MINISTERUL EDUCAȚIEI NAȚIONALE
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Tema 3

Gestionarea comenzilor

Student: Spinean Sebastian

Grupa: 302210

Cuprins:

1. Obiectivul temei

Obiectivul principal:

Obiectivul principal al acestei teme este de a implementa o aplicatie de procesare a comenzilor unor clienti pentru un deposit. De asemenea, aplicatia trebuie sa procesese niste comenzi dintr-un fisier text, sa execute niste operatii asupra unei baze de date sis a genereze rapoarte in format pdf.

Obiective secundare:

Obiective secundare:	Descriere	Capitol
Determinarea cazurilor de utilizare	Se identifica cazuile de utilizare si se descriu prin intermediul diagramelor use case	2
Alegerea structurilor de date	Se aleg structurile de date potrivite	3
Impartirea pe clase si pachete	Impartirea codului in pachete si clase	3
Dezvoltarea algoritmilor	Implementarea algoritmilor	4
Implementarea unei baze de date	Crearea tabelelor relationale	2

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

Cerintele functionale:

In cadrul proiectului se disting urmatoarele cerinte functionale:

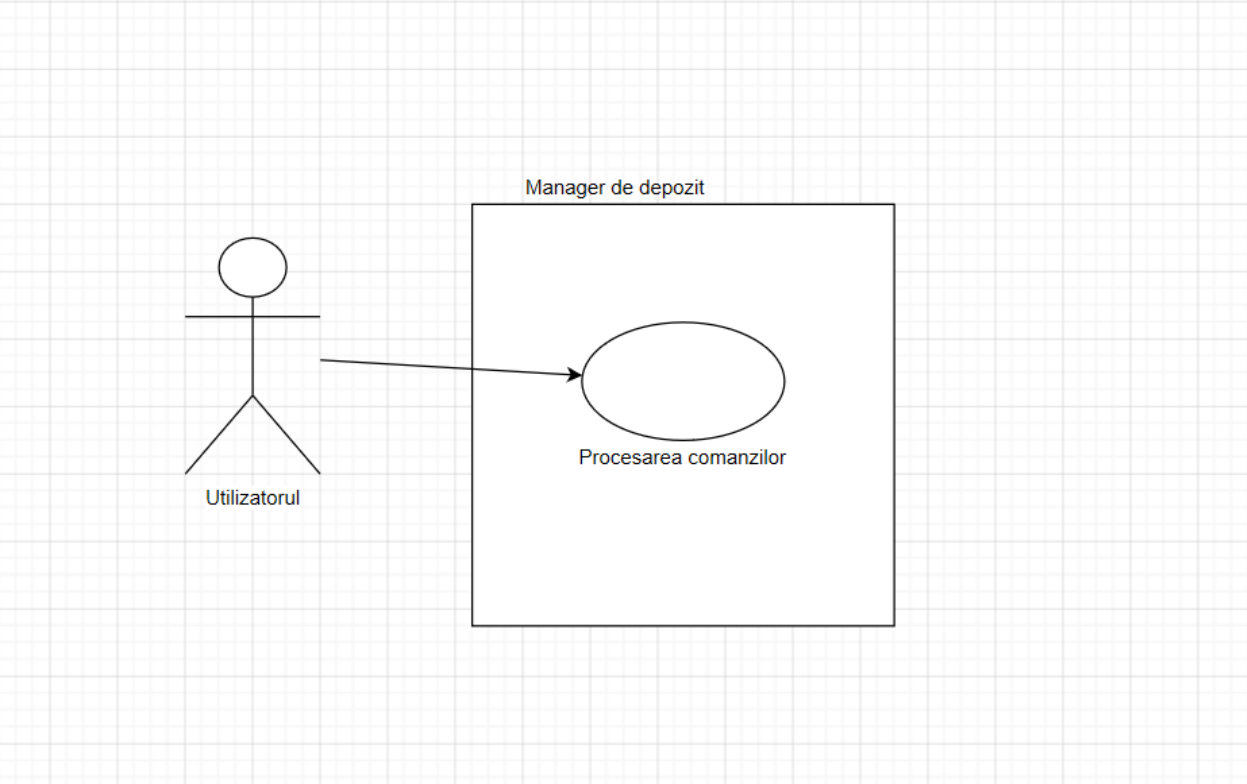
- Implementarea unei baze de date utilizand MySQL Workbanch
- Implementarea unei clase de citire din fisier si procesare a comenzilor
- Generarea de rapoarte in format pdf

Analiza problemei:

Aceasta tema presupune implementarea unei aplicatii care sa gestioneze comenzile unor clienti. Astfel, se construieste o baza de date utilizand MySQL Workbanch. Aceasta baza de date este formata din patru tabele: Client, Product, Order_table, OrderItem. Tabela Client memoreaza datele despre clienti. Astfel, fiecare client primeste un id unic, se retine numele acestuia si adresa sa. De asemenea, numele clientilor este unic, neputand exista mai multi clienti cu acelasi nume. Tabela Product retine datele despre produse. Fiecarui produs ii este asociat un id unic, se retine numele lui, se centralizeaza un stock si pretul acestuia. De fiecare data cand se da o comanda de adaugare a unui produs care mai exista in baza de date se mareste stock-ul si se modifica pretul daca este cazul. Tabela Order_table memoreaza comenzile unui client. Fiecare comanda are un id unic. Toate comenzile unui client se adauga la aceasi comanda existenta deja, astfel incat pe numele unui client nu pot exista doua comenzi cu id-uri diferite. Tabela mai memoreaza totalul platit de client acesta modificandu-se de fiecare data cand se mai executa o instructiune de comanda pentru un client. In tabela OrderItem se memoreaza toate produsele de pe o comanda.

Diagrame use case:

Pentru analiza cazurilor de functionare se folosesc diagramele use case. Astfel, actorul este utilizatorul care foloseste acest proiect. Acest proiect gestioneaza o baza de date si executa comenzi asupra sa.



Cazul de utilizare:

Actorul principal: Utilizatorul

Scenariul cu success:

1. Utilizatorul introduce datele intr-un fisier text.
2. Utilizatorul salveaza fisierul in folderul curent in care se gaseste si proiectul.
3. Se ruleaza aplicatia primind ca parametru numele fisierului in care se gasesc datele de intrare.
4. Se realizeaza operatiile asupra bazei de date si se genereaza rapoartele in format pdf.

Scenariu alternativ:

a. Comenzile introduse in fisierul text de intrare nu sunt valide
   1. Aplicatia nu recunoaste comenzile si nu le executa
   2. Utilizatorul se reintoarce la pasul 1 si reintroduce comenzile in fisier

b Fisierul cu datele de intrare nu este salvat in folderol current in care se gaseste si proiectul

1. Aplicatia nu reuseste sa gaseasca fisierul
2. Se afiseaza un mesaj de eroare
3. Utilizatorul revine la pasul 2 si salveaza fisierul in folderul corespunzator.

3 Proiectare

Codul este impartit in patru pachete:

- Presentation
- businessLayer
- dataAccessLayer
- model

Astfel, codul este structurat pe modelul arhitecturii Layered. Fiecare layer are un scop principal si apeleaza functii din layer-ul de sub el. Fiecare pachet din acest program corespunde unui layer. Pachetul Presentation se ocupa de citirea datelor de intrare si interpretarea acestora. In pachetul businessLayer este continuta logica de program. Aici se prelucreaza datele si se apeleaza operatiile asupra bazei de date. Conexiunea la baza de date si legatura efectiva se realizeaza in pachetul dataAccessLayer. Pachetul model contine niste clase care modeleaza tabelele din baza de date.

Diagrama de pachete:

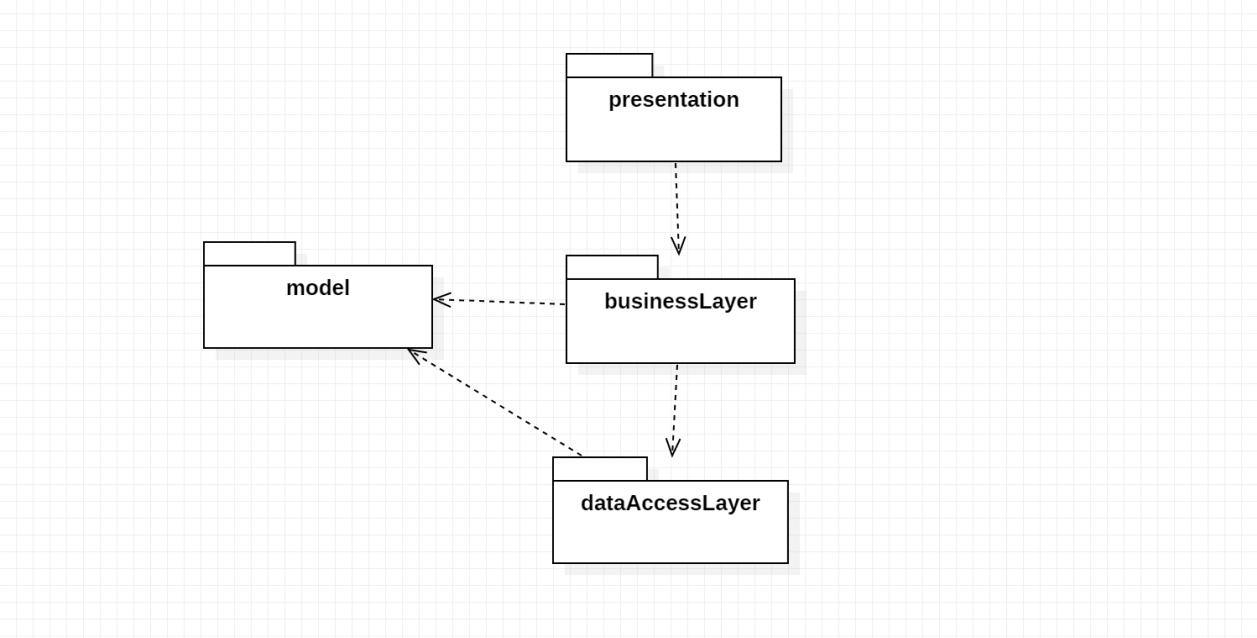
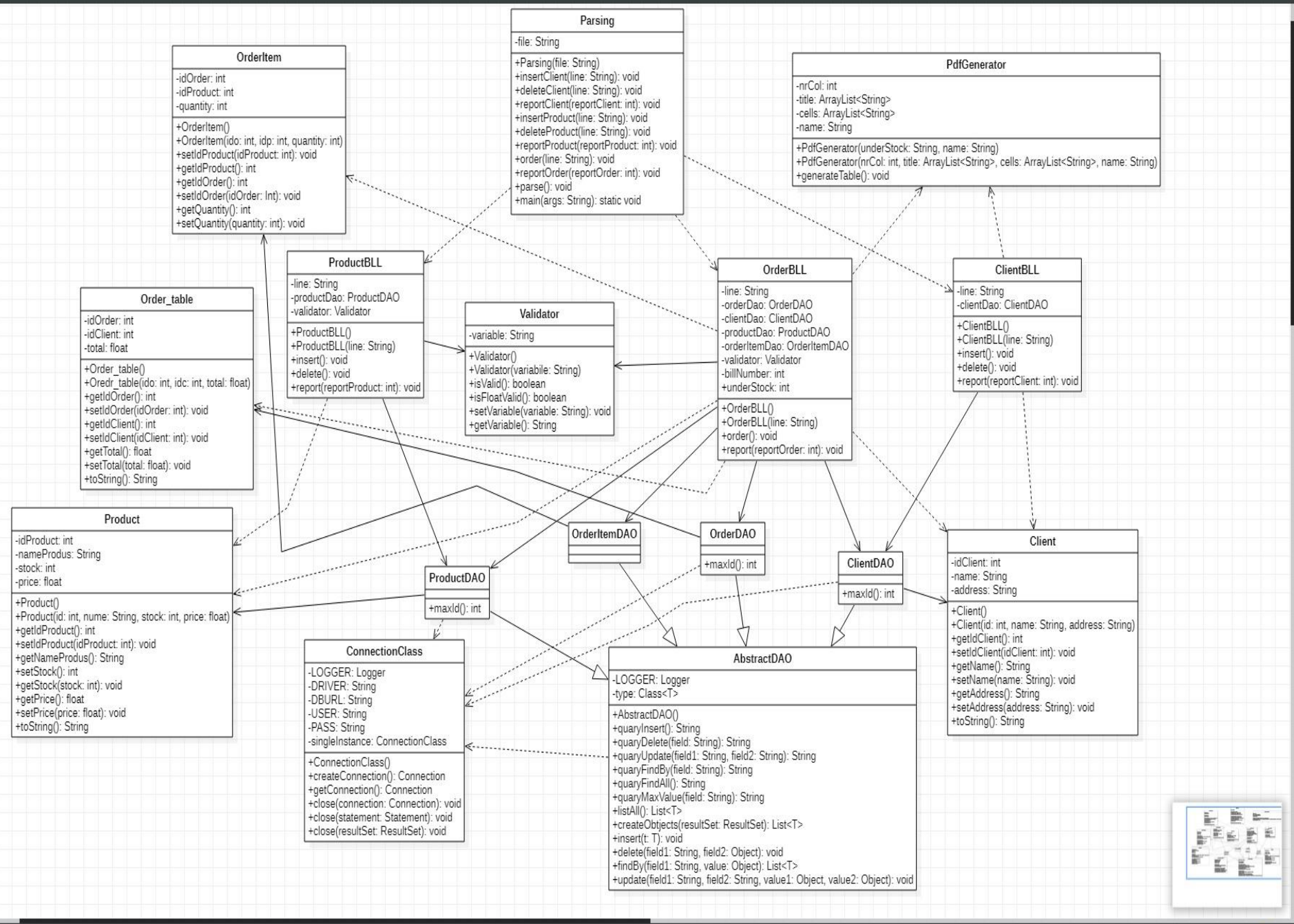


Diagrama de clase:

Format Model Tools View Window Debug Help



80%

Dupa cum se poate vedea din diagrama de clase UML, codul aplicatiei este impartit in 16 clase.

Pachetul Model:

Acest pachet este format din patru clase: Client, Product, Order_table, OrderItem. Aceste clase modeleaza structurile tabelelor din baza de date. Astfel, cele patru tabele au aceleasi nume cu tabelele din baza de date si aceleasi variabile instanta, ca nume si ca tip de data, cu atributele tabelelor.

Pachetul Presentation:

Acest pachet este format dintr-o singura clasa numita Parsing. Aceasta clasa realizeaza citirea comenzilor dintr-un fisier de tip text si identificarea acestora. Astfel, in functie de instructiune se decide c ear trebui sa faca aplicatia.

Pachetul BusinessLayer:

In acest pachet sunt grupate clasele care se ocupa de logica programului. In acest caz, clasele sunt: ClientBLL, ProductBLL, OrderBLL, Validator si PdfGenerator. Aceste clase se ocupa cu procesarea datelor, generarea rapoartelor in format pdf si cu apelarea operatiilor asupra bazei de date.

Pachetul DataAccessLayer:

Acest pachet este format din sase clase si anume: ConnectionClass, AbstractDAO, ClientDAO, ProductDAO, OrderDAO si OrderItemDAO. Cu ajutorul acestor clase se realizeaza conexiunea cu baza de date. De asemenea, aceste clase detin metode care opereaza direct asupra tabelelor din baza de date.

4 Implementare

Pachetul Model:

Clasa Client:

Aceasta clasa se mapeaza pe tabela Client din baza de date.

Variabile instanta:

Aceasta clasa are trei variabile instanta. Astfel, fiecare client se caracterizeaza printr-un idClient de tip int, un name de tip String si o adresa, address, tot de tip String. Aceste variabile instanta au acelasi nume si acelasi tip cu atributele tabelului Client.

Constructori:

Clasa client detine doi constructori. Primul este un constructor fara parametri in timp ce al doilea contine trei argumente cu care initializeaza variabilele instanta.

Metode:

Aceasta clasa detine metode de getters si setters pentru fiecare variabila instanta. De asemenea, mai exista o metoda de toString care returneaza datele unui client sub forma de String.

Clasa Product:

Aceasta clasa se mapeaza pe tabela Client din baza de date.

Variabile instanta:

Clasa are patru variabile instanta. idProduct reprezinta id-ul unic al produselor, nameProdus reprezinat numele, stock reprezinta cantitatea totala din acel produs si price reprezinta pretul.

Constructori:

Aceasta clasa detine doi constructori. Primul este constructorul fara parametri iar celalalt primeste cate un parametru pentru fiecare variabila instanta.

Metode:

Clasa Product detine cate o metoda de getters si setters pentru fiecare variabila. In plus mai exista metoda toString care returneaza datele unui produs sub forma de String.

Clasa Order_table:

Aceasta clasa corespunde tabelului order_table.

Variabile instanta:

Clasa are trei variabile instanta: idOrder – seminifca id-ul unic al fiecarei comenzi, este de tip int. idClient – reprezinta clientul care a facut comanda, tot de tip int. total – marcheaza pretul platit de client la acea comanda, de tip float.

Constructori:

Clasa are doi constructori: unul fara parametri si unul cu. Cel cu parametrii initializeaza toate variabilele instanta.

Metode:

De asemenea, clasa contine metode de getters si setters pentru fiecare dintre variabilele instanta si o metoda de toString pentru a afisa detaliile sub forma de String.

Clasa OrderItem:

Aceasta clasa corespunde tabelului orderItem din baza de date.

Variabile instanta:

Clasa contine trei variabile instanta: idOrder reprezentand numarul comenzii, idProduct reprezentand produsul de pe comanda si quantity semnificand cantitatea.

Constructori:

La fel ca si celalalte clase din pachet are doi constructori: fara parametrii si cu parametrii. Cel cu parametrii initializeaza toate variabilele instanta.

Metode:

Clasa are metode de getters si setters pentru fiecare variabila.

Pachetul Presentation:

Clasa Parsing:

Aceasta este singura clasa din acest pachet. Clasa Parsing se ocupa cu citirea din fisier a comenzilor si interpretarea acestora.

Variabile instanta:

Clasa are o singura variabila instanta de tip String. Aceasta variabila semnifica numele fisierului din care se citesc comenzile.

Constructori:

Aceasta clasa are un singur constructor care primeste un parametru de tip String. Cu acest parametru se initializeaza variabila instanta.

Metode:

Clasa Parsing are zece metode.

O metoda foarte importanta este metoda parse. Aceasta metoda nu are niciun parametru. Ea se ocupa cu citirea datelor din fisier. Astfel, in aceasta metoda se citeste linie cu linie din fisier si identifica ce comanda este. Linia este despartita in cuvine si in functie de primul cuvant se verifica daca este comanda de insert, delete, order sau report. In functie de al doilea cuvant se verifica daca comanda se aplica asupra clientilor, produselor sau comenzilor. Dupa ce s-a identificat comanda se apeleaza o alta metoda a clasei

Metodele care se aplica asupra clientilor sunt: insertClient, deleteClient si reportClient. Primele doua metode primesc ca argument linia citita din fisier iar cealalta un parametru de tip int ce reprezinta al catelea report se genereaza pentru tabela Client. Aceste trei metode apeleaza alte metode din clasa ClientBLL.

Metodele care se aplica asupra produselor sunt: insertProduct, deleteProduct si reportProduct. Aceste trei metode functioneaza asemanator cu cele pentru Client.

Asupra tabelei Order_table actioneaza doar metodele order si reportOrder. Metoda order primeste ca parametru linia citita din fisier in timp ce metoda reportOrder primeste un int care reprezinta numarul de rapoarte generate pentru tabela order_table. Aceste metode apeleaza alte metode din clasa OrderBLL.

De asemenea, aceasta clasa contine metoda main. Metoda main primeste ca argument numele fisierului din care se face citirea datelor. In aceasta metoda se apeleaza metoda parse.

Pachetul businessLayer:

Clasa Validator:

Aceasta clasa este utilizata pentru a valida anumite date de intrare.

Variabile instanta:

Aceasta clasa are o singura variabila instanta de tip String numita variable.

Constructori:

Clasa Validator are doi constructori: unul cu un parametru de tip String si una fara niciun parametru.

Metode:

Clasa dispune de doua metode: isValid si isFloatValid care nu primesc niciun parametru. Ambele metode returneaza true daca variabila este valida si false in caz contrar. Metoda isValid verifica daca variable care este de tip String este are forma unui numar intreg pozitiv in timp ce metoda isFloatValid verifica daca variabila instanta are forma unui numar real pozitiv.

Clasa ClientBLL:

Aceasta clasa executa partea de logica a programului legata de tabela Client.

Variabile instanta:

Exista doua variabile instanta: line care este de tip String si semnifica o linie citita din fisier si clientDao care este de tip ClientDAO.

Constructori:

Si aici exista doi constructori: unul fara parametrii si unul cu un parametru de tip String prin care se initializeaza variabila line.

Metode:

Clasa Client dispune de trei metode corespunzatoare celor trei comenzi care se pot efectua asupra clientilor.

Metoda insert nu are niciun parametru. Aceasta metoda este apelata cand din fisier se citeste comanda de “Insert client”. In cadrul ei se extrage din linia de comanda numele si adresa clientului care trebuie inserat. Pentru id se extrage cel mai mare id din tabela si se incrementeaza. Dupa acea este executata comanda de insert asupra tabelului Client din baza de date. In cazul in care mai exista deja un client cu acel nume nu se va executa comanda deoarece in tabel coloana name are adaugata constrangerea UNIQE.

Metoda delete nu are niciun parametru. Metoda extrage din linia de comanda numele clientului care trebuie sters si executa operatia de stergere din tabela dupa nume.

Metoda reportClient primeste ca parametru un int care simbolizeaza numarul de rapoarte generate. Intr-o lista de clienti sunt stocati toti clientii din tabela. Atributele tuturor clientilor sunt adaugate sub forma de String intr-un ArrayList. De asemenea, intr-un alt ArrayList de stringuri sunt adaugate numele coloanelor din tabela. Dupa acea se genereaza un pdf care contine o tabela cu datele clientilor.

Clasa ProductBLL:

Aceasta clasa executa partea de logica a programului legata de tabela Product.

Variabile instanta

Aceasta clasa contine trei variabile instanta: line care este de tip String si semnifica o linie citita din fisier, productDao care este de tip ProductDAO si validator de tipul Validator

Constructori:

Clasa are doi constructori: cu un parametru String si unul fara parametrii.

Clasa are metodele insert, delete si report corespunzatoare celor trei actiuni care se pot executa asupra tabelei Product. Metoda insert extrage numele produsului, cantitatea si pretul din linia citita din fisier. Daca cantitatea si pretul sunt valide se executa operatia de insert in cazul in care nu mai exista in baza de date acest produs deja. Daca exista atunci se mareste stock-ul si se modifica pretul daca este cazul.

Metoda de delete face stergerea unui unui produs dupa nume.

Metoda reportProduct functioneaza asemanator cu cea din clasa Client.

Clasa OrderBLL:

Aceasta clasa executa logica de program pentru tabelele order_table si orderItem.

Variabile instanta:

Clasa are 8 variabile: line de tipul String, orderDao de tip OrderDAO, clientDao de tip ClientDAO, productDao de tip ProductDAO, orderItemDao de tip OrderItemDAO, validator de tipul Validator. De asemenea mai exista doua variabile statice de tip int billNumber si underStock.

Constructori:

Metoda dispune de doi constructori: fara parametrii si cu un parametru String.

Metode:

Clasa are doua metode: order si report

Metoda order creaza o noua comanda. Din linia citita din fisier se selecteaza numele clientului, produsul droit si cantitatea. Daca cantitatea este valida se merge mai departe si se verifica daca exista acel client si acel produs in baza de date. Se creaza un obiect order de tip Order_table avand datele noii comenzi. Dupa aceea se verifica daca nu mai exista deja o comanda pentru clientul respective si daca stockul este sufficient. In acest caz se creaza o noua comanda si se micsoreaza stockul. Daca pe numele acelui client mai exista o comanda atunci aceasta comanda se adauga la cealalta marindu-se totalul comenzii si scazandu-se stockul produsului. Pentru ambele cazuri se genereaza in format pdf o factura. In cazul in care stockul este insuficient se genereaza un pdf cu mesajul stock insuficient.

Metoda report functioneaza ca si in clasele ClinetBLL si ProductBLL.

Clasa PdfGenerator:

Aceasta clasa se ocupa de generarea pdf-urilor.

Variabile instanta:

Clasa are patru variabile instanta: nrCol reprezinta numarul de coloane al tabelului, title este un ArrayList si reprezinta titlurile coloanelor din table, cells este de tip ArrayList si contine celulele din table iar String-ul name semnifica numele pdf-ului generat.

Constructori:

Clasa contine doi constructori: Primul primeste cate o variabila corespunzatoare fiecarei variabile instanta. Acest constructor apeleaza metoda generateTable. Al doilea constructor primeste doi parametrii de tip String, primul semnificand mesajul care trebuie afisat in pdf iar cel de-al doilea numele pdf-ului.

Metode:

O singura metoda: metoda generateTable care construieste un table. Este folosita pentru generarea raporturilor.

Pachetul businessLayer:

Clasa ConnectionClass:

Realizeaza conexiunea propriu-zisa la baza de date.

Variabile instanta:

O variabila de tipul Logger si trei variabile de tip String: DRIVER, DBURL, USER, PASS. Mai exista o variabila singleInstance de tip ConnectionClass.

Constructori:

Un singur constructor fara parametrii.

Metode:

Exista cinci metode. Metoda createConnection realizeaza conexiunea la baza de date. Metoda getConnection obtine conexiunea create. Exista trei metode de close care inchid o conexiune, un statement si un ResultSet.

Clasa AbstractDAO<T>:

Aceasta clasa este una abstracta folosita pentru a defini operatiile care se executa asupra bazei de date in mod generic. Parametrul T ar putea fi orice calasa model care mapeaza o tabela din baza de date. Pentru aceasta clasa generica s-a folosit tehnica reflexiei.

Variabile instanta:

O variabila de tip Logger, o variabila numita type de tip Class<T>

Constructori:

Metoda detine un singur constructor fara parametrii.

Metode:

Clasa contine 12 metode.

Metodele queryInsert, queryDelete, queryUpdate, queryFindBy, queryFinaAll si queryMaxValue returneaza sub forma de String structura unui query SQL. Aceste metode sunt generice si folosesc tehnica reflexiei.

Metoda listAll returneaza sub forma unei liste de tip T continutul unei tabele.

Metoda createObjects primeste ca argument un ResultSet si construieste o lista de elemente de tip T.

Metoda insert adauga intr-o tabela un obiect t de tip T. Valorile campurilor care trebuie inserate in tabel se obtin de asemenea prin reflexie.

Metodele de delete si update executa operatiile de stergere respectiv de modificare a unei valori intr-o tabela.

Metoda findBy returneaza o lista de obiecte de tip T care indeplinesc o anumita conditie.

Clasa ClientDAO:

Aceasta clasa mosteneste clasa AbstractDAO inlocuind parametrul generic T cu clasa Client.

Se mai defineste in aceasta clasa metoda maxId care returneaza id-ul maxim din tabelul Client.

Clasa ProductDAO:

Aceasta clasa mosteneste clasa AbstractDAO inlocuind parametrul generic T cu clasa Product.

Se mai defineste in aceasta clasa metoda maxId care returneaza id-ul maxim din tabelul Product.

Clasa OrderDAO:

Aceasta clasa mosteneste clasa AbstractDAO inlocuind parametrul generic T cu clasa Order_table.

Se mai defineste in aceasta clasa metoda maxId care returneaza id-ul maxim din tabelul Order_table

Clasa OrderItemDAO:

Aceasta clasa mosteneste clasa AbstractDAO inlocuind parametrul generic T cu clasa OrderItem.

5 Rezultate

Pentru testare s-a rulat aplicatia utilizand un fisier de intrare avand urmatorul continut:

Insert client: Ion Popescu, Bucuresti

Insert client: Luca George, Bucuresti

Report client

Insert client: Sandu Vasile, Cluj-Napoca

Report client

Delete client: Ion Popescu, Bucuresti

Report client

Insert product: apple, 20, 1

Insert product: peach, 50, 2

Insert product: apple, 20, 1

Report product

Delete product: peach

Insert product: orange, 40, 1.5

Insert product: lemon, 70, 2

Report product

Order: Luca George, apple, 5

Order: Luca George, lemon, 5

Order: Sandu Vasile, apple, 100

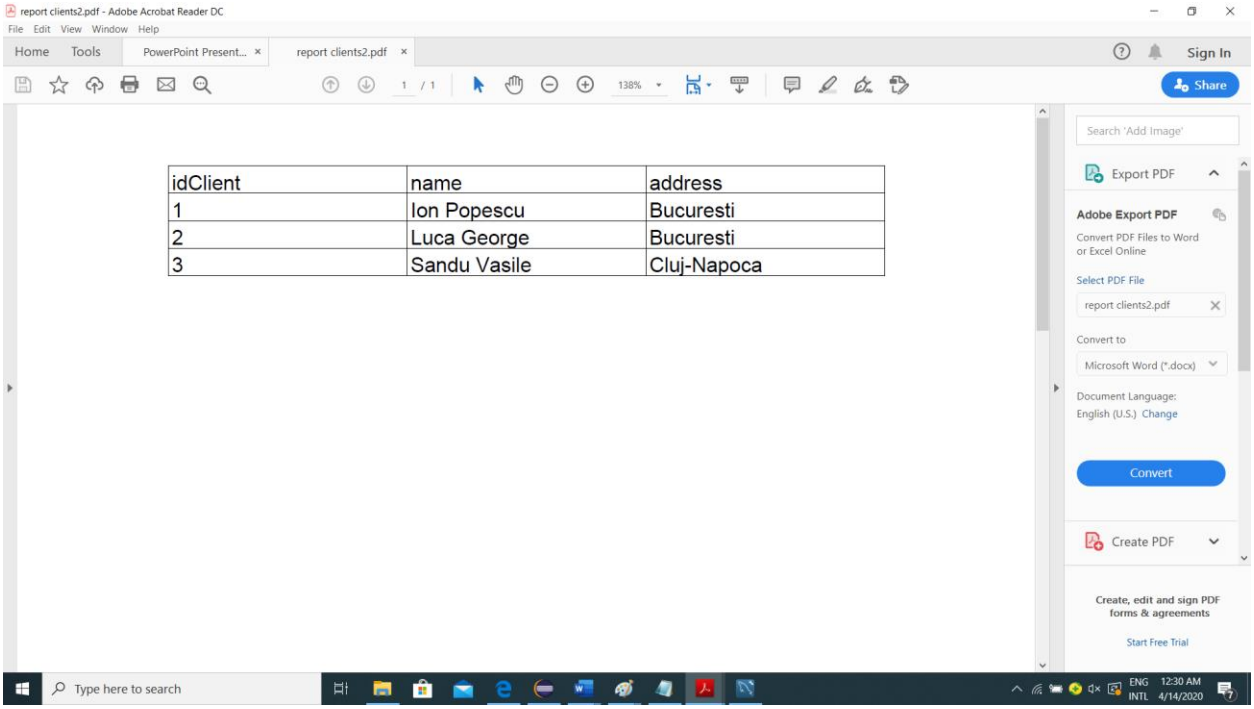
Report client

Report order

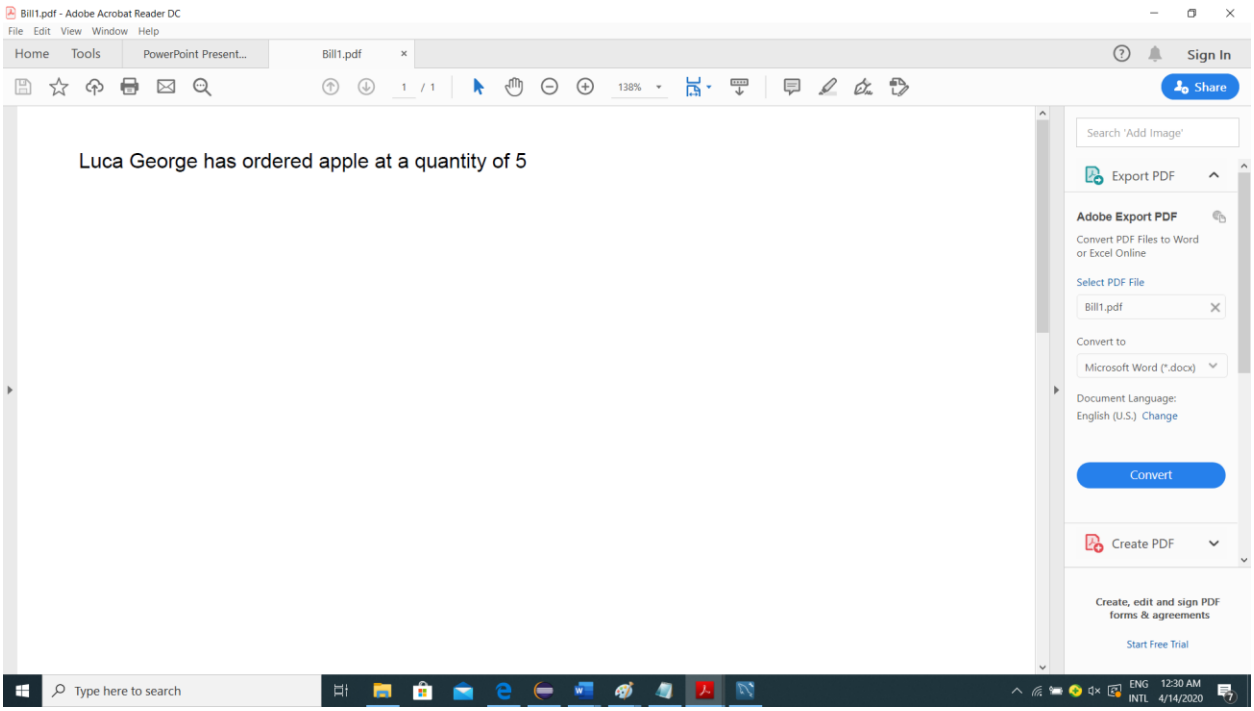
Report product

Dupa rularea acestui fisier datele au fost introduse in baza de date si s-au generat urmatoarele pdf-uri: 4 rapoarte pentru clienti, 3 rapoarte pentru produse, un raport pentru order, doua facturi si un pdf pentru stock insuficient.

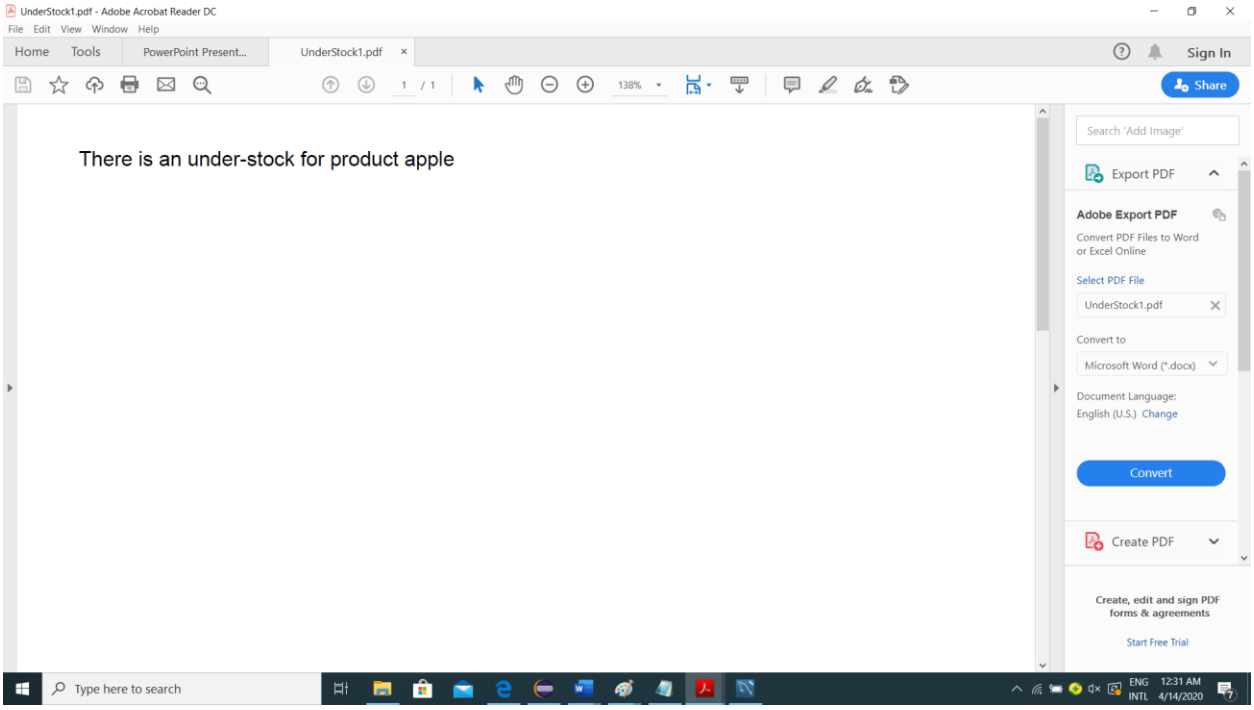
Al doilea raport pentru clienti



Prima facture



Mesaj de stock insuficient.



6 Concluzii

In concluzie aceasta aplicatie gestioneaza comenzile unor client folosind o baza de date. Ca si posibilitati de dezvoltare s-ar putea adauga mai multe tipuri de comenzi.

7 Bibliografie

https://www.baeldung.com/java-pdf-creation

http://tutorials.jenkov.com/java-reflection/index.html