

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

**ORDER MANAGEMENT**

Documentatie Tema 3

Runcan Nicoleta

Grupa 30229

An 2019-2020

**Cuprins**

**1.Obiectivul temei**

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

**Analiza problemei**

**Scenariu de utilizare**

**3.Proiectare**

**Decizii de proiectare**

**Diagrama de clase**

**Structuri de date**

**Interfete**

**4.Implementare**

**5.Rezultate**

**6.Concluzii**

**7.Bibliografie**

**1.Obiectivul temei**

Obiectivul acestei teme a fost de a crea o aplicatie care menegereaza comenzile clientilor pentru un depozit. Mai exact ce se intampla in momentul in care un client doreste sa efectueze o anumita comanda si cum trebuie gestionata aceasta comanda, cum se gestioneaza aprovizionarea stocurilor de produse si inca multe alte lucruri. Astfel, trebuie verificate mai multe aspecte: daca sunt suficiente produse in stoc din cele cerute de catre clientul respectiv, daca exista produsul cerut. La fel ca si in cadrul celorlalte teme, obiectivul principal este de a imbunatati abilitatile studentilor de a lucra cu diferite paradigme ale programarii orientate pe obiect si de a asimila cat mai multa informatie. De asemenea aceasta tema a urmarit si familiarizarea legarii unei aplicatii java la o baza de date. Baza de date relationala este folosita pentru a stoca produsele, clientii si comenzile. In plus, aplicatia este structurata si pe pachete. Totodata, aceasta tema a presupus citirea din fisiere text, dar si scriere rezultatelor aplicatiei in fisiere pdf.

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

Aproape zilnic sunt efectuate o multitudine de comenzi, iar viata oamenilor, in momentul de fata,nu ar putea fi conceputa fara efectuarea comenzilor necesare. La o prima impresie, efectuarea unei comenzi pare o activitate simpla, dar patruzand mai in adancimea problemei in fapt, observam ca pentru gestionarea unui depozit si a comenzilor care sunt trimise catre acest depozit, este necesar sa se tina cont de mult mai multe aspect.

Astfel, aplicatia noastra ar trebui sa fie capabila sa proceseze un sir de cerinte dintr-un fisier text, sa realizeze operatiile citite de acolo, si sa genereze rapoarte intr-un format pdf. De asemenea in momentul in care in acel fisier text se regaseste o cerinta care sa presupuna inserarea unei comenzi, ar trebui generata o facture pentru acea comanda care sa contina numele clientului care a cerut comanda, produsul pe care l-a comandat si de asemenea pretul total pe care trebuie sa-l plateasca pentru acea comanda. Dar sa nu uitam de faptul ca unul dintre lucrarile care trebuie analizate cu atentie este faptul ca aceasta aplicatie este legata strans de o bza ade date, asupra careia trebuie sa se efectueze modificari in functie de diferitele cerinte din acel fisier text. In aceasta baza de date se vor pastra in 4 tabele toate datele cu care lucreaza aceasta plicatie in fiecare moment. Aceste 4 tabele sunt: tabelul Client, in care o sa fie stocate toate datele despre clienti si mai exact: un id unic pentru fiecare client, numele lui si adresa; tabelul Product in care se pastreaza numele unic al produsului, si pretul acestuia ; tabelul Stock in care se pastreaza stocul fiecarui produs, iar cel din urma dar si cel mai important este tabelul Order, in care se stocheaza comenzile; acesta este format din 4 coloane: id-ul unic al comenzii, numele clientului care efectueaza comanda, numele produsului comandat si de asemenea cantitatea in care doreste sa comande produsul respective.

**Scenariu de utilizare:**

Scenariul initial de utilizare, presupunea citirea unor date dintr-un fisier text, mai exact a unor comenzii(cerinte), care sunt dupa urmatorul format: “Insert client: Ion Popescu, Bucuresti”, sau “Report client” sau “Delete Product: peach”, mai pe scurt, sunt cerinte care presupun inserarea clientilor, a produselor si a comenzilor in baza de date, sau stergerea acestora, sau efectuarea acelor rapoarte de tip pdf cu starea tabelului in momentul executarii cerintei respective, mai exact tabelul corespunzator cerintei(de exemplu pentru comanda “Report client” se va genera un raport pdf care contine un tabel cu toate datele care se gasesc in acel moment in tabelul Client din baza de date. Aceste rapoarte nu se vor crea doar pentru tabelul Client, ci pentru toate tabelele daca in fisierul text din care se citesc aceste cerinte va exista o comanda care sa presupuna acest lucru: spre exemplu(Report Client sau Report Product sau Report Order). In acest fisier text, nu este un numar finit de cerinte, ci poate sa oscileze pana la un numar foarte mare. In momentul in care acesta este deschis se citeste fiecare cerinta pe rand si se executa succesiv fiecare dintre ele, adica in momentul in care a fost citita prima, in functie de ce presupune aceasta, se vor apela metodele necesare efectuarii ei, iar acest lucru se va repeta pana cand nu va mai exista nicio cerinta in fisier.

In urma realizarii si a fisierului .jar; utilizarea acestei aplicatii presupune scrierea in Command Prompt, a urmatoarei linii de comanda cu urmatorii parametrii **:**

java –jar PT2020\_Group\_FirstName\_LastName\_Assignment\_2.jar in.txt, unde in.txt va putea fi inlocuit cu numele fisierului care detine cerintele pe baza caruia aplicatia executa diferite operatii. Fisierul in care eu tin cerintele se numeste “file-in.txt”. Astfel aplicatia va rula, executand comenzile respective din fisier, fapt ce va genera modificari asupra tabelolor din baza de date, dar si alte operatii asupra acestora, si in acelasi timp, creara facturilor in format pdf pentru comenzi si a raporturilor pentru tabele in functie de tipul cerintelor din fisier.

**3.Proiectare**

**Decizii de proiectare:**

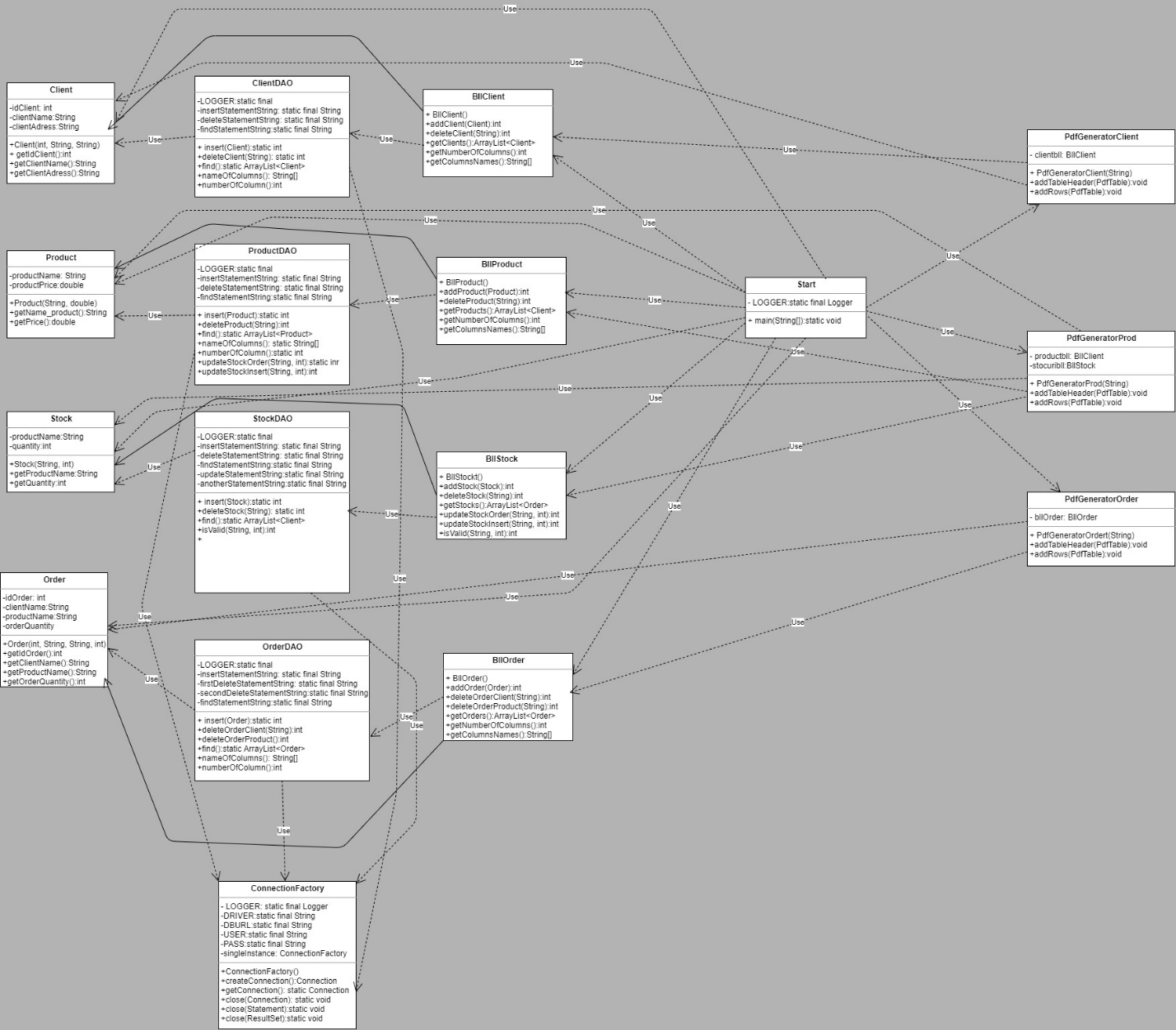
Atat pentru proiectarea, dar si pentru implementarea acestei aplicatii, este nevoie de cateva cunostinte minime despre cum se lucreaza cu MySql Workbench, in care se se creeaza baza de date, dar si cateva cunostinte minime despre lucrul cu baze de date, mai exact cum se creeaza una, cum se creeaza un tabel intr-o baza de date, cum se face o inserare intr-un tabel, cum se face o sterge intr-un tabel sau cum se fac difertite query-uri asupra tabelelor respective. Pentru a putea crea o aplicatie Java care sa lucreze cu aceasta baza de date, este nevoie sa se creeze legatura dintre aplicatie si aceasta; astfel prima clasa care trebuie implementata este cea in care se creeaza aceasta legatura. In programul meu, aceasta clasa se numeste ConnectionFactory si se afla in pachetul dataAccessLayer alaturi de alte patru clase:clientDAO, productDAO, stockDAO si orderDAO, care la fel reusesc sa insereze sau sa stearga date(diferite operatii asupra tabelelor bazei de date), prin intermediul acestui program care se implementeaza in Java. De asemenea am decis sa folosesc “Layerd Arhitecture”, adica toata clasele trebuie sa fie impartite in cel putin 4 pachete. Am pachetul Model, businessLayer, dataAccessLayer-conexiunea la baza de date, Presentation-se ocupa cu datele de intrare/iesire. De asemenea programul meu, mai contine si un alt pachet Start, in care se afla clasa main cu metoda Start, in care eu am ales sa fac citirea acelor cerinte din fisier si de asemenea sa fac parsarea lor, si in functie de fiecare cerinta sa apelez metodele corespunzatoare efectuarii cerintei curente.

**Structuri de date:**

Singura structura de date folosita este ArrayList-ul, care faciliteaza retinerea datelor despre fiecare tabel in parte. Este folosit, in cazul in care efectuez o interogare ca si aceasta “SELECT \* FROM tabel” care imi returneaza toate datele despre acel tabel, toate randurile, iar cu acele date o sa creez cate un obiect specific pentru fiecare rand din tabelul respective, pe care o sa-l adaug inntr-un ArrayList

**Diagrama de clase:**

Diagrama de clase este folosita in modelarea orientate pe obiect pentru a descrie structura statica a sistemului, modului in care este el structurat. Mai jos este schitata diagram de clase a acestui program.

****

**Interfete:**

Nu a fost definita nicio interfata noua.

**4 Implementare:**

In acest capitol se vor descrie pachetele, clasele si metodele importante, componente a acestor clase, cu logica din spatele lor.

Pentru inceput, o sa mentionez cateva detalii despre baza de date, cu care lucreaza aplicatia mea. Aceasta poarta numele “warehouse” si contine urmatoarele patru tabele: Primul tabel se numeste Client si contine 3 coloane:idClient-cheie primara, clientName, clientAdress; urmatorul este Product si contine 2 coloane: productName—cheie primara si productPrice; al treilea se numeste Stock si are 2 coloane: productName—cheie primara si cheie straina pentru clientName din Product si de asemea Quantity; iar cel de-al patrulea tabel, “Order” are 4 coloane:idOrder-cheie primara, clientName, productName si orderQuantity.

**Pachetul Model:**

Contine patru clase**: Client, Product, Order si Stock**; clasele acestea sunt un corespodent al aplicatiei cu cele patru tabele din baza de date. Mai exact fiecare dintre ele contine un numar de variabile instanta egal cu numarul de coloane din fiecare tabel, si de asemenea numele acestor variabile instanta este foarte asemenator cu cel al numelor coloanelor fiecarui tabel corespunzator. De asemena fiecare dintre ele contine cate un constructor in care se intializeaza aceaste variabile instanta cu parametrii contructorului respective. La nivel de metode, singurele metode pe care le au aceste clase, sunt metodele mutatoare(getters) care returneaza valorile parametrilor.

**Pachetul dataAccessLayer**

Acest pachet contine 5 clase: **ConnectionFactory, clientDAO, productDAO, stockDAO, orderDAO,** prin aceste clase, cu ajutorul metodelor existente in ele se creeaza legatura cu baza de date despre care am vorbit, si de asemenea diferite modificari asupra tabelelor componente acesteia. Clasa **ConnectionFactory** creeaza legatura propriu-zisa cu baza de date. Despre celelalte 4 clase se poate spune ca sunt foarte identice, si contin apropae aceleasi metode, si apelarea lor va exercita aproximativ aceleasi rezultate, doar ca pe tabele diferite; si este foarte usor de sesizat, pe ce tabel va exercita opreatii fiecare dintre ele, deoarece le-am ales un nume sugestiv in functie de numele tabelelor create.Pentru fiecare dintre celelalte 4 clase, sunt declarate cateva variabile statice care sunt initializate cu, cate un Sql statement. Un exemplu ar fi: “INSERT INTO PRODUCT(productName, productPrice) VALUES(? ? ? ?). Un statement de insert regasindu-se in fiecare din cele 4 clase, la fel ca si unul de DELETE ca si acesta“DELETE FROM Client WHERE clientName=?”, dar si unul de select “SELECT \* From Order”, ca si acesta(la acesta schimbandu-se doar numele tabelului). De asmenea in fiecare dintre aceste clase este implementata o metoda care reuseste sa faca legatura dintre aceste Sql statement-uri si aplicatia Java, astfel incat rezultate lor asupra tabelelor sa fie cele dorite. Metoda de insert din cele patru clase are ca si parametru cate un obiect de tipul unei clase din Model corespunzatoare pentru fiecare clasa DAO, adica pentru insert-ul din clasa clientDAO, se va regasi un parametru de tipul Client .In aceasta metoda de insert se lucreaza cu insertStatement-ul, mai exact, seteaza valori pentru fiecare din semnele de intrebare care se regasesc in Statement; astfel se va insera un nou rand in tabel cu datele setate, iar in cazul in care nu se va putea efectua inserarea, un mesaj cu motivul pentru care nu s-a putut efectua inserarea va fi afisat. Pentru metoda de delete se procedeaza aproximativ la fel, adica fiecare delete trebuie sa se efectueze in functie de o anumita valoare a unei coloane, iar in aceasta metoda, de asemenea se seteaza aceasta valoare cu valoarea parametrului metodei, care pentru toate cele 4 clase este un String. O alta metoda comuna pentru toate cele 4 clase este cea care lucreaza cu statement-ul de “SELECT \*”, datele care se obtin in urma acestei operatii sunt pastrate intr-un obiect de tipul ResultSet, iar dupa alte instructiuni asupra acestui ResultSet, se va crea cate un obiect de tipul fiecarei clae din Model corespunzatoare clasei din DAO si se va adauga ArrayList-ului.

Alte doua metode commune celor 4 clase: una prin care se returneaza pentru fiecare cate un sir cu numele coloanelor tabelului corespunzator, iar cealata care returneaza numarul de coloane al tabelului corespunzator.

De specificat este faptul ca in clasa Order, se efectueaza doua metode de stergere, una in functie de numele clientului, iar cealalta in functie de numele prodului, deoarece, in momentul in care este ceruta o cerinta de stergere a unui client, dar acel client a solicitat o comanda, este necesara si stergerea acesteia, si de asemenea si atunci cand un produs trebuie sters, dar este facuta o comanda cu acel produs, trebuie sters si comanda respectiva.

Clasa stockDAO contine cateva metode necomune cu celelalte clase, cum ar fi doua metode de Update in care se lucreaza cu doua sql statement-uri de update. Mai exact un update al stocului care trebuie folosit in momentul in care este adaugat un produs cu acelasi nume, iar el nu nu trebuie inserat din nou in tabel, si nici nu s-ar putea acest lucru deoarece aceast camp este cheie primara, sa se faca doar un update la stocul acestuia, iar celalat este folosit pentru momentele in care se efectueaza o comanda, iar din stocul produsului comandat din depozit, trebuie scazuta valoarea cantitatii comenzii. Si de asemena mai contine o metoda isValid() care ne returneaza 0 sau 1 daca stringul dat ca parametru este echivalent cu vreu-un produs din tabelul Stock si daca valoarea cantitatii acestuia este mai mica sau mai mare decat valoarea int data ca si parametru. Aceasta metoda este folosita pentru cazul in care se efectueaza o comanda si trebuie sa stim daca exista sufieciente produse de acel tip in stoc pentru a efectua comanda.

**Pachetul businessLayer**: 4 clase**: BllClient, BllProduct, BllOrder, BllStock,** fiecare cu cate un cosntrucotr gol si cu cate un numar de metode egal cu clasa DAO corespunzatoare fiecareia, in care se apeleaza metodele de acolo.

**Pachetul Presentation: 3 clase de pdfGenerator** pentru client, produs si order. In aceste clase se creeaza raporturile pdf propriu-size corespunzatoare fiecarui tabel din baza de date. Aici se folosesc si metodele din DAO prin care se returnau numele coloanelor si numarul acestora, De mentionat aici este doar faptul ca pentru Report Product, tabelul nu este identic cu cel din baza de date, ci am mai adaugat o coloana pentru stock.

**Pachetul Start-** contine clasa Start cu metoda main. Eu aici am ales sa fac citirea din fisier, si parsarea acestuia. Aici se citeste fiecare comanda, dupa cum am mai explicat si la functionarea programului, se fac split-uri pentru a putea avea acces la datele care ne sunt necesare din comanda, iar in functie de cuvintele cheie de la inceputul cerintei citite din fisier se vor apela metodele implementate in celelate clase pentru a obine rezultate dorite. De mentionat ca insert-ul si delete-ul in Product este insotit de cel in Stock. De asemenea cum am mai zis si mai sus, aici se stabileste si cum se face stergerea, adica daca un client trebuie sters, dar are o comnda, aceasta trebuie stearsa, iar la fel se intampla si pentru produs. Un alt lucru de mentionat este ca tot aici, atunci cand trebuie apelata metoda de inserare a unei comenzi, generez si o facture a acesteia in format pdf, cu numele acesteia, produsul si pretul total, iar daca stocul produsului nu este suficient, afisez intr-un pdf un mesaj care sa spuna acest lucru.

**5.Rezultate:**

S-ar putea afirma ca rezultatele pe care la obtinem in urma implementarii unei aplicatii sunt cele mai importante, dar mai important este ca aceste rezultate sa fie cele asteptate. Pentru aceasta aplicatie, rezultate pot fi urmarite in tabelele din baza de date, dar de asemenea si in acele rapoarte pdf care o sa fie generate. In cele 4 tabele ale bazei de date, trebuie sa existe acele randuri pe care am dorit sa le inseram, dar bineinteles, din acestea trebuie sa lipseasca acele randuri pe care am dorit sa le stergem. De asemenea, pentru inserarile de produse cu acelasi nume, nu exista 2 randuri diferite cu acest produs, ci doar unul, pentru care in tabelul de Stock, cantitatea specifica acestui produs este insumarea cantitatilor celor doua inserari. De asemenea, pentru comenzile inserate in tabelul Order, cantitatea produsului comandat, din tabelul Stock trebuie sa fie modificata, adica trebuie sa fie scazuta din valoarea stocului produsului respectiv, valoarea cantitatii comandate.

De asemenea, rezultatete pot fi urmarite si prin acele rapoarte de tip pdf, in care se gasesc, de asemenea datele despre tabele respective din diferite momente de tip.Pentru comenzile efectuate sunt generate si facturi, in care se gasesc: numele clientului care a depus comanda, numele produsului comandat si pretul pe care trebuie sa-l plateasca clientul pentru aceasta comanda. De asemenea, daca o comanda a fost plasata, iar cantitatea care exista in stoc pentru produsul pe care il cere clientul nu este suficienta cerintei lui, un mesaj care sa ne spuna ca stocul nu este suficient va fi generat in locul facturii.

**6.Concluzii:**

As incepe concluzionarea acestei documentatii prin faptul ca exista o multitudine de modalitati de implementare a acestei aplicatii, si totodata, ar putea exista si multe alte imbunatatiri care ar putea sa ii fie adaugate. De asemenea ar putea sa fie gandita din mai multe perspective, astfel incat, domeniul de aplicabilitate sa ii fie din ce in ce mai mare si sa fie cat mai usor de folosit. De asemenea, tin sa specific ca am aprofundat si mai mult prin aceasta tema notiunile de programare orientata pe obiect, cunostiintele in tot ce inseamna limbajul Java, lucrul cu fisiere atat de tip txt, cat si pdf la nivel de scriere si printare in el, legarea unei aplicatii la baze de date, cat de importanta este implementarea unui plan al aplicatiei, inainte de implementarea efectiva a codului, dar si multe altele.

**6.Bibliografie**

* <https://www.baeldung.com/java-pdf-creation>
* <https://dzone.com/articles/layers-standard-enterprise>
* <https://mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/>