Tema 3: Order Management

Andercou Alexandru Stefan

Grupa 30221

1.Obiectivul temei

Proiectul generat este o aplicatie minimala pentru managementul unui magazin sau depozit care are in vedere managementul produselor dintr-un depozit ,a clientiilor magazinului si a comenziilor facute in interiroul magazinului. Programul ofera interfete grafice pentru controlul separat a celor trei mari componente a magazinului.

De asemenea programul se foloseste de o baza de date de tip mysql , de un server Mysql si de workbench.

2.Analiza problemei

De la tema problemei se despind cele 3 mari componente ale modelului: clienti,comenzi si produse mapate pe clasele : Clienti , Orders si Produs. De asemena se identifica necesitatea accesului la o baza de date prin ConnectionFactory si conexiunea intre baza de date si modelul de date prin pachetul dao: si interfete grafice care sa ofere accesul utilizatorului la modelul de date din pachetul presentation.

Pentru realizarea functionalitatii proiectului se identifica interactiunea intre utilizator si interfete.

Proiectul are un singur utilizator care functioneaza ca si client dar si ca administrator: acesta poate adauga ,edita,sterge si a vizualiza clienti, comenzi si produse.

Pentru tratarea separata a clientiilor , a produselor si a comenziilor efectuate se pot crea trei interfete separate. Pentru ai oferi utilizatorului posiilitatea de a alege pe ce obiecte va efectua operatii exista interfata Menu cu 3 butoane separate care duc la 3 interactiuni ulterioare separate.

Use Case<utilizatorul opereaza in interfata pentru clienti>

Actor<utilizator>

1. Daca utilizatorul a apasat pe butonul cu labelul Clients din fereastra Menu acesta va schimba fereastra in fereastra pentru operatii pe clienti. Interfata pentru clienti are 5 butoane si un tabel cu datele clientiilor.

2.1 Daca utilizatorul apasa pe butonul newClient se va deschide o fereastra separata cu 5 campuri de tip text. Dintre acestea doua pentru nume care este separat in first name si last name ,una pentru adresa,una pentru email si una pentru varsta.

2.1.2 se completeaza campuriile text

2.1.3 se apsasa pe butonul cu labelul Submit

2.1.3 In cazul in care datele au fost introduse corect:

2.1.3.1 se insereaza un nou client in baza de date

2.1.3.2 se updateaza id-ul clientului creat cu un id egal cu id-ul maxim existent in acel moment+1.

2.1.3.3 se inchide fereastra pentru introducerea datelor

2.1.3.4 se actualizeaza datele tabelui din fereastra pentru clienti

2.1.3.5 Interfata se intoarce in pasul 2 ,putandu-se alege o alta operatie

2.1.3 Daca in campul age nu s-a introdus un numar intreg aplicatia va esua si va aprea in consola din programul java un mesaj de eroare, interfetele for continua sa existe.

2.2 Daca utilizatorul doreste sa editeze un client

2.2.1 schimba valoriile din tabelul JTable pentru o singura inregistrare dorita

2.2.2 Selecteaza un camp din inregistrarea dorita, de preferinta primul camp

2.2.3 apasa butonul cu labelul Edit Client

2.2.4 se identifica id-ul clientului editat

2.2.5 se completeaza un obiect de tip Client cu datele din tabel

2.2.6 se updateaza baza de date pentru clientul cu indexul gasit si datele din obiect

Daca id-ul nu este valid apare o eroare in consola programului java

Alrfel

2.2.7 se actualizeaza datele din tabel care sunt afisate in fereastra.

2.3 Daca utilizatorul doreste sa stearga un client

2.3.1 Se selecteaza inregistrarea dorita prin oricare din campuriile din tabel

2.3.2 se apasa butonul cu labelul DeleteClient

2.3.3 se identifica id-ul inregistrarii selectate

2.3.4 se sterge inregistrarea din baza de date pe baza id-ului

2.3.5 se actualizeaza tabelul si se afiseaza modificat

2.4 Daca se apasa butonul Refresh

2.4.1 se actualizeaza tabelul cu datele din baza de date

2.4.2 se afiseaza tabelul cu datele actuale

2.5 Daca se apasa butonul Menu

Fereastra se schimba din fereastra pentru clienti in fereastra initiala pentru selectarea Interfetei.

Use Case<utilizatorul opereaza in interfata pentru produsei>

Actor<utilizator>

1. Daca utilizatorul a apasat pe butonul cu labelul Products din fereastra Menu acesta va schimba fereastra in f ereastra pentru operatii pe produse. Interfata pentru produse are 5 butoane si un tabel cu datele produselor.

2.1 Daca utilizatorul apasa pe butonul Add Product se va deschide o fereastra separata cu 3 campuri de tip text: unul pentru nume ,unul pentru pretul produsului si unul pentru cantitatea de produse existata in stoc din acel tip.

2.1.2 se completeaza campuriile text

2.1.3 se apsasa pe butonul cu labelul Submit

2.1.3 In cazul in care datele au fost introduse corect:

2.1.3.1 se insereaza un nou produs in baza de date

2.1.3.2 se updateaza id-ul produsului creat cu un id egal cu id-ul maxim existent in acel moment+1.

2.1.3.3 se inchide fereastra pentru introducerea datelor.

2.1.3.4 se actualizeaza datele tabelui din fereastra pentru produse.

2.1.3.5 Interfata se intoarce in pasul 2 ,putandu-se alege o alta operatie

2.1.3 Daca in campul pentru pret sau cantitate nu s-a introdus un numar intreg aplicatia va esua si va aprea in consola din programul java un mesaj de eroare, interfetele for continua sa existe.

2.2 Daca utilizatorul doreste sa editeze un produs

2.2.1 schimba valoriile din tabelul JTable pentru o singura inregistrare dorita

2.2.2 Selecteaza un camp din inregistrarea dorita, de preferinta primul camp

2.2.3 apasa butonul cu labelul Edit Product

2.2.4 se identifica id-ul produsul editat

2.2.5 se completeaza un obiect de tip Produs cu datele din tabel

2.2.6 se updateaza baza de date pentru produsul cuid-ul gasit si datele din obiect

Daca id-ul nu este valid apare o eroare in consola programului java

Alrfel

2.2.7 se actualizeaza datele din tabel care sunt afisate in fereastra.

2.3 Daca utilizatorul doreste sa stearga un produs

2.3.1 Se selecteaza inregistrarea dorita prin oricare din campuriile din tabel

2.3.2 se apasa butonul cu labelul Delete Product

2.3.3 se identifica id-ul inregistrarii selectate

2.3.4 se sterge inregistrarea din baza de date pe baza id-ului

2.3.5 se actualizeaza tabelul si se afiseaza modificat

2.4 Daca se apasa butonul Refresh

2.4.1 se actualizeaza tabelul cu datele din baza de date

2.4.2 se afiseaza tabelul cu datele actuale

2.5 Daca se apasa butonul Menu

Fereastra se schimba din fereastra pentru clienti in fereastra initiala pentru selectarea Interfetei.

Use Case<utilizatorul opereaza in interfata pentru comenzi>

Actor<utilizator>

1. Daca utilizatorul a apasat pe butonul cu labelul Orders din fereastra Menu acesta va schimba fereastra in f ereastra pentru operatii pe comenzi. Interfata pentru comenzi are 5 butoane si un tabel cu datele produselor.

2.1 Daca utilizatorul apasa pe butonul Add Order se va deschide o fereastra separata cu 3 campuri de tip text: unul pentru nume ,unul pentru id-ul clientului ,unul pentru id-ul produsului si unul pentru cantitatea dorita din acel produs.

2.1.2 se completeaza campuriile text, fiind posibil sa se treaca printre interfetele pentru clineti,produse si oferte prin interfata si butonul Menu pentru identificarea datelor dorite.

2.1.3 se apsasa pe butonul cu labelul Submit

2.1.3 In cazul in care datele au fost introduse corect iar cantitatea nu depaseste cantiatea existenat in stock pentru produs.

2.1.3.1 se insereaza comanda in baza de date

2.1.3.2 se updateaza id-ul comezii creat cu un id egal cu id-ul maxim existent in acel moment+1.

2.1.3.3 se creaza un pdf care reprezinta chitanta pentru comanda

2.1.3.4 se actualizeaza cantiatea produsului comandat

2.1.3.5 se afiseaza o fereastra cu un mesaj de succes,

2.1.3.5 se inchide fereastra pentru introducerea datelor.

2.1.3.6 se actualizeaza datele tabelui din fereastra pentru comenzi

2.1.3.7 Interfata se intoarce in pasul 2 ,putandu-se alege o alta operatie

2.1.3 Daca in campul idClient sau idProdus nu s-a introdus un numar intreg aplicatia va esua si va aprea in consola din programul java un mesaj de eroare, interfetele for continua sa existe.

Daca in campuriile idClient si idProdus s-a introdus un numar intreg dar care nu reprezinta un id valid se va deschide o fereastra de eroare si comanda va esua.

Daca numarul din campul cantitate este mai mare decat cantiaatea de produse din baza de date se va deschide o fereastra de eroare si comanda va esua.

2.2 Daca utilizatorul doreste sa editeze un produs

2.2.1 schimba valoriile din tabelul JTable pentru o singura inregistrare dorita

2.2.2 Selecteaza un camp din inregistrarea dorita, de preferinta primul camp

2.2.3 apasa butonul cu labelul Edit Orders

2.2.4 se identifica id-ul comanda editata

2.2.5 se completeaza un obiect de tip Orders cu datele din tabel

2.2.6 se updateaza baza de date pentru produsul cuid-ul gasit si datele din obiect

Daca id-ul nu este valid apare o eroare in consola programului java

Alrfel

2.2.7 se actualizeaza datele din tabel care sunt afisate in fereastra.

2.3 Daca utilizatorul doreste sa stearga o comanda

2.3.1 Se selecteaza inregistrarea dorita prin oricare din campuriile din tabel

2.3.2 se apasa butonul cu labelul Delete Order

2.3.3 se identifica id-ul inregistrarii selectate

2.3.4 se sterge inregistrarea din baza de date pe baza id-ului

2.3.5 se actualizeaza tabelul si se afiseaza modificat

2.4 Daca se apasa butonul Refresh

2.4.1 se actualizeaza tabelul cu datele din baza de date

2.4.2 se afiseaza tabelul cu datele actuale

2.5 Daca se apasa butonul Menu

Fereastra se schimba din fereastra pentru clienti in fereastra initiala pentru selectarea Interfetei.

3.Proiectare

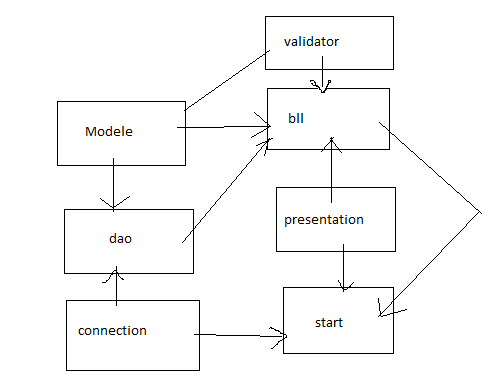
Am folosit o structura arhitecturala de tip Layered Arhitecture. Pentru aceasta am dezvoltat 7 pachete care se impart in pachete pentru accesul la data de baze: connection , si conectiunea intre modele si baza de date dao, pachete de tip model:modele , pachet de prezentare:prezentation , 2 pachete pentru logica de businiss: bll,validator si un pachet pentru programul principal clasa: Start.

Modelele de date se folosesc in pachetel de acces la datel bazei de date dao,si in pachetele pentru logica business-ului .

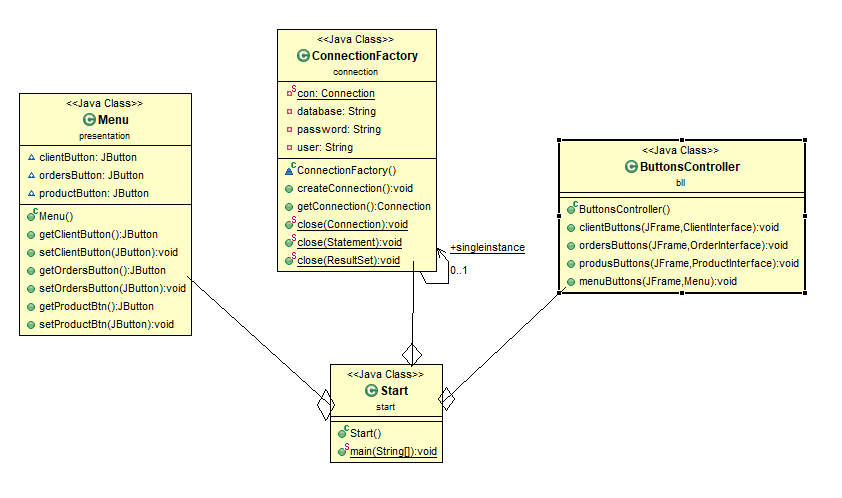
Pachetul presentation este folosit in pachetul bll si in metoda de start a programului.

Pachetul dao:” data acess objects” se foloseste de conexiunea la baza de date si de modele pentru a genera obiectele folosite in logica de business.

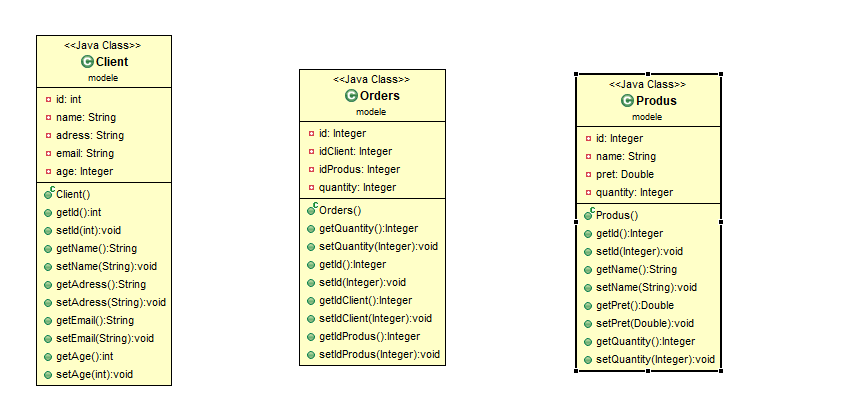
Validator e un pachet de tip de logica de aplicatie si este strins legat de pachetul bll.



Pachetele presentation ,bll si connection sunt folosite in metoda main din obiectul Start din pachetul de start. Pacehtul connection prin obiectul Singleton ConnectionFactory este folosit in Start pentru a crea conexiunea la baza de data .Pachetul presentation prin clasa Menu este folosita pentru a porni interfata grafica initiala care sa-i ofere utilizatorului oprtiunea de a alege pe ce obiecte fundamentale din modelul de date va functiona: clienti,produse sau comenzi.Pachetul bll prin clasa ButtonController se foloseste pentru a oferi logica de bussiness interfetei Menu.

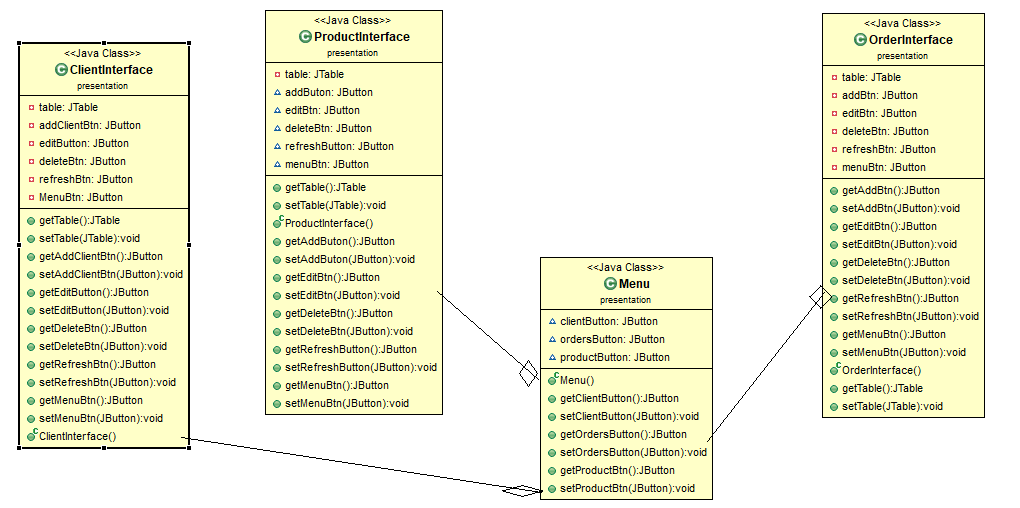


In pachetul de modele am declarat clasele: Client, Orders si Produs care corespund tabele din baza de date mysql atat in denumire cat si in numele fielduriilor pentru automatizarea procesului de conversie din baza de date in modele prin java Reflection. Pentru fiecare camp exista gettere si settere. Clasa Client are campuriile id de tip intreg, name,adress,email de tip String si age tot de tip intreg. Clasa Orders are campurile:id,idClient si idProdus care sunt de tip intregi .Clasa Produs are campuriile id si quantity care sunt de tip intreg , campul name care este de tip String si pret care este de tip Double.

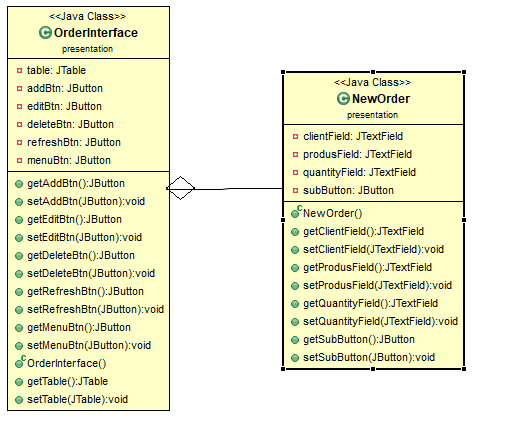
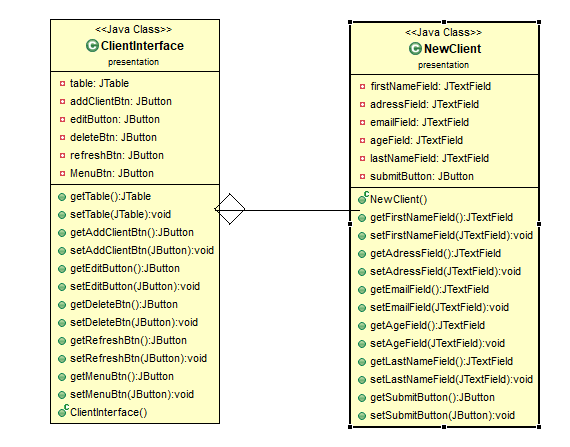


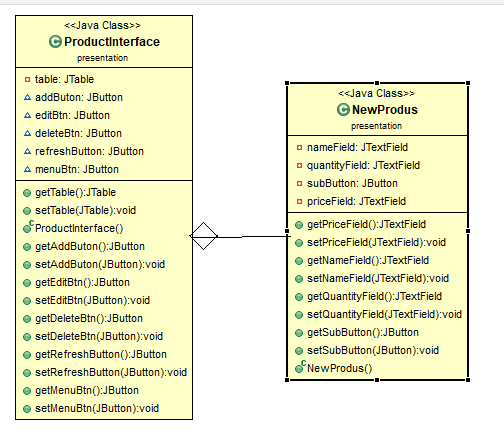
In pachetul presentation avem 7 interfete grafice, dintre acestea intre clasa Menu si restul Interfetelor exista o conexiune prin butoanele interfetei Menu si actionListenurilor conectati la aceste butoane. La apasarea unuia din cele trei butoane , se creaza un nou obiect care mosteneste JPanel, comform butonului apasat, care devine contextul ferestrei programului principal.

Interferfele grqafice ClientInterface,OrderInterface si ProductInterface au fiecare cate 5 butoane si un table pentru date. Interfata Menu are doar 3 butoane.



Intre interfetele grafice principale :ClientInterface , OrderInterface si ProductInterface si interfetele secundare NewClient ,NewOrder,NewProdus sunt conectate prin butoabele de adaugare a unor clienti,oferte sau produse noi.





In pachetul pentru logica de bussiness avem 5 clase.

Clasa TablePopulator are rolul de-a completa un tabel de tip JTable pe baza unor obiecte de orice tip,creind headerul tabelui prin java Reflection, aceasta re o singura functie populate.

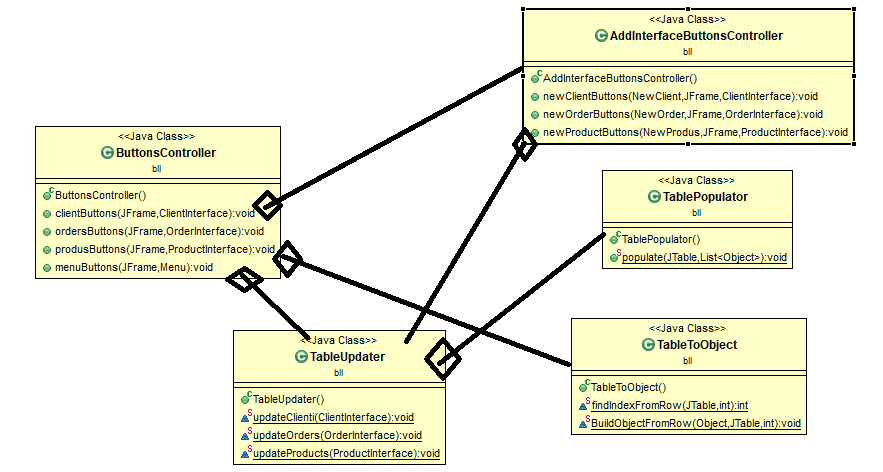
Clasa TablPopulator este folosita in clasa TableUpdater care specializeaza popularea tabelor pentru datel din kodelul de date aceasta are metodele: updateClienti,updateOrders,updateProducts care iau ca parametrii o interfata grafica, un obiect care mosteneste clasa JPanel.

Clasa TableUpdater este folosita de clasa AddInterfaceButtonsController care defineste logica pentru butoanele interfelor grafice: NewClient,NewOrder,NewProdus, foliste pentru introducerea datelor pentru crearea de noi clienti ,produse si comenzi.

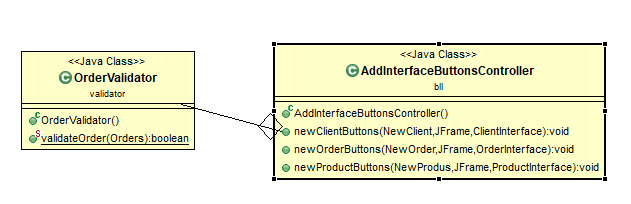
Table Updater mai este folosita de clasa ButtonsController in actionListenul tuturor butoanelor cu exceptia butonului Menu. ButtonsController are definita doar metodele pentru fiecare din cele 4 interfete grafice principale: clientButtons,ordersButtons, produsButtons, menuButtons care iau ca argumente un JFrame si o clasa care mosteneste un JPanel.

TableToObject este folosita pentru a transforma datele dintr-un tabel de tip JTable in obiecte din modelul de date si pentru aflarea id-ului , aceasta se foloseste pentru de update si delete descrise in actionListenul butoanelor din interfetel grafice.

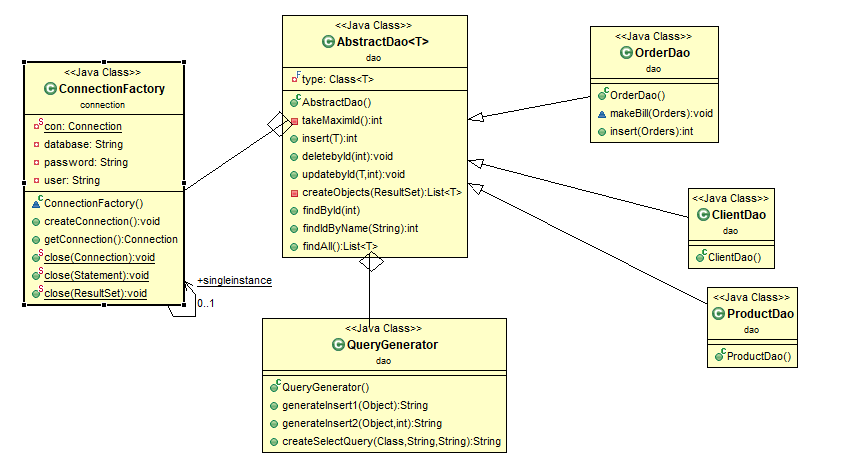
AddInterfaceButtonsController defineste logica pentru intefetele create in operatia de adaugare de noi inregistrari , pentru aceasta are metodele newClientButtons,newOrderButtons,newProductButtons, care iau ca argumente: doua obiecte de tip JPanel si un frame de tipul JFrame.



Pachetul validator contine o singura clasa OrderValidator u o singura metoda validateOrder aceasta este folosita o singura data in celalt pachetul pentru logica aplicatiei bll ,in clasa AddInterfaceButtonsController pentru crea unei noi comenzi.



Pachetul dao descrrie obiectele necesare pentru accesul la datele oferite prin conexiunea la baza de date din pachetul connection. Acest pachet are 5 clase, o super clasa AbstractDao care descrie operatiile pe o baza de date generica : adaugare,stergere,editare si gasite,3 clase care mostenesc si sprecializeaza clasa AbstractDao:OrderDao,ClientDao si ProductDao si o clasa care genereaza cateva dintre querirule folosite pentru accesul bazei de date.:QueriesGenerator



4.Implementare

Pentru actualizarea tabelelor se foloseste obiectul TableUpdater care la randul sau se foloseste de clasa TablePopulator. TablePopulator are o singura metoda numita populate care folosinduse de clasele si metodele din java.lang.reflect: Class, cu metodele getClass(),getDeclaredFields, Field cu metodele setaccesible,get ,getName si metodele si clasele din librairia javax.swing:JTable cu metoda setModel ,DefaulTableModel cu metoda insertRow() genereaza un tabel JTable cu headerul format din denumirea campuriilor din obiect si datele din obiecte.

Pentru a declara un obiect de tipul DefaultModel este nevoie de un de un obiect de tip String[] reprezentant numele coloanelor tabelului , pemtru a genera acest obiect care este un array cu lungime fixa ,cream la inceput uyn ArrayList<String> si traversand fieldurile din clasaunui obiect dat ca argument folosind metodele getClass() pentru a extrage clasa si getDeclaredFields pentru a extrage fieldurile, extragen numele campurilor prin metoda din clasa Fields getName si le adaugam in ArrayList cu metoda add

Pentru a folosi functia din DefaultTableModel insertRow() trebuie sa se dea ca argumente un obiect de tipul Object[] si un integer reprezentant un index.Pentru craerea de Object[] folosim un proces similar cu cel de la generarea coloanelor, dar in loc de getName folosim metoda get care ia ca argument obiectul care vrem sa-l punem in tabel.

Pentru a transforma un rand dintr-un tabel intr-un obiect folosim metoda BuildObjectFromRow din TableToObject care olosinduse de java reflection construieste obiectul pe baza denumirii campurilor obiectului si a headerului tabelului. Pentru realizarea acestui fapt foloseste atat java reflection cat si elemente din pachetul javax.swing/

Foloseste un obiect de tipul DefaulTableModel din Jtable obtinut prin ,metoda getModel ,din aceasta clasa foloseste metodele getColumnCount ,getColumnName si getValueAt , pe langa acesta foloseste si obiecte de tip Class si Field cu metodele: getDeclaredFields(),setAccessible,field.getName(),field.set.

ableModel dm=jT.getModel();

In clasele AbstractDao si QueriesGenerator folosim java Reflection pentru a generaliza procesele de extrafgere a datelor si de scriere a datelor in baza de date. Aceasta este folosita in metodele :createObjects.deleteById,updateById, insert,takeMaximId, findAll,findIdByName,findById, findIdByName, createSelectQuery, generateInsert1,generateInsert2,updateFieldsName

In metoda createObjects folosim o serie de obiecte si metode de tip java reflection pentru a transforma un ResultSet intr-oi lisrta de obiecte de tipul List<T> unde T este genericul clasei Abstract. Metoda foloseste clasele: Field,PropertyDescriptor si Method. Metodele utilizate sunt getWriteMethod din PropertyDescriptor, getDeclaredFields din obiectul de tip Class<T> type , metoda invoke din clasa Method si metoda din obiectul field de tip Field getName().

Pentru generarea de queries se mai foloseste metoda din obiectele de tip Class getSimpleName.

Pentru generarea de queries se folosesc StringBuilder cu metodele append ,replace si toString si operatorul “+” din clasa Strings.

Tot in clasa AbstractDao se face acces la baza de date : pentru acest fapt se folosesc obiectele:din java.sql :Connection,Statement,PreparedStatement,ResultSet si metodele : din Statement si PreparedStatement :execute, executeQuery,executeUpdate,setInt,setString, si din ResultSet: getInt,getString,getaObject,next().

Din obiectul de tip Connection se folosesc metodele: createStatement and prepareStatement.

Pentru crearea de bonuri in clasa OrderDao folosim metoda makeBill, aceasta ia ca argument un obiect de tip Orders , gaseste produsul si clientul in baza de date folosiduse de ProductDao si de ClientDao, gasseste data si ora actuala folosind obiectele SimpleDataFormat,Date si metoda System.currentTimeMillies(),creaza stringuri pentru fiecare linie din bon si creaza un pdf.

Pentru realizarea pdf-urilor sa folosit libraria pdfbox din apache introdusa ca dependinta in fisierul pom din programul Mavern. Din aceasta librarie se folosesc obiectele:PDDocument cu instanta document ,PDPage cu instanta page,PDPageContentStream cu instanta contentStream. Din obiectul document folosim metoda addPage,save ,close si din contentStream folosim metodele:setFont,beginText,endText,close,showText,newLineAtOffset,newline.

Obiectul ConnectionFactory este un obiect de tipul Singleton ,acesta contine o singura instanta de tip final ConnectionFactory. In aceasta ConnectionFactory avem metodele createConnection ,close(Connection),close(Statement),close(ResultSet), getConnection.

In metoda createConnection folosim obiectul DriverManager cu metoda acestuia :getConnection si a variabilelor de instanta user,password si database instantiaza obiectul privat Connection,acesta este apelat in metoda main din clasa Start.

Metodele de tip close apeleaza metodele close pentru fiecare obiect de din pachetul java.sql.

Interfetele garfice folosesc tipuri de obiecte din pachetul javax,swing:JFrame, JPanel,JTable,JButton,JLabel,JTextField ,JLabel,DefaultTableModel,TableModel si obiecte din pa. chetele din java.awt.event: ActionListener si ActionEvent.

Fiecare din interfetel principale care extind clasa JPanel: ClientInterface,OrderInterface,ProductInterface contine: 5 butoane fiecare cu un ActionListener adaugat in clasa ButtonController ,un JTable controlat de clasele TablePopulator ,TableToObject si TableUpdater si o eticheta cu un text imutabil de tipul JLabel. Interfetele secundare: NewClient,NewProdus si NewOrder au intre 5 si 3 campuri text de tipul JTextField si tot atatea obiecte de tipul JLabel si un singur buton de tipul JButton de submit.

Pentru accesul la datele dintr-un tabel se folosesc metodele din obiectul de TableModel obtinut prin metoda getModel. Metodel utilizate sunt:getValueAt(),getColumnCount(). Din obiectele de tipul JTable folosim metoda getSelectedRow() pentru operatiile de editare si stergere.

Pentru controlul butoanelor clasele ButtonControler si AddInterfceButtonController aceseaza butoanele din interfete si folosind metoda addActionListener adauga cate un nou obiect de tipul ActionListener generat cu keywordul new la fiecare apelare a metodei addActionListener ,moment in care se si defineste metoda actionPerformed.

In metodele actionPerformed se genereaza obiecte noi mse face schimbul de context a ferestrei principale ,se genereaza noi frameuri de tip JFrame(frame2).

,se actualizeaza tabele din interfete uitlizand obiectul TableUpdater.

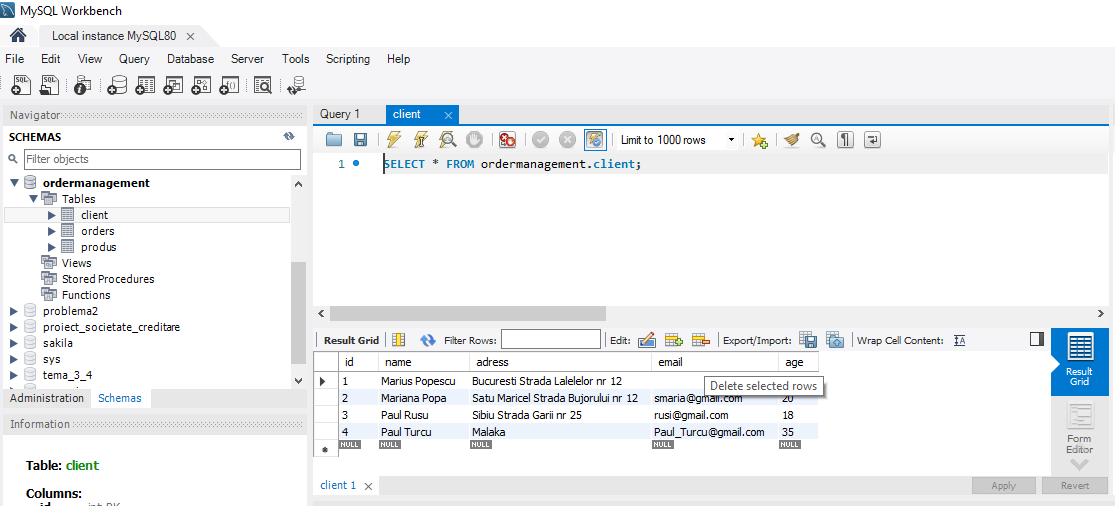
OrderValidator din pachetul validator se foloseste de clasele:ProductDao si ClientDao pentru extrage informatii legate de clienti si produse pe baza id-urilor din obiectul de tip Orders prin metoda findById daca id—urile nu corespund obiectele de produs sau client vor avea valoarea null si metoda booleana validateOrder va returna false.

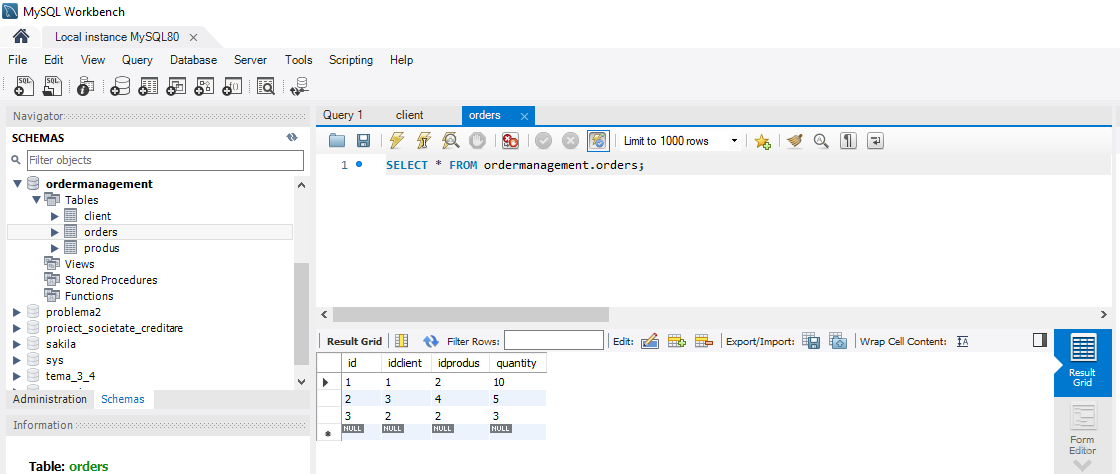
Daca id-urile corespund dar cantitatea de produse e insuficienta se va return din nou false.

5.Rezultate

Corectitudinea interfetelor si a claselor se realizeaza prin incercari de foloisre a interfetelor grafice si prin verificarea datelor din bazele de date mysql ordermanagement.

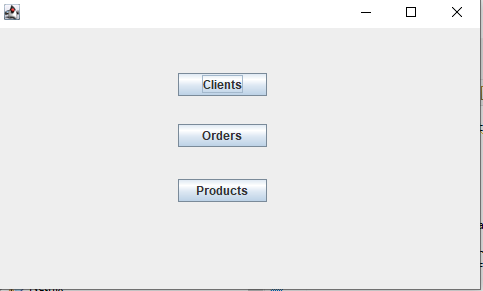
In baza de date exista 3 tabele ,printrr-o comanda de select se pot verifica datele fiecarui tabel ,de asemenea se genereaza fisiere de tip dumb care impreuna cu workbench atesta corectitudinea azplicatiei si in mod special a claselor de acces de date



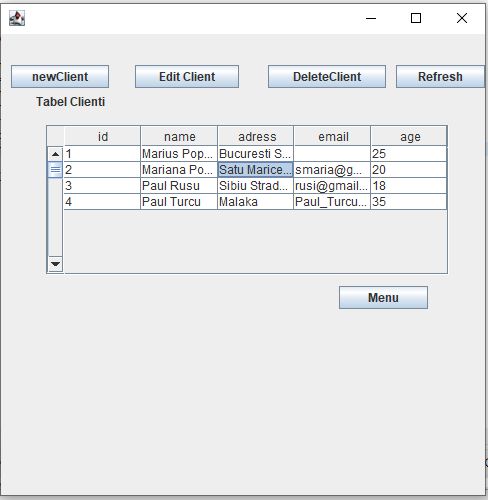


Pe interfetele grafice se pot verifica datele si functionalitatea acestora.

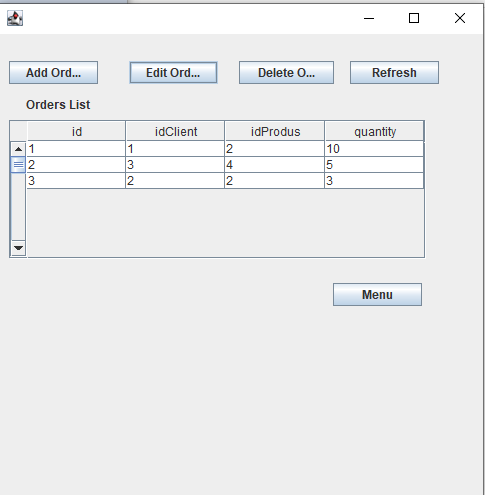
Interfata Menu : are 3 butoabe care au rolul schimbarii de context a ferestei programului.



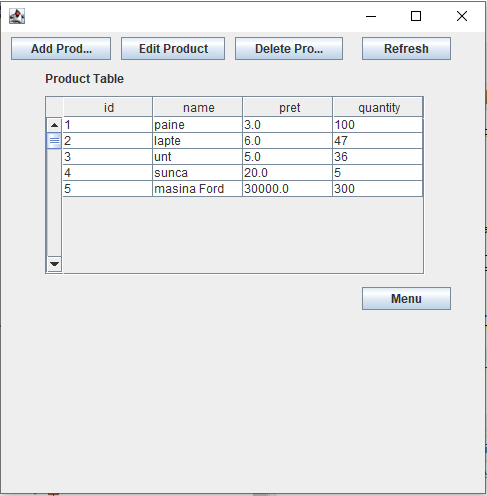
Fereastra de clienti,de comenzi si produse au aceeasi structura diferalaburile butoanelor,datele din tabele si mesajele din etichete.



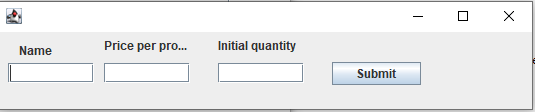
Interfata pentru comenzi



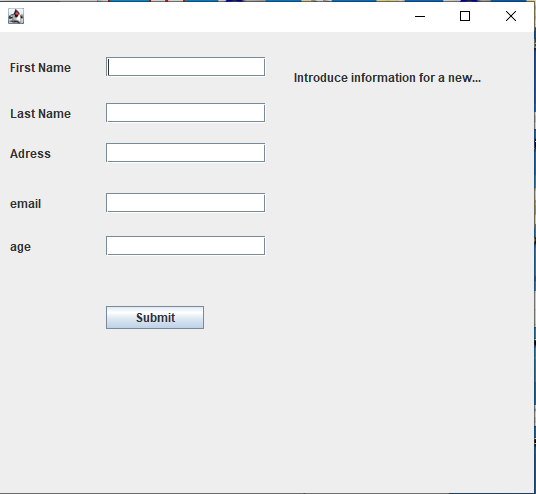
Interfata pentru produse



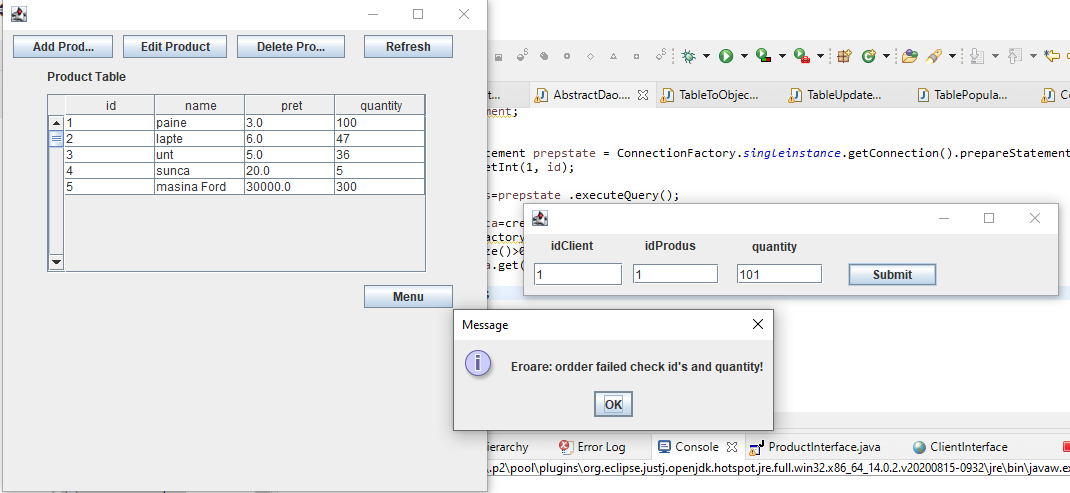
Fereasta pentru adaugarea de produse



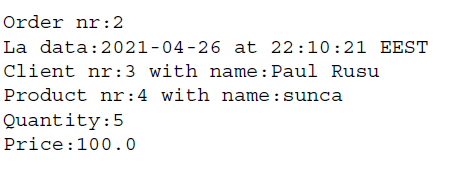
Interfata pentru adaugarea unui nou client



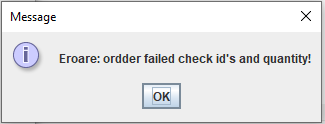
Daca apare o eroare in fereastra de adaugarea de pcomenzi apare o fereastra de eroare



Pdf pe post de chitanta daca comanda a reusit



Fereastra de eroare daca comanda a esuat



.6.Concluzii

In urma executarii temei am invatat sa utilzez Java Reflection si elemente de tip Java.swing pentru a genera tabele pentru interfete grafice . Am invatat cum sa accesam o baza de date mysql folosind java.sql si sa utilizam Reflection pentru a accesa baze de date intr-un mod generic,cum sa generam pdf-uri folosinf apache.pdfbox,cum sa generam sql-dumps si javadoc.

Tot aceasta tema reprezinta o posibila interfata utilizabila intr-o aplicatie pentru organizarea unui magazin , dar mai important este un exemplu de utilizare a modelului Layered Arhitecture care separa: accesul la baze de date,de model,de prezentare si de logica aplicatie.

. 7.Bibliografie

Connect to MySql from a Java application

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

Layered architectures

<https://dzone.com/articles/layers-standard-enterprise>

Reflection in Java

<http://tutorials.jenkov.com/java-reflectio/index.html>

Creating PDF files in Java

<https://www.baeldung.com/java-pdf-creation>

JAVADOC

<https://www.baeldung.com/javadoc>

SQL dump file generation

<https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html>