DOCUMENTATIE

Tema numarul 3

Popovici Eusebiu-Ionut

Grupa 30224

CUPRINS

1. Obiectivul temei

Obiectivul principal: Obiectivul principal al acestei teme este crearea si implementarea unei aplicatii de gestionare a unui magazin cu clienti,produse si comenzi,cu o interfata grafica interactiva prin intermediul caruia utilizatorul poate face operatii de tipul CRUD pe clienti si produse si poate realiza o comanda selectand un client si un produs.

Obiectivele secundare: Pentru indeplinirea obiectivului principal au fost urmati urmatorii pasi: - Analiza problemei,modelarea,scenarii,cazuri de utilizare

-Proiectarea aplicatiei de gestionare a clientelor,produselor si a comenzilor

-Implementarea propiu-zisa a aplicatiei si legarea acesteia de o baza de date

-Testarea aplicatiei

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

Cerinte funcitonale:

- Aplicația trebuie să permită utilizatorului să aleagă operațiunea dorită: Operații pe tabela de clienți, operații pe tabela de produse și operații pe tabela de comenzi

- Aplicația trebuie să permită utilizatorului să adauge un client nou.

- Aplicația trebuie să permită utilizatorului să editeze un client existent.

- Aplicația trebuie să permită utilizatorului să șteargă un client existent.

- Aplicația trebuie să permită utilizatorului să adauge un produs nou.

- Aplicația trebuie să permită utilizatorului să editeze un produs existent.

- Aplicația trebuie să permită utilizatorului să șteargă un produs existent.

- Aplicația trebuie să permită utilizatorului să creeze o comandă prin selectarea unui client și a unui produs dintr-un tabel de clienți și un tabel de produse și introducerea unei cantități valide, iar apoi inserarea acestei comenzi în tabela de comenzi.

Cerinte non-functionale:

- Aplicația trebuie să fie intuitivă și ușor de utilizat.

-Aplicatia trebuie sa aiba interfetele grafice corespunzatoare pentru a executa operatiile cerute.

Cazuri de utilizare:

-Utilizatorul alege din interfata principala ce operatie si-ar dori sa execute(pe clienti,produse sau comenzi.

-Pentru a adauga un client sau produs nou trebuie inserate valori corespunzatoare in toate text-field-urile din interfetele secundare,iar apoi se apasa butonul “ADD”.

-Pentru a edita un client sau un produs trebuie inserate valori corespunzatoare in toate text-field-urile din interfetele secundare,iar apoi se apasa butonul “EDIT”.

-Pentru a sterge un produs sau un client trebuie sa scriem doar in text-field-ul corespunzator ID-ului,dupa care se va apasa butonul de “DELETE”.

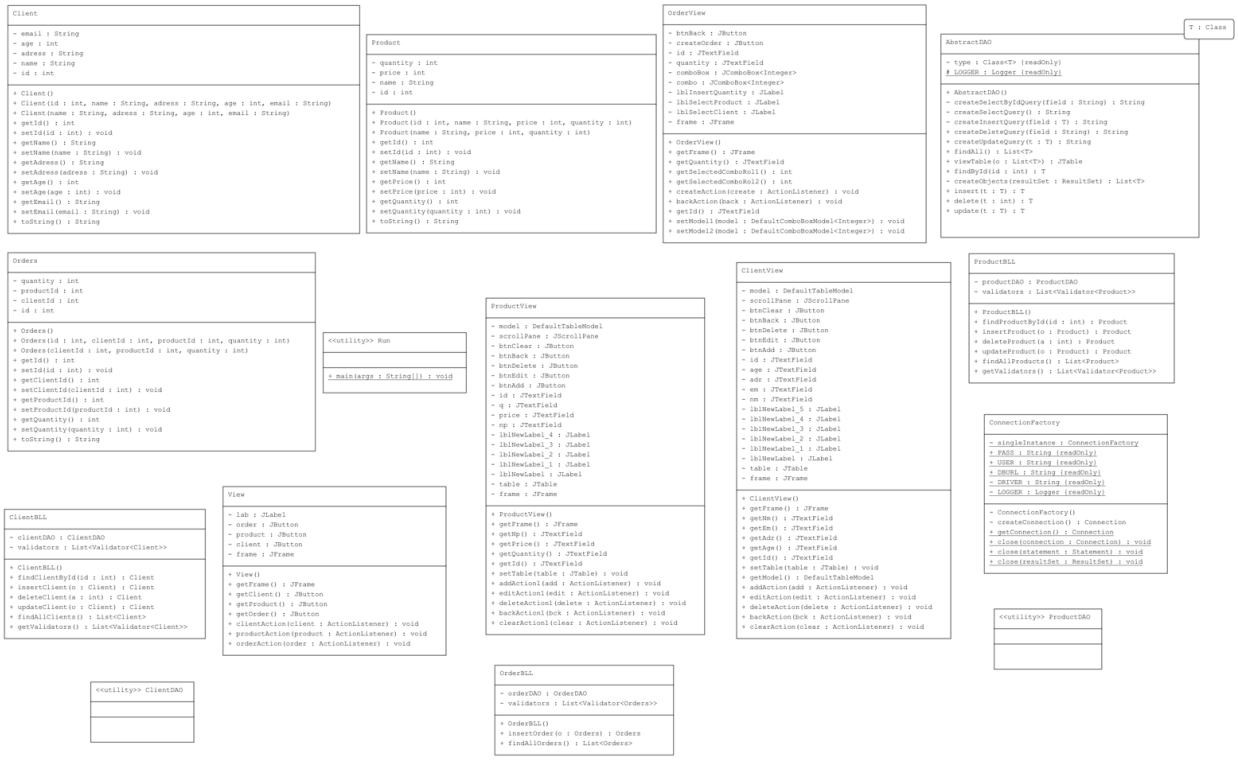
-Baza de date corespunzătoare clienților/produselor se actualizează în funcție de operația realizată la punctul anterior si va afisata in ambele interfete.

-Pentru a crea o comanda utilizatorul trebuie sa introduci cantitatea si id-ul comenzii pe care vrei sa o creezi,si sa selectezi din combo-box-urile din interfata id-ul clientului si produsului care vrei sa faca comanda,dupa care se va apasa butonul de “CREATE ORDER”.

3.Proiectare

Proiectarea OOP a aplicatie va aborda principiile programarii orientate pe obiecte,totul fiind incapsulat in clase,care se vor afla in diferite pachete pentru ca proiectul sa fie usor de citit si de inteles(Pachetele: GUI, Logical, Model,DAO).

Diagrama UML va fi folosita pentru a arata relatiile dintre clase si pachete.



Algoritmii utilizati: In cadrul acestui proiect nu s-au folosit algoritmi special,in schimb s-a folosit comunicarea cu baza de date prin intermediul reflection-ului,o clasa in care avem cate o singura metoda de add,edit,delete si viewTable prin care putem face aceste operatii in toate tabelele noastre din baza de date,apelandu-se fiecare operatie prin intermediul claselor BLL a fiecarei tabele.

4.Implementare

In fiecare pachet se afla cel putin o clasa.

In pachetul Connection avem clasa ConnectionFactory: Clasa “ConnectionFactory” face legătura cu baza de date și conține o metodă care deschide baza de date, o metoda care o închide, o metodă pentru închiderea unei operații și una pentru închiderea rezultatului obținut în urma unei operații.

In pachetul Model avem clasele Client,Orders si Product: Clasa “Client” e corespunzătoare tabelei clienților care conține înregistrări ce reprezintă de fapt obiecte de această clasa. Cu aceasta clasă se mapează tabelul clienților din interfață. Clasa “Product” e corespunzătoare tabelei produselor care conține înregistrări ce reprezintă de fapt obiecte de această clasa. Cu aceasta clasă se mapează tabelul clienților din interfață. Clasa “Orders” e corespunzătoare tabelei comenzilor care conține înregistrări ce reprezintă de fapt obiecte de această clasa.

In pachetul Logical avem clasele ClientBLL,OrdersBLL,ProductBLL: Clasa "ClientBLL" implemetează efectiv operațiile de insert/update/delete/find asupra tabelei corespunzătoare clienților. Clasa "ProductBLL" implemetează efectiv operațiile de insert/update/delete/find asupra tabelei corespunzătoare produselor. Clasa "OrderBLL" implemetează efectiv operațiile de insert/delete/find asupra tabelei corespunzătoare comenzilor. De asemenea, conține și o metodă care creează un pdf ce reprezintă factura pentru o anumită comandă plasată, aceasta fiind apelată în metoda "makeOrder" care realizează de fapt inserarea în tabelă.

In pachetul DAO avem clasele ClientDAO,OrdersDAO,ProductDAO si AbstractDAO: Clasa "AbstractDAO" conține metode care aplică tehnica reflexiei pentru manipularea eficienta a tabelelor din baza de date. Conține metode care creează interogări pe tabele pentru insert/update/delete/find și metode care aplică efectiv aceste interogări: o update() folosește createUpdateQuery() -> realizează actualizarea unei înregistrară din tabel o insert() folosește createInsertQuery() -> adaugă obiect nou în tabel o findall() folosește createInsertQuery() -> returnează o listă cu toate obiectele din tabel o findById() folosește createSelectQuery(field) cu field = "id" -> găsește înregistrarea din tabel care are un anumit id o deleteById folosește createDeleteQuery(field) cu field = "id" -> sterge o înregistrare din tabel care are id-ul specificat. Pe lângă acestea mai există o metodă care creează prin reflexie un tabel dintr-o listă de obiecte și o metodă care creează un obiect din ceea ce returnează interogarea asupra tabelului. Clasele "ClientDAO", "ProductDAO", "OrderDAO", "OrderTableDAO" moștenesc clasa "AbstractDAO". Primele 3 folosesc metodele nemodificate din clasa părinte, în timp ce clasa "OrderTableDAO" suprascrie metoda „findAll()" folosind interogarea creată de metoda "joinHeader()".În plus față de clasa părinte, aceasta mai conține metoda "ordersDates" care utilizează interogarea creată de metoda " createSelectJoin()" și returnează un nou obiect cu datele din înregistrarea ce a rezultat în urma selecției făcute. Această clasă nu are un tabel corespondent în baza de date, ea fiind creată pentru a ușura afișarea tabelului de comenzi care conține doar id-uri.

In pachetul GUI avem clasele View,ProductView,OrderView,ClientView si Controller: In toate aceste clase vedem interfetele corespunzatoare care contin butoane pentru operatii si text-field-uri pentru adaugarea de date,in timp ce in clasa Controller,clasa cea mai complexa din aceasta interfata,avem executarea fiecarei operatie atunci cand butoanele din interfete sunt apasate.Tot in clasa Controller se leaga interfata grafica cu partea de back-end implementata in celelalte pachete.

5.Rezultate

In acest proiect nu au fost folosite testarile unitare pentru a obtine rezultate.

6.Concluzii

Realizând această temă am învățat o tehnică foarte utilă și des folosită în programarea orientata pe obiect, și anume reflexia. Utilizând această tehnică este ușor de realizat o clasa abstractă care poate fi moștenită, evitându-se astfel scrierea de mai multe ori al aceluiași cod care folosește clase diferite.

Posibile modalidati de dezvoltari ulterioare: Am putea face ca la fiecare comanda sa poata fi adaugate mai multe produse,nu doar unul si sa adaugam campuri in plus pentru clientii si produsele din magazinul nostru. Totodata am putea implementa o noua baza de date pentru un orar care sa ne arate intervalele orare in care magazinul este deschis si in care se pot face comenzi.

7.Bibliografie

1. https://jenkov.com/tutorials/java-reflection/index.html
2. https://dzone.com/articles/layers-standard-enterprise
3. https://mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/