**Order Management**

**Moldovan Vlad Madalin**

**CUPRINS:**

**1.Obiectivul temei**

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

**3. Proiectare (decizii de proiectare, diagrame UML, structure de date, proiectare clase, interfete, relatii , packages, algoritmi, interfata utilizator)**

**3.1 Decizii de proiectare**

**3.2 Diagrama UML**

**3.3 Packages**

**4.Implementare**

**4.1 Clasa Functions**

**4.2 Clasa ConnectionFactory**

**4.3 Clasa ClientDAO**

**4.4 Clasa OrderClientDAO**

**4.5 Clasa OrderItemDAO**

**4.6 Clasa ProductDAO**

**4.7 Clasa Client**

**4.8 Clasa OrderClient**

**4.9 Clasa OrderItem**

**4.10 Clasa Product**

**4.11 Clasa Bill**

**4.12 Clasa ParseFile**

**4.13 Clasa Start**

**5.Concluzii**

**6.Bibliografie**

**1.Obiectivul temei**

Obiectivul principal al temei este de a implementa o aplicatie de Order Management pentru procesarea comenzilor unui depozit . Operatile de care aceasta aplicatie dispune sunt : adaugare client , stergere client , adaugare produs , stergere produs , adaugare comanda , generare report ( client , produs , order ) . Obiectivul secundar al temei este de a genera facturi pentru fiecare comanda inregistrata si reporturile sub forma de tabel in format PDF , dar si de a utiliza o baza de date pentru stocarea datelor .

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

Cerinta problemei este de a implementa operatile de adaugare client , stergere client , adaugare produs , stergere produs , adaugare comanda , generare report pentru procesarea comenzilor dintr-un depozit . Comenzile vor fi preluate dintr-un fisier .txt .

Exemplu de operatii:

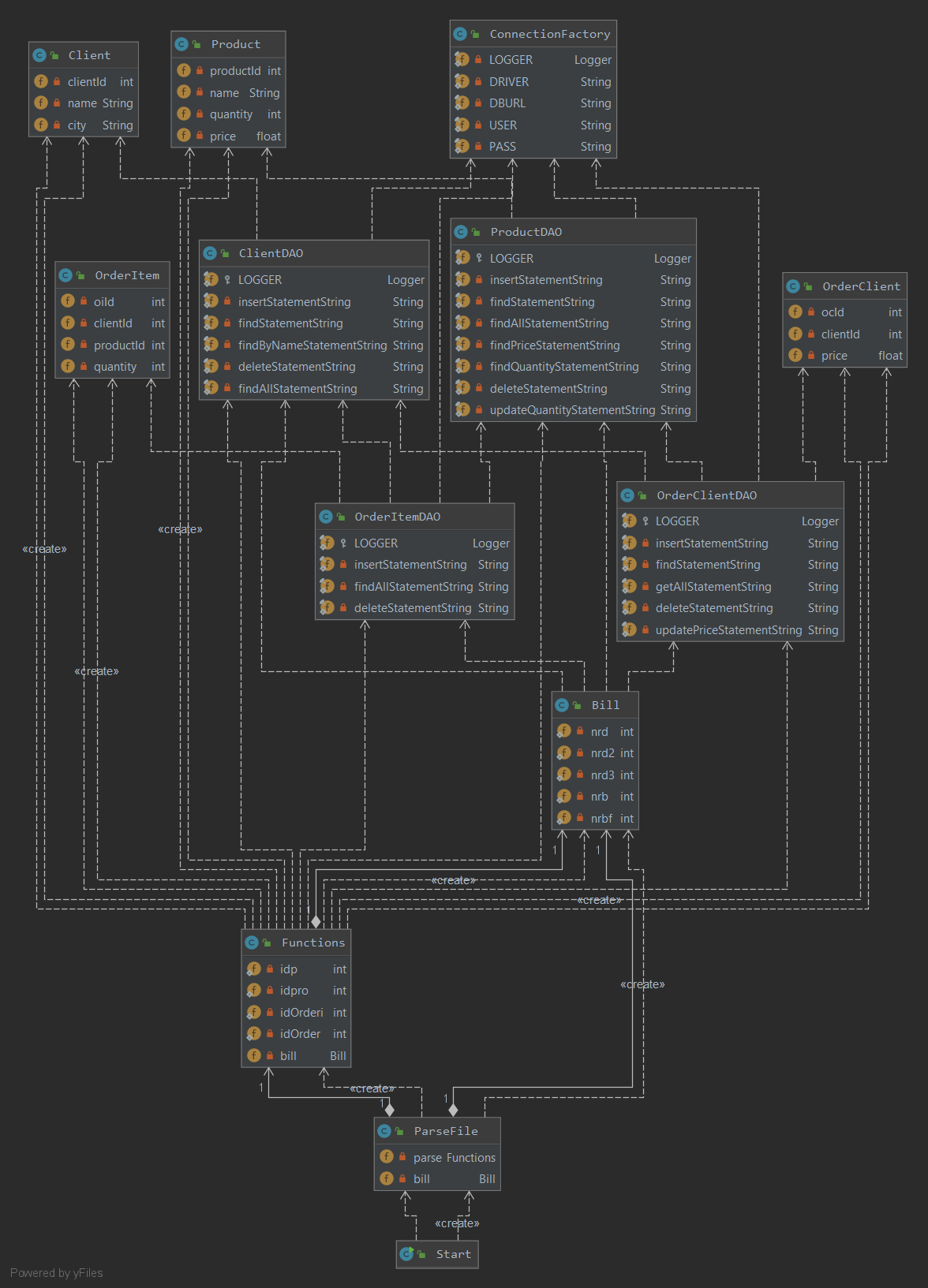
* “Insert client: Ion Popescu, Bucuresti” -> Adaugarea in baza de date a clientului cu numele Ion Popescu din orasul Bucuresti .
* “Delete client: Ion Popescu” -> Stergerea din baza de date a clientului cu numele Ion Popescu.
* “Insert product: apple, 20, 1” -> Adaugarea in baza de date a produsului apple in cantitate de 20 cu pretul 1 .
* “Order: Ion Popescu, apple, 5” -> Crearea unei comenzi pe numele Ion Popescu pentru achizitionarea obiectului apple in cantitate de 5 .
* “Report client” , “Report product” , “Report order” -> Acestea sunt comenzi pentru vizualizarea tabelelor client , product , order din baza de date a aplicatie la un moment dat .

**3.** **Proiectare (decizii de proiectare, diagrame UML, structure de date, proiectare clase, interfete, relatii , packages, algoritmi, interfata utilizator)**

**3.1 Decizii de proiectare**

Pentru implementarea bazei de date au fost folosite 4 tabele : Client ( clientId , name , city ) , Product ( productid , name , quantity , price ) , Order\_Item ( oiId , clientId , productId , quantity ) , Order\_Client ( OcId , clientId , price ) .Tabelele au fost implementate in acest mod pentru a usura implementarea operatiilor pentru aceasta aplicatie . Pentru procesa unei comenzi si inregistrarea ei in baza de date se vor utiliza tabelele Order\_Item si Order\_Client . In tabelul Order\_Item se va introduce numele clientului ,produsul si cantitatea comandata , iar in Order\_Client se va introduce numele clientului si pretul total pe care acesta il are de platit ( De exemplu: daca fisierul text va contine aceste doua comenzi “Order: Luca George, apple, 5” , “Order: Luca George, lemon, 5” atunci in tabelul Order\_Item se vor adauga doua linii noi : una care va contine clientul luca George , produsul apple in cantitate de 5 , iar cealalta clientul Luca George , produsul lemon in cantitate de 5 , iar in tabelul Order\_Client se va introduce linia care contine clientul Luca George cu pretul total de platit 15) . Pentru generarea facturilor , aplicatia va genera pentru fiecare comanda in parte o factura prin care se va specifica numele clientului , produsul si cantitatea comandata ( Exemplu de facture : “Bill: client Luca George,product apple,quantity 5.” ) . In cazul in care cantitatea pe care clientul o solicita este mai mare decat cantitatea existenta a produsului in baza de date se va genera un mesaj de eroare ( Exemplu : “Cantitatea ceruta este prea mare”). Dupa ce toate operatiile din fisierul text au fost implementate se va genera o facture finala pentru toti clientii care au facut comenzi in care se va specifica numele clientului si pretul total pe care acesta il va avea de platit ( Exemplu de facture finala : “Total bill: client Luca George,total price 15.0 . “ ) . Daca in fisierul text se doreste adaugarea unui produs de doua ori , atunci prima data produsul va fi introdus in baza de date , iar a doua oara cantitatea produsului va fi schimbata prin insumarea celor doua cantitati ( De exemplu : daca operatia “Insert product: apple, 20, 1” este introdusa de 2 ori in fisierul text atunci la final baza de date va contine produsul apple in cantitate de 40 ) . In cazul in care se va sterge un client care are comenzi in baza de date , atunci se vor sterge prima data comenzile aflate pe numele acestuia , iar dupa aceea se va sterge clientul din baza de date .

**3.2 Diagrama UML**



**3.3 Packages**

Pachetele utilizate de aceasta aplicatie sunt urmatoarele : businessLogicLayer , connectionLayer , dataAccesLayer , modelLayer , presentationLayer , startLayer .

* businessLogicLayer - > Acest pachet contine logica aplicatie , aici este implementata clasa Functions care contine metodele pentru realizarea operatiilor corespunzatoare cerintei .
* connectionLayer -> Acest pachet contine clasa pentru realizarea conexiunii aplicatiei la baza de date , aceasta clasa este ConnectionFactory
* dataAccesLayer -> Acest pachet contine clasele care vor interactiona direct cu baza de date pentru extragerea si adaugarea informatiilor necesare pentru realizarea aplicatie . Clasele pachetului sunt : ClientDAO , OrderClientDAO , OrderItemDAO , ProductDAO .
* modelLayer -> Acest pachet contine clasele model pentru tabelele din baza de date . Clasele pachetului sunt : Client , OrderClient , OrderItem , Product .
* presentationLayer -> Acest pachet contine partea de procesare si afisare a datelor . Clasele pachetului sunt : Bill , ParseFile .
* startLayer -> Acest pachet reprezinta pachetul din care se va porni aplicatia , fiind constituit din clasa main a aplicatiei , clasa Start .

**4.Implementare**

**4.1 Clasa Functions**

Clasa Functions este utilizata pentru implementarea metodelor necesare procesarii operatiilor acestei aplicatii . Atributele clasei sunt : idp ( reprezentand id-ul clientului ) , idpro ( reprezentand id-ul produsului ) , idOrderi ( representant id-ul OrderItem ) , idOrder ( reprezentand id-ul idOrder ) .

Metode:

* public Functions()

- constructorul clasei

* public void insertClient(String a)

- metoda care introduce un client

- parametrul “a” contineliniadinfisiercorespunzatoareinserariiunuiclient*.*

* public void deleteClient(String a)

- metoda care sterge un client si order-urile acestuia

- parametrul “a” contine linia din fisier corespunzatoare stergerii unui client.

* public void insertProduct(String a)

- metoda care introduce un produs.

- parametrul “a’ contine linia din fisier corespunzatoare inserarii unui produs.

* public void deleteProduct(String a)

- metoda care sterge un produs.

- parametrul “a” contine linia din fisier corespunzatoare stergerii unui produs.

* public void generateOrder(String a)

- metoda care va genera reporturile pentru client,product si order.

- parametrul “a” contine linia din fisier corespunzatoare generarii unui report.

**4.2 Clasa ConnectionFactory**

Clasa ConnectionFactory este utilizata pentru a realiza conexiunea aplicatiei la baza de date

Metode:

* public ConnectionFactory()

- constructorul clasei

* public static Connection getConnection()

- metoda pentru obtinerea conexiunii la baza de date

* public static void close(Connection connection)

- metoda pentru inchiderea bazei de date

* public static void close(Statement statement)

- metoda pentru inchiderea unui statement

* public static void close(ResultSet resultSet)

- metoda pentru inchiderea setului de rezultate

**4.3 Clasa ClientDAO**

Clasa ClientDAO este clasa care comunica cu tabelul Client din baza de date .

Metode:

* public static String findById(int clientId)

- metoda care gaseste numele unui client dupa id-ul acestuia

* public static int findByName(String name)

- metoda care gaseste id-ul unui client dupa numele acestuia

* public static void findAll(PdfPTable table)

- metoda care genereaza toate elementele din tabelul Client al bazei de date

* public static int insert(Client client)

- metoda care insereaza un client in tabelul client din baza de date

* public static int delete(String client)

- metoda care sterge un client din baza de date dupa numele acestuia

**4.4 Clasa OrderCientDAO**

Clasa OrderClientDAO este clasa care comunica cu tabelul order\_client din baza de date .

Metode:

* public static void updatePrice(int id, float nr)

- metoda care modifica pretul unui orderClient dupa id-ul specificat

* public static Boolean findById(int id)

- metoda care verifica daca un orderClient cu id-ul specificat exista sau nu

* public static int insert(OrderClient order)

- metoda care insereaza un orderClient in tabelul order\_client

* public static ArrayList<String> getFinalBil()

- metoda care genereaza toate elementele tabelului order\_client

* public static int delete(int id)

- metoda care sterge un orderClient din tabelul order\_client al bazei de date

**4.5 Clasa OrderItemDAO**

Clasa OrderItemDAO este clasa care comunica cu tabelul order\_item din baza de date.

Metoda:

* public static int insert(OrderItem item)

-metoda care insereaza un orderItem in tabelul order\_item al bazei de date

* public static void findAll(PdfPTable table)

- metoda care genereaza toate elementele din tabelul order\_item al bazei de date

* public static int delete(int id)

- metoda care sterge un orderItem din tabelul order\_item al bazei de date dupa id-ul acestuia

**4.6 Clasa ProductDAO**

Clasa ProductDAO este clasa care comunica cu tabelul product din baza de date

Metode:

* public static void updateQuantity(String product, int nr)

- metoda care modifica pretul unui produs

* public static int findByName(String name)

- metoda care gaseste id-ul unui produs dupa numele acestuia

* public static int delete(String product)

- metoda care va sterge un produs dupa numele acestuia

* public static float findById(int id)

- metoda care gaseste pretul unui produs dupa id-ul acestuia

* public static String findName(int id)

- metoda care gaseste numele unui produs dupa id-ul acestuia

* public static int findQuantity(int id)

- metoda care gaseste cantitatea unui produs dupa id-ul acestuia

* public static int insert(Product product)

- metoda care insereaza un produs in tabelul product al bazei de date

* public static void findAll(PdfPTable table)

- metoda care genereaza toate elementele tabelului product din baza de date

**4.7 Clasa Client**

Clasa Client reprezinta modelul pentru tabelul client din baza de date . Clasa contine urmatoarele atribute : clientId , name , city .

Metode:

* public Client(int clientId,String name, String city)

- constructorul clasei

* public String getName()

- metoda pentru obtinerea numelui unui client

* public String getCity()

- metoda pentru obtinerea orasului unui client

* public int getIdClient()

- metoda pentru obtinerea id-ului unui client

**4.8 Clasa OrderClient**

Clasa OrderClient reprezinta modelul pentru tabelul order\_client din baza de date . Clasa contine urmatoarele atribute : ocId , clientId , price .

Metode:

* public OrderClient(int ocId, int clientId, float price)

- constructorul clasei

* public int getOcId()

- metoda pentru obtinerea id-ului unui orderClient

* public int getClientId()

- metoda pentru obtinerea id-ului clientului

* public float getPrice()

- metoda pentru obtinerea pretului

**4.9 Clasa OrderItem**

Clasa OrderItem reprezinta modelul pentru tabelul order\_item din baza de date . Clasa contine urmatoarele atribute :

oiId , clientId , productId , quantity .

Metode:

* public OrderItem(int oiId, int clientId, int productId, int quantity)

- constructorul clasei

* public int getOiId()

- metoda pentru obtinerea id-ului orderItem

* public int getClientId()

- metoda pentru obtinerea id-ului clientului

* public int getProductId()

- metoda pentru obtinerea id-ului produsului

* public int getQuantity()

- metoda pentru obtinerea cantitatii comandate

**4.10 Clasa Product**

Clasa Product reprezinta modelul pentru tabelul product din baza de date . Clasa contine urmatoarele atribute : productId , name , quantity , price.

Metode:

* public Product(int productId,String name, int quantity, float price)

- constructorul clasei

* public String getName()

- metoda pentru obtinerea numelui unui produs

* public int getQuantity()

- metoda pentru obtinerea cantitatii unui produs

* public float getPrice()

- metoda pentru obtinerea pretului unui produs

* public int getProductId()

- metoda pentru obtinerea id-ului unui produs

**4.11 Clasa Bill**

Clasa Bill este clasa care genereaza pdf-urile .

Metode:

* private void addTableHeader(PdfPTable table)

-Metoda pentru denumirea header-ului pentru tabelul din pdf-ul pentru "Report Client".

* public void generateReport(String a)

-Metoda care va genera report-urile sub format pdf

* private void addTableHeader2(PdfPTable table)

-Metoda pentru denumirea header-ului pentru tabelul din pdf-ul pentru "Report Product”.

* private void addTableHeader3(PdfPTable table)

-Metoda pentru denumirea header-ului pentru tabelul din pdf-ul pentru "Report Order".

* public void generateBill(String msg)

-Metoda care va genera factura pentru fiecare order.

* public void generateFinalBill()

-Metoda care genereaza factura finala a unui client.

**4.12 Clasa ParseFile**

Clasa ParseFile este clasa pentru parsarea fisierului.

Metode:

* public ParseFile()

- constructorul clasei

* public void readFile(String input)

- metoda pentru citirea si parsarea fisierului

**4.13 Clasa Start**

Clasa Start reprezinta clasa main a aplicatiei

**5. Concluzii**

Au fost implementate operatiile de adaugare client , stergere client , adaugare produs , stergere produs , adaugare comanda , generare report ( client , produs , order ) pentru procesarea comenzilor unui depozit impreuna cu emiterea facturilor . Din aceasta tema am invatat sa lucrez mai bine cu baze de date , dar si sa generez un fisier pdf si Javadoc .

**6. Bibliografie**

1. <https://mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/>

2. <https://www.baeldung.com/java-pdf-creation>

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

4. <https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html>

5. Suportul pdf oferit pentru aceasta tema impreuna cu clasesle suport oferite .