**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

FUNDAMENTAL PROGRAMMING TECHNIQUES

ASSIGNMENT 3

**ORDER MANAGEMENT**

**Viman Andrei-Liviu**

**Grupa: 30226**

**Cuprins**

1.Obiectivul

2.Analiza problemei si scenarii de utilizare

3.Proiectare

4.Implementare

5.Rezultat

6.Concluzii

7.Bibliografie

1.Obiectivul

Obiectivul acestei teme este de a proiecta si de a implementa, un program care proceseaza operatiile cu o baza de date pe care am facut-o in mySQL. Se incearca a impementa o baza de date pentru nevoile clientilor care vor sa comande ceva, asfel fiind trei tabele cu un nume cat se poate de sugestiv: Clients, Orders si Products. Asfel noi o sa putem face operatii pe tabela prin intermediul unei interfete grafice ( operatiile sunt: insert, update, delete si select ). Pentru a putea functiona exact ca si o comanda la sfarsit va trebui sa generam in mai mule fisiere cu extensia .pdf mai multe bonuri pentru fiecare persoana care a plasat una sau mai multe comenzi. Ca o mica observatie noi in codul din java va trbui sa ne connectam la o baza de date si de fiecare data cand inseram o comanda trebuie sa verificam daca vor fi suficiente produse in stoc, daca da o sa putem sa inseram acea comanda daca nu se va arunca o exceptie si inserarea nu va avea loc, asfel urmarind in permanenta stocul asa zisului magazin. Cu alte cuvinte putem spune ca aceasta tema incearca sa simuleze relatia nevoilor unui client cu un magazin. Tot aici am incercat sa folosim o arhitectura pe mai multe nivele, ceea ce inseamna ca a fost nevoie de mult mai multe pachete pentru a putea realiza acest lucru.

2.Analiza problemei si scenarii de utilizare

Realizarea problemei a fost facuta cu ajutorul exemplului incarcat pe git, asfel fiecare tabela avand ca atribute cu nume exact ca si coloanele tabelelor deoarece numai in acest fel am putut implementa tema cu ajutorul reflexiei. Modelarea ei a fost facuta exact asa cum s-a cerut pe mai multe nivele de executie cu ajutorul mai multor pachete, fiecare pachet reprezentand un anumit nivel:

1.BLL este pachetul care contine clasa cu care creem bonul dar si clasele care implementeaza propriu zis operatiile de pe tabele, de aici fiind apelate metodele din clasele din DAO.

2.Connection este pachetul care contine o singura clasa, cu ajutorul careia facem connexiunea la baza de date din mySQL, avand functi de creeare, de logare si de inchidere a unei conexiuni.

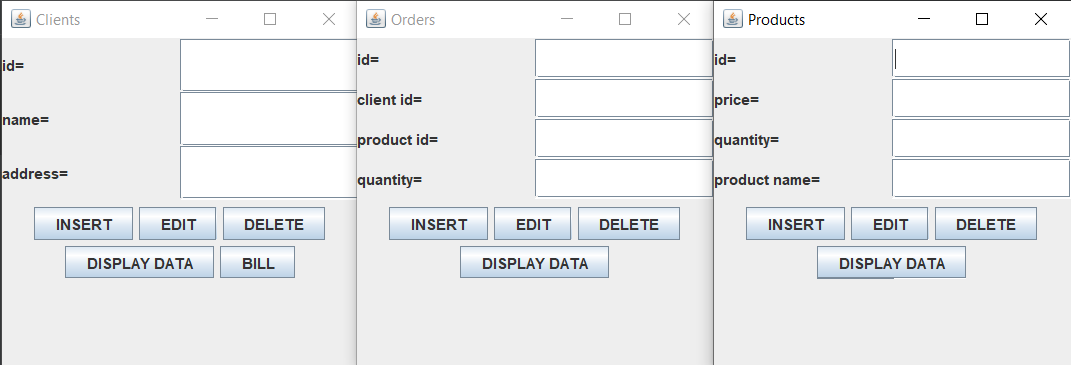
3.Controller este pachetul care contine clasa care se ocupa de implementarea ascultatorilor butoanelor.

4.DAO ete pachetul care contine patru clase, trei dintre ele fiind goale metodele fiind implementate pentru cazul general cu ajutorul reflexiei in clasa AbstractDAO.

5.Model este pachetul care contine cele trei clase asociate tabelelor bazei de date.

6.View este pachetul care implementeaza interfata grafica prentru o utilizare mul mai usoara a bazei de date.

Pentru a putea simula mai usor acest asa zis magazin am ales sa implementez si o interfata grafica care sa il ajute pe administrator sa poata urmarii mai usor evolutia comenzilor pe care clientii le plaseaza. Interfata are trei frameuri unl pentru fiecare tabela, iar fiecare tabela avand la randul ei cate un buton pentru afisarea tabelei, inserarea datelor in tabela, actualizarea lor dar si stergera lor din tabela. Frameul pentru client are un buton in plus care va genera bonurile daca exista comenzi adaugate in baza de date.



3.Proiectare

Unified Modeling Language (prescurtat UML) este un limbaj standard pentru descrierea de [modele](https://ro.wikipedia.org/w/index.php?title=Modelare_orientat%C4%83_pe_obiect&action=edit&redlink=1) și [specificații](https://ro.wikipedia.org/w/index.php?title=Specifica%C8%9Bie&action=edit&redlink=1) pentru [software](https://ro.wikipedia.org/wiki/Software). Limbajul a fost creat de către consorțiul [Object Management Group](https://ro.wikipedia.org/w/index.php?title=Object_Management_Group&action=edit&redlink=1) (OMG) care a mai produs printre altele și standardul de schimb de mesaje intre sisteme [CORBA](https://ro.wikipedia.org/w/index.php?title=CORBA&action=edit&redlink=1). UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte, pentru business Process Design etc.

Diagram

Description automatically generated

A screenshot of a computer

Description automatically generated

Pentru a gasi o solutie cat mai buna, am folosit mai multe pachet cu nume sugestiv si am ales sa folosesc o structura pe mai multe nivele. In model am incapsulat date si am implementat functionalitati, in controller am implementat cate o subclasa care care implementiaza action listener, mai exact ascultatorii pentru butoare care asteapta ca noi sa apasam pe unul dintre ele. In view am implementat asa cum s-ar spune, asa zisa legatura dintre utilizator si calculator. Acest pattern sta la baza majoritara a aplicatiilor java cu interfata grafica. De remarcat ca am folosit pachetul java.SWING pentru implementarea interfetei.

La ce revine analiza problemei? Este foarte simplu, aceasta tema are un singur scop acela fiind de a ne invata sa lucram cu bazele de date si cu reflexia in Java, deoarece sa recunoastem ca in ziua de azi orice program ca sa ruleze are nevoie de o baza de date, ele fiind indispensabile vietii curente.

4.Implementare

Am decis sa implementez clasele cat mai intuitiv posibil, cu un cod cat mai lizibil si cat mai usor de inteles deoarece in viitor va fi nevoie ca o ehipa intreaga sa inteleaga codul pentru a putea lucra toti la acel proiect. O sa descriu fiecare clasa pe rand:

Text

Description automatically generated **Clasa Clients** este clasa care contine toate atributele minime necesare pentru descrierea unui client. Am avut nevoie de nume sugestive pentru ele deoarece ca reflexia sa functioneze am avut de acelasi nume la atribute ca si la numele tabelei client din mySQL. Aceste atribute sunt: id care este un numar de ordine pe care il primeste pentru a putea fi inserat in tabela, numele lui si adresa. Aceasta clasa mai contine si gettere si settere pentru fiecare atribut dar si metoda de toString. Toate acestea au fost generate automat.

**Clasa Products** este clasa care contine toate atributele necesare pentru descrierea unui produs. Am avut nevoie de nume sugestive pentru ele deoarece ca reflexia sa functioneze am avut de acelasi nume la atribute ca si la numele tabelei client din mySQL. Aceste sunt id produsului cantitatea de pe stoc, pretul pe bucata dar si numele acestuia. Aceasta clasa mai contine si gettere si settere pentru fiecare atribut dar si metoda de toString. Toate acestea au fost generate automat.

**Text

Description automatically generated**

**Clasa Main** am instantiat un view care l-am transmis ca si parametru controllerului. Am facut acest lucru pentru a respecta patternul model view controller, un pattern destul de utilizat in java deoarece face codul mult mai lizibil si mult mai usor de iterpretat in caz ca cineva vrea sa aduca modificari ulterioare; urmand ca la sfarsit sa fac vizibil viewul.

**Text

Description automatically generated Clasa Orders** este clasa care contine toate atributele necesare descrierii unei comenzi. Am avut nevoie de nume sugestive pentru ele deoarece ca reflexia sa functioneze am avut de acelasi nume la atribute ca si la numele tabelei client din mySQL. Acestea sunt id comenzii, id clientului, dar si id produs, si in ultimul rand cantitatea cumparta. Cu aceasta tabela am facut legatura intre tabela clients si tabela products. . Aceasta clasa mai contine si gettere si settere pentru fiecare atribut dar si metoda de toString. Toate acestea au fost generate automat.

**Text

Description automatically generated Clasa ConnectionFactory** este clasa care realizeaza conexiunea la baza de date, cu alte cuvinte putem spune ca se logheaza la serverele mySql. Avem nevoie de aceasta clasa pentru a putea insera sterge sau edita datele din tabele.

**Clasa AbstractDAO** este o clasa abstracta foarte importanta deoarece aici se gasesc toate instructiunile in SQL necesare inserarii, stergeri, si editarii. Fiecare metoda are cate o sub bunctie care genereaza interogarile propriu zise iar mai apoi acestea sunt apelate din alte metode pentru o compatibilitate cat mai mare. Aici este implementata reflexia, nefiind nevoiti asfel sa scriem de mai multe ori acelasi lucru economisind asfel o multime de timp. Pe langa functiile de insert update si delete, mai exista o functie care gaseste un client, o comanda sau un produs dupa id. Tot aici se gaseste functia care selecteaza toate datele dintr-o tabela dar si functia care construieste obiectele propriu zise. Pentru a putea afisa in GUI intr-un Jtabel am facut metode care sa puna intr un tablou de tipul Object antetele tabelelor, adicab numele lor; si intr-o matrice tot de tipul Object toate datele din tabela. Pentru aceasat clasa am avut foarte mult de lucru pentru a putea fi implementata deoarece au fost de prins o multime de exceptii deoarec pentru a putea crea un obiect pe cazul general, aparent este foarte complicat. Tot aici in fiecare functie care lucreaza pe tabela a fost nevoie sa ne conectam la baza de date si si aici am intampinat cateva dificultati. Dupa ce m-am prins cum trebuie sa mearga lucrurile totul a devenit mult **Text

Description automatically generated**mai usor.

**Graphical user interface

Description automatically generatedGraphical user interface, application, website

Description automatically generatedGraphical user interface, application

Description automatically generated Clasele ClientsDAO, OrdersDAO, ProductsDAO** sunt toate clase goale deoarece am reusit sa implementez totul pe caz general, ele extinzand doar clasa AbstractDAO.

**Clasele ClientBLL, OrdersBLL, ProductsBLL** sunt clasele logie de business, adica implementeaza metodele de lucru cu baza de date pentru tabela fiecare tabela in parte. Aici fiecare metoda contine cel mult cate un rand deoarece a fost doar nevoie sa apelam functiile specifice din DAO unde se afla metodele specifice pentru principalele operatii care se pot realiza pe o tabela, ele fiind selectarea datelor dintr-o tabela dupa un id specificat, dar si selectarea tuturor datelor, inserarea lor editarea lor dar si stergerea lor. Aceste clasa au fost usor de implementat si sunt foarte asemantoare deoarece am reusit sa fac reflexia acest lucru facand programarea mult mai usoara in Java.

**Clasa view** reprezinta implementarea interfetei grafice facute cu ajutorul pachetului java.swing, ea fiind foarte usor de implementat, de aceea am si ales-o dar si pentru ca deja o stiam folosi de semestrul trecut de la cursul de poo.

In aceasta clasa am mai multe text fielduri unde utilizatorul poare introduce datele din tabele, ele fiind id clientului produsului si comenzi, dar si adresa si numele clientului. Deoarece era nevoie am adaugat si cantitatea unei comenzi dar si cantitatea aflata pe stoc in magazin. Se mai poate introduce si numele produsului din magazin dar si pretul lui pe bucta. Tot in aceasta clasa am un JFrame unde o sa afisez datele din tabela la apasarea butonului de afisare , am ales JFrame deoarece in el se afiseaza text mai frumos si se redimensioneaza in timp ce marim interfasa, daca se doreste acest lucru. In continuare voi discuta despre multele doua butoane implementate care si ele la randul lor au nume sugestiv fiecare buton facand o operatie pe tabela; fiecare puton avand numele operatiei pe care o face.

Tot in aceasta clasa am adaugat ascultatori pentru fiecare buton, dar implementarea propriu-zisa fiind in clasa controller. In aceasta clasa mai exista mai metode cu care luam stringurile introdus in text fielduri.

**Clasa controller** are mai multe subclase, cate o subclasa pentru fiecare buton, mai exact cate o clasa pentru pecare ascultator al fiecarui buton. Am ca si atribute un model si un view deoarece controllerul trebuie sa faca legatura intre model si view.

Graphical user interface, application

Description automatically generated Pentru ascultatorii butoanelor am transformat acolo unde a fost nevoie datele in intregi sau in numere cu virgula flotanta de exemplu la id-uri sau la pret sau la cantitate, daca nu faceam acest lucru nu puteam insera datele in tabela. Pot spune ca controllerul este foarte strans legat de view, el avand ca prim atribut un obiect de tipul view.

**Clasa Bill** este clasa cu care generez asa zisul bon. Cum l-am genereat ? am facut legatura intre toate tabelele urmand ca mai apoi pentru fiecare client sa vad ce a vrut sa comande apui m am uita in tabela produs pentru a putea vedea cat costa produsul ales de client si am inmultit cantitatea din comanda cu pretul din produs urmand ca la final sa fac suma preturilor tuturor produselor pentru a o putaea adauga la finalul fisierului .pdf exact ca si la un bon pe care il primesti de la magazin.

5.Rezultat

Aceasta tema are ca rezultat invatarea studentilor sa foloseasca reflexia dar si lucrul du bazele de date. Acest lucru reusind sa il fac cu ajutorul interfetei grafice facand asfel o baza de date mult mai interactiva si mult mai usor de folosit. Daca era cu interogari mai complexe si sa aiba o stabilitae mai buna ea ar putea fi folosita chiar si de magazinele mici deoarece functioneaza asemanator cu apleicatiile pe care proprietarii magazinelor le cumpara cu abonament lunar.

6.Concluzii

Graphical user interface, text, application, website

Description automatically generated Pot spune ca a fost prima data in viata mea cand am lucrat cu reflexiain java si chiar a fost destul de interesant. Am intelezi cum trebuie sa se modeleze viata reala datele din mai multe domenii . Principalul obiectiv al acestei teme este de a ne invata sa lucram cu reflexia deoarece acest lucru ne poate usura foarte mult viata deoarece nu mai suntem nevoiti sa scriem asa de mult cod; sa recunoastem ca un programator bun va sti intodeauna sa isi refoloseasca codul pentru a creea o mai buna stabilitate peste timp a programului. Tot aici pot spune ca este prima data in viata mea cand am folosit JavaDoc, am generat-o si se poate vedea mai jos:

7.Bibliografie

<https://ro.wikipedia.org/wiki/Model-view-controller>

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

[Layers of a Standard Enterprise Application - DZone Java](https://dzone.com/articles/layers-standard-enterprise)

[Introduction to JavaDoc | Baeldung](https://www.baeldung.com/javadoc)

[Creating PDF Files in Java | Baeldung](https://www.baeldung.com/java-pdf-creation)

[Java Reflection Tutorial (jenkov.com)](http://tutorials.jenkov.com/java-reflection/index.html)

[Utcn Dsrl / pt-reflection-example · GitLab](https://gitlab.com/utcn_dsrl/pt-reflection-example)

[Utcn Dsrl / pt-reflection-example · GitLab](https://gitlab.com/utcn_dsrl/pt-reflection-example)