DOCUMENTATIE TEMA 3

WAREHOUSE

Student: Moldovan Teodora
Grupa: 30225
Facultatea de Automatica si Calculatoare
Profesor Laborator : Antal Marcel

Cuprins

1. Obiective

1.1. Obiectiv Principal

Aceasta tema presupune crearea unei aplicatii pentru procesarea comenziilor venite de la clienti pentru un depozit. Pentru a stoca produsele, clietii si comenziile se vor folsi baze de date relationale. Aplicatia trebuie sa foloseasca (minim) clase pentru:

- Model: reprezinta modelele de date pentru aplicatie
- Business Logic: contin logica aplicatiei
- Prezentare: contin interfata grafica cu utilizatorul
- Data access: clase care contin legatura cu baza de date

1.2. Obective Secundare

Obiectiv Secundar	Descriere	Capitol
Analiza problemei si presupuneri	Se analizeaza modul de abordare a problemei propuse.	2
Alegerea structurilor de date	Prezentarea structurii de date alese si a argumentelor	3
Impartirea pe clase	Diagrama de clase si descriere a functionalitatii claselor	3,4
Dezvoltarea algoritmilor	Propunerea unor metode de implementare a algoritmilor pentru operatiile propuse	3
Implementarea solutiei	Detalii referitoare la implementare in urma analizei realizate	4
Testare	Testarea functionarii corecte a aplicatiei si modurile de realizare a acesteia	6

2. Analiza Problemei

Datorita cresterii in dificultate a organizarii si intretinerii unui depozit, apare necesitatea utilizarii unui sistem de management pentru a mentine evidenta clientiilor, a comenziilor si a produselor. Acest proiect are ca scop realizarea unui astfel de sistem prin intermediul unei aplicatii. Aplicatia trebuie sa permita utilizatorului sa efectueaza operatii asupra clientiilor, produselor si comenziilor. Managerul depozitului trebuie sa poata adauga si sterge clienti sau produse, sa modifice datele deja existente despre un client sau un produs si sa introduca o comanda noua. Pentru a facilita vizualizarea clientiilor, produselor si comenziilor deja existente, datele necesare vor fi vizibile in interfata grafica sub forma unor tabele. La

adaugarea unei noi comenzi, in cazul in care aceasta este valida si poate fi indeplinita, se va genera un bon pentru comanda respectiva.

3. Proiectare

3.1. Structuri de date

Una dintre primele probleme aparute in faza de proiectare presupune alegerea unor structuri de date corespunzatoare modului de lucru cu clasele si stocarii anumitor valori pe perioada efectuarii operatiilor dorite.

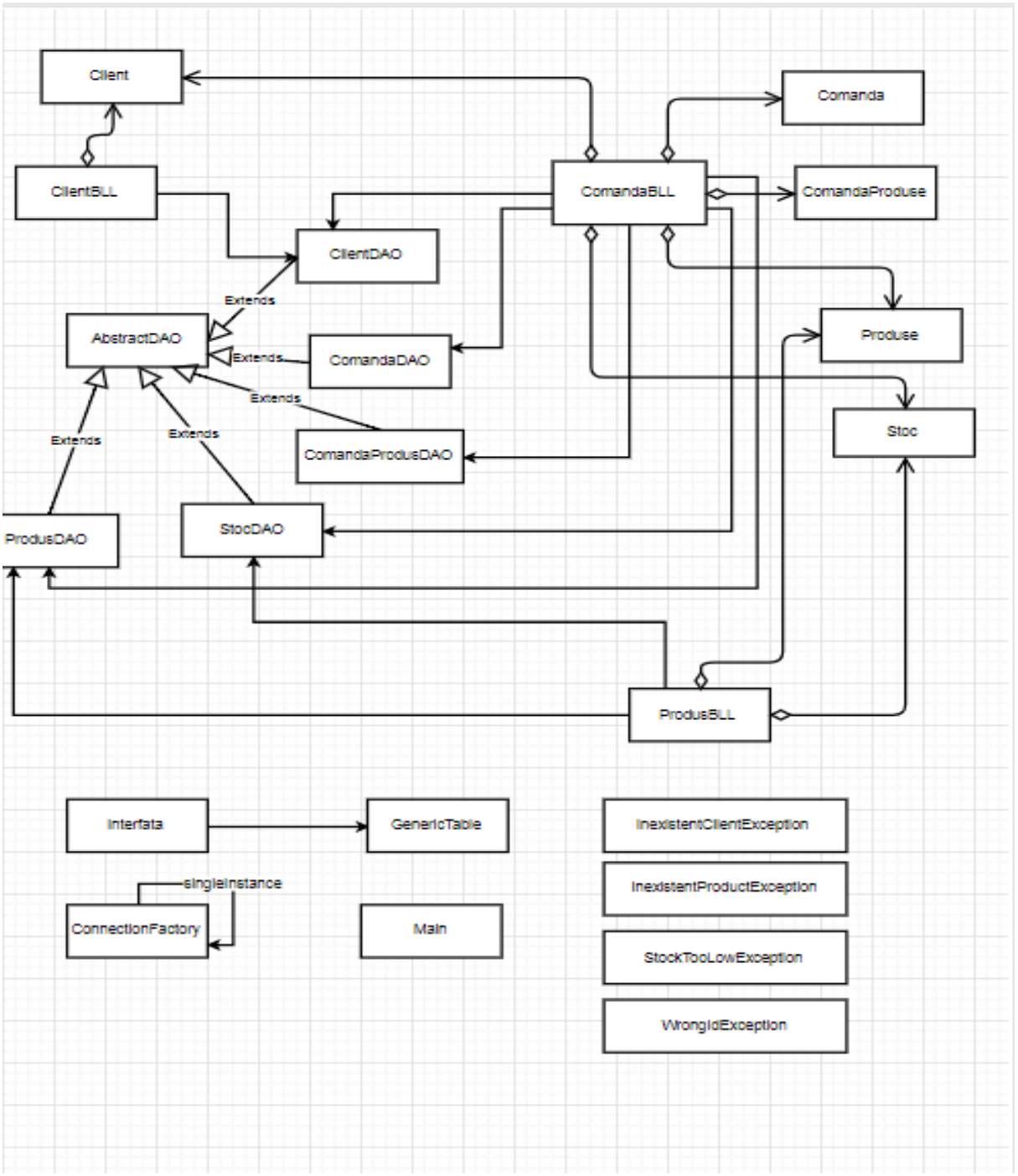
Pentru aceasta tema se folosesc bazele de date relationale pentru stocarea tabelelor si a informatiilor necesare aplicatiei. Se considera cazul minimal in care se tine cont de clienti, produse si comenzi.

Baza de date contine urmatoarele tabele:

- Client (id, nume, prenume, adresa)
- Produs (id, nume, pret)
- Stoc (id_produs, cantitate)
- Comanda (id, id_client)
- ComandaProdus(id_comanda, id_produs, cantitate)

Din motive legate de implementare si pastrarea integritatii datelor in baza de date, s-a decis ca tabela Produs sa nu contina si stocul, ci sa se creeze o tabela separata pentru stoc, in care id_produs este cheie straina si face legatura cu un singur produs existent, iar tabela ComandaProdus este legatura dinte tabela Comanda, care contine informatii despre clientul care a efectuat comanda, si tabela Produse . Atributul cantitate din aceasta tabela reprezinta cantitatea pe care clientul o comanda din respectivul produs.

3.2. Diagrama de clase



3.3. Algoritmi

Operatiile pe care utilizatorul trebuie sa le poata efectua asupra bazei da date sunt adaugare, stergere si editare pentru clienti si produse, adaugarea unei comenzi noi si vizualizarea clientiilor, produselor si comenziilor existente. Pentru a putea implementa aceste operatii este necesara intelegerea modului in care se face legatura cu baza de date si a mecanismului de reflectie de care dispune Java.

Mecanismul de reflectie 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. Principalele clase care definite in acest pachet sunt:

- java.lang.reflect.Field
- java.lang.reflect.Method
- java.lang.reflect.Constructor

Clasa Class ne ofere cate o pereche de metode pentru obtinerea atributelor, metodelor respectiv ai contructorilor. Una dintre aceste metode permit accesul la metodele publice, incluzand si cele mostenite, iar cealalalta metode permite accesul doar la metodele publice si nepublice declarate in cadrul clasei. De exemplu getFields() va fi metoda care ne da lista tuturor campurilor de date publice, iar getDeclaredFields() returneaza doar cele declarate in cadrul clasei.

Pentru a putea face legatura cu baza de date este necesara definirea unor obiecte, al caror nume sa fie acelasi cu al tabelelor existente si care sa aiba ca atribute coloanele din tabela respectiva. Acestea vor fi folosite de niste clase cu extensia DAO (Data Access Object). S-a ales sa se creeze o clasa generica care utlizeaza tehnica reflection care contine metodele de crearea a unui obiect, editare si stergere, iar interogariile care sunt transmise bazei de date vor fi generate dinamic tot prin reflectie.

De asemenea, mecanismul de reflectie este folosit si pentru afisarea tabelelor din baza de date.

4. Implementare

Metoda de implementare aleasa contine 7 pachete:

- businessLayer
- dao
- dataAccessLayer
- exception
- GUI
- Main
- Model

4.1 GUI

Pachetul GUI contine doua clase: Interfata si GenericTable.

Clasa GenericTable contine metoda generica createTable, care primeste ca argument un ArrayList de obiecte de tipul T si creeaza un model de tabel (Default Table Model). Metoda ia campurile obiectului transmis si le introducere in headerul tabelului, iar apoi ia valorile campurilor pentru fiecare obiect. Metoda este apelata din clasa Interfata, care creeaza trei astfel de tabele: clientTbl, produsTbl si comandaTbl (toate fiind de tipul JTable).

Clasa Interfata contine aranjarea componentelor in frame la rularea aplicatiei si actionListeneri pentru butoane. Prima fereastra a aplicatiei contine trei butoane : Clients, Products, Orders.

Daca se selecteaza Clients, atunci este afisata o fereastra care contine un JTable cu clientii existenti fiecare avand id, nume, prenume, adresa si patru butoane: Add Client, Update Client, Delete Client si Back.

Daca se selecteaza Add Client, utilizatorul trebuie sa introduca numele, prenumele si adresa noului client si apoi sa apese din nou pe butonul Add Client.

Daca se selecteaza Update Client, utilizatorul trebuie sa introduca id-ul clientului a carui date doreste sa le editeze si sa completeze cel putin unul dintre campurile Last Name, First Name sau Address. Campurile care raman necompletate se vor pastra asa cum există deja in baza de date.

Daca se selecteaza Delete Client, in fereastra care se deschide se va introduce id-ul clientului care se doreste a fi sters si se va apasa din nou Delete Client.

Prin apasarea butonului Back in oricare dintre ferestre se va reveni la fereastra anterioara.

In cazul in care din fereastra initiala s-a selectat Products sau Orders, operatiile se efectueaza similar cu mentiunea ca pentru Orders se poate doar adauga o comanda.

4.2 BusinessLayer

Pachetul BusinessLayer contine 3 clase: ClientBLL, ProdusBLL, ComandaBLL.

Clasele ClientBLL si ProdusBLL contin metodele insert, update si delete, necesare pentru a putea verifica daca datele introduse de utilizator sunt corecte si pentru a crea obiectul care va fi introdus in baza de date, invocand metoda corespunzatoare .

Clasa ComandaBLL contine pe langa metoda de verificare a corectitudinii datelor si o metoda care , la adaugarea unei comenzi, genereaza automat un bon pentru clientul si comanda inserata (in cazul in care aceasta este valida) in format pdf. Fisierul generat contine o lista cu produsele comandate si totalul de plata.

Pe langa verificariile de corectitudine in ceea ce priveste tipul datelor introduse (numeric, String) , in cazul operatiilor care presupun folosirea datelor deja existente in baza de date se verfica si existenta acestora. In oricare caz de eroare ar fi una dintre date, se va afisa un mesaj pentru a instiinta utilizatorul, iar executia este oprita pentru a nu introduce date inconsistente in baza de date.

4.3 DataAccessLayer

Pachetul DataAccessLayer contine o singura clasa ConnectionFactory care dispune de metodele necesare pentru crearea unei conexiuni si inchiderea unei conexiuni, a unui Statement sau a unui ResultSet.

Conexiunea propriu-zisa cu baza de date este plasata intr-un obiect de tipul Singleton, ceea ce limiteaza instantierea unei clase la un obiect.

4.4 dao

Pachetul dao contine 6 clase: AbstractDAO, ClientDAO, ProdusDAO, ComandaDAO, ComandaProdusDAO, StocDAO.

Clasa AbstractDAO este o clasa generica care contine metodele de inserare, cautare, editare si stergere din baza de date. Modul de lucru in aceasta clasa a fost prezentat in sectiunea 3.3 Algoritmi. Interogariile executate asupra bazei de date sunt generate

dinamic prin mecanismul de reflectie, fapt pentru care este necesara existenta cate unei metode pentru fiecare tip de query in parte.

Restul claselor din acest pachet extind clasa AbstractDAO si sunt instantiate cand este nevoie sa se execute operatii asupra bazei de date.

4.5 Model

Pachetul model contine 5 clase: Client, Produs, Comanda, ComandaProdus, Stoc. Fiecare dintre acestea este un model al unei tabele din baza de date. Pentru a le putea folosi prin mecanismul de reflectie, atributele trebuie sa coincida cu coloanele din tabela respectiva.

4.6 Main

Acest pachet contine o singura clasa cu metoda main si realizeaza lansarea in executie a aplicatiei.

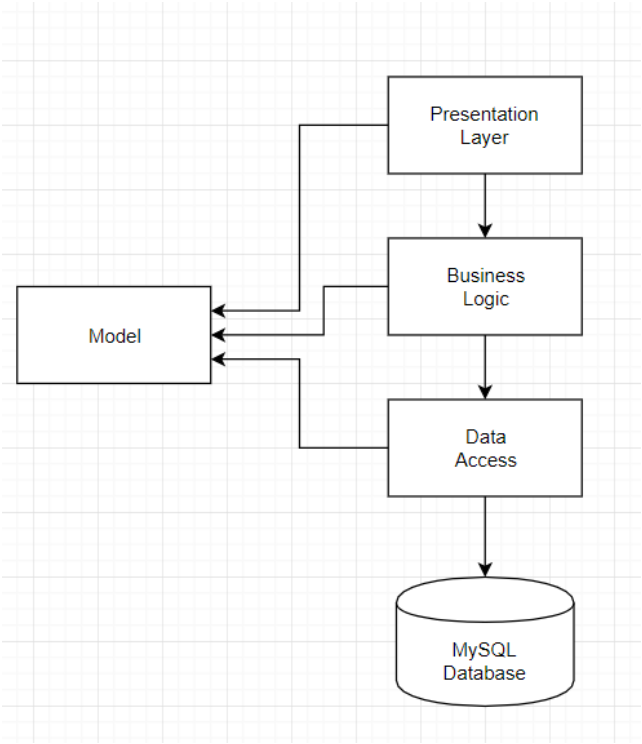
4.7 Exception

Pachetul exception contine 4 clase: InexistentClientException, InexistentProductException, StockTooLowException, WrongIdException. Toate aceste clase extind clasa Exception si sunt folosite pentru a afisa mesaje in cazul in care una dintre datele introduse de utilizator nu este o data valida.

Clasa StockTooLowException este folosita pentru a afisa mesaj de eroare in cazul in care se doreste executarea unei comenzi pentru un produs al carui stoc este mai mic decat cantitatea solicitata.

5. Arhitectura

Implementarea problemei utilizeaza o arhitectura stratificata dupa modelul prezentat in figura urmatoare.



Aplicatia este impartita pe mai multe nivele, fiecare avand un scop diferit si apeland metode ale nivelurilor inferioare.

Nivelul Model contine clasele mapate la tabelele bazei de date.

Nivelul Presentation Layer contine clasele care definesc interfata cu utilizatorul si modul de lucru al acesteia (View + Controller).

Nivelul Business Logic contine clasele care incapsuleaza logica aplicatiei.

Nivelul Data Access contine clasele care realizeaza conxiunea cu baza de date si care contin interogariile necesare pentru a efectua operatiile de inserare, stergere si editare.

Model corepunde pachetului model, presentation layer pachetului GUI, Business Logic pachetului businessLayer, iar DataAccess corespunde pachetelor dataAccessLayer si dao.

6. Testare si Rezultate

Primul pas in testarea corectitudinii aplicatiei este verificarea daca datele de intrare sunt introduse corect de catre utilizator si apoi preluate corect pentru folosire. In cazul in care datele introduse de utilizator nu sunt corecte (nu au formatul numeric sau sir de caractere, in functie de necesitate sau nu există in baza de date), acesta va fi instiitat printr-un mesaj de eroare ca nu a introdus corect datele.

Pentru verficarea ulterioara a corectitudinii operatiilor este necesara vizualizarea in paralel a introducerii datelor si a tabelelor din baza de date. Tabelele care contin clienti, produse si comezi sunt afisate in interfata grafica sub forma unui tabel, iar dupa executarea unei operatii se revine tot timpul in fereasta corespunzatoare tabelei pe care s-a efectuat operatia dorita si se poate observa modificarea realizata.

Totusi, pentru o verificare mai amanuntita, de exemplu in cazul introcerii unei comenzi, trebuie sa se analizeze baza de date din MySQL Workbench. La adaugarea unei comenzi de vor modifica tabelele comanda, comandaprodus si stoc astfel:

- Id-ul comenzii si cel al clientului care a efectuat comanda vor fi introduse in tabela comanda
- Id-ul comenzii, id-ul produsului si cantiatea solicitata vor fi introduse in tabela comandaprodus
- Stocul produsului al carui id s-a introdus va fi decrementat cu o valoare corespunzatoare cantitatii solicitate

Operatie	PASS/FAIL
Adaugare client	PASS
Editare client	PASS
Stergere client	PASS
Adaugare produs	PASS
Editare produs	PASS
Stergere produs	PASS
Adaugare comanda	PASS

7. Concluzii si Dezvoltari Ulterioare

Acest proiect este unul care lasa programatorului o posibilitate crescuta de a interpreta problema si modalitatiile de rezolvare, de aceea din punct de vedere al dezvoltariilor ulterioare există numeroase posibilitati.

Pentru o mai simpla utilizare a aplicatiei si a verificarii corectitudinii la introducerea datelor se pot modifica tabelele care afiseaza datele din baza de date astfel incat sa contina mai multe campuri, nu doar cele ale unei tabele. De exemplu, pentru produse se poate afisa pe langa id, nume si pret, stocul existent la momentul respectiv. Aceasta se poate realiza atat prin intermediul limbajului Java, dar eliminand din generalitatea unor clase, sau prin crearea unor View-uri din baza de date.

Alte posibilitati de dezvoltare posibile sunt impartirea folosirii aplicatiei pe utilizatori diferiti (de exemplu clienti si manageri de depozit), adaugarea mai multor detalii despre produse si clienti sau adaugarea posibilitatii de a adauga mai multe produse diferite intr-o singura comanda.

8. Bibliografie

https://en.wikipedia.org/wiki/Singleton_pattern

http://www.coned.utcluj.ro/~salomie/PT_Lic/4_Lab/HW3_Tema3/Tema3_HW3_Indications.pdf

http://www.ms.sapientia.ro/~manyi/teaching/oop/oop_romanian/curs16/curs16.html

https://www.draw.io/

https://developers.itextpdf.com/examples/itext-action-second-edition/chapter-1

https://mvnrepository.com/artifact/com.itextpdf/itextpdf/5.0.6

https://bitbucket.org/utcn_dsrl/pt-layered-architecture/src

https://bitbucket.org/utcn_dsrl/pt-reflection-example/src