DOCUMENTATIE

TEMA 1

NUME STUDENT: SAMOILA DENIS-VALENTIN
GRUPA: 30223

Cuprins

1. Obiectivul temei
2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie

1.Obiectivul temei

Obiectivul principal al acestei teme este de a invata lucrul cu bazele de date, conexiunea cu acestea, dar si prelucrarea de informatii, precum: inserare, modificare si stergere.

Pentru acest proiect avem de adaugat client, produse, comenzi, modificarea si stergerea acestora.

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

Cerinte functionale: -furnizarea unei interfețe grafice cu utilizatorul, cu ferestre separate pentru operațiile legate de clienți și produse, permițând utilizatorilor să interacționeze ușor cu sistemul

-implementarea unei ferestre pentru crearea comenzilor de produse, permițând utilizatorilor să selecteze produse și clienți existenți și să specifice cantitatea dorită pentru comandă. Se va afișa un mesaj în cazul în care nu sunt suficiente produse disponibile, iar stocul produsului va fi decrementat după finalizarea comenzii

-Utilizarea tehnicilor de reflexie pentru a genera dinamic antetele tabelelor bazate pe proprietățile obiectelor și pentru a popula tabelele cu valorile corespunzătoare

Cerinte non-functionale: -utilizarea convențiilor de denumire Java pentru clase, metode și variabile

-documentarea claselor folosind Javadoc și generarea fișierelor JavaDoc corespunzătoare pentru o mai bună documentare a codului

-stocarea datelor aplicației într-o bază de date relațională cu cel puțin trei tabele: Client, Product și Order

-implementarea unei arhitecturi stratificate cu cel puțin patru pachete: dataAccessLayer, businessLayer, model și presentation

-definirea unei clase Bill imutabile în pachetul Model folosind înregistrări Java, cu fiecare comandă generând un obiect Bill stocat într-o tabelă Log. Asigurarea că facturile pot fi doar inserate și citite din tabela Log, fără a fi permise actualizări

Use-case:

Crearea unui client:

Angajatul poate introduce datele unui nou client în sistem

Angajatul accesează funcționalitatea de adăugare a unui client nou în system

Angajatul completează informațiile despre noul client (nume, adresă)

Angajatul salvează datele clientului în sistem

Exceptii:

Date incomplete: Dacă angajatul nu completează toate câmpurile obligatorii, sistemul afișează un mesaj de eroare și solicită completarea acestora

Eșec de conexiune: În cazul în care sistemul întâmpină probleme la salvarea datelor în baza de date, se afișează un mesaj de eroare și se revine la pasul anterior

Editarea unui client:

Angajatul poate modifica informațiile despre un client existent în system

Exceptii:date invalide, esec de conexiune

Stergerea unui client existent:

Angajatul poate șterge un client existent din system

Exceptii: actiune anulata

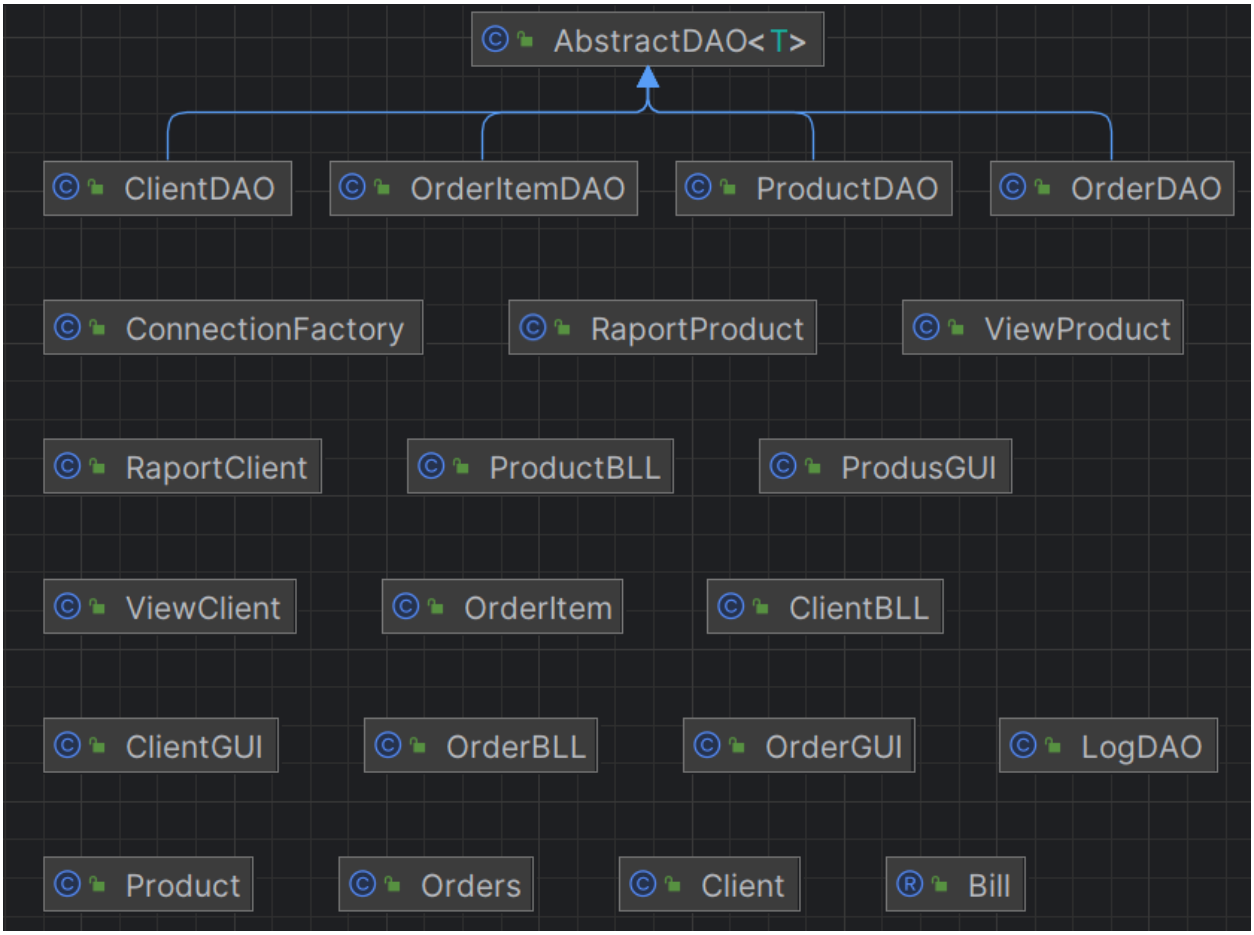
Crearea unei comenzi de produse:

Angajatul poate crea o comandă pentru produse în system

Exceptii: produse indisponibile, esec de conexiune

3.Proiectare

Diagrama UML de clase



Structuri de date:

List<OrderItem> orderItemList = orderItemDAO.findAll();

Aceasta este folosită pentru a stoca o colecție dinamică de obiecte, de exemplu, lista de elemente comandate sau lista de elemente dintr-o comandă

HashMap<Integer,Product> productMap=new HashMap<>();

productMap.put(productid,product);

Această structură este utilizată pentru a realiza mapări între chei și valori. În unele cazuri, este folosită pentru a asocia obiecte cu ID-urile lor corespunzătoare, pentru a facilita operațiile de căutare sau actualizare

FileWriter myWriter=new FileWriter(“bill.txt”);

Această structură este utilizată pentru a scrie date într-un fișier text. În acest caz, este folosită pentru a genera și scrie facturile de vânzare într-un fișier text

4.Implementare

Clasa **ClientBLL** este responsabilă pentru gestionarea operațiilor legate de clienți în cadrul aplicației. Aceasta interacționează cu obiectele **Client**, **Order** și **OrderItem**, precum și cu clasele de acces la date **ClientDAO**, **OrderDAO** și **OrderItemDAO**

Metodele acestei clase sunt:

1. **insertClient(Client client)**: Inserează un client în baza de date.
2. **findID(Client client)**: Găsește ID-ul unui client dat.
3. **deleteClient(Client client)**: Șterge un client din baza de date, împreună cu comenzile și elementele comenzilor asociate acestuia

Clasa **OrderBLL** controlează operațiile legate de comenzile de produse. Aceasta gestionează interacțiunea între clienți, produse și comenzile de vânzare. Utilizează obiectele **Orders**, **OrderItem**, **Product**, **Client** și interactează cu clasele de acces la date **OrderDAO**, **OrderItemDAO**, **ProductBLL**, **ClientBLL** și **ClientDAO**

Principalele metode ale acestei clase sunt:

1. **insertOrder(Client client, Product product, int productQuantity)**: Inserează o comandă nouă în baza de date, actualizând, de asemenea, stocul produselor și generând o factură într-un fișier text.
2. **findOrderID(Client client)**: Găsește numărul comenzii pentru un client dat.

Clasa **ProductBLL** este responsabilă pentru operațiile legate de produse și stocul acestora. Aceasta interacționează cu obiectele **Product** și **OrderItem**, precum și cu clasele de acces la date **ProductDAO** și **OrderItemDAO**

Principalele metode ale acestei clase sunt:

1. **insertProduct(Product product)**: Inserează un produs în baza de date sau actualizează stocul unui produs existent.
2. **findID(Product product)**: Găsește ID-ul unui produs dat.
3. **getProduct(int id)**: Găsește produsul din baza de date cu ID-ul dat.
4. **deleteProduct(Product product)**: Șterge un produs din baza de date, împreună cu comenzile și elementele comenzilor asociate acestuia.
5. **updateProduct(Product product, int Quantity)**: Actualizează stocul unui produs în funcție de cantitatea comandată, returnând **-1** în cazul în care stocul este insuficient și **1** în cazul în care actualizarea s-a efectuat cu succes

5.Testare

Testarea a fost realizata prin intermediul unui fisier text, testandu-se fiecare caz posibil de folosire

6.Concluzii

Proiectul acesta m-a invatat sa fac legatura dintre o baza de date MySql si o aplicatie Java.

O dezvoltare ulterioara ar putea fi ca doar angajatul sa faca aceste tipuri de modificari, prin logarea in contul acestuia inainte de aparitia ferestrelor de modificare.

7.Bibliografie

Cursuri si link-urile de la site-urile de pe laboratoare.