**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

**Tema 3**

la disciplina

**Tehnici de programare**

**Titlu**

***„OrderManagement”***

Nume si prenume : Trif Gheorghe Andrei

Grupa: 30225

An academic : 2018-2019

**1.Obiectivul temei**

Propuneti, proiectati si implementati un sistem de gestionare al comenzilor facute de anumiti clienti pentru un depozit.

|  |  |  |
| --- | --- | --- |
| Obiective secundare | Descriere | Capitolul |
| Dezvoltare de use case-urii si scenarii in legatura cu intelegera temei propuse. | Se va dezvolta un comportament pe care programul va incerca sa le urmeze. | 1 |
| Analiza problemei si modelarea acesteia | Se va prezenta proiectarea OOP a problemei ,diagrame UML ,pachete si structuri de date folosite. | 2 |
| Dezvoltarea algoritmilor | Se va perzenta operatile alese  si descrierea lor. | 3 |
| Impelmentarea solutiei | Se va prezenta codul sursa aferent proiectari. | 3 |
| Testare aplicatiei | Prezentarea ca de cazuri favorabile de rulare si modul de utilizare ,dar si anumite defecte. | 4 |
| Dezvoltare ulterioara | Noi ideii de a inbunatati proiectul existent. | 5 |

Scopul acestui proiect este de a implementa un sistem de gestiune al comenzilor pentru un depozit si de a descrie modul prin care acest sistem poate fi creat si manipulat cu ajutorul paradigmelor OOP.

Comenzile sunt utilizate in lumea reala pentru ca un anumit client sa poata obtine serviici de la un anumit producator,depozit .Obiectivul acestui sistem este acela de a usura munca angajatilor unui astfel de depozit prin a face mai eficient modul de organizare si de gestionare al comenzilor in conditile in care exista un numar foarte mare de clienti si produse.Pentru a putea crea un astfel de sistem avem nevoie de o baza de date in care sa fie disponibile toate informatiile necesare unui client despre un produs cat si anumite informatii despre client,cum ar fii: nume prenume , adresa pentru a putea finaliza fiecare comanda in mod cat mai simpliu.

Comenzile sunt compuse atat din informatii despre client cat si informatii despre produs,deoarece acest lucru nu poate fi modificat intre client -comanda -produs exista o relationare pe care am implementat-o cu ajutorul limbajului Java la nivelul fiecarei operatii a acestui sistem de gestiune.

2.Analiza problemei si modelarea acestei

2.1.Analiza problemei

În programare o comanda poate fi privita ca o combinatie de informatie intre doua sau mai multe

entitati care relationand una cu cealalta dau nastere unei comezi.Expmplul cel mai banal fiind acela cand un client oarecare comanda un anumit pordus la un magazin online.

Aceste tipuri de informatie pot fi stocate foarte usor si incapsulate impreuna cu ajutorul limbajului mySql care este indicat petntru acest aspect de lucru cu o varietate foarte mare de informatie .

Cateva proprietati ale bazelor de date:

1. O bază de date este un depozit de date destinat să sprijine stocarea eficientă a datelor, dar şi regăsirea şi întreţinerea acestora.

2. Datele pot fi stocate în diferite formate, cum ar fi: tabelar, ierarhic sau graf. În cazul în care datele sunt stocate în format tabelar, atunci avem de a face cu o bază de date relaţională.

Prin analiza problemei, ne referim la un prim set abstract de operații și proprietăți prin care încercăm să depistăm eventualele însușiri și comportamente ale proceselor necunoscute. Programarea orientată pe obiect ne oferă aici un avantaj clar, tocmai fiindcă ea permite să tratăm problema de la un nivel superiror, fără a mai fi constrâși, într-o așa măsură, de caracteristicile tehnice.

Un model al informaţiilor este o reprezentare abstractă, formală a entităţilor cu ajutorul căreia se precizează proprietăţile, relaţiile şi operaţiile ce pot fi efectuate asupra acestora. Entităţile care se modelează pot proveni din lumea reală, cum ar fi dispozitivele existente în cadrul unei reţele, sau pot să fie le însele abstracte, cum ar fi entităţile utilizate în cadrul sistemelor de facturare.

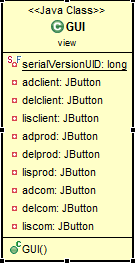
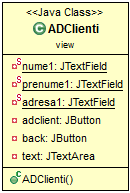
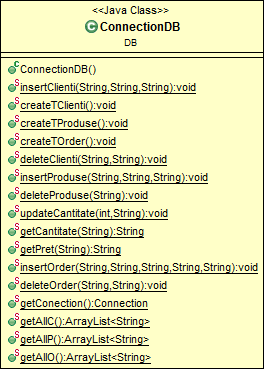
Scopul principal care se ascunde în spatele acestui concept este acela de a formaliza descrierea domeniului unei probleme fără a avea în vedere constrângerile prin care descrierea respectivă este afectată la momentul transformării acesteia în cadrul implementării .

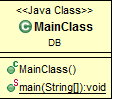
Modelarea este un aspect deosebit de important, deoarece trebuie luate în considerare aspecte ce trebuie avute în vedere în cazul eventualelor modificări ulterioare, fără a afecta în mod semnificativ utilizarea produsului. Modelarea permite compatibilizarea cu modelele anterioare şi prevede soluţii pentru extensiile viitoare.

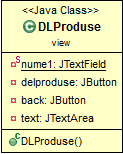
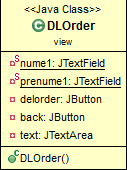
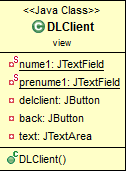
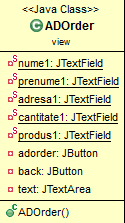
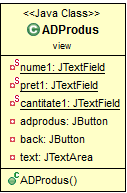
O astfel de aplicatie a fii folosita defoarte multe persoane sau poate chiar de o companie astfel un prim pas in creearea sa e sa determinam intrarile pe care un anumit utilizator ar dori sa le poata accesa.Acest lucru se face cel mai probabil prin cadrul interfetei dintre program si utilizator.Aceasta interfata trebuie sa fie foarte usor de folosit dar si foarte precisa astfel incat sa nu induca in eroare utilizatorii.Deoarece in zilele noastre mai exista persoane care nu sunt invatate cu astfel de lucruri si care ,la prima vedere le poate crea o stare de confuzie.

In cadrul acestui sistem de gestiune al comenzilor principala sa caracteristica se deduce chiardin nume " sistem de gestiune " acest lucru ne duce automat cu gandul la usurarea efortului si a munci depuse de un anumit angajat din momentul in care acel depozit primeste o comanda pana cand comanda ajunge la client.

2.2.Proiectarea claselor





Sistemul de gestiune al comenzilor este alcatuit din 13 clase :printre acestea exista 3 clase de adaugare Client Produs si Comanda (Order) , 3 clase de stergere Client Produs si Comanda si 3 clase de listare Client Produs si Comanda .Taoate aceste clase sunt conectate in cu ajutorul interfetei grafice si a butoaneleor .Toate aceste interfete graficie sunt conectate cu cujutorul clasei ConnectionDB la baza de date.

In clasa GUI este creata interfata dintre utilizator si programul java.Aceasta contine butoanele care au rolul de a efectua o anumita operatie aleasa de utilizator cum ar fii adaugarea unui client sau a unui produs sau stergerea lor.

Clasa ConnectionDB este clasa de baza a aplicatiei ea face legatrua dintre java si mySql printr-o serie de metode cu rol de legatura .Tot aici sunt si metodele de creare a tabeleelor necesare in baza de date precum si metodelor care fac anumite modificari in baza de date.

Clasele de adaugare ADClienti ,ADProdus ,ADOrder sunt interfetele grafice in care un utilizator poate face unda dintre aceste operatii care va modifica automat baza de date .

Clasele de stergere DLClienti ,DLProduse ,DLOrder sunt interfetele grafice in care un utilizator poate face unda dintre aceste operatii care va modifica automat baza de date .Aceste clase au rolul de a usura stergerea care in mod normal pate fii problematica pentru un utilizator neexperimenta ,sintaxa Sql fiind mai greu de scris.

Clasele de listare LSClienti ,LSProduse ,LSOrder sunt interfetele grafice in care un utilizator poate face unda dintre aceste operatii care va modifica automat baza de date .Aceste clase au rolul de a afisa datele din table astfel incat utilizatorul sa nu fie nevoit sa foloseasca alte medii de lucru cu baze de date.

In final mai exista o clasa Worning care are rol de afisare a unui advertisment.

3.Impementarea Claselor

3.1.Clasele AD (inserare)

Aceasta clase au rolul de adaugare a unui client produs sau comanda in baza de date .Acest lucru se face prin luare anumitor campuri date de utilizator cu ajutorul carora se executa un Querry in MySql.

String a1=*nume1*.getText();

String a2=*prenume1*.getText();

String a3=*adresa1*.getText();

**try** {

Methodmethod1=ConnectionDB.**class**.getDeclaredMethod("insertClienti",String.**class**,String.**class**,String.**class**);

method1.invoke(**null**,a1,a2,a3);

text.append("Client adaugat!!\n");

}**catch**(Exception e1) {

System.***out***.println(e1);

}

Dupa adaugare unei comenzi respectivul utilizator va fii instintat de costul total al comenzi iar daca in cazul in care in stoc ne exista cantitatea necesara dintr-un anumit produs acesa va primi o alerta cum ca depozitul nu are cantitatea dorita de el.

3.2.Clasele DL( delete )

Aceste clase au la baza aceeasi idee ca si clase de adaugare doar ca ele executa o interogare sql de stergere

.

delproduse.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

String a1=*nume1*.getText();

**try** {

Method method1=ConnectionDB.**class**.getDeclaredMethod("deleteProduse",String.**class**);

method1.invoke(**null**,a1);

text.append("Produsul a fost sters din stoc!!\n");

}**catch**(Exception e1) {

System.***out***.println(e1);

}

});

3.3.Clasele de LS(listare)

In aceste clase am creat un JTable (tabel) in care sunt afisate toate campurile care au legatura cu un anumit client ,produs sau comanda .Astfel cu ajutorul unei metode care ruleaza o interogare de select am reusit sa iau toate informatiile din baza de date si sa le introduc in tabelul creat.

**try** {

Method method1=ConnectionDB.**class**.getDeclaredMethod("getAllP");

arr= (ArrayList<String>) method1.invoke(**null**);

}**catch**(Exception e1) {

System.***out***.println(e1);

}

**for**(**int** i=0; i<arr.size();i=i+4) {

row[0] = arr.get(i);

row[1] = arr.get(i+1);

row[2] = arr.get(i+2);

row[3] = arr.get(i+3);

model.addRow(row);

}

3.4.Clasa ConnectionDB

In aceasta clasa sunt toate metodele care efectueaza operatii in baza de date de la conectare pana la inserare ,stergere ,selectare , cateva metode de determinare a unui anumit cam din baza de date.

**public** **static** **void** insertClienti(String a1,String a2,String a3){

**try** {

Connection conn=*getConection*();

PreparedStatement post=conn.prepareStatement("INSERT INTO clienti (nume, prenume, adresa) VALUES ('"+a1+"','"+a2+"','"+a3+"')");

post.executeUpdate();

}**catch**(Exception e){

System.***out***.println(e);

}

}

**public** **static** **void** createTClienti(){

**try** {

Connection conn=*getConection*();

PreparedStatement post=conn.prepareStatement("CREATE TABLE IF NOT EXISTS clienti(id int NOT NULL AUTO\_INCREMENT,nume varchar(50) ,prenume varchar(50), adresa varchar(50),PRIMARY KEY(id))");

post.executeUpdate();

}**catch**(Exception e){

System.***out***.println(e);

}

}

Cea mai importanta dintre metode este metoda de conexiune la baza de date

**public** **static** Connection getConection() {

**try** {

String driver = "com.mysql.cj.jdbc.Driver";

String url = "jdbc:mysql://localhost/store";

String user = "root";

String pass = "omegaacer1";

Class.*forName*(driver);

Connection conn=DriverManager.*getConnection*(url,user,pass);

**return** conn;

}**catch**(Exception e) {

System.***out***.println(e);

}

**return** **null**;

}

Aceasta preia informatiile necesare conectari (url ,user,pass ) si cu ajutorul unui drivermanager realizeaza conexinea .

3.4.Java refelection

Reflectia este caracteristica al limbajului java.Acesta permite unui program sa se examineze si sa isi modifice propria structura.Reflectia la nivel de clase ne permite sa accesam informatii de atributele ,constructorii cat si despre metodele clasei respective.   
Prin reflecție putem invoca metode în timpul rulării, indiferent de specificatorul de acces utilizat cu acestea. Putem invoca o metodă prin reflecție dacă îi cunoaștem numele și tipurile de parametri.

**try** {

Method method1=ConnectionDB.**class**.getDeclaredMethod("deleteProduse",String.**class**);

method1.invoke(**null**,a1);

text.append("Produsul a fost sters din stoc!!\n");

}**catch**(Exception e1) {

System.***out***.println(e1);

}

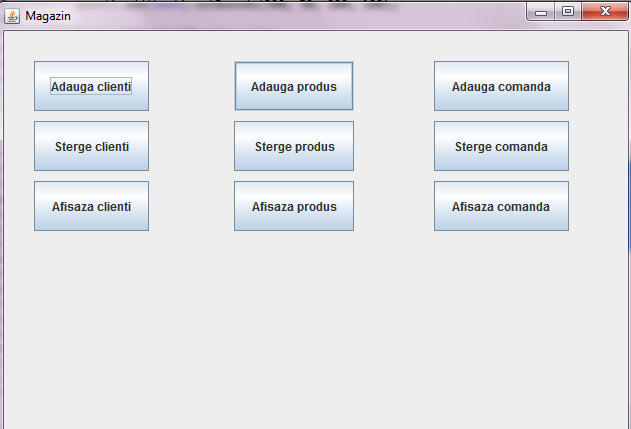
In exemplul folosit mai sus seobserva folosirea metodei getDeclereadMethod pentru folosirea acestei metode in calsa in care avem nevoie fara ca cele doua clase sa aibe vreo legatura una cu celalta.   
 Prin reflecție putem accesa variabilele și metodele private ale unei clase cu ajutorul obiectului de clasă și invocăm metoda folosind obiectul așa cum sa discutat mai sus.

Avantajele folosirii reflexiei:

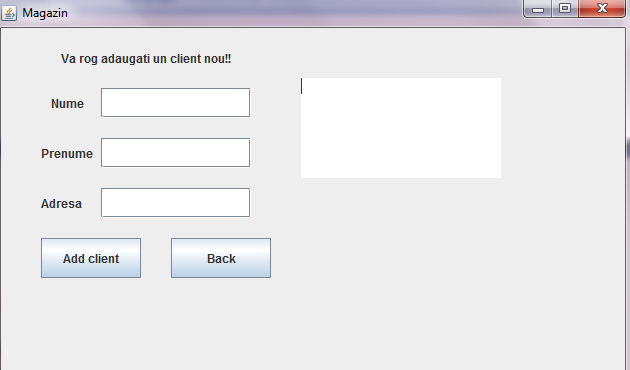
Caracteristicile extensibilității: O aplicație poate folosi clase externe, definite de utilizator, prin crearea de instanțe de obiecte de extensibilitate utilizând numele lor complet calificate.

Instrumente de depanare și testare: Debuggerii utilizează proprietatea reflecției pentru a examina membrii particulari pe clase.

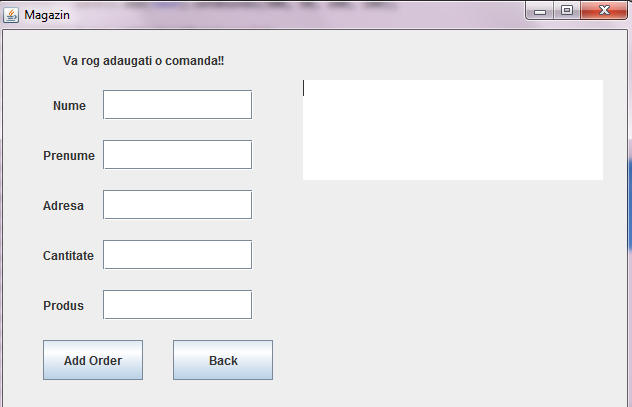
4.Testarea aplicatiei

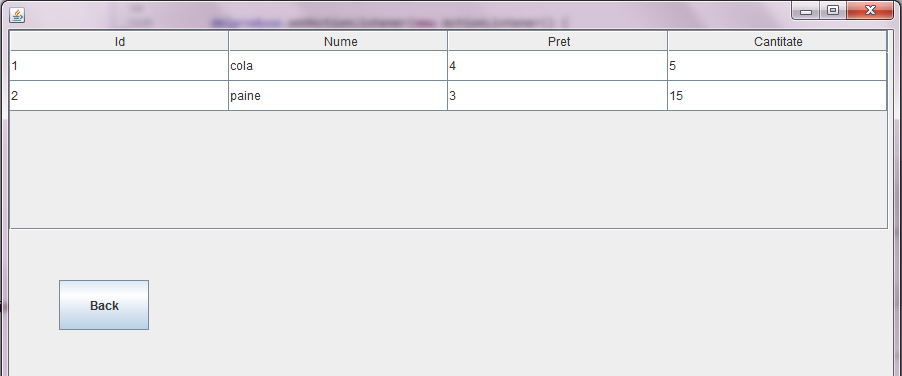


Testarea acestei aplicatii este foarte simpla si usor de facut.Primul lucru de facut este selectarea unei optiuni din cele care apar la pornirea aplicatiei.Dupa ce se alegea o operatie utilizatorul trebuei sa completeze campurile neceasre rularii operatiei.Spre exemplu pentru a insera un client in baza de date utilizatorul trebuie sa competeze campurile de nume prenume si adresa si sa apese pe butonul de add client.



In cazul unei adaugari de comanda utilizatroul va trebui sa completeze campurile despre client nume prenume adresa cat si campurile despre produs nume si cantitate .Dupa adaugra comenzi care se face de pe butonul de add comanda utilizatorului ii vor aparea in caseta text datele despre comanda cat si pretul total al comenzi.





Utilizatorul poate vedea in aceste tabele toate informatiile necesare despre un pordus ,clint sau o caomanda .Acest lucru usureaza fortem mult acest proces deoarece nu mai este nevoie de o aplicatie care sa se ocupe de gestionarea bazei de date cum ar fii phpmyadmin in care un utilizator ne experimientat se poate pierde foarte usor.

5.Dezvoltarea ulterioara

Din aceasta tema mi-am dat seama de numaratele moduri de accesare a bazelor de date cat si despre folosirea caracteristicilor limbajului java cum ar fii reflection care in astel de situarii ajuta foarte mult la simpificarea metodelor de crearea si lucru pe bazele de date.

O prima dezvoltare ar fii extinderea bazei de date prin adaugarea de tabele si relatii mai complexe cat si campuri care depind mai mult unele de altele.

O alta ulterioara dezvoltare a acestei aplicatii de gestiune ar putea fii crearea de operatii noi nu dar operatii care sa ajute clentii cat si operatii care sa ajute furnizorii depozitului respectiv.

Bibliografie

[1].https://www.geeksforgeeks.org/reflection-in-java/

[2]. https://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/

[3]. https://ibytecode.com/blog/jdbc-mysql-create-database-example/

[4]. http://tutorials.jenkov.com/java-reflection/index.html

[5].http://coned.utcluj.ro/~salomie/PT\_Lic/4\_Lab/HW3\_Tema3/Tema3\_HW3\_Indications.pdf