

DOCUMENTATIE

TEMA 3

Luca Andrada Aurelia

Grupa 30228

CUPRINS

**1.** Obiectivul temei

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

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

**4.** Implementare si testare

**5.** Rezultate

**6.** Concluzii, ce s-a invatat din tema, dezvoltari ulterioare

**7.** Bibliografie

**OBIECTIVUL TEMEI**

**Cerinta:**

**Specification**: Consider an order management application for processing customer orders for a warehouse . Relational databases are used to store the products, the clients and the orders . Furthermore, the application uses (minimally) the following classes: 1. Domain specific classes: Order, Customer and Product 2 . Business Logic (warehouse-specific processing) classes: OrderProcessing, WarehouseAdministration, ClientAdministration.

3. Presentation classes: GUI related classes . 4. Data access classes: Database access related classes Other classes and packages can be added to implement the full functionality of the application .

**Requirements** : a. Analyze the proposed application, determine the structure and behavior of its classes and draw an extended UML class diagram .

b. Design, implement and test the application classes . Use javadoc for documenting classes.

c. Define, design and implement a system of utility programs (examples : reports for under-stock, totals, filters, etc .) .

d. Design and implement a comprehensive demo driver for the order management application .

**Obiective:**

Aceasta aplicatie are ca si obiectiv interactionarea in timp real cu baza de date a depozitului si cu utilizatorul, proiectarea si implementarea unui program ce va fi folosit in procesarea unor comenzi. Aceste comenzi vor veni de la anumiti clienti, care vor dori sa cumpere anumite produse in anumite cantitati de la un magazin, practic sa plaseze o comanda. Produsele se pot gasi in stocul magazinului in cantitati suficiente sau insuficiente. Comenzile se definesc printr-un anumit client care le plaseaza, un id de identificare,id-ul produsului comandat, cantitatea dorita si starea acesteia. Clientii se definesc printr-un ID unic, ce ne va ajuta sa-i diferentiem. Dupa efectuarea unei comenzi si realizarea acesteiea, produsele cumparate vor fi scazute din stocul magazinului. De asemenea, trebuie sa avem si posibilitatea de a improspata stocul sau de a actualiza datele unui client sau a unei comenzi in cazul in care se doreste aceasta operatie. Utilizatorul poate sa vizualizeze, sa adauge, sa caute, sau sa actualizeze produsele. La finalizarea comenzi se va afisa si pretul total.

**Analiza problemei si modelarea acesteia**

Primul pas, pentru a putea rezolva cerintele temei , este dat de nevoia de intelegere buna a bazelor de date MySQL. Acest lucru inseamna ca toate comenzile pe care dorim sa le executam (vizualizare, stergere, actualizare, comanda) se vor face si pe baza de date.

Problema consta in a implementa o aplicatie pentru managementul unor comenzi efectuate de diferiti clienti , se poate utiliza de oricine si este foarte utila.Aplicatia foloseste o baza de date externa formata in MySql WorkBench unde sunt retinute toate datele despre clientii,produse ,comenzi si furnizori. In cadrul aplicatiei se mai pot efectua si alte operatii, operatii de a adauga un client in baza de date , de a sterge un client,de a face update la informatiile despre un client si aceeasi operatii pentru produse , furnizori si comenzi . In caz de introducere a unei cantitati intregi la o valoare string sau neintroducere a campurilor, aplicatia va afisa o atentionare , ea asteptand introducerea altor date care trebuie sa fie corecte. Modificarea in baza de date se face in timp real , dupa fiecare operatie efectuata in interfata aplicatiei baza de date este modificata corespunzator. Aplicatia ofera si posibilitatea de a vedea ceea ce se afla in baza de date ,cum ar fi clientii care se afla in baza de date sau produsele care se afla in baza de date si am facut asta adagand cate un tabel in aplicatie care sa fie in corespondenta cu cel din baza de date .

**Cazuri de utilizare**

Aplicatia este foarte usoara deoarece am folosit un meniu care ajuta utilizatorul sa gaseseasca usor ceea ce vrea sa faca.

Pentru a intelege functionalitatea acestui program, utilizatorul va parcurge urmatorii pasi:

1. Utilizatorul ruleaza aplicatia.
2. Utilizatorul selecteaza din meniu ceea ce doreste.Acesta are 4 optiuni:

-Client

-Furnizor

-Produs

-Comenzi

Astfel daca dorim sa editam/inseram/stergem/cautam/afisam un client , mergem pe eticheta “Client”. Daca dorim sa vizualizam /stergem sau sa facem o comanda apasam pe eticheta “Comenzi”, daca dorim adaugarea/editarea/stergerea/cautarea si afisarea produselor respectiv furnizorilor mergem pe eticheta “Produs” respectiv “Furnizor”.

1. Programul executa operatia si afiseaza rezultatul, daca un produs a fost adaugat, sters, actualizat, daca o comanda a fost adaugata, stearsa, actualizata.
2. Daca se doreste iesirea din aplicatie, utilizatorul va apasa pe butonul de Exit (X).

**PROIECTARE**

Diagrama pentru clasa:

Cu ajutorul acestei diagrame UML se pot evidenția legăturile dintre clase, pachete, respectându-se altfel conceptul OOP.

UML este un limbaj vizual de modelare, dar inca nu este un limbaj de programare deoarece nu dispune de întreg sprijinul semantic și vizual pentru a înlocui limbajele de programare. Limbajul este destinat vizualizării, specificării, construirii și documentării sistemelor de aplicații, dar are limitări în ceea ce privește generarea codului.

Diagramele sunt grafuri care prezintă simboluri ale elementelor de modelare (model element) aranjate astfel încât să ilustreze o anumită parte sau un anumit aspect al sistemului.

Cu ajutorul diagramei UML, de mai jos, a avut loc proiectarea, fiind evidentiate pachetele, clasele, metodele, dar si legaturile dintre ele, respecandu-se modelul OOP.

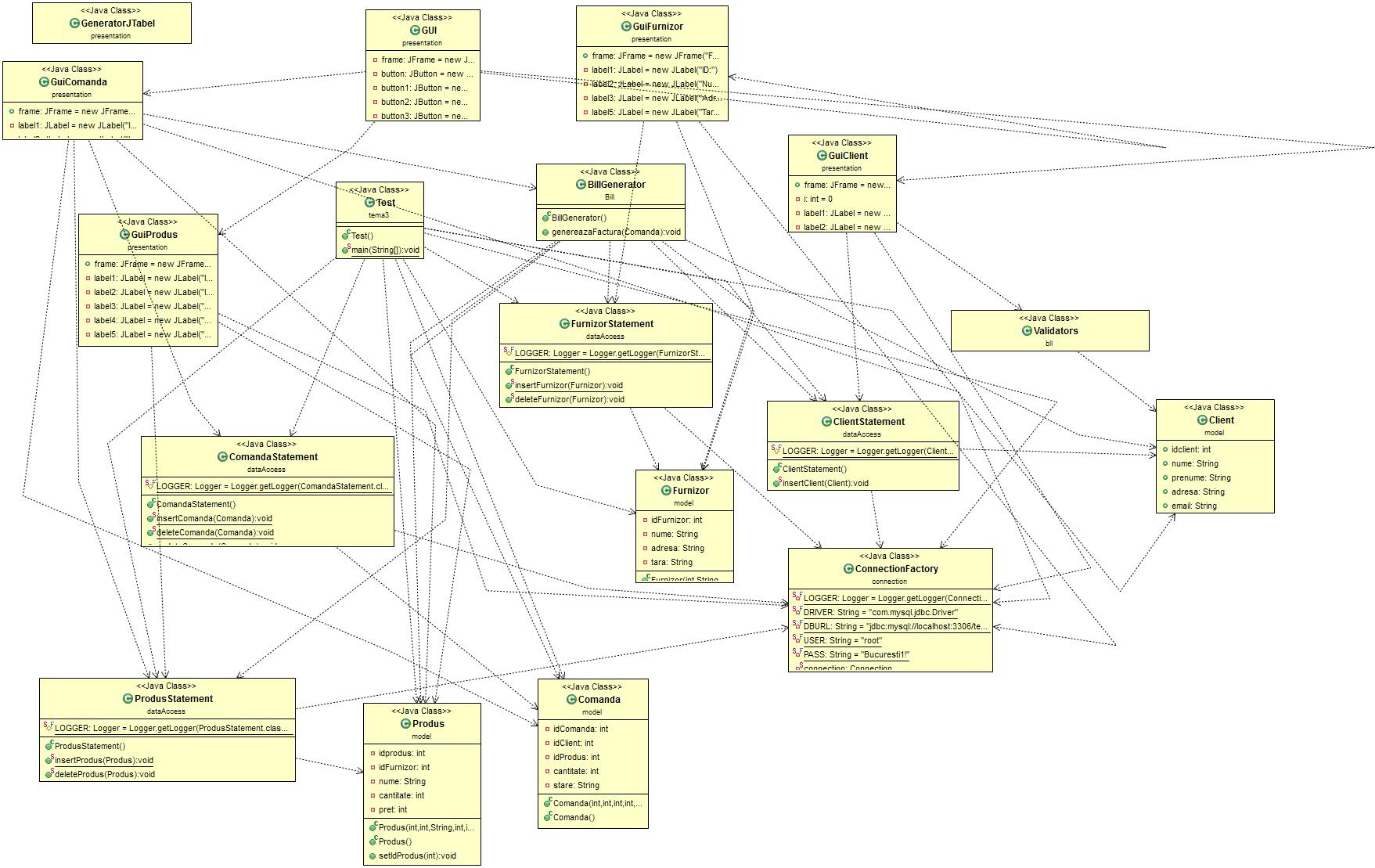
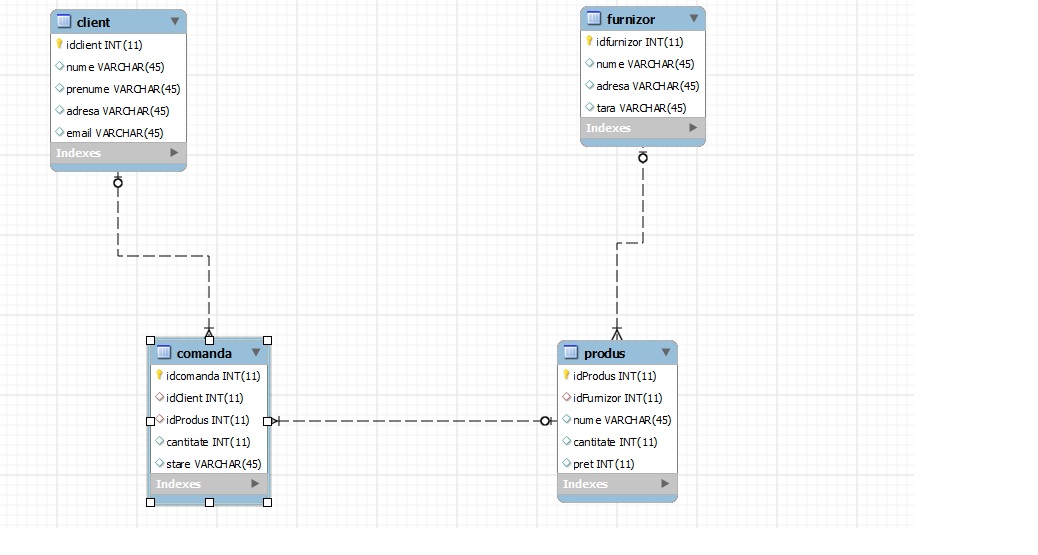


Diagrama pentru baza de date:



**Proiectare clase:**

Pentru aceasta aplicatie am folosit o arhitectura de tip Layered Architecture , unde se folosesc mai multe straturi alocand diferitele responabilitatii ale unui produs software. Astfel in proiectul nostru avem mai multe pachete care repreinta cate un strat . Pachetele folosite in proiect sunt : bll(bussines logic layer), connection , data acces ,model, presentation, tema3(test) si Bill (pentru factura, nu era necesar acest pachet deoarece factura poate fi realizata si sub forma unei metode in interiorul clasei Comanda din pachetul model ) .

Pachetul connection contine o clasa numita ConnectionFactory clasa care face legatura intre baza de date si aplicatia nostra.

Pachetul model contine cele 4 clase de baza fara de care proiectul nu ar fi functional . Contine clasa Cient care este clasa entitate pentru tabela client din baza de date , aceasta clasa contine proprietatile clinetului, id-ul care este de unic(primary key), un nume, un prenume, o adresa si un email pentru a putea avea cateva informatii despre clientul respectiv.Mai avem clasa Furnizor, clasa entitate pentru tabela furnizor, din baza de date, si contine cateva informatii despre furnizorii produselor: id-ul care este unic pentru fiecare furnizor(primary key), nume si tara de provenienta. Pachetul mai contine si clasa Produs care este clasa entitate pentru tabela produs din baza de date , aceasta clasa contine un id-ul prodului care este unic(primary key) ,id-ul furnizorului(foreign key), denumirea produsului , cantitatea si pretul produsului . Ultima clasa din acest pachet este Comanda care este clasa entitate pentru tabela comanda din baza de date folosita , clasa care are un id pentru comanda(primary key) ,un id pentru clientul care a plasat comanda respectiva, un id pentru produsul comandat (ambele foreign key), o cantitate a produsului si o stare(Preluat,in asteptare,Livrat etc).

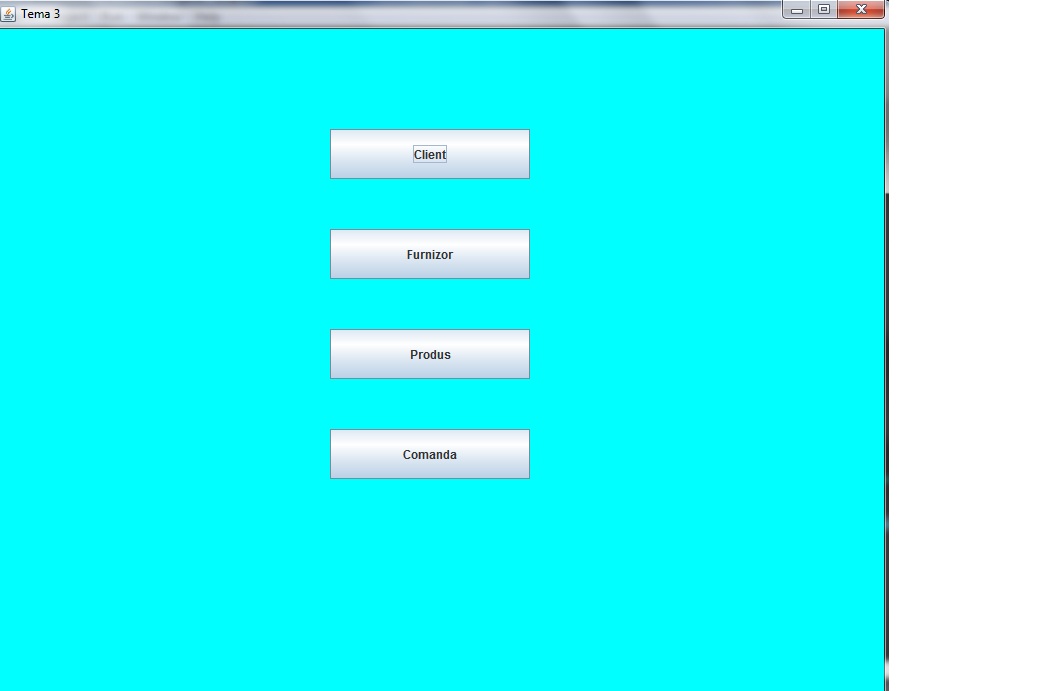
Pachetul data acces contine cele 4 tabele model care acum sunt obiecte de acces pentru tabelele din baza de date. Pachetul contine 4 clase : Client Statement, Produs Statement, Furnizor Statement, Comanda Statement . Clase in care se executa toate operatiile necesare ( insert, delete, update, select all) asupra tabelelor ( client, furnizor, produs, comanda ) din baza de date creata de noi .

Pachetul bll contine inca un pachet care se numeste validators pachet care contine toti validatori pentru fiecare clasa specifica in parte . In acest pachet avem interfata comuna tuturor validatorilor folositi si apoi avem o clasa care implementeaza interfata si care se numeste EmailValidator clasa care valideaza dadca a fost introdus correct email-ul in interfata si daca se respecta modelul dat, acest validator este folosit pentru clasa Customer

Pachetul presentation ar trebui sa fie pachetul cu cele mai multe clase deoarce am implementat 5 ferestre, iar pentru fiecare fereastra ar fi trebuit sa avem cate 2 clase,View clasa care contine fereastra principala si in care vom adauga butoane sau text flied-uri in functie de necesitate si clasa Controller in care vom adauga ascultatori la butoanele implementate in clasa anterioara, adica vom descrie comportamentul lor. Deci in consecinta ar fi trebuit sa avem 10 clase, dar din comoditate am ales sa implementez cele doua clase, View si Controler, in una singura, asa ca am ajuns sa avem doar 5 clase: 4 pentru fiecare tabel din baza de date in care avem cate 4 butoane specifice operatiilor efectuate asupra tebelei respective ( add, delete, edit, View All ) si atatea text field-uri de cate vom avea nevoie pentru introducerea unor noi date in tabel. Clasa GUI din acest pachet este clasa cu fereastra principala. La rularea aplicatiei se va deschide aceasta fereastra unde vom avea 4 butoate, cate unul pentru fiecare tabel din baza de date care la apasare ne va duce la interfata corespunzatoare acelui tabel. (Client, Produs, Furnizor , Comanda ) .

**Interfata grafica**

Interfata grafica este foarte simpla si foarte usor de utilizat. Atunci cante incepem rularea aplicatiei se deschide fereastra principala cu cele 4 butoate ca in figura urmatoare :



Prin apasarea oricarui buton utilizatorul va alege pe ce tabela vrea sa faca operatii. In functie de ce buton va alege, spre exemplu primul buton Client, se va deschide o fereastra noua in care se vor putea face toate operatiile dorite pentru tabela client. Pentru a introduce un nou client vom introduce in text field-urile respective id-ul clientului, numele, prenumele, adresa si emailul dupa care vom apasa butonul add. Pentru a sterge un client din tabela nu este necesar sa completam toate text flied-urile ci doar primul cu id clientului pe care dorim sa il sterge, iar pentru a fi siguri daca clientul a fost introdus sau sters din tabela avem butonul View All la apasarea caruia sa va deschide o noua fereastra in care vor fi trimisi toti clientii cu toate datele despre ei din baza de date. Dupa fiecare operatie este necesar sa apasam acest buton deoarece tabela nu se reimprospateaza singura. Ultimul dar nu cel din urma este butonul edit prin care vom putea schimba anumite date despre un client cu un anumit id, dar trebuie sa fim atenti deoarece daca nu completam toate text field-urile , lasam unele goale cu ideea ca acestea vor ramane identice, clientul va fi actualizat in baza noastra de date ca la anumitul camp la care nu am completat nimic fiind gol deoarece in baza noastra de date doar componentele care sunt primary key nu au voie sa fie null ( id client spre exemplu ).

Toate celelalte sunt identice cu exceptia interfetei care se deschide la apasarea butonului comanda unde avem 5 butoane in loc de 4. Butonul in plus se numeste Bill, iar la apasarea acestuia se va genera automat o factura, un fisier text cu id-ul comenzii drept nume in care vom trece toate datele cunoscute despre aceea comanda cum ar fi: data si ora la care a fost facuta, clientul, cu toate datele despre acesta, care a facut comanda, toate datele cunoscute despre produsul comandat , toate datele despre furnizorul produsului comandat si toate datele despre comanda facuta: cantitate si stare . Acest fisier .txt va fi generat in fisierul in care se afla proiectul deci nu este foarte dificil de gasit.

De fiecare data cand introducem un nou client, furnizor sau produs acesta este automat introdus in baza noastra de date fara alte modificari singura diferenta la introducerea unei comezii noi este ca de fiecare data cantitatea produsului a carui id a fost introdus , la creearea unei noi comenzi, va fi actualizata scazand cantiatea comandata din aceasta iar daca valoarea va fi mai mica de 0 va aparea un Text Message care ne va atentiona ca nu exista cantitatea ceruta si ne va sugera o cantitate maxima pe care o putem comanda .

**Implementare si testare**

La lansarea in executie a aplicatiei, se va genera in fereastra de comunicare cu utilizatorul, formata din mai multe zone de introducere a datelor si mai multe butoane.

Implementarea, codul propriu-zis a fost scris după modelul UML văzut mai sus. Se vor descrie acum în detaliu clasele, metodele utilizate pentru realizarea acestui program.

**Concluzii, ce s-a învățat din temă, dezvoltări ulterioare**

Scopul acestei teme a fost atins, astfel incat programul ruleaza. Operatiile au fost implementate cat mai simplu, dorindu-se o intelegere si liniaritate a codului. Prin urmare, scopul a fost atins cu succes , programul ruland si efectuand cu succes cerintele impuse de profesor.

Aplicatia,cu toate ca are o utilizare medie, se poate extinde intr-un program mult mai complex.

Consider ca programul se poate imbunatati atat la nivelul algoritmului si a operatiilor , cat si la nivelul design-ului prin:

* Adaugarea urmatoarelor elemente:

-imbunatatirea aspectului interfetei grafice, facand-o mai atragatoare si mai dinamica.

-se pot implementa si alti algortimi asupra bazei de date. Se poate efectua acest lucru inca din momentul in care se introduce o comanda de la tastatura . Adica in momentul in care un client nu doreste sa mai comande un anumit produs acesta sa se poata strerge din cos si chiar si din baza de date .

-sa se poata efectua o inregistrare clientiilor in aplicatie . Astfel ca utilizatorii sa aiba doar anumite drepturi asupra datelor din baza de date iar administratorul sa aiba drepturi depline . Anume , stergerea datelor , updatarea lor si/sau selectarea sa se poata face numai cu acordul administratorului .

**Ce am invatat din tema ?**

In primul rand, consider ca mi-am imbogatit cunostintele OOP-ului, reusind acum sa aplic mult mai bine teoria si conceptele invatate de-a lungul semestrului intai din acest an.

In al doilea rand, datorita faptului ca am utilizat Java Swing, consider ca am invatat multe lucruri utile pe viitor si multe noutati bune de pus in practica.

Totusi, consider ca cel mai important lucru invatat este sa ma conectez la o baza de date si sa folosesc diferite operatii (inserare, stergere, modificare sau interogari)

Nu in ultimul rand, odata cu acest proiect am inteles modul de gandire si de concepere a acestuia, reusind sa imi structurez ideile si sa le analizez corect. In urma argumentelor mentionate, am reusit sa duc la final cerinta temei proiectului.

**Bibliografie**

**-Pentru diferite probleme intampinate in crearea aplicatiei:**

1. Java de la 0 la expert ( ediția a II – a ), Autor (i) : Ștefan Tănasa, Cristian Olaru, Ștefan Andrei, Editura : Polirom, An apariție : 2007

2. [http://stackoverflow.com](http://www.serkey.com/tag/java-polynomial-division/)

3. Introducere în universul JAVA, Autor (i) : Horia Georgescu, Editura : Tehnica

4. <http://docs.oracle.com/javase/tutorial/uiswing/>

5. <http://wikipedia.com>

6. Tutoriale Youtube

7. <http://docs.oracle.com/javase/tutorial/uiswing/>

8.<http://www.tutorialspoint.com/uml/uml_basic_notations.htm>

**-Surse scolare:**

1. Cursul de Tehnici de programare al domnului profesor Ioan Salomie

2. Cursul de Programare Orientata Obiect al domnului profesor Giosan Ionel

3. Documentatia laboratorului de Programare Orientata Obiect