

DOCUMENTATIE

TEMA 3

NUME STUDENT: HOBAN CRISTIAN MIHAI
GRUPA: 30227

CUPRINS

CUPRINS	2
1. Obiectivul temei	3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare	3
3. Proiectare	4
4. Implementare	5
5. Rezultate	11
6. Concluzii	12
7. Bibliografie	12

1. Obiectivul temei

Obiectivul principal al acestei teme este de a dezvolta o aplicație Java care să permită utilizatorilor să introducă și să gestioneze clienți și produse într-o bază de date, precum și să plaseze comenzi folosind aceste informații.

Obiectivele secundare ale aceste teme includ:

1. Implementarea unei interfete grafice prietenoase pentru utilizator, care sa permita introducerea si vizualizarea datelor despre clienti, produse si comenzi;
2. Verificarea introducerii tuturor datelor necesare in momentul in care se realizeaza schimbari in baza de date;
3. Crearea si gestionarea unei baze de date, pentru a stoca informatiile amintite mai sus;
4. Implementarea funcțiilor de adaugare, editare si stergere a clientilor si produselor din baza de date;
5. Realizarea functiei de plasare a comenzilor, tinand cont de clientii existenti, de produsele existente si de stocul curent al acestora;

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Aplicatia trebuie sa permita utilizatorului in primul rand selectarea tabelului asupra caruia vrea sa produca modificari sau pe care doreste sa il vizualizeze prin apasarea unuia din cele 3 butoane dedicate. Spre exemplu, daca a apasat pe butonul cu clienti, acum o noua fereastră se va deschide unde are posibilitatea sa introduca, modifice sau stearga clienti, dar si sa vizualizeze clientii care sunt deja in baza de date

Pasii pe care utilizatorul trebuie sa ii urmeze pentru a introduce un client:

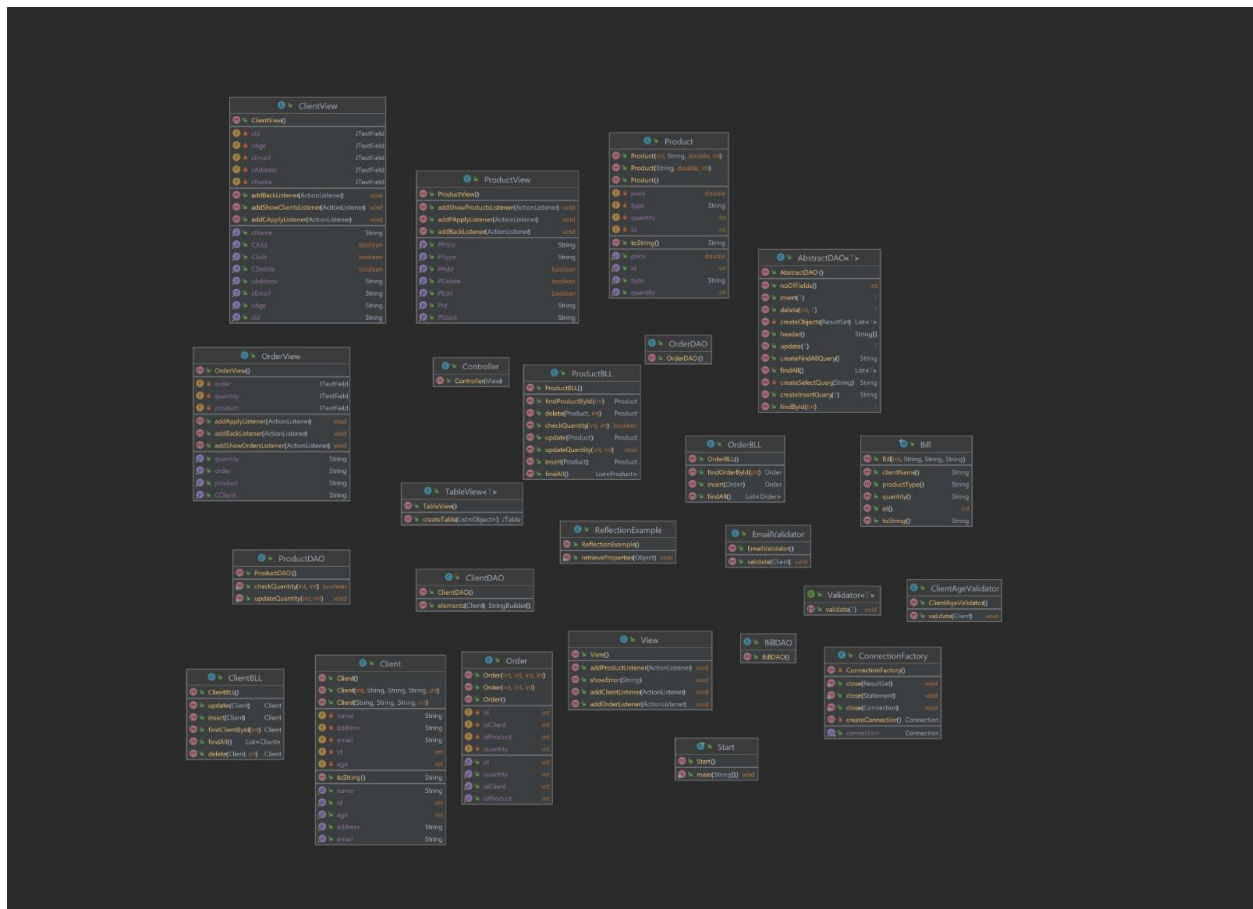
1. Apasa pe butonul “Client”;
2. Se deschide o fereastră noua; trebuie in primul rand sa bifeze checkbox-ul corespunzator etichetei “Add”;
3. Introduce datele despre noul client;
4. Apasa butonul “Apply”, iar daca toate datele au fost introduse se va afisa un mesaj pe ecran cum ca un nou client a fost introdus;
5. Acesta poate fi vizualizat prin apasarea butonului “Show all clients”.

3. Proiectare

Proiectarea OOP reprezinta un model de programare care ofera o buna capacitate de gestionare a unui program.

In acest capitol voi prezenta proiectarea OOP a aplicatiei, in prima faza cu ajutorul unei diagrame UML.

Diagrama UML:



Dupa cum se observa din diagrama, aplicatia este structurata in mai multe clase, fiecare din ele apartinand unui pachet:

- **bll**
 - ClientBLL
 - OrderBLL
 - ProductBLL
- **connection**
 - ConnectionFactory
- **dao**
 - AbstractDAO
 - ClientDAO
 - OrderDAO
 - ProductDAO
- **model**
 - Client
 - Order
 - Product
- **presentation**
 - ClientView
 - Controller
 - OrderView
 - ProductView
 - TableView
 - View
- **start**
 - ReflectionExample
 - Start

4. Implementare

Connection Factory

In aceasta clasa se realizeaza conexiunea dintre codul Java si baza de date;

AbstractDAO

In aceasta clase sunt implementate metode pentru crearea *interogarilor* si si totodata apelarea acestora. Pentru a realiza toate functionalitatile impuse in cerinta, a trebuit sa implementez urmatoarele interogari:

- SELECT
- INSERT
- DELETE
- UPDATE

Chiar teoretic trebuia sa realizez pentru fiecare cate o metoda in parte, m-am folosit de reflexie pentru a implementa o singura metoda pentru fiecare dintre cele 4 query-uri.

Metoda pentru crearea interogarii SELECT:

```
private String createSelectQuery(String field) {  
    StringBuilder sb = new StringBuilder();  
    sb.append("SELECT ");  
    sb.append(" * ");  
    sb.append(" FROM schooldb.");  
    sb.append(type.getSimpleName());  
    sb.append(" WHERE " + field + " =?");  
    return sb.toString();  
}
```

Client, Order, Product

In aceste clase am declarat pentru clienti, comenzi si produse attributele necesare:

Client:

- id
- nume
- adresa
- email
- varsta

Produs:

- id
- tip
- pret
- cantitate in stoc

Comanda:

- id comanda;

- id-ul clientului care plaseaza comanda;
- id-ul produsului pus in comanda;
- numarul de produse cumparate;

ClientBLL, OrderBLL, ProductBLL

In aceste clase, am apelat metodele create in *AbstractDAO* pentru fiecare clasa in parte. Aici a trebuit sa declara un obiect de tip ClientDAO, OrderDAO sau ProductDAO, clase care extind AbstractDAO, pentru a apela metodele prin intermediul acestora.

Apelarea metodelor in ClientBLL:

```

1 usage
public List<Client> findAll(){
    List<Client> c = new ArrayList<>();
    c = clientDAO.findAll();
    return c;
}

1 usage
public Client insert(Client client) throws IllegalAccessException {
    Client c = clientDAO.insert(client);
    return c;
}

1 usage
public Client delete(Client client, int i) throws IllegalAccessException {
    Client c = clientDAO.delete(i, client);
    return c;
}

1 usage
public Client update(Client client) throws IllegalAccessException {
    Client c = clientDAO.update(client);
    return c;
}

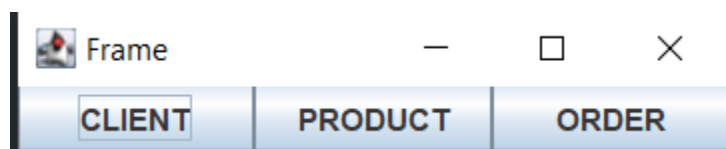
```

Pachetul “presentation”

In acest pachet am implementat mai multe ferestre de interfete grafice astfel:

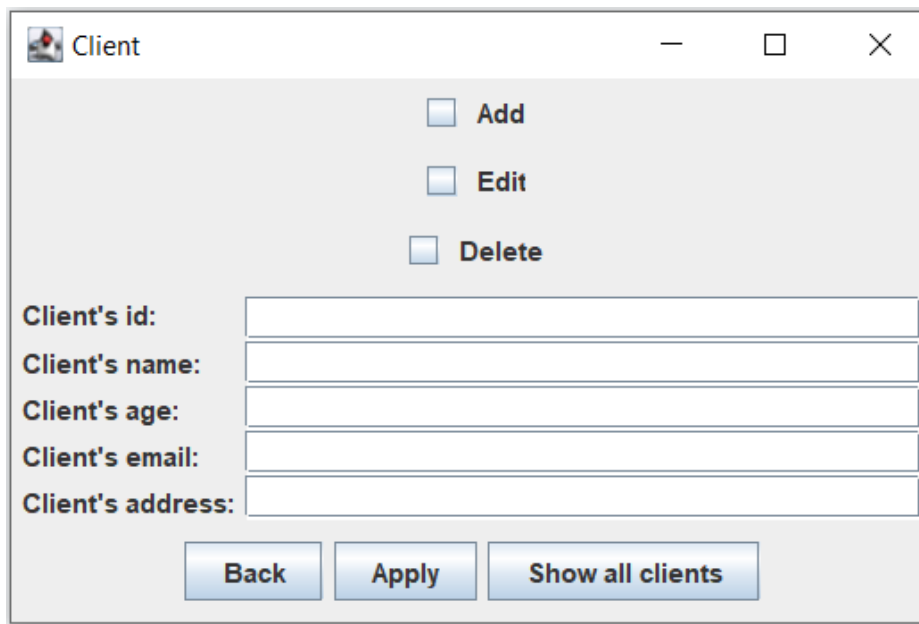
- View

Aceasta afiseaza o interfata cu 3 butoane: “Client”; “Product”; “Order”, fiecare dintre acestea directionandu-te catre alta fereastra in care se pot realiza modificari asupra tabelului selectat prin butonul apasat anterior.



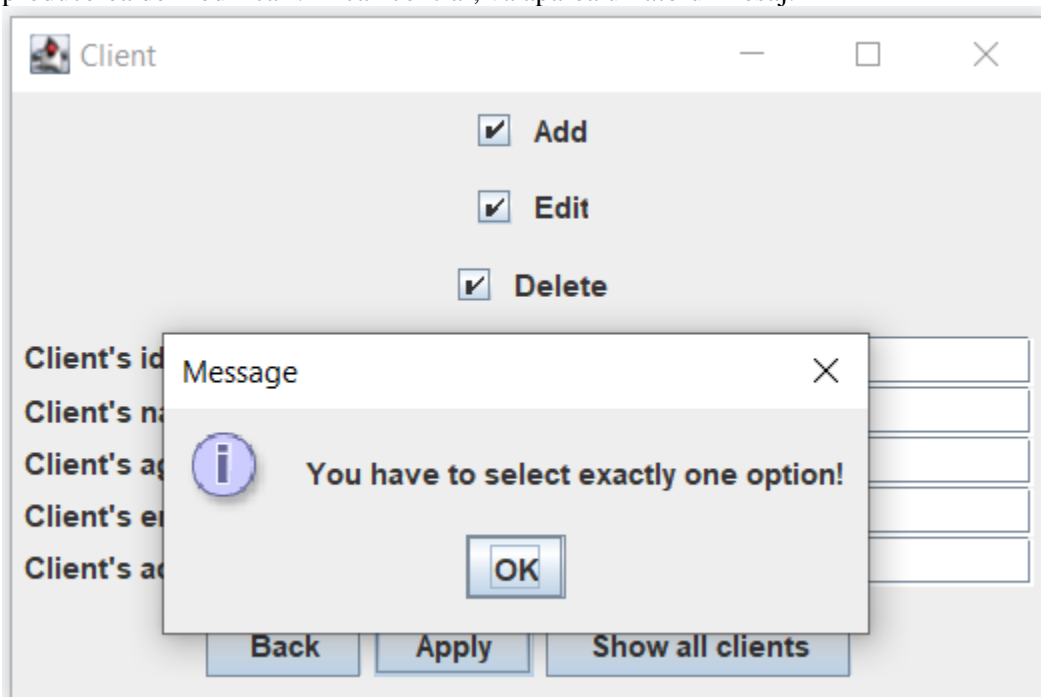
- ClientView

Aici este implementata fereastra care se deschide in momentul apasarii pe butonul “Client”. Fereastra este urmatoarea:



The 'Client' window has a title bar with a standard icon, a minus button, a maximize button, and a close button. The main area contains three radio buttons labeled 'Add', 'Edit', and 'Delete', all of which are currently unselected. Below these are five text input fields, each preceded by a label: 'Client's id:', 'Client's name:', 'Client's age:', 'Client's email:', and 'Client's address:'. At the bottom of the window are three buttons: 'Back', 'Apply', and 'Show all clients'.

Dupa cum spuneam mai sus, este necesar ca doar una din cele trei selectii sa fie bifata pentru a reusi producerea de modificari. In caz contrar, va aparea umatorul mesaj:



This image shows the same 'Client' window as before, but with the 'Add', 'Edit', and 'Delete' radio buttons all selected. A 'Message' dialog box is overlaid on top of the window. The dialog box has a title bar with a close button and contains an information icon (a lowercase 'i' inside a circle) followed by the text 'You have to select exactly one option!'. At the bottom of the dialog box is an 'OK' button. The 'Client' window's input fields and bottom buttons are partially visible behind the dialog box.

Butonul “Back” este pentru momentul in care ai terminat cu modificarile asupra unei anumite tabele, si vrei sa modifichi in alta. Apasarea lui te directioneaza la interfata principala.

Apasarea butonului “Show all clients”, va deschide o fereastra noua, implementata in clasa “TableView”, si aici folosindu-ma de reflexie. Astfel am realizat o singura interfata care este apelata de doate cele 3 tabele.

Interfata arata astfel:



id	name	address	email	age
1	Cristian	strada Apa	hc@yahoo.com	44
2	CDE	wqq	ege	23
35	crstiaisands	dgyyre	sssss@ttt@	12
37	Daniel	Strada strada	dada@yahoo.com	92
38	Dorel	STRSTR	dodododo@yahoo.c...	50

- ProductView

Lucreaza exact ca “ClientView”.

- OrderView

Folosindu-ne de aceasta interfata, putem realiza comenzi cu produse si clienti existenti.

Interfata arata in felul urmatoar:



Order

Orders's id:

Client's id:

Product's id:

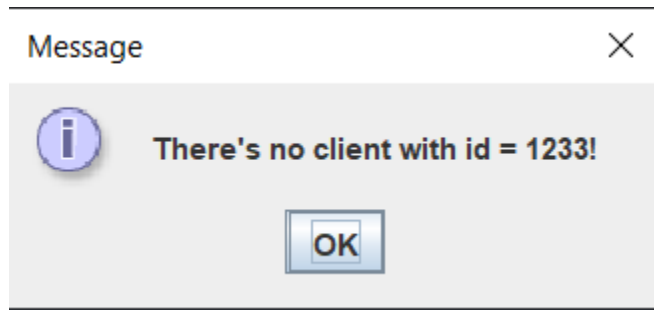
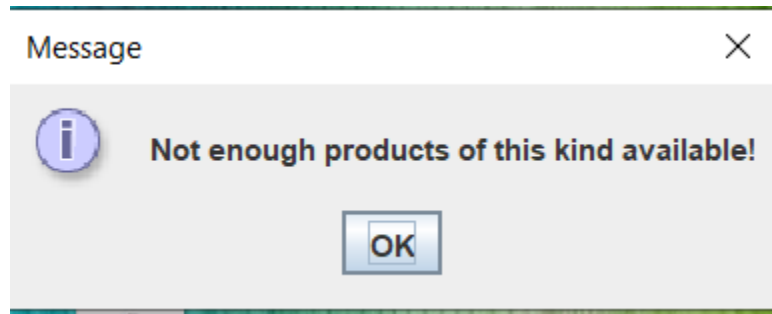
Quantity

Back **Make an order** **Show orders**

Pentru aceasta am realizat anumite validari si anume:

- Este obligatorie introducerea unui client sau unui produs existent;
- Cantitatea introdusa in comanda este trebuie sa fie cel mult egala cu cantitatea precizata in baza de date.

In caz contrar, mesajele urmatoare vor fi afisate:



5. Rezultate

Am testat aplicatia in toate modurile in care m-am gandit, respectand bineinteles cerintele si nu am gasit nicio eroare sau exceptie care sa nu fie tratata.

6. Concluzii

Pe parcursul realizării acestui proiect, am înțeles conceptul de reflexie lucru care la început îmi era destul de ambiguu. Am învățat să conectez un cod java de o bază de date și să realizez interogări care să acționeze asupra acestora.

Ca dezvoltări ulterioare m-am gândit la câteva:

- validarea datelor de intrare: eu până acum am verificat doar dacă datele sunt introduse în tabel și nu ce fel de date au fost introduse;
- realizarea update-ului să se poată realiza și la nivelul unui singur câmp din rând, să nu trebuiască să fii nevoit să introduci tot rândul încă odată;
- să poți realiza o comandă cu mai multe tipuri de produse;

7. Bibliografie

1. <https://dsrl.eu/courses/pt/>
2. <http://www.mkymong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/>
3. <http://tutorials.jenkov.com/java-reflection/index.html>