# TEHNICI DE PROGRAMARE

# TEMA 3

# *Realizat de : Ghiurca Dorin*

# *An: 2*

# *Grupa: 30223*

***CUPRINS***

1. **Obiectivul temei ...............................................................................4**
2. **Analiza problemei, modelare, scenarii, cazuri de utilizare.................4**
   1. Analiza problemei.......................................................................5
   2. Modelare....................................................................................5
   3. Scenarii, Cazuri de utilizare........................................................6
3. **Proiectare.........................................................................................7**
4. Diagrama UML............................................................................6
5. Designul Claselor............................................................................7
6. Algoritmi....................................................................................11
7. Interfata Utilizator.....................................................................12
8. **Rezultate, concluzii, dezvoltari ulterioare.........................................16**

4.1 Lucruri noi invatate.....................................................................17

4.2 Dezvoltari ulterioare....................................................................17

**6. Bibliografie.............................................................................................18**

**1. Obiectivul temei**

Luati in considerare o cerere de procesare a comenzilor clientilor unui depozit/magazin.

Bazele de date relationale sunt utilizate pentru a stoca produse, clienti si comenzi. Mai mult decat atat, aplicatia utilizeaza (minim) urmatoarele clase:

1. Clase specifice domeniului: Comanda, Client si Produs;

2. Business Logic Classes (procesare specifica depozitului): OrderProcessing, WarehouseAdministration, ClientAdministration;

3. Clase de prezentare: clase legate de GUI;

4. Clase de acces la date: Clase legate de accesul la baza de date;

Alte clase si pachete pot fi adaugate pentru a implementa functionalitatea completa a aplicatiei.

**Cerinte**

**A.** Analizati aplicatia propusa, determinati structura si comportamentul clasei si desenati o diagrama UML extinsa a claselor

**B.** Proiectati, implementati si testati clasele de aplicatii. Utilizati javadoc pentru a documenta clasele.

**C.** Definiti, proiectati si implementati un sistem de programe de utilitate (exemple: rapoarte pentru stocuri, totalui, filtre etc).

**D.** Proiectati si implementati un driver demo corespunzator pentru aplicatia de gestionare a comenzilor.

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

**2.1 Analiza problemei**

Companiile care se ocupa cu producerea si distribuirea diferitelor produse, pe masura ce se dezvolta au din ce in ce mai mare nevoie de un sistem de mangement al marfurilor/produselor.

Este foarte dificil ca gestiunea acestora sa se realizeze fara o aplicatie de gestionare deoarece se poate ajunge foarte usor la erori, inconsistenta datelor si o manipulare si distribuire foarte dificila, lucru deloc placut consumatorului/clientului.

Pe masura ce activitatea companiei creste, controlul costurilor de operare a afacerii devine critic. Din totalul acestor costuri, cele logistice au o pondere semnificativa, mai ales in cadrul firmelor de distributie. Pe langa aspectul financiar, organizarea eficienta a depozitelor conduce la un timp mediu de livrare scurt, care se poate consitui in avantaj competitiv important.

In functie de arhitectura depozitului, in urma unui proces de analiza, se selecteaza structura care poate conduce la fluxuri eficiente de operare a activitatilor specifice. Odata activat acest modul, orice tranzactie care afecteaza depozitul respectiv, se va efectua conform parametrilor specificati.

In urma implementarii unui asemnea sistem de gestionare se obtin urmatoarele avantaje:

* Optimizarea stocurilor;
* Faciliteaza procesele de inventariere;
* Reduce semnificativ timpul dedicat manipularii marfurilor;
* Ofera informatii in timp real cu privire la stocul marfurilor;
* Contribuie la cresterea productivitatii angajatilor din depozit;
* Ofera vizibilitate crescuta asupra tuturor activitatilor desfasurate in depozit.

**2.2 Modelare**

Programul are in centru trei clase principale: Client, Product si Order. Acestea sunt clasele care stau la baza aplicatiei. In clasa Client sunt pastrati clientii care realizeaza comenzi. Clasa Product constituie clasa in care sunt implementate produsele ce se vor gasi in depozit. Clasa Order este clasa in care sunt pastrate informatiile despre fiecare comanda.

Pe fiecare dintre aceste clase mentionate anterior exista si cate o clasa de business logic si anume: ClientBLL, ProductBLL, OrderBLL. Acestea sunt clasele care codifica regulile de business ce determina datele care pot fi create, stocate si modificate.

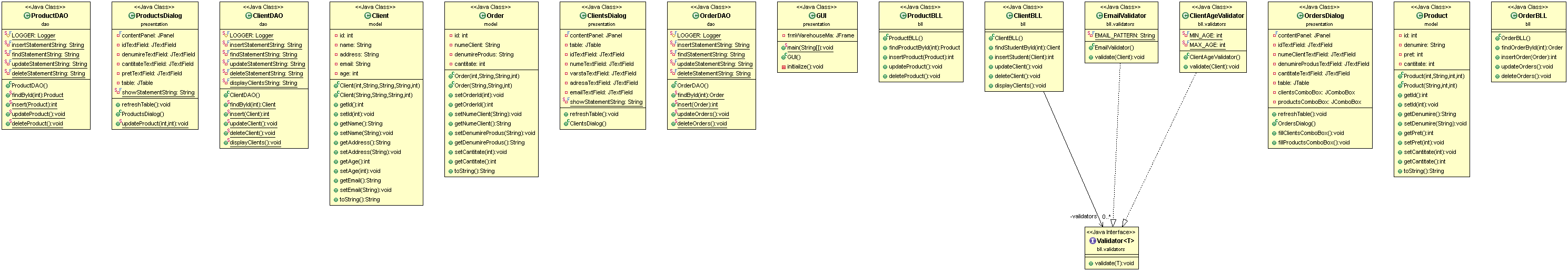
In continuarea acestor clase urmeaza clasele DAO(Data Access Object): ClientDAO, ProductDAO si OrderDAO. Acestea sunt clasele care ofera o interfata abstracta pentru accesul la baza de date. Aici se realizeaza operatiile principale de interogare a bazei de date cu scopul obtinerii datelor dorite de utilizator.

**2.3 Scenarii, cazuri de utilizare**

In momentul lansarii aplicatiei, interfata grafica permite utilizatorului sa aleaga in prima faza de ce date are nevoie: legate de clienti, produse sau comenzi. Dupa selectarea optiunii dorite, se va directiona utlizatorul la urmatoarea faza si anume la permiterea accesului la informatiile dorite. Acesta poate sa adauge, sa modificice, sa stearga sau sa afieze clienti, produse sau comenzi, in functie de optiunea aleasa la pasul anterior.

**3.Proiectare**

**3.1 Diagrama UML**

****

**3.2 Designul claselor**

Programul are in prim plan trei clase: clasa Client, clasa Product si clasa Order. Acestea sunt clasele specifice oricarei aplicatii de gestiune a unui depozit. Urmeaza pe urma clasele de bussiness logic: ProductDAO, ClientDAO, OrderDAO, clase ce codifica regulile de business si care acceseaza mai departe clasele de tip DAO(Data Access Object).

**1) Clasa Client**

Atribute:

Exista mai multe atribute cu scopul de a oferi informatii despre fiecare client: id, nume, adresa, email, varsta. Acestea sunt utilizate in momentul adaugarii unui nou client in baza de date.

Constructori:

Exista doi constructori, fiecare cu mai multi parametrii. Primul constructor primeste ca parametrii fiecare din atributele mentioanate anterior si pe urma creaza pe baza acestor parametrii un nou client. Cel de-al doilea constructor primeste aceeasi parametrii ca si constructorul mentionat anterior cu exceptia id-ului. Acest constructor are rolul de a usura munca utilizatorul de a adauga clienti in baza de date prin faptul ca utilizatorul nu trebuie sa retina id-ul ultimului client, acesta incrementandu-se automat la fiecare client adaugat in baza de date.

Metode:

Clasa are mai multe metode de tip set si get, metode ce au ca scop preluarea informatiilor despre client sau modificarea acestora in functie de dorintele utilizatorului.

**2) Clasa Product**

Atribute:

Exista mai multe atribute cu scopul de a oferii informatii despre fiecare produs: id, denumire, pret, cantitate. Acestea sunt utilizate in momentul adaugarii unui nou produs in baza de date.

Constructori:

Exista doi constructori, fiecare cu mai multi parametrii. Primul constructor primeste ca parametrii fiecare din atributele mentioanate anterior si pe urma creaza pe baza acestor parametrii un nou produs. Cel de-al doilea constructor primeste aceeasi parametrii ca si constructorul mentionat anterior cu exceptia id-ului. Acest constructor are rolul de a usura munca utilizatorul de a adauga produse in baza de date prin faptul ca utilizatorul nu trebuie sa retina id-ul ultimului produs, acesta incrementandu-se automat cu fiecare produs adaugat in baza de date.

Metode:

Clasa are mai multe metode de tip set si get, metode ce au ca scop preluarea informatiilor despre produse sau modificarea acestora in functie de dorintele utilizatorului.

**3) Clasa Order**

Atribute:

Exista mai multe atribute cu scopul de a oferii informatii despre fiecare comanda: id, nume client, denumire produs si cantitate. Acestea sunt utilizate in momentul adaugarii unei comenzi in baza de date.

Constructori:

Exista doi constructori, fiecare cu mai multi parametrii. Primul constructor primeste ca parametrii fiecare din atributele mentioanate anterior si pe urma creaza pe baza acestor parametrii o noua comanda. Cel de-al doilea constructor primeste aceeasi parametrii ca si constructorul mentionat anterior cu exceptia id-ului. Acest constructor are rolul de a usura munca utilizatorul de a adauga comenzi in baza de date prin faptul ca utilizatorul nu trebuie sa retina id-ul ultimei comenzi, acesta incrementandu-se automat la fiecare comanda adaugata in baza de date.

Metode:

Clasa are mai multe metode de tip set si get, metode ce au ca scop preluarea informatiilor despre comenzi sau modificarea acestora in functie de dorintele utilizatorului.

**4) Clasa ClientBLL**

Atribute:

Clasa detine doar un singur atribut – o lista de validatori cu scopul de valida informatiile introduse de catre utilizator in baza de date.

Constructori:

Clasa detine un constructor fara parametrii care are ca scop adaugarea informatiilor introduse de catre utilizator spre a fi validate.

Metode:

Clasa detine mai multe metode cu scopul de a realiza toate operatiile specifice claselor de baza: adaugare, modificare, stergere si afisare.

**5) Clasa ProductBLL**

Atribute:

Clasa detine doar un singur atribut – o lista de validatori cu scopul de valida informatiile introduse de catre utilizator in baza de date.

Constructori:

Clasa detine un constructor fara parametrii care are ca scop adaugarea informatiilor introduse de catre utilizator spre a fi validate.

Metode:

Clasa detine mai multe metode cu scopul de a realiza toate operatiile specifice claselor de baza: adaugare, modificare, stergere si afisare.

**6) Clasa OrderBLL**

Atribute:

Clasa detine doar un singur atribut – o lista de validatori cu scopul de valida informatiile introduse de catre utilizator in baza de date.

Constructori:

Clasa detine un constructor fara parametrii care are ca scop adaugarea informatiilor introduse de catre utilizator spre a fi validate.

Metode:

Clasa detine mai multe metode cu scopul de a realiza toate operatiile specifice claselor de baza: adaugare, modificare, stergere si afisare.

**7) Clasa ClientDAO**

Atribute:

Clasa contine mai multe atribute de tip string, atribute in care se pastreaza cerintele pentru interogare. Aceste atribute vor fi folosite in momentul conectarii la baza de date unde, pe baza acestor cerinte, se modifica sau se retrage informatia dorita in functie de fiecare atribut.

Constructori:

Clasa nu utilizeaza nici un constructor.

Metode:

Se gasesc mai multe metode, fiecare cu scopul de realiza operatiile de baza specifice clasei Client: adaugare, modificare stergere si afisare.

**8) Clasa ProductDAO**

Atribute:

Clasa contine mai multe atribute de tip string, atribute in care se pastreaza cerintele pentru interogare. Aceste atribute vor fi folosite in momentul conectarii la baza de date unde pe baza acestor cerinte, se modifica sau se retrage informatia dorita in functie de fiecare atribut.

Constructori:

Clasa nu utilizeaza nici un constructor.

Metode:

Se gasesc mai multe metode, fiecare cu scopul de realiza operatiile de baza specifice clasei Product: adaugare, modificare stergere si afisare.

**9) Clasa OrderDAO**

Atribute:

Clasa contine mai multe atribute de tip string, atribute in care se pastreaza cerintele pentru interogare. Aceste atribute vor fi folosite in momentul conectarii la baza de date unde pe baza acestor cerinte, se modifica sau se retrage informatia dorita in functie de fiecare atribut.

Constructori:

Clasa nu utilizeaza nici un constructor.

Metode:

Se gasesc mai multe metode, fiecare cu scopul de realiza operatiile de baza specifice clasei Order: adaugare, modificare stergere si afisare.

**3.2 Algoritmi**

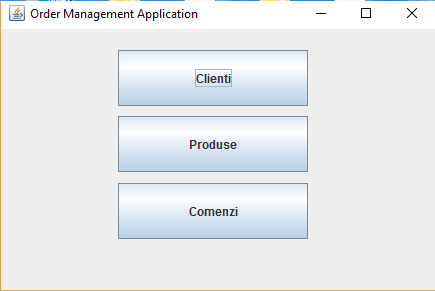
Pentru realizarea operatiilor asupra bazei de date, algoritmii folositi sunt foarte asemanatori intre ei.

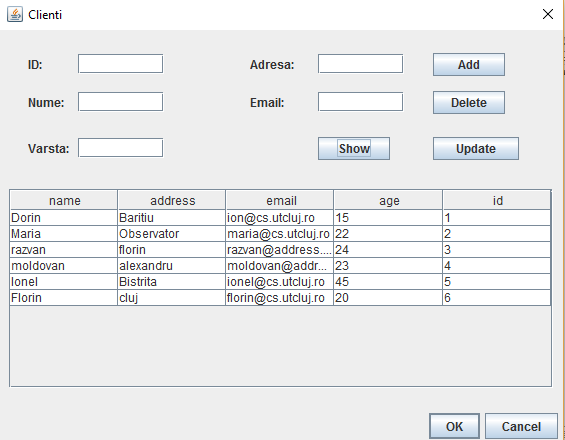
In prima faza trebuie realizata conexiunea la baza de date. Acest lucru se realizeaza prin intermediul unui obiect de tip Connection. Trebuie specificat driver-ul, URL-ul , precum si utilizatorul si parola pentru conexiuna la baza de date. Un lucru care este bine a fi mentionat este ca dupa fiecare conexiune la baza de date si dupa terminarea operatiilor, aceasta conexiune trebuie inchisa pentru a nu compromite siguranta si mentenanta datelor.

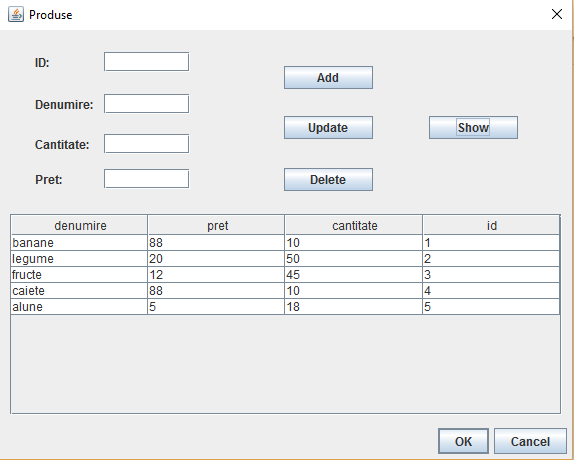
Dupa cum s-a mai mentionat, operatiile asupra bazei de date sunt asemanatoare. Acest lucru se datoreaza faptului ca se realizeaza interogari asupra datelor disponibile. Pentru a avea acces si pentru a folosi si modifica datele, mai intai se declara variabila de tip string care constituie interogarea propriu zisa care se va realiza asupra bazei de date. In continuare se utilizeaza un obiect de tip PreparedStatement pentru a trimite declaratiile catre baza de date, lucru ce faciliteaza prelucrarea datelor care vor fi interogate. Dupa realizarea acestui lucru se executa interogarea cu ajutorul metodei executeQuery. Rezultatul interogarii se pastreaza intr-o variabila de tip ResultSet. In cele din urma se preiau rezultatele din aceasta variabila si se actualizeaza baza de date in functie de operatia pe care am realizat-o asupra bazei de date. In urma realizarii operatiilor mentionate anterior, se inchide atat conexiunea la baza de date cat si statement-ul in care s-au gasit operatiile trimise catre baza de date si obiectul de tip ResultSet in care s-au preluat datele.

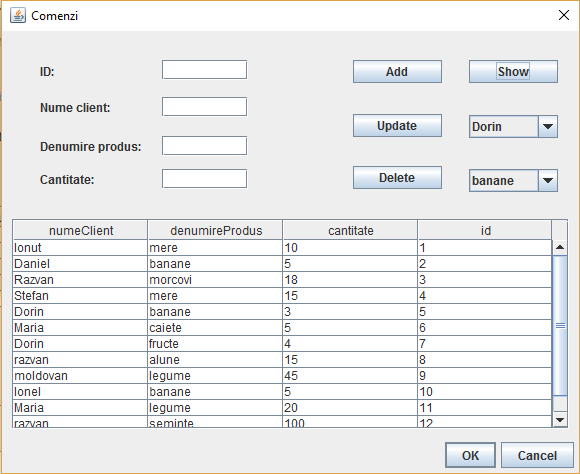
**3.4 Interfata utilizator**

Interfata este compusa din mai multe componente. In prima faza, se deschide o fereastra care ne cere sa alegem la care dintre cele 3 clase de baza dorim sa avem acces: Client, Product si Order. Dupa aceasta alegere ni se va deschide o fereastra in care putem accesa functiile clasei respective. Avem mai multe text field-uri in care putem introduce informatiile dorite referitoare la clasa Client, Product sau Order. Exista mai multe butoane care in urma actionarii se realizeaza operatia specificata in denumirea acestor butoane.







****

**4.Rezultate, concluzii, dezvoltari ulterioare**

Dupa finalizarea temei am constatat ca o problema relativ simpla la prima vedere, necesita o mare atentie si multa munca. Lucrul cu baza de date presupune o mare atentie deoarce pot aparea pe parcurs o multitudine de probleme.

**4.1 Lucruri noi invatate**

Realizarea acestei teme m-a determinat sa invat gestionarea unei baze de date, precum si conectarea la acesta prin intermediul limbajului de programare Java.

Gestionarea unei baze de date – lucru simplu la prima vedere – poate pune o multitudine de probleme. Initial m-am confruntat cu o multitudine de exceptii care m-au determinat sa amplific cunostintele in ceea ce priveste folosirea unei baze de date pentru o mai buna intelegere a cauzei aparitiei acestor erori sau exceptii.

Un alt lucru important pe care pot spune ca l-am invatat este realizarea documentatiei. Am observat ca realizarea documentatiei ficarei teme, ma poate ajuta sa inteleg mai bine rezolvarea problemei, precum si oferirea unei explicatii mult mai usor in legatura cu ceea ce se intampla in proiect.

**4.2 Dezvoltari ulterioare**

Sunt o multime de lucruri care se pot schimba sau adauga in cadrul aplicatiei in functie de contextul de folosire, pentru a imbunatati functionalitatea sau partea de interfata cu utilizatorul.

**Posibile imbunatatiri:**

* Posibilitatea de a stoca mai multe informatii despre utilizator;
* Posibilitatea ca dupa realizarea unei comenzi, clientul sa poata observa stagiul in care se afla comanda;
* Posibilitatea adaugarii mai multe functii in proiect pentru gestiunea bazei de date;

**BILBIOGRAFIE**

* http://www.coned.utcluj.ro/~salomie/PT\_Lic/3\_Lab/HW3\_Tema3/HW3\_Indications.pdf
* https://www.javatpoint.com/steps-to-connect-to-the-database-in-java
* https://docs.oracle.com/javase/tutorial/reflect/
* http://stackoverflow.com/questions/21361781/how-to-connect-to-database-connection-in-java
* https://www.youtube.com/watch?v=qGelg87DpQA