FUNDAMENTAL PROGRAMMING TECHNIQUE

ASSIGNMENT 3

ORDER MANAGEMENT

Diaconu Andrei
Grupa 30227

CUPRINS

Introducere ….….....3
Analiza problemei ….....4
Proiectare si Implementare
Package BLL.....6
Package Validators ….....6
Package Model …….....7
Package Connection ….....7
Package Presentation 8
Pacakge DAO.....11
Concluzii.....12
Bibliografie.....13

1 Introducere

Bazele de date , cunoscute si ca BD, sunt modalitatea de stocare a unor date cu posibilitati de gasirea rapida a informatiilor. Cand este vorba de un cumul mic de informatii nu pare o problema complicate , insa cand ne referim la milioane de informatii ale milioanelor de utilizatori din intreaga lume si gestionarea lor intr-un mod propice , problema nu se mai mentine la acelasi nivel de simplitate.

De obicei aceste baze de date se salveaza intr-unul sau mai multe fisiere care urmeaza a fi controlate de un sistem de gestiune. Cel mai renumit tip de baze de date , de altfel cel care l-am si abordat in proiectul acesta este cel relational.

Contin pe langa utilizatori , grupuri de utlizatori si id-urile acestora prin care se identifica procedure sau triggere si modalitati de Securitate . Acestea din urma sunt utilizate de exemeplu in cazul unor tranzactii sau salvarea unor parole , nepermitand stocarea lor in modul plane.

Ca sistem de gestiune vom folosi MySql. Desi se obisnuieste folosirea acestuia impreuna cu PHP, acesta are totusi o plaja larga de limbaje de programare care permit crearea unor API-uri. Bineinteles aici vom folosi Java ,MySql Server prin care ne vom crea prorpiul server localhost dar si WorkBench-ul in care ne vom crea tabelele fara a fi populate.

Obiectivul principal al temei este implementarea unei aplicatii Java, interactionand cu utilizatorul prin GUI , care permite procesare clinetilor, produselor si comenziilor unui depozit.Bazele de date relationale stocheaza aceste 3. Folosindu-ne de pachete Layerd Architecture si de tehnica reflexiei. Pentru o descriere mai ampla a metodelor folosite din interiorul fiecarei clase , vom folosi Javadoc comments.

Asadar pentru dezbaterea intregului management am create o interfata grafica (Presentation->GUI) . In acelasi pachet se afla si ControllerGUI care continue alte innerClasses scopul acestora este gestionarea fiecarui buton din interfata. Pentru fiecare tabel am creat cu acelasi field-uri clase in pachetul Model. In pachetul DAO m-am ocupat de crearea query-urilor in mod abstract extins pentru fiecare clasa in parte, iar in BLL apelam methoda ce continue query-ul , de la caz la caz. Start contine pornirea intregii simulari. De asemenea in BLL mai exista o serie de

validators, prin care trec elementele inainte de a fi inserate in tabelul corespunzaotr selectat de utilizator.

2 *Analiza problemei*

Cand ma gandesc la analiza problemei, ma gandesc la idea in mare , la schita problemei.Ea trebuie sa functioneze pe niste inputuri abstracte , asadar prin intermediul acestor gasim comportamentul problemei si rezolvarea ar trebui sa decurga de la sine.

Odata regasit firul principal al managementului problemei trebuie sa avem in vedere ca comunicarea cu utilizatorul trebuie sa fie un simpla a aplicatiei, fiind pusa in functiune in cazuri imporante . Asadar nu am dori ca un client sa faca o comanda cu mai multe produse decat ce avem in stock , sau fara sa-I facem o facture pe produsul cumparat . Asadar tot ce trebuie sa faca utilizatorul este de a introduce in acele gap-uri exact ce scire in descrierea din stanga . Daca sunt inculcate anumite reguli se va afisa mesaj de eroare corespunzator in consola programului . Mesaj care continue detalii explicite ale dificultatii tocmai intalnite.

Asadar aplicatie trebuie permiterea extragerii unui client sau a unui produs precum si updatarea datelor despre acestea pe baza unui id.Totodata se poate sterge un intreg client sau prods. Vizualizarea este si ea permisa.In ceea ce priveste comenziile , se pot initia atribuind un id de produs la un id de client daca acestea exista si cantitatea produsului dorit de client este mai mica sau egala decat stock-ul current al produsului comanda se adauaga cu success in tabel.

Scenarii:

- Adaugare : Utilizatorul introduce informatiile necesare (Nume , Adresa, Email) in cazul clientului si (Nume , Pret , Stock) in cazul produsului , id-ul este autogenerate de MySql. In urma apasarii butonului OK , in consola trebuie sa apara generarea datelor sau eroare in caz negativ , si totodata tabelul sql updatat

- Stergere : Utilizatorul introduce un simplu id. Interfata isi determina singura din ce parte a fost cheamata metoda de stergere. Si sterge intreg row-ul in care se gaseste id-ul tocmai introdus de utilizator.

- Updatare: Utilizatorul adauga in primul rand id-ul , prin care se anunta astfel ce produs sau ce client urmeaza a fi updatat . I se vor revizui in tabel field-urile completate . Cele lasate libere nu vor fi modificate deloc

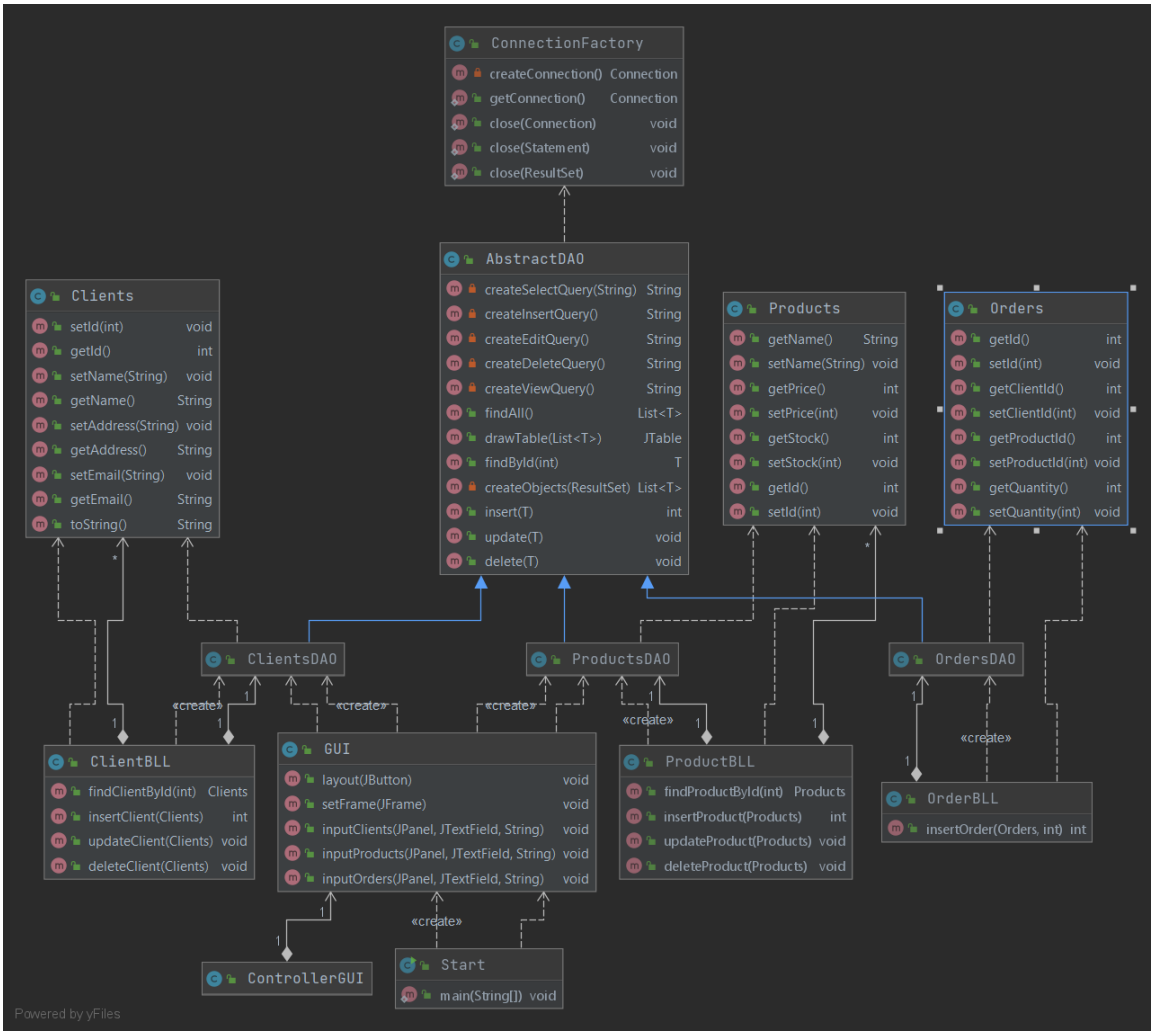
- Adaugare Comanda: Se face pe baza id-urilor asadar trebuie scris atat id-ul clientului cat si id-ul produsului. Se trece mai departe doar in cazul in care acestea exista . Apoi se verifica daca ce s-a introdus in campul de cantitate este convenabil avand in vedere stock-ul produsului respective. Daca s-a adaugat cu success se va putea vedea atat in consola cat si in tabelul din MySql.

Scenariile negative despre care tot am vorbit contin erori ale utilzatorilor care trebuie gestionate cumva. In cazul acestui proiect ii avertizam prin mesaje de eroare in consola. Daca se incearca gasirea unui client/produs care nu exista avem astfel de mesaj care specifica ca nu s-a gasit in tabel. De asemnea daca la introducerea unui Client numele acestuia continue cifre este anuntat ca nu este un format correct. Daca mail-ul acestuia nu are format-ul correct cu @... de asemenea este avertizat. La Produs validoarele sunt folosite pentru a receptiona informatii legate de pret si stock deoarece nu se doreste introducerea unui stock negativ sau pret negative “Must pe positive” este mesajul afisat in consola. “Under stock!” este mesajul la crearea unei comenzi noi in care utlizatorul introduce mai multe bucati dintr-un produs care nu se afla de atatea ori pe stock.

3 Proiectare si Implementare

In primul si primul rand am inceput prin proiectarea tabelelor in MySql .Acestea fiind Client(id, nume, adresa, email) , Produs(id, nume, pret ,stock) si Orders(id, clientId, produsId, quantity). In toate cele 3 tabele id este primary key. Iar pentru aceasta am si setat functia de auto-increment.

Odata sigur ca tabele sunt implementate correct am trecut la partea de cod Java. Primul pas aici a fost adaugarea librariei externe care permite conectarea la baza de date mysql-connector-java:8.0.24.jar.



PACHETUL BLL

SUBPACHETUL VALIDATORS

Continue o interfata care contine antetul functiei validate . Am folosit metoda interfeti si implemtarii acestuia pentru fiecare tip de validare deoarce prezinta cea mai mare parte de clean code.

Asadar EmailValidator suprascrie metoda din validate prin Regex verifica daca clinetul introdus are formatul mailului correct. In acelasi prinicipiu si numele

care verifica tot prin regex daca nu cumva prin numele clientului se afla vreo cifra , lucru care ar conduce catre o eroare

In cee ace priveste produsele si tabelul Products avem un singur validator care verifica 2 campuri.Stock si price care trebuie sa fie positive, acesta facand mai mutl referire la un typo al utilizatorului si ar baga un minus din greseala in fata si ar fi foare rau pentru depozitul gestionat.

CLASELE BLL

ClientBLL in constructor initializeaza lista de validatori si adauga validatori pentru clasa respective. In acest caz avem Email si Nume . Tot odata creaza o instanta a clasei ClientDAO. Obiectul clientDAO va apela toate functiile, deoarce el continue query-urile. Aici se afla pasii la care se apleaza functiile mari descrise in DAO.

OrderBLL continue o singur functie de insert care verifica daca cantitatea din comanda este in stock. In caz negative se va arunca o eroare despre cum am vorbit la pasii anteriori.

ProductBLL in aceeasi masura cu ClientBLL insa are un singur validator care se refera la pret si stock.

PACHETUL CONNECTION

CLASA CONNECTIONFACTORY

Ca final Stringuri stabilim server-ul nostru la care urmeaza sa ne conectam impreuna cu id-ul si parola . In constructor ne folosim de Driver, iar in createConnection practic incepem conexinea la server-ul nostrum localhost utilizand cele specificate.In caz de eroare primim un mesaj corespunzator.

getConnection ne va folosi sa luam direct conectarea stabilita in clasele ce urmeaza. In ceea ce priveste close-urile avem pentru fiecare field in parte . Trebuie sa inchidem atat conexiunea create cat si query-ul si rezulatul acestuia.

PACHEUL MODEL

Pachetul Model respecta pentru fiecare clasa aceeasi structura : Continue 3 constructori unu fara parametrii si 2 cu parametrii. Eu insa in acest proiect am

folosit doar cel cu 3 constructori deoarce am decis ca id-ul pentru fiecare tabel sa fie auto incrementat.

Aceasta decizie am luat-o gandindu-ma ca aceasta aplicatie ar putea fi imprumutata de un manager al unui depozit care nu ar avea timp sa caute de feicare data la ce client a ajuns. Treaba aceasta revenindu-I direct mediului de lucru , asadar aducem aplicatia la un nivel mai simplist.

CLASA CLIENTS

Id : id-ul prin care se identifica si va fi recunoscut si in celalte clase/ tabele

Nume: Numele care se va scrie pe facture in momentul achizitionarii unui produs

Email: Email-ul la care i se vor trimite informatii referitoare la produs(de exemplu sa-l anunte cand a ajuns in stock un produs pe care-l voia el)

Address: Adresa la care se va livra coletul

CLASA PRODUCTS

Id:id-ul prin care se identifica si va fi recunoscut si in celelalte clase/tabele

Nume: numele produsului cu tot cu detalii

Pret: pretul corespunzaztor

Stock: cantitatea din stock , care uremaza a fi decrementata de fiecare data cand un client comanda acel produs

CLASA ORDERS

Id:id-ul prin care se identifica si va fi recunoscut si in celalate clase/tabele

clientId: clientul care face comanda

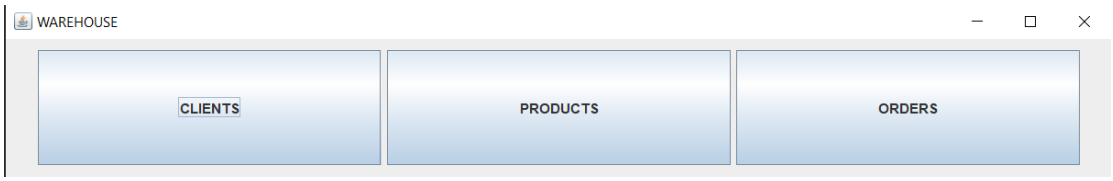
productid: produsul care-l allege clientul respectiv

quantity:cate bucati doreste clientul respectiv

PACHETUL PRESENTATION

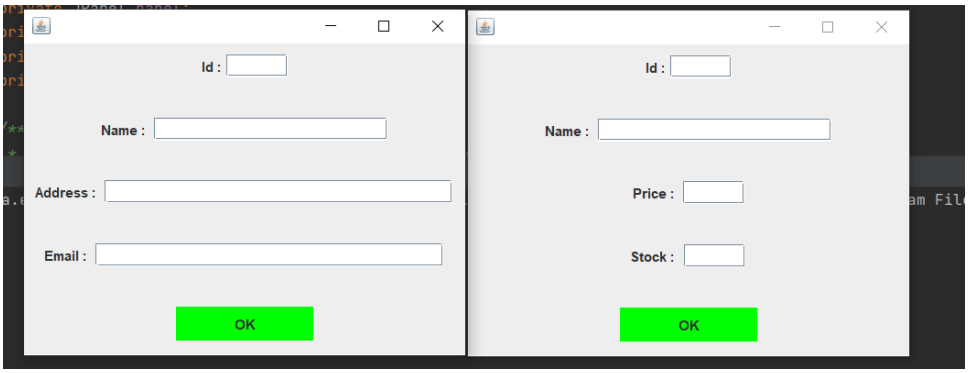
CLASA GUI

Beneficiaza de 2 constructori. Unul fara parametrii care este apelata in Clasa Start. Ea creaza Frame-ul principal din care se vor putea allege unul din cele 3 tabele cu care se va lucra



Al 2-lea constructor primeste ca parametru un String. Acest constructor va fi apelat din Controller. Am folosit un case pentru manipularea acestui String table. In mare parte frame-urile create sunt asemanaotare insa difera prin informatiile oferite inainte de TextField-uri si numarul lor.

Daca table va avea insert/add in String inseamna ca vom avea o interfata care prelucreza toate campurile (si sunt obligatorii) in afara de id deoarce cum am mai spus id-ul este autoincrementat in mysql. Update/Edit primeste si un TextField de id , care este essential deoarce pe baza lui incepe cautarea in tabele . Daca nu este gasit elemental cu id-ul introdus vom primi o eroare. In cazul in care este totusi gasit se vor introduce in text-fielduri doar campurile care se doresc a fi modificate. Daca unele TextFielduri raman goale acestea nu se updateaza asa in tabele (null) ci nu-si schimba valoarea.



Referitor la View , ambele tabele vor fi afisate conform ultimelor actualizari. Adica daca eu doresc sa adaug acum un client nou si nu-I dau refresh in sql , el tot va aparea in JTable-ul meu.Tabelul este cules din clients/products DAO apeland metoda drawTable pentru toate produsele/clientii. Acestea sunt determinate cu findAll care apeleaza la randul ei createObjects. Vom vorbi in cele ce urmeaza despre aceasta. Tabelel inainte sa le adaug intr-un frame le adaug in JScrollPane

care ne permite vizualizarea interegului tabel daca este cazul. Pe exemplul prezentat nu era nevoie.

id	name	address	email
1	Andu	Cluj Napoca 37	andu@gmail.com
2	Andrew	Cluj Napoca Aug-Bu...	john@gmail.com
3	Mike	Chiago	mike@yahoo.com
8	Alex	Bucuresti	Alex@gmail.com
10	TheBoom	Petrosani 17	theboom@gmail.com
13	Matei	Petrila	Matei@gmail.com

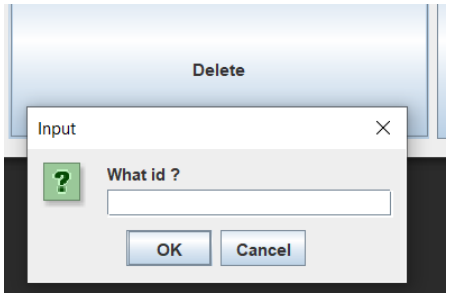
Vreau sa amintesc ca pentru fiecare frame in partea acestea la apasarea butonului X sau OK in cazul inserarii/updatarii nu se inchide toate aplicatia . Nu avem un EXIT_ON_CLOSE, decat la mainframe . Astfel dupa ce realizam orice fel de activitate aplica nu ne obliga sa o repornim ci putem continua activitatile. Pentru aceasta am folosit getDefaultOperation pe frame-ul current asupra caruia lucram.

CLASA CONTROLLER

Contine static 2 GUI objects : unul referindu-se la interfata principala cand apelam clients , products sau orders, si detele pentru primele 2 tabele. Pentru toate restul se foloseste al doilea GUI object deoarce acesta trebuie sa creeze o alta interfata grafica a userului.

Contine 11 Inner Classes care gestioneaza toate cele 11 butoane ale interfetelor. Pentru fiecare buton apasat stabilim prin String table locul de unde am apsat butonul. Avand la fel interfetele pentru clients si products , mai in detaliu pentru add , delete si insert se poate sa apelam products apoi clients dar sa vrem sa vedem tabelul din products. Trebuie de fiecare data sa se stie de unde s-a dat click pe fiecre buton.

Pentru delete nu a mai fost nevoie de o noua interfata grafica . A fost de ajuns un pop-up prin care intreaba utilizatorul ce id doreste sa stearga. Odata apsat butonul ok daca exista acel id stergerea se va efectua cu success.



In cadrul acestei clase as vrea sa mai aduc in vedere ultima innerClass si anume OK5Controller. Care permite adugarea unei comenzi noi. Aici se intampla toate acele verificari: de client , de produs si apeleaza aceea care conti cantitatea-stock.

Sub actiunea unor try-catch-uri am folosit printarea cu ajutorul FileWriter-ului si al PrintWriter-ului factuarea comenzii clinetului. Acesta se va regasi in fisierul sursa al progamului sub numele de bill.txt

PACHETUL DAO

CLASA ABSTRACTDAO

Cea mai importanta clasa a acestui proiect , de aici desfasurandu-se toate activitatiile de la conectare si verificare pana la scriere si updatare a tabelului original (MySql).

Methodele create___Query() sunt create folosind StringBuildere si returneaza un String. In esenta acesta creeaza query-ul , dupa cum sugereaza si numele , care urmeaza a fi apelate. Insert-ul , care trebuie sa ocoleasca fieldul id atunci cand creaza VALORILE (?,?,?) .Ca sa oprim “,” si sa punem paranteza trebe sa contorizam valorile. La edit se respecta acelasi principiu insa trebuie inlocuit la sfarsit cu un WHERE id = .Puteam folosi ca si in exemplul oferit cu un String field insa noi mereu vom cauta dupa id fiind singurul primary key.

Asadar odata create query-urile ne orientam atentia catre functii. FindAll() apeleaza functia deja generata createObjects pe baza query-ului de viewAllTable . Stie pe ce tabel sa apeleze acest query deoarce acesta este receptionat cu type. Paramterul createObjects este resultSet , rezultat care reprezinta outputul query-ului executat din statement, imediat dupa ce conexiunea a fost facuta.

Insert-ul obtine conexiunea si pregateste statementul. Pentru fiecare field in parte execeptand id-ul dupa cum am sugerat mai sus. Odata setati toti paramterii se poate executa cu executeUpdate(). Daca rs are next s-a executat cu succea iar outpuul acestei methode va fi id-ul tocmai generat de sql. In caz de eroare se va returna -1 pentru ca insertedIndex ramane neschimbat.

Edit-ul si delete-ul functioneaza cam pe acelasi principiu.

O methoda mai deosebita este drawTable care in data retine pentru fiecare camp in parte informatiile necesare creand in ProprietyDescriptor si o invocare de methoda cu proprietatile de abia extrase pentru acel field. In coloane se introduce numele de coloane din tabele . Cand se creeaza JTable-ul constuctorul va primi aceasta matrice si acest array.Putem vorbi si de eficientizare aici deoarce numarul de coloane ale tabelului nu este alocat aleator , ci calculate cu type.getDeclaredFields().length;

4 CONCLUZII

In finalul acestei documentatii as vrea sa amintesc faptul ca acest proiect mi-a dat prilejul sa-mi exersez cunostiile de semestrul trecut referitoare la baze de date. Totodata am invatat cum functioneaza reflexia si am inteles mai bine ce inseamna abstarctizarea unei clase si cat de mult ne scuteste din munca daca este implemtatat correct

Referitor la alte imbunatatiri, intr-adevar se poate lucra pe partea de aspect si totodata la intocmirea unei comenzi in loc sa introduce id-ul clientului si al produsului sa poti selecta dintr-un tabel vizibil

5 BIBLIOGRAFIE

- https://ro.wikipedia.org/wiki/Baz%C4%83_de_date

  24.04.2021 14:40

- https://www.oracle.com/ro/technical-resources/articles/java/javadoc-tool.html

  22.04.2021 13:10

- https://stackoverflow.com/questions/1959467/no-digits-java-regex-pattern

  21.04.2021 17:15