**Documentatie Tema 3**

Hitu Octavian

Grupa:30226

Profesor Laborator : Moldovan Dorin Vasile

**Cuprins**

1.Obietivul temei

2. Analiza problemei, modelared, scenario, cazzuri de utilizare

2.1. Analiza Problemei

2.2. Modelare

2.3. Cazuri de utilizare

3. Proiectare

3.1. Diagrama UML

3.2. Clasa ConnectionFactory

3.3. Clasa Client

3.4. Clasa Product

3.5. Clasa Order

3.6. Clasa AbstarctDao

3.7 Clasele ClientDao, ProductDao, OrderDao

3.8. Clasele CLientbll, Productbll, Orderbll

3.9. Clasa View

3.10. Clasa Controller

4. Rezultate

5. Concluzii

1.Obietivul temei

Proiectul pe care l-am implementat reda intr-o forma cat mai buna un depozit, acestea gasindu-se peste tot in lume. Acest proiect are la baza o baza de data in mysql peste care au fost implementate anumite functii folosind anumite tehnicii precum reflexia. In acest poriect se ragasesc 3 mari tabele destinate clientilor,produselor si comenzilor pe care s-au implementat functii de adaugare, editare, stergere si de afisare a tabelelor. In capitolele ce urmeaza voi explica cum se pot prelucra datelele din tabele si cum am implementat proiectul. Pentru a intelege toate aspectele , noi trebuie sa intelegem mai intai ce inseamna conceptul de reflexive, acesta fiind folosit in program.

Reflexia este folosita pentru examinarea sau modificarea comportamentului metodelor , a claselor sau a interfetelor grafice in timpul rularii.

2. Analiza problemei, modelared, scenario, cazzuri de utilizare

2.1. Analiza Problemei

Daca este sa ne gandim intr-o situatie reala, problema proiectului nostru se regaseste peste tot in viata noastra, de exemplu in magazine sau in orice depozit care face vanzari sau lucreaza cu anumite produse care se vor lista spre vanzare. De aceea fiecare institutie de genul are nevoie de o baza de date pentru a putea stoca clientii,produsele si comenzile de un program si de o interfata pentru a putea prelucra usor acele date. Functiile destinate prelucrarii datelor se fac dintr-o interfata care prin care poti introduce elemente, edita si sterge.

2.2. Modelare

Modelarea aplicatiei se bazeaza pe conceptele programarii orientate pe obiecte, acestea fiind:

Clasele- prin clasa intelegem o forma a unui obiect deoarece aceasta reprezinta o descriere generalizata a acestuia. Clasa defineste toate atributele pe care un obiect le are ,iar metodele ce fac parte din acea clasa definesc functionalitatea acelui obiect;

Obiect- un obiect reprezinta o componenta software care incorporeaza atribute si operatii . Operatiile se pot efectua asupra atributelor .

Incapsularea- aceasta are rolul de a uni informatiile importante ale unui obiect. Ea totodata restrictioneza accesul la date si metode din exterior.

Mostenirea- din punct de vedere ierarhic aceasta permite claselor similare sa se suprapuna. Datorita ei clasele inferioare pot folosi metode din clasele superioare.

Polimorfismul- prin definitie aceasta reprezinta abilitate de a lua mai multe forme .Prin polimorfism aceeasi metoda ce este folosita intr-o superclasa poate fi suprascrisa fiind reutilizara in subclase ,ele avand o functionalitate diferita

2.3. Cazuri de utilizare

Aplicatia dispune de o interfata care permite introducerea, editarea si stergerea unor date din tabela prin introducerea caracteristicilor de la tastatura .

In urmatoarele randuri voi prezenta parametrii de intrare din interfata grafica prentru fiecare tabel fiecare field uri reprezentand coloanele acelei tabele:

Client:

* ID- acesta este un numar unic pentru fiecare client
* First name- reprezinta numele de familie al clientului
* Last name – reprezinta prenumele clientului
* Adress- reprezinta adresa clientului

Product:

* ID- reprezinta un numar unic pentru fiecare produs
* Product name-reprezinta numele produsului
* Price- reprezinta pretul prosului per bucta
* Quantity – reprezinta cantitatea pe care depozitul o are pe stoc pentru acel produs

Order:

* ID- acesta este un numar unic pentru fiecare comanda
* Client id- reprezinta un foreign key care are ca primary key id-ul din tabelul client
* Product id- reprezinta un foreign key care are ca primary key id-ul din tabela product
* Quantity order – reprezinta numarul de produse pe care un client doreste sa le cumpere

3. Proiectare

Aplicatia are la baza 14 clase acestea fiind grupate in package-uri : bll, connection, controller, dao, model si presentation. . Acesta este organizata confrom sablonului Model-View-Controller (MVC). Acest tip de organizare reprezinta separarea interfetei utilizator într-o vedere (View) care interactioneaza cu Modelul după nevoi, si un Controller care raspunde la cererile utilizatorului, interactionand atat cu View-ul, cat si cu Modelul.

3.1. Diagrama UML

Graphical user interface

Description automatically generated

3.2. Clasa ConnectionFactory

In aceasta clasa s-a facut conexiunea cu serverele din mysql pentru a putea putea modifica continutul acestora prin intermediul interfetei pe care am implematat-o.

3.3. Clasa Client

Aceasta clasa contine atributele destinate fiecarui client, acestea fiind id-ul acestuia, numele de familie, prenumele si adresa acestuia . Aici au fost implementate motodele de get si de set , dar si metoda de toString care afiseaza toate atributele pe care un client le are.

3.4. Clasa Product

Aceasta clasa contine atributele destinate fiecarui produs, acestea fiind id-ul acestuia,numele produsului ,pretul pentru o bucata si cantitatea pe care acest depozit o detine pentru acel produs. Aici au fost implementate motodele de get si de set , dar si metoda de toString care afiseaza toate atributele pe care un produs le are.

3.5. Clasa Order

Aceasta clasa contine atributele destinate fiecarei comenzi , acestea fiind id-ul acestuia, id-ul clientului care doreste acea comanda, id-ul produului pe care clientul o doreste si cantitatea pe care o doreste clientul din acel prous . Aici au fost implementate motodele de get si de set , dar si metoda de toString care afiseaza toate atributele pe care o comanda le are.

3.6. Clasa AbstarctDao

Acesta clasa este cea mai complexa din tot proiectul si aici se implemenetaza prin reflexie toate functionalitatile din aplicatie. In acesta clasa gasim functia findbyid care are rolul de a cauta intr-o anumita tabela o linia la care se gaseste id-ul dat de noi.Pentru aceasta functie am creat o alta care are rolul de a crea un query acesta fiind: select \* from ? where id= ? . Tot aici regasim functia findall care are rolul de a afisa toate datele dintr-o tabela. La fel ca la fucntia anterioara am facut un query separat pentru mysql care arata asa : select \* from ?. Functia de insert are rolul de a insera in tabela care trebuie un client, un produs sau o comanda cu datele pe care i se vor da in interfata . Si pentru aceasta functie am facut o functie de creare de query care arata asa : insert into ? values(?..) . Avem functia de update care va reinnoi datele din tabela cu datele introduse de noi in interfata. Pentru update am facut o functie separata destinata queriului : update ? set ? where id= ? . Tot aici avem si functia de delete care va sterge date din tabela in functie de input-urile nostare din interfata . la fel ca la celelate functii avem si la aceasta una care creaza un query pentru mysql : deleet from ? where id= ? . Urmatoarele doua functii sunt destinate opeartiei de view din interfata acestea fiind getheaderof table si gettable. Aceste doua functii se ocupa cu preluarea datelor din tabele pentru a putea fi afisare in interfata. Prima functie se ocupa de head-ul de tabel cu numele de coloane si a doua se ocupa cu tabelul in sine si coloane si linii.

3.7 Clasele ClientDao, ProductDao, OrderDao

Acest proiect este implementat prin reflexie de aceea clasele ClientDao, ProductDao, OrderDao nu au cod scris in ele , acestea doar extinzand clasa AbstractDao<”tip”> de tipul client, produs sau comanda. Fiecare clasa va putea folosi astfel fiecare functie implmentata in AbstarctDao ca si cum ar fi scrise explicit in ele.

3.8. Clasele CLientbll, Productbll, Orderbll

Aceste trei clase sunt foarte asemanatoare deaoarece au metode care fac acelasi lucru doar ca sunt implementate pentru client, produs sau comanda. Aceste clase se folosesc de clasele ClientDao, ProductDao si ProductDao care defapt au metodele implementate in AbstractDao. Aceste clase implementeaza operatiile propriuzise. Putem gasi „selectq” care afiseaza toate coloanele si liniie dintr-o tabela, „findbyid” care cauta un anumit element dupa id-ul sau, „insert” care insereaza un anumit element in tabela, „update” care editeaza un anumit elemnt, „delete” care sterge un element si functiile de „gettable2” si ”getheader” care afiseaza headerul si tabela.

3.9. Clasa View

Aceasta este una din clasele care care fac posibila implemenatrea aplicatiei dupa sablonul Model-View-Controlller. Intial aici au fost declarate text Panel-urile, Text field-urile si buton care vor apare in interfata. Prin view s-a putut crea interfata grafica in care avem un panel principal cu 3 butoane care ne vor duce in panel-uri separate fiecare pentru client,comanda sau produs. Fiecare panel separat va arata identic doar ca va fi destinat unui tabel diefrit. In acestea avem textfield-urile unde vom introduce datele pentru functia pe care dorim sa o facem . Vom avea totodata si butoanele de add, delete, edit , view si back. Add va adauga cate un element dat de noi, delete va sterge un element, edit va edita elementul , view va crea un frame nou in care va afisa tabelul actual cu toate modificarile facute pe el , iar butonul de back situat in fiecare panel secundar va fi cel care va duce utilizatorul la panelul principal. Mai avem o metoda care se numeste „newframefromview” care va ajuta la implementarea functionalitatii butonului de view creand frame-ul header-ul si tabelul.

3.10. Clasa Controller

Clasa Controller la fel ca si cea VIew face posinila implementarea aplicatiei dupa sablonul Model-View-Controller. Aici am declarat initial 4 variabile de tip view, clientbll, orderbll si productbll dupa care amfacut actionlister-ii fiecaraui buton. Butoanele se repeta pentru fiecare panel secundar. In listenerul pentru butonul de adaugare am citit fiecare element introdus de utilizator dupa care le-am pus intr-un element nou care poate fi de tipul produs, client sau comanda, la final facand functia de insert pentru acest element. Functiile de edit si deleet vor functia similar cu cea de adaugare, adica se vor citi elementele date de utilizator, se vor atribui unui element nou si pe acel element se vor face fie functia de update sau delete. Tot aici avem si Actionlisterii pentru butonele de view care vor afisa un nou frame in care vom avea tabelele. In acestia am preluat header-ul de tabel si tabelul in sine dupa care prin functia newframefromview descrisa mai sus in view vom afisa tabelul.

4. Rezultate

Rezultatele acestui proiect pot fi vazute prin pornirea aplicatiei si folosirea ei propriuzisa. Am reusit sa creez o interfata sugestibila si usor de utilizat pentru orice utilizator. Aplicatia reuseste sa modifice o baza de date cu usurinta prin apasarea catorva butoane. De exemplu daca doresti sa introduci un nou client in momentul in care vei ajunge la panel-ul principal vei apasa pe client care va deschide un panel secundar. Urmatorul pas este acela de a introduce datele pentru noul client in locurile special destina. Ultimul pas este acela de a apasa pe buton add client care are rolul de a introduce in tabelul din baza de date noul client. Ca verificare se poate apasa pe butonul view pentru a vizualiza ca noul nostru client s-a introdus cu succes in tabel. Aplicatia ruleaza bine si are toate butoanele functionale . . Dupa crearea fiecarei clase descrise in capitolele de mai sus si rularea aplicatiei , pot spune ca proiectul simuleaza foarte bine aplicatia de management a unui depozit.

5. Concluzii

Prin acest proiect am putu sa vad concret si sa inteleg avantajul reflexiei inca de la primele faze ale implementarii aplicatiei. Pot spune ca am invatat importanta organizarii timpului acordat unui proiect, structurarea sa si analizarea cu atentie a problemei inainte de inceperea implementarii propriuzisa a proiectului.

Din punctul meu de vedere , posibilele dezvoltari care pot fi aduse aplicatiei sunt : scrierea unei chitante adresate fiecarui client care face o comanda , imbunatatirea interfetei din punct de vedere estetic pentru a fi mai primitoarea si mai usor de inteles de catre utilizator, crearea unor validatori pentru introducerea datelor de la tastura, de exemplu un validator pentru adresa, pentru nume sa verifice ca prima litera este mare etc .

In concluzie , cred ca am am respectat conditiile prestabilite pentru aceasta aplicatie si ca acest proiect reuseste cu succes sa fie o aplicatie pentru gestionarea unui depozit. Desi interfata este simplista ea reuseste sa surprinda toate functiile necesare acestui tip de aplicatie.