**TEMA 3**

**Order Management**

**NUME STUDENT : GLIGOR MIHAI**

**GRUPA : 302210**

**PROFESOR DE LABORATOR : DAN MITREA**

Cuprins

[Obective 2](#_Toc72089528)

[Analiza problemei , cazuri de utilizare 3](#_Toc72089529)

[Proiectare si implementare 4](#_Toc72089530)

[Rezultate si concluzii 11](#_Toc72089531)

[Bibliografie 15](#_Toc72089532)

# Obiective

**Obiectiv principal:**

* Proiectarea si implementarea unui sistem care sa administreze, completeze si editeze o baza de date cu scopul managementu-lui eficient a unor comenzi plasate de catre clintii care il folosesc

**Obiective secundare**

* Analizarea problemei si identificarea cerintelor acesteia
* Proiectarea aplicatiei
* Implementarea aplicatiei
* Testarea si controlul calitatii produsului final

## Analiza problemei , cazuri de utilizare

**Problema :**

Problema in cazul comenzilor unor produse consta in tinerea evidentei clinentilor si stocului de produse valabile in momentul unei comenzi , pentru a nu permite unui client sa comande peste limita stocului disponibil ,astfel evitand erorile si neintelegerile.

**Solutia :**

Creearea unui sistem care sa tina evidenta numarului de produse si a clentilor pentru a nu lasa loc de eroare umana , care sa aiba o interfata cu utilizatorul usor de folosit si intuitiva , facilitand astfel un management al produselor mai simplu si mai eficient

**Functionalitatile** sistemului de order management

* Adaugarea unui nou client
* Stergerea unui client
* Editarea clientilor
* Vizualizarea clientilor
* Adaugarea unui nou produs
* Stergerea unui produs
* Editarea unui produs
* Vizualizarea produselor
* Crearea comenzilor prin selectarea unui produs de catre un anumit client intr-o anumita cantitate pentru a creea o comanda valida
* Decrementarea stocului curent dupa efectuarea unei comenzi
* Afisarea unui mesaj corespunzator in cazul in care stocul este insuficient

**Order Management**

Operatii cu clienti

Adauga produs

Editeaza produs

Sterge produs

Afiseaza produse

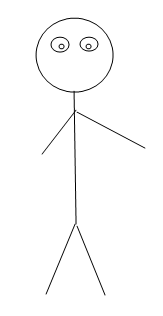
Operatii cu clienti

Adauga client

Editeaza client

Sterge client

Afiseaza clienti



Comenzi

Selecteaza client

Selecteaza produs

Selecteza cantitate

### Proiectare si implementare

**Proiectare la nivel general**

**Order management**

(efectuarea actiunii selectate)

**Selectare categorie Rezultat ( client, produs , comanda )**

**Proiectare la nivel de pachete**

Aplicatia Order Management contine sase pachete :

- pachetul main care contine doar metoda main pentru pornirea aplicatiei .

- pachetul Model care contine atributele si structura clientilor produselor si comenzilor .

- pachetul businessLayer care contine metodele necesare pentru actiunile asupra tabelelor din baza de date.

- pachetul connection care contine clasa responsabila cu stabilirea unei conexiuni intre aplicatie si baza de date .

- pachetul reflection care contine clasa Reflection care permite generarea JTable ului.

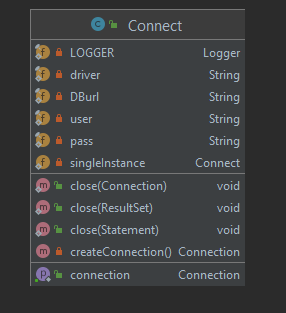
- pachetul GUI care contine toate interfetele grafice necesare aplicatiei .

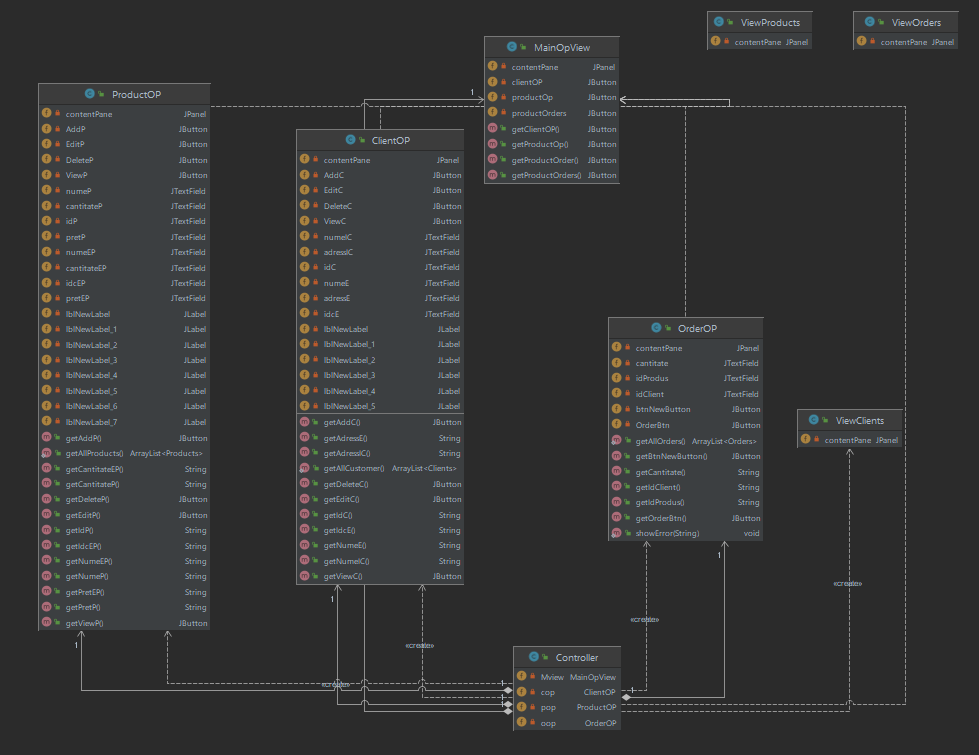
O imagine care conține text, negru, captură de ecran

Descriere generată automat

O imagine care conține text

Descriere generată automat





O imagine care conține text

Descriere generată automat

**Proiectarea la nivel de clase si metode**

Clasa Main

Contine metoda main in care sunt instantiate obiectele necesare rularii aplicatiei si in care este setata vizibilitarea GUI – ului .De asemenea aici avem loggerul care face conexiunea cu baza de date.

Clasa Clients

Contine atributele clientilor adica idC, name si adress, constructorii necesari cat si gettere si settere

Clasa Orders

Contine atributele produselor adica idP ,name, quantity,price , constructorii necesari cat si gettere si settere

Clasa Orders

Contine atributele comenzilor adica idO, nameClient , nameProduct ,quantity, constructorii necesari cat si gettere si settere

Clasa Connect

Este clasa responsabila cu stabilirea conexiunii cu o baza de date existenta . Aceasta are ca atribute numele driverului folosit , locatia bazei de date ( DBurl ) , username ul si parola necesare pentru accesarea bazei de date. Plasam conexiunea cu baza de date intr-un obiect singleton dupa care apar urmatoarele metode: - createConnection care creeaza o noua conexiune prin intermediul driverului - getConnection care returneaza singleInstance.createConnection() ( getter pentru conexiune ) - close ( Connection connection ) care inchide conexiunea curenta - close ( Statement statement) care inchide statement – ul curent - close ( ResultSet resultSet) care inchide setul de rezultate curent

Clasa ClientsActions

Contine ca atribute loggerul , si trei string -uri care corespund codului sql necesar pentru efectuarea actiunii indicate de numele string -ului ( insert delete edit ) . Fiecarei actiuni ii corespunde o metoda care o efectueaza : - insertClient( Clients c ) primeste ca parametru un obiect de tipul client . La inceput stabileste o conexiune cu baza de date dupa care initializeaza statementul sql declarat sub forma de string in atribute , scrie elementele care trebuie sa inlocuiasca “ ? ” in query dupa care inchide statementul si conexiunea . - deleteClient( Clients c ) primeste ca parametru un obiect de tipul client . La inceput stabileste o conexiune cu baza de date dupa care initializeaza statementul sql declarat sub forma de string in atribute , scrie elementele care trebuie sa inlocuiasca “ ? ” in query dupa care inchide statementul si conexiunea . - editClient( Clients c ) primeste ca parametru un obiect de tipul client . La inceput stabileste o conexiune cu baza de date dupa care initializeaza statementul sql declarat sub forma de string in atribute , scrie elementele care trebuie sa inlocuiasca “ ? ” in query dupa care inchide statementul si conexiunea.

Clasa ProductActions

Contine ca atribute loggerul , si trei string -uri care corespund codului sql necesar pentru efectuarea actiunii indicate de numele string -ului ( insert delete edit ) . Fiecarei actiuni ii corespunde o metoda care o efectueaza : - insertProduct ( Products p ) primeste ca parametru un obiect de tipul product . La inceput stabileste o conexiune cu baza de date dupa care initializeaza statementul sql declarat sub forma de string in atribute , scrie elementele care trebuie sa inlocuiasca “ ? ” in query dupa care inchide statementul si conexiunea. - deleteProduct ( Products p ) primeste ca parametru un obiect de tipul product . La inceput stabileste o conexiune cu baza de date dupa care initializeaza statementul sql declarat sub forma de string in atribute , scrie elementele care trebuie sa inlocuiasca “ ? ” in query dupa care inchide statementul si conexiunea. - editProduct ( Products p ) primeste ca parametru un obiect de tipul product . La inceput stabileste o conexiune cu baza de date dupa care initializeaza statementul sql declarat sub forma de string in atribute , scrie elementele care trebuie sa inlocuiasca “ ? ” in query dupa care inchide statementul si conexiunea.

Clasa OrdersActions

Contine ca atribute loggerul , si un string care corespunde codului sql necesar pentru efectuarea actiunii indicate de numele string -ului ( insert) . Clasa contine si metoda necesara pentru aplicarea actiunii respective - insertProduct ( Orders o ) primeste ca parametru un obiect de tipul Order . La inceput stabileste o conexiune cu baza de date dupa care initializeaza statementul sql declarat sub forma de string in atribute , scrie elementele care trebuie sa inlocuiasca “ ? ” in query dupa care inchide statementul si conexiunea.

Clasa MainOpView

Contine interfata grafica principala de unde se pot alege cele trei actiuni principale indeplinite de aplicatie : - Operatiile pe clienti - Operatiile pe produse - Comenzile Fiind compusa dintr- un simplu jFrame cu trei butoane corespunzatoare actiunilor , fiecare din acestea fiind folosit ca un punct de acces catre o alta interfata grafica specifica .

Clasa ClientOP

Contine interfata grafica responsabila pentru actiunile asupra clientului , astfel contine patru butoane de adaugare editare stergere si vizualizare clienti .In dreptul butonului Add Client se afla doua textField uri pentru introducerea numelui si adresei clientului (id ul nu este necesar deoarece este pus automat de catre baza de date) .In dreptul butonului de Edit Client se afla trei textField uri , primul pentru id , necesar pentru a identifica in mod unic clientul pe care dorim sa il editam , dupa care inca doua pentru introducerea numelui respectiv adresei noi .Butonul Delete Client are un singur textField un dreptul lui deoarece este suficient sa scriem id-ul clientului pe care dorim sa il stergem .La final avem butonul View Clients care afiseaza un JTable cu toti clientii aflati in tabela clients.Pentru fiecare textField exista cate o metoda care returneaza textul scris in acesta, de asemenea fiecare buton are o metoda care il returneaza pentru a putea fii folosit in celelate clase. La final avem metoda getAllCustomers care creeaza o conexiune cu baza de date si efectueaza un query pentru a extrage toate elementele din tabelul clienti , pe care mai apoi le pune intr-un array list pe care il returneaza ,folosit ulterior la popularea JTable ului.

Clasa ProductOP

Contine interfata grafica responsabila pentru actiunile asupra produsului , astfel contine patru butoane de adaugare editare stergere si vizualizare produse .In dreptul butonului Add Product se afla trei textField uri pentru introducerea numelui cantitatii si pretului (id ul nu este necesar deoarece este pus automat de catre baza de date) .In dreptul butonului de Edit Product se afla patru textField uri , primul pentru id , necesar pentru a identifica in mod unic produsul pe care dorim sa il editam , dupa care inca trei pentru introducerea numelui, cantitatii si pretului nou .Butonul Delete Product are un singur textField un dreptul lui deoarece este suficient sa scriem id-ul produsului pe care dorim sa il stergem .La final avem butonul View Products care afiseaza un JTable cu toate produsele aflate in tabela products. Pentru fiecare textField exista cate o metoda care returneaza textul scris in acesta de asemenea fiecare buton are o metoda care il returneaza pentru a putea fii folosit in celelate clase . La final avem metoda getAllProducts care creeaza o conexiune cu baza de date si efectueaza un query pentru a extrage toate elementele din tabelul produse , pe care mai apoi le pune intr-un array list pe care il returneaza ,folosit ulterior la popularea JTable ului.

Clasa OrdersOP

Contine interfata gafica responsabila pentru plasarea comenzilor . Aceasta contine trei textField uri . In primul introducem id ul clientului care doreste sa puna comanda apoi in cel de-al doilea punem id ul produsului dorit si in cele din urma in cel de-al treilea introducem cantitatea pe care clientul doreste sa o cumpere .In partea de jos a interfetei mai exista doua butoane , unul pentru vizualizarea comenzilor si celalalt pentru a plasa comanda. Pentru fiecare textField exista cate o metoda care returneaza textul scris in acesta ,de asemenea fiecare buton are o metoda care il returneaza pentru a putea fii folosit in celelate clase. La final avem metoda getAllOrders care creeaza o conexiune cu baza de date si efectueaza un query pentru a extrage toate elementele din tabelul orders , pe care mai apoi le pune intr-un array list pe care il returneaza ,folosit ulterior la popularea JTable ului.

Clasa ViewClients

Reprezinta o interfata grafica simpla compusa dintr-un JPanel , o eticheta care indica faptul ca sunt afisati clienti un, JScrollPane si primeste ca parametru un JTable t pe care il va afisa si care va contine toti clientii

Clasa ViewProducts

Reprezinta o interfata grafica simpla compusa dintr-un JPanel , o eticheta care indica faptul ca sunt afisate produse ,un JScrollPane si primeste ca parametru un JTable t pe care il va afisa si care va contine toate produsele

Clasa ViewOrders

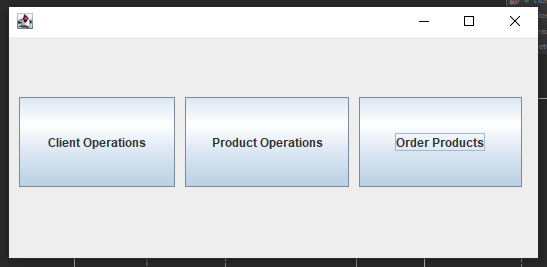
Reprezinta o interfata grafica simpla compusa dintr-un JPanel , o eticheta care indica faptul ca sunt afisate comenzi ,un JScrollPane si primeste ca parametru un JTable t pe care il va afisa si care va contine toate comenzile

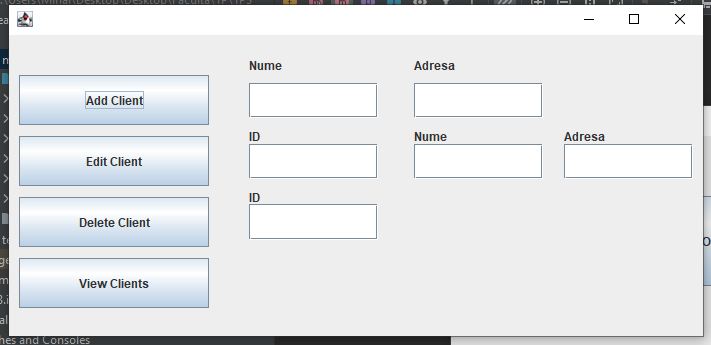
Clasa Controller

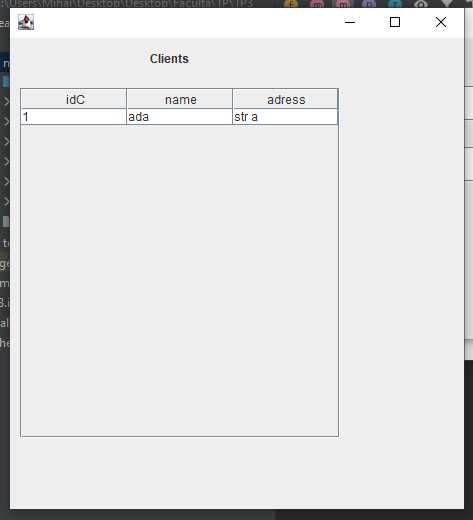
Este responsabila cu managementul interfetelor grafice si functionalitatea acestora . Aceasta contine obiecte de tipul fiecarei interfete grafice si action listener pentru fiecare buton folosit , astfel ,daca apasam butonul Client Operations interfata ClientOP va devenii vizibila , dupa care mai exista patru action listeners pentru fiecare buton al acelei interfete folosite pentru apelarea metodelor insert edit delete si view table. Daca apasam butonul Product Operations interfata ProductOP va devenii vizibila , dupa care mai exista patru action listeners pentru fiecare buton al acelei interfete folosite pentru apelarea metodelor insert edit delete si view table. Daca apasam butonul Order Products interfata OrdersOP va devenii vizibila si aceasta va contine la randul ei doua butoane cu action liteners , unul pentru view table si celalalt pentru plasarea comenzii , daca cantitatea scrisa in textfield la punerea comenzii depaseste stockul disponibil atunci va aparea un mesaj pentru a informa utilizatorul altfel stock ul curent va fii decrementat..

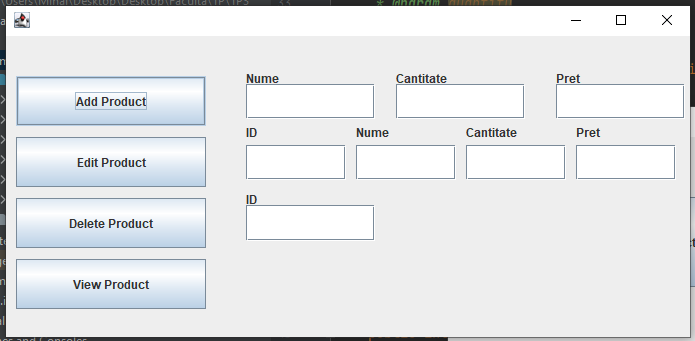
#### Rezultate si concluzii

Rezultatul obtinut consta intr-o aplicatie cu o interfata usor de folosit care simplifica in mod considerabil managementul comenzilor si tinerea evidentei acestora.

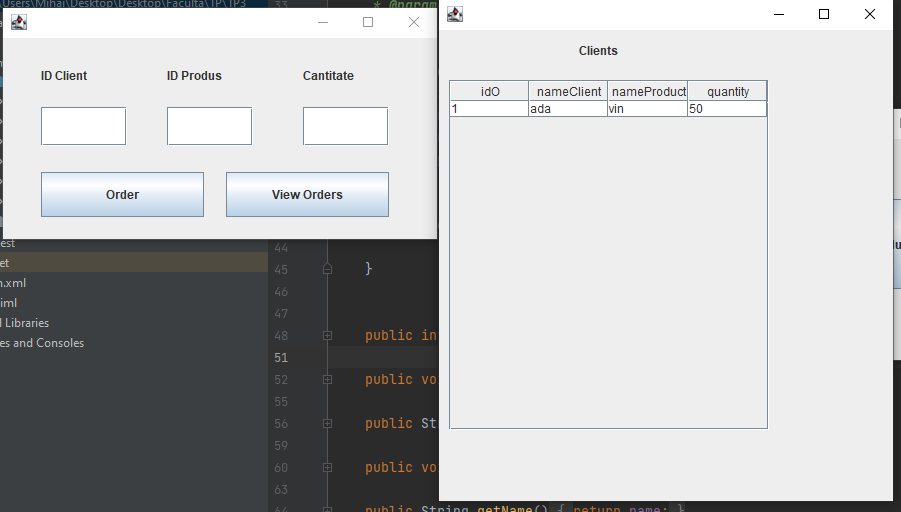












##### Bibliografie

Connect to MySql from a Java application

<https://www.baeldung.com/java-jdbc>

<http://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-reflection/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>