DOCUMENTAȚIE

NUME: IONAȘ ALEX-RĂZVAN

Cuprins

[DOCUMENTAȚIE 1](#_Toc101338328)

[TEMA *3* 1](#_Toc101338329)

[1. Obiectivul temei 3](#_Toc101338330)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc101338331)

[3. Proiectare 4](#_Toc101338332)

[4. Implementare 4](#_Toc101338333)

[5. Rezultate 9](#_Toc101338334)

[6. Concluzii 9](#_Toc101338335)

[7. Bibliografie 10](#_Toc101338336)

# Obiectivul temei

Obiectivul principal presupune dezvoltarea unei aplicații Java capabilă de a gestiona comenzile clienților unui depozit.

De asemenea, trebuie creată o interfață prin intermediul căreia utilizatorul aplicației să introducă date ce urmează a fi prelucrate.

Obiectivele secundare care stau la baza îndeplinirii obiectivului principal sunt următoarele:

1. abstractizarea informațiilor preluate din cerință
2. determinarea claselor și atributelor, aceasta reprezentând partea de structură a proiectului
3. determinarea metodelor specifice pentru a îndeplini funcționalitatea cerută
4. implementarea și testarea aplicației de gestiune a comenzilor

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

Cum am precizat, aplicația trebuie să implementeze următoarele cerințe:

1. utilizatorul introduce date pentru adăugarea unui client nou
2. utilizatorul introduce date pentru adăugarea unui produs nou
3. utilizatorul poate modifica un client existent
4. utilizatorul poate modifica un produs existent
5. utilizatorul poate șterge un client
6. utilizatorul poate șterge un produs
7. utilizatorul poate crea comenzi pe baza clienților și produselor deja existente în baza de date
8. utilizatorul poate vizualiza atât produsele, cât și clienții din baza de date

De asemenea, aplicația trebuie să

1. fie intuitivă și ușor de folosit
2. să furnizeze rezultate corecte
3. să furnizeze rapid rezultatele cerute

Scenariile posibile sunt legate de felul în care utilizatorii folosesc aplicația. Pentru a avea un scenariu de succes, utilizatorul trebuie să introducă date valide în aplicație. De exemplu pentru introducerea unui client nou, trebuie introdus un ID care nu există deja în baza de date, un nume, un prenume și un email după care se va apăsa butonul “Adaugă”.

Dacă ID=ul este unic, clientul va fi adăugat în baza de date. În caz contrar, acesta nu va fi adăugat. Similar se procerează pentru introducerea unui produs în baza de date.

# 3. Proiectare

Pentru proiectare, am folosit o arhitectură de tip “Layer” (straturi). Aceasta presupune un layer de prezentare care reprezintă clasele care definesc interfața aplicației. Avem opt clase pentru realizarea interfeței.

Următorul layer este “Business Layer” care conține clasele care încapsulează logica aplicației. În cazul nostru, avem trei clase la acest layer: “ClientBLL”, “ProductBLL”, “ComandaBLL” care fac legătura dintre baza de date și interfață.

Ultimul layer este “Data Acces Layer” care conține clasele care conțin interogările peste baza de date și conexiunea care se realizează cu aceasta. “AbstractDAO”, “ClientDAO”, “ProductDAO”, “ProductDAO” și “ConnectionFactory” sunt clasele care fac parte din acest layer.

Toate aceste layere au la bază clasele aflate in pachetul “Model” care ilustrează modul în care datele sunt stocate în baza de date.

# 4. Implementare

Descrierea clasei ClientBLL:  
 În clasa ClientBLL se implementează metodele pentru efectuarea operațiilor cerute asupra clienților din baza de date. Aceasta are ca atribute un “ClientDAO” , clasă care extinde “AbstractDAO”. Are următoarele metode:

1. findAll – această metoda caută toți clienții din baza de date și returnează o listă cu acestia, care va fi afișată în interfață sub forma unui tabel

2. insertInto - se inserează în baza de date clientul ale carui date au fost preluate din interfață prin intermediul parametrului "args".

3. deleteFrom - se șterge din baza de date clientul ale carui date au fost preluate din interfață prin intermediul parametrului "args".

4. editClient - se editeaza în baza de date clientul ale carui date au fost preluate din interfață prin intermediul parametrilor "id", "fieldNames" și "values"

Descrierea clasei ProductBLL:

În clasa ProductBLL se implementează metodele pentru efectuarea operațiilor cerute asupra produselor din baza de date. Aceasta are ca atribute un “ProductDAO” , clasă care extinde “AbstractDAO”. Are următoarele metode:

1. findAll – această metoda caută toate produsele din baza de date și returnează o listă cu acestea, care va fi afișată în interfață sub forma unui tabel

2. insertInto - se inserează în baza de date produsul ale carui date au fost preluate din interfață prin intermediul parametrului "args".

3. deleteFrom - se șterge din baza de date produsul ale carui date au fost preluate din interfață prin intermediul parametrului "args".

4. editProduct - se editeaza în baza de date produsul ale carui date au fost preluate din interfață prin intermediul parametrilor "id", "fieldNames" și "values"

Descrierea clasei ComandaBLL:

În clasa ComandaBLL se implementează metodele pentru efectuarea operațiilor cerute asupra clienților din baza de date. Aceasta are ca atribute un “ComandaDAO” , clasă care extinde “AbstractDAO”. Are următoarele metode:

1. findAll – se cauta toate comenzile. Funcția este folosită pentru determinarea ID-ului comenzii curente.

2. createOrder - se inserează în baza de date comanda ale cărei date au fost preluate din interfață prin intermediul parametrului "args".

3. scadeStoc - la plasarea unei comenzi se scade stocul produsului respectiv la valoarea parametrului „stocNou”

Descrierea clasei AbstractDAO

Clasa AbstractDAO se ocupa cu efectuarea operatiilor asupra bazei de date. Parametrul T reprezinta clasa sau obiectul pe care realizam operatiile.

Aceasta are următoarele metode:

1. insertInto - se creează query-ul pentru inserearea unui obiect pe baza argumentelor oferite.

Acest query va fi de forma “INSERT INTO “typeof T” (field1, field2, …, field n) values (value1, value2, …, value n);

2. createSelectQuery – se creează query-ul pentru selectarea unui obiect pe baza unui camp oferit ca parametru. Acest query va fi de forma “ SELECT \* FROM “typeof T” WHERE field = ?”

3. createFindAllQuery – se creează query-ul pentru selecarea tuturor obiectelor dintr-o tabelă. Acest query va fi de forma “SELECT \* FROM “typeof T”

4. deleteQuery – se creeaza query-ul pentru ștergerea unui obiect pe baza id-ului oferit ca parametru. Acest query va avea forma “DELETE FROM “typeof T” WHERE ID = “id” ; “

5. updateQuery - se creeaza query-ul pentru actualizarea unui obiect pe baza parametrilor.

Acest query va fi de forma: “ UPDATE “typeof T” SET field1 = “value1” … fieldn = “value n” WHERE ID = id; “ Parametrii sunt “id”, o listă de stringuri pentru campuri (field1, field2, ...), și o listă de stinguri pentru valori (value1, value2 ...)

Pentru aceste query-uri, „typeof T” reprezintă tipul obiectului la care facem referire, de exemplu “Client” sau “Comanda”.

6. findAll - se efectueaza operatia de cautare a tuturor obiectelor din tabela T si se returneaza o listă cu acestea, urmând a fi afișate în interfață. Se foloseste o conexiune, un preparedStatement care va fi returnat de către metoda createFindAllQuery și un resultSet care va fi trimis ca parametru metodei createObjects.

7. findById - se efectuează operația de căutare a obiectului din tabela T cu id-ul "id" și se returnează obiectul. PrepareStatment-ul pentru această metoda va fi cel rezultat din metoda createSelectQuery unde se trimite ca parametru “id”.

8. createObjects – această metodă va crea în mod generic, pe baza constructorilor fiecărei clase din pachetul “Model” o listă de obiecte, pe baza unui resultSet trimis ca parametru, care în cele din urmă va fi returnată pentru afișare în interfață sau pentru determinarea numărului de rezultate, care poate fi folosit pentru determinarea id-ului unei noi inserări in tabel.

9. insert - se inserează în tabelul T obiectul preluat de la interfață prin parametrul "args".

Se creează un prepareStatement cu ajutorul metodei insertInto descrisă mai sus.

10. deleteFrom - se șterge din tabelul T obiectul cu id-ul "id", transmis de la una dintre clasele din pachetul “BLL”. Query-ul pentru această metoda se creează și returnează de către metoda deleteQuery unde se transmite id-ul ca parametru.

11. updateTable - în tabelul T, obiectului cu id-ul "id", i se vor modifica câmpurile transmise prin parametrul “fieldNames” cu valorile transmise prin parametrul „values”. Query-ul va fi generat de către metoda “updateQuery” care primește parametrii menționați mai sus.

Descrierea clasei GUI

Clasa GUI reprezintă prima fereastră care se deschide atunci când rulăm programul.

Aceasta conține trei butoane: “Client”, care ne va deschide meniul pentru clienți, “Comanda”, care ne va deschide meniul pentru comenzi și “Produs”, care ne va deschide meniul pentru produs.

Descrierea clasei ClientWindow

Clasa ClientWindow reprezintă meniul pentru clienți. Aceasta conține patru butoane pe care le voi descrie în continuare.

1. Butonul “Adauga client” va deschide un formular pentru adăugarea unui client

2. Butonul „Sterge client” va deschide un dialogBox unde se va introduce id-ul clientului care se dorește să fie șters

3. Butonul “Editeaza client” va deschide un formular pentru editarea informațiilor unui client

4. Butonul “Vizualizare clienti” va efectua query-ul de “select \* all from client” și va afișa rezultatul în interfață intr-un Jtable

Descrierea clasei AddClientForm

Clasa AddClientForm reprezintă formularul de adăugare al unui client. Conține patru textField-uri, unde se va introduce numele, prenumele, id-ul și email-ul clientului. De asemenea, conține și un buton care, la apăsare, preia datele introduse și efectuează operația de “INSERT INTO CLIENT (id, name, prenume, email) values (String.valueOf(textField\_id), textField\_Name.getText(), …);

Descrierea clasei EditClientForm

Clasa EditClientForm reprezintă formularul de editare a informațiilor unui client. Conține patru textField-uri, unde se va introduce id-ul clientului, celelalte câmpuri putând fi completate la nevoie. De exemplu, putem modifica doar câmpul de nume al unui client, necesar fiind doar să introducem id-ul acestuia, care poate fi preluat foarte ușor din tabelul creat la apăsarea butonului “Vizualizare Clienti”. De asemenea, conține și un buton care, la apăsare, preia datele introduse și efectuează operația de “UPDATE CLIENT SET field1 = value1, field2 = value2 … );”

Descrierea clasei ProdusWindow

Clasa ProdusWindow reprezintă meniul pentru produse. Aceasta conține patru butoane pe care le voi descrie în continuare.

1. Butonul “Adauga produs” va deschide un formular pentru adăugarea unui produs

2. Butonul „Sterge produs” va deschide un dialogBox unde se va introduce id-ul produsului care se dorește să fie șters

3. Butonul “Editeaza produs” va deschide un formular pentru editarea informațiilor unui produs

4. Butonul “Vizualizare produse” va efectua query-ul de “select \* all from produs” și va afișa rezultatul în interfață intr-un Jtable

Descrierea clasei AddProdusForm

Clasa AddProdusForm reprezintă formularul de adăugare al unui produs. Conține patru textField-uri, unde se va introduce numele, pretul, id-ul și stocul produsului. De asemenea, conține și un buton care, la apăsare, preia datele introduse și efectuează operația de “INSERT INTO produs (id, name, price, stock) values (String.valueOf(textField\_id), textField\_Name.getText(), …);

Descrierea clasei EditProdusForm

Clasa EditProdusForm reprezintă formularul de editare a informațiilor unui produs. Conține patru textField-uri, unde se va introduce id-ul produsului, celelalte câmpuri putând fi completate la nevoie. De exemplu, putem modifica doar câmpul de nume al unui produs, necesar fiind doar să introducem id-ul acestuia, care poate fi preluat foarte ușor din tabelul creat la apăsarea butonului “Vizualizare Produse”. De asemenea, conține și un buton care, la apăsare, preia datele introduse și efectuează operația de “UPDATE PRODUCT SET field1 = value1, field2 = value2 … );”

Descrierea clasei ComandaWindow

Clasa ComandWindow este responsabilă pentru plasarea de comenzi. Aceasta conține, în partea stângă indicații pentru plasarea unei comenzi, iar în partea dreaptă două comboBox-uri, unul pentru selectarea clientului, iar celălalt pentu selectarea produsului. De asemenea, conține și un textField pentru introducerea cantității dorite. Pentru simplitatea proiectului, am considereat doar cazul în care un client poate comanda un singur produs. De asemenea, dacă se introduce o cantitate mai mare decât cea din stoc, un mesaj de eroare va fi afișat pe ecran prin intermediul unui dialogBox, iar baza de date va rămâne neschimbată, nicio operație efectuându-se asupra acesteia. Tot în partea dreaptă se află si butonul “Comanda” care, la apăsare, va prelua datele din comboBoxuri si textField, le va transmite ca parametrii metodei insertInto din ComandaBLL, care le va transmite mai departe in AbstractDAO, unde se va efectua query-ul descris mai sus.

De asemenea, la apăsarea butonului, se creează și un fișier denumit formal “order” + id-ul comenzii, în care vor fi precizate, id-ul comenzii, id-ul, numele, prenumele și email-ul clientului, data la care a fost efectuată comanda, dar și cantitatea și prețul total al comenzii.

Descrierea claselor din pachetul Model

Clasele din acest pachet, sunt destul de asemănătoare. Fiecare modelează unu la unu tabelele create în baza de date. Acestea au constructori, folosiți în special de către metoda createObject din clasa AbstractDAO, dar și getteri și setteri. Pe scurt, nimic special aici.

Descrierea clasei MainClass

Clasa MainClass instanțiază un obiect de tipul GUI și îl face vizibil. Astfel, pe ecran, va apărea fereastra principală.

# 5. Rezultate

În urma testării, toate metodele funcționează conform așteptărilor cu exceptia cazului în care inserăm un obiect cu un id deja existent în baza de date. Deși în baza de date NU se inserează acest obiect, nefiind permise id-uri duplicate, pe ecran va apărea un mesaj de succes.

# 6. Concluzii

În concluzie, aplicația poate fi dezvoltată foarte mult. Pot fi adăugate tabele noi, de exemplu “furnizor” sau “angajat”. De asemenea poate fi implementat cazul în care un client comandă mai multe produse, în acest caz fiind necesară crearea unei tabele noi, întrucât vom avea o relatie M:M care trebuie rezolvată.

# 7. Bibliografie

# 

<https://www.vogella.com/tutorials/JUnit/article.html>

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

<https://www.javatpoint.com/string-tokenizer-in-java>

<https://www.javatpoint.com/java-get-current-date>

<https://www.geeksforgeeks.org/java-swing-jtable/>

<https://stackoverflow.com/questions/58939/jcombobox-selection-change-listener>

<https://www.baeldung.com/javadoc>

https://dzone.com/articles/layers-standard-enterprise