

Tehnici de programare in Java

Tema 3

Automatica si Calculatoare

Calculatoare si tehnologia informatiei

Tincu Diana

An 2

Grupa 30221

1. Obiectivul temei

Proiectati si implementati o aplicatie de tip OrderManagement pentru procesarea comenzilor unor clienti dintr-un deposit. Bazele de date relationale sunt folosite pentru a stoca produsele, clientii si comenzile acestora. Mai mult, aplicatia create trenuie sa fie structurata in pachete folosind layered architecture descrisa in prezentarea suport a acestei teme si ar trebui sa contina (minim) urmatoarele clase:

- Model classes: represent the data models of the application
- Business Logic classes: contain the application logic
- Presentation classes: GUI related classes
- Data access classes: classes that contain the access to the database

Nota: Alte clase si pachete pot fi adaugate pentru implementarea complet functionala a aplicatiei

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

In implementarea aplicatiei am respectat urmatoarele cerinte:

- Am folosit clase de maximum 300 de linii si metode de maxim 30 de linii si am respectat conventiile Java naming
- Am folosit JavaDoc pentru documentarea claselor
- Am folosit relatiile cu baza de date. Am creat tabelele orders, client si product
- Am creat o interfata grafica pentru operatiile pe client: adaugare client nou, editare client, stergere client, afisarea tuturor clientilor intr-un table (JTable)
- Am creat o interfata grafica pentru operatiile pe produse: adaugare produs nou, editare produs, stergere produs, afisarea tuturor produselor intr-un table (JTable)
- Am creat o interfata grafica pentru operatiile de creare comanda: utilizatorul va fi capabil sa selecteze un produs existent, sa selecteze un client existent, sis a insereze o cantitate dorita a produsului pentru a crea o comanda valida. In cazul in care nu sunt suficiente produse pe stoc, se va afisa un understock message pe ecran. Dup ace comanda este finalizata, stocul produsului respective va fi decrementat.
- Am folosit reflection technique pentru a crea o metoda care primeste o lista de obiecte si genereaza capul tabelii extragand prin reflectie proprietatile obiectului si dupa populeaza tabelul cu valorile elementelor din lista.
- Aplicatia va contine cel putin 4 pachete: dataAccessLayer, businessLayer, model si presentation
- Am creat o facture pentru fiecare comanda sub forma unui fisier de tip text
- Am folosit reflection technique pentru a crea o clasa generic care contine metode pentru accesarea bazei de date: create object, edit object, delete object and find object. Interogariile

pentru accesarea bazei de date pentru un obiect specific care corespunde unuia dintre tabele vor fi generate dynamic prin reflective.

Pentru modelare am creat 5 pachete denumite astfel: bll, connection, dao, model si presentation. Acestea contin clasele:

- bll: ClientBll, OrderBll, ProductBll
- connection: ConnectionFactory
- dao: AbstractDao, ClientDao, OrderDao, ProductDao
- model: Client, Orders, Product, TableHeader, WriteFile
- presentation: AddClient, AddProduct, ClientPresentation, DeleteClient, DeleteProduct, EditClient, EditProduct, MainPresentation, OrderPresentation, ProductPresentation

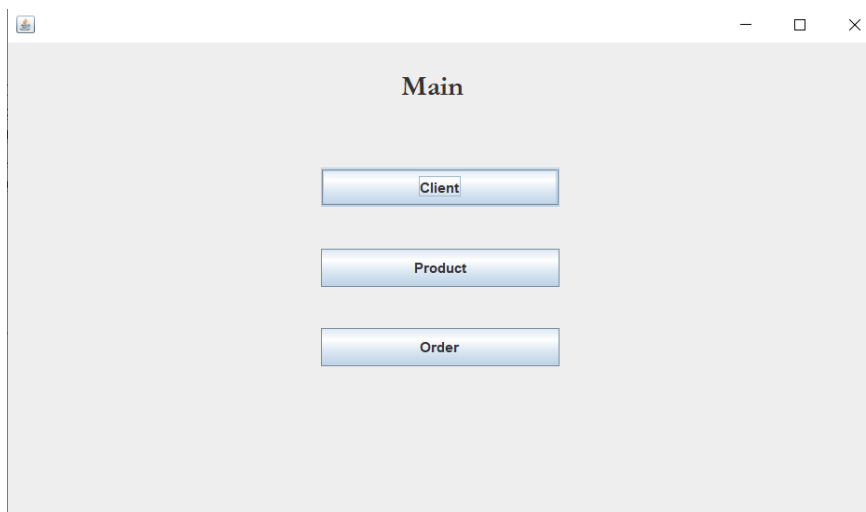
Cazuri speciale apar atunci cand in stoc exista mai putine produse fata de cantitatea introdusa de utilizator in campul specific cantitate din fereastra de plasare a comenzii. Acest caz special il tratam prin afisarea unei casute care transmite un mesaj de understock, utilizatorul fiind astfel fortat sa introduca o cantitate mai mica pentru a plasa cu success comanda.

Alte cazuri speciale care pot sa apara sunt:

- ca utilizatorul care doreste sa comande selecteaza mai multi client sau mai multe produse fapt ce duce la aruncarea unei exceptii deoarece o comanda poate fi facuta de catre un singur client si poate contine doar un tip de produs si cantitatea acestui
- Ca utilizatorul sa nu introduca corect cantitatea produsului, adica sa introduca in casuta pentru cantitate si alte tipuri de caractere fata de cifre fapt ce va duce din nou la aruncarea unei exceptii

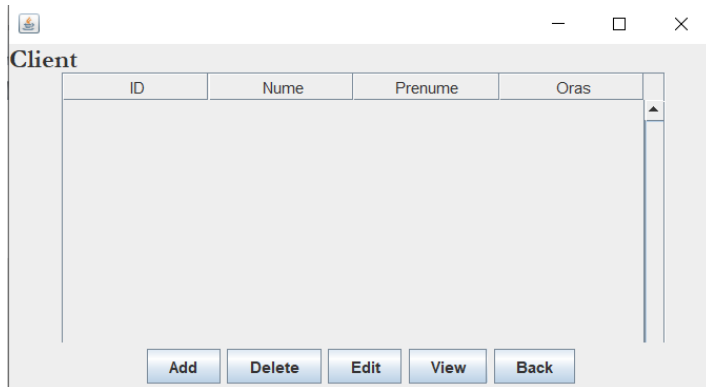
Interfata grafica se va folosi astfel:

Vom rula clasa MainPresentation si se va deschide interfata:



Din aceasta interfata putem selecta ce fel de operatii dorim sa facem (pe tabla client, pe tabela product sau pe tabela orders).

Daca apasam butonul client urmatoarea fereastra se va deschide:

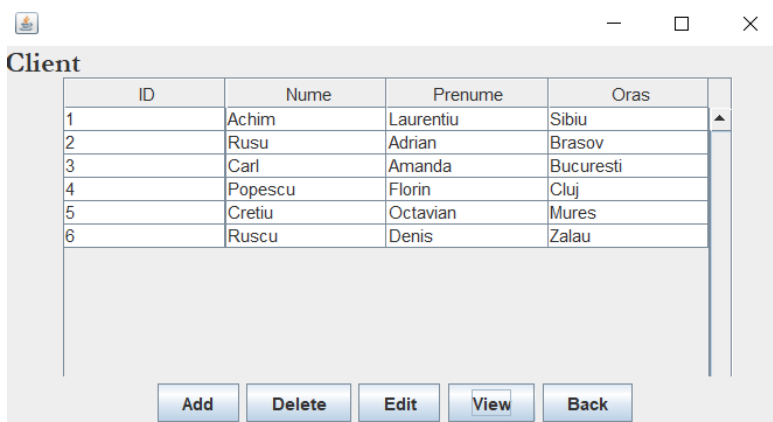


Client

ID	Nume	Prenume	Oras
----	------	---------	------

Add Delete Edit View Back

Pentru a vedea clientii din baza de date apasam butonul View:

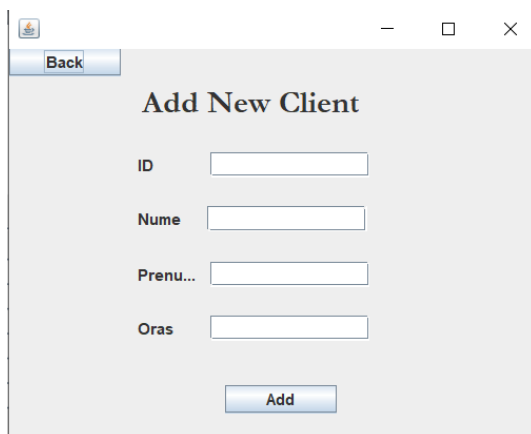


Client

ID	Nume	Prenume	Oras
1	Achim	Laurentiu	Sibiu
2	Rusu	Adrian	Brasov
3	Carl	Amanda	Bucuresti
4	Popescu	Florin	Cluj
5	Cretiu	Octavian	Mures
6	Ruscu	Denis	Zalau

Add Delete Edit View Back

Pentru a adauga un client nou vom apasa butonul de Add, ni se va deschide o interfata precum cea de mai jos in care vom introduce in fiecare casuta datele despre client. Nici o casuta nu va fi lasata goala!



Back

Add New Client

ID

Nume

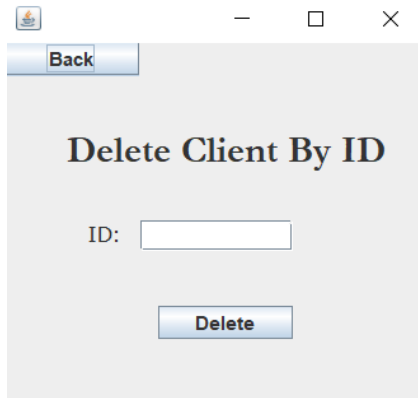
Prenu...

Oras

Add

Pentru a ne intoarce inapoi in ClientPresentation apasam butonul de Back din coltul stanga sus.

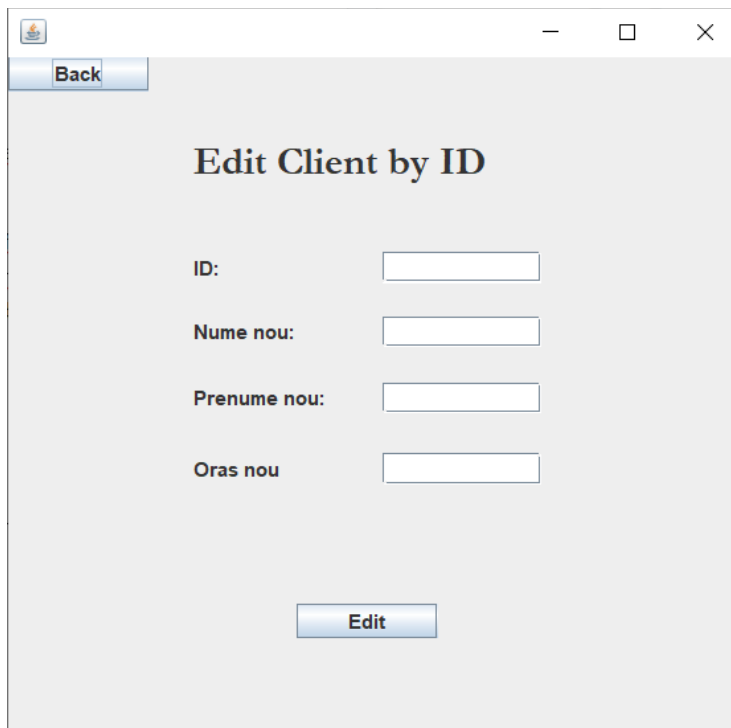
Pentru a sterge un client vom apasa pe butonul Delete si i se va deschide o fereastră in care se va cere sa introducem id-ul clientului pe care dorim sa il stergem din baza noastra de date.



The screenshot shows a Windows-style dialog box titled "Delete Client By ID". It has a "Back" button in the top-left corner. The main content area contains a label "ID:" followed by a text input field. Below the input field is a "Delete" button.

Pentru a ne intoarce inapoi in ClientPresentation apasam din nou butonul din coltul stanga sus al ferestrei de Delete.

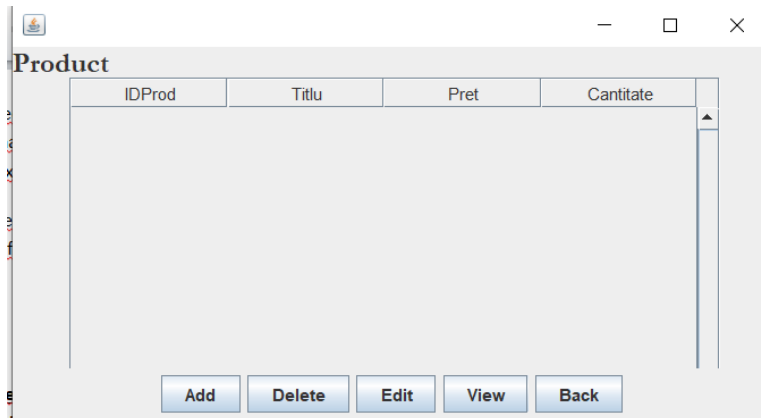
Pentru a edita un client din baza de date, apasam pe butonul Edit, si se va deschide o fereastră in care vom introduce ID-ul clientului caruia dorim sa ii schimbam datele, si noile date ale clientului pe care dorim sa i le asociem. Pentru ca editarea sa fie efectuata trebuie sa apasam butonul Edit din josul ferestrei.



The screenshot shows a Windows-style dialog box titled "Edit Client by ID". It has a "Back" button in the top-left corner. The main content area contains four labels with corresponding text input fields: "ID:", "Nume nou:", "Prenume nou:", and "Oras nou". At the bottom center of the dialog is an "Edit" button.

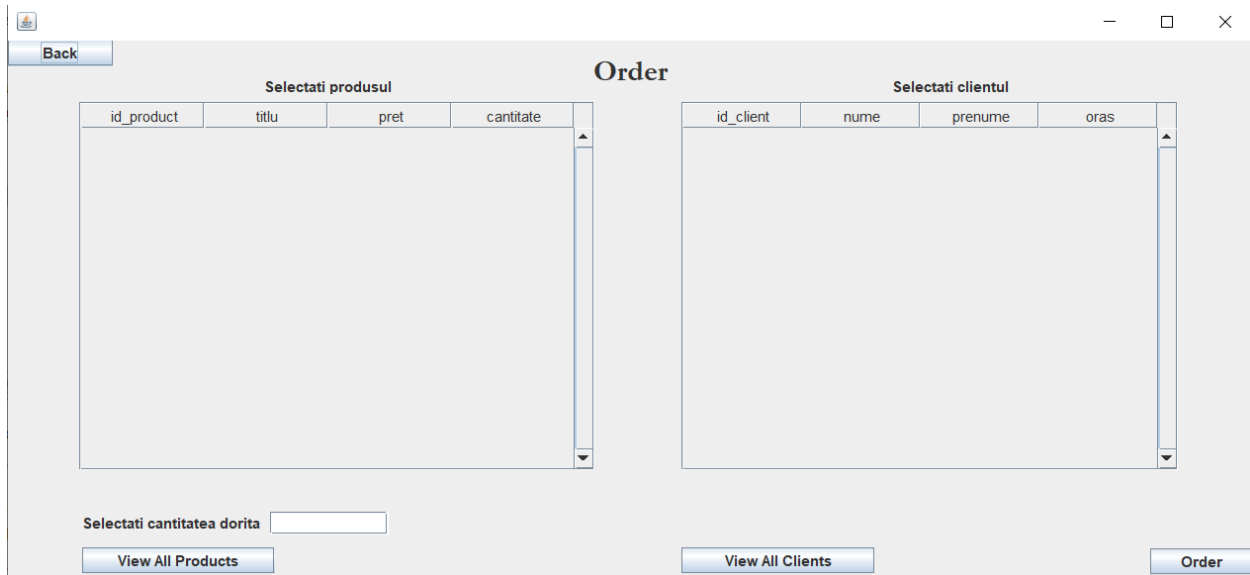
Pentru a selecta un alt tip de operatie apasam butonul de Back din fereastra ClientPresentation si ne intoarcem inapoi la fereastra de MainPresentation din care selectam urmatorul tip de operatie pe care dorim s ail executam.

Daca am selectat butonul Product se va deschide o fereastra din care putem selecta operatia pe care dorim sa o efectuam asupra tabelului product din baza de date:



Aceasta fereastra este similara cu cea pentru operatiile pe tabela client. Asadar butonul de Add deschide o noua fereastra care este destinata pentru adaugarea unui nou produs in tabela product din baza de date unde atributele produsului sunt preluate din campurile respective, butonul Delete este pentru a sterge un produs din tabela product din baza de date care are id- ul egal cu cel scris in casuta de text din fereastra de Delete Product By ID, butonul de View afiseaza in tabela de deasupra butoanelor din interfața Product toate produsele care se gasesc in baza de date in tabelul product. Butonul de Back ne intoarce inapoi in fereastra Main.

Daca selectam butonul de Order din fereastra Main se va deschide o fereastra din care vom selecta ce operatii dorim sa executam pe tabela orders din baza de date:



Pentru a plasa o comanda trebuie sa afisam clientii si produsele in tabelele respective din interfata. Primul tabel (cel din stanga) va contine produsele iar al doilea va contine clientii. Pentru a popula tabelele vom apasa pe butoanele View All Products si respective View All Clients. Dupa aceasta vom selecta cate un produs si cate un client si vom introduce in casuta pentru cantitate cantitatea dorita a produsului si la final apasam butonul de Order din coltul dreapta jos pentru a plasa comanda si pentru a o insera in baza de date si a crea chitanta.

3. Proiectare (decizii de proiectare, diagrame UML, structuride date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

Structuri de date folosite:

In clasa ClientBll am folosit o lista de client List<Client> pentru implementarea metodei ViewAllClients care afiseaza toti clientii din tabela client din baza de date.

In clasa ProductBll am folosit o lista de produse List<Product> pentru implementarea metodei ViewAllProducts care afiseaza din tabela product din baza de date.

In clasa AbstractDao am folosit un ArrayList<T> in care am adaugat instantele obiectului de tip T.

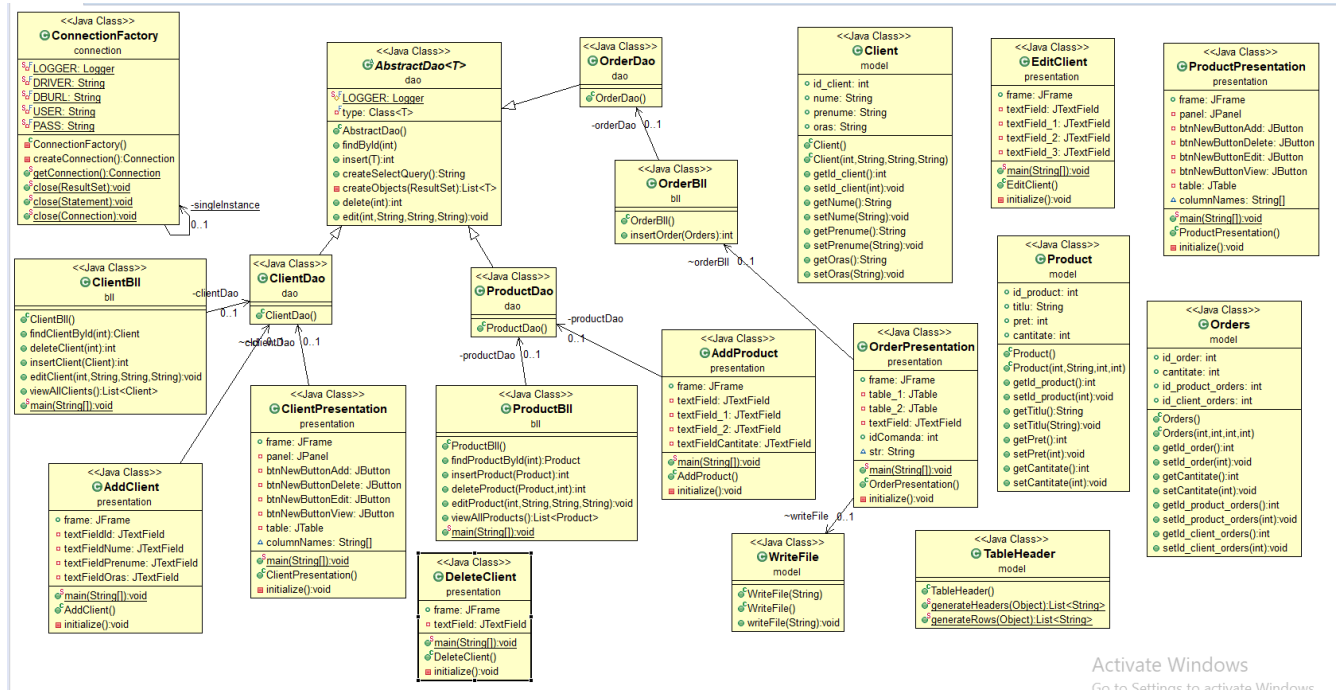
In clasa TableHeader am folosit o lista de tip ArrayList<String> pe care am folosit- o pentru a genera capul tabelii si pentru a popula tabelul respectiv.

Clasele create:

- ClientBll
- OrderBll
- ProductBll
- ConnectionFactory
- AbstractDao
- ClientDao
- OrderDao
- ProductDao
- Client
- Orders
- Product
- TableHeader
- WriteFile
- AddClient
- AddProduct
- ClientPresentation
- DeleteClient
- EditClient

- EditProduct
- MainPresentation
- OrderPresentation
- ProductPresentation

Diagrama UML rezultata:



4. Implementare

ClientBill

In aceasta clasa am implementat metodele pentru operatiile pe tabela client: findClientById , deleteClient, insertClient, editClient, ViewAllClients.

Aceasta clasa are ca si variabila de instanta pe clientDao care este de tipul ClientDao.

OrderBill

In aceasta clasa am implementat o metoda pentru inserarea in tabelul orders din baza de date.

Aceasta clasa are ca si variabila de instanta pe orderDao care este de tipul OrderDao.

ProductBll

In aceasta clasa am implementat metodele pentru operatiile pe tabela product: findProductById , deleteProduct, insertProduct, editProduct, ViewAllProduct.

Aceasta clasa are ca si variabila de instanta pe productDao care este de tipul ProductDao.

ConnectionFactory

In aceasta clasa am creat legatura cu baza de date si asadar am legat proiectul nostru la baza de date.

In aceasta clasa avem:

- O metode pentru crearea legaturii: createConnection() care returneaza conexiunea create
- Un getter pentru conexiune getConnection() care returneaza conexiunea
- O metoda close() care are un parametru de tipul ResultSet si nu returneaza nimic ci apeleaza metoda close pentru un obiect de tipul ResultSet
- O metoda close() care are un parametru de tipul Statement si nu returneaza nimic ci apeleaza metoda close pentru un obiect de tipul Statement
- O metoda close() care are un parametru de tipul Connection si nu returneaza nimic ci apeleaza metoda close pentru un obiect de tipul Connection

AbstractDao

```
public abstract class AbstractDao<T>
```

Aceasta este o clasa abstracta generica. Ea contine metode care implementeaza operatiile pe tabelele din baza de date in mod generic. Metodele putand fi apelate pe oricare dintre tabelele client, product sau orders.

Metodele pe care aceasta clasa le contine sunt:

- findById:

```
public T findById(int givenId)
```

Aceasta metoda cauta in baza de date obiectul care are id-ul egal cu givenId

- insert:

```
public int insert(T object)
```

Aceasta metoda insereaza in baza de date in tabelul corespunzator obiectul de tipul T

- createSelectQuery

```
public String createSelectQuery()
```

Aceasta metoda executa o interogare pe un tabel din baza de date si anume selecteaza toate liniile din tabela care au id-ul egal cu cel specificat la momentul apelarii metodei

- createObject

```
private List<T> createObjects(ResultSet resultSet)
```

Aceasta metoda ia rezultatul unei interogari si il transforma intr-o lista de obiecte pe care mai apoi o returneaza

- delete

```
public int delete(int id)
```

Aceasta metoda sterge din tabelul din baza de date obiectul cu id-ul specificat ca si parametru

- edit

```
public void edit(int id, String s1, String s2, String s3)
```

Aceasta metoda editeaza obiectul dintr-o tabela care are id-ul dat ca prim parametru cu valorile date de urmatorii trei parametrii s1, s2 si s3.

ClientDao

Aceasta clasa extinde clasa AbstractDao.

Aceasta clasa contine un constructor care apeleaza super() pentru clasa parinte AbstractDao.

OrderDao

Aceasta clasa extinde clasa AbstractDao.

Aceasta clasa contine un constructor care apeleaza super() pentru clasa parinte AbstractDao.

ProductDao

Aceasta clasa extinde clasa AbstractDao.

Aceasta clasa contine un constructor care apeleaza super() pentru clasa parinte AbstractDao.

Client

Aceasta clasa corespunde cu tabela client si reprezinta un obiect/ o linie din tabela client.

Ea contine attributele unui client care se gasesc si in tabela client din baza de date, anume: id_client, nume, prenume si oras.

Pentru fiecare atribut am creat un getter si un setter.

Orders

Aceasta clasa corespunde cu tabela orders si reprezinta un obiect/ o linie din tabela orders.

Ea contine attributele unei comenzi care se gaseste si in tabela orders din baza de date, anume: id_order, cantitate, id_product_orders si id_client_orders.

Pentru fiecare atribut am creat un getter si un setter.

Tot in aceasta clasa am creat si un constructor.

Product

Aceasta clasa corespunde cu tabela product si reprezinta un obiect/ o linie din tabela product.

Ea contine attributele unui produs care se gaseste si in tabela product din baza de date, anume: id_product, titlu, pret si cantitate.

Pentru fiecare atribut am creat un getter si un setter.

TableHeader

Aceasta clasa contine metode pentru generarea capului de tabel si a liniilor acestuia pentru a-l popula. Se utilizeaza reflection technique pentru a genera attributele folosind proprietatile obiectului transmis ca si parametru.

WriteFile

Aceasta clasa contine o metoda care creeaza un fisier bill.txt pentru a genera o chitanta la fiecare comanda.

5. Rezultate

Dupa fiecare plasare de comanda se va genera o chitanta care va fi de forma celei de mai jos:

```
Nr comanda: 1
->ID client:6
->ID produs: 6
->Cantitatea: 1
Total plata: 10

Nr comanda: 2
->ID client:6
->ID produs: 6
->Cantitatea: 1
Total plata: 10

Nr comanda: 3
->ID client:6
->ID produs: 6
->Cantitatea: 1
Total plata: 10
```

6. Concluzii

In concluzie am reusit sa implementez o aplicatie care sa se ocupe de managementul comenzilor unui produs si care sa aiba o interfata grafica prietenoasa si usor de inteles pentru orice utilizator.

7. Bibliografie

<https://www.roseindia.net/java/example/java/swing/jtable-display-database-data.shtml>

<https://stackoverflow.com/questions/38336113/jtable-if-any-row-is-selected>

<https://stackoverflow.com/questions/29345792/java-jtable-getting-the-data-of-the-selected-row>

<https://www.programcreek.com/java-api-examples/?class=javax.swing.JTable&method=getSelectedRow>

<https://stackoverflow.com/questions/67063964/java-beans-introspectionexception-method-not-found-isidc-propertydescriptor>

<https://docs.oracle.com/javase/7/docs/api/java/beans/PropertyDescriptor.html>

<https://www.javatpoint.com/java-int-to-string>

<https://stackoverflow.com/questions/7080205/popup-message-boxes>

<https://countwordsfree.com/comparetexts>