DOCUMENTATIE
TEMA 3

Elev: Rad Vladut Cosmin George
Grupa: 30225
An: 2 CTI

CUPRINS

1. Introducere

În zilele noastre, tehnologia ocupă un rol foarte important în majoritatea domeniilor. Aceasta ne ușurează munca de zi cu zi și ne ajută să economisim timp important. Datorită evoluției miraculoase a tehnologiei, bazele de date nu mai înseamnă nenumărate foi și dosare, ci niște simple click-uri.

O bază de date, uneori numită și bancă de date (abreviat BD), reprezintă o modalitate de stocare a unor informații și date pe un suport extern (un dispozitiv de stocare), cu posibilitatea extinderii ușoare și a regăsirii rapide a acestora. La prima vedere sarcina poate părea banală. Totuși, în condițiile în care este vorba de a lucra cu milioane de elemente, fiecare putând consta din cantități de date care trebuie accesate simultan prin Internet de către mii de utilizatori răspândiți pe întreg globul; și în condițiile când disponibilitatea aplicației și datelor trebuie să fie permanentă (de ex. pentru a nu pierde ocazia de a încheia afaceri), soluțiile bune nu sunt de loc simple. De obicei o bază de date este memorată într-unul sau mai multe fișiere. Bazele de date sunt manipulate cu ajutorul sistemelor de gestiune a bazelor de date. Cel mai răspândit tip de baze de date este cel relațional, în care datele sunt memorate în tabele. Pe lângă tabele, o bază de date relațională mai poate conține: indecși, proceduri stocate, declanșatori, utilizatori și grupuri de utilizatori, tipuri de date, mecanisme de securitate și de gestiune a tranzacțiilor etc.

Java Database Connectivity (JDBC) este o interfață de programare pentru limbajul Java, care definește modul în care un client poate accesa o bază de date. Face parte din platforma Java Standard Edition, de la Oracle Corporation. Aceasta oferă metode de interogare și actualizare adatelor dintr-o bază de date și este orientată spre bazele de date relaționale. O punte JDBC-ODBC permite conexiuni la orice sursă de date accesibilă ODBC din mediul gazdă Java virtual machine.

JDBC („Java Database Connectivity”) permite să existe implementări multiple și să fie utilizate de aceeași aplicație. API oferă un mecanism pentru încărcarea dinamică a pachetelor Java corecte și înregistrarea acestora la Managerul de driver JDBC. Driver Manager este utilizat ca fabrică de conexiuni pentru crearea conexiunilor JDBC.

2. Problema și soluția

Se cere să se implementeze un program în limbajul de programare Java, cu interfață grafică, utilizând JDBC, care procesează comenzile unor clienți pentru un depozit.

Se vor utiliza bade de date relaționale pentru a stoca produsele, clienții și comenzile. Mai mult, aplicație trebuie să fie structurată în pachete, utilizând o arhitectură "stratificată", care să conțină cel puțin următoarele:

- Model classes- reprezintă modelele de date ale aplicației
- Business Logic classes- conțin logica aplicației
- Presentation classes- clasele GUI
- Data access classes- clasele care conțin accesul la baza de date.

Cu ajutorul interfeței grafice, utilizatorul va avea posibilitatea de a adăuga, șterge sau modifica un client din baza de date, de a adăuga, șterge sau modifica un produs din baza de date și de a adăuga o comandă validă în baza de date, în funcție de ID-ul unui client și ID-ul produsului. Atunci când comanda va fi plasată, stocul produsului va scădea cu numărul respectiv comenzii.

3. Obiective

Obiectivul principal este de a implementa un program cu interfață grafică prin care utilizatorul poate gestiona baza de date după bunul plac, într-un mod simplu și rapid. Baza de date conține 3 tabele: clients, products și orders. Programul va prelua datele din bază, le va modifica după cerințele utilizatorului și va afișa rezultatele în interfața grafică.

Obiectivele secundare sunt:

- Analizarea problemei și identificarea cerințelor.
- Proiectarea programului.
- Implementarea programului.
- Testarea programului.

4. Analiza

Proiectul este alcătuit din 13 clase structurate corespunzător în 6 pachete, fiecare dintre aceste clase având un rol bine definit.

4.1 Pachetul dataAccessLayer

Pachetul dataAccessLayer conține clasa care face legătura cu baza de date.

Clasa **ConnectionFactory** are atributele private necesare pentru a realiza conexiunea cu baza de date, cum ar fi URL-ul, numele bazei de date, USER-ul și parola. Metoda createConnection încearcă să creeze conexiunea, iar getConnection returnează conexiunea. Avem 4 metode close, fiecare cu parametrii diferiți, necesare pentru a încheia conexiunea cu baza de date.

4.2 Pachetul model

Pachetul model conține clasele model, fiecare tabel din baza de date fiind reprezentat printr-o astfel de clasă.

Clasa **Client** corespunde tabelului clients al bazei de date și are următoarele atribute private: ID, nume, prenume și adresa, acestea fiind accesate și modificate din interiorul altor clase prin intermediul getter-elor și setter-elor. Clasa conține și un constructor cu ajutorul căruia putem să creăm un obiect Client mai ușor și mai rapid.

Clasa **Order** corespunde tabelului orders al bazei de date și are următoarele atribute private: id_order, id_client, id_product și cantit, acestea fiind accesate și modificate din interiorul altor clase prin intermediul getter-elor și setter-elor. Și această clasă conține un constructor cu ajutorul căruia putem să creăm un obiect Order mai ușor și mai rapid.

Clasa **Product** corespunde tabelului products al bazei de date și are următoarele atribute private: ID, nume, cantitate și preș, acestea fiind accesate și modificate din interiorul altor clase prin intermediul getter-elor și setter-elor, la fel ca în clasele de mai sus. La fel ca celelalte clase prezentate mai sus, clasa conține un constructor cu ajutorul căruia putem să creăm un obiect Product mai ușor și mai rapid.

4.3 Pachetul businessLayer

Acest pachet conține logica aplicației, clasele care operează asupra datelor.

Clasa **OperațiiClient** conține metodele care efectuează operații asupra clienților. Aceasta are un atribut privat, o listă de clienți. Metoda findById caută un anumit client în baza de date și îl returnează în cazul în care îl găsește. Metoda insertClient va adăuga în baza de date un nou client, pe baza informațiilor introduse de către utilizator. Metoda deleteClient va șterge din baza de date clientul cu ID-ul introdus. Metoda updateClient va modifica clientul corespunzător în funcție de datele introduse de către utilizator. Metoda showTableClients va crea un tabel JTabel nou în care va introduce toți clienții din baza de date, va adăuga clientul în lista de clienți și va returna tabelul.

Clasa **OperațiiProduct** conține metodele care efectuează operații asupra produselor. Aceasta are un atribut privat, o listă de produse. Metoda findById caută un anumit produs în baza de date și îl returnează în cazul în care îl găsește. Metoda insertProduct va adăuga în baza de date un nou produs, pe baza informațiilor introduse de către utilizator. Metoda deleteProduct va șterge din baza de date produsul cu ID-ul introdus. Metoda updateProduct va modifica produsul corespunzător în funcție de datele introduse de către utilizator. Metoda showTableProducts va crea un tabel JTabel nou în care va introduce toate produsele din baza de date, va adăuga produsul în lista de produse și va returna tabelul.

Clasa **OperațiiOrder** conține metodele care efectuează operații asupra comenzilor. Aceasta are un atribut privat, o listă de comenzi. Metoda findById caută o anumită comandă în baza de date și o returnează în cazul în care o găsește. Metoda insertOrder va adăuga în baza de date o nouă comandă, pe baza informațiilor introduse de către utilizator. Metoda showTableOrders va crea un tabel JTabel nou în care va introduce toate comenzile din baza de date, va adăuga comanda în lista de comenzi și va returna tabelul.

4.4 Pachetul controller

Pachetul controller se ocupă de efectuarea operațiilor cerute prin intermediul interfeței grafice.

Clasa **Controller** conține toți actionListener-ii pentru butoanele din interfața grafică. În constructor am declarat frame-ul și l-am făcut vizibil și am adăugat actionListener pentru fiecare buton, în funcție de proprietățile sale. Primul buton de adăugare adaugă un client în baza de date, în funcție de datele introduse de utilizator. Dacă datele sunt incorecte, se va afișa pe ecran un mesaj de eroare. La fel și în cazul celorlalte butoane de adăugare, pentru produse și pentru comenzi. Avem 2 butoane de delete, unul pentru clienți și altul pentru produse, care vor șterge din baza de date clientul sau produsul cu ID-ul corespunzător, introdus de către utilizator. Din nou, în cazul în care ID-ul introdus este greșit sau nu există, se va afișa pe ecran un mesaj de eroare. Există 2 butoane update, care vor actualiza datele clientului sau produsului corespunzător. În caz de eroare, se afișează un mesaj. Butoanele de refresh reactualizează tabelele după ce a avu loc o modificare. ActionListener-ul pentru butonul de adăugare a unei comenzi are niște condiții suplimentare față de restul. O comandă se poate adăuga doar în cazul în care există în baza de date atât clientul cu ID-ul corespunzător, cât și produsul cu ID-ul corespunzător. În caz contrar, se va afișa un mesaj de eroare. Mai mult, cantitatea cerută trebuie să fie mai mică sau egală cu stocul disponibil. Dacă această condiție nu este îndeplinită, se va afișa pe ecran un mesaj corespunzător. Dacă comanda este validă și se adaugă în baza de date, se va afișa pe ecran o chitanță, care conține prețul final al comenzii.

4.5 Pachetul presentation

Acest pachet conține toate clasele de interfață grafică, care extind JFrame.

Clasa **Frame** conține frame-ul principal, care va apărea la rularea programului. Aici am creat un TabbedPane, care va conține câte un Panel pentru fiecare din următoarele: clienți, produse și comenzi. Fiecare Panel conține butoanele, text field-urile și tabelele corespunzătoare.

Clasa **ExceptionFr** reprezintă frame-ul care va apărea pe ecran în cazul în care datele introduse în câmpuri nu sunt corecte.

Clasa **Stock** reprezintă frame-ul care va apărea pe ecran în momentul în care, într-o comandă, se va cere o cantitate de produs mai mare decât cea disponibilă, lucru care face imposibilă plasarea comenzii.

Clasa **Receipt** conține frame-ul care apare atunci când comanda s-a plasat. Acesta afișează prețul final, reprezentând, de fapt, o chitanță.

4.6 Pachetul main

Acesta conține **clasa Main**, unde am creat conexiunea cu baza de date și am instanțiat controller-ul.

5. Design

Pentru a ușura înțelegerea și munca utilizatorului, interfața grafică este simplă, dar concisă și la subiect. Utilizatorul va introduce datele dorite și va alege operația pe care vrea să o efectueze. Are la dispoziție 3 Panel-uri, corespunzătoare fiecărui tabel al bazei de date.

6. Concluzie

În concluzie, această aplicație poate fi folosită pentru buna gestiune a comenzilor unui depozit. Aceasta adaugă, șterge sau modifică date în baza de date, simplu și rapid, prin intermediul interfeței grafice.

Există totuși numeroase posibilități de dezvoltare ulterioară. Punctul slab al acestei aplicații e reprezentat de faptul că tabelele nu se actualizează în interfața grafică în timp real, exact în momentul modificării datelor. Acest lucru îngreunează puțin urmărirea în timp real a datelor. Butonul de refresh ar putea sa fie unul singur, pentru întregul frame, nu unul pentru fiecare Panel, iar acesta să actualizeze datele în timp real. O altă posibilitate de dezvoltare ulterioară este adăugarea mai multor operații sau tabele. O operație folositoare ar fi aceea de afișare a întregului preț pentru un client. În momentul de față, programul afișează prețul pentru o singură comandă, adică pentru un singur produs. Aplicația s-ar putea dezvolta astfel încât clientul să poată comanda mai multe produse deodată și să se afișeze prețul final, al întregii comenzi. O altă operație folositoare care ar putea fi implementată este anularea unei comenzi. În cazul în care clientul se răzgândește și dorește să anuleze comanda, cantitatea de produse să revină din nou în stoc, adică să nu se piardă.

7. Bibliografie

[1] Java Database Connectivity - Wikipedia

[2] Bază de date - Wikipedia

[3] Open Database Connectivity - Wikipedia

[4] Establishing JDBC Connection in Java - GeeksforGeeks

[5] JDBC - Statements, PreparedStatement and CallableStatement - Tutorialspoint