

**DOCUMENTATIE TEMA 3**

**Kasler Madalina**

**Grupa 30221**

**Profesor Laborator: Dorin Moldovan**

Luați în considerare o aplicație OrderManagement pentru procesarea comenzilor client pentru un depozit. Bazele de date relaționale sunt utilizate pentru a stoca produsele, clienții și comenzile. În plus, aplicația ar trebui să fie structurată în pachete utilizând o arhitectură stratificată prezentată în prezentarea suportului (ASSIGNMENT\_3\_SUPPORT\_PRESENTATION.pdf) și ar trebui să utilizeze (minim) următoarele clase:

Clasele de modele -reprezintă modelele de date ale aplicației

Clasele de logică a afacerilor-conțin logica aplicației Clase de prezentare - clase legate de GUI Clase de acces la date Clase care conțin accesul la baza de date.

Cerinte

• Proiectare orientată spre obiecte, clase cu maximum 300 de linii, metode cu maximum 30 de linii

• Utilizați javadocpentru clase de documentare și generați fișierele JavaDoc corespunzătoare.

• Utilizați baze de date relaționale pentru stocarea datelor pentru aplicație, minim trei tabele: Client, Produs și Comandă

* Creați o interfață grafică cu utilizatorul care include:

O fereastră pentru operațiunile clientului: adăugați un client nou, editați clientul, ștergeți clientul, vizualizați toți clienții dintr-un tabel (JTable) oO fereastră pentru operațiile produsului: adăugați un produs nou, editați produsul, ștergeți produsul, vizualizați toate produsele dintr-un tabel (JTable). O fereastră pentru crearea comenzilor de produs - utilizatorul va putea selecta un produs existent, selecta un client existent și introduce o cantitate dorită pentru ca produsul să creeze o comandă validă. În cazul în care nu există suficiente produse, va fi afișat un mesaj sub stoc. După finalizarea comenzii, stocul produsului este decrementat.

Utilizați tehnici de reflecție pentru a crea o metodă care primește o listă de obiecte și generează antetul tabelului prin extragerea prin reflexie a proprietăților obiectului și apoi populează tabelul cu valorile elemente din listă  Documentație de bună calitate Arhitectură stratificată (aplicația va conține cel puțin patru pachete: dataAccessLayer, businessLayer, model și prezentare)

Creați o factură pentru fiecare comandă ca fișier text sau fișier .pdf.

Utilizați tehnica de reflecție pentru a crea o clasă generică care conține metodele de accesare a DB: creați obiect, editați obiect, ștergeți obiect și găsiți obiect.

Interogările pentru accesarea DB pentru un anumit obiect care corespunde unui tabel vor fi generate dinamic prin reflecție.

2. Obiectivul temei

2.1 Obiectivele principale

Ca si obiectiv principal al acestei teme, a fost creearea unei aplicatii care va pune la dispozitia un sisteme de management al unui deposit. Aceasta aplicatie o sa permita utilizatorului sa introducă clienți, produse, dar să facă și comenzi. Daca comanda a fost validă, se va putea genera și o factură pentru aceasta, cu numele clientului, produsele, cantitatea si costul. Daca comanda nu va fi valabila, se va afisa o eroare pentru a instiinta clientul că stocul de produse nu este suficient.

2.2 Obiectivele secundare

1. JDBC

Un driver JDBC este o componentă software care permite unei aplicații Java să interacționeze cu o bază de date . Drivere JDBC sunt similare cu drivere ODBC , furnizori de date ADO.NET , și furnizorii de OLE DB . Pentru a vă conecta la baze de date individuale, JDBC (Java Database Connectivity API ) necesită drivere pentru fiecare bază de date. Driverul JDBC oferă conexiunea la baza de date și implementează protocolul pentru transferul interogării și rezultatului între client și bază de date. Driverele de tehnologie JDBC se încadrează în una din cele patru categorii. Podul JDBC-ODBC Driver nativ-API Driver de rețea-protocol ( driver de middleware ) Driver de bază de date-protocol (driver Java pur) sau driver subțire. Driver JDBC

Driverul JDBC tip 1, cunoscut și sub denumirea de podul JDBC-ODBC , este o implementare a driverului bazei de date care folosește driverul ODBC pentru a se conecta la baza de date. Driverul convertește apelurile metodei JDBC în apeluri funcționale ODBC. Șoferul este deoarece se folosește de ODBC , care , la rândul său depinde de bibliotecile native ale subiacent dependente de platforma sistemului de operare JVM rulează pe. De asemenea, utilizarea acestui driver duce la alte dependențe de instalare; de exemplu, ODBC trebuie instalat pe computerul cu driver și baza de date trebuie să accepte un driver ODBC. Utilizarea acestui driver este descurajată dacă este disponibilă alternativa unui driver Java pur. Cealaltă implicație este că orice aplicație care utilizează un driver de tip 1 nu este portabilă, având în vedere legătura dintre driver și platformă. Această tehnologie nu este potrivită pentru un mediu cu tranzacții ridicate. De asemenea, driverele de tip 1 nu acceptă setul complet de comenzi Java și sunt limitate de funcționalitatea driverului ODBC. Sun (acum Oracle) a furnizat un driver JDBC-ODBC Bridge: sun.jdbc.odbc.JdbcOdbcDriver . Acest driver este cod nativ și nu Java și este sursă închisă. Sun's / Oracle's JDBC-ODBC Bridge a fost eliminat în Java 8 (alți furnizori sunt disponibili). Dacă un driver a fost scris astfel încât încărcarea să provoace crearea unei instanțe și, de asemenea, apelurile DriverManager.registerDriver cu instanța respectivă ca parametru, atunci acesta se află în lista driverelor DriverManager și este disponibil pentru crearea unei conexiuni. Uneori se poate întâmpla ca mai mult de un driver JDBC să se poată conecta la o anumită adresă URL. De exemplu, atunci când vă conectați la o anumită bază de date la distanță, ar putea fi posibil să utilizați un driver bridge JDBC-ODBC, un driver JDBC-to-generic-protocol-network sau un driver furnizat de furnizorul bazei de date. În astfel de cazuri, ordinea în care sunt testați driverele este semnificativă, deoarece DriverManager va folosi primul driver pe care îl găsește, care se poate conecta cu succes la adresa URL dată. Mai întâi DriverManager încearcă să utilizeze fiecare driver în ordinea în care a fost înregistrat. (Driverele enumerate în jdbc.drivers sunt întotdeauna înregistrate mai întâi.) Acesta va sări peste orice driver care nu are cod de încredere decât dacă a fost încărcat din aceeași sursă ca și codul care încearcă să deschidă conexiunea. Testează driverele apelând la rândul lor metoda Driver.connect, trecându-le adresa URL pe care utilizatorul a transmis-o inițial metodei DriverManager.getConnection . Primul driver care recunoaște adresa URL face conexiunea. Driver JDBC .

2. MYSQL

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 lânga 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.

Alte tipuri de baze de date sunt modelul ierarhic, modelul orientat pe obiecte și, mai nou, modelul XML.

Există multe tipuri de baze de date. Cea mai bună baze de date pentru o anumită organizație depinde de modul în care organizația intenționează să o utilizeze.

Baze de date relaționale. Bazele de date relaționale au avut cea mai mare pondere în anii 1980. Elementele dintr-o bază de date relațională sunt organizate ca un set te tabele, cu rânduri și coloane. Tehnologia bazelor de date relaționale oferă cel mai eficient și flexibil mod de a accesa informațiile structurate.

Baze de date orientate pe obiecte Informațiile dintr-o bază de date orientată pe obiecte sunt reprezentate sub formă de obiecte, așa cum se întâmplă în programarea orientată pe obiecte.

Baze de date distribuite O bază de date distribuită constă în două sau mai multe fișiere aflate în site-uri diferite. Baza de date poate fi stocată pe mai multe computere aflate în aceeași locație fizică sau aflate în diferite rețele.

Depozite de date. Un depozit central de date, reprezentând un tip de baze de date creat special pentru interogări și analize rapide.

Baze de date NoSQL. O bază de date NoSQL sau o bază de date nerelațională, permite stocarea și gestionarea datelor nestructurate sau semi-structurate (spre deosebire de o bază de date relațională, care definește modul în care trebuie compuse toate datele introduse). Bazele de date NoSQL au fost populare ca aplicații web, devenind din ce în ce mai uzuale și mai complexe.

Baze de date grafice. O bază de date grafice stochează datele sub forma unor entități și a relațiilor dintre acestea.

Baze de date OLTP O bază de date OLTP este o bază de date rapidă, pentru analize, destinată rulării unui număr mare de tranzacții efectuate de mai mulți utilizatori.

Reflectie

Acest mecanism ne este oferit de catre pachetul java.lang.reflect. Reflectia inseamna autoexaminare, adica permite determinarea structurii clasei. In limitele managerului de securitate putem afla metodele clasei, constructorii clasei si restul membrilor clasei. Cateodata putem sa modificam starea obiectlui prin apelul metodelor specifice sau putem construi obiecte noi . Mecanismul de reflectie este utilizat de componentele Java (Java beans) pentru determinarea capabilitatilor obiectelor pe timpul executiei. Membrii clasei sunt atributele si metodele. Metodele se impart la randul lor in doua categorii, constructori si metode obisnuite.

3. Proiectare

Pentru ca implementarea acestei aplicatii sa fie mult mai simplă, am ales să imi structurez proiectul in mai multe pachete: BusinessLogic, DAO, DataAccess, Model, Presentation. In DataAccess se gasesc cele 2 clase Controller si ConnectionFactory care fac 2 lucruri diferite. Prima asigura conexiunea dintre interfata grafica si datele algorimul simulatorului, iar cea de a doua asigura conexiunea cu baza de date. In BusinessLogic se gasesc implementarile efective ale algoritmilor din spatele programului. In pachetul DAO se regasesc claselele AbstractDao, ProductDao, BillDao, OrderDao și ClientDao. AbstractDao conține metodele care definesc operațiile comune de pentru accesarea tabelelor din baza de date(Interogare, Ștergere, Modificare). Aceste operații sunt definite pe un tip genric <T>, unde T poate fi orice clasa din pachetul MODEL, care reprezint tabelele din baza de date, având exact același nume și aceleași variabile instanță, dar și aceleași tipuri de date precum câmpurile tabelelor. Celelate 4 clase din acest pachet moștenesc clasa AbstractDao, deci implicit metodele acesteia. În fiecare dintre celelalte clase există câte o metodă particulară pentru fiecare dintre cele 4 tabele corespunzătoare din baza de date relațională, și anume: insert. Fiecare dintre aceste 4 metode reprezintă oprația de adăugare a unei noi înregistrări în fiecare dintre cele 4 tabele: Client, Comanda, Produs și Factura.

4.Implementare

Clasa ConnectionFactory – aceasta clasa este cea care face posibila conexiunea cu baza de date, avand doar metode care se ocupa de realizarea acestui lucru.

Clasa AbstractDao—aceasta clasa face parte din pachetul DAO si in aceasta sunt definite prin intermediul unor metode majoritatea operațiilor comune care permit accesul la baza de date. Aceste metode sunt definite pe tipul generic <T> care poate sa fie orice obiect dintre urmatoarele: Client, Order, Product si Bill. Metodele createSelectQuery, deleteQuery și updateQuery contribuie la construirea comenzii în limbaj sql, în funcție de parametrul rimit(de obicei condiția de pe clauza WHERE). O altă metodă importantă a acestei clase este metoda createObjects care preia rezultatul unei interogări din tabel și creează un obiect de tipul respectiv cu câmpurile obțiunute în urma interogării executate.

Clasele ClientDao, OrderDao, BillDao si ProductDao sunt toate clasele care moșteneșc clasa AbstractDao. Toate aceste obiecte din clase vor beneficia de toate metodele implementate in AbstractDao prin clauza mostenirii. In fiecare clasa va mai fi implementata metoda de insert pentru fiecare tip de obiect.

Clasele ViewClient, ViewOrder si ViewProudct sunt clasele de tip View care se vor ocupa de cele 3 ferestre care se vor deschide odata cu aplicația, pentru a se realiza interfața grafică.

Clasa Controller este clasa care se va ocupa de conexiunea dintre interfata utilizatorului si algoritmul din spate.

5.Concluzii

În concluzie, realizarea acestui proiect a fost foarte utila, pentru ca am reamintit o mare parte din conceptele invatate in semestrele anterioare, atat la Programare Orientata pe Obiect cat și la Baze de Date.

O dezvoltare ulterioara ar fi o interfața grafică mai usor de folosit, poate un buton de search pentru un anumit produs ca sa ii fie mai usor clientului sa aleaga ce tip de produse vrea sa caute, poate impartire pe cateogrii.