Order Management

Profesor: Student:

Ana-Maria Nanes Cristiana Tintesan

Grupa:30227

Disciplina: Tehici de Programare

2019-2020

# Cuprins

[Cuprins 2](#_Toc39676004)

[1.Analiza problemei 3](#_Toc39676005)

[2.Obiectivul lucrarii 3](#_Toc39676006)

[2.1 Obiectivul principal 3](#_Toc39676007)

[2.2 Obiectivul secundar 4](#_Toc39676008)

[3.Proiectare 5](#_Toc39676009)

[3.1 Decizii de proiectare 5](#_Toc39676010)

[3.4 Implementarea pachetelor 6](#_Toc39676011)

[3.5 Diagrama de clase 7](#_Toc39676012)

[4.Implementare 7](#_Toc39676013)

[4.1 Pachetul connection 7](#_Toc39676014)

[4.2 Pachetul dao 8](#_Toc39676015)

[4.3 Pachetul Model 8](#_Toc39676016)

[4.4 Pachetul bll 9](#_Toc39676017)

[4.5 Pachetul presentation 9](#_Toc39676018)

[4.6 Pachetul start 10](#_Toc39676019)

[5.Rezultate 10](#_Toc39676020)

[6.Concluzii 12](#_Toc39676021)

[7.Bibliografie 12](#_Toc39676022)

# 1.Analiza problemei

Cerinta acestui assignment: Implementati o aplicatie pentru a procesa comenzile unor clienti. Folositi o baza de date pentru a stoca informatiile necesare despre clienti, produse, si comenzi. Mai mult, aplicatia trebuie sa contine minim urmatoarele clase:

* Model classes – reprezinta datele pentru aplicatie.Vom avea o clasa pentru fiecare tabel din baza de date;
* Business Logic classes – reprezinta logica aplicatiei;
* Presentation classes – aici sunt incluse clasele pentru a realiza citirea din fisierul text de intare si pentru a apela metodele necesare pentru a indeplini cerintele.
* Data access classes- contine clasele necesare pentru a accesa baza de date.

Alte clase sau pachete pot fi adaugate pentru a asigura functionarea aplicatiei.

Pentru a putea indeplini cerintele temei, trebuie in primul rand, sa intelgem ce este o baza de date. O baza de date este o colectie de tabele conectate, cu tipuri variate de date, asupra carora pot fi efectuate diferite operatii.Un tabel este format dintr-un antet in care sunt speficicate numele coloanelor din tabel.

# 2.Obiectivul lucrarii

## 2.1 Obiectivul principal

Obiecticul principal al proiectului este de a implementa o aplicatie care lucreaza cu o baza de date, pentru a stoca informatii despre clienti, produse, comenzi si facturi.Pentru aceasta aplicatie sunt implementate urmatoarele operatii:

* Insert – operatie care creaza un nou rand in tabel, care contine informatiile date ca si parametru;
* Delete – operatie care sterge un rand dintr-un tabel speficicat.Funtia delete primeste ca si parametru nume clientului sau al produsului pe care dorim sa il stergem din tabel.
* Update – operatie care este implementata doar pentru tabelul produselor.Prin update actualizam stocul produselor, atunci cand este generate o comanda a unui clinet.
* Find – Operatie care gaseste un client/produs in tabel in funtie de numele acestuia, trasmis ca si parametru.
* Select All – aceasta operatie va returna intregul tabel: antetul si datele corespunzatoare din tabel.

O baza de date valida trebuie sa implementeze corect operatiile descrise mai sus.Acest lucru vom incerca si noi sa realizam cu tabelele: client, product, orders si bill.

## 2.2 Obiectivul secundar

|  |  |  |
| --- | --- | --- |
| Obiectiv secundar | Descriere | Capitol |
| Alegerea structurilor de date | Structurile de date folosite pentru a duce la capat obiectivul principal | 3 |
| Structurarea in pachete si apoi in clase a proiectului | Sunt folosite sase pachete in acest proiect: bll in care sunt clasele:ClientBLL, ProductBLL, OrdersBLL si BillBLL; connection in care este clasa ConnectionFactory, dao in care sunt clasele: AbstractDAO, ClientDAO, ProductDAO, OrderDAO si BillDAO; Model-contine clasele Client, Product, Orders, Bill; presentation in care sunt clasele: ReadFile si Controller si pachetul Start care contine clasa principala Start prin care se ruleaza programul. | 3 |
| Implementarea solutiei | Vor fi descriese, pentru fiecare clasa in parte, campurile si metodele importante | 4 |
| Testare | Vor fi descries cateva scenarii de testare.In functie de un anumit set de date de intrare, se vor prezenta rezultatele rularii programului. | 5 |
| Dezvoltarea de use case-uri si scenarii care pot sa apara | Intr-un sistem de software un “use-case” este o lista de actiuni sau pasi de evenimente care definesc in mod obisnuit interactiunile dintre un rol si un sistem in atingerea unui obiectiv. | 3 |

# 3.Proiectare

# 3.1 Decizii de proiectare

Pentru realizarea proiectului, am luat urmatoarele decizii de implementare:

* Am impartit proiectul in cinci pachete care vor fi detaliate ulterior.Proiectul contine in total 17 clase;
* Am incercat sa folosesc cat mai multe metode simple, care sa nu contina multe linii de cod cu scopul de a oferi o intelegere rapida si usoara a codului;
* Folosirea unor nume sugestive pentru variabile;
* Pastrarea codului cat mai lizibil si usor de urmarit.

3.2 Structuri de date folosite

* In clasa Controller este declarant un ArrayList de clienti, de produse si de comenzi pentru a putea stoca clinetii, produsele si comenzile citite din fiserul text de intrare commands.txt

Private List<Client> c1 = new ArrayList<Client>();

Private List<Product> p1 = new ArrayList<Product>();

Private List<Orders> o1 = new ArrayList<Orders>();

3.3 Use case-uri/Scenarii

Utilizarea programului presupune citirea dintr-un fisier de text a datelor de intrare.Pentru un client se pot citi urmatoarel comenzi: Insert client: Nume Prenume, Oras; Delete client: Nume Prenume si Report client.Pentru un produs se pot citi urmatoarel comenzi din fiser: Insert Product: Nume,Cantitate,Pret si Report product. Pentru comenzi se pot solicita urmatoarele instructiuni: Report order si Order: NumeClient, NumeProdus,Cantitatea comandata. Comenzile de report client/product/order au rolul de a genera cate un fieser pdf cu nume unic in care care sunt afisate tabelele corespunzatoare dupa operatiile de insert/delete/update.Daca sunt introduse doua produse cu acelasi nume, atunci se va insera in tabel un singur produs cu numele respectiv, iar stocul produslui respectiv va fi egal cu suma celor doua cantitati citite pentru produsul cu acelasi nume.De exemplu, daca in fiserul de intrare avem: Insert product: apple, 20, 1 si Insert product:apple,20,1, in tabelul product vom avea un singur rand pentru produsul apple, iar acesta va fi: apple, 40, 1.

## 3.4 Implementarea pachetelor

Proiectul este format din cinci pachete esentiale:

* Pachetul connection – Contine clasa ConnectionFactory care are rolul de a asigura conexiunea cu baza de date;
* Pachetul Model – Contine clasele: Client, Product, Orders si Bill, in care sunt declarate campurile necesare pentru a stoca informatii despre clienti, produse, comenzi si facturi;
* Pachetul dao – Contine clasele AbstractDAO, ClientDAO, ProductDAO, BillDAO si OrderDAO.In aceste clase sunt descrise metodele de inserare, stergere, actulizare si afisare;
* Pachetul bll – Contine clasele: ClientBLL, ProductBLL, OrderBLL si BillBLL, in care sunt apelate metodele din pachetu dao
* Pachetul presentation – Contine clasele ReadFile si Controller, prin care se face cititrea din fisier, formarea fisierelor .pdf si interpretarea comnezilor pentru a realiza operatia ceruta;
* Pachetul start – Contine clasa Start, care este clasa principala a aplicatie deoarece ea ruleaza programul.

## 3.5 Diagrama de clase

A screenshot of a cell phone

Description automatically generated

# 4.Implementare

## 

## 4.1 Pachetul connection

Aceast pachet contine o singura clasa: ConnectionFactory, care are rolul de a crea conexiunea cu baza de date din MySQL. Schema din MySQL are numele database1.Numele de utilizator este root, iar parola Cristiana16.Codul din aceasta clasa a fost preluat din resursele puse la dipozitia studentilor.

## 4.2 Pachetul dao

* Clasa AbstractDAO: contine metodele: createSlectQuery, createSelectAllStatement, createInsertStatement, createDeteleStatement, createUpdateStatement prin care sunt interpretate comenzile de select, select \*, insert, delete, update din sql.Metoda findByName primeste ca si patametru un String si are ca scop cautarea unui client sau produs in tabel, dupa numele acetuia trasmis ca si parametru.Metodele insert, delete, update, selectAll se conecteaza la baza de date pentru a actualiza tabele in fuctie de operatiile efectuate.
* Clasele ClientDAO, ProductDAO, OrderDAO si BillDAO extind clasa AbstractDAO si sunt folosite pentru a implementa operatiile necesare pentru fiecare din cele patru tabele.De exemplu, in clasa ClientDAO sunt regasite metodele insert, delete, selectAll, findByName deoarece pentru tabelul client sunt necesare operatiile de inserare, stergere, afisare si gasirea unui client dupa numele acestuia.

public Client insert(int id, String nume, String city) {

ArrayList<Object> list = new ArrayList<Object>();

list.add(id);

list.add(nume);

list.add(city);

return super.insert(list);

}

Parametrii acestei metode corespund coloanelor din tabelul client: id este id-ul clinetului, care trebuie sa fie un numar unic deoarece nu pot exista doi clienti cu acelasi id; nume este numele clinetului, iar city reprezinta orasul in care locuieste clientul.

## 4.3 Pachetul Model

* Clasa Client: contine variabilele private int id, private String nume, private String city.Denumerile si numarul acestor variabile trebuie sa corespunda cu numarul si denumirile coloanelor create in tabelul client din MySQL. Metodele getId, getNume, getCity, setId, setNume si setCity sunt utilizate pentru a obtine informatii despre client sau pentru a modifica anumite valori, iar metoda toString are rolul de a afisa un client.
* Clasa Product: contine variabilele private int id, private String nume, private int quantity si private float price.Denumerile si numarul acestor variabile trebuie sa corespunda cu numarul si denumirile coloanelor create in tabelul product din MySQL. Metodele getId, getNume, getQuantity, getPrice, setId, setNume, setQuantity si setPrice sunt utilizate pentru a obtine informatii despre produs sau pentru a modifica anumite valori, iar metoda toString are rolul de a afisa un produs.
* Clasa Orders: contine variabilele private int id, private int idc – reprezinta id-ul clinetului care a cerut comanda, idp – este id-ul produsului cerut si quantity indica cantitatea de produs ceruta.
* Clasa Bill: contine variabilele private int id-trebuie sa fie un numar unic pentru fiecare factura noua, private String numeClient-indica numele clientului pentru care s-a facut facutura, private String numeProdus- numele produsului consumat, private int qunaity-cantitatea consumata si private float pretTotal – indica pretul pe care clientul trebuie s ail achite.

## 4.4 Pachetul bll

Pachetul bll include clasele ClientBLL, productBLL, OrderBLL si BillBLL.In aceste clase sunt create metode pentru a realiza operatiile cerute sau pentru a afisa mesaje de eroare in cazul in care aceastea esueaza.Aici sunt apelate metodele din pachetul dao.

## 4.5 Pachetul presentation

* Clasa Controller – este clasa de baza a programului.Pentru a stoca clientii este declarat un arrayList c1.Pentru a gestiona clientii sunt folosite metodele insertClient si deleteClient.Metoda insertClient primeste doi parametrii: str care este de tipul String si reprezinta linia citita din fisier, daca aceasta incepe cu “Insert client:” si idc care este de tipul int si reprezinta id-ul unic al clientului.

public void insertClient(String str, int idc) {

String[] strArray, strArray2;

strArray = str.split(":");

strArray2 = strArray[1].split(",");

ClientBLL clientBLL = new ClientBLL();

try {

clientBLL.insert(idc++, strArray2[0], strArray2[1]);// inseram un nou client

} catch (Exception ex) {

LOGGER.log(Level.INFO, ex.getMessage());

}

c1 = clientBLL.selectAll();

}

Pentru a separa numele clientului si orasul acetuia este folosita functia split care separa stringul initial dupa caracterele : sau , .

Pe acealsi principiu functioneaza si metodele deleteClient, insertProduct, deleteProduct, insertBill si insertOrderUpdateProduct. Aceasta ultima metoda relizeaza atat inserarea unei noi comenzi, cat si actualizarea stocului pentru care s-a efectuat comanda.Stocul este decrementat cu cantitatea de produs citita din fisier.

In clasa Controller, este realizata si crearea fisierelor pdf pentru afisarea tabelelor, a facturilor si a mesajului corespunzator atunci cand se cere o comanda pentru care nu avem stoc suficient.

Pentru realizarea acestor fisiere, in proiect o fost inclus fisierul itextpdf -5.1.0.

* Clasa ReadFile: contine metodele openFile pentru deschiderea fisiersului text de intrare commands.txt.In cazul in care, fisierul nu s-a deschis cu success, este afisat un mesaj de eroare.Metoda readFile citeste cate o linie din fisier in variabila str.In fuctie de continutul lui str, este apelata metoda corespunzatoare din clasa Controller.Metoda closeFile se ocupa de inchiderea fisierului de intrare.

## 4.6 Pachetul start

In pachetul start, este inclusa o singura clasa, clasa Start. In aceasta clasa este declarata o singura metoda, metoda statica main, care primeste ca argument un array de String-uri si se ocupa de rularea programului.

# 

# 5.Rezultate

In urma testarii programumlui, s-a constatat:

* Conexiunea cu baza de date s-a realizat cu success;
* Operatiile de insert, delete pentru tabelul client functioneaza corect;
* Operatiile de insert, delete, update, selectAll pentru tabelul product functioneaza corect;
* Operatiile de insert, selectAll pentru tabelul Orders functioneaza corect
* Operaria de insert pentru tabelul Bill functioneaza corect.

Daca dorim sa rulam de doua ori programul, este esential sa moficam in clasa ReadFile parametrii functiilor de insert si createPDF deoarece aceastea pornesc la fiecare rulare de la valoarea initiala 1, iar tabelele nu ne permit sa avem doua obiecte din tabel cu acelasi id.Campul id este cheie primara pentru toate tabelele aplicatiei.

Exemplu de date de intrare:

Insert client:Ion Popescu,Bucuresti

Insert client:Luca George,Bucuresti

Report client

Insert client:Sandu Vasile,Cluj-Napoca

Report client

Delete client:Ion Popescu,Bucuresti

Report client

Insert product:apple,20,1

Insert product:peach,50,2

Insert product:apple,20,1

Report product

Delete Product:peach

Insert product:orange,40,1.5

Insert product:lemon,70,2

Report product

Order:Luca George,apple,5

Order:Luca George,lemon,5

Order:Sandu Vasile,apple,100

Report client

Report order

Report product

Rezultatele rularii.Cum vor arata cele pantru tabele dupa executarea acestor comenzi:

* Tabelul client:

|  |  |  |
| --- | --- | --- |
| id | nume | city |
| 2 | Luca George | Bucuresti |
| 3 | Sandu Vasile | Cluj-Napoca |

* Tabelul product:

|  |  |  |  |
| --- | --- | --- | --- |
| id | nume | quantity | price |
| 1 | apple | 35 | 1 |
| 4 | orange | 40 | 1.5 |
| 5 | lemon | 65 | 2 |

* Tabelul orders:

|  |  |  |  |
| --- | --- | --- | --- |
| id | idc | idp | quantity |
| 1 | 2 | 1 | 5 |
| 2 | 2 | 5 | 5 |

* Tabelul bill:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | numeClient | numeProdus | cantitate | pretTotal |
| 1 | Luca George | Apple | 5 | 5 |
| 2 | Luca George | lemon | 5 | 10 |

# 6.Concluzii

In urma realizarii acestui assignment, am reusit sa inteleg cum se face legatura dintre sql si java, cum putem insera, sterge, selecta, actualiza un tabel din sql folosind un limbaj de programare.Tema nu a fost usor de realizat, a necesitat multe ore de munca.

Posibilitati ulterioare de dezvoltare: asigurarea genarii unui id unic pentru fiecare client, produs, comanda sau factura, atunci cand programul este rulat de mai multe ori.

# 

# 7.Bibliografie

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

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