FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

DOCUMENTATIE

la disciplina

Tehnici de Programare

ORDER MANAGEMENT

Irimus Ileana-Maria, grupa 30223

An academic 2019 – 2020

Cuprins

1. Obiectivul temei 2

1.1 Obiective secundare 2

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

3. Proiectare (diagrame UML, structuri de date, proiectare clase, packages) 3

4. Implementare 4

5. Rezultate 6

6. Concluzii 9

7. Bibliografie 9

1. Obiectivul temei

Obiectivul principal al acestei teme este proiectarea si implementarea unei aplicatii OrderManagement pentru procesarea comenzilor clientilor pentru un depozit. Bazele de date relationale sunt utilizate pentru a stoca produsele, clientii si comenzile.

1.1Obiective secundare

Modelarea in clase si pachete – presupune impartirea proiectului in anumite pachete fiecare pachet fiind mai departe fragmentat in clase cu denumiri sugestive. Acest obiectiv va fi prezentat in detaliu in capitolul 3.

Alegerea structurilor de date - vor fi prezentate structurile utilizate in realizarea proiectului. Acest obiectiv va fi prezentat in detaliu in capitolul 3.

Implementarea claselor – se va descrie fiecare clasa cu metodele si campurile corespunzatoare fiecareia. Acest obiectiv va fi prezentat in detaliu in capitolul 4.

Testarea - se vor prezenta scenariile pentru testare din cmd. Acest obiectiv va fi prezentat in detaliu in capitolul 5.

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

Cerinte functionale

Luati in considerare o aplicatie OrderManagement pentru procesarea comenzilor clientilor pentru un depozit. Bazele de date relationale sunt utilizate pentru a stoca produsele, clientii si comenzile. In plus, aplicatia trebuie sa fie structurata in pachete folosind o arhitectura stratificata si utilizeaza (minim) urmatoarele clase:

• clasele de modele - modelele de date ale aplicatiei;

• clasele de logica de afaceri - implementeaza logica aplicatiei;

• clasele de prezentare - implementeaza intrarea / iesirea utilizatorului;

• clasele de acces la date - implementeaza accesul la baza de date.

Aplicatia trebuie sa permita procesarea comenzilor dintr-un fisier text dat ca argument, sa efectueze operatiunile solicitate, sa salveze datele din baza de date si sa genereze rapoarte in format pdf. Alte clase si pachete pot fi adaugate pentru a implementa functionalitatea completa a aplicatiei. De asemenea, implementati un parser pentru a citi comenzile in pachetul Presentation (in locul utilizatorului grafic standard interfata) si un generator de fisiere pdf pentru a genera rapoartele.

3. Proiectare (diagrame UML, structuri de date, proiectare clase, packages)

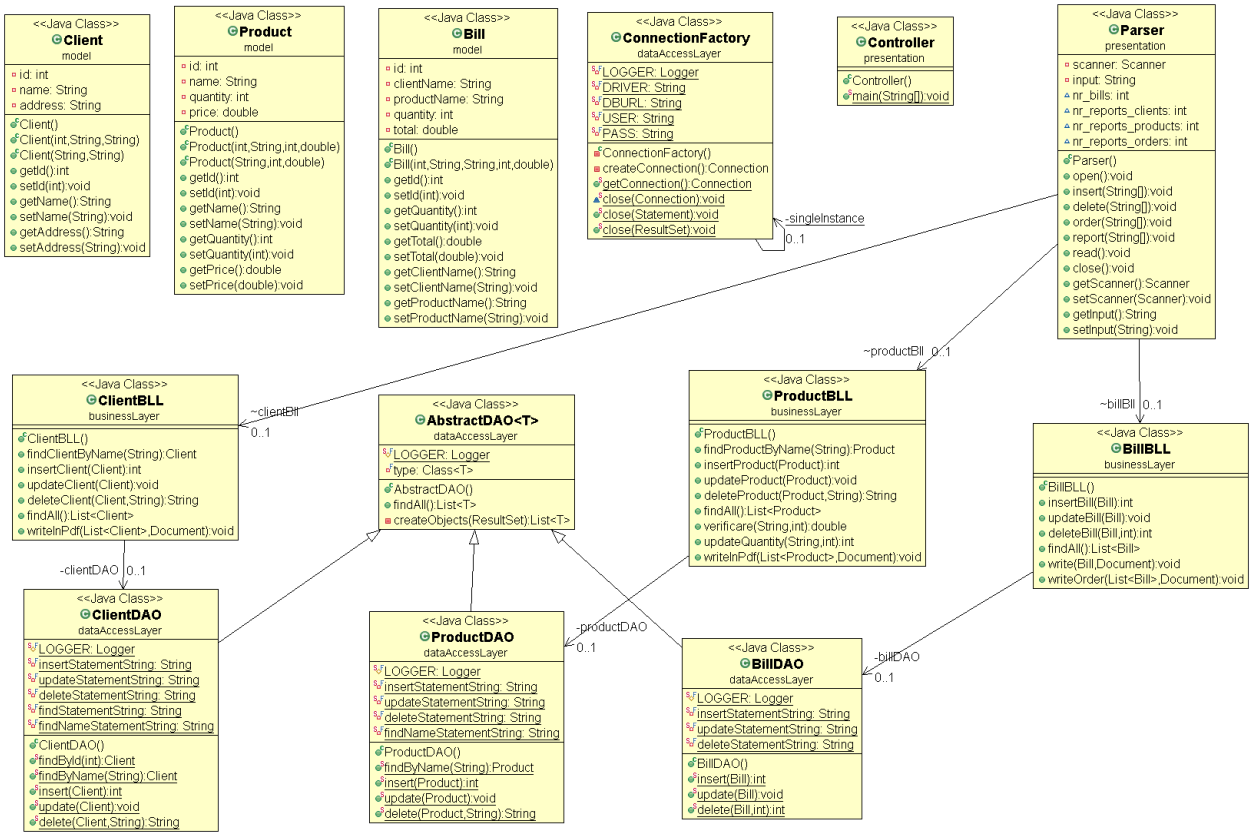

Diagrama UML de clasa

Aceasta aplicatie este proiectata dupa conventia specifica programarii orientate pe obiect, avand model, controller si parser (in loc de View), logica programului fiind impartita in trei elemente interconectate. Scopul acestei divizari este de a separa reprezentarea interna de modul/informatia prezentata utilizatorului. Astfel, modelul gestioneaza direct datele de intrare ale utilizatorului primite de la controller. Parser se ocupa cu convertirea datelor de intrare din fisierul text. Controllerul primeste informatia, o valideaza, iar mai apoi o transmite modelului. In acest caz modelul este reprezentat de clasele Client, Product si Bill. Remarcam existenta altor doua pachete dataAccessLayer si businessLayer pentru implementarea logicii aplicatiei, respective pentru implementarea accesului la baza de date mySQL.

Pentru realizarea acestei teme am decis sa utilizez List ca structura de date principala. Interfata List Java, java.util.List, reprezinta o secventa ordonata de obiecte. Elementele continute intr-o lista Java pot fi introduse, accesate, iterate si eliminate in functie de ordinea in care apar intern in lista Java. Ordinea elementelor este motivul pentru care această structura de date este numita List. Fiecare element dintr-o lista Java are un index. Interfata Java List este o interfata Java standard si este un subtip al interfetei Java Collection, adica Lista moștenește de la Collection. Fiind un subtip de colectie, toate metodele din interfata Collection sunt de asemenea disponibile in interfata List. Deoarece List este o interfata, trebuie sa instantanezi o implementare concreta a

interfetei pentru a o folosi. Se poate alege intre urmatoarele implementari de lista in API-ul Java Collections:

- java.util.ArrayList;
- java.util.LinkedList;
- java.util.Vector;
- java.util.Stack.

4. Implementare

Proiectul are in componenta sa 4 pachete: presentation, model, dataAccessLayer si businessLayer.

Pachetul model contine cele trei clase: Client, Product si Bill (echivalentul clasei Order, dar din cauza faptului ca “order” este key word in mySQL am fost nevoita sa o redenumesc).

Clasa Client descrie obiectul client, descris de un int id, un String name si un String address. Pe langa cei 3 constructori, aceasta clasa include setters si getters pentru fiecare dintre campurile descrise mai sus.

Clasa Product descrie obiectul produs, descris de un int id, un String name, un int quantity, reprezentand cantitatea de produse(in bucati) introdusa si un double price, reprezentand pretul unui produs/buc. Pe langa cei 3 constructori, aceasta clasa include setters si getters pentru fiecare dintre campurile descrise mai sus.

Clasa Bill descrie obiectul comanda/factura, descris de un int id, un String clientName, reprezentand numele clientului care face comanda, un String productName, reprezentand numele produsul comandat, un int quantity, reprezentand cantitatea de produs comandata(in bucati) si un double total, reprezentand suma totala a comenzii ce va aparea pe factura. Pe langa cei 2 constructori, aceasta clasa include setters si getters pentru fiecare dintre campurile descrise mai sus.

Pachetul dataAccessLayer contine cele 5 clase: ClientDAO, ProductDAO, BillDAO, AbstractDAO si ConnectionFactory. Prin aceste clase este realizat accesul la baza de date mySQL, prin intermediul mai multor query:

- Insert: "INSERT INTO <nume_tabela>(<camp1>,<camp2> …)" + " VALUES (?,? …)";
- Update: "UPDATE <nume_tabela> SET camp1 = ?, camp2 = ? … WHERE campx=?";
- Delete: "DELETE FROM <nume_tabela> WHERE campx=?";
- Find: "SELECT * FROM <nume_tabela> WHERE campx = ?";

Clasa ClientDAO face legatura dintre obiectul Java Client si tabela client din mySQL. In aceasta clasa sunt implementate operatiile de baza realizate cu ajutorul unui String statement (un query). Astfel, inserarea unui client in tabela este realizata in metoda insert, actualizarea informatiilor despre un anumit client folosind numele acestuia este realizata in metoda update, iar stergerea din tabela a unui client folosind numele acestuia este realizata in metoda delete. Tot aici sunt implementate si doua metode de find, dupa id, respectiv dupa nume.

Clasa ProductDAO face legatura dintre obiectul Java Product si tabela product din mySQL. In aceasta clasa sunt implementate operatiile de baza realizate cu ajutorul unui String statement (un query). Astfel, inserarea unui produs in tabela este realizata in metoda insert, actualizarea informatiilor despre un anumit produs folosind numele acestuia este realizata in metoda update, iar stergerea din tabela a unui produs folosind numele acestuia este realizata in metoda delete. Tot aici este implementata si metoda de find dupa nume.

Clasa BillDAO face legatura dintre obiectul Java Bill si tabela bill din mySQL. In aceasta clasa sunt implementate operatiile de baza realizate cu ajutorul unui String statement (un query). Astfel, inserarea unui bill in tabela este realizata in metoda insert, actualizarea informatiilor despre un anumit bill folosind id-ul acestuia este realizata in metoda update, iar stergerea din tabela a unui bill folosind id-ul acestuia este realizata in metoda delete.

Clasa AbstractDAO implementeaza doua metode: createObject ce returneaza o lista de obiecte T, care primeste drept parametru un obiect de tipul ResultSet si ia informatiile din acest parametru si le pune intr-o lista care va fi utlizata ulterior; si metoda findAll care va returna toate campurile dintr-o tabela specificata.

Clasa ConnectionFactory realizeaza conexiunea propriu-zisa dintre proiectul java si baza de date mySQL. Aceasta clasa contine metode de creare a conexiune, de inchidere a conexiunii, a statementului, dar si de inchidere a obicetului de tip ResultSet declarat.

Pachetul businessLayer contine cele 3 clase: ClientBll, ProductBll si BillBll.

Clasa ClientBll contine o instanta de tip ClientDAO si apeleaza metodele din clasa ClientDAO de gasire a clientului dupa nume, inserarea unui client, actualizarea datelor unui client, stergerea din tabela a unui client, dar si metoda de gasire a tuturor clientilor din tabela. Tot aici este realizata si scrierea datelor din mySQL in documentul de tip pdf generat si afisarea unui mesaj de success.

Clasa ProductBll contine o instanta de tip ProductDAO si apeleaza metodele din clasa ClientDAO de gasire a produsului dupa nume, inserarea unui produs, actualizarea datelor despre un produs anume, stergerea din tabela a unui produs, dar si metoda de gasire a tuturor produselor din tabela. Pe langa aceste metode, se observa exista unor functii de verificare a cantitatii de produs din tabela, dar si de actualizare a cantitatii de produs daca este nevoie. Tot aici este realizata si scrierea datelor din mySQL in documentul de tip pdf generat si afisarea unui mesaj de success.

Clasa BillBll contine o instanta de tip BillDAO si apeleaza metodele din clasa BillDAO de inserare a unei comenzi noi, actualizarea datelor unei comenzi, stergerea din tabela a unei comenzi, dar si metoda de gasire a tuturor comenzilor din tabela. Tot aici este realizata si scrierea datelor din mySQL in documentul de tip pdf generat si afisarea unui mesaj de success. Daca dorim sa afisam un raport despre comenzi se apeleaza metoda writeOrder, in schimb daca dorim realizarea unei facturi, se apeleaza metoda write; diferenta dintre acestea 2 fiind afisarea sau nu a unui total al comenzii.

Pachetul presentation contine cele 2 clase: Parser si Controller.

Clasa Parser contine un obiect de tip Scanner scanner, un String input, dar si 3 instante de tipul ClientBll, ProductBll si BillBll. In aceasta clasa este realizata deschiderea fisierului text primit ca argument din linia de comanda, dar si citirea din acesta si, bineinteles, inchiderea fisierului. Se remarca prezenta a 4 metode: insert, delete, order si report. Metoda insert presupune extragerea din fisierul text a instructiunii de inserare si informatii despre obiectul ce trebuie adaugat in baza de date: pentru client – nume, adresa, iar pentru produs – nume, cantitate si pret. Metoda delete presupune extragerea din fisierul text a instructiunii de stergere si informatii despre obiectul ce trebuie sters din baza de date: numele atat pentru client cat si pentru produs, in cazul in care acestea se afla deja in tabela. Metoda order presupune extragerea din fisierul text a instructiunii de comanda si informatii despre persoana care face comanda (numele), produsul comandat si cantitatea de produs comandat; astfel de fiecare data cand se efectueaza o comanda, este realizata si o facture noua intr-un document pdf, astfel: daca cumva stocul este insuficient sau daca produsul nu se afla in tabela, se va genera un document pdf cu mesajul “Stoc insuficient/Nu exista produsul”; altfel, totalul facturii va fi calculat in functie de cantitatea de produs si pretul/bucata si va fi scris in pdf, impreuna cu detalii comenzii (clientul, numele produsului). Metoda report presupune extragerea din fisierul text a instructiunii de creare a unui raport si obiectul despre care trebuie facut raportul: client, produs, comanda; din nou se va genera cate un document pdf pentru fiecare raport si va include intreaga tabela (client, product sau bill) in stare curenta.

Clasa Controller contine metoda main cu o instanta de tip Parser pentru a deschide, citi sim ai apoi a inchide fisierul text primit ca input din linia de comanda.

5. Rezultate

Testarea este realizata din Command Line prin comanda “java -jar Assignment3.jar <calea fisierului cu datele de intrare> ”.

Assignment3.jar este fisierul jar al acestui proiect. Un JAR (arhiva Java) este un format de fisier de pachete utilizat in mod obisnuit pentru a aglomera multe fisiere de clasa Java si metadate si resurse asociate (text, imagini etc.) intr-un singur fisier pentru a distribui software-ul aplicatiei sau bibliotecile pe platforma Java. In cuvinte simple, un fisier JAR este un fisier care contine versiunea comprimata a fisierelor .class, fisiere audio, fisiere imagine sau directoare. Chiar și software-ul WinZip poate fi folosit pentru a extrage continutul unui .jar. Asadar, le puteti utiliza pentru sarcini precum compresia de date fara pierderi, arhivarea, decompresia si dezambalarea arhivei.

De asemenea, rezultatele vor fi afisate sub forma unor rapoarte de tip pdf in functie de datele din fisierul text de intrare. iText si PdfBox sunt biblioteci Java utilizate pentru crearea si manipularea fisierelor pdf. Desi rezultatul final al bibliotecilor este aceeasi, ele functioneaza in mod diferit. iText este o biblioteca Java creata initial de Bruno Lowagie care permite crearea de PDF, citirea PDF-ului si manipularea acestuia. iText are o structura ierarhica. Cea mai mica unitate de text este un „Chunk”, un String cu un font predefinit. O „frază”(Phrase) combină mai multe bucati si permite definirea spatiului de linie. „Paragraful” este o subclasa

a „Frazei” si permite definirea mai multor atribute de layout, de ex. margini. Clasa „Anchor” este o subclasa a „Paragrafului” si serveste ca baza pentru hiperlink-uri in PDF-ul generat. Daca doriti sa utilizati iText este nevoie descarcarea unui jar iText.

Am atasat mai jos cateva dintre documentele pdf rezultate:

Order report

Client Name	Product Name	Quantity
Luca George	apple	5
Luca George	lemon	5

Bill

Client Name	Product Name	Quantity	Total
Luca George	apple	5	5.0

Client report

Name	Adress
Ion Popescu	Bucuresti
Luca George	Bucuresti
Sandu Vasile	Cluj-Napoca

Product report

Name	Quantity	Price
apple	40	1.0
orange	40	1.5
lemon	70	2.0

6. Concluzii

In concluzie, aceasta aplicatie poate fi una foarte utila si eficienta pentru procesarea comenzilor clientilor pentru un deposit.

In urma realizarii acestei teme, am invat sa realizez conexiunea dintre un cod Java cu clasele si pachetele sale si o baza de date mySQL, dar si cu argumentele trimise din linia de comanda. O posibila dezvoltare ulterioara ar fi posibilitatea adaugarii mai multor comenzi pe aceeasi factura a aceluiasi client.

7. Bibliografie

Connect to MySql from a Java application
- *http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/*
- *http://theopentutorials.com/tutorials/java/jdbc/jdbc-mysql-create-database-example/*

Layered architectures
- *https://dzone.com/articles/layers-standard-enterprise*

Reflection in Java
- *http://tutorials.jenkov.com/java-reflection/index.html*

Creating PDF files in Java
- *https://www.baeldung.com/java-pdf-creation*

JAVADOC
- *https://www.baeldung.com/javadoc*

SQL dump file generation
- *https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html*
- *https://dev.mysql.com/doc/refman/5.7/en/using-mysqldump.html*