

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

ORDER MANAGEMENT

Documentatie Tema 3

Elev: Ţoc Roxana-Ştefania

Grupa: 30224

Profesor indrumător: Antal Marcel

Cuprins:

1.Obiectivul temei

1.1 Obiectivul principal

1.2 Obiectivul secundar

2.Analiza problemei, modelarea si cazuri de utilizare

3.Proiectare

4.Implementare

5.Rezultate

6.Concluzii, dezvoltari ulterioare

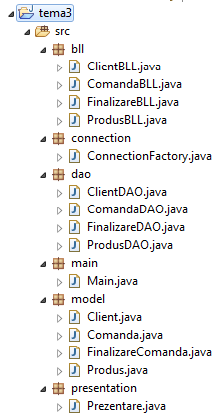
7. Bibliografie

1.Obiectivul temei

1.1 Obiectivul principal

Proiectul are ca scop implementarea unei aplicatii de management a vanzarilor pentru procesarea comenzilor clientilor pentru un deposit. Bazele de date relationale sunt utilizate pentru a stoca clientii, produsele si comenzile. Totodata, aplicatia va fi structurata in pachete folosind o arhitectura pe mai multe nivele si trebuie sa contina urmatoarele clase:

* Model classes : modelele de date ale aplicatiei
* Business Logic classes : implementeaza logica aplicatiei
* Presentation classes : implementeaza datele de intare si iesire ale utilizatorului
* Data acces classes : implementeaza accesul la baza de date



Aplicatia trebuie sa permita procesarea comenzilor dintr-un fisier text dat ca argument. Trebuie implementate si efectuate operatiile de : adaugare client in baza de date , stergerea unui client, adaugarea unui produs, stergerea unui produs, crearea unei comenzi si generarea unor rapoarte in format pdf pentru clienti, produse si comenzi.

1.2 Obiectivul secundar

-Dezvoltarea de use case-uri si scenarii: Realizarea de operatii intr-o baza de date. Se introduc clientii si produsele, iar apoi se genereaza comenzi .

-Alegerea structurilor de date: Folosirea ArrayList in loc de vector pentru usurinta de utilizare

-Impartirea pe clase: Am ales clasele Client, Produs, Comanda si FinalizareComanda pentru a putea inregistra informatiile dorite despre fiecare client, produs, comanda si comanda finala a clientului, iar aceste clase le-am introdus in pachetul “model”. Clasa Main din pachetul “main” si clasa Prezentare din pachetul “presentation”

le-am utilizat pentru a inregistra datele de intrare si iesire. Clasa ConnectinFactory din pachetul “connection” realizeaza conectarea cu baza de date in care se vor inregistra pe parcurs informatiile citite din fisierul de intrare. Clasele ClientDAO, ProdusDAO, ComandaDAO, FinalizareDAO din pachetul “dao” cuprind implementarile operatiilor ce vor fi executate de-a lungul programului. In pachetul “bll” avem clasele ClientBLL, ProdusBLL, ComandaBLL, FinalizareBLL si in fiecare clasa se face legatura cu clasa DAO asociata.

-Dezvoltarea algoritmilor: Algoritmii dezvoltati sunt cei de operare cu o baza de date si gestionarea acesteia

-Implementarea solutiei: Prezentarea claselor si a metodelor realizate si totodata ideile aplicate in solutionarea problemei

2.Analiza problemei, modelarea si cazuri de utilizare

2.1 Analiza problemei

Abstractizarea transpune structura unei probleme din realitate in entitate. Pentru a implementa o metoda cat mai corecta a acestei aplicatii trebuie sa analizam problema pe doua planuri: cel al bazei de date si cel al programarii orientate pe obiecte.

Din punctul de vedere al bazei de date vom avea nevoie de tabele pentru clienti, produse, comenzi si o tabela care surprinde comanda finalizata si care are rolul sa lege cele 3 tabele enumerate anterior.

Din punctul de vedere al implementarii, obiectele vor fi urmatoarele:

- impartirea proiectului dupa arhitectura Layered Architecture

- crearea conexiunii cu baza de date

- crearea claselor din pachetul “model” pentru fiecare tabela din baza de date

- citirea si gestionarea datelor de intrare din fisierul text

- administrarea datelor din baza de date

-generarea rapoartelor in format pdf a clientilor, produselor, comenzilor, precum si bonurile pentru comenzile realizate

2.2 Modelarea si cazuri de utilizare

Datele din fisierul de intrare sunt sub forma urmatoare:

- primul cuvant de pe fiecare linie decide operatia care va avea loc asupra bazei de date: insert, delete, report si order

- pentru operatiile insert, delete si report trebuie specificat si modelul de date care urmeaza sa fie prelucrat: client, product pentru insert si delete, iar pentru report sunt client, product si order

- pentru operatiile insert, delete si order dupa “:” sunt precizate detaliile care trebuie inserate, sterse sau care urmeaza sa fie comandate

Aplicatia poate fi utilizata cu mare succes chiar si atunci cand trebuie stocate cantitati mari de date. Utilizarea programului trebuie sa fie insa restransa la utilizarea unor structuri de mici dimensiuni, clasele principale utilizate in program continand un numar relativ mic de variabile de instanta. Astfel, prea multe detalii ale obiectelor instantiate nu pot fi descrise. Aplicatia poate insa sa fie extinsa pentru a suporta aceste limitari, fapt care va fi detaliat in subcapitolul documentatiei intitulat dezvoltari ulterioare.

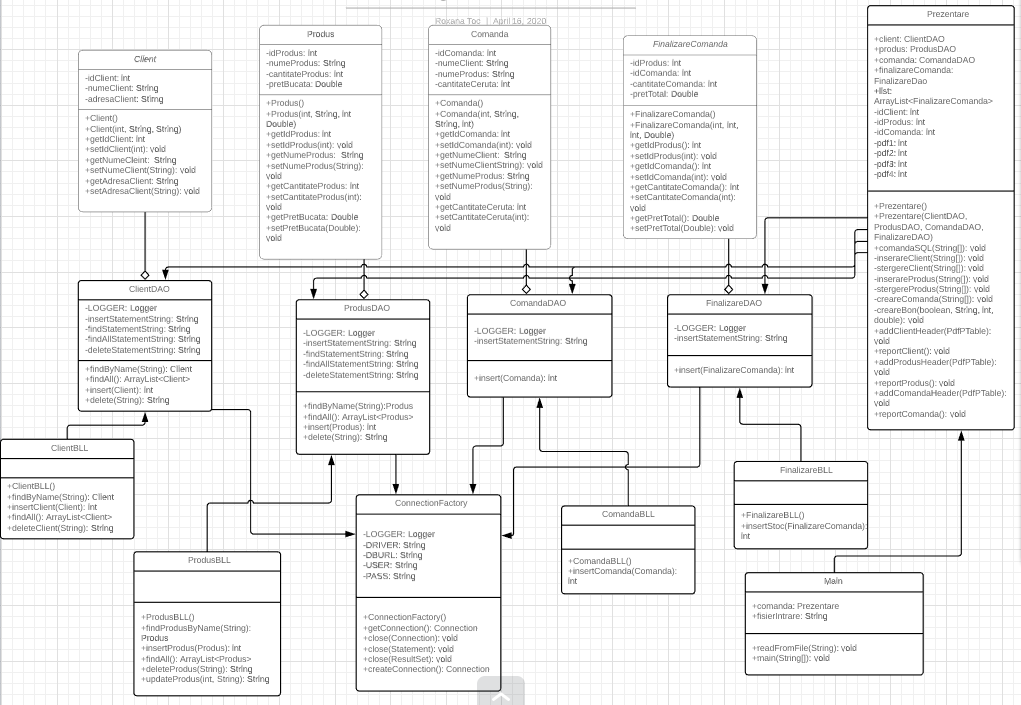
3.Proiectare

3.1 Structuri de date

Ca structura de date, am utilizat un ArrayList de clienti si produse unde am adaugat toti clentii si toate produsele pentru a putea cauta mai usor daca un client sau un produs se afla deja in baza de date.

3.2 Diagrama de clase

Unified Modeling Language (UML) este un limbaj standard pentru descrierea de modele si specificatii pentru software. Este folosita pentru reprezentarea vizuala a claselor si a interdependentelor, taxonomiei si a relatiilor de multiplicitate dintre ele. Diagramele de clasa sunt folosite si pentru reprezentarea concreta a unor instante de clasa, asadar obiecte, si a legaturilor dintre acestea.



4.Implementare

Proiectul este structurat dupa arhitectura Layered Architectured care cuprinde pachetele: bll (Business Logic Layer), dao (Data Access Layer), model, connection, presentation, main.

Nivelul superior al proiectului se afla in pachetul model, cel intermediar il reprezinta pachetul bll, iar nivelul inferior in care se stocheaza datele si se realizeaza conexiunea cu baza de date sunt pachetele connection si dao.

**Pachetul Model**

Contine clasele Client(), Produs(), Comanda(), FinalizareComanda() care au ca atribute: idClient, numeClient, adresaClient, idProdus, numeProdus, cantitateProdus, pretBucata, idComanda, cantitateCeruta, cantitateComandata, pretTotal si fiecare clasa contine metodele corespunzatoare pentru gettere si settere cu ajutorul carora aceste atributele pot fi verificate si modificate in timpul procesarii informatiilor in celelalte clase.

**Pachetul Bll**

In acest pachet se face legatura intre nivelul superior si cel inferior al proiectului. Legatura cu pachetul Dao se realizeaza prin crearea unei instante de tip DAO dintr-o instanta de

tip Model. In clasele ClientBLL si ProdusBLL sunt apelate metodele findByName, findAll care reprezinta select-urile pentru baza de date. Pentru a popula tabelele, in clasele ClientBLL, si ProdusBLL sunt apelate metodele insert, detele si update, iar in clasele ComandaBLL si FinalizareBLL este apelata metoda insert.

**Pachetul Dao**

Clasa ClientDAO

In aceasta clasa sunt incluse metodele prin care se lucreaza cu baza de date. In fiecare metoda se va face conexiunea la baza de date prin care se vor accesa query-urile pentru fiecare metoda pe care vrem sa o accesam in baza de date. In aceasta clasa avem metoda findByName(nume) care primeste ca parametru numele clientului cu ajutorul caruia se cauta in tabela “client” din baza de date, iar clientul cu numele respectiv va fi returnat ca obiect de clasa Client. Metoda delete(nume) primeste un nume al unui client care va fi sters din tabela de clienti. Prin adaugarea clauzei WHERE se realizeaza filtrarea rezultatelor.

Metoda findAll() va returna un ArrayList de tipul Client care contine toti clienti din tabela. Aceasta metoda nu are nevoie de nici o clauza.

Metoda insert(Client) primeste un obiect de tip Client care va fi inserat in tabela, iar aceasta metoda returneaza id-ul la care a fost introdus. Este foarte important sa respectam ordinea si tiparul datelor pe care dorim sa le inseram, acestea fiind surprinse prin atributele clasei Client.

Clasa ProdusDAO

In aceasta clasa sunt incluse metodele prin care se lucreaza cu baza de date. In fiecare metoda se va face conexiunea la baza de date prin care se vor accesa query-urile pentru fiecare metoda pe care vrem sa o accesam in baza de date. In aceasta clasa avem metoda findByName(nume) care primeste ca parametru numele produsului cu ajutorul caruia se cauta in tabela “produs” din baza de date, iar produsul cu numele respectiv va fi returnat ca obiect de clasa Produs. Metoda delete(nume) primeste un nume al unui produs care va fi sters din tabela de produse. Prin adaugarea clauzei WHERE se realizeaza filtrarea rezultatelor.

Metoda findAll() va returna un ArrayList de tipul Produs care contine toate produsele din tabela. Aceasta metoda nu are nevoie de nici o clauza.

Metoda insert(Produs) primeste un obiect de tip Produs care va fi inserat in tabela, iar aceasta metoda returneaza id-ul la care a fost introdus. Este foarte important sa respectam ordinea si tiparul datelor pe care dorim sa le inseram, acestea fiind surprinse prin atributele clasei Produs.

Metoda update(cantitate, nume) se apeleaza in primul rand atunci cand se insereaza in tabela un produs deja existent. Astfel, gasim produsul prin numele transmis ca parametru si ii setam noua cantitate adunand cantitatea care trebuia inserata la cantitatea deja existenta. In al doilea rand, aceasta metoda mai este apelata in momentul in care un client cumpara o anumita cantitate, fiind nevoie sa setam cantitatea ramasa scazand din cantitatea initiala, cea cumparata.

Clasa ComandaDAO

Metoda insert(Comanda) primeste un obiect de tip Comanda care va fi inserata in tabela, iar aceasta metoda returneaza id-ul la care a fost introdusa. Este foarte important sa respectam ordinea si tiparul datelor pe care dorim sa le inseram, acestea fiind surprinse prin atributele clasei Comanda.

Clasa FinalizareDAO

Metoda insert(FinalizareComanda) primeste un obiect de tip FinalizareComanda care va fi inserata in tabela, iar aceasta metoda returneaza id-ul la care a fost introdusa. Este foarte important sa respectam ordinea si tiparul datelor pe care dorim sa le inseram, acestea fiind surprinse prin atributele clasei FinalizareComanda.

**Pachetul Connection**

In acest pachet se realizeaza conexiunea cu baza de date. Metodele de realizare a conexiunii si inchiderile conexiunii , statement-ului si a result set-ului sunt cele de pe bitbucket.

**Pachetul Main**

Acest pachet contine clasa Main care cuprinde doua metode. Metoda readFromFile(String) primeste sub forma unui String fisierul de intrare al programului.Se citeste fiecare linie a fisierului pana la finalul acestuia si este salvata in String-ul s. Aceste String este parsat, adica tot ce e pana la caracterul “:” sau unde nu exista, pana la finalul liniei, reprezinta actiunea asupra bazei de date. Totodata, aceasta parsare se realizeaza si la intalnirea caracterului “,” separand atributele care urmeaza sa fie inserate/ sterse/ comandate. Toate aceste detalii vor fi memorate in String-ul rezultat care vor fi trimise printr-o instanta “comanda” a clasei Prezentare in metoda care stabileste de fiecare date ce operatie urmeaza sa fie realizata.

Metoda main(String[]) in care se seteaza ca parametru al programului fisierul de intare. Aceasta se realizeaza printr-o instanta de tip Main.

**Pachetul Presentation**

Acest pachet contine doar clasa Prezentare care are ca atribute cate o instanta a fiecarei clasa DAO, o lista de tipul ArrayList<FinalizareComanda> in care se va inregistra fiecare comanda ajunsa la finalizare, atributele statice: idClient, idProdus si idComanda prin care vom incrementa si vom seta id-ul fiecarui client inserat, produs inserat si comanda inserata. Mai exista 4 atribute statice care sunt utilizate pentru crearea raporturilor de tip pdf si care vor fi incrementate de fiecare data cand se va face un nou raport.

Metoda comandaSQL(String[]) care preia de fiecare data, asa cum am spus mai sus, rezultatele primite din fisierul de intrare. In String-ul “comanda” sunt inregistrare informatiile primite si apelam metoda compareTo pentru a compara lexicographic String-urile si pentru a vedea ce operatie urmeaza sa facem. In primul rand, daca operatia pe care dorim sa o facem este “Insert client” se ve creea o noua metoda care implementeaza aceasta operatie. La fel se procedeaza si pentru “Delete client”, “Insert product”, “Delete product”, “Order”, “Report client”, “Report product”, “Report order”.

Metoda inserareClient(String[]) are ca parametru datele clientului care urmeaza sa fie inserat: id-ul, numele si adresa. Se instantiaza un client caruia id-ul se incrementeaza de fiecare data, in dateClient[1] se regaseste numele, iar in dateClient[2] se adresa acestuia. Apelam metoda “insert” realizata in clasa ClientDAO si inseram clientul instantiat. In final crestem variabila idClient pentru a putea fi utilizata atunci cand se insereaza un alt client.

In metoda stergereClient(String[]) se apeleaza metoda “delete” realizata in clasa ClientDAO si stergem clientul dupa numele pe care il preluam din dateClient[1].

Metoda inserareProdus(String[]) are ca parametru datele produsului care urmeaza sa fie inserat: id-ul, numele, cantitatea si pretul pe bucata. Se creaza o variabila de tip Produs si se apeleaza metoda “findByName” realizata in clasa ProdusDAO pentru a verifica daca produsul pe care dorim sa il inseram exista sau nu. Daca acesta nu exista, apelam metoda “insert”si inseram produsul cu datele potrivite. Crestem variabila idProdus pentru a putea fi utilizata atunci cand se insereaza un alt produs. Insa daca produsul exista deja, acesta nu este inserat din nou ci se apeleaza metoda “update” explicata mai sus care actualizeaza noua cantitate a produsului.

In metoda stergereProdus(String[]) se apeleaza metoda “delete” realizata in clasa ProdusDAO si stergem produsul dupa numele pe care il preluam din dateProdus[1].

Metoda creareComanda(String[]) realizeaza comanda unui client aflat in baza de date, acest lucru fiind verificat prin apelarea metodei findByName. Totodata, clientul poate comanda doar produse aflate in baza de date, fiind verificat in acelasi mod. Daca exista in depozit cantitatea produsului pe care clientul o doreste, comanda este inserata in baza de date prin apelarea metodei “insert” realizata in clasa ComandaDAO cu datele comenzii corespunzatoare. Totodata, calculam finalizarea acestei comenzi si o inseram in baza de date prin metoda “insert” realizata in clasa FinalizareDAO. Toate comenzile sunt inserate intr-o lista pentru a putea crea mai apoi raportul comenzilor. In continuare, se calculeaza cantitatea vanduta si se actualizeaza prin apelarea metodei “update”. In final, am creat un bon pentru fiecare comanda. Daca in depozit nu exista cantitatea comandata de catre un client, pe bon se va afisa “Nu exista atatea produse in stoc”.

In metoda creareBon se va realiza bonul pe care l-am mentionat mai sus care cuprinde: numele produsului, cantitatea si pretul total.

Metoda reportClient() genereaza raportul clientilor care se afla in baza de date. In metoda addClientHeader am creat titlurile coloanelor: ID, Nume, Adresa.

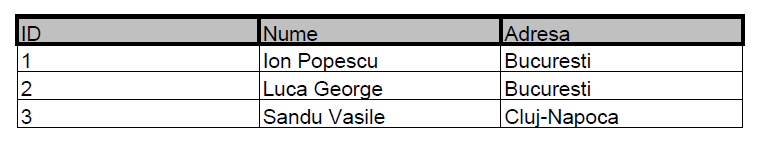
Metoda reportProdus() genereaza raportul produselor care se afla in baza de date. In metoda addProdusHeader am creat titlurile coloanelor: ID Produs, Nume Produs, Cantitate Produs, Pret Bucata.

Metoda reportComanda() genereaza raportul comenzilor care se afla in baza de date. In metoda addComandaHeader am creat titlurile coloanelor: ID Produs, ID Comanda, Cantitate comanda, Pret Total.

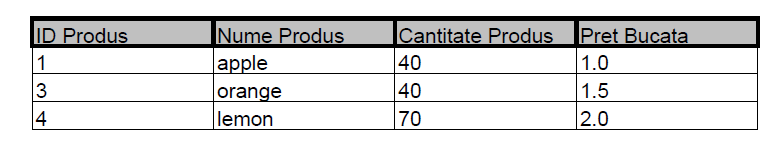
Atunci cand datele nu se insereaza corect se arunca exceptii, la fel si in cazul in care nu se gaseste un anumit fisier sau document.

5. Rezultate

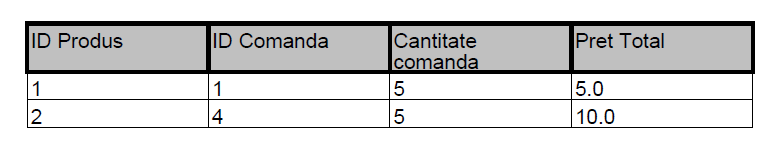
Raport clienti:



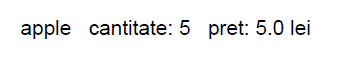
Raport produse:



Raport comenzi:



Bonuri generate:



6.Concluzii, dezvoltari ulterioare

Concluzia principala pe care am dedus-o in urma realizarii acestui proiect este faptul ca pentru gestionarea unei aplicatii precum cea prezentata in paginile anterioare este mult mai usor de creat si gestionat folosind programe si limbaje specializate care au legatura cu stocarea datelor in format de text plan si anume bazele de date relationale

precum MySQL.

O posibilitate de dezvoltare ulterioara ar fi crearea unor constrangeri mai puternice pentru campurile in care se introduc datele. De asemenea ar putea fi adaugate campuri suplimentare pentru a oferi mai multe informatii despre clienti, produse si comenzi.

7. Bibliografie

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

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

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

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