

**FACULTATEA: Automatică și Calculatoare**

**SPECIALIZAREA: Calculatoare și Tehnologia Informației**

**DISCIPLINA: Tehnici de Programare**

**Tema 3**

**Îndrumător de laborator: Realizator:**

Antal Marcel Sălăjanu Anamaria

Contents

[Obiectivul temei 2](#_Toc103082917)

[Analiza problemei 3](#_Toc103082918)

[Proiectare 4](#_Toc103082919)

[Decizii de proiectare 4](#_Toc103082920)

[Diagrama UML 4](#_Toc103082921)

[Proiectarea pe clase și pachete 4](#_Toc103082922)

[Implementare 6](#_Toc103082923)

[Concluzii 8](#_Toc103082924)

# Obiectivul temei

Obiectivul celei de-a treia teme de laborator este acela de a crea și implementa o aplicație de tip OrderManagement cu scopul procesării unor comenzi ale mai multor clienți. Pentru a putea stoca intemii introduși este necesara o baza de date care va fi implementata cu ajutorul MySQL.

Aplicația va conține minim următoarele clase:

* Clasele Model : Comanda, Client, Produs, Cumpărături – acestea sunt produselele din baza de date care conțin câmpurile 1 la 1
* Logica business: clase care conțin logica aplicației
* Clasele de prezentare: aici se pune partea de interfața
* Clasele de acces la date: clase care conțin query-urile și leagă baza de date de restul aplicației

In plus avem mai multe clase adăugate care ajuta la implementare si fluidizare codului asigurând corectitudine aplicației.

Primul obiectiv al temei este de a avea o implementare a unei aplicații care are ca si scot ordonarea unui magazin cu posibilitatea de a putea adaugă într-o baza de date niște produse care ajuta la sistemizarea unui magazin care este in viata reala.

Pentru a putea adaugă acele obiecte si clienții care vor sa ne intre in magazin vom folosi o structura de tip SQL si acestea trebuie sa fie aceleași cu structurile din clasa de Java pentru a putea fi in concordanța cu baza de date creata pentru acest scop.

# Analiza problemei

O baza de date are scopul si sunt niște fișiere care se conectează cu aplicația noastră si asigura o buna conexiune cu aplicația noastră pe care am avut-o de implementat.

Sistemul de gestiune a bazei de date (SGBD) este ansamblul software interpus între utilizatori și baza de date și este interpretor de cereri de acces sau regăsire de date în baza de date, execută cererea și returnează rezultatul.

Funcțiile principale ale SGBD: SELECT, INSERT, UPDATE, DELETE. Tehnica de reflexie aste necesara pentru a putea realiza operațiunile generice a unei baze de date funcționala si necesara pentru orice fel de baza de date.

Modelul ne reprezintă baza noastră de date care este in corespondenta cu aplicația Java, astfel sunt hard-codate datele, având aceeași denumire si tip ca si cele din baza de date care a fost creată pentru acest proiect. Aici este si efectul operațiilor de select , deleter , update, insert și totodată conexiunea la baza de date. Astfel acest pachet este o poarta intre baza de date si acțiunile si comenzile executate din aplicația Java care sunt introduse de către un utilizator. Aceste funcții definesc structura necesara aplicației.

Implementarea cu clasa generica a fost aleasa deoarece simplifica mult complexitatea codului si posibilitatea de adaptare este mult mai ușoara, scutindu-ne de metodele de select , deleter , update, insert pentru fiecare tabela din baza noastră de date. Acestea pot sa primească tipul dorit (Produs, Client, Comandă sau Producător) printr-un parametru dat din clasa de validare a datelor care va fi explicata mai jos.

# Proiectare

## Decizii de proiectare

Pentru realizarea proiectului am folosit mediul de dezvoltare intelliJ IDEA și limbajul de programare Java. De asemenea am mai folosit mediul de dezvoltare MySQL Workbench pentru a realiza baza de date și realizarea unei conexiuni stabile cu aplicația aleasa spre dezvoltare.

## Diagrama UML

Unified Modeling Language (UML) este un limbaj standard pentru descrierea de modele și specificații pentru software. Acesta a fost dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase și instanțe ale acestora (obiecte).

Diagrama pe clase este o structură statică care descrie de fapt structura unui sistem prezentând clasele sale, atributele acestora, metodele conținute în fiecare clasă și relațiile dintre obiecte.

## Proiectarea pe clase și pachete

Pentru a putea implementa avem nevoie de câteva funcționalități de baza care se numără printre:

• produsele care se afla in magazin la momentul actual

• comenzile cerute de către client

• modul in care aceste produse si comenzi sunt stocare, respectiv afișate

• datele despre produse (preț, cantitate), clienți (vârsta, email)

Adăugarea unui produs se refera la adăugarea unui ID unic pentru acesta si adăugarea unui nume specific cu o anumita cantitate si un preț specific, prețul putând sa fie si cu virgula sau fără.

Adăugarea unei comenzi este o legătura intre aceasta si un client, deoarece clientul x comanda n produse cu o valoare totala y. In funcție de disponibilitatea produselor trebuie evaluata posibilitatea efectuării comenzii la momentul in care clientul dorește sa o efectueze sau va trebui sa aștepte pana la adăugarea a mai multor produse in stoc.

Modalitatea in care se stochează datele pentru acest tip de aplicație este in deosebi de special pentru ca nu putem admite duplicate (aceste trebuie sa aibă un di unic) si sa fie salvate pentru a putea accesa valorile acestora pentru acțiuni ulterioare asupra lor chiar si in momentul închiderii aplicației.

Aplicația este structurată în 7 pachete, fiecare conținând clase cu funcționalitate bine definită și esențială pentru a avea o corectitudine cat de bine reprezenta tata.

**Pachetul Model:**

Acest pachet conține clasele care sunt același cu cele din baza mea de date: Produs, Client, Comandă. Aceste patru clase conțin o referință si sunt de același tip ca si cele din baza mea de date care a fost creată si legata la proiect anterior.

**Pachetul DAO:**

Cuprinde clasele care conțin accesul la baza de date. Am ales să creez în acest pachet clasa abstractă AbstractDAO care are rolul de a interpreta query-urile trimise in mod generic de către utilizator si a afișa rezultatele corespunzătoare prin intermediul variabilelor generice care sunt înlocuite la fiecare apelare cu tipul necesar. Acest pachet mai conține încă 4 clase: ClientDAO, ProdusDAO si ComandaDAO, care sunt goale dar extind clasa DAO generica pentru a putea apela metodele din ea pentru tipul dorit.

**Pachetul Validators:**

Conține clase care validează datele ce vor să fie introduse în tabel:

* EmailValidator – se asigură că adresa de email introdusă de client este una validă, conform unui anumit model.
* StocValidator – se asigură că la introducerea unui produs nou în baza de date, stocul acestuia va fi un număr mai mare ca 0, astfel asigurând tot timpul existenta produsului in magazin.

Aceste clase implementează interfața Validator care are declarată metoda generică validate unde este definita implementarea pentru acești validatori cereați de noi.

**Pachetul BLL:**

Are în componența sa clasele care conțin logica propriu-zisă a aplicației. Aceste clase conțin atât validatorii, cât și apelarea metodelor generice de creare a query-urilor, folosind tipul potrivit.

**Pachetul Connection:**

Acest pachet conține o singură clasă numită ConnectionFactory. Clasa respectivă face legătura dintre aplicație si baza de date care este folosita pentru a putea lua datele pentru acest magazin.

**Pachetul Presentation:**

Are în componența sa clasele care conțin interfața grafică, astfel cuprinde toate elementele de grafica care au in componenta lor un GUI.

# Implementare

Am ales implementarea claselor din pachetul model cu câmpuri conform tabelelor din baza de date.

Clasa Client conține câmpurile: ID de tip int, nume de tip String, prenume de tip String, adresa de tip String, telefon de tip int, email de tip String.

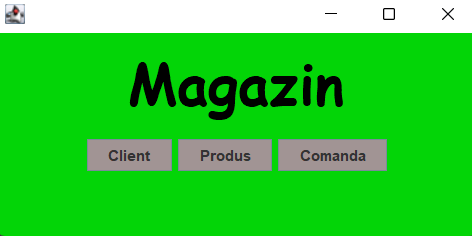
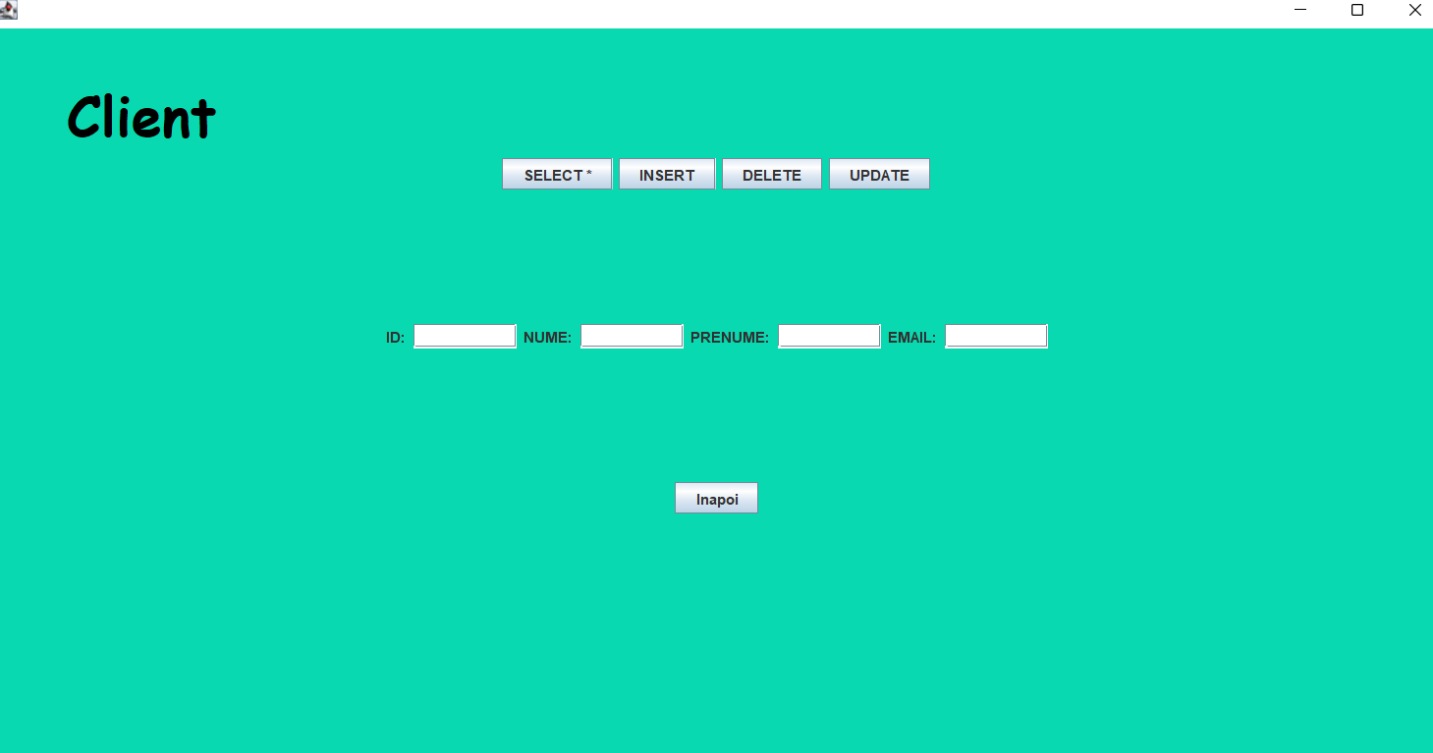
Clasa Comanda conține câmpurile: ID (int), IDProdus (int), IDClient (int), cantitate (int), pret (float).

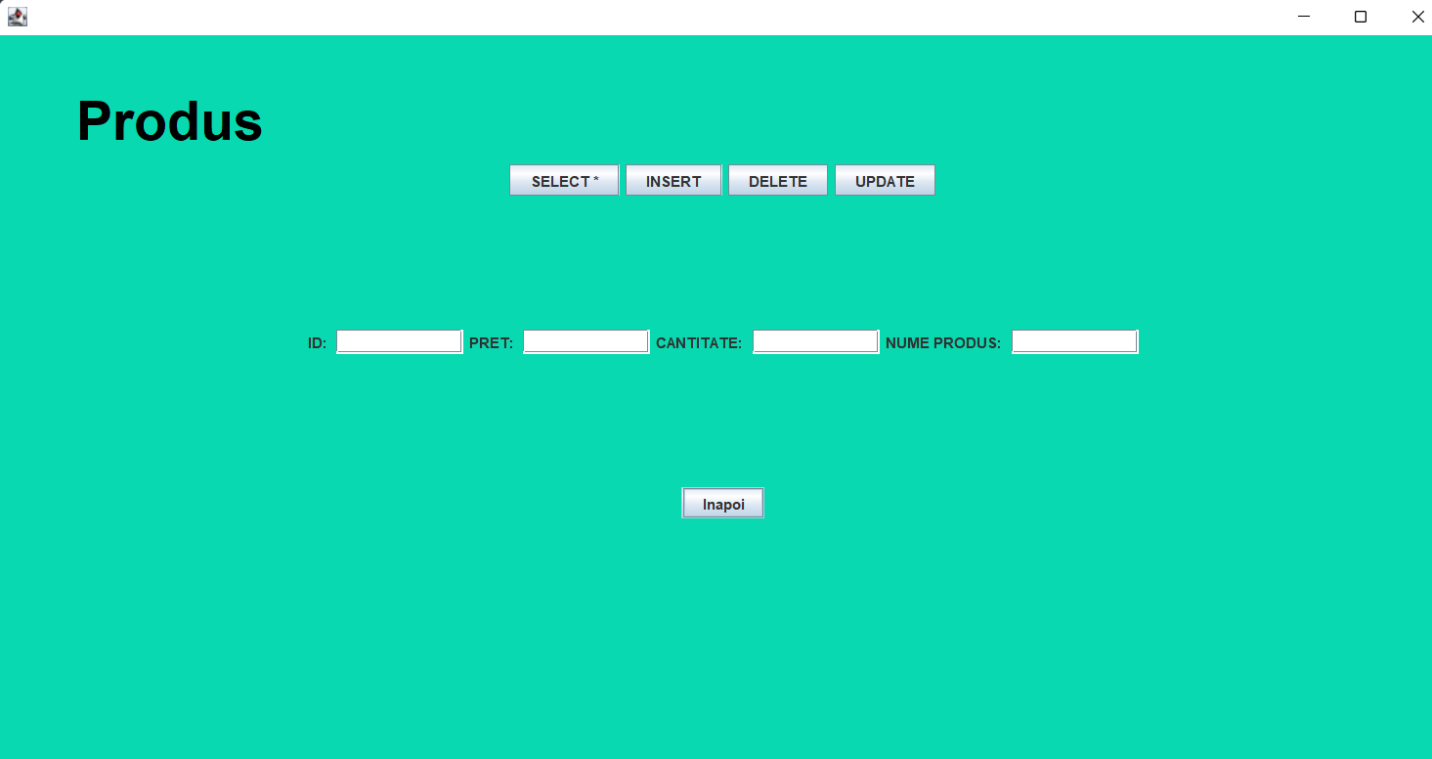
Clasa Produs conține câmpurile: ID (int), nume (String), pret (float), gramaj (int), stoc (int).

Implementarea interfeței a fost făcută cu ajutorul a cinci clase.

Clasa Magazin conține fereastra principală în care selectăm pe care tabelă dorim să facem operații. În funcție de asta, unul dintre cele 4 butoane va deschide o nouă fereastră.

Clasele ClientGUI, ProdusGUI, ComandaGUI, CumparaturiGUI sunt implementate pe același modalitate. Avem câte un buton pentru fiecare dintre cele 5 operații și textfield-urile din care vom citi datele corespunzătoare fiecărei operații, apoi se va afișa tabelul creat cu Jtable. Tabelul este adăugat într-un JscrollPane. Fiecare buton are implementat câte un ActionListener în care se execută operația corespunzătoare butonului si posiilitatea de a face legatura dintre mai multe interfața grafice (in cazul butonului de BACK).



Ferestrele de prezentare sunt similare pentru fiecare opțiune din meniul principal.

# Concluzii

Ca si o concluzie, acest proiect a fost un bun ajutor pentru a putea dezvolta o noua cunoștința, mai exact cea de a leagă baza de data cu un proiect de tip Java si a putea modifica datele din aceasta. Abstractizarea datelor si refolosirea codului a fost o provocare deosebita, din cauza modalității de abordare a problemei, astfel fiind nevoie doar de un set de quarc-uri pentru toate tabelele folosite.

În cadrul dezvoltărilor ulterioare se pot aminti următoarele: îmbunătățirea interfeței, modificarea comenzii după ce aceasta a fost introdusă si o modalitate de a adaugă Anuțe reduceri pentru a facilita interesul clienților si o posibila întoarcere la acest magazin.