



TEHNICI DE PROGRAMARE

DOCUMENTATIA TEMEI 3

MANAGEMENTUL COMENZILOR

Hulea Andrei-Florin
Grupa 30225

Cuprins:**1. Obiectivul temei**

-Se va prezenta obiectivul principal al temei printr-o fraza si un tabel sau o lista cu obiectivele secundare. Obiectivele secundare reprezinta pasii care trebuie urmati pentru indeplinirea obiectivului principal. Fiecare obiectiv secundar va fi descris si se va indica in care capitol al documentatiei va fi detaliat.

2. Analiza problemei

-Modelare, scenarii, cazuri de utilizare Se va prezenta cadrul de cerinte functionale formalizat si cazurile de utilizare ca si diagrame si descrieri de use-case. Descrierile use-case-urilor se vor face sub forma unui flow-chart ori sub forma unei liste continand pasii executiei fiecarui use-case.

3. Proiectare

-Se va prezenta proiectarea OOP a aplicatiei, diagramele UML de clase si de pachete, structurile de date folosite, interfetele definite si algoritmi folositi (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

4. Implementare

-Se va descrie fiecare clasa cu campurile si cu metodele importante. Se va descrie implementarea interfetei utilizator.

5. Rezultate

-Se vor prezenta scenariile pentru testare cu Junit sau alt framework de testare.

6. Concluzii

-Se vor prezenta concluziile, ce s-a invatat din tema, dezvoltari ulterioare.

7. Bibliografie

-Se vor mentiona resursele bibliografice care au fost folosite pentru dezvoltarea temei

1. Obiectivul temei

Obiectivul acestei teme este implementarea unei aplicații OrderManagement care procesează comenzile clienților pentru un depozit. Se folosesc baze de date relationale pentru a stoca produsele, clienții și comenzile. De asemenea, aplicația trebuie să fie structurată pe pachete folosind o arhitectură stratificată. Și ar trebui să urmeze minimal următoarele clase:

- Clasele model – modelele de date ale aplicației ;
- Clasele pentru logica de afaceri – implementează logica aplicației ;
- Clasele de prezentare – implementează input-ul / output-ul user-ului ;
- Clasele pentru acces de date – implementează accesul la baza de date ;

Aplicația trebuie să permită procesarea comenzilor dintr-un fișier text dat ca argument, să efectueze operațiile cerute, să salveze modificările efectuate în baza de date și să genereze rapoarte în format pdf. Alte clase și pachete pot fi adăugate pentru a implementa funcționalitatea completă a aplicației.

De asemenea, trebuie implementat și un parser pentru citirea comenzilor din fișier în layer-ul de prezentare (în loc de interfața grafică standard) și un generator de fișiere pdf care să genereze rapoartele, conform următoarelor exemple :

- Adăugarea unui client în baza de date ;
- Stergerea unui client din baza de date ;
- Adăugarea unui produs în baza de date ;
- Stergerea unui produs din baza de date ;
- Crearea unei comenzi pentru client ;
- Generarea de rapoarte pentru clienți, produse și comenzi;

Obiectivul acestei teme este implementarea unei aplicații care simulează mai multe cozi, scopul acesteia fiind minimizarea timpului de așteptare pentru clienți și determinarea unui timp mediu de așteptare

Obiective secundare:

- Dezvoltarea de use-case-uri
- Alegerea corectă a structurilor de date
- Impartirea pe clase
- Dezvoltarea algoritmilor
- Implementarea soluțiilor
- Testarea programului

Use case-uri

Programul va fi apelat din linia de comanda cu ajutorul fisierului .jar creat . In primul rand , trebuie sa ne aflam in directorul corect pentru a apela programul sau sa punem calea relativa a argumentelor fata de directorul in care ne aflam . De asemenea dupa ce introducem 'java -jar' , numele jar-ului urmat de argumentul fisierului de intrare in care avem datele dupa care generam clientii, produsele, comenzile si rapoartele . In fisierul de intrare vom avea comenzile ce trebuie executate (acestea pot fi citite si efectuate de catre program datorita parser-ului) . In cazul in care user-ul nu introduce valori corecte pentru argumente programul nu va rula.

Proiectare

Pentru realizarea acestui proiect am folosit JDBC si Pdf file writer. JDBC (Java Database Connectivity) este o interfata standard SQL de acces la baze de date. JDBC este constituita dintr-un set de clase si interfete scrise în Java, furnizând mecanisme standard pentru proiectantii aplicatiilor de baze de date. Pachetul care ofera suport pentru lucrul cu baze de date este java.sql. Folosind JDBC este usor sa transmitem secvente SQL catre baze de date relationale. Cu alte cuvinte, nu este necesar sa scriem un program pentru a accesa o baza de date. Este de ajuns sa scriem un singur program folosind API-ul JDBC si acesta va fi capabil sa trimita secvente SQL bazei de date dorite. O conexiune (sesiune) la o baza de date reprezinta un context prin care sunt trimise secvente SQL si primite rezultate. Intr-o aplicatie pot exista mai multe conexiuni simultan la baze de date diferite sau la aceeasi baza. O data facuta conectarea, se poate folosi obiectul Connection rezultat pentru a se crea un obiect de tip Statements, cu ajutorul caruia putem trimite secvente SQL catre baza de date.

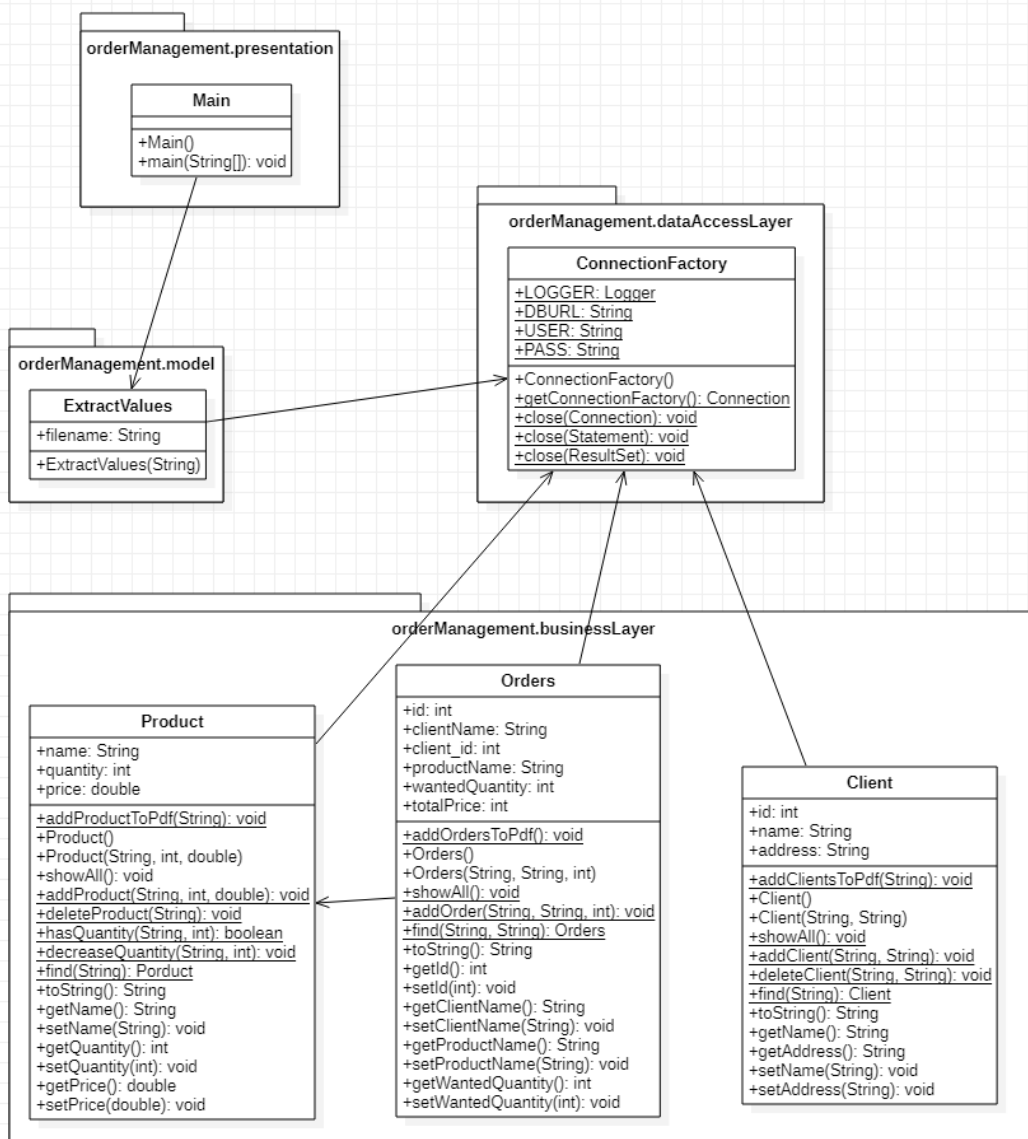
Structuri de date

Structura de date principala pe care am folosit-o este ArrayList-ul pentru a stoca datele preluate din fisier care trebuie parsate. ArrayList-ul de stringuri format in va contine pe fiecare pozitie cate un rand citit din fisierul de intrare. Dupa aceasta fiecare element al ArrayList-ului de stringuri este impartit cu ajutorul functiei split() si niste conditii de if pentru a se putea executa operatiile cerute in fisier.

Ex: ArrayList <String> data = new ArrayList <~> () ;



Diagrama de clase



2. Implementare

Clasa Main

În clasa main avem metoda din ExtractValues de extragere și parsare a valorilor din fișier și pe baza acestor valori extrase, cu ajutorul metodelor din clasele Client, Product și Orders, se vor efectua operațiile pe baza de date cerute în fișier și report-urile vor fi scrise în fișiere de tip pdf.

De asemenea, cu ajutorul .jar-ului creat putem apela programul din linia de comandă. Prima oară trebuie să ne asigurăm că ne aflăm în directorul corect sau inserăm argumentele cu tot cu calea lor relativă pentru fișierul de intrare. Dacă ne aflăm în fișierul în care se află și jar-ul și fișierele, următoarea linie trebuie scrisă în linia de comandă :

Ex : java -jar JarName.jar input_file.txt

Clasa ExtractValues

Această clasă o vom folosi pentru a extrage valorile necesare rularii programului din fișierul text dat ca și argument. În această clasă se extrag valorile din fișierul de intrare linie cu linie după care se separă fiecare linie cu ajutorul funcției split() și se execută operațiile cerute în urma identificării acestora cu ajutorul condițiilor.

Extragerea valorilor se face cu ajutorul clasei File pentru a selecta fișierul dorit și clasei Scanner care ne ajută să separăm liniile din fișier pentru a le împărți în valorile dorite și atribui variabilelor dorite.

Exemplu extragere:

```
File f = new File(this.filename);
Scanner sc = new Scanner(f);

while(sc.hasNextLine()){
    data = data + " " + sc.nextLine();
}
```

Clasa ConnectionFactory

În această clasă definim în mod static realizarea unei conexiuni la baza noastră de date astfel încât să se realizeze o singură conexiune, printr-un obiect 'Singleton'. Clasa conține numele driver-ului, numele bazei de date la care ne conectăm (dburl), username-ul și parola pentru accesul la serverul MySQL. De asemenea mai avem și metode pentru crearea unei conexiuni, returnarea unei conexiuni, închiderea unei conexiuni, statement sau result set.

Ex:

```
Connection con = ConnectionFactory.getConnection();
PreparedStatement ps = con.prepareStatement( "select * from client ;" );
ResultSet rs = ps.executeQuery() ;
```

(De asemenea , acestea trebuie puse intr-un bloc de try-catch)

Clasa Client

Dupa cum se poate deduce din numele acestei clase , in aceasta clasa vom defini obiectul de tip client si operatiile cu clienti din baza de date . Clientii sunt definiti de nume si adresa . De asemenea , tabelul client este in relatie de many to 1 cu tabelul orders . Contine metode pentru adaugare, stergere, cautare ,scriere in pdf cat si gettere si settere . Metodele implementate sunt urmatoarele :

public static void addClientsToPdf(String path) - aceasta metoda scrie intr-un tabel in pdf clientii cu toate datele acestora sub forma unui tabel cu ajutorul PdfWriter . Conexiunea la baza de date se realizeaza cu ajutorul clasei ConnectionFactory cu metoda ConnectionFactory.getConnection() ;

public Client(String name, String address) – constructor pentru un obiect de tip client care ii seteaza numele si adresa ;

public static void showAll() – metoda care afiseaza toti clientii prezenti in tabelul din baza de date cat si datele acestora . Conexiunea la baza de date se realizeaza cu ajutorul clasei ConnectionFactory cu metoda ConnectionFactory.getConnection() ;

public static void addClient(String nameAux, String addressAux) – metoda care adauga un client in tabelul din baza de date avand ca parametrii numele si adresa noului client . Conexiunea la baza de date se realizeaza cu ajutorul clasei ConnectionFactory cu metoda ConnectionFactory.getConnection() ;

public static void deleteClient(String nameAux) – metoda care sterge un client din tabelul din baza de date avand ca argument numele acestuia . Numele clientului este suficient pentru stergerea acestuia din baza de date dar deoarece numele este cheie primara a tabelului si cheie straina in tabelul de orders, trebuie sa stergem valorile ce apartin de clientul respectiv si din orders. Conexiunea la baza de date se realizeaza cu ajutorul clasei ConnectionFactory cu metoda ConnectionFactory.getConnection() ;

public String toString() – metoda folosita pentru verificarea corectitudinii codului ce returneaza datele unui client sub forma de string ;

public String getName() - returneaza numele clientului ;

public String getAddress() – returneaza adresa clientului ;

public String setName(String name) – seteaza numele clientului ;

public String setAddress(String address) – seteaza adresa clientului ;

Clasa Product

In mod evident , in aceasta clasa se vor efectua operatiile cu produsele prezente in baza de date . Un produs este definit prin nume de tip String , cantitate de tip intreg si pret de tip double . Se vor defini metodele pentru adaugare, stergere, afisare si scriere in fisierul pdf de produse . Metodele implementate sunt urmatoarele :

public static void addProductToPdf(String path) – metoda care scrie un produs in fisierul de tip pdf . Are ca argument adresa fisierului unde va fi scris produsul . Scrierea in fisier se realizeaza cu ajutorul PdfWriter iar conexiunea la baza de date se realizeaza cu ajutorul clasei ConnectionFactor cu metoda getConnection() ;

public Product(String name, int quantity, double price) – constructor pentru obiectul de tip produs care ii seteaza numele, cantitatea si pretul ;

public static showAll() - metoda care afiseaza toate produsele prezente in tabelul din baza de date cat si datele acestora . Conexiunea la baza de date se realizeaza cu ajutorul clasei ConnectionFactory cu metoda ConnectionFactory.getConnection() ;

public static void addProduct(String nameAux, int quantityAux, double priceAux) – metoda care adauga un nou produs in baza de date avand ca argumente numele, cantitatea si pretul produsului . Conexiunea la baza de date se realizeaza cu ajutorul clasei ConnectionFactor cu metoda getConnection() ;

public static void deleteProduct(String productName) – metoda care sterge un produs din baza de date avand ca argument numele acestuia. Deoarece numele produsului este cheie primara in acest tabel si cheie secundara in tabelul de orders inainte sa stergem din acest tabel trebuie sa stergem din tabelul orders valorile ca apartin de acest produs . Conexiunea la baza de date se realizeaza cu ajutorul clasei ConnectionFactor cu metoda getConnection() ;

public static boolean hasQuantity(String productName,int wantedQuantity) – metoda care returneaza true daca produsul cautat are stocul mai mare decat wantedQuantity si 0 daca acesta are stocul mai mic. Conexiunea la baza de date se realizeaza cu ajutorul clasei ConnectionFactor cu metoda getConnection() ;

`public static void decrease Quantity(String productName, int wantedQuantity)` – metoda care la adăugarea unui nou order în baza de date verifică dacă produsul respectiv are destule iteme în stoc. Dacă stocul este suficient de mare, cantitatea din order va fi scăzută din stoc iar în caz contrar order-ul nu se va mai realiza deoarece stocul este insuficient. Conexiunea la baza de date se realizează cu ajutorul clasei `ConnectionFactory` cu metoda `getConnection()` ;

`public static Product find(String productName)` – returnează produsul căutat ;

`public String toString()` – returnează produsul sub formă de string ;

Clasa Orders

În clasa `Orders` se vor efectua toate comenzile și operațiile necesare pentru adăugarea acestora. Un obiect de tip `Order` este definit prin id, numele clientului, numele produsului și cantitatea dorită. Conexiunea se va realiza la fel ca la clasa `Product` și `Client` cu ajutorul `ConnectionFactory`. Metodele implementate sunt următoarele :

`public static void addOrdersToPdf(String path)` – metoda care scrie toate comenzile prezente în baza de date sub formă de tabel într-un fișier pdf.

`public Orders(String clientName, String productName, int wantedQuantity)` – constructorul clasei care setează numele produsului și cumpărătorului cât și cantitatea dorită ;

`public static void showAll()` – metoda care afișează toate comenzile prezente în baza de date ;

`public static void addOrder(String clientNameAux, String productNameAux, int wantedQuantityAux)` – metoda care adaugă o nouă comandă în baza de date dacă stocul produsului dorit permite acest lucru. În caz contrar comanda nu se va mai realiza datorită stocului insuficient ;

`public static void find()` – metoda returnează comanda căutată ;

`public static void toString()` - metoda care returnează comanda sub formă de string ;

3. Testare

Testarea a fost realizata in mod practic cu ajutorul exemplelor date .

4. Concluzii si dezvoltari ulterioare

In urma acestui proiect am reusit sa imi dezvolt intelegerea si abilitatile de a lucra cu baze de date in java.

O posibila imbunatatire poate fi adusa la capitolul eficientei pentru baze de date de dimensiuni mult mai mari prin folosirea eficienta a metodei de cascada sau prin folosirea unor metode de introducere si stergere mai eficiente.

Bibliografie

<https://www.javatpoint.com/example-to-connect-to-the-mysql-database>

<https://www.mysql.com>

<https://howtodoinjava.com/library/read-generate-pdf-java-itext/>

<https://en.wikipedia.org/wiki/Parsing>