Tema 3

Order management

Stefanovici Miruna Andreea

Grupa 30229

**Continut**

**1.***Obiectivul temei*

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

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

**4.***Implementare*

**5.***Rezultate*

**6***.Concluzii*

**7.***Bibliografie*

**-1- Obiectivul temei**

Obiectivul temei consta in simularea plasarii unor “comenzi”, ale unor “clienti”, ce au ca cerinte anumite cantitati de “produse” existente intr-un “depozit”. Pentru realizarea acestei simulari, rolul “depozitului” va fi jucat de o baza de date relationala, “clientii”, “produsele” si “comenzile” fiind reprezentate de instante ale unor clase. “Depozitul”, pe langa “produsele” pe care le detine, va reprezenta si comunicarea dintre “client” si “personal”, astfel ca pe langa tabelele specific produselor, vor exista sit abele ce vor inregistra clientii si comenzile facute de acestia.

Pentru ca scopul principal sa poata fi atins, trebuie urmarita o serie de pasi:

-analizarea potentialelor scenarii de folosire a aplicatiei de catre un utilizator, a interactiunii dintre user si program;

-alegerea modului de proiectare, a structurilor de date folosite in cadrul acesteia;

-alegerea modului de implementare a algoritmilor;

-descrierea claselor prin evidentierea rolului fiecarui camp si fiecarei metode;

-analizarea rezultatelor in urma implementarii, pe testarea unor exemple ce creeaza mai multe scenarii de abordare.

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

Scenariul de utilizare abordat este urmatorul :

* se presupune ca “depozitul” este existent;
* sunt introduse comenzi prin anumite cuvinte cheie;
* se considera ca fiecare cuvant cheie reprezinta cate o “actiune” infaptuita asupra “depozitului”;
* se considera ca dupa ce comenzile au fost introduse si receptate de catre aplicatie, “depozitului” ii sunt facute modificarile cerute;
* ca rezultat, se analizeaza mai intai noile date ale “depozitului”;
* se verifica daca in urma comenzilor facute de catre clienti au fost generate “facturi”;
* se verifica daca in urma receptarii cuvantului cheie ce reprezinta cererea unui “Raport”(cerere facuta de catre utilizator), au fost generate rapoartele respective ce consta in liste cu anumite date despre “deposit”, ca de exemplu lista comenzilor facute sau lista produselor existente.

Cuvintele cheie ce reprezinta “modificarile” ce trebuie aduse depozitului sunt citite dintr-un fisier text.

“Rapoartele” si “facturile” vor consta in documente PDF.

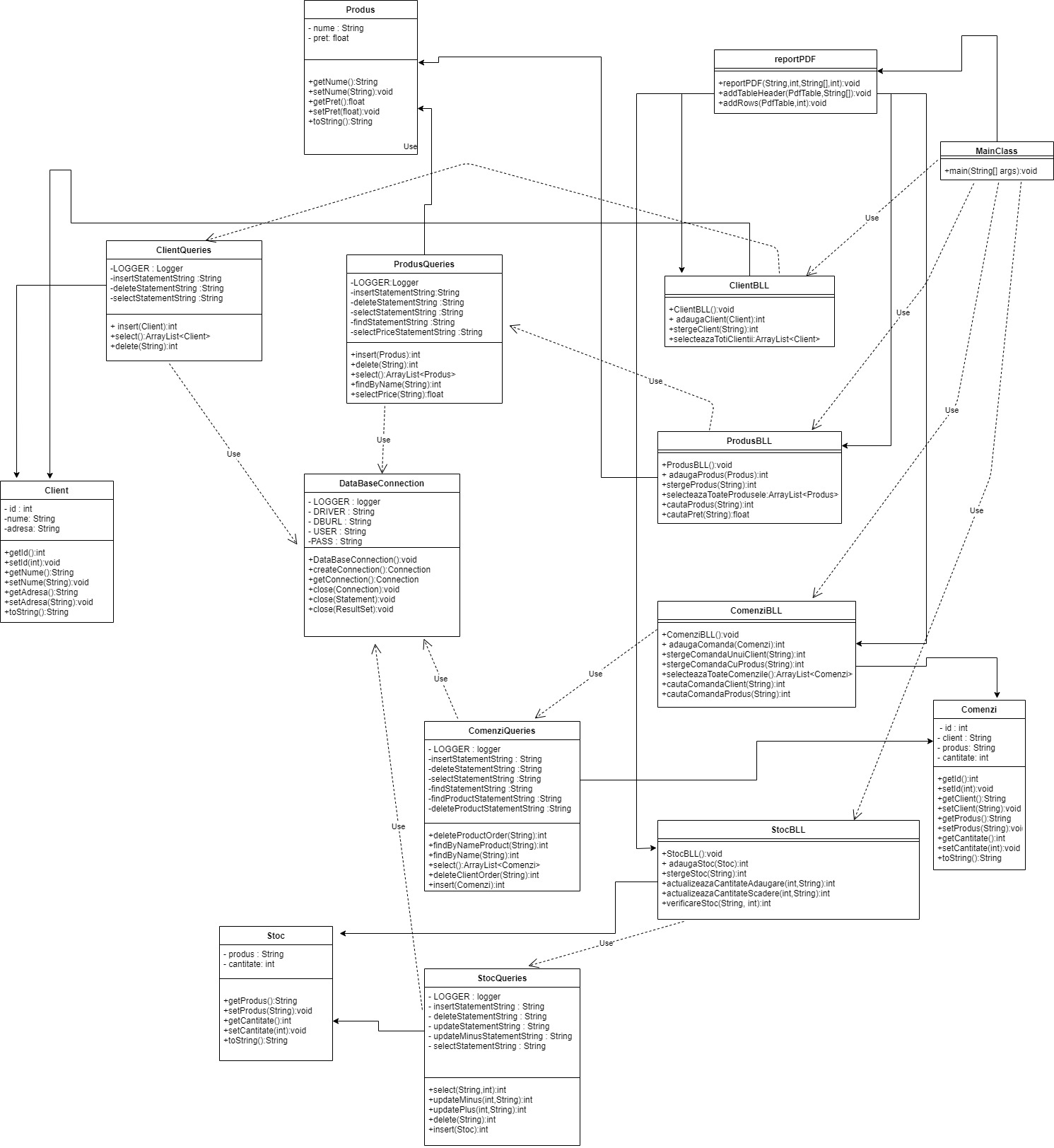
**-3-Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)**

In ceea ce priveste proiectarea aplicatiei, conceptul din spate consta in conexiunea cu o baza de date relationala.O baza de date reprezinta un mod de a stoca anumite informatii, conectate intre ele ca utilizare, cu posibilitatea de a putea fi accesate si modificate.O baza de date este memorata in fisiere. Aceste fisiere sunt manipulate cu ajutorul sistemelor de gestiune a bazelor de date SGBD. Modelul clasic de baza de date, model folosit in cadrul acestei aplicatii, este cel relational.Acest model consta in faptul ca datele sunt memorate in tabele. Totusi, pe langa aceste tabele, o baza de date poate dispune si de procedure stocate, trigger-e, mecanisme de Securitate si de gestiune a tranzaciilor, indecsi, utilizatori si grupuri de utilizatori. Accesul unui user la o baza de date create se realizeaza prin intermediul unui driver.Acesta este cel prin care se face comunicarea cu informatiile respective, este cel care formeaza legatura dintre aplicatie si baza de date. JDBC, Java Databse Conectivity, driver-ul folosit in cadrul aplicatiei, este o interfata standard SQL de acces la baze de date. Contine clase si interfete Java ce ofera posibilitatea de a proiecta programe ce sunt in stransa legatura cu bazele de date.Pachetul folosit este *java.sql*. Astfel, folosind driverul JDBC, programul va putea trimite secvente SQL bazei de date, manipuland astfel informatiile respective din aceasta. De asemenea, ofera si posibilitatea de a extrage informatii din tabele.Pentru crearea bazei de date se utilizeaza interfata grafica de acces la un server MySQL(un sistem de gestiune a bazelor de date relationale) MySQL-Workbench. Acesta este utilizat frecvent, fiind o optiune accesibila ce ofera un simplist management al bazelor de date si support pentru proiectarea vizuala a acestora.

Baza de date folosita in cadrul proiectului se numeste “Warehouse” si dispune de 4 tabele *: Client, Produs, Comenzi* si *Stoc*.

Clasele vor reprezenta cele patru tabele, vor contine metodele specifice quqery-urilor, vor realiza conexiunea dintre baza de date si operatiile cerute si de asemenea vor constitui mecanisme de generare ale unor documente PDF. Datorita necesitatii organizarii in functie de scop, acestea vor fi impartite in pachete : **model**(clasele ce vor reprezenta cele patru tabele),**dataAccessLayer** ( accesul la baza de date si clasele ce contin metodele specifice query-urilor), **businessLayer**( clase representative pentru logica aplicatiei), **presentationLayer**(clasa utilizata pentru generarea de PDF-uri) si **parsingAndLogic** (clasa ce contine metoda main in care are loc citirea comenzilor din fisier si efectuarea operatiilor in functie de acestea).

**Diagrama UML**(*exista si ca poza separata si este adaugata impreuna cu toate celelalte fisiere)*



**-4-Implementare**

Urmeaza descrierea claselor in functie de pachetul in care se afla.

**Pachetul MODEL**

*Clasa Client*

Reprezinta tabelul “Client” din baza de date. Are ca variabile instanta un identificator unic “id”(int) ,ce va fi cheie primara in tabel, un String “nume” ce va reprezenta numele clientului si un String “adresa” ce va reprezenta adresa clientului.Ca metode, clasa nu contine decat getter-e si setter-e, precum si metoda toString() implementata pentru initialele verificari ale corectitudinii pe care le-am facut la inceputul proiectului.

*Clasa Produs*

Reprezinta tabelul “Produs” din baza de date. Are ca variabile instanta un String “nume” ce va reprezenta numele produsului( va fi cheie primara in table) si un float “pret” ce va reprezenta pretul produsului.Ca metode, clasa nu contine decat getter-e si setter-e, precum si metoda toString() implementata pentru initialele verificari ale corectitudinii pe care le-am facut la inceputul proiectului.

*Clasa Comenzi*

Reprezinta tabelul “Comenzi” din baza de date. Are ca variabile instanta un identificator unic “id”(int) ,ce va fi cheie primara in table, un String “client” ce va reprezenta numele clientului ce plaseaza o comanda, un String “produs” ce va reprezenta numele produsului cerut si un int “cantitate” ce va reprezenta cantitatea ceruta de catre client.Ca metode, clasa nu contine decat getter-e si setter-e, precum si metoda toString() implementata pentru initialele verificari ale corectitudinii pe care le-am facut la inceputul proiectului.

*Clasa Stoc*

Reprezinta tabelul “Stoc” din baza de date. Are ca variabile instanta un un String “produs” ce va reprezenta numele produsului existent in “depozit” si un int “cantitate” ce va reprezenta cantitatea totala a produsului.Ca metode, clasa nu contine decat getter-e si setter-e, precum si metoda toString() implementata pentru initialele verificari ale corectitudinii pe care le-am facut la inceputul proiectului.

**Pachetul dataAccessLayer**

*Clasa DataBaseConnection*

Este clasa ce contine datele necesare conectarii la baza de date creata. Contine de asemenea si metodele necesare crearii conexiunii, precum si a inchiderii acesteia, a unui statement si a unui Result Set.Clasa folositia aici, responsabila cu efectuarea unei conexiuni, este DriverManager.Cand un driver-ul JDBC este incarcat in memorie folosind aceasta Clasa, se incepe procesul de stabilire a conexiuni. Deoarece poate exista o multitudine de drivere inregistrate, trebuie specificat exact driverul folost. De aceea, se pastreaza in String-ul **DBURL** o adresa specifica. Astfel, DriveverManager-ul o sa parcurga lista driver-elor inregistrate pana cand unul intre ele va recunoaste adresa salvata in DBURL.Daca nu exista, va fi afisata o exceptie de tipul SQLException.Reponsabila cu aceasta actiune este metoda **createConnection()**. Dupa crearea conexiunii, vom putea crea obiecte de tip Statement, cu ajutorul carora se pot trimite secvente SQL catre baza de date, precum si obiecte de tipul ResultSet cu ajutorul carora putem extrage informartii din tabelele bazei de date.Pentru inchiderea acestora la sfarsitul executarii unui query,precum si pentru oprirea conexiunii la baza de date s-au declarant metodele **close()**.

*Clasa ClientQueries*

“ClientQueries” este clasa ce contine metodele reprezentative operatiilor SQL pentru baza de date, pentru tabelul “Client”. In fiecare metoda se va face conexiunea la baza de date. Are ca si variabile instanta String-uri ce reprezinta query-urile in limbaj sql, in interiorul carora se afla simboluri ‘?’ care vor fi inlocuite cu anumite valori. Metodele sunt specifice operatii pe baze de date.Metoda **insert()** consta in declararea unui statement ce va reprezenta query-ul SQL din string-ul insertStatementString. Simbolurile '?' vor fi inlocuite cu variabilele instanta ale obiectului de tip "Client". Se va realiza instructiunea prin apelerea metodei "executeUpdate", iar in caz de Warning,se va afisa mesajul aferent warning-ului.Dupa efectuarea operatiei, se vor apela metodele din clasa DataBaseConnection de inchidere a statemenet-ului si a conexiunii. Metoda **select()** consta in declara un obiect de tip "Client" si un ArrayList de "Client".Se declara un statement ce va reprezenta query-ul SQL din string-ul selectStatementString Se va realiza instructiunea prin apelerea metodei "executeUpdate", iar in caz de Warning,se va afisa mesajul aferent warning-ului. Obiectul de tip ResultSet va contine rezultatul query-ului, fiecare element al acestui ResultSet reprezentand un rand din tabel.Valorile fiecarei celule din acel tabel poate fi extras prin specificarea numelui coloanei asociate.Datele sunt astfel extrase, obiectul de tip "Client" este format si adaugat in lista de clienti.Dupa efectuarea operatiei, se vor apela metodele din clasa DataBaseConnection de inchidere a statemenet-ului, a resultSet-ului si a conexiunii.Se returneaza lista de client.Metoda **delete**() consta in repetarea aceluasi process, de data aceasta folosind String-ul deleteStatementString.Are ca effect stergerea unui client cu un anumit nume. Toate simbolurie ‘?’ sunt inlocuite prin apelarea pentru fiecare statement, a functiei de **setString**().

*Clasa ComenziQueries*

“ComenziQueries” este clasa ce contine metodele reprezentative operatiilor SQL pentru baza de date, pentru tabelul “Comenzi”.Pe langa operatiile de insert,select si delete care se regasesc sub aceeasi forma ca si in clasa Client(doar ca acum in functie de datele din tabelul “Comenzi”), in aceasta clasa mai exista si metode pentru selectare in functie de un anumit atribut :**findByName(String numeClient)** si **findByNameProduct(String numeProdus)**. Ambele respecta acelasi pattern. Sa luam ca exemplu functia findByName(String numeClient). Metoda are rolul de a cauta comenzile unui anumit client in baza de date.Se declara o variabila "exista" care are initial valoarea 0. Se realizeaza conexiunea la baza de date si se declara RestulSet-ul "rs". Simbolul '?' va fi inlocuit cu numele clientului cautat. Astfel, dupa ce se executa query-ul, se verifica pentru fiecare rand, daca numele clientului respectiv este egal cu cel cautat. Daca este egal, inseamna ca anumita comanda a fost gasita,"exista" devine 1 si se iese din bucla.Astfel se returneaza valoarea lui “exista” : 1 daca a fost gasit clientul cu numele respective si 0 altfel.Operatiile de delete sunt de mai multe feluri:in functie de numele unui client sau in functie de numele unui produs.

*Clasa ProdusQueries()*

“ProdusQueries” este clasa ce contine metodele reprezentative operatiilor SQL pentru baza de date, pentru tabelul “Produs”.Si in aceasta clasa, pe langa insert, select ( echivalent cu select\* ) si delete ,findByName care se regasesc sub aceeasi forma ca si in clasele anterioare, regasim o metoda cu un comportament usor diferit. Metoda **selectPrice(String numeProdus**) cauta pentru un anumit produs, pretul lui. Se declara variabila "pret" care va tine valoarea pretului pentru produsul cu denumirea"numeProdus". Se parcurg randurile tabelului,cu ajutorul unui ResultSet si se salveaza valoare pretului in variabila"pret", iar atunci cand se gaseste produsul cautat, se iese din bucla.Metoda va returna pretul produsului cautat.

*Clasa StocQueries*

"StocQueries" este clasa ce contine metodele reprezentative operatiilor SQL pentru baza de date, pentru tabelul “Stoc”. In fiecare metoda se va face conexiunea la baza de date.Aici, pe langa metodele clasice, avem ca noutate oepratia de update, care va actualiza cantitatea produsului in functie de operatiile facute : daca se mareste stocul ( se adauga din nou acelasi produs), se va mari cantitatea in tabel, iar cand se fac comenzi, cantitatea va scadea.Reprezentative sunt functiile **updatePlus**() si **updateMinus**() care realizeaza logica explicate, ambele primind ca parametrii cantitatea ce se va scadea( la updateMinus() ) sau se va aduna ( la updatePlus() ) si numele produslui pentru care se va realiza actiunea.

**Pachetul PresentationLayer**

*Clasa reportPDF*

"reportPDF" este clasa ce se ocupa cu generarea documentelor PDF in urma detectarii, in fisierul commands.txt, a comenzii "Report". La fiecare comanda "Report -nume tabel-" va fi scris, intr-un fisier PDF, un tabel al tuturor datelor existente in tabelul respectiv, la momentul cererii raportului (instructiunie echivalenta cu SELECT \*). Cele trei tabele vizare de aceasta comanda sunt: "Client", "Produs" si "Comenzi".Atunci cand va fi creat un obiect de tip reportPDF, va primi ca parametrii :pdfNume - numele fisierului in care va fi adaugat tabelul,nrColoane - numarul de coloane al tabelului, coloane - lista numelor fiecarei coloane din tabel,selectie - in functie de valoare, se face afisarea dintr-un anumit tabel,In constructor vor fi apelate cele doua metode necesare crearii unui output sub forma tabularala.Astfel : **addTableHeader**(PdfPTable table,String[] coloane) are rol de formare al header-ului si de aceea are nevoie de o lista de nume ale coloanelor.Pentru fiecare nume de coloana se va crea o celula noua in care va fi adaugat numele coloanei.In metoda **addRows**(PdfPTable table ,int selectie) ,in functie de parametrul “selectie”, va fi creat un obiect "ArrayList" de "T" unde T reprezinta unul dintre urmatoarele obiecte : Client, Produs sau Comenzi.Obiectul "ArrayList" va avea valoarea returnata de functia de selectiedin fiecare tabel ( SELECT \* ), apelata prin intermediul obiectului BLL asociat, Valoarea este chiar lista de clienti/produse/comenzi, ce va fi parcursa.Pentru fiecare element parcurs se va forma cate o celula cu valoarea fiecarei variabile instanta.Parametru selectie va fi 1 pentru tabela “Client”, 2 pentru tabela “Produs” si 3 pentru tabela “Comenzi”.

**Pachetul businessLayer**

Contine clasele ClientBLL, ProdusBLL, StocBLL si ComenziBLL. Fiecare dintre aceste clase contine strict metode ce apeleaza metodele din clasele Queries associate.De exemplu, "ComenziBLL" este clasa reprezentativa implementarii logicii aplicatiei in ceea ce priveste tabela "Comenzi".Fiecare metoda cu nume sugestiv va apela cate o metoda din clasa "-nume tabel-.Queries" logic asociata. Fiecare metoda va respecta tipul metodei din clasa logic asociata, precum si parametrii acesteia.Aceasta clasa va apela metodele din "ComenziQueries".Acelasi lucru se intampla si pentru celelalte clase.Acest layer, intr-adevar, nu este complet dezvoltat, deoarece nu contine o implementare complete a logicii aplicatiei. De exemplu, ca dezvoltare ulterioara, fiecare dintre aceste clase poate sa continea metode de validare ale anumitor date ce se doresc a fi introduse in tabelele respective.

**Pachetul parsingAndLogic**

*Clasa MainClass*

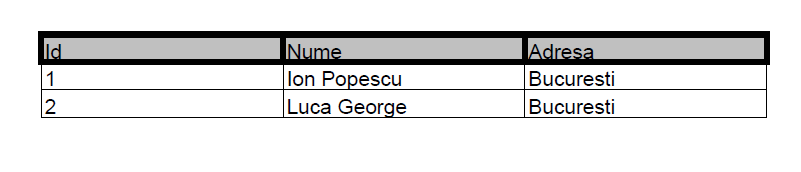
MainClass" este clasa in care se realizeaza citirea din fisierul text.Fisierul text va contine cuvinte cheie precum "insert", "delete", "order"sau "report". In functie de cuvantul cheie, se vor realiza operatiile necesare ce duc la realizarea comenzii, tratand si exceptiile ce pot aparea.

Se declara cate un obiect de tip ClientBLL, de tip ProdusBLL, de tip StocBLL si de tip ComenziBLL. Aceasta actiune reprezinta cumva un fel de initializare a tabelelor.Variabia idClient se va incrementa de fiecare data cand se adauga un nou client, variabila idComanda se va incrementa de fiecare data cand se va adauga o noua comanda iar variabilele numarReport, numarRaportClienti, numarRaportProduse si numarRaportComenzi se vor incrementa la adaugarea cate unui noi raport dedicate unui anumit tabel, vor fi convertite in String si vor face parte din numele noului PDF generat pentru a evita suprascrierea.Utilizand functia “split” se va extrage din fiecare linie a fisierului text cuvintele cheie pentru a putea organiza operatiile, dupa care se incepe executarea acestora efectiva. Cand dorim sa inseram un client, extragem datele din linia fisierului, cream un nou obiect de tip Client si apelam functia de adaugare in tabel.Cand se doreste stergerea unui client, acesta fiind referintiat de coloana “Client” in tabela Comenzi, se va verifica daca anumitul client a depus vreo comanda si daca da, se vor sterge mai intai toate comenzie ce-i apartin si abia dupa se va sterge si din tabela Client.Cand se doreste inserarea unui produs se verifica mai intai daca acesta nu cumva a fost inserat deja, iar daca a fost inserat, se va actualiza doar cantitatea lui in tabela stoc. Daca nu a fost inserat, se va apela metoda de inserare in tabel.La comanda de stergere a unui produs se va proceda la fel ca si la comanda de stergere a unui client.Cand se realizeaza un order, se verifica mai intai daca acel produs exista in tabel.Daca exista, se verifica daca anumita cantitate ceruta este mai mica decat totalul din stoc.Daca respecta acest lucru, inseamna ca se poate comanda, iar factura poate fi generate.Pentru calcularea platii se va cauta pretul produsul comandat si se va inmulti cu anumita cantitate ceruta.Daca insa nu se incadreaza in stoc, va fi generat un pdf cu mesajul “Under-stock!”.Cand se detecteaza cererea de “Report”, se verifica mai intai pentru ce tabel se doreste report-ul, iar in functie de alegere, se vor trimite parametrii specificati in explicatia de la clasa reportPDF.

**-5- Rezultate**

Rezultatele se vor vedea in baza de date(vizualizata in MySQL Workbecnh). Se verifica daca datele existente corespund cu cele asteptate si de asemenea, se verifica si corectitudinea PDF-urilor generate.Instructiunile din fisierul de tip text sunt procesate intr-o singura rulare si de aceea, inserarea anumitor date care ulterior au fost sterse poate fi verificata in urma PDF-urilor generate de fiecare data cand este citita din fisierul text comanda “Report”.

De exemplu, clientul “Ion Popescu” va fi inserat in baza de date, iar cateva comenzi mai tarziu va fi dorit sters. In baza de data nu se va regasi, dar in urma “Report”-ului se va observa inserarea lui anterioara .



”Report”-ul fiecarui tabel va fi afisat sub forma tabulara, pe cand “factura” cu totalul pretului sau cu mesajul “Under-stock!” va fi scris in PDF ca un simplu String. PDF-urile sunt generate cu ajutorul folosirii bibliotecilor iText si PdfBox.

**-6- Concluzii**

In urma realizarii acestui proiect, m-am familiarizat cu lucrul cu baze de date prin intermediul Java si cred ca voi putea imbunatati punerea in practica a cunostintelor dobandite.

In ceea ce priveste dezvoltarea ulterioara a proiectului respectiv, consider ca as putea adauga urmatoarele:

* folosirea unor metode ce valideaza datele adaugate, ce impun anumite reguli de introducere a informatiilor in tabele;
* lucrul prin tehnica Reflection pentru a evita repetitia codului ceea ce ar scurta mult din timpul de implementare.

**-7- Bibliografie**

1.https://www.stackoverflow.com

2.https://www.wikipedia.org

3.https://www.geeksforgeeks.com