Documentație Tema 3

Order Management

Student: Iurieț Denis - Ionuț

Grupa: 30223

Profesor îndrumător: Dorin Moldovan

Table of Contents

[1.Obiectivul Temei 3](#_Toc8214957)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc8214958)

[3.Proiectare 3](#_Toc8214959)

[4.Implementare 6](#_Toc8214960)

[5. Concluzii 7](#_Toc8214961)

[6.Bibliografie 7](#_Toc8214962)

# 1.Obiectivul Temei

Obiectivul acestui proiect este de a modela și implementa o aplicație care realizează operațiile de bază pentru managementul unor comenzi în cadrului unui depozit. Aplicația are ca funcționalități adăugarea, editarea, ștergerea și vizualizarea clienților, a produselor și a comenzilor.

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

Aplicația are ca și scop plasarea unor comenzi realizate de un user. Procesarea datelor și a comenzilor se realizează cu ajutorul unei baze de date în care sunt stocate toate informațiile.

Pentru modelarea aplicației s-a utilizat baza de date MySQL. MySQL este un [sistem de gestiune a bazelor de date](https://ro.wikipedia.org/wiki/Sistem_de_gestiune_a_bazelor_de_date) [relaționale](https://ro.wikipedia.org/w/index.php?title=Baze_de_date_rela%C8%9Bionale&action=edit&redlink=1), produs de compania suedeza [MySQL AB](https://ro.wikipedia.org/wiki/MySQL_AB) și distribuit sub [Licența Publică Generală GNU](https://ro.wikipedia.org/wiki/Licen%C8%9Ba_Public%C4%83_General%C4%83_GNU). Este cel mai popular [SGBD](https://ro.wikipedia.org/wiki/SGBD) [open-source](https://ro.wikipedia.org/wiki/Open-source) la ora actuală, fiind o componentă cheie a stivei [LAMP](https://ro.wikipedia.org/wiki/LAMP) ([Linux](https://ro.wikipedia.org/wiki/Linux), [Apache](https://ro.wikipedia.org/wiki/Apache), MySQL, [PHP](https://ro.wikipedia.org/wiki/PHP)).

Reflection este un API care este utilizat pentru a examina si modifica comportamentul metodelor, claselor, interfetelor la runtime. Aplicatia utilizeaza tehnica de reflection pentru o modelare eficienta si polimorfa a metodelor de adaugare, stergere si actualizare in baza de date (pentru crearea si executarea interogarilor in baza de date), cat si pentru crearea tabelelor specifice clienilor si produselor.

Modul de utilizare a aplicatiei:

1. Se selectează panoul de gestiune a unuia dintre cele trei tabele.
2. Se alege operația ce se dorește a fi realizată.
3. Pentru toate cele trei tabele, se pot efectua aceleași operații și anume: adăugare, ștergere, editare și vizualizare.
4. În funcție de operația dorită, se va cere completarea unor câmpuri specifice, acțiune urmată de apăsarea butonului “Enter” pentru confirmare.

# 3.Proiectare

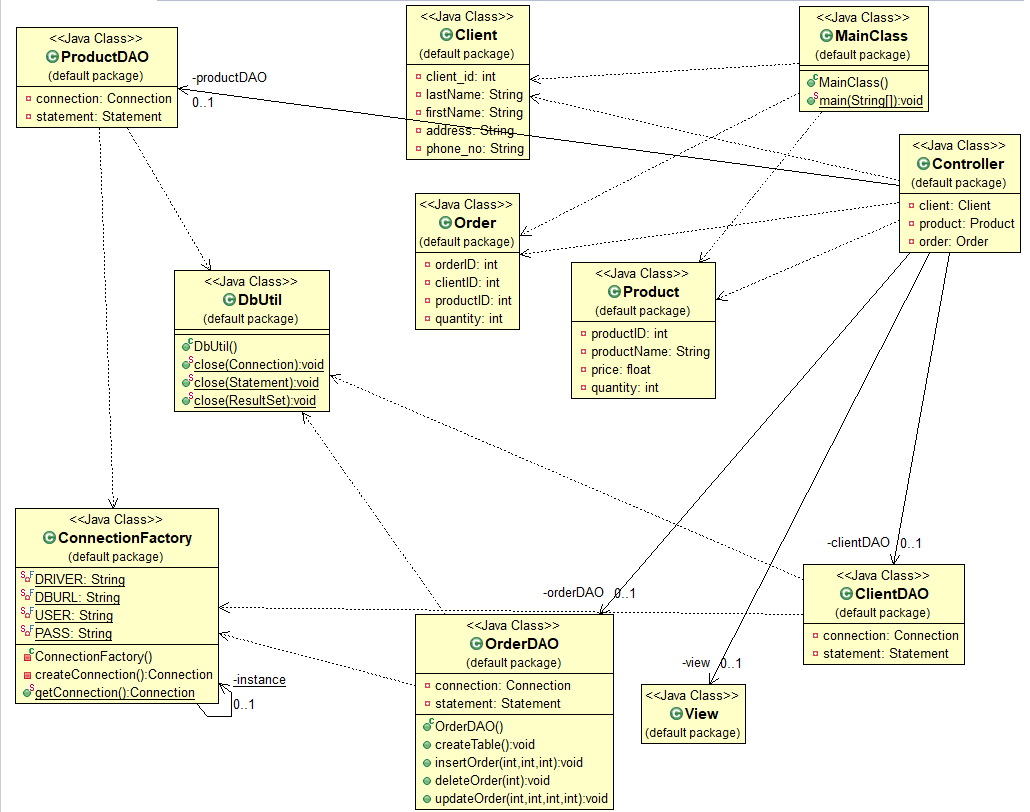
Proiectul este structurat în pachete, conținând mai multe clase, respectându-se modelul de proiectare 3-Tire Architecure. Așadar, proiectul conține un pachet pentru fiecare dintre cei 3 layer-e: Presentation, Business Layer și Data Layer. De asemenea, pentru realizarea conexiunii la baza de date există un pachet pentru conexiune și un pachet care conține clasele reprezentative pentru tabelele stocate în baza de date.

Pe plan intern, pentru a implementa aplicația, am hotărât ca un client să aibă un nume (nume de familie și prenume), un număr de telefon și o adresă, pe lângă id-ul de identificare a acestuia care este unic, în cazul produselor am hotărât ca fiind potrivit să aibă un nume, un pret si un camp numit quantity care se referă la numarul produselor de acel tip de pe stoc.

Pentru a efectua legaturile logice dintre clienti si produse am avut nevoie de încă o tabelă pentru gestiunea comenzilor si anume: orders ce conține un id pentru comandă, un id client, id produs și cantitatea ce se comandă.

**Diagrama UML**

UML este notaţia internaţională standard pentru analizaşi proiectarea orientatăpe obiecte. Diagramele UML de clase sunt folosite in modelarea orientata obiect pentru a descrie structura statica a sistemului, modului in care este el structurat. Ofera o notatie grafica pentru reprezentarea: claselor -entitati ce au caracteristici comune relatiilor - relatiile dintre doua sau mai multe clase , Reprezentarea UML a claselor.



**Proiectare Clase**

**Clasele Customer, Product, Orders**

S s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s

Aceste clase au fost create pentru a fi echivalentul tabelelor din baza de date. Fiecare camp din tabel va reprezenta o varibila in clasa. Este de mentionat ca aceste variabile vor avea exact acelasi nume ca si denumirea coloanelor din tabele si la fel si clasa se va numi ca si tabela. Aceste clase nu au nimic deosebit, doar gett - ere si sett – ere pentru variabilele din acestea si un constructor folosit cand se va instantia un obiect de aceasta clasa.

**Clasa ConnectionFactory**

Aceasta clasa este folosita pentru a obtine conexiunea cu baza de date. Metoda getConnection ( ) returneaza conexiunea. Tot in aceasta clasa gasim si metodele necesare pentru a inchide conexiunea, statementul folosit pentru query- uri si rezultatul obtinut in urma query- ului. Aceste metode au acelasi nume, doar parametrii transmisi diferind ( Connection, Statement, ResultSet ). Are ca si campuri 3 Stringuri reprezentand datele specifice conectarii la baza de date si un camp instance care este instantiat la clasa curenta. Pentru securizarea conectiunii se utilizeaza un contructor privat, asadar pentru realizarea conexiunii se va utliza un factory method. Clasa contine metode pentru conectarea la Driver, pentru inchiderea bazei de date, pentru inchidrea unui obiect de tip Statement cat si pentru inchiderea unui obiect de tip ResultSet

**Clasa ClientDAO**

In aceasta clasa vom realiza operatii precum adaugarea unui nou client in baza de date, editarea unui anumit client din baza de date si chiar stergerea acestuia.

Se considera in cazul editarii si stergerii ca se va alege un client existent din baza de date, in cazul in care nu se intampla acest lucru, nu se va edita / sterge nimic din baza de date.

Totodata in aceasta clasa se gaseste si metoda formClient() ce extrage informatiile referitoare la tabela clients din baza de date ordermanagement.

**Clasa ProductDAO**

În aceasta clasa vom realiza operatii precum adaugarea unui nou produs in baza de date, editarea unui anumit produs din baza de date si chiar stergerea acestuia. Se considera in cazul editarii si stergerii ca se va alege un produs existent din baza de date, in cazul in care nu se intampla acest lucru, nu se va edita / sterge nimic din baza de date.

Totodata in aceasta clasa se gaseste si metoda formProduct() ce extrage informatiile referitoare la tabela product din baza de date ordermanagement.

**Clasa View**

Contine toate componentele utilizate pentru realizarea interfetei grafice cu utilizatorul. Aplicatie are o fereastra principala si alte 3 ferestre auxiliare pentru realizarea operatiilor specifice celor trei categorii de actiuni. Ferestrele contin diferite componente care ajuta la introducerea datelor si totodata fereastra pentru clienti si produse contin un tabel care ilustreaza continutul bazei de date.

S s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s ss s s s s s s s s s s ss s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s

**Clasa Controller**

Reprezinta mediatorul dintre clasa View si clasele din BussinesLayer. Clasele din pachetul model sunt Clients, Product, Orders si Components care reprezinta tabelele din DB . Fiecare clasa care un corespondent in BusinessLayer in care se mediaza si valideaza datele din UI si BD.

Fiind o aplicatie Java se respecta principiile programarii orientate pe obiect. Asadar s-a folosit incapsularea prin utilizarea modificatorilor private asupra campurilor clasei. De asemenea se foloseste principiul polimorfismului, iar abstractizarea este cheia de modelare a aplicatiei. Concret, pentru realizarea aplicatiei am ales doar acele caracteristi potrivite( ex: numele produsului, pret, etc. ).

# 4.Implementare

**Clasa Clients** are ca si campuri numele, adresa, numarul de telefon de tip String si un camp pentru id de tip int si un camp budget de tip float. Campul de tip budget s-a adaugat pentru o viitoare extindere. Toate campurile sunt de tip privat si au metode de getter si setter. De asemnea metoda toString a fost suprascrisa. Metoda toObject returneaza un sir de obiecte de tip Object continand valorile unui client. Clasa are un constructor vid si unul cu parametrii.

**Clasa Product** are ca si campuri numele tip String si un camp pentru id si unul pentru cantitate de tip int si un camp price de tip float. Toate campurile sunt de tip privat si au metode de getter si setter. De asemnea metoda toString a fost suprascrisa. Metoda toObject returneaza un sir de obiecte de tip Object continand valorile unui produs. Clasa are un constructor vid si unul cu parametrii.

**Clasa Orders** are un camp pentru id, unul pentru id client, unul pentru id-ul produsului si unul pentru numarul de produse din comanda de tip int, un Toate campurile sunt de tip privat si au metode de getter si setter. Clasa are un constructor vid si unul cu parametrii.

**Clasa Controller** are ca si campurile de tipul claselor din pachetul businessLayer, un camp de tipul View, doua Stringuri care vor contine informatiile pentru factura si un camp de tipul Order. Constructorul clasei are rolul de a adauga ActionListner-i pentru fiecare buton. De asemena clasa va contine toate clasele care implemeneaza metodele pentru ActionListenerii de mai sus. Astfel clasa contine o subclasa pentru fiecare creare de ferestra de client, de produs, de order. In clasele pentru inserarea unui client/produs se extrag valorile din textField-rui si se creaza un obiect de tipul dorit. Mai apoi se apeleaza metoda de adaugare din businessLayerul corespunzator si se modifica continutul Ui-ului cu ajutorul DefaultTabelModel. In metoda actionPerformed din clasele de ediatare a unui client/produs sunt implementate doua modalitati de update. Cand se cunoaste id-ul sau cand se selecteaza un client/produs in tabel. La selectarea din tabel a unei linii se vor completa automat textfield-urile aferente. Se apeleaza metoda de update din ClientDAO sau ProductDAO si apoi utilizanduse DefaultTabelModel se actualizeaza si UI-ul. In clasa pentru butonul de adaugarea unui produs in cos se insereaza in baza de date in tabelul de components legatura dintre id-uri si cantitatea. De asemenea se adauga in stringurile pentru factura produsul si informatiile despre client. Tot aici se actualizeaza valoarea cantitatii produsului si se calculeaza valoarea comenzii. Daca qantitatea de produs este prea mare se genereaza un mesaj de Under-stock. Adaugarea comenzii se realizeaza in clasa corespunzatoare actionListener-ului pentru butonul de FinishOrder si de asemenea acolo se genereaza si factura. Stergerea unui client/produs poate implica si stegerea unei inregistrari din tabela complement asadar, se verifica existenta unei astfel de legaturi si in caz afiramtiv mai intai se sterge acea inregistrare si ulterior clientul/produsul. Generarea raportului v-a determina apelarea metodelor pentru calcularea statisticilor si afisarea lor in text area.

# 5. Concluzii

Aplicatia este un sistem complex de gestiune a unor comezi, clienti si produse. Utilizarea unei baze de date relationale usureaza managementul lor si contureaza o structura usor de descifrat.

Aplicatia poate avea o multituidine de dezvoltari viitoare. Pentru inceput, s-ar putea retrage bani din contul clientului care realizează comandă, s-ar putea adauga o zona de login pentru angajatii care utilizeaza aceasta aplicatie.

Tema a constituit un bun mod de aprofundare a utilizarii bazei de date si manipularea ei cu ajutorul limbajului de pogramare Java. Am invatat sa folosesc Java Reflection si cat de utilă poate fi pentru manipularea unor obiecte de tipuri diferite.

# 6.Bibliografie

<https://stackoverflow.com/>

<https://www.geeksforgeeks.org/>