ORDERS MANAGEMENT

*-DOCUMENTATIE TEMA 3-*

NUME: BODE ANDREEA-NICOLETA

GRUPA:30228

PROFESOR: MARCEL ANTAL

Cuprins

[Cuprins . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2](#_Toc100802412)

[1.Obiectivul temei . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2](#_Toc100802413)

[2.Analiza problemei, modelare, scenarii, cazuri de utilizare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3](#_Toc100802414)

[Cazuri de utilizare: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3](#_Toc100802415)

[3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packag) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4](#_Toc100802416)

[Packages . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4](#_Toc100802417)

[4.Implementare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .5](#_Toc100802418)

[Clasa . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .5](#_Toc100802419)

[Java doc . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .6](#_Toc100802420)

[5.Rezultate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6](#_Toc100802421)

[6.Concluzii si dezvoltari ulterioare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7](#_Toc100802422)

[7.Bibliografie . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7](#_Toc100802423)

# 1.Obiectivul temei

Consider an application Orders Management for processing client orders for a warehouse. Relational databases should be used to store the products, the clients, and the orders. The application should be designed according to the layered architecture pattern and should use (minimally) the following classes:

• Model classes - represent the data models of the application

• Business Logic classes - contain the application logic

• Presentation classes – GUI related classes

• Data access classes - classes that contain the access to the database

Acest proiect are ca scop gestionarea bazelor de date prin procesarea mai multor comenzi plasate de mai multi clienti in cadrul unui magazin. In baza de date sunt stocati clientii, produsele, dar si comenzile.

Scopul proiectului este de a intelege cat mai bine cum se gestioneaza o baza de date, utilizand ca aplicatie MySQL Workbench. Programul trebuie sa contina mai multe operatii atat pe tabela client, produs, cat si pe tabela comanda. Pe tabela client se vor putea realiza operatii de adaugare, stergere si actualizare, dar si de vizualizare a intregului tabel. Tabele cu produse va suporta operatii de adaugare, stergere si actualizare, cat si de vizualizare a intregii baza de date. Tabela corespunzatoare comenzilor va avea operatii de adaugare, stergere si de vizualizare a tuturor comenzilor.

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

O bază de date este o colecție organizată de informații sau de date structurate, stocate electronic într-un computer. O bază de date este controlată, de regulă, de un sistem de management al bazelor de date (DBSM). Cumulat, datele, DBMS și aplicațiile asociate reprezintă un sistem de baze de date, denumit prescurtat bază de date.

Datele din cele mai obișnuite tipuri de baze de date sunt distribuite de regulă pe linii și coloane, în diferite tabele, pentru eficientizarea procesării și interogării datelor. Datele pot fi accesate, gestionate, modificate, actualizate, controlate și organizate cu ușurință. Majoritatea bazelor de date utilizează un limbaj structurat de interogare (SQL) pentru scrierea și interogarea datelor.

Pe baza de date se pot face urmatoarele operatii:

* Insert: se adauga un rand nou in tabel cu datele specificate de catre utilizator
* Update : un anumit rand specificat isi va modifica valorile in functie de coloanele care s-au specificat ca se doresc a fi modificate
* Deelete : se va sterge randul specificat din tabel si datele se vor pierde
* Find : se pot cauta diferite date din tabel in functie de o anumita coloana specificata
* Select all: se pot vizualiza toate datele din tabel

# Cazuri de utilizare:

Utilizator va rula proiectul avand aplicatia My Workbench deschisa si cu parola corecta introdusa, astfel programul se va conecta la baza de date, altfel conectarea va esua. Apoi acesta va introduce pe interfata grafica datele pe care se doreste sa se fac operatiunile ( in cazul in care nu va respecta tipul fiecarui field, programul va esua). Daca toate datele sunt introduse corect, spre exemplu email-ul respecta formatul, clientul se afla in intervalul de varste acceptate si face o comanda cu pretul intre cel minim si maxim acceptat programul se va rula cu succes.

# 3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packag)

Diagram, engineering drawing

Description automatically generated

In cele de mai sus este prezenta diagrama UML a proiectului. Unified Modeling Language (prescurtat UML) este un limbaj standard pentru descrierea de modele și specifi-cații pentru software. Diagrama UML este foarte cunoscuta pentru eficiența și claritatea în reprezentarea unor elemente abstracte. Din diagrama prezenta se pot observa dependentele dintre clase.

# Packages

Programul are in totalitate 8 packages care va respecta arhitectura dorita. Atfel incat toate operatiile sa nu le apelam direct, ci prin intermediul claselor din package-ul bll.

Cele 8 packeges sunt: bll ( contine packages-ul validators), connection, dao, model, presentation, start, view. In package-ul bll se gasesc 3 clase: ClientBLL, OrderBLL, ProductBll, prin intermediul acestor clase se apeleaza toate operatiile pe fiecare apel ( insert, update, delete si viewAll), aici se gasesc si package-ul validators care contine clase AgeValidator, CantitateValidator, EmailValidator, Validator, care verifica daca datele introduse sunt corecte.

Pachetul connection contine clasa ConnectionFactory care realizeaza conexiunea la baza de date. Pachetul dao continue 4 clase: AbstractDAO, ClientDA0, ProductDAO, OrderDAO. Clasa AbstractDAO implementeaza toate operatiile, iar celelalte 3 extinde aceasta clasa.

Pachetul model contine cele 3 clase: Client, Comanda, Product. Fiecare clasa contine atributele corespunzatoare fiecarui tabele, gettere si settere.

Pachetul presentation contine toate cele 4 controllere: Controller, Controller1, Controller2, Controller3 care apleaza prin ajutorul claselor din pachetul bll operatiile din AbstractDA0.

Acest pachet va contine si clasa View unde este construita interfata principala, de unde se alege pe ce tabele se doreste sa se faca operatiile.

Mai avem pachetul de start unde se gaseste clasa Main care va porni executia programul.

Pachetul View contine 3 clase: View1, View2, View3 fiecare construind interfata pentru tabela client, produs, respective comanda.Diagram

Description automatically generated

# 4.Implementare

# Clase

* Clasa AbstarctDAO este o clasa abstract care creeaza fiecare query si metoda corespunzatoare prin care se vor transmite parametrii.

Text

Description automatically generated

In cele de mai sus s-a realizat creare interogarii select si se construieste corespunzator secventei in sql : SELECT \* FROM table\_name WHERE conditon = value;

In aceasta clasa se creaza conform pozei de mai sus toate interogarile: createSelectQuery(String field); , createInsertQuery(String field, int id, List<String> fields); , createDeleteQuery(int id); , createObject(Result result );

Corespunzatoare acestor metode s-au creat urmatoarele metode care le apeleaza pe cele prezentate mai sus cu valorile date de catre utilizator, preluand coloanele din tabel. Metodele sunt: insert(T t); , update(T t, int id, List<String> field1), delete(); , viewAll();

Se poate face cautari in tabela dupa id, fapt realizat in metoda findById( int id), findAll();

AbstractDAO este o clasa care implementeaza metodele la caz general, fiecare metoda poate fi apelata pe alt tabel, cu alte valori.

Clasa Controller creaza ascultatori pentru cele 3 butoane. La apasare butonul de ClientOperations se va deschide deschide interfata creata in View1 corespunzatoare operatiilor pe tabela Client. La apasarea butonul de ProductOperation se va deschide interfata View2 pe care se vor face operatiile pe tabela product, iar la apasarea butonul de OrderOperation se vor face operatiile pe table comanda.

Fiecare clasa View1, View2, View3 are cate un controller ( Controller1, Controller2, respective Controller3) unde se vor crea asculatori pe fiecare buton gasit pe interfata.

# Java doc

Javadoc este un generator de documentație creat de Sun Microsystems pentru limbajul Java (deținut acum de Oracle Corporation) pentru a genera documentație API în format HTML din codul sursă Java.

Formatul „comentarii document” utilizat de Javadoc este standardul industrial de facto pentru documentarea claselor Java. Unele IDE-uri cum ar fi IntelliJ IDEA, NetBeans și Eclipse, generează automat Javadoc HTML. Mulți editori de fișiere ajută utilizatorul să producă sursa Javadoc și folosesc informațiile Javadoc ca referințe interne pentru programator.

Javadoc nu afectează performanța în Java, deoarece toate comentariile sunt eliminate în momentul compilării. Scrierea de comentarii și Javadoc este pentru a înțelege mai bine codul și, astfel, a-l menține mai bine.

# 5.Rezultate

Utilizatorii pot adauga, sterge, actualiza si vizualiza date din tabele, se poat crea clienti, produse, dar si cleinti. La adaugrea unei comenzi daca aceasta respecta criteriile impuse la cantitate, aceasta va fi procesata cu succes si se va genera un fisier txt servind ca bon fiscal.

In urma rulatii programului se poate observa ca functioneaza corect si ca face toate operatiile dorite spre a se implementa.

# 6.Concluzii si dezvoltari ulterioare

Metoda se poate imbunatatii, mai ales in cazul comenzii. O idee ar fi sa se poata plasa comenzi cu mai multe produse pe acelasi bon, pe aceiasi comanda. O alta imbunatatire ar putea fi calcularea unui pret a tuturor produselor adaugate in comanda.

Totodata, o alta imbunatatire ar putea fi adusa si bazei de date devenind una mult mai compleza, cu tabele intermediare si suportand mai multe operatii.

# 7.Bibliografie

<https://gitlab.com/utcn_dsrl/pt-layered-architecture>

<https://gitlab.com/utcn_dsrl/pt-reflection-example>

<https://stackoverflow.com/questions/49831334/trying-to-display-all-data-from-database-table-into-jtable-using-reflection>

<https://stackoverflow.com/questions/58507222/insert-values-to-dynamic-mysql-table-without-knowing-its-columns>

<https://www.javatpoint.com/java-jtable>

<https://docs.oracle.com/javase/tutorial/uiswing/components/table.html>

<https://www.javatpoint.com/steps-to-connect-to-the-database-in-java>

<https://www.baeldung.com/simplifying-the-data-access-layer-with-spring-and-java-generics>