

Obiectivul principal al acestei teme este acela de a crea o aplicatie implementata in Java, care proceseaza comenzi ale unor clienti pentru un deposit. Aceasta aplicatie va comunica cu o baza de date si va realizeaza operatii cu aceasta. O bază de date este o colecție organizată de informații sau de date structurate, stocate electronic într-un computer. Datele din cele mai obișnuite tipuri de baze de date sunt distribuite de regulă pe linii și coloane, în diferite tabele, pentru eficientizarea procesării și interogării datelor. Datele pot fi accesate, gestionate, modificate, actualizate, controlate și organizate cu ușurință. Majoritatea bazelor de date utilizează un limbaj structurat de interogare (SQL) pentru scrierea și interogarea datelor. Vom implementa operatii precum inserarea , stergerea , editarea , cautarea s.a.m.d .

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

***2.1.Analiza problemei***

Principalul scop este de a implementa un program care executa correct operatiile specifice bazei de date , utilizand paradigmele Programarii Orientate pe Obiect . Pentru a realiza acest lucru , trebuie sa ne conectam la baza noastra de date , sa implementam operatii de baza cu baza de date , pentru a putea introduce in aceasta clienti , produse , comenzi si facturi , si multe alte operatii ( stergere ,listare , editare ) .

De asemenea , trebuie sa folosim anumite tehnici precum “Java Reflection” pentru a crea o metoda care primeste o lista de obiecte si genereaza antetul tabelului , extragand proprietatile obiectului si apoi populeaza tabelul cu valorile elementelor din lista .

***2.2.Modelare***

Primul pas pe care trebuie sa il facem pentru a putea implementa cu success programul , este sa ne conectam la baza de date . Aceasta conectare am realizat-o in pachetul “connection” , in clasa “ ConnectionFactory “ .

Apoi , trebuie sa adaugam in pachetul “model“ , cate o clasa pentru fiecare tabel pe care il avem in baza de date ( atributele fiecarei clase trebuie sa fie exact de acelasi timp ca si coloanele din tabelele bazei de date ).

Mai departe , in pachetul “dao” , in clasa“AbstractDAO “ , am implementat operatiile bazei de date cu ajutorul JavaReflection ( insert , delete , edit , list ) . In acelasi pachet am mai definit clasele “ClientDAO”,”ProductDAO”,”OrderDAO” , pentru operatii specific fiecarui tablel.

In pachetul “presentation“ , am implementat 6 clase care au legatura cu interfata grafica . Prima clasa este clasa “View “, care la instatierea acesteia se deschide fereastra principal a aplicatiei. Clasa View are ca si componente un panel pe care se afla 3 butoane numite sugestiv “ Client”, “Order”, “Product” , in functie de ce operatii doreste sa faca utilizatorul. Daca apasam butonul “Client” se va face o instant a clasei “ClientFrame”, din care putem realiza toate operatiile cu un obiect de tip Client corespunzator cu tabela “client” din baza de date “warehouse”. Daca apasam butonul “Product” se va face o instanta a clasei “ProductFrame”, din care putem realiza toate operatiile cu un obiect de tip Product corespunzator cu , tabela “product” din baza de date “warehouse” . Daca apasam butonul “Order” se va face o instanta a clasei “OrderFrame”, din care putem realiza toate operatiile cu un obiect de tip Order corespunzator cu , tabela “orders” din baza de date “warehouse” . Ultimele doua clase ale acestui pachet sunt “GenerateBill” , care genereaza o chitanta pentru o comnada in format “.txt” si clasa “GeneratePdfBill” care genereaza o chitanta pentru o comnada in format “.pdf”.

In final, in pachetul start, se afla o clasa “Main” cu o metoda main din care pornim aplicatia.

***2.3.Scenariul***

Pentru ca programul sa poata rula correct utilizatorul trebuie sa introduca comenzile :

* pentru inserarea clientului

In fereastra principala, apasam butonul “CLIENT”. In fereastra “Client” se introduc

In textFieldurile corespunzatoare toate informatiile pentru identificarea unui client si se apasa butonul “New Client”.

* pentru inserarea produsului

In fereastra principala, apasam butonul “PRODUCT”. In fereastra “Product” se introduc in textFieldurile corespunzatoare toate informatiile pentru identificarea unui produs si se apasa butonul “Add Product”.

* pentru inserarea unei comenzi

In fereastra principala, apasam butonul “ORDER”. In fereastra “Order” se introduc in textFieldurile corespunzatoare toate informatiile pentru identificarea unei comenzi si se apasa butonul “Add Order”.

* pentru stergerea clientului

In fereastra principala, apasam butonul “CLIENT”. In fereastra “Client” introducem numele clientului pe care dorim sa il stergem si se apasa butonul “Delete Client”.

* pentru stergerea produsului

In fereastra principala, apasam butonul “PRODUCT”. In fereastra “Product” se introduce id-ul produsului pe care dorim s ail stergem si se apasa butonul “Delete Product”.

* pentru vizualizarea listei de clienti

In fereastra principala, apasam butonul “CLIENT”. In fereastra “Client” apasam butonul “View Clients” si putem vizualiza lista de client in tabelul asezat in partea dreapta a ferestrei.

* pentru vizualizarea listei de produse

In fereastra principala, apasam butonul “PRODUCT”. In fereastra “Product” apasam butonul “ViewProducts” si putem vizualiza lista de cproduse in tabelul asezat in partea dreapta a ferestrei.

* pentru vizualizarea listei de comenzi

In fereastra principala, apasam butonul ”ORDER”. In fereastra “Order” apasam butonul “ViewOrders” si putem vizualiza lista de comenzi in tabelul asezat in partea dreapta a ferestrei.

* Pentru generarea chitantei unei comanzi

In fereastra principala, apasam butonul ”ORDER”. In fereastra “Order” apasam butonul “Generate Bill”. Se va genera o chitanta sub 2 forme: fiesier text sau fisier pdf.

In cazul generarii comenzii , daca cantitatea de produs dorita este mai mare decat cantitatea de produs exista in stoc , va aparea o fereastra cu un mesaj de atentionare. De asemenea daca dorim sa facem o comanda cu un produs care nu exista, se va generea de asemena o fereastra cu un mesaj de atentionare.

***2.4.Cazuri de utilizare***

In cazul generarii comenzii , daca cantitatea de produs dorita este mai mare decat cantitatea de produs exista in stoc , va aparea o fereastra cu un mesaj de atentionare. De asemenea daca dorim sa facem o comanda cu un produs care nu exista, se va generea de asemena o fereastra cu un mesaj de atentionare.

**3.Proiectare**

Respectand paradigmele programarii orientate pe obiect si folosind Layered Architecture , am impartit programul in mai multe pachete, fiecare continand mai multe clase. Voi prezenta mai jos structura fiecarui pachet:

• Pachetul ‘ connection ‘ care contine clasa : ConnectionFactoy ;

• Pachetul ‘ dao ‘ care contine clasele : AbstractDAO , ClientDAO , OrderDAO , ProductDAO ;

• Pachetul ‘ model ‘care contine clasele : Client , Order , Product,Bill ;

• Pachetul ‘ presentation ‘ care contine clasele : GenerateBill,GeneratePdfBill, ClientFrame , OrderFrame , ProductFrame , View ;

• Pachetul ‘ start ‘ care contine clasa Start .

Diagrama UML de clase a proiectlui Java .

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

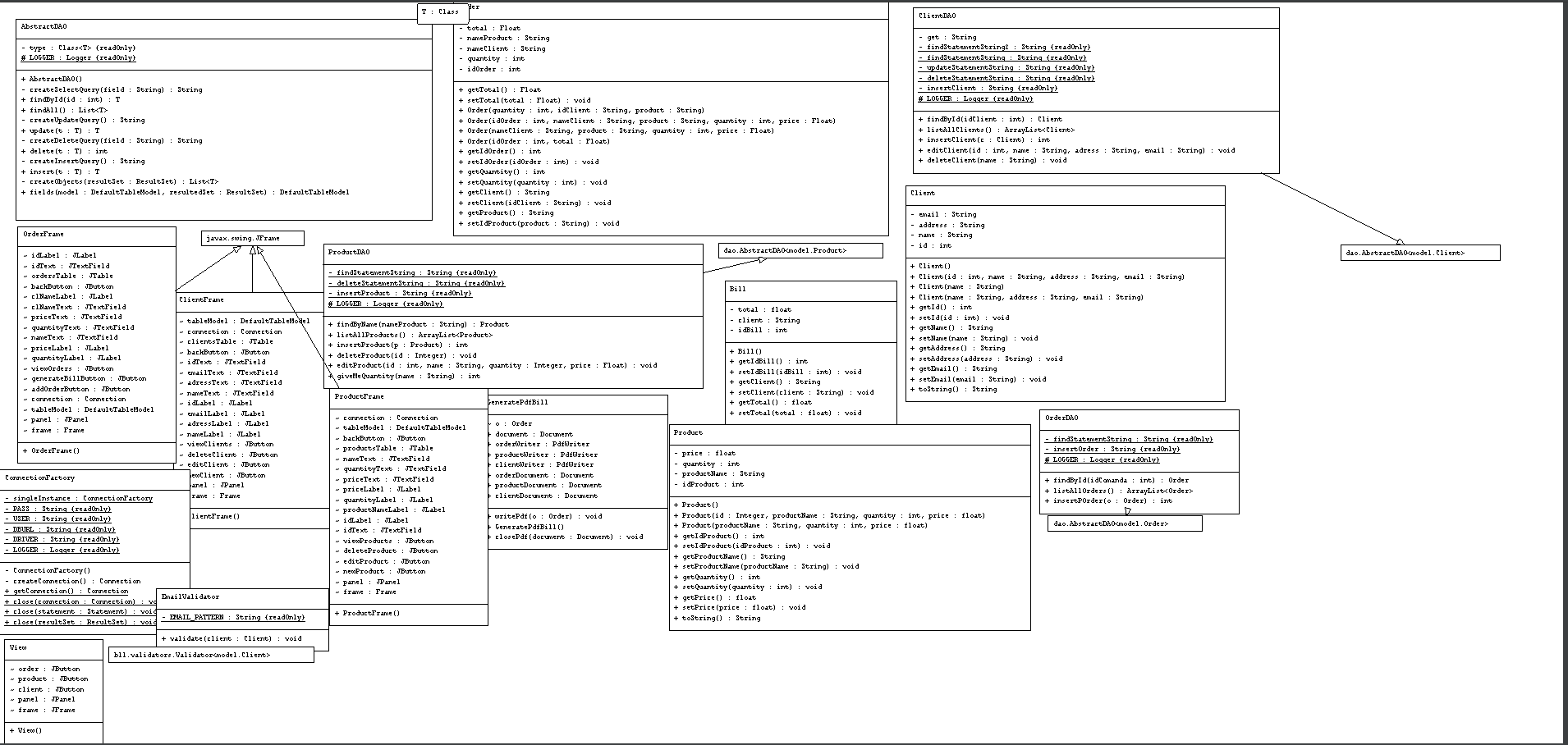
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

fig 1 . – diagram de clase a proiectului

Tabele din baza de date ‘warehouse’ Din MySQL descries in limbajul SQL:

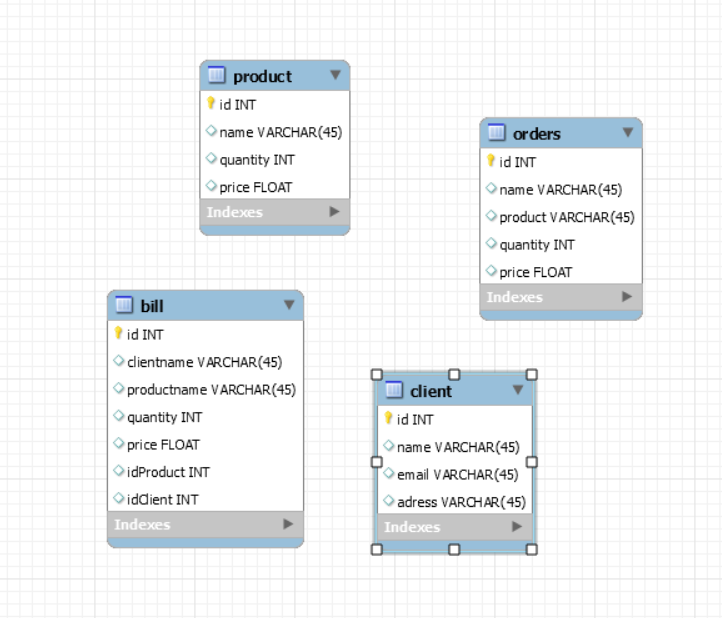


fig 2 . –diagrama generata de MySQL pentru baza de date ‘warehouse’

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

. . . .

**4.Implementare**

Primul pachet implementat este pachetul “model“ , care contine 4 clase ( cate o clasa pentru fiecare tabela din baza de date “ warehouse ” ) :

⮊ Clasa “Client ” : contie urmatoarele atribute si metode :

⬧ id , name , address , email

⬧ getters si setters pentru fiecare atribut

⬧ mai multi contructori

⬧ o metoda toString() pentru afisarea unui obiect de tip Client

⮊ Clasa “ Product “ : contine urmatoarele atribute si metode :

⬧ idProduct , productName , quantity , price

⬧ getters si setters pentru fiecare atribut

⬧mai multi constructori

⬧ o metoda toString() pentru afisarea unui obiect de tip Product

⮊ Clasa “ Order “ : contine urmatoarele attribute si metode :

⬧ idOrder , quantity , nameClient , nameProduct,total

⬧ getters si setters pentru fiecare atribut

⬧ mai multi constructori

⮊ Clasa “ Bill “ : contine urmatoarele attribute si metode :

⬧ clientName , idBill , totalPrice

⬧ getters si setters pentru fiecare atribut

⬧ un constructor fara parametrii

Al doilea pachet implementat este pachetul “connection “ , care contine o singura clasa : “ConnectionFactory”.

In aceasta clasa sunt metode care realizeaza conexiunea dintre programul Java si baza de date “warehouse”.

Al treilea pachet implementat este pachetul “dao” care contine 4 clase: “AbstractDAO” , “ClientDAO”, “ProductDAO” si “OrderDAO . ”

Clasa “AbstractDAO” :contine urmatoarele metode:

⬧LOGGER,type

⬧un constructor

⬧ metoda createSelectQuery

⬧metoda createUpdateQuery

⬧metoda createInsertQuery

⬧ metoda createDeleteQuery

⬧ metoda insert

⬧ metoda update

⬧ metoda delete

⬧ metoda findById

⬧ metoda findAll

⬧ metoda createObjects

Al patrulea pachet implementat este pachetul “presentation” care contine 6 clase: “View” , “ClientFrame”, “ProductFrame”, “OrderFrame ,” GenerateBill” si “GenereatePdfBill”.

In clasa “View ” exista un constructor care adauga componentele in fereastra principala aaplicatiei.

In clasa “ClientFrame” exista un constructor care adauga componentele in fereastra Client a aplicatiei.

In clasa “ProductFrame” exista un constructor care adauga componentele in fereastra Product a aplicatiei.

In clasa “OrderFrame” exista un constructor care adauga componentele in fereastra Order a aplicatiei.

Clasa “GenerateBill” contine:

⬧metoda genereate(Object o) –care genereaza chitanta unei comenzi intr-un fisier.txt

Clasa “GeneratePdfBill” contine:

⬧metoda writePdf(Ordert o) –care genereaza chitanta unei comenzi intr-un fisier.pdf

⬧ metoda close()-care inchide fisierul .pdf

⬧ un constructor gol

Ultimul pachet implementat este pachetul “start” care contine o clasa: “Main” . Clasa “Main” contine o metoda main din care instantiem clasa “View” si din care se porneste aplicatia propriu-zisa.

**5.Rezultate**

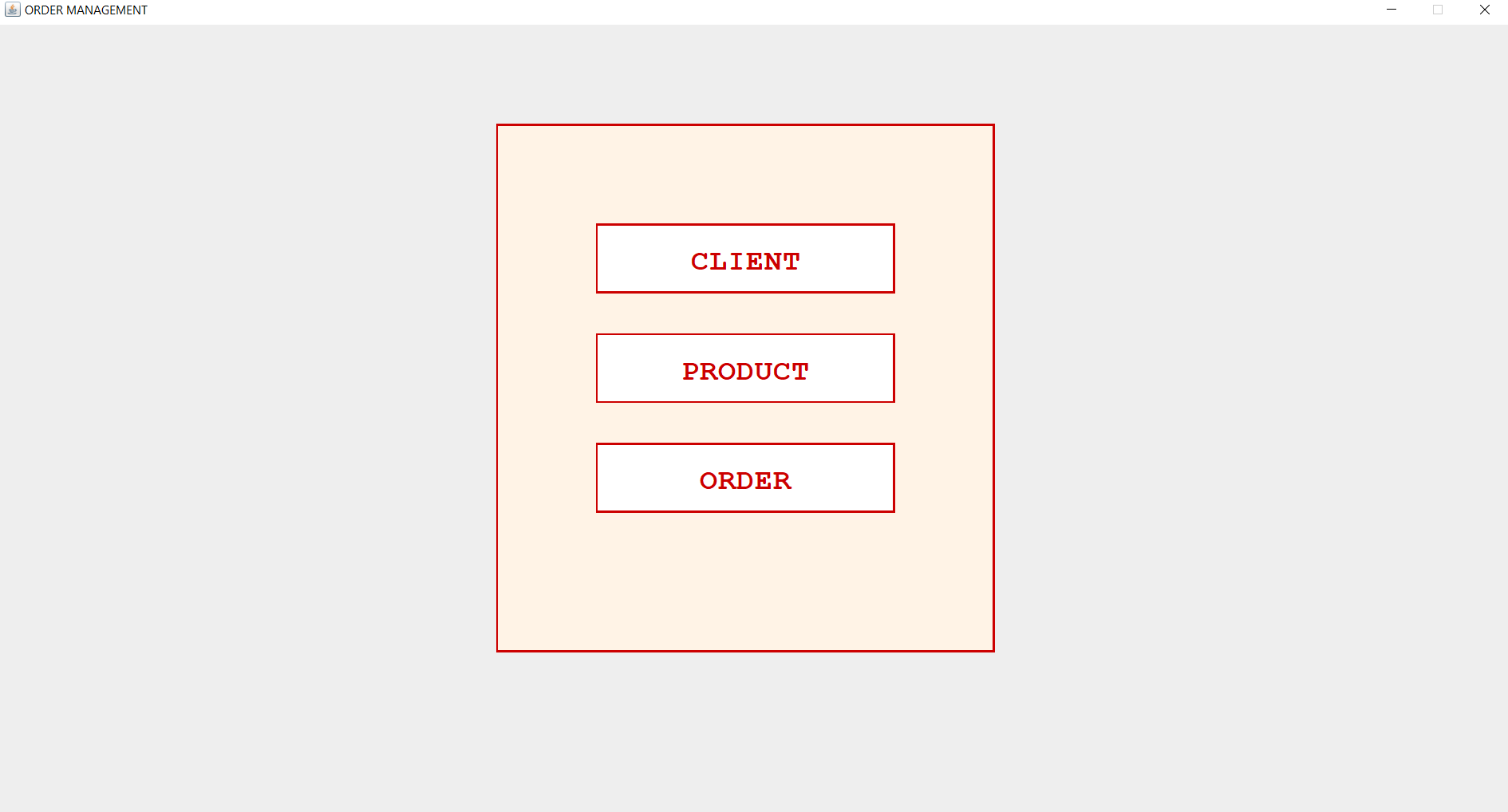


fig 3 . –fereastra principala a aplicatiei; se pot apasa cele 3 butone: “CLIENT” ,”PRODUCT”,”ORDER” , pentru a accesa celelalte 3 ferestre

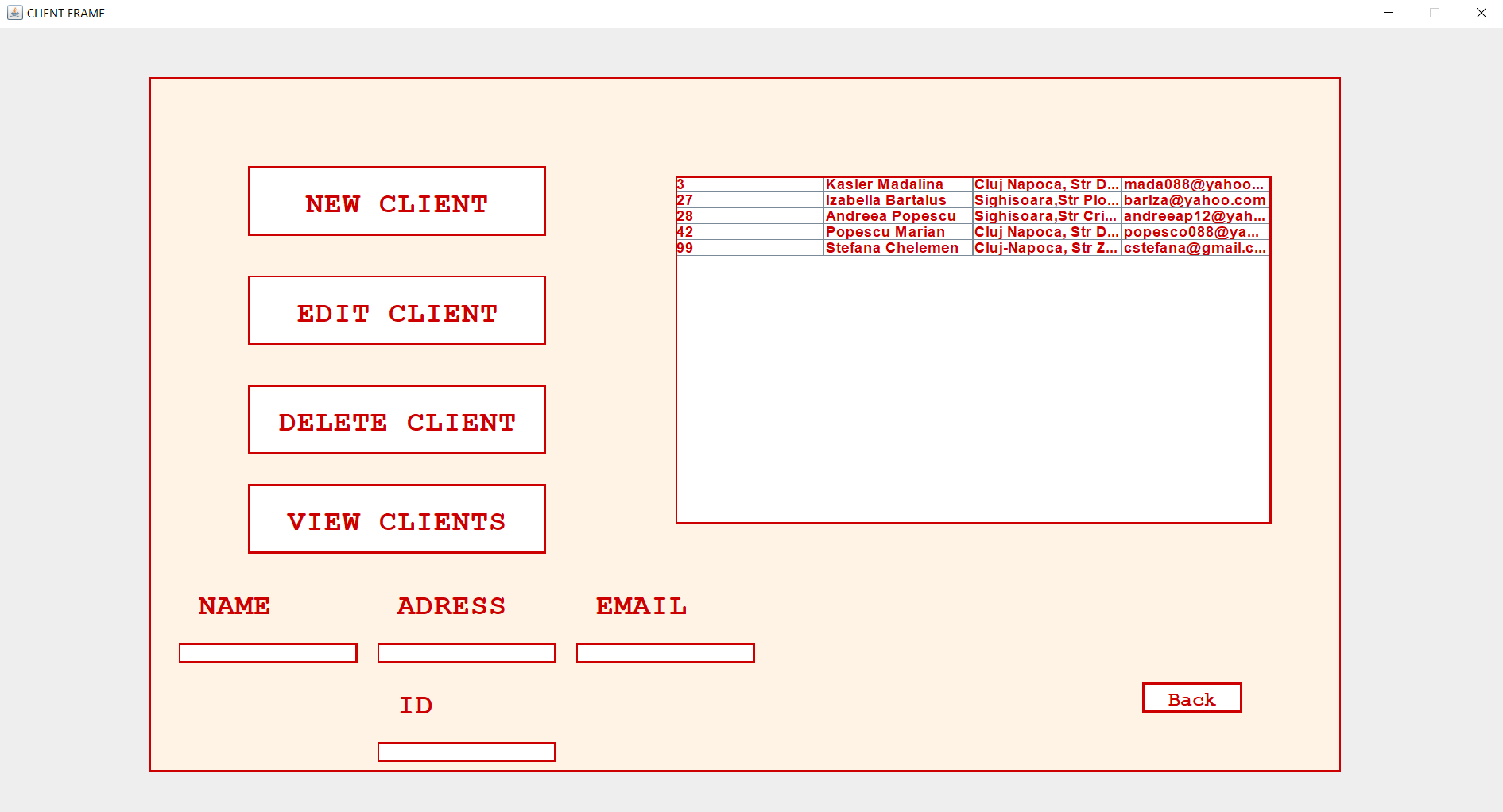


fig 4 . –fereastra “Client ”a aplicatiei; se pot apasa cele 4 butone pentru operatiile aferente numelor butoanelor: “NEW CLIENT” ,”EDIT CLIENT”,”DELETE CLIENT” , “VIEW CLIENTS”

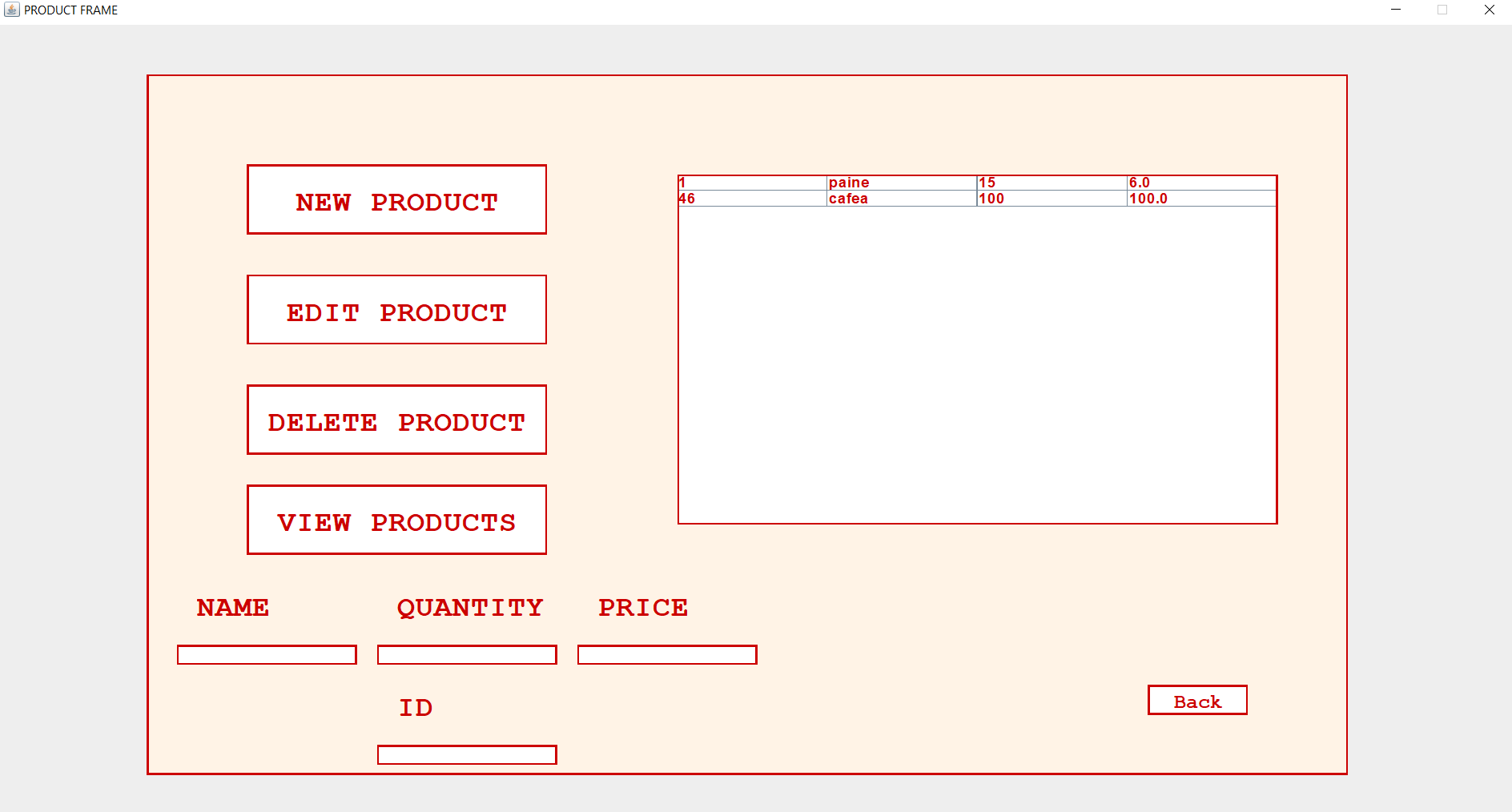


fig 5 . –fereastra “Product ”a aplicatiei; se pot apasa cele 4 butone pentru operatiile aferente numelor butoanelor: “NEW PRODUCT” ,”EDIT PRODUCT”,”DELETE PRODUCT” , “VIEW PRODUCTS”

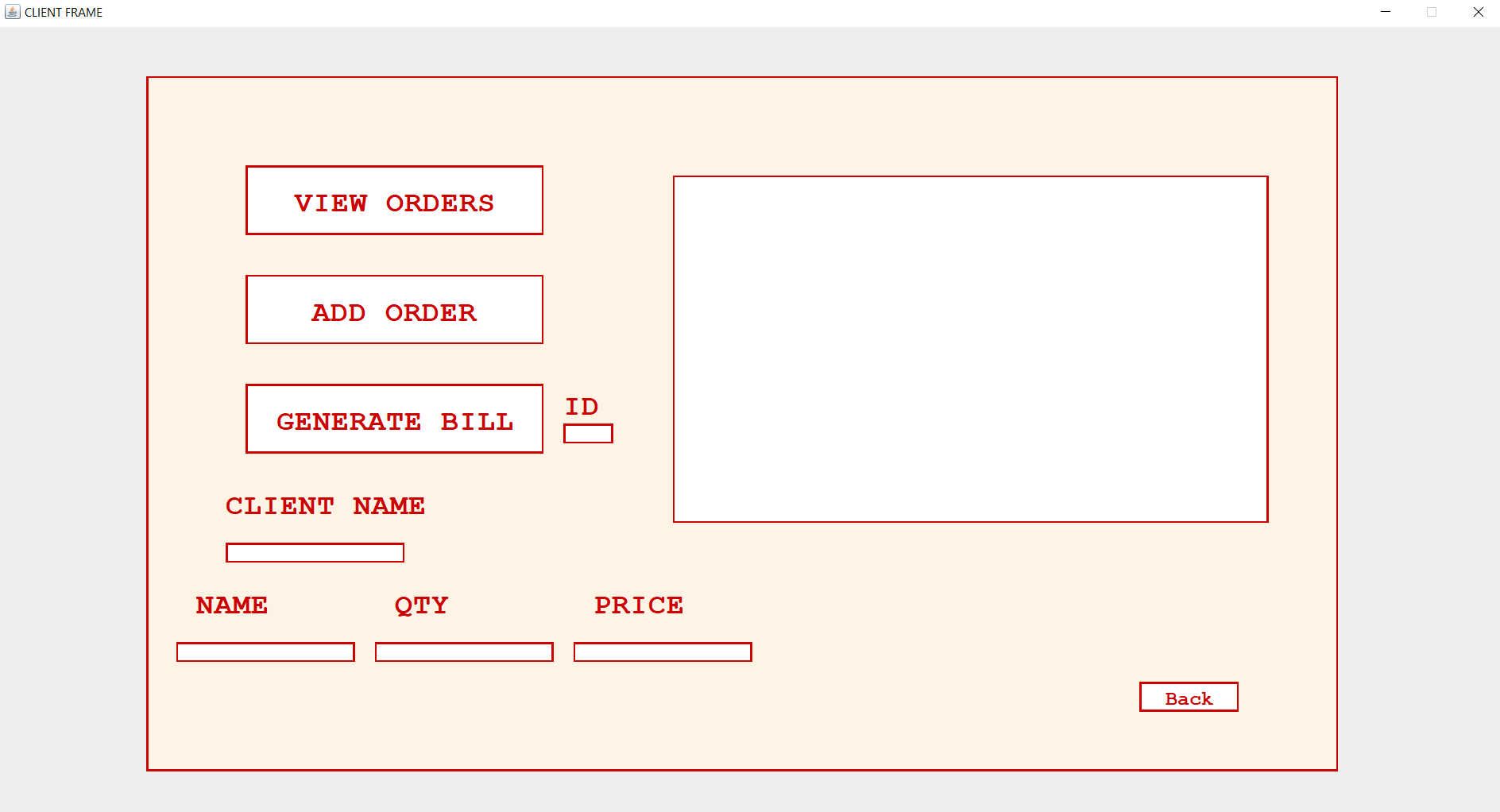


fig 6 . –fereastra “Orders”a aplicatiei; se pot apasa cele 3 butone pentru operatiile aferente numelor butoanelor: “VIEW ORDERS” ,”ADD ORDER”,”GENERATE BILL” ;

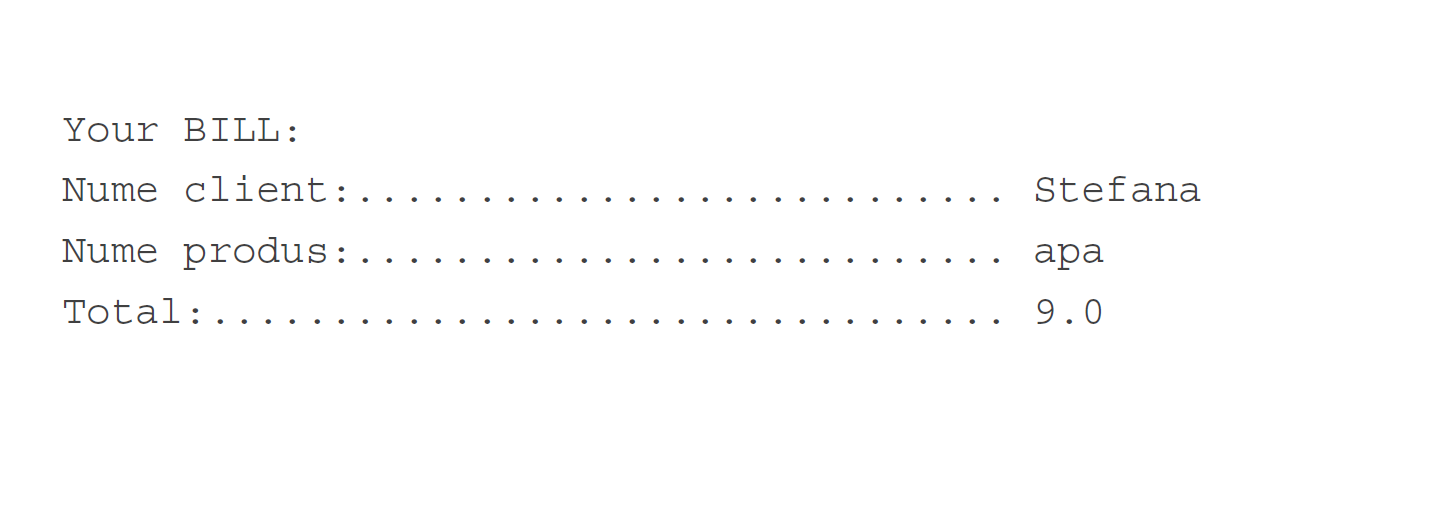


fig 7 . –exemplu de chitanta a unei comenzi generata in PDF

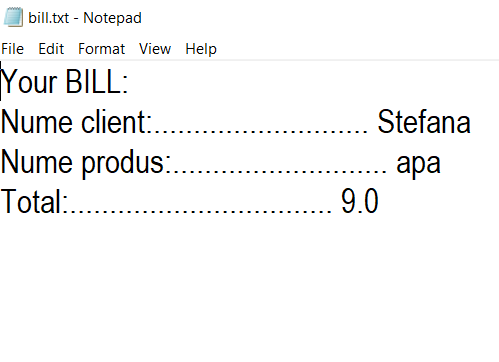


fig 8 . –exemplu de chitanta a unei comenzi generata intr-un fisier txt

**6.Concluzii**

In concluzie aceasta tema a fost foarte utila pentru deoarece prin intermediul ei am aprofundat cunostiintele despre baze de date si despre gestionarea unei baze de date prin intermediul unui program Java si am invatat tehnici noi de programare precum JavaReflection.

**7.Bibliografie** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

<https://www.baeldung.com/java-jdbc> . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

<http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/> . . . . . . . . . . . . . . . .

<https://dzone.com/articles/layers-standard-enterprise> . . . . . . . . . . . . . . . .

<http://tutorials.jenkov.com/java-reflection/index.html> . . . . . . . . . . . . . . . .

<https://www.baeldung.com/java-pdf-creation> . . . . . . . . . . . . . . . .

<https://www.baeldung.com/javadoc> . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

<https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html> . . . . . . . . . . . . . . . .