DOCUMENTATIE

TEMA NR. 3

Mudura Ana-Andreea

Grupa 30221

CUPRINS

1.Obiectivul problemei..... 2

2.Analiza problemei..... 3

3.Proiectare..... 5

4.Implementare..... 8

5.Rezultate..... 12

6.Concluzii..... 14

7.Bibliografie..... 15

1.Obiectivul temei

Obiectivul temei este realizarea unei aplicatii de gestionare a bazelor de date, dupa realzatie client – produs – comanda, utilizand princii de reflexie pentru gestionarea fiecarei clase in parte si utilizarea unei arhitecturi pe nivele (organizarea fiecari parti din proiect in pachete distincte).

Se va discuta si exemplifica in capitolele urmatoare:

- Analiza problemei: cazuri functionale,diagrame, use-case-urile temei abordate
- Proiectare: se va discrie proiectul din punct de vedere a paradigmelor oop si a principiilor folosite(layered architecture/model - view – controller)
- Implementarea: se va descrie detaliat fiecare clasa in parte si interfata/interfetele utilizator creeate
- Rezultate: mediul de testare/testarea aplicatiei
- Concluzii: aspecte noi invatate si imbunatatiri ale aplicatiei pe viitor
- Bibliografie

2.Analiza problemei

Aplicatia de gestionare a bazei de date se imparte pe 3 cazuri distince: tabela clientilor,tabela produselor si tabela comenzilor

Pentru tabela de *clienti* avem urmatoarele date de intrare:

-id-ul unic al clientului

-numele clientului

-adresa de email

Pentru tabela de *produse* avem urmatoarele date de intare:

-id-ul unic al produsului

-numele produsului

-cantitatea care exista in stoc

-pretul fiecarui produs in functie de bucata

Pentru tabela de *comenzi* avem urmatoarele date de intrare:

-se selecteaza un id al unui client deja existent

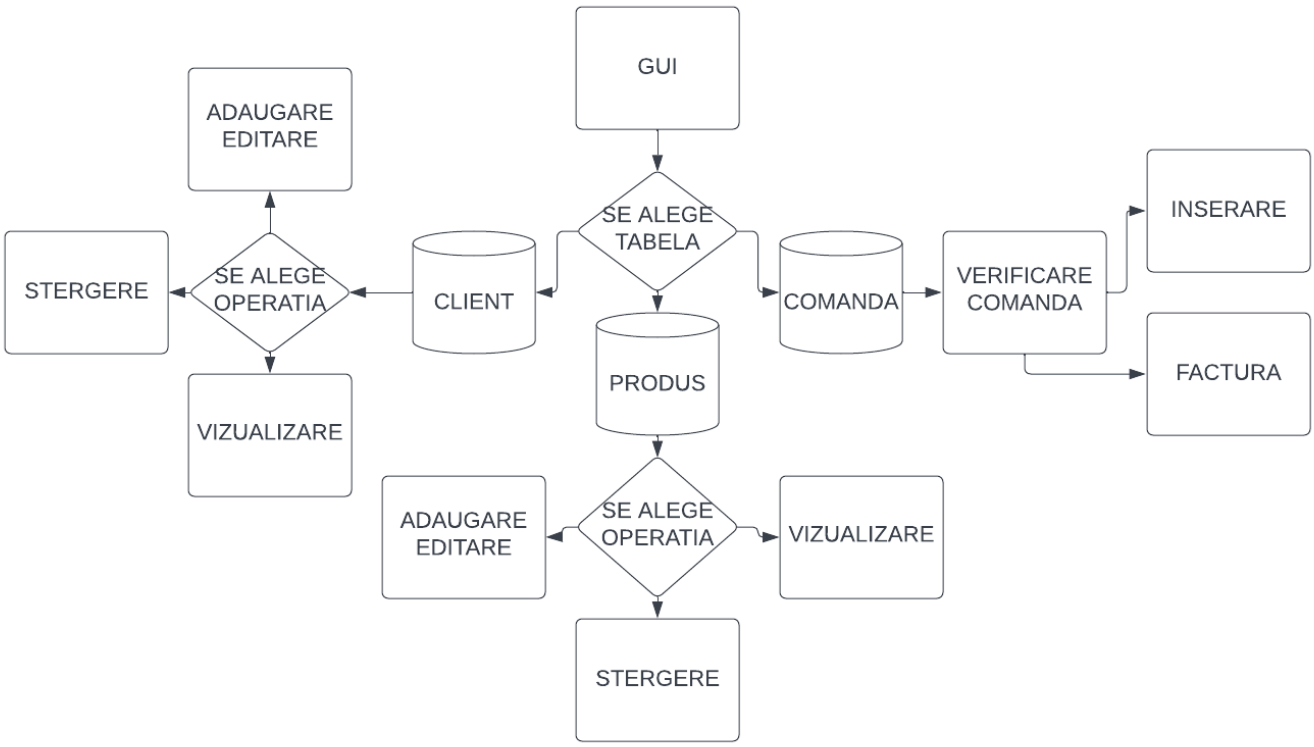
-se selecteaza un id al unui produs deja existent

-se alege un numar de comanda unic

Pentru tabelele de clienti si produs se pot efectua operatii de inserare in tabela(date extrase de la interfata grafica), se poate modifica un camp selectat tot de la interfata grafica sau se poate sterge clientul / produsul in functie de id – ul dat de la interfata grafica. Pentru tabela de comenzi de comenzi se selecteaza id – ul unui client deja existent si id – ul unui produs deja existent, se alege un id unic al comenzii, se intoduce cantitatea dorita pentru achizitionare apoi se testeaza cu ajutorul

conexiunii creeate daca exista destule produse in stoc. In urma testarii, daca se poate realiza comanda, se insereaza toate datele in tabela pentru comenzi.

In flow chartul de mai jos este exemplificat use-case-ul aplicatiei, pasii de executie vor fi explicati mult mai in detaliu in capitolele ce urmeaza



3.Proiectare

Aplicatia de gestionarea a datelor este un proiect complex. In realizarea temei au fost folosite paradigmele de programare orientate pe obiect, principiul reflectarii care se realizeaza cu ajutorul asbtractizarii, arhictectura stratificata, principiile CRUD in gestionarea bazelor de date, conexiuni cu bazele de date folosind limbajul SQL si lucrul cu fisiere text.

Încapsularea (care mai poate fi interpretata ca si ascunderea de informații) asigura faptul ca obiectele încapsulate nu pot schimba starea interna a altor obiecte in mod direct(accesul se poate realiza doar prin intermediul metodelor proprii). In alta ordine de cuvinte, starea obiectului curent poate fi schimbata doar de metodele puse la dispoziție de obiectul respectiv.

Pentru "ascunderea informatiilor", toate datele introduse din interfata care apoi sunt folosite de catre program sunt private, inclusiv clientul, produsul si datele comenzilor, datele clientului si a produsul sunt extrase din interfata grafica dedicata utilizatorului.

Reflectarea este o caracteristică a limbajului de programare Java. Permite unui program Java care se execută să examineze sau să „introspecteze” asupra lui însuși și să manipuleze proprietățile

interne ale programului. De exemplu, este posibil ca o clasă Java să obțină numele tuturor membrilor săi și să le afișeze.

Java Reflection face posibilă inspectarea claselor, interfețelor, câmpurilor și metodelor în timpul execuției, fără a cunoaște numele claselor, metodelor etc. în timpul compilării. De asemenea, este posibil să instanțiați noi obiecte, să invocați metode și să obțineți/setati valori de câmp folosind reflectarea.

Arhitectura stratificata este o metoda de organizare a proiectelor si consta in impartirea sa in patru mari categorii: prezentare, aplicatie, domeniu si infrastructura. Toate categoriile functioneaza ca o singura mare entitate software indiferent de continutul clasei acestora.

CRUD(create,read,update,delete sau pentru bazele de date: insert, select,update si delete): Acestea sunt cele mai de bază patru operațiuni care pot fi efectuate cu majoritatea sistemelor tradiționale de baze de date și sunt coloana vertebrală pentru interacțiunea cu orice bază de date.

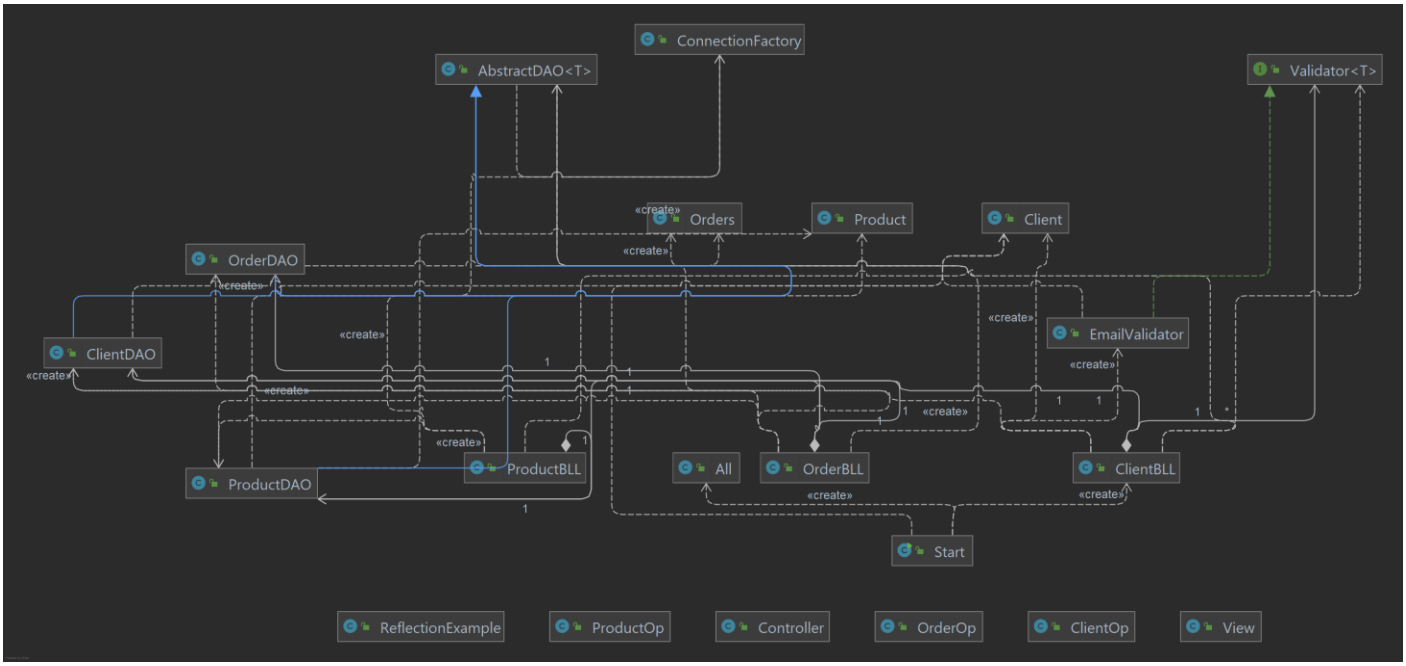
SQL este un limbaj de programare specific pentru manipularea datelor în sistemele de manipulare a bazelor de date relaționale, iar la origine este un limbaj bazat pe algebra relațională

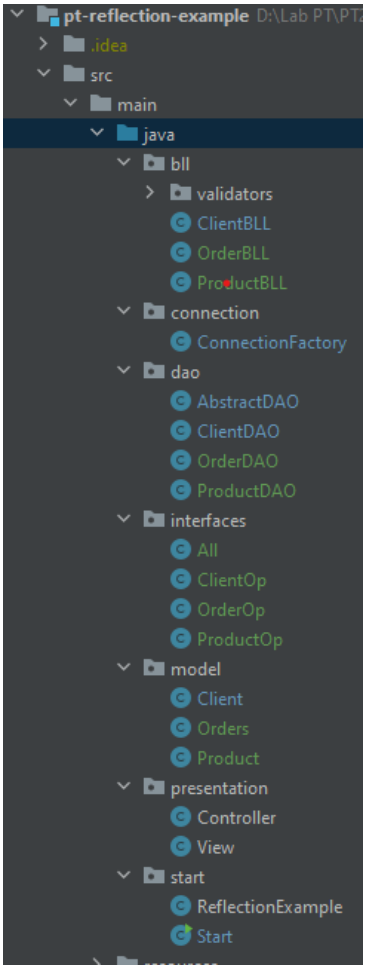
Ca si structuri de date care ies in evidenta au fosit folosite: conexiuni, statementuri,query-uri de executat sub forma de string buildere, Logger si tipul de data abstracta Class care putea sa fie inlocuita cu una dintre cele trei optiuni: client, produs sau comanda.

Ca si algoritmi folositi se regasesc metodele de ToString, algoritmii de inserare,cautare/ select pe tabela, actualizarea tabelei si stergerea unui element din tabela, toate fiind scrise respectand conventiile de scriere a limbajului SQL.

Pentru gestionarea tabelelor s-a folosit MySQL workbench: un mediu de dezvoltare integrat pentru serverul MySQL. Are utilități pentru modelarea și proiectarea bazelor de date, dezvoltarea SQL și administrarea serverului

In diagrama de mai jos se pot observa relatiile exitente intre clase si interfetele creeate si ierarhia pachetelor impreuna cu clasele pe care le detin.


ConnectionFactory
AbstractDAO<T>
Validator<T>
Orders
Product
Client
OrderDAO
ClientDAO
EmailValidator
ProductDAO
ProductBLL
All
OrderBLL
ClientBLL
Start
ReflectionExample
ProductOp
Controller
OrderOp
ClientOp
View
«create»
1


pt-reflection-example D:\Lab PT\PT
.idea
src
main
java
bll
validators
ClientBLL
OrderBLL
ProductBLL
connection
ConnectionFactory
dao
AbstractDAO
ClientDAO
OrderDAO
ProductDAO
interfaces
All
ClientOp
OrderOp
ProductOp
model
Client
Orders
Product
presentation
Controller
View
start
ReflectionExample
Start

Interfetele predefinite sunt JPanel care intra in ajutorul construirii interfetelor grafice, fiecare tabela are JPanelul ei separat pentru interactiune si ActionListener, care reprezinta o interfata care primeste un eveniment de la buton si executa codul/actiunile definite de catre programator.

Interfata definita este Validator care este supusa principiului de reflexie care primeste un string care reprezinta emailul unui client si testeaza daca este intodus in formatul corect al unui email(exemplu@site.com). Verificarea se realizeaza cu ajutorul unui pattern matcher folosind un string incorporat cu Regex.

4.Implementarea

Implementarea aplicatiei a fost realizata folosind paradigmele POO mentionate in capitolul de mai sus: incapsularea si mostenirea, principiul reflectarii, conceptul arhitectural pe straturi si principiile CRUD ale limbajelor Java si SQL.

Pentru usurarea intelegerii vor fi descrise clasele pe pachete, astfel este pusa in evidenta arhitectura stratificata:

1.BLL

Pachetul contine clasele de Client,Produs si Comanda, unde in interioriul lor exista logica de business ale fiecarei componente. Fiecare clasa: ClientBLL,ProductBLL si OrderBLL contin metodele curd de inserare,selectie si stergere a unui element din tabela din baza de date.

Pachetul contine si interfata de validare a emailului, caracteristica a clientului impreuna cu clasa de validare care contine metoda care foloseste pattern matcherul cu ajutorul expresiilor Regex.

2.Connection

Pachetul contine clasa de ConnectionFactory care leaga proiectul de baza de date care exista in MySQL workbench. Clasa contine Stringurile care au in posesie link-ul jdbc catre driver, baza de date creeata, user name-ul si parola utilizatorului care doreste sa o acceseze. ConnectionFactory are ca si metode creeare conexiunii, returnarea ei si inchiderea sa si a ResultSeturilor respectiv Statemehtuilor.

3.DAO

Pachetul contine clasa abstracta DAO care detine metodele CRUD ale proiectului, ele fiind modelate pe principiul reflectarii: metoda de inserare in tabela care itereaza pe campurile fiecarei componenta(produs, client si comanda) si intoduce datele preluate de la interfata, metoda de update care cauta in functie de id – ul unic al fiecarei componenta campul care trebuie modificat si il actualizeaza cu valoarea preluata de la interfata grafica, metoda de stergere care cauta id- ul unic al componentei si il sterge complet din tabela. Pe langa metodele CRUD mai exista si metodele de creeare a query-urilor care respecta conventiile de interogari SQL: creeare stringurilor de selectie, insertie si actualizare.

Pe langa clasa principala AbstractDao exista si clasele ClientDao, OrderDao si ProductDao care extind modelul abstract, drept urmare fiecare componenta dispune de metodele CRUD ale programului principal.

4.Interfaces

Pachetul contine cele 4 intefete grafice: interfata de client, cea de produs si de comanda, iar cele trei pot fi selectate utilizand un butoanele din interfata grafica All care face conexiunea intre ele.

Interfata All contine doar 3 butoane care in urma apasarii lor, folosind ActionListenerul, deschide una din cele 3 componente ale proiectului: clientul, produsul sau comanda.

Interfata ClientOp contine Text Fieldurile care vor urma sa fie completate cu date si butianele aferente operatiilor CRUD. In Text Fielduri se pot insera date ce vor fi dupa inserate in tabela(id unic,email si numele sau), se poate selecta campul care urmeaza sa fie actualizat cu valoarea noua introdusa, se poate selecta un ID pentru stergere si se pot vizualza clientii deja existenti in tabela prin intermediul unui JTabel.

Interfata ProductOp este asemenataoare interfetei Client, doar datele ce trebuie intoduse in Text Fielduri difera. Produsul contine un id unic, numele sau, cantitatea care exista in stoc la momentul respectiv si pretul pe bucata. La fel ca si la client, toate produsele impreuna si atributele lor pot sa fie vizualizate intr-un JTabel.

Interfata OrderOp este putin diferita. Aceasta contine trei Text Fielduri in care se selecteaza id – ul unui client deja existent in baza de date, id – ul unui produs deja existent in baza de date si cantitatea care vrea clientul sa o achizitioneze. In momentul apasarii butonului se desteaza daca exista destule produse existente in stoc, daca da, se va efectua comanda, iar in Text Fieldul dedicat statusului se va afisa mesajul de "Order Completed", daca nu, se va afisa un mesaj de "Order Failure - understock". In urma apasarii butonului, daca comanda este reusita,se va efectua o factura sub forma de fisier txt care contine numarul comenzii, numele clientului, numele produsului, pretul produsul pe bucata si suma totala care trebuie platita.

Interfetele au fost realizate cu ajutorul bibliotecii JavaXSwing dedicata manipularii elementelor GUI.

4.Model

Pachetul contine modelul fiecarei componente Client, Product si Order. Toate datele care reprezinta defapt campurile din tabelele din baza de date sunt incapsulate in fiecare clasa, astfel clasele ce fac parte din business logic pot sa manipuleze datele ce sunt extrase de la interfata grafica. Clasele Client,Product si Order contin toate campurile de id,nume email(client),id,nume,cantitate,pret(produs) si id,client id si product id(comanda) impreuna cu setterele, getterele (incapsulare) si constructorii claselor.

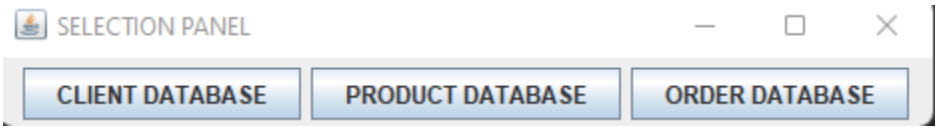
5.Presentation

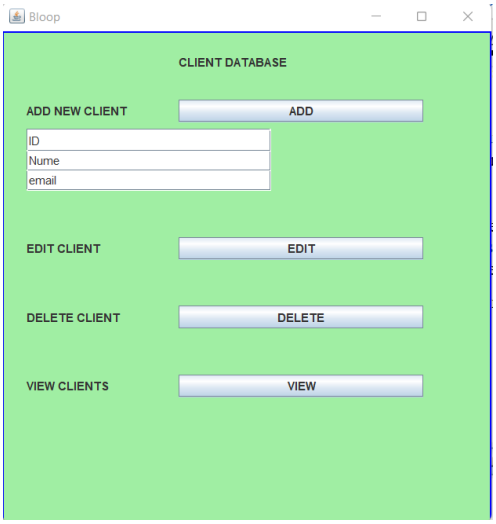
Pachetul de prezentare a fost mutat in interiorul interfetelor, metodele de action listenere au fost incorporate direct in interfete pentru specializarea fiecarei metode.

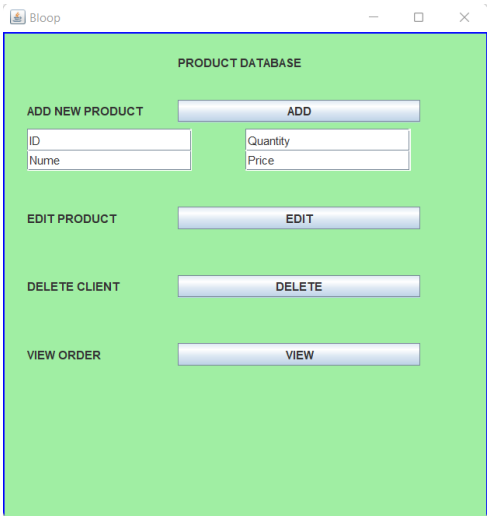
6.Start

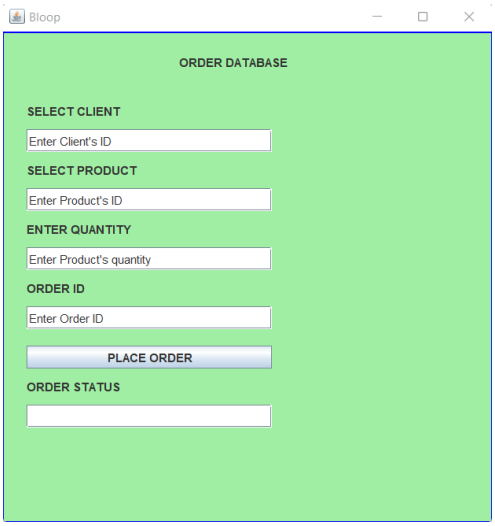
Pachetul de start contine clasa de Start unde este construit Frame – ul principal care permite selectia bazelor de date pe care se vor efectua operatiile, principiile CRUD urmand sa fie implementate cu ajutorul metodelor din ActionListeneri.

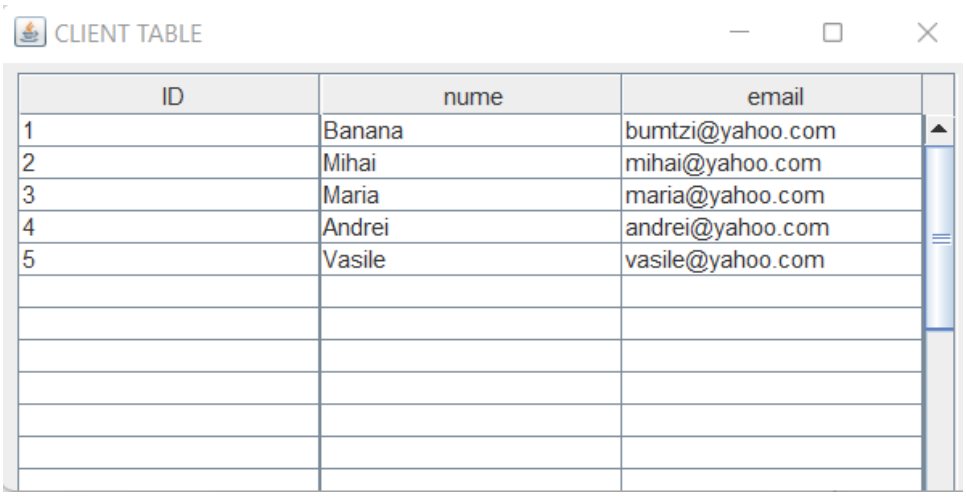
Mai jos sunt atasate interfetele grafice pentru fiecare caz in parte impreuna cu bazele de date din MySQL workbench si datele afisate intr-un JTabel impreuna cu un exemplu de factura generata.



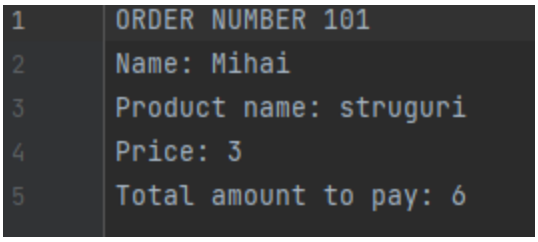








ID	nume	email
1	Banana	bumtzi@yahoo.com
2	Mihai	mihai@yahoo.com
3	Maria	maria@yahoo.com
4	Andrei	andrei@yahoo.com
5	Vasile	vasile@yahoo.com



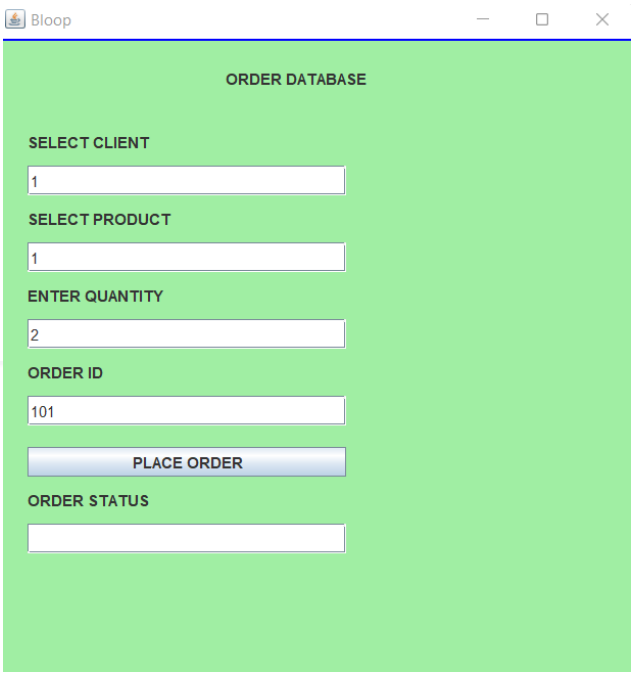
5.Rezultate

Pentru testarea aplicatiei nu a fost folosit un framework dedicat, corectitudinea algoritmului putand fi foar verificata prin urmariarea executiei si verificarea datelor si a modificarilor in MySQL workbench.

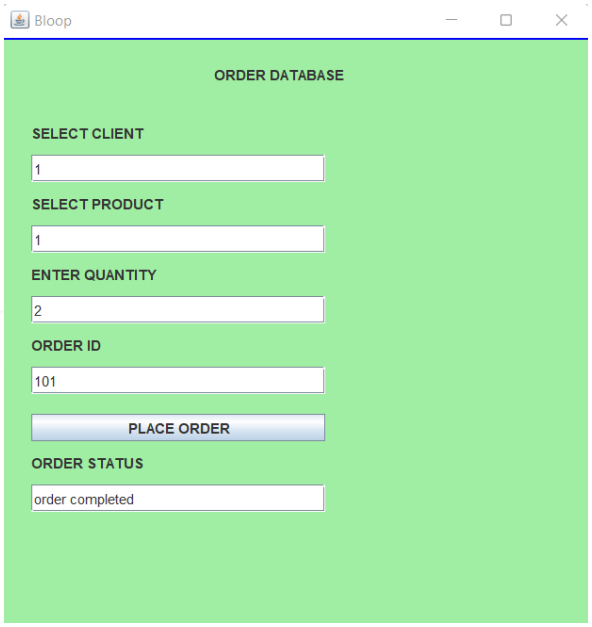
Pentru exemplificarea corectitudinii vor trasa o executie de comanda ce va genera inserarea unei comenzi noi in tabele, modificarea statusului in intefata si generarea unei facturi.

Am ales clientul cu ID-ul 1 care doreste sa achizitioneze produsul cu ID-ul 1 care are ca si nume pere si exista in stoc cu un numar de 10 bucati la pretul de 3 lei bucata. Clientul doreste sa achizitioneze 2 bucati, drept urmare in depozit vor ramane 8 bucati, iar pe factura emisa el va trebuii sa achite suma de 6 lei.

Mai jos este este trasata executia sub forma de imagini:



	ID	nume	quantity	price
▶	1	pere	10	3
	2	struguri	13	3
	3	pepene	5	10
	5	banane	20	5
	7	lapte	10	4
*	NULL	NULL	NULL	NULL



	ID	nume	quantity	price
▶	1	pere	8	3
	2	struguri	13	3
	3	pepene	5	10
	5	banane	20	5
	7	lapte	10	4
*	NULL	NULL	NULL	NULL

6.Concluzii

Prin realizarea proiectului au fost introduse notiuni noi, cum ar fi principiul reflectarii, care ajută la dezvoltarea abilitatilor și să le revizuirea eficacitatatii, mai degrabă decât să continuam să faceți lucrurile așa cum le-ați făcut întotdeauna. Este folosita in testarea software, este folosita mai nou in meta programming.

A mai fost intodus si Arhitectura Stratificata care desparte logica, prezentarea si modelul in pachete diferite pentru a simplifica intelegerea si pentru a despartii componentele, astfel ele sa functioneze fara dependente una fata de cealalta

Imbunatatiri:

-estetica interfetei grafice

-functionarea metodei de find by id a produsului prin tehnica reflectarii(au fost intampinate probleme de argumente ilegale la metoda de creeare obiecte si invocarea de noi instante a claselor abstractizate

-dezvoltarea unui mediu de testare pentru aplicatie

-abstractizarea metodei de afisare a continutului de tabele intr-un JTabel

7.Bibliografie

1.Reflection in Java - https://jenkov.com/tutorials/java-reflection/index.html

2. Reflection in Java - https://www.geeksforgeeks.org/reflection-in-java/

3. Layered architecture in Java – the 4 main layers of layered architecture: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html

4. How to populate JTabel from database : https://stackhowto.com/how-to-populate-jtable-from-database/

5. Step by Step: Making a Simple Crud Application: https://www.mitrais.com/news-updates/step-by-step-making-a-simple-crud-application-using-java-servlet-jsp/