MINISTERUL EDUCAŢIEI ŞI CERCETĂRII ŞTIINŢIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICA SI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

DISCIPLINA TEHNICI DE PROGRAMARE

Documentatie Tema3

- Order Management –

Popa Alexandra Maria

Grupa 30224

CTI-ro

Cuprins:

1.Obiectivul Temei:

Enuntul Temei : Propunerea, proiectarea si implementarea unui sistem de management al comenzilor pentru un depozit. Aplicatia trebuie sa utilizeze baze de date relationale pentru stoacrea clientilor, a produselor si a comenzilor.

Obiective :

Obiectiv principal : Obiectivul acestei teme este proiectarea si implementarea unei aplicatii de gestionare a comenzilor primate de catre un depozit.

Obiective secundare:

- organizarea pe clase si pachete-> utilizarea diagramelor de clase si pachete
- dezvoltarea algoritmilor folosind POO -> algoritmi de preluarea si procesarea a informatiilor introduse in interfata grafica
- utilizarea structurilor de date -> specific pentru lucrul cu diferiti algoritmi ce au dus la realizarea simularii
- formularea de scenarii -> modalitati introducere, editate si stergere clientilor, produselor sau comenzilor stocate in baza de date
- implementarea si testarea solutiilor obtinute -> testare prinvizualizarea datelor obtinute in urma unor opeartii efectuate pe baza de date; aceste informatii sunt afisate intr-un table in interfata grafica sau printr-un mesaj , in caz de eroare.

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

Problema enuntata necesita cunostiinte leagate de lurcrul baze de date si gestionarea acesteia pentru realizarea unor operatii de manipularea a datelor stocate.De asemenea, se cere utilizarea tehnicii *Reflection* cat si cunoasterea programarii orientate pe obiect pentru crearea si manipularea obiectelor. Totodata, este nevoie de cunoasterea modalitatilor de crearea a

interfetei grafice si de afisarea a tabelelor din baza de date, cat si preluarea datelor de intrare prin intermediul acestei interfete .

Din analiza problemei rezulta faptul ca avem nevoie de un set de date de intrare si un set de date de iesire. Datele de intrare sunt specifice pentru fiecare tip de data stocata in baza de date : client, produs sau comanda, cat si in functie de operatia care se doreste a fi realizata. Pentru toate tipurile de tabele stocate in baza de date se pot face urmatoarele operatii in funtie de datele existente in acestea: adaugare client / produs / comanda, eliminare client / produs / comanda , editare client / produs /comanda si afisarea tabelului impreuna cu continutul actualizat in urma procesarii operatiilor. Prin afisarea acestor tabele se poate verifica stocul unui anumit produs, sau se poate vizualiza comanda plasata impreuna cu pretul total si informatiile referitoare la acesata. Pentru tabela ce contine date de tip client este necesara introducerea unui ID (unic) , a numelui, adresei si email-ului necesare operatiilor de tip adaugare si editare. Pentru adaugarea si editarea produselor din baza de date, trebuie preluate prin intermediul interfetei grafice informatii precum : ID-ul produsului (unic), denumirea, cantitatea disponibila in stoc, cat si pretul unui prous. Adaugarea si editarea comenzilor din baza de date necesita introducerea de catre utilizator urmatoarelor informatii : ID comanda(unic), ID client si ID produs existente in baza de date si cantitatea dorita din produsul ales. Pentru toate tipurile de tabela, operatiile de stergere necesita doar introducerea ID –ului respectivului client / produs / comenzi ce se doreste a fi eliminate/a. Totodata editarea necesita introducerea tuturor informatiilor referitoare la respectivul tip de obiect si schimbarea unui singur atribut ce urmeaza sa fie actualizat.

In cazul introducerii unor informatii ce nu se afla in concordanta cu cerintele aplicatiei, se va afisa un mesaj in consola, ce notifica faptul ca datele sunt invalide sau inexistente in baza de date. De exemplu, daca se introduce o adresa de email care nu respecta conditia de a se termina in “@yahoo.com”, utilizatorul va primi un mesaj de eroare in introducerea datelor nesesarea adugarii unui nou client. Totodata,in cazul in care cantitatea de produse ceruta de utilizator prin intermediul unei comenzi nu exista in stoc, se va afisa mesajului “Stoc insuficient”.

Metoda de introducere a datelor a fost special aleasa pentru ca utilizatorul sa poata vizualiza cu usurinta datele introduse de acesta sis a le poata modifica in cazul in care este necesar acest lucru.

Dupa introducerea datelor initiale de catre utilizator,acestea sunt preluate si modelate in functie de operatia ceruta. Datele sunt preluate, iar prin intermediul lor se realizeaza interogari, populari, actualizari si stergeri asupra bazei de date. Rezultatele obtinute vor fi puse la dispozitia utilizatorului sub forma unui table ce cotine toate informatiile necesare despre clienti, produse sau comenzi. Afisarea trebuie sa fie clara pentru utilizator, scopul fiind ca produsul final sa fi cat mai practic si usor de folosit.

3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator) :

Proiectarea aplicatiei de management a comenzilor primate de catre un depozit a necesitat proiectarea a 4 pachete de clase specifice: BusinessLogicClasses, DAOClasses, DBAccessClasses, Model si Presentation, plus un pachetul ce contine clasa principala, din care ruleaza aplicatia-*App* . *BusinessLogicClasse*s care cuprinde clasele care alcatuiesc logica aplicatiei : *ClientBLL, ComandaBLL, ProdusBLL; DAOClasse*s ce contine clase care fac legatura directa cu baza de date prin manipularea acesteia cu ajutorul anumitor metode definite si implementate : *AbstractDAO, ClientDAO, ComandaDAO, ProdusDAO; DBAccessClasses* ce contine clasa care face conexiunea cu baza de date: *ConnectionFactory*; *Model* ce contine clase corespunzaoare tabelelor din baza de date: *Clientul, Produs, Comanda si PresentationClasses* ce cuprinde clase ce realizeaza interfata grafica si totodata comunicarea utilizatorului cu modelul si cu logica din spatele aplicatiei. Interfata grafica se bazeaza pe sablonul MVC ce presupune separarea proiectului in *Model, View si Controller*, in functie de nevoi, reliefand astfel principiile POO.In continuare voi prezenta proiectarea in mare a claselor din pachetele ce

formeaza aplicatia, metodele acestora fiind descrise mai pe larg in capitolul 4.

Model :

Clientul -> aceasta clasa are 4 variabile instanta ce reprezinta atributele clientilor stocati in baza de date: *idClient* –unic de tipul *int* si nume, adresa si email de tipul *String*. Clasa contine metode de tip setter si getter : *getIdClient( ), getNume( ), getAdresa( ), getEmail( ), setIdClient( ), setNume( ), setAdresa( ), setEmail( )* si implementeaza metoda *toString( ).*

Produs -> contine 4 variabile instanta: *idProdus*- unic de tip *int*, denumire- de tip *String*, cantitatea-cantitatea produsului disponibila in stoc si *pretProdus* de tip *int*. Clasa contine metode de tip setter si getter : *getIdProdus( ), getDenumire( ), getCantitate( ), getPretProdus( ), setIdProdus( ), setDenumire( ), setCantitate( ), setPretProdus( )* si implementeaza metoda *toString( )*.

Comanda -> contine 5 variabile instanta de tip int: *idComanda*- ID-ul comenzii, unic; *idClient*- ID-ul clientului ce face comanda; *idProdus*- ID-ul produsului comandat; *cantitate*- cantitatea dorita din produsul comandat si *pretComanda*- pretul total al comenzii plasate.

DBAccessClass :

ConnectionFactory -> aceasta clasa realizeaza conexiunea la baza de date existenta in MySQL prin metodele : *createConnection( ), getConnection( ), close(Connection connect), close(PreparedStatement statement) si close(ResultSet resultSet)*. Totodata contine variable instant de tip *Logger* si *String* : LOGGER*, DRIVER, DBURL, USER, PASSW si connection* de tipul *Connection*;

DAOClasses :

AbstractDAO -> contine metode ce descriu interogari ale bazei de date l nivel general. Contine o constanta type de tip *Class<T>* si o variabila instant de tip *Logger*. Metodele implementate in intriorul acestei clase sunt: *createSelectQuery(String field), createInsertQuery( ), insertItem(T obj),*

createDeleteQuery(String field), deleteItem(int id), findById(int id), createUpdateQuery(String field, String someField), updateItem (String field, String someField, Object obj1, Object obj2), createViewAllQuery( ), listTable ( ), CreateObjects(ResultSet resultSet).

Clasele *ComandaDAO, CustomerDAO, ProductDAO, OrderItemDAO* extind clasa AbstractDAO avand ca scop reutilizarea metodelor acestei aceasta clase.

BusinessLogicClasses:

Clasele ce urmeaza a fi prezentate definesc logica din spatele procesarii datelor de intrare.

ClientBLL -> contine o variabila instanta de tip *ClientDAO- client*, si urmatoarele metode : getters / setters (*getClient( ),setClient(ClientDAO client)*), *validareDate(Clientul c), insertNewClient(Client c), deleteClient(int id), updateClient(Client c), viewTabel( ).*

ProdusBLL -> contine o variabila instanta de tip *ProdusDAO- produs*, si urmatoarele metode : getters / setters (*getProdus( ),setProdus (ProdusDAO produs)*), *validareDate(Produs p), insertNewProdus (Produs p), deleteProdus (int id), updateProdus (Produs pNou), viewTabel( )*.

ComandaBLL -> contine 3 variablie instanta: comanda- de tip *ComandaDAO*, client- de tip *ClientDAO* si produs- de tip *ProdusDAO*, si urmatoarele metode : getters / setters (*getComanda( ),setComanda (ComandaDAO comanda)), validareDateComanda(c), insertNewComanda (Comanda c), deleteComanda (int id), updatePretComanda (Comanda c, int pretProdus), updateComanda (Comanda cNoua), viewTabel( ).*

PresentationClasses :

Fiecarei tip de tabela stocata in baza de date ii corespunde o fereastra de introducere si vizualizare de catre untilizator a datelor specifice . Ca urmare vom avea nevoie pentru fiecare fereastra de o clasa de tip *View* si una de tip *Controller*.

ViewPrincipal -> corespunde ferestrei principale in care sunt prezentate optiunile puse la dispzitia utilizatorului, in functie de operatiie pe care doreste sa le utilizeze si in functie de ce tip de obiect doreste sa manipuleze. Contine o variabila instanta -*optiuni* -de tip *JComboBox<String>* , un buton de tip *JButoom* care face trecerea spre fereastra corespunzatoare alegerii utilizatorului si o variabila instanta de tip *JLabel*. Totodata clasa contine metode de get si set cat si metode ce adauga ascultatori (*action listener*) pentru *comboBox* si buton.

ViewClient -> foloseste 5 variabile instanta de tip *JLablel* care indica semnificatia campurilor text : *clientiL, idClientiL, numeClientiL, adresaClientiL, emailClientiL* ; 4 de tip *JTextField*, editabile, pentru introducerea datelor necesare simularii : *idClientiTF, numeClientTF, adresaClientiTF, emailClientiTF;* 4 variabile de tip *JButton*, la apasarea carora se executa operatia dorita :*insertClientB, stergereClientB, editareClientB, vizualizareClientB*; o variabila instanta de tip *JTable*, una de tip *JScrollPane* si una de tip *DefaultTableModel* pentru afisarea datelor din baza de date. Casa contne de asemenea metode de set si get, metode ce adauga ascultatori pe butoane, precum si o metoda ce seteaza tabelul afisat in interfata *setTable(JTable tabel).*

Implementarea claselor *ViewComanda* si *ViewProdus* este asemanatoare cu implementarea clasei *ViewClient*, diferenta constand in numarul si denumirea campurilor ce preiau datele de intrare.

ControllerPrincipal -> reprezinta partea de control al aplicatiei, ce decide ce pasi urmeaza sa faca modelul. Clasa contine o variabila instanta- *viewPrincipal*, de tipul *ViewPrincipal* , ce realizeaza legatura cu clasa *ViewPrincipal.* Constructorul clasei contine un ascultator pentru butonul *Continua* , care este adaugat la *viewPrincipal*. Clasa de control contine doua clase interne *ComboBoxListener* si *ViewPrincipalListener* ce implementeaza *ActionListener* si contin metoda *actionPerformed(ActionEvent e)*.

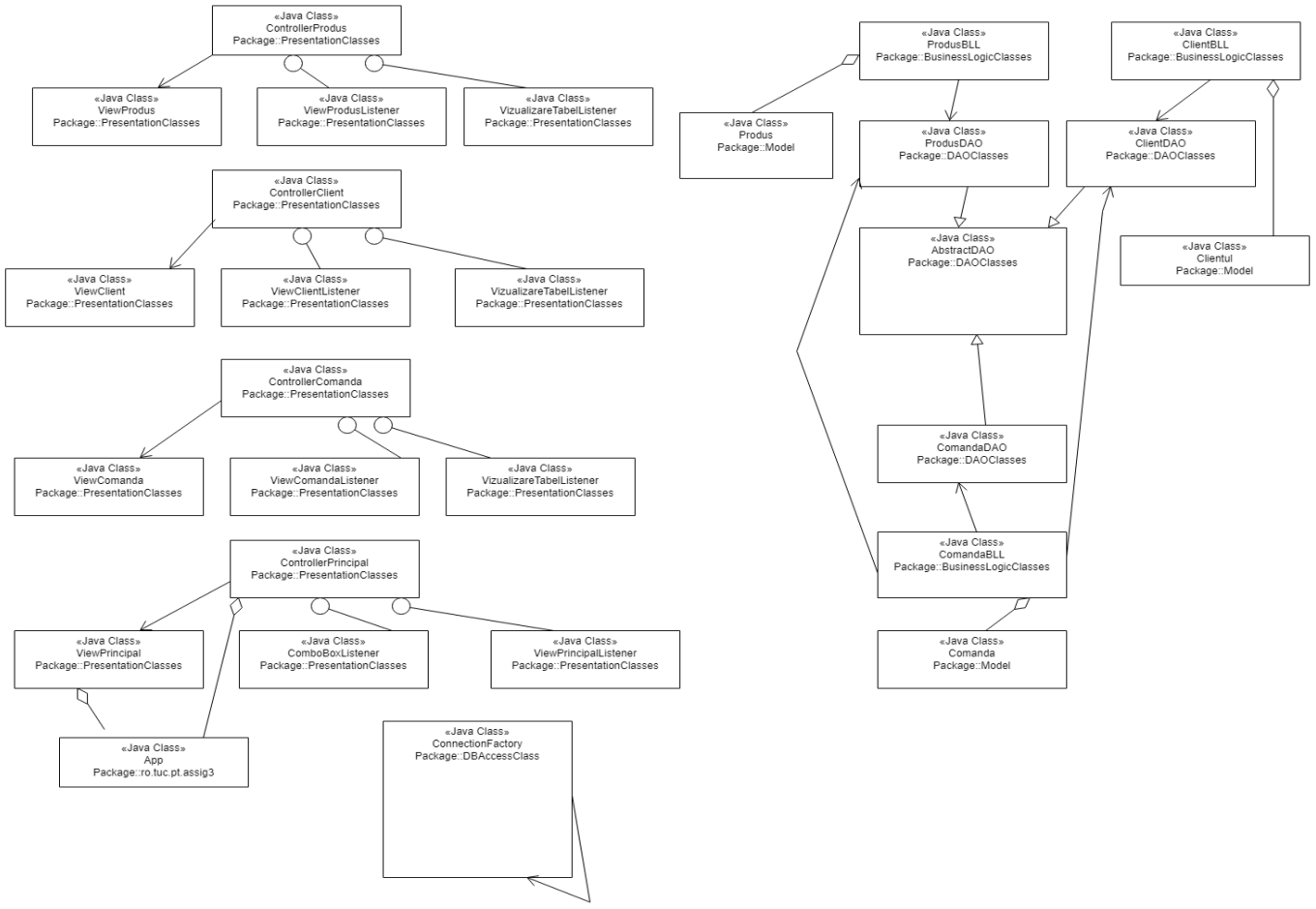
ControllerClient -> contine o variabila instanta- *viewClient* de tipul *ViewClient*, ce realizeaza legatura cu clasa *ViewClient.* De asemenea,

constructorul clasei contine cate un ascultator pentru fiecare buron , care este adaugat la *viewClient*. Clasa contine la randul ei doua clase interne *ViewClientListener* si *VizualizareTabelListener* implementeaza *ActionListener* si contin metoda *actionPerformed(ActionEvent e)*.

Clasele *ControllerProdus* si *ControllerComana* contin la randul lor variabile instanta de tipul view-ului corespunzator , adauga ascultatori pe butoane pentru ca mai apoi sa implementeze logica de preluare si afisarea datelor prin intermediul celor doua clase interne corespunzatoare fiecaria.

App -> contine programul principal care initializeaza interfata si leaga componentele din pachetele aplicatiei impreuna.

In continuare este prezentata diagrama UML de clase a proiectului:



4. Implementare :

In acest capitol se vor descrie deciziile de implementare ale metodelor fiecarei clase in parte, impreuna cu algoritmii ce sunt folositi in realizarea acestora .

ConnectionFactory :

createConnection( ) -> realizeaza conexiunea cu baza de date

getConnection( ) -> preia conexiunea din clasele ce instanteaza obiectul de tip *ConnectionFactory*

close(Connection connect), close(PreparedStatement statement) si close(ResultSet resultSet) -> inchid conexiunile create pentru accesarea bazei de date

AbstractDAO :

createSelectQuery(String field) -> creaza interogarea corespunzatoare MySQL in urma careaia se gaseste un obiect in interiorul unuei tabele, utilizand cautarea dupa ID-ul acestuia.

createInsertQuery( ) -> creaza interogarea specifica inserarii unui obiect in tabela corespunzatoare tipului sau

insertItem(T obj) -> insereaza in tabela corespunzatoare obiectul de tip *T*

createDeleteQuery(String field) -> creaza interogarea specifica stergerii unuei tuple dintr-o anumita tabela

deleteItem(int id) -> sterge obiectul din baza de date in functie de ID-ul corespunzator

findById(int id) -> realizeaza cautarea dupa ID in baza de date a obiectului

createUpdateQuery(String field, String someField) -> creaza o interogare de actualizare a informatiilor corespunzatoare unui obiect in tabela

updateItem (String field, String someField, Object obj1, Object obj2) -> realizeaza actualizarea informatiilor despre un anumit tip de *Object*

createViewAllQuery( ) -> creaza interogarea ce are rolul de afisare a tuturor tuplelor dintr-o tabela

listTable ( ) -> returneaza o lista de obiecte ce urmeaza a fi afisate in interfata grafica prin intermediul unui tabel

CreateObjects(ResultSet resultSet) -> primeste rezultatele unei interogari si le transforma in obiecte, pectru ca mai apoi sa poata sa fie procesate si prelucrate prin logica implementata in aplicatie

ClientBLL:

validareDate(Clientul c) -> primeste un obiect de tip *Clientul* introdus in interfata grafica verifica daca String-urile care ii continute au un numar de caracterea < 30 si daca adresa de email respecta un anumit format

insertNewClient(Client c) -> verifica daca clientul ce se doreste a fi inserat nu exista deja in baza de date si il insereaza uitilizand metoda *insertItem( )* din clasa *AbstractDAO*

deleteClient(int id) -> verifica daca clientul exista in baza de date si apoi il sterge

updateClient(Client cNou) -> compara fiecare camp al clientului nou cu campul clentului vechil cand acestea difera, se actualizeaza campul diferit

viewTabel( ) -> metoda ce returneaza o matrice de obiecte, necesara pentru afisarea tabelului de clienti in interfata grafica

ProdusBLL :

validareDate(Produs p) -> similara metodei *validateDate* din clasa *ClientBLL*

insertNewProdus (Produs p) -> verifica daca produsul ce se doreste a fi inserat nu exista deja in baza de date si il insereaza uitilizand metoda *insertItem( )* din clasa *AbstractDAO*

deleteProdus (int id) -> verifica daca produsul exista in baza de date si apoi il sterge

updateProdus (Produs pNou)-> actualizeaza un sigur camp dintr-o inregistrare pe baza comparatiei cu vechea valoare

viewTabel( )-> metoda ce returneaza o matrice de obiecte, necesara pentru afisarea tabelului de clienti in interfata grafica

ComandaBLL :

validareDateComanda(c)-> valideaza datele corespunzatoare comenzii ce urmeaza a fi procesata

insertNewComanda (Comanda c)-> verifica daca comanda ce se doreste a fi inserata nu exista deja in baza de date si o insereaza uitilizand metoda *insertItem( )* din clasa *AbstractDAO*

deleteComanda (int id)-> verifica daca o anumita comanda exista in baza de date si apoi o sterge

updatePretComanda (Comanda c, int pretProdus) -> actualizeaza pretul comenzii in functie de noua cantitate introdusa

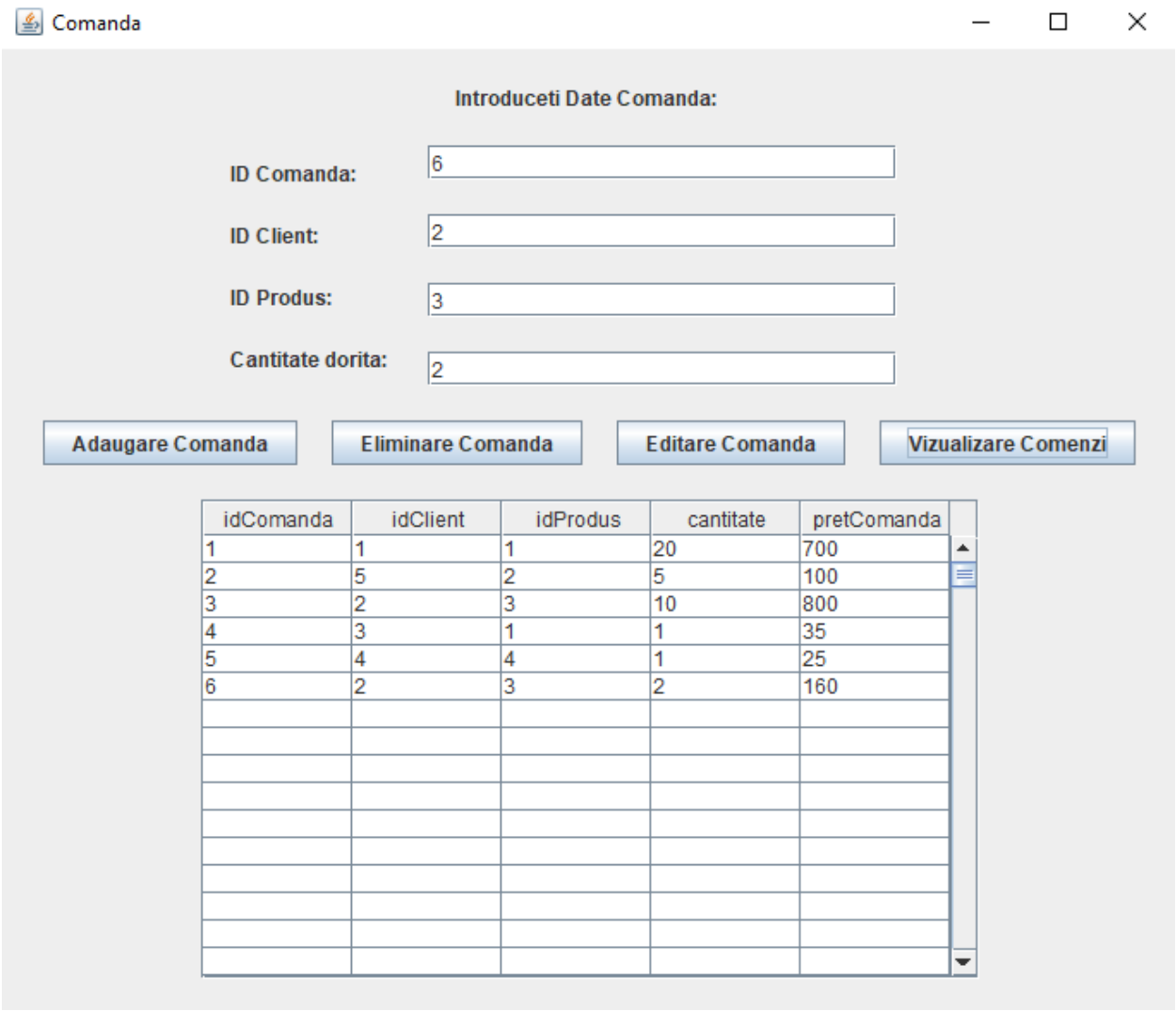
updateComanda (Comanda cNoua) -> actualizeaza un sigur camp dintr-o inregistrare pe baza comparatiei cu vechea valoare

viewTabel( ) -> metoda ce returneaza o matrice de obiecte, necesara pentru afisarea tabelului de clienti in interfata grafica

Metodele din Clasele *BLL* apeleaza la randul lor metode de manipulare a bazei de date din clasa *AbstractDAO* prin intermediul unor obiecte de tip *ClientDAO, ProdusDAO, ComandaDAO* si aplica logica aplicatiei pe datele din tabelele bazai de date.

5. Rezultate:

Rezultatele obtinute in urma plasarii unei comenzi sunt prezentate prin intermediul intergetei grafice.



6.Concluzii :

In urma acestei teme am invatat sa imi structurez mai bine codul in clase si pachete de lucru, astfel incat acesta sa respecte pricipiile POO. Am fost pusa in situatia de a realiza o interfata grafica compusa din mai multe clase de tip *View* si *Controller* ce afiseaza rezultatele actualizate din baza de date intr-un tabel de tip *JTable*, ceea ce a adus un plus considerabil cunostiitelor legate de lucrul cu modelul MVC, cat si in ceea ce priveste modului de a scrie cod. De asemenea, am dobandit cunostiinte legate de tehnica *Reflection* utilizata foarte frecvent in realizarea aestui proiect.

Ca imbunatatiri ulterioare, aplicatia mea ar putea beneficea de imbunatatirea interfetei graficeprin adaugarea uor parole si a logarii in

functie de client sau administrator. Totodata, ca o dezvoltare ulterioare ar fi generarea facturilor pentru clienti in functie de comanda plasata.

7. Bibliografie :

- Indrumator de laborator POO
- Curs POO
- Curs TP
- https://www.w3schools.com
- https://stackoverflow.com/
- http://coned.utcluj.ro/~salomie/PT_Lic/4_Lab/HW3_Tema3/Tema3_HW3_Indications.pdf
- https://stackoverflow.com