**DOCUMENTAȚIE**

Tema 3

NUME STUDENT: LAURA DAROLTI

GRUPA: 5

CUPRINS

1. Obiectivul temei

2. Proiectare

3. Implementare

4. Rezultate

5. Concluzii

6. Bibliografie

1. Obiectivul Temei

Obiectivul acestui proiect este de a implementa o aplicație de gestionare a comenzilor pentru procesarea comenzilor clienților într-un depozit. Bazele de date relaționale sunt utilizate pentru a stoca produsele, clienții și comenzile. Aplicația este proiectată conform modelului de arhitectură pe straturi utilizează următoarele clase:

• Clasele Model - reprezintă modelele de date ale aplicației

• Clasele Logicii de Afaceri - conțin logica aplicației

• Clasele de Prezentare - clase legate de interfața grafică a utilizatorului (GUI)

• Clasele de Acces la Date - clase care conțin accesul la baza de date

2.Proiectare

Pentru proiectarea OOP a aplicației, am creat următoarele clase si pachete:

Pachetul “Model” conține clasele “Client”, “Product” si “Order”:

Clasa Client este un model de date care reprezintă o entitate de client în aplicație. Aceasta conține proprietăți precum numele, ID-ul, adresa, vârsta și adresa de email. Această clasă este utilizată pentru a stoca și gestiona informațiile despre clienți în cadrul sistemului.

Clasa oferă un constructor care permite crearea unui nou obiect Client prin furnizarea detaliilor necesare despre client.

Clasa Product reprezintă un model de date care prezintă un produs în aplicație. Aceasta conține proprietăți precum nume, ID, preț și cantitate. Clasa este folosită pentru stocarea și gestionarea informațiilor despre produse în cadrul sistemului. Ea oferă metode pentru a accesa și modifica atributele produsului.

Clasa Order reprezintă o comandă efectuată de un client pentru un produs specific în aplicație.

Clasa ConnectionFactory oferă metode pentru crearea și gestionarea conexiunilor la baza de date în aplicație. Aceasta încapsulează logica pentru stabilirea unei conexiuni la o bază de date MySQL utilizând driverul, URL-ul, numele de utilizator și parola specificate. Constructorul se asigura că este creată doar o singură instanță a clasei ConnectionFactory.

Pachetul ”Presentation” reprezintă interfața grafică a utilizatorului pentru gestionarea sistemului de comenzi si conține clasele “ClientView”,”ProductView” si”OrderView”:

Clasa “ClientView” oferă funcționalități pentru adăugarea, editarea și ștergerea informațiilor despre clienți.

Metoda addClient() cere utilizatorului să introducă informațiile despre client prin câmpurile de introducere și creează un nou obiect Client pe baza introducerii. Noul client este apoi adăugat în lista de clienți și inserat în baza de date.

Metoda displayClients() recuperează lista de clienți din baza de date și actualizează dinamic clientInfoPanel pentru a afișa informațiile despre clienți. Metoda șterge conținutul existent, creează componente JPanel pentru fiecare client și le adaugă în panou.

Metoda editClient() permite utilizatorului să selecteze un client din listă și să modifice detaliile acestuia. Aceasta prezintă informațiile clientului selectat în câmpurile de introducere, care pot fi editate. Modificările sunt apoi salvate și actualizate în baza de date.

Metoda deleteClient() permite utilizatorului să selecteze un client din listă și să îl șteargă. Clientul selectat este eliminat din listă și șters din baza de date.

Clasa ProductView reprezintă interfața utilizator pentru gestionarea produselor. Permite utilizatorilor să adauge, editeze și șteargă produse, precum și să vizualizeze produsele existente. Clasa interacționează cu clasele ProductController și ProductDAO pentru a efectua operațiile necesare.

Analog, Clasa OrderView reprezinta interfață utilizator pentru gestionarea comenzilor. Ea permite utilizatorului sa selecteze clientul si produsul pentru care dorește sa facă comanda si sa introducă cantitatea produsului. Metodele implementate aferente clasei sunt getOrder, addOrder, displayOrders, editOrder, si deleteOrder.

Pachetul “BusinessLogic” este responsabil pentru implementarea metodelor de logica a aplicației si conține următoarele clase:

Clasa ClientController este responsabilă de gestionarea datelor clientului. Aceasta oferă metode pentru editarea clienților existenți, ștergerea clienților și inserarea de noi clienți în baza de date.

Clasa ProductController este responsabilă de efectuarea operațiilor asupra produselor. Aceasta oferă metode pentru editarea produselor existente, ștergerea produselor și inserarea de noi produse în sistem.

Clasa OrderController este responsabilă de gestionarea datelor comenzilor.

La introducerea unei noi comenzi, metoda preia informațiile despre produs din ProductDAO în funcție de numele produsului. Calculează suma totală pe baza prețului produsului și a cantității de produse. Apoi, introduce comanda în baza de date folosind OrderDAO și actualizează cantitatea produsului prin scăderea cantității de produse.

La editarea unei comenzi, metoda actualizează informațiile comenzii precum numele clientului, numele produsului, suma totală și cantitatea de produse. De asemenea, calculează noua sumă totală pe baza prețului produsului și cantității de produse.

Pachetul “DAO”:

Clasele ClientDAO,ProductDao si OrderDao furnizează operații de acces la date pentru entitatea Client, Product respectiv Order. Acestea permit inserarea, actualizarea, ștergerea și obținerea informațiilor despre clienți din baza de date apelând metodele insert(), getAllClients(), updateClient(), delete().

Diagrama UML de clase este următoarea:

DAO

ClientDao ProductDao OrderDao

Insert():void

getAllClients:List<Client>

updateClient():void

delete():void

Connection

ConnectionFactory

createConnection():Connection

getConnection():Connection

close():void

close():Statement

close():ResultSet

Presentation

ClientView ProductView OrderView

Initialize():void

displayClients():void

addClient():void

editClient():void

deleteClient():void

Model

Client

Product

Order

BusinessLogic

ClientController ProductController OrderController

insertClient():void

editClient():void

deleteClient():void

3.Implementare

Implementarea proiectului consta in crearea unor componente pe straturi, ce comunica intre ele pentru a realiza sistemul de gestiune a comenzilor.

Intrările utilizatorului sunt transmise claselor de controller corespunzătoare (ClientController, ProductController și OrderController).

Clasele de controlor primesc intrarea, efectuează orice validări sau transformări necesare și apelează metodele corespunzătoare în clasele DAO corespunzătoare (ClientDAO, ProductDAO și OrderDAO).

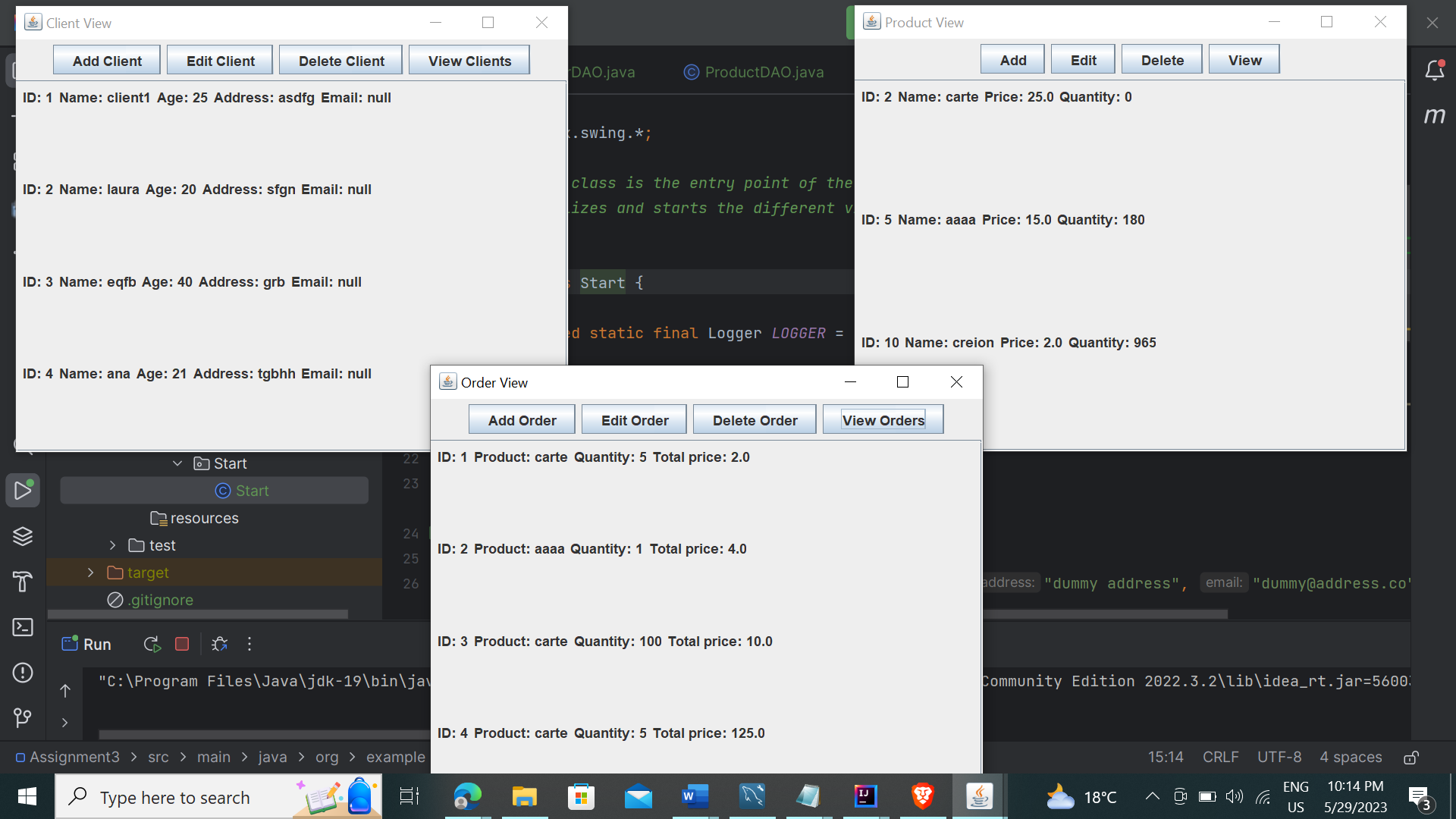
Clasele DAO interacționează cu baza de date utilizând clasa ConnectionFactory pentru a executa interogări SQL sau declarații pentru extragerea, inserarea, actualizarea sau ștergerea datelor.

Rezultatele sau răspunsurile de la baza de date sunt returnate prin clasele DAO către clasele de controlor.

Clasele de controller pot efectua orice prelucrări suplimentare, dacă este necesar, și trec datele înapoi la clasele de vizualizare corespunzătoare pentru afișarea acestora utilizatorului.

În ansamblu, comunicarea între clase implică clasele de vizualizare care captează intrările utilizatorului, clasele de controlor care coordonează acțiunile și interacționează cu clasele DAO, iar clasele DAO care gestionează operațiile la nivel de bază de date prin intermediul clasei ConnectionFactory.

4.Rezultate



5.Concluzii

In concluzie, acest proiect oferă o implementare de bază a unui sistem de gestionare a datelor pentru clienți, produse și comenzi. Proiectul include diverse componente, precum clasele entităților (Client, Product, Order), clasele de acces la date (ClientDAO, ProductDAO, OrderDAO) pentru interacțiunea cu baza de date, clasele de vizualizare (ClientView, ProductView, OrderView) pentru interacțiunea cu utilizatorul și clasele de control (ClientController, ProductController, OrderController) pentru coordonarea acțiunilor între vizualizări și DAO-uri.

6.Bibliografie

[Connect Java to a MySQL Database | Baeldung](https://www.baeldung.com/java-connect-mysql)

[Reflection in Java - GeeksforGeeks](https://www.geeksforgeeks.org/reflection-in-java/)