**ASSIGNMENT 3**

**ORDER MANAGEMENT**

**NUME : OSTAFIE STANCA**

**GRUPA: 30227**

**PROFESOR LABORATOR : DAN MITREA**

CUPRINS

[1. OBIECTIVUL TEMEI 3](#_Toc66881492)

[2. ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE 3](#_Toc66881493)

[3. PROIECTARE 4](#_Toc66881494)

[4. IMPLEMENTARE 6](#_Toc66881495)

[5. REZULTATE 7](#_Toc66881496)

[6. CONCLUZII 7](#_Toc66881497)

[7. BIBLIOGRAFIE 8](#_Toc66881498)

# 1. **OBIECTIVUL TEMEI**

Obiectivul principal al acestei teme este proiectarea si implementarea corecta a unei aplicatii ce are ca scop gestiunea comenzilor plasate de catre clienti la un depozit. Pentru implementarea acestei cerinte vom folosi baze de date relationale pentru a stoca informatiile despre clienti, produse si comenzi.

Vom amintii si urmatoarele obiective secundare ale temei:

* Implementarea clasei “AbstractDAO” ne permite sa cream instructiuniile necesare operatiilor cu baze de date, dintre care amitin : SELECT \*, INSERT, UPDATE si DELETE, cat si sa executam instructiuniile pe tabelele noastre : Client, Product si Comanda.
* Implementarea calselor “ClientBLL” , “ProductBLL” si “OrderBLL” este utila pentru apelarea operatiilor specifice fiecarui obicet in parte. Astfel ca fiecare clasa va avea un atribut cu extensia DAO, atribut care ne va permite sa apelam toate functiile din clasa “AbstractDAO”, dar de aceasta data cu un anumit obiect, dorit de noi.
* Implemnetarea claselor “View”, “ViewClient”, “ViewProduct” si “ViewOrder” ne ajuta sa obtinem o interfata adecvata, fiecare obiect avand propria sa fereastra pentru afisare si executarea operatiilor dorite.

# **2. ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE**

* **Analiza problemei**

Scopul acestei teme este de a implementa o aplicatie menita sa permita inserarea, stergerea sau modificarea unor linii din tabelele noastre ; Client, Product, Order ; create in baza de date mydb. De asemenea, folosind tehnica de reflection, aplicatia trebuie sa stie sa creeze un tabel cu toate tuplele din cele 3 tabele mentionate, individual pentru fiecare tabela in parte.

* **Modelare**

Este important de retinut ca utilizatorul va furniza datele necesare operatiilor, astfel ca trebuie completate toate campurile din interfata. Pentru o functionare corecta, datele trebuie introdusecorespunzator, in functie de menirea fiecarui camp, exemplu: pentru campruile Name se va introduce un String, pentru campurile Id se va introduce un numar intreg etc. Datele din interfata sunt preluate sub forma de text si transformate in tip-uri corespunzatoare. Odata pornita aplicatia trebuie sa alegem una din cele 3 orptiuni : Client, pentru a efectua operatii pe tabela Client; Product , pentru a se efectua operatii pe tabela Product si Order, pentru a se efectua operatii pe tabela Comanda.

* **Scenariul**

La rularea aplicatiei se va deschide o interfata grafica in care ni se va cere sa selectam una din cele 3 optiuni existente : Client, Product si Order. Fiecare optiune va deschide o intefata grafica noua in care se vor putea efectua operatiile pe diferite obiecte : Client, Product sau Comanda. In noua interfata deschisa utilizatorul va introduce valori in campurile prezentate in functie de specificul numelui. Astefl daca, pentru a efectua operatiile de insert si update utilizatorul trebuie sa asigure o valoare valida, existenta in baza de date, pentru campul id, cat si valori potrivite pentru celelalte campuri. De retinut ca pentru aceste doua operatii niciun camp nu poate ramne gol, in cazul in care se omite cel putin un camp se va genera un mesaj de eroare. Pentru operatia de delete este suficient sa completam doar campul “id” deoarece stergerea se va face in functie de acest camp, totusi valoarea pentru id trebuie sa fie una valida, adica sa existe in tabela noastra. Pentru a verifica corectitudinea datelor introduse in campul id, exista functia de ViewAll care va prezenta inreg continutul tabelei.

* **Cazuri de utilizare**

La rularea acestei aplicatii se va deschide o intefata grafica ce este compusa din trei butoane JButton. Design-ul pentru aceasta interfata este unul simplu si usor de inteles de catre orice utilizator. Prin aplicarea unui eveniment asupra acestor butoane fiecare va deschide un nou JFrame in care vom putea sa manipulam informatiile despre clienti, produse sau comenzi. JFrame-ul pentru Client si Product este alcatuit din 4 JTextField-uri si 4 JButton, iar JFrame-ul pentru clasa Coamnda este alcatuit din 4 JButton, 2 JTextField-uri si 2 JComboBox. Datele din fiecare intefata vor fi preluate si modelate in conformitatea cu cerintele problemei,. Fiecare buton are un nume sugectiv ceea ce va facilita munca cu aceasta interfata.

Dupa aplicarea unui eveniment, click, aspra unuia din cele trei butoane din JFrame-ul principal,utilizatorul va avea acces la o noua fereastra JFrame in care poate sa efectueze operatii SQL pe una din cele 3 tabela existente. Pentru fereastra Client se va putea vizualiza continutul tabelei, insera un nou cliet, modifica sau sterge. Aceste operatii sunt accesibile si in cazul celorlalte doua ferestre. Singura deosebire apare la nivelul ferestrei Comanda unde pentru IdClient si IdProduct vom avea cate o component JComboBox in care se vor putea selecta doar id-uri existente deja in tabela Client si Product.

# 3. PROIECTARE

Acest proiect respecta arhitecura clasica Model View Controller (MVC), fiind impartiti in mai multe pachete denumite adecvat. Inpachetul “model” avem implementate clasele “Client” , “Product”, “Comanda”, in pachetul “dao” avem implementate clasa generica “AbstractDAO” si clasele care mostenesc aceasta clasa, insa cu obiectele definite in pachetul “model”, “ClientDAO”, “ProductDAO”, “OrderDAO”.In pachetul “bll” avem implementate clasele “ClientBLL”, “ProductBLL” si “OrderBLL”. In pachetul “connection” este implementata clasa “ConnectionFactory”, iar in clasa “presentation” avem clasele ce alcatuiesc interfata grafica “View”, “ViewClient”, “ViewProduct”, “ViewOrder” si “Controller”.

Principala clasa a acestui proiect este clasa “AbstractDAO” in care avem functii ce creeaza textul necesar pentru fiecare interogare, precum si functiile care vor executa aceste interogari. Principalul aspect al acestei clase este faptul ca este definita pe un tip de date generic, fapt ce face posibila inlocuirea tipului generic cu orice obiect, fara a fi nevoie sa repetam acelasi cod de mai mule ori. Aceasta clasa este mostenita de catre clasele “ClientDAO” ce va inlocui tipul generic cu un obiect de clasa Client, “ProductDAO” ce inlocuieste tipul de date generic cu un obiect de clasa “Product”, si “OrderDAO” ce inlocuieste tipul de date generic cu un obiect “Comanda”. Fiind definite astfel, toate obiectele de aceasta clasa vor putea sa execute codul descris in AbstractDAO fara modificari.

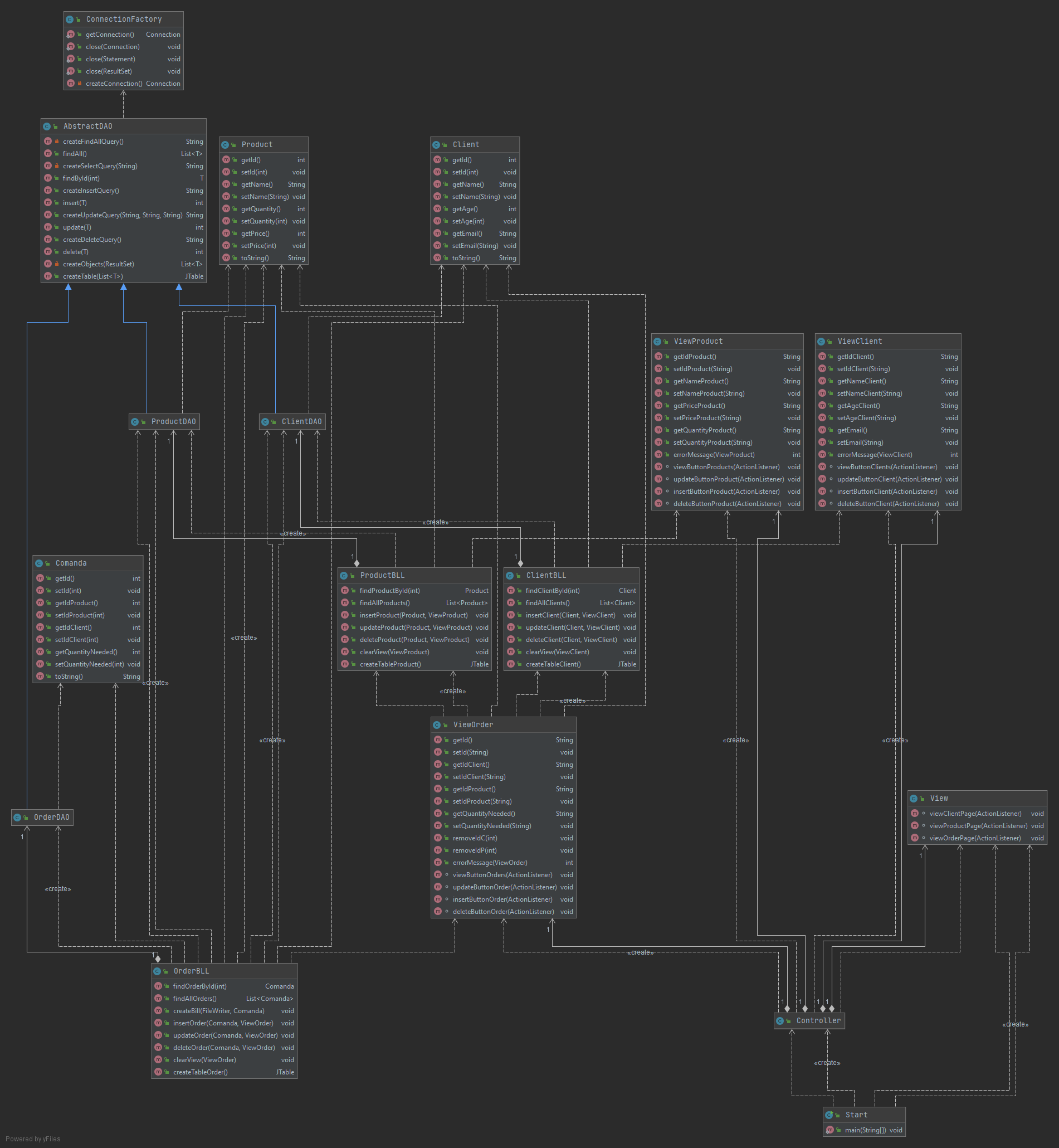
In pachetul “model” dispunem de clasele “Client”, “Product” si “Comanda”. Aceste clasele au ca atribute numele coloanelor definite in baza de date. Astfel ca pentru clasa “Client” vom avea atributele : id, name, age, email; pentru clasa “Product” avem atributele : id, name, price, quantity; iar pentru clasa “Comanda” avem atributele : id, idClient, idProduct si quantityNeeded. In aceste clase avem implementate settere si gettere pentru fiecare atriut in parte.

Pachetul “bll” contine clasele “ClientBLL”, “ProductBLL” si “OrderBLL”. Aceste clase contin functii ce apeleaza functiile implementate in clasa “AbstractDAO” pentru fiecare tip de obiect in parte : client, product si comanda. Fiecare clasa are ca atribut un obiect de clasa “ClientDAO”, “ProductDAO” sau “OrderDAO”. In fiecare clasa vom avea o functie pentru : cautarea unui obiect dupa campul id, selectarea intregului continut dintr-o tabela, inserarea unui nou obiect, modificarea unui obiect existent si stergerea unui obiect existent in tabela.

Clasa View contine elemente de structura necesare pentru interfata grafica a proiectului. Aceasta face interactiunea cu aplicatia mai practica si mai rentabila pentru utilizator. In cadrul acestei clase am utilizat elemente precum : JTextField( utilizat pentru prelucrarea datelor), JButton, JPanel( intermediar), JFrame, JComboBox, JTable. In clasele “ViewClient”, “ViewProduct” si “ViewOrder” decidem modul in care aranjam mare parte din aceste structuri in interfata grafica.

Clasa Controller este clasa in care se traduce interactiunea utlizatorului cu interfata grafica in actiuni ce vor si executate de catre partea de Model .

Clasa “Start” este cea in care se leaga toate componentele mentionate mai sus, permitandu-ne sa rulam aplicatia.

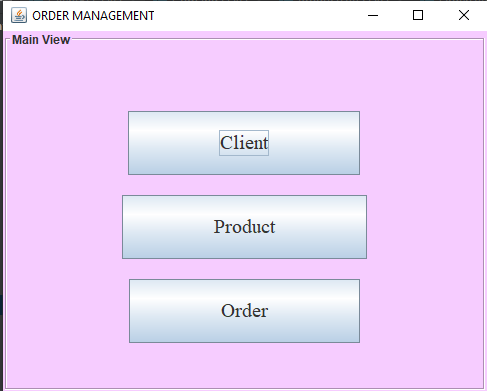


# 4. IMPLEMENTARE

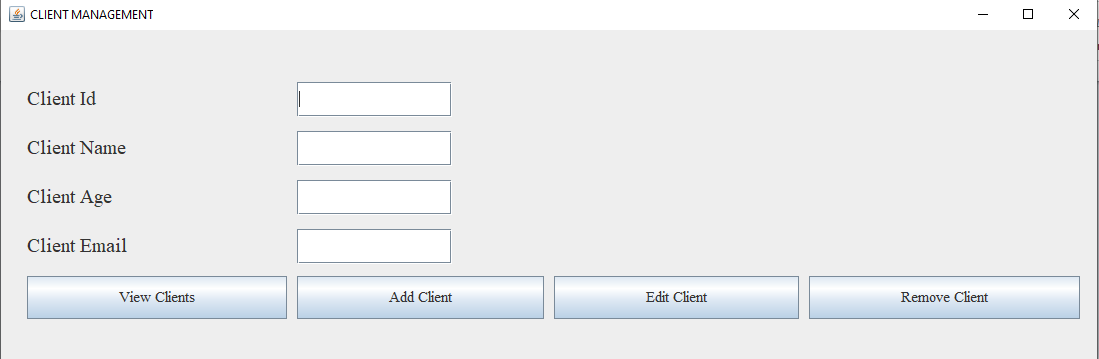
* Clasa “Client” are patru atribute : atributul id (int), name(String), age (int), email(String). Pentru aceste atribute avem implementate metode de set si get cu ajutorul carora setam valoarea unui atribut sau obtinem valoarea stocata in acesta. Tot in aceasta clasa avem suprascrisa metodat toString().
* Clasa “Product” ” are patru atribute : atributul id (int), name(String), price (int), quantity(int). Pentru aceste atribute avem implementate metode de set si get cu ajutorul carora setam valoarea unui atribut sau obtinem valoarea stocata in acesta. Tot in aceasta clasa avem suprascrisa metodat toString().
* Clasa “Comanda” ” are patru atribute : atributul id (int), idClient(int), idProduct (int), quantityNeeded(int). Pentru aceste atribute avem implementate metode de set si get cu ajutorul carora setam valoarea unui atribut sau obtinem valoarea stocata in acesta. Tot in aceasta clasa avem suprascrisa metodat toString().
* Clasa “AbstractDAO” are ca atribut un obiect generic de clasa T numit type. Pentru acest obiect de clasa T se vor defini functii ce executa interogari intr-o baza de date. Dintre aceste functii vom aminti :
* public List<T> findAll() : aceasta functie creeaza un query specific selectarii tuturor tuplelor dintr-o tablea astfel : SELECT \* FROM type.getSimpleName(). In aceasta functie se creeaza o conexiune la baza de date mydb si se executa statement-ul specific acestei operatii. In urma executiei va rezulta o lista de obiecte T ce va fi returnata de catre funtie.
* public int insert(T t) : aceasta functie insereaza un obiect de tipul T la tabela cu acelasi nume astfel : INSERT INTO type.getSimpleName() VALUES(?,?,?,?). Functia are ca parametru un obiect de clasa T pe care dorim s ail inseram in tabela noastra. Pentru a efectua aceasta operatie cream o conexiune la baza de date, stabilim interogarea iar pe urma ii setam paramterii pe care urmeaza sa ii inseram exact cum apar in tabela noastra.
* public int update(T t) : aceasta functie are ca rol modificarea unui obiect de clasa T ce are id-ul egal cu cel al parametrului transmis astfel : UPDATE type.getSimpleName() SET field1 = ?, field2 =?, field3=? WHERE id = ?. Pentru a efectua aceasta operatie se extrag valorile din campurile parametrului t si se transmit instructiunii prezentate mai sus. Functia va returna numarul de linii afectate de aceasta instructiune. Deoarece face modificari in functie de cheia primara aceasta functie va returna valoarea 1 daca reuseste sa modifice cu success obiectul transmis.
* public int delete(T t) : aceasta functie doreste sa elimine un obiect de tabela in functie de valoarea campului id. Se va crea o conexiune la baza de date, se va extrage valoarea pentru campul id si se va executa urmatoarea instructiune : DELET FROM type.getSimpleName() WHERE id = ?. Functia va returna 1 daca stergerea s-a efectuat cu succes si 0 in caz contrar.
* Clasele “ClientBLL”, “ProductBLL” si “OrderBLL” au o functionalitate asemanatoare astfel ca vom prezenta clasa “OrderBLL”. Aceasta clasa are ca atribut un obiect de clasa OrderDAO numit orderDAO. Vom prezenta doar cateva din functiile implementate aici :
* public void insertOrder(Comanda comanda) throws IOException : aceasta functie insereaza o noua Comanda la tabela cu acelasi nume din baza de date. Special pentru aceasta functie este faptul ca la fiecare comanda plasata cu success trebuie sa generam o chitanta si sa modificam cantitatea de care dispune produsul din comanda. Pentru inceput ne asiguram ca nu este violate conditia de cheie primara, in alte cuvine : ca nu inceracam sa inseram o comanda cu un id deja existent in tabela. La pasul urmator ne asiguram ca produsul pe care dorim sa il adaugam la comanda exista iar cantitatea sa este suficienta pentru a plasa comanda. Cu alte cuvine: verificam ca cantitatea necesara pentru comanda sa fie mai mica decat cantitatea de care dispune produsul. Daca aceste 2 conditii sunt indeplinite incercam sa inseram produsul in tabel. Daca functia returneaza o valoare diferita de 0 atunci generam chitanta pentru comanda si modificam cantitatea produsului.
* Clasele “View”, “ViewClient”, “ViewProduct” si “ViewOrder” genereaza partea de interfata grafica ce se poate vizualiza la rularea aplicatiei. In aceaste clase se descrie modul de aranjare al componentelor in interfetele grafice si adaugam ActioListener() pentru butoane.
* Clasa “Controller” se coupa cu implementarea “evenimentelor” in momentul in care se efectueaza o actiune asupra unei componente din interfata ce genereaza un ActionListener().

# 5. REZULTATE

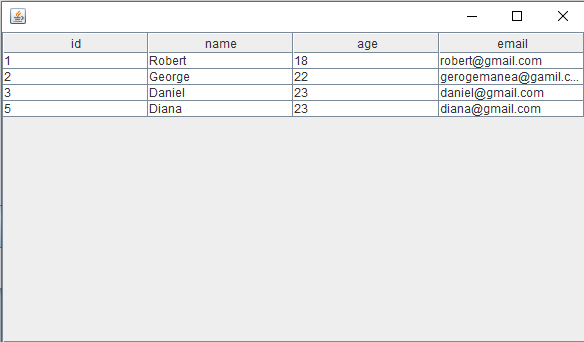
* Fereastra principal :



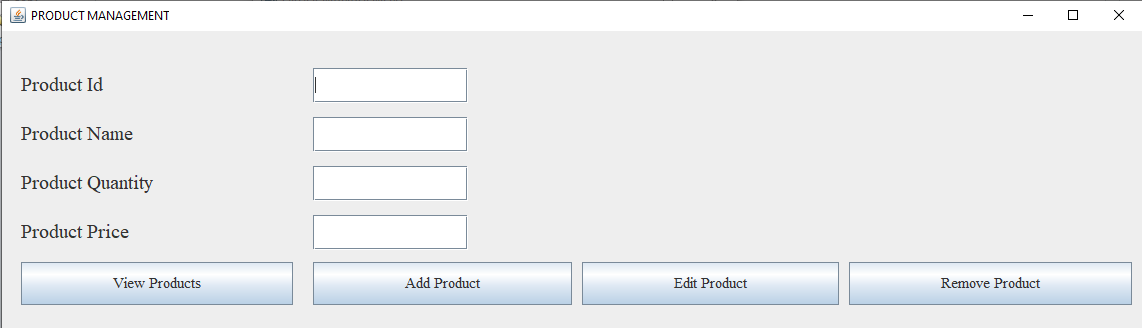
* Fereastra Client :



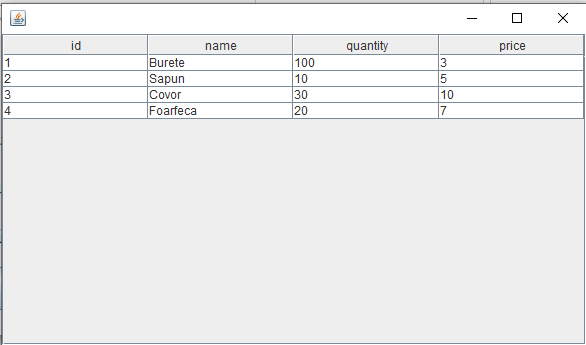
* Tabela Client :



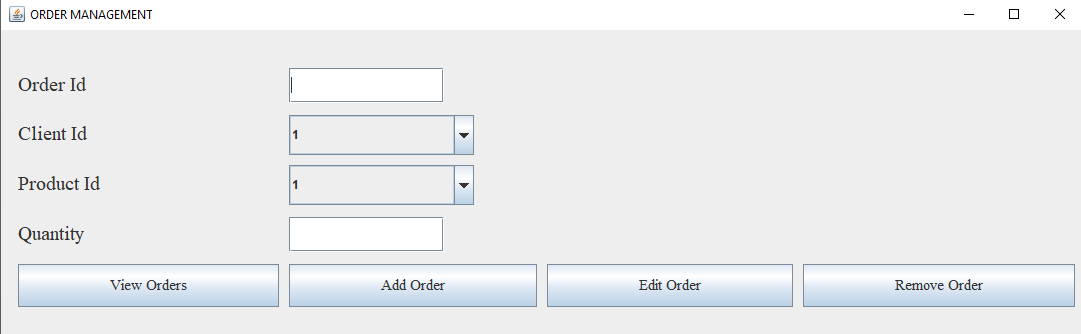
* Fereastra Product :



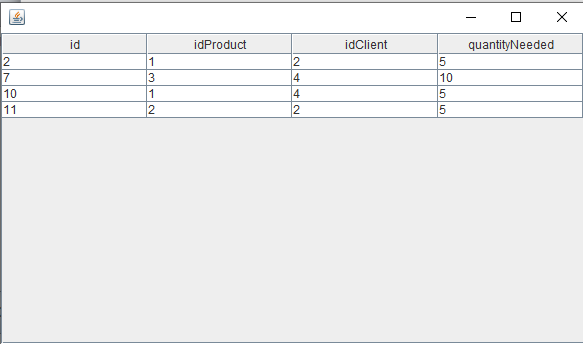
* Tabela Product:



* Fereastra Order :



* Tablea Comanda:



# 6. CONCLUZII

Pentru mine aceasta tema a ridicat anumite dificultati in ceea ce priveste conecatrea la baza de date si aducerea aplicatiei la o forma adecvata. Totodata am intampinat unele dificultati la implementarea functiei de createTable folosind tehnica de Reflection.

# 7. BIBLIOGRAFIE

* <https://www.geeksforgeeks.org/>
* <https://stackoverflow.com/>