DOCUMENTATIE

TEMA *3*

NUME STUDENT: Ionas Andreea-Georgiana
GRUPA: 30227

CUPRINS

1) Obiectivul temei

(i) Principalul obiectiv al acestei teme consta in implementarea unei aplicatii pentru gestionarea comenzilor clientilor pentru un depozit. Prin aceasta aplicatie vom putea insera, edita si sterge clienti, produse si orders la un depozit.

(ii) Obiectivele secundare ale acestei teme sunt:

Analizarea problemei si identificarea cerintelor	*Determinarea unei strategii de realizare a aplicatiei si stabilirea clara a cerintelor.*
Proiectarea aplicatiei de gestionare a comenzilor	*Crearea claselor necesare si stabilirea unei modalitati de implementare.*
Implementarea aplicatiei de gestionare a comenzilor	*Legarea claselor si a programului Java cu baza de date.*
Testarea aplicatiei de gestionare a comenzilor	*Introducerea unor date in baza de date si rularea programului pentru a determina corectitudinea aplicatei.*

** Toate obiectivele secundare vor fi detaliate si descrise in capitolul 4. Implementare.*

2) Analiza problemei, modelare, scenarii, cazuri de utilizare

Cerinte functionale:

- Aplicatia trebuie sa simuleze un depozit care primeste comenzi de diferite produse de la clienti si trebuie sa onoreze comenzile.
- Datele trebuie introduse prin intermediul interfetei grafice.
- Datele trebuie validate apoi introduse in baza de date.
- La plasarea unei comenzi se va verfica stocul iar in cazul in care stocul este insufficient, comanda nu va fi plasata.

Cerinte non functionale:

- Datele trbuie citite din baza de date.

- Datele introdse trbuie scrise in baza de date.

- Interfata trebuie sa fie usor de inteles si utilizat.

Descriere user-case:

Use Case: adaugare produs
Primary Actor: angajat
Main Success Scenario:

1) *Angajatul selecteaza optiunea de a adauga un produs nou.*
2) *Aplicatia va afisa un formular in care angajatul sa introduca detaliile produsului.*
3) *Angajatul va introduce detaliile.*
4) *Angajatul apasa butonul de "Add".*
5) *Aplicatia stocheaza produsul in baza de date si afiseaza un mesaj de success.*

Alternative Sequence: Sunt introduse valori invalide

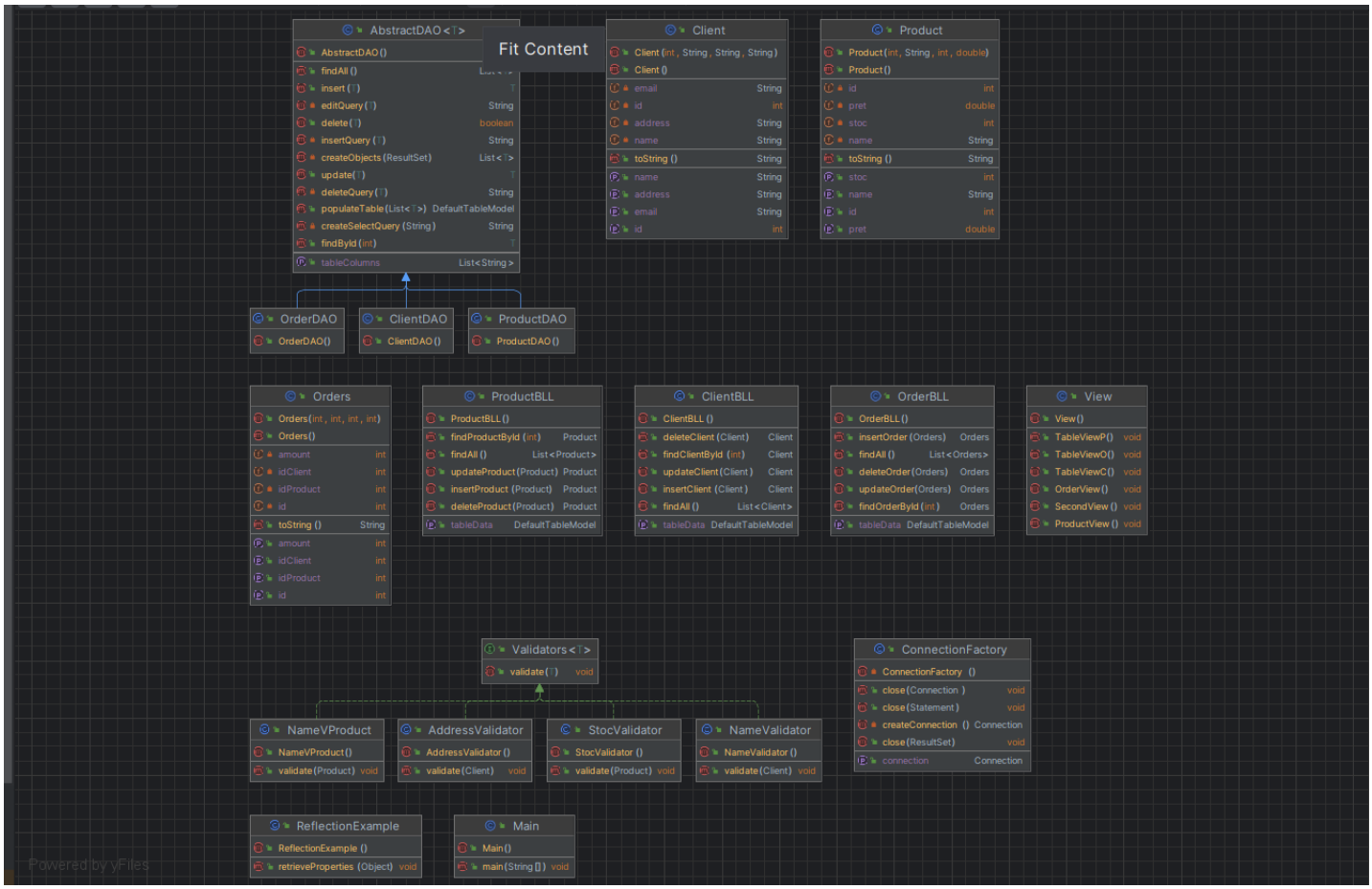
1) *Angajatul introduce o valoare negativa pentru stoc.*
2) *Aplicatia afiseaza un mesaj de eroare si roaga angajatul sa introduca o valoare valida.*
3) *Scenariul se reia de la pasul 3.*

3) Proiectare

Proiectarea OOP a aplicatiei:

In primul rand, aplicatia foloseste Arhitectura stratificată (Layered Architecture) care este un stil arhitectural utilizat în dezvoltarea software, care împarte aplicația în mai multe straturi logice distincte, fiecare cu un rol și responsabilități bine definite. Această abordare are ca scop separarea preocupărilor și organizarea clară a funcționalității aplicației.

- Business Logic -- acesta este stratul în care se află logica de afaceri a aplicației. Aici sunt implementate regulile de afaceri, procesele și operațiile specifice aplicației. Acest strat se concentrează pe manipularea și procesarea datelor, aplicarea regulilor și gestionarea fluxului de lucru al aplicației. Clasele din acest pachet sunt:
    a) ClientBLL
    b) ProductBLL
    c) OrderBLL
    d) Si un pachet in care avem metode de validare a datelor (Validators)



- Connection -- acest strat gestionează conexiunea și interacțiunea cu bazele de date utilizate de aplicație. El se ocupă de configurarea și gestionarea conexiunilor cu baza de date, executarea interogărilor și tranzacțiilor, precum și de maparea datelor între structurile aplicației și schema bazei de date. Clasele din pachet:
    a) Connection Factory

- Data Access – acesta este stratul responsabil de interacțiunea cu resursele de stocare a datelor, cum ar fi bazele de date sau serviciile web. Aici se efectuează operațiile de citire, scriere și modificare a datelor. Acest strat abstractizează modul de acces la date, astfel încât celelalte straturi să nu fie dependente de detalii specifice ale stocării datelor.
    a) AbstractDAO
    b) ClientDAO
    c) ProductDAO
    d) OrderDAO

- Model -- întâlnim clasele principale care descriu principalele obiecte sau entități cu care lucrăm în aplicație. Aceste clase modelează structura și comportamentul acestor obiecte și permit manipularea și gestionarea datelor.
    a) Client
    b) Orders
    c) Product

- Presentation -- Acesta este stratul care interacționează direct cu utilizatorul și furnizează o interfață grafică sau o interfață utilizator de linie de comandă. Acest strat se ocupă de prezentarea datelor și interacțiunea utilizatorului cu aplicația.
    a) View

- Start – strartul care permite programului să inspecteze și să manipuleze dinamic obiecte, metode și atribute ale claselor în timpul execuției. Reflection oferă informații despre clase, cum ar fi numele, metodele, câmpurile și constructorii, și permite accesul la acestea și invocarea metodelor în mod dinamic.
    a) Main
    b) ReflectionExample.

Clasele din Data Access, in principal clasa AbstractDAO implementam metodele de inserare, editare, stregere si afisare a datelor. Iar clasele ClientDAO, ProductDAO si OrderDAO extind clasa AbstractDAO.

*Mai multe informatii in capitolul 4. Implementare

4) Implementare

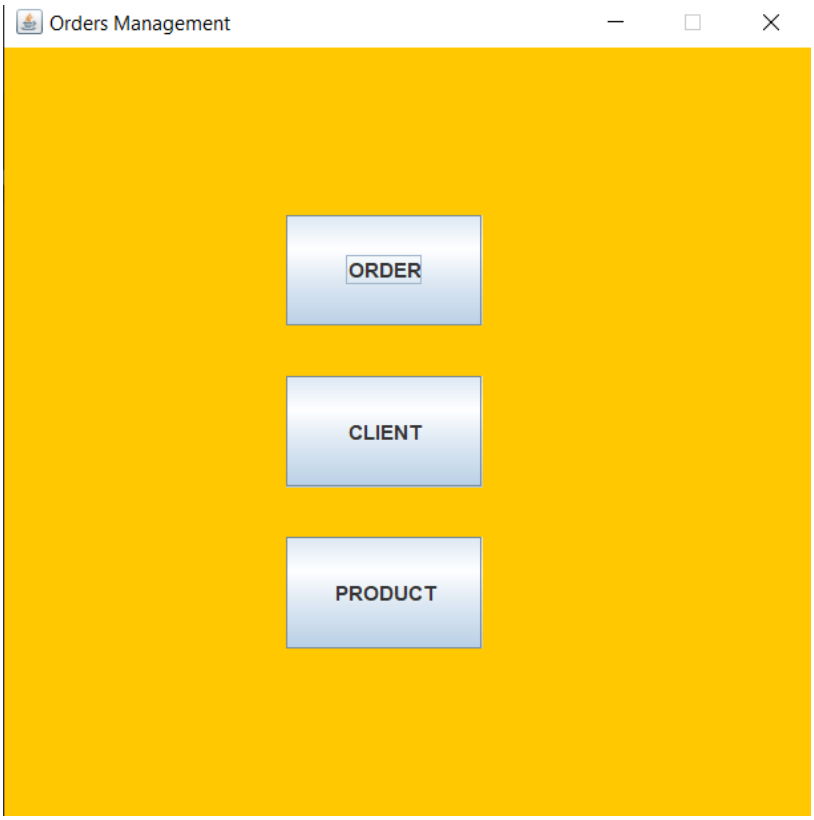
In clasele ProductBLL, OrderBLL si ClientBLL se creaza obiecte de tipil product, client si orders care mai apoi apeleaza metodele din AbstractDAO: findAll(), findById(), insert(), edit(), delete().

In clasa ConnectionFactory se realizeaza conexiunea cu baza de date utilizata, aceasta clasa are un constructor, o metoda createConnection() care creaza conexiunea cu baza de date, o metoda getConnection(), care ne returneaza conexiunea create, si 3 metode de close() pentru a incheia conexiunea, statement-ul si setul de rezultate.

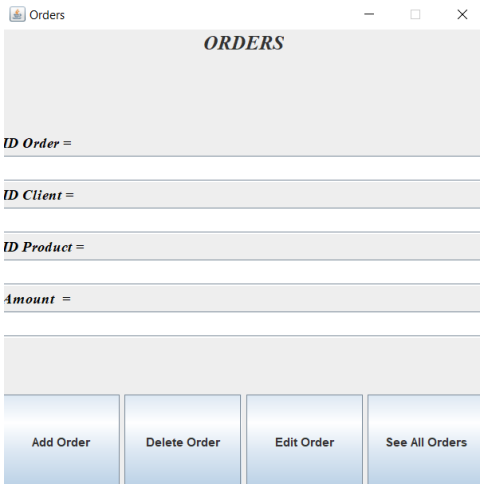
In clasa AbstractDAO avem un constructor, avem metode care creeaza query-urile in functie de operatia pe care vrem sa o efectuam: delete, insert, find all si update. De asemenea, avem metodele care implementeaza aceste operatii.

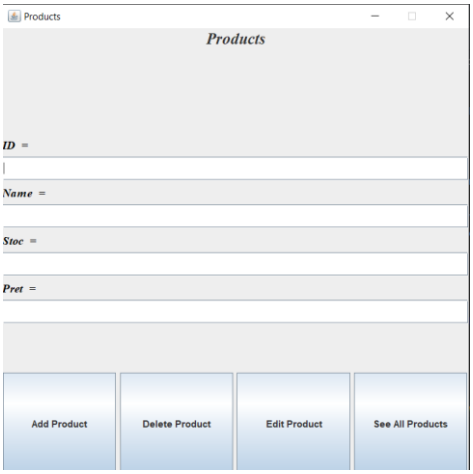
In clasele din pachetul Model avem obiectele de baza folosite in acest proiect: Client, Product si Order. Clasa Client care ca atribute un id, un nume, o adresa si o adresa de email, iar ca metode un constructor si getters si setters.
Clasa Product are un id, un nume, un stoc si un pret, iar ca metode avem un constructor, getters si setters.
Clasa Order are un id, un id corespunzator clientului, un id coresounzator produsului si o cantitate, iar ca metode avem un constructor, getters si setters.

Clasa View este corespunzatoare interfetei grafice care are o pagina principala din care putem alege cu ce obiect dorim sa lucram:

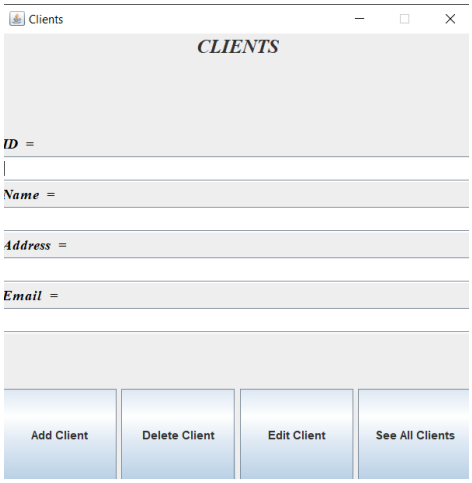


Apoi dupa ce am ales obiectul cu care dorim sa lucram, se va deschide o fereastra in care





putem sa inseram caracteristicile specifice:



Datele introduse in campurile din interfata vor fi stocate intr-o baza de date si afisate sub forma de table la apasarea butonului see all.

In clasa Reflection Exemple avem o metoda retrieve properties care ne perimite sa afisam proprietatile obiectelor. Cu ajutorul Java Reflection, putem obține informații despre clase, metode și câmpuri, crea și manipula dinamic obiecte, accesa și invoca metode, precum și modifica valori de câmpuri. Aceasta deschide noi posibilități în gestionarea și adaptarea comportamentului aplicației în funcție de condițiile din timpul execuției.

5) Rezultate

In urma implementarii si testarii aplicatei de manageriere a comenzilor putem observa urmatorul comportament:

Daca dorim sa inseram un client:

- id-ul: 99, numele : Pop Ioan, adresa : Observator 74 si email-ul : ioan.pop@gmail.m

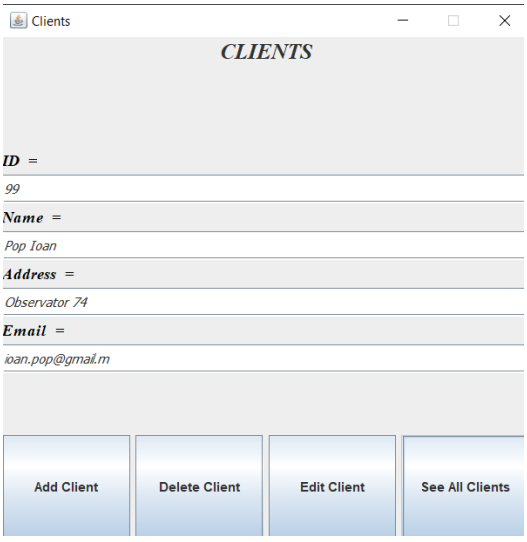
Produs:

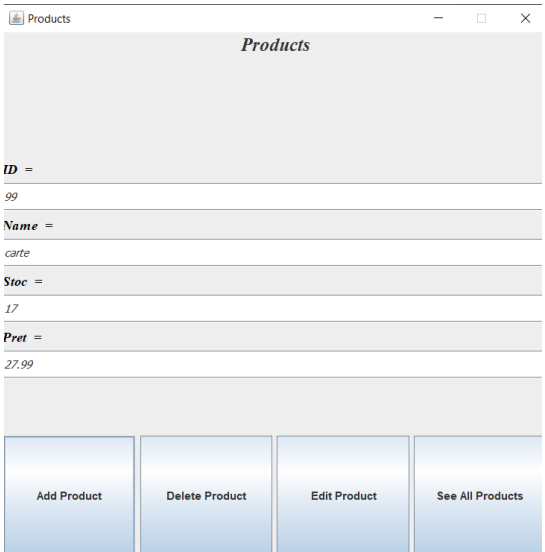
- id-ul: 99, denumire: carte, stoc: 17, pret: 27.99

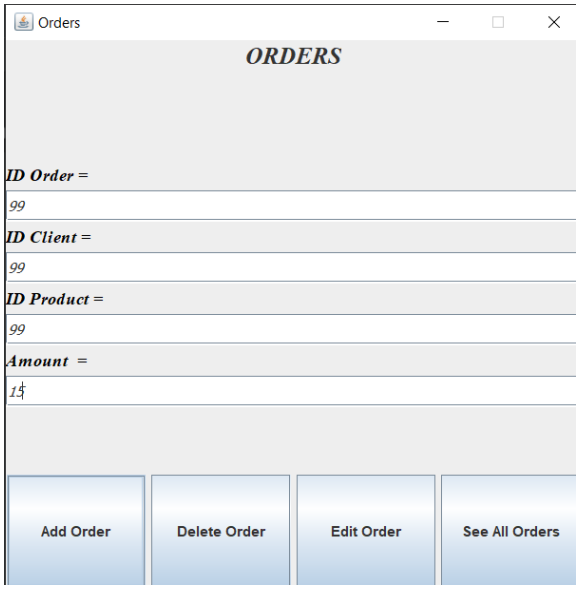
Apoi dorim sa editam clientul cu id-ul 99, sa locuiasca in Bucuresti 12

Apoi dorim sa plasam o comanda:

- id: 99, id client: 99, id prdus: 99, cantitate: 15.







Clientul:

3	Pop Andreea	Zorilor 19	andrePop@gmail.com
4	Angela Constantin	Dorobantilor 56	angel.const@yahoo.ro
99	Pop Ioan	Bucuresti 12	ioan.pop@gmail.m

Order:

4	6	7	3
99	99	99	15

Produsul (dupa ce order-ul a fost plasat):

4	Ciocolata	0	12.6
5	Fanta	6	8.0
6	Umbrela	159	45.99
7	Ruj Dior	9	800.0
99	carte	2	27.99

6) Concluzii

Din aceasta tema care a constat in dezvotarea unei aplicatii Java pentru a plasa comenzi si de a introduce produse intr-o baza de date MySQL am invata cum sa utilizam baza de date MySQL si cum sa efectuam operatii CRUD (Create, Read, Update, Delete) pentru gestionarea datelor, am invatat cum sa cream interfete grafice pentru a face interactiunea cu aplicatia mai intuitiva si eficienta si am invatat cum sa gestionam erorile care pot sa apara. De asemenea, am invatat sa lucram cu Java Reflection mai exact am invatat cum să obținem informații despre clase, metode și câmpuri într-un mod dinamic, permițându-ne să explorăm și să accesăm caracteristicile obiectelor în timpul rulării programului.Am invatat cum să creăm noi instanțe ale claselor în timpul de execuție și să accesăm și să modificăm valorile câmpurilor acestora.

7) Bibliografie

1. *Java Reflection: https://www.oracle.com/technical-resources/articles/java/javareflection.html*

2. *Java Doc: https://www.oracle.com/ro/technical-resources/articles/java/javadoc-tool.html*

3. *Java and Database: https://docs.oracle.com/javase/tutorial/jdbc/basics/sqldatasources.html*

4. *Layered Architecture: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html*