ORDER MANAGEMENT

Stefan Tudor

Grupa 30229

Profesor laborator: Mitrea Dan

Cuprins:

1.Cerinte functionale………………………………………………

2.Obiective………………………………………………………...

3.Proiectare………………………………………………………..

4.Implementare……………………………………………………

5.Rezultate………………………………………………………...

6.Concluzii………………………………………………………..

7.Bibliografie……………………………………………………..

**1.CERINTE FUNCTIONALE**

In acest proiect se cere implementarea unei aplicatii ce manipuleaza o baza de date ce simuleaza un depozit in care se afla produse ce poti fi comandate de anumiti clienti.Comenzile acestora vor fi si ele inregistrate in baza de date “OrderManagement”.Operatiile ce se pot efectua asupra bazei de date sunt:

* Delete Client : formatul pentru aceasta operatie este “Delete Client: nume client, adresa client”.Sterge un client cu parametrii mentionati din tabela responsabila de clienti.
* Delete Product : formatul pentru aceasta operatie este “Delete Product: nume produs”. Sterge un produs cu parametrul mentionat din tabela responsabila de produse.
* Insert client: formatul pentru aceasta operatie este: “Insert client: nume client, nume adresa”.Insereaza un client in tabela responsabila de clienti.
* Insert product: formatul pentru aceasta operatie este: “Insert product: nume produs, cantitate, pret.Insereaza un produs in tabela responsabila de produse.
* Order: formatul pentru aceasta operatie este: nume client, nume produs, cantitate.Insereaza o comanda in tabelul responsabil de comenzi si creeaza o chitanta PDF cu numele clientului,numele produsului, cantitatea si pretul total.In cazul in care cantitatea dorita depaseste cantitatea produsului din tabelul cu produse, atunci se va genera un document PDF in care se va afisa comanda intreaga impreuna cu un mesaj de eroare care mentioneaza faptul ca stocul este deposit de cantitatea dorita.
* Report client: aceasta operatie creeaza un document PDF in care se va scrie tabelul cu datele despre toti clientii aflati in tabela la momentul in care se citeste aceasta operatie.
* Report order: aceasta operatie creeaza un document PDF in care se va scrie tabelul cu datele despre toate comenzile aflate in tabela la momentul in care se citeste aceasta operatie.
* Report product: aceasta operatie creeaza un document PDF in care se va scrie tabelul cu datele despre toate produsele aflate in tabela la momentul in care se citeste aceasta operatie.

Toate comenzile si datele pentru inserare si sterge for vi citite din fisierul “commands.txt”.

Pentru executare programului se va folosi formatul:

java -jar <nume\_fisier.jar / cale spre fisierul jar> commands.txt

Fisierul .jar se afla in folderol “target” al proiectului iar rezultatele generarii JAVADOC se afla in folderol “src” al proiectului intr-un folder numit “JAVADOC\_FILES”.

Tot in folderul proiectului se vor afla rezultatele sub forma de documente PDF ale comenzilor.De asemenea, in folderol proiectului se afla si fisierul de timp “dump” al bazei de date.

**2.OBIECTIVE**

**Obiectivul** **principal** acestui proiect este de a implementa un proiect ce poate sa manipuleze o baza de date prin intermediul limbajului Java.

**Obiectivele secundare** ale acestui proiect si realizarea acestora realizeaza indeplinirea obiectivului principal.

Aceste obiective secundare sunt:

* Alegerea structurilor de date necesare
* Impartirea claselor
* Dezvoltarea unor anumiti algoritmi ce pot ajuta la o buna functionare a programului
* Implementarea unei solutii
* Testarea programului

**3.PROIECTARE**

Ca si prim pas de proiectare a acestei aplicatii, am ales sa reprezint tabelele din baza de date prin intermediul unor clase cu nume sugestive si care contin aceleasi coloane ce se gasesc in tabele. Pe urma, am creat clasele pentru accesul la baza baza de date, clase ce vor fi responsabile pentru interogari, update-uri si stergeri din table.Aceste clase vor face parte din pachetul “dataaccess”. De asemenea, am creat clasele din BusinessLayer care sunt responsabile de apelul metodelor din clasele referinta din pachetul “dataacess”/ DataAccessLayer.

In continuare, am creat clasa “ConnectionToDB” din pachetul “connection”, care este responsabila pentru crearea unei conexiuni la baza de date create in MySQL.In pachetul “presentation” se afla clasele responsabile pentru generarea documentelor PDF cat si a chitantelor in format PDF.De asemenea, in acest pachet se afla clasa “FileParser”, care este responsabila de citirea comenzilor si a datelor din fisierul “commands.txt” si de a executa aceste comenzi.

**Clasele pentru descrierea tabelelor** sunt construite astfel:

Clasa **“Client”**:

Contine un “int” ce descrie ID-ul clientului, doua variabile instanta de tipul “String” prin care sunt descrise numele clientului si adresa clientului( orasul in care acesta locuieste).

Clasa **“Product”**:

Contine un “int” ce descrie ID-ul produsului, o variabila “int” pentru descrierea cantitatii, o variabila de tip “Double” pentru descrierea pretului si o variabila de tipul “String” pentru descrierea numelui produsului.

Clasa **“OrderItem”** :

Contine o variabila de tipul “int” ce descrie ID-ul comenzii, doua variabile de tipul “String “ ce descriu numele clientului si numele produsului si o variabila de tipul “Double” ce descrie pretul unui produs.

Clasa **“Order”:**

Contine o variabila de tipul “int” pentru ID-ul comenzii, doua variabile de tipul “String” ce descriu numele clientului si numele adresei clientului, o vaiabila “int” pentru descrierea cantitatii si doua variabile de tipul “Double” ce descriu pretul unui produs si pretul total al comenzii.

Clasa **“ConnectionToDB”**:

Contine o variabile de tipul “Logger” si patru variabile de tipul “String” care sunt folosite pentru initializarea Driver-ului, URL-ului responsabil pentru conexiunea la baza de date, numele utilizatorului si parola.

Pachetele folosite in acest proiect sunt: “java.sql”,”java.util.logging”,”java.io”,”com.itextpdf.”.

In **pachetul business** se afla clasele: “ClientBLL”,”OrderBLL”,”OrderItemBLL” si “ProductBLL”.

In **pachetul “connection”** se afla clasa “ConnectionToDB”.

In **pachetul “dataaccess”** se afla clasele: “ClientDAO”, “OrderDAO”, “OrderItemDAO” si “ProductDAO”.

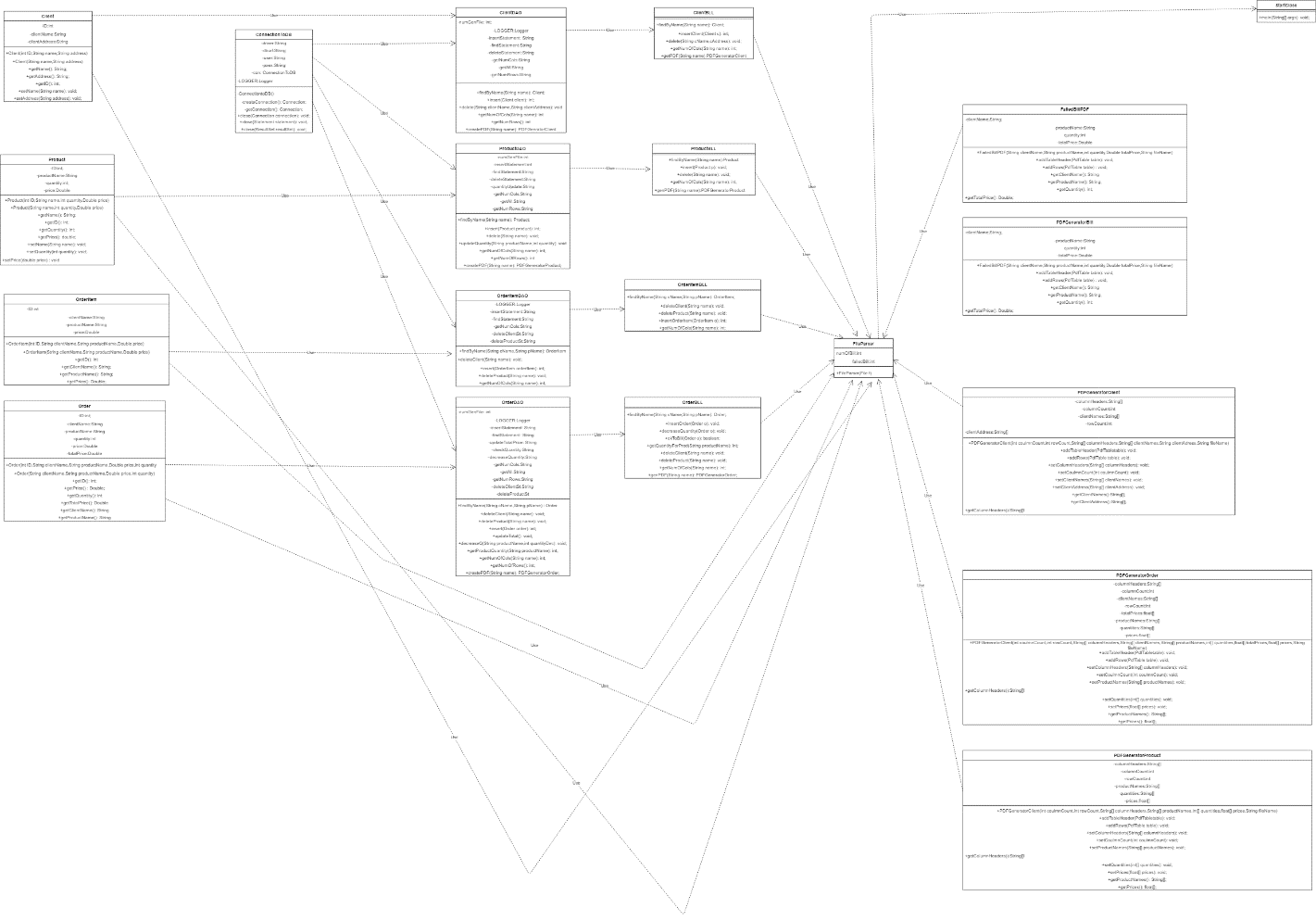
In **pachetul “presentation** se afla clasele: FailedBillPDF, FileParser, PDFGeneratorClient, PDFGeneratorOrder si PDFGeneratorProduct.

In **pachetul “model”** se afla clasele: Client, Product, Order si OrderItem.

In **pachetul “start”** se afla clasa StartClass care este responsabila pentru pornirea programului.

**Diagrama UML:**

In aceasta parte este prezentata diagrama UML a proiectului si relatiile dintre clase.



**4.IMPLEMENTARE**

In aceasta parte a documentatiei va fi prezentata implementarea claselor si a metodelor din acestea.

In clasa **“Client”**: metodele din aceasta clasa sunt gettere si settere alaturi de constructori.

In clasa **“Product”** metodele din aceasta clasa sunt gettere si settere alaturi de constructori.

In clasa **“OrderItem”** metodele din aceasta clasa sunt gettere si settere alaturi de constructori.

In clasa **“Order”** metodele din aceasta clasa sunt gettere si settere alaturi de constructori.

In clasa **“ConnectionToDB”:**

* Metoda **createConnection()** creeaza o conexiune la baza de date prin intermediul obiectului DriverManager care foloseste variabilele instanta “dburl”,”root” si “pass”.
* Metoda **getConnection()** returneaza conexiunea creata prin intermediul metodei createConnection().
* Metodele **close()** care primesc diferiti parametrii de tipul Connection, Statement sau ResultSet inchid conexiunile la baza de date, cu Statement-ul folosit si ResultSet-ul folosit.

In clasa **“ClientDAO”:**

* Metoda **findByName(String name)** creeaza o conexiune la baza de date si pregateste un PreparedStatement pentru a fi executat cu String-ul “findStatement”.Se seteaza numele clientului in query si se executa aceasta interogare.Se vor prelua numele clientului si adresa sa din variabila “resultSet” prin intermediul metodet **getString(<numele coloanei din table>)**.Se construieste un obiect de tipul Client cu valorile returnate.La final se returneaza obiectul creat.
* Metoda **insert(Client client)** creeaza o conexiune la baza de date si pregateste un PreparedStatement pentru a fi executat cu String-ul “insertStatement”.Inainte de inserare se seteaza numele clientului si adresa, apoi se executa update-ul pentru tabelul de clienti.
* Metoda **delete(String clientName, String clientAddress)** creeaza o conexiune la baza de data si pregateste un PreparedStatement pentru a fi executat cu String-ul “deleteStatement”.Se seteaza numele clientului si adresa sa si se executa update-ul.In cele din urma clientul cu numele **clientName** si adresa **clientAddress** va fi sters.
* Metoda **getNumOfCols(String name)** creeaza o conexiune la baza de date si prin intermediul unui PreparedStatement impreuna cu String-ul “getNumCols” returneaza prin executia unui query numarul de coloane din **tabelul Client**.
* Metoda **getNumOfRows()** returneaza numarul de randuri dintr-un tabel.
* Metoda **createPDF(String name)** creeaza o conexiune la baza de date si executa un query care returneaza in variabila resultSet toate datele din tabelul Client.Se va folosi si o variabila de tipul **“ResultSetMetaData”** pentru a extrage numele coloanelor.Se folosesc trei vectori de tipul String pentru a stoca numele clientilor,adresele si numele coloanelor.La finalul metodei se va crea un obiect de tipul **PDFGeneratorClient** cu datele extrase prin intermediul variabilei de tipul “ResultSet”.Se incrementeaza numarul fisierului PDF generat.

In clasa **“OrderDAO”**:

* Metoda **findByName(String cName,String pName)** creeaza o conexiune la baza de date si pregateste un PreparedStatement pentru a fi executat cu String-ul “findStatement”.Se seteaza numele clientului si numele produsului in query si se executa aceasta interogare.Se vor prelua numele clientului ,numele produsului, pretul si cantitatea din variabila “resultSet” prin intermediul metodei **getString(<numele coloanei din table>)**.Se construieste un obiect de tipul Order cu valorile returnate.La final se returneaza obiectul creat.
* Metoda **deleteProduct(String name)** creeaza o conexiune la baza de date si pregateste un PreparedStatement pentru a fi executat cu String-ul “deleteProductSt”.Sterge din tabela pt comenzi randul cu produsul cu numele name.
* Metoda **insert(Order order)** creeaza o conexiune la baza de date si pregateste un PreparedStatement cu String-ul “insertStatement” .Se seteaza pe pozitiile 1,2,3,4 valori avand numele clientului,numele produsului,cantitatea si pretul .
* Metoda **updateTotal()** creeaza o conexiune la baza de date si calculeaza pretul total al unei comenzi.
* Metoda **decreaseQ(String productName,int quantity)** creeaza o conexiune la baza de date si face update la cantitatea din tabela de produse unde se gaseste numele produsului gasit in comanda.Se va scadea cantitatea din stoc cu parametrul primit numit “quantity”.
* Metoda **getProductQuantity(String productName)** creeaza o conexiune si returneaza cantitatea produsului aflat in tabela de produse.
* **Metoda numOfCols(String name)** returneaza numarul de coloane din tabela de comenzi.
* **Metoda numOfRows()** returneaza numarul de randuri din tabela de comenzi.
* Metoda **createPDF(String name)** creeaza o conexiune la baza de date si prin intermediul unui PreparedStatement impreuna cu String-ul “getAll” returneaza prin interogare numele coloanelor,numele clientilor, numele produselor si va genera un obiect de tipul PDFGeneratorOrder cu valorile returnate de ResultSet.

In clasa **ProductDAO:**

* Metoda **findByName(String name)** creeaza o conexiune la baza de date si pregateste un PreparedStatement pentru a fi executat cu String-ul “findStatement”.Se seteaza numele clientului si numele produsului in query si se executa aceasta interogare.Se vor prelua numele produsului, pretul si cantitatea din variabila “resultSet” prin intermediul metodei **getString(<numele coloanei din table>)**.Se construieste un obiect de tipul Product cu valorile returnate.La final se returneaza obiectul creat.
* Metoda **insert(Product product)** creeaza o conexiune la baza de date si pregateste un PreparedStatement cu String-ul “insertStatement” .Se seteaza pe pozitiile 1,2,3 valori avand numele produsului,cantitatea si pretul.
* Metoda **delete(String productName**) creeaza o conexiune la baza de date si sterge din tabela Product randul cu produsul care are numele productName.
* Metoda **updateQuantity(String productName,int quantity)** creeaza o conexiune la baza de date si pregateste un CallableStatement prin intermediul caruia se apeleaza procedura stocata “update\_quantity\_product” , care face update la cantitatea unui produs in caz ca acesta este introdus de mai multe ori in baza de date.
* Metoda **getNumOfCols(String name)** returneaza numarul de coloane din tabela Product.
* Metoda **getNumOfRows()** returneaza numarul de randuri dintr-o tabela.
* Metoda createPDF(String name) **)** creeaza o conexiune la baza de date si prin intermediul unui PreparedStatement impreuna cu String-ul “getAll” returneaza prin interogare numele coloanelor, numele produselor,cantitatile si preturile si va genera un obiect de tipul PDFGeneratorProduct cu valorile returnate de ResultSet.

In clasa **“OrderItemDAO”** :

* Metoda **findByName(String cName,String pName)** creeaza o conexiune la baza de date si pregateste un PreparedStatement pentru a fi executat cu String-ul “findStatement”.Se seteaza numele clientului si numele produsului in query si se executa aceasta interogare.Se vor prelua numele clientului,numele produsului si pretul din variabila “resultSet” prin intermediul metodei **getString(<numele coloanei din table>)**.Se construieste un obiect de tipul OrderItem cu valorile returnate.La final se returneaza obiectul creat.
* Metoda **insert(OrderItem orderItem)** creeaza o conexiune la baza de date si pregateste un PreparedStatement cu String-ul “insertStatement” .Se seteaza pe pozitiile 1,2,3 valori avand numele produsului,numele clientului si pretul.
* Metoda **deleteClient(String name)** sterge din tabela OrderItem un rand la care se gaseste clientul cu numele name.
* Metoda **deleteProduct(String name)** sterge din tabela OrderItem un rand la care se gaseste produsul cu numele name.
* Metoda **getNumOfCols(String name)** returneaza numarul de coloane din tabela OrderItem.

In clasele **“ClientBLL”, “OrderBLL”** , **“OrderItemBLL”** si **“ProductBLL”** se vor afla metode prin intermediul carora se apeleaza metode din pachetul de “dataacess”/ din DataAccessLayer. Metode ce se apeleaza din aceste clase, ce fac parte din pachetul “business”/ din BusinessLayer , sunt:

* Insert
* Delete
* getNumCols
* genPDF
* deleteClient
* deleteProduct
* okToBill(Order o): verifica daca o comanda indeplineste conditia pentru a i se face o chitanta: cantitatea de produs de pe comanda trebuie sa nu depaseasca cantitatea de produs din tabelul cu produse.Aceasta metoda realizeaza aceasta verificare.
* decreaseQuantity

Clasele “FailedBillPDF”,”PDFGeneratorClient”,”PDFGeneratorOrder”,”PDFGeneratorOrder”,”PDFGeneratorBill” sunt asemanatoare insa difera datele care sunt trimise catre constructorii lor.In interiorul acestora se afla metode precum : **addTableHeader(PdfPTable table)** care adauga un header tabelului, precum si **addTableRow(PdfPTable)** care adauga randuri tabelului.

In clasa **“FileParser”** se va trimite trimite fisierul din care se vor citi comenzi pentru a manipula baza de date.In cazul in care se detecteaza una din secventele : “ Insert client” sau “Insert product” urmate de nume sau de nume si adresa, atunci se va apela prin intermediul unei clase BLL din pachetul “business”, responsabila de tabela client/product, metoda de insert.Daca se detecteaza secventa “Delete client” sau “Delete product” precedate de nume sau de nume si adresa, se va apela prin intermediul unei clase BLL din pachetul “business”, responsabila de tabela client/product, metoda de delete.In cazul in care se intalneste cuvantul “Order” precedat de datele unei comenzi, se va crea o chitanta sub format PDF si comanda va fi inserata in tabela orders, acest lucru intamplandu-se doar in cazul in care metoda “okToBill” apelata cu comanda respectiva indica faptul ca acesteia i se poate face o chitanta.Dupa ce comanda este inserata, se va scadea din tabela de produse, cantitatea de produs a comenzii respective pentru produsul respectiv al comenzii.

In cazul in care se intalneste una din secventele “Report client”,”Report product” sau “Report order” se va apela metoda “genPDF” din clasa BLL corespunzatoare din pachetul “business”.

**5.TESTARE**

Testarea se face prin intermediul documentelor PDF. Se verifica daca datele introduse sunt corecte si daca ce s-a returnat corespunde cu ceea ce se afla in tabelele din baza de date.

**6.CONCLUZII SI DEZVOLTARI ULTERIOARE**

Ca si dezvolatre ulterioare a acestui proiect se poate crea o interfata grafica pentru a introduce datele clientilor, ale produselor si ale comenzilor.

**7.BIBLIOGRAFIE**

<https://stackoverflow.com/>

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