Documentatie – Tema 3

Order Management

Nume: Dascal Raluca Georgiana

Grupa: 30226

Profesor coordonator: Dorin Moldovan

Cuprins:

1. Cerinte Functionale

Sa se realizeze o aplicatie care prin conectarea la o baza de date sa fie capabila sa adauge,,sa stearga si sa furnizeze anumite informatii despre elemtentele gasite in tabele acestei baze de date.

2. Obiective

2.1 Obiectiv Principal

Dezvoltarea unei aplicatii care sa stocheze intr-o baze de date informatii despre potentialii client, produsele din stoc, si comenzile pe care clientii le adauga. Aplicatia trebuie atat sa gestioneze tabelele din baza de date (adaugare,stergere actualizare a valorilor din tabele prezentate mai sus) cat si sa efectueze o factura (intru-un document pdf) sau sa afiseze un mesaj de eroare in cazul in care cantitatea ceruta pentru un produs este mai mare decat stocul disponibil. Pe langa aceasta functionalitate aplicatia trebuie sa mai respecte cateva reguli precum: folosirea programarii orientate pe obiect(concept aprofundat in semestrul trecut), crearea unor clase cu maxim 300 de linii de cod, implemantarea unor metode cu maxim 30 de linii de cod, folosirea conventiilor pentru nume Java, utilizarea unor fisiere text pentru citire si generarea unor documente PDF pentru returnarea detaliilor cerute, generarea fisierului JavaDoc precum si generarea unui fisier executabil .jar care sa respecte formatul de apel.

2.2 Obiective Secundare

Obiective secundare	Capitol
Descriera generala a aplicatiei	3
Descriera algoritmilor utilizati	5.2
Prezentarea claselor	6
Rezultatele obtinute	7

3. Descrierea Generala a Aplicatiei

Aceasta aplicatie este utila directorilor de centre comerciale deoarece permite stocarea informatiilor atat despre clientii lui, cat si despre produse si stoc. Aplicatia prezinta o baza de date dinamica, adica directorul poate cu usurinta adauga sau sterge un client sau un produs cand doreste. Actualizarea stocului se realizeaza automat in cazul in care directorul doreste sa introduca in stoc un produs deja existent, sau cand un client plaseaza o comanda. Mai mult de atat aceasta aplicatie genereaza la fiecare comanda plasaTA cate o facturA cu informatiile despre client,produs,cantitate si suma totala de plata. In cazul in care directorul doreste sa vada tabele, fara insa a accesa baza de date, el poate sa faca asta cu ajutorul unor intructiuni prezentate mai jos.

4. Analiza problemei

In figura de mai jos (fig 4.1) este descrisa intr-o forma grafica relatia dintre utilizator (directorii centrelor comerciale), aplicatie si furnizarea rezultatelor obtinute in urma parcurgerii unui algoritm. Am ales sa realizez aplicatie si in aceasta forma de prezentare pentru a fi mai usor de inteles pentru un potent utilizator atat din punct de vedere functional cat si din punct de vedere a datelor necesare pentru a putea fi utilizata.

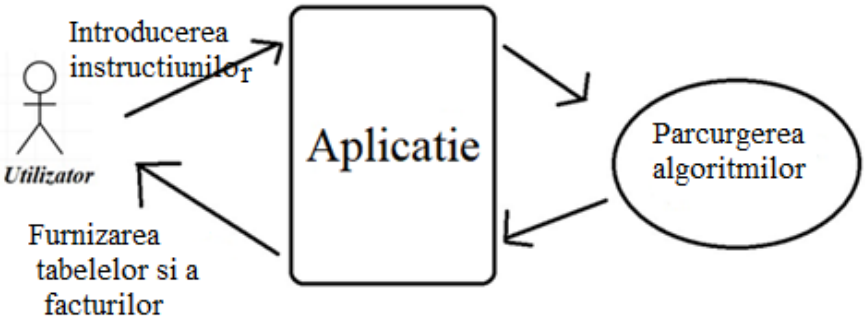


Fig 4.1

5. Proiectare

5.1 Diagrama de clase

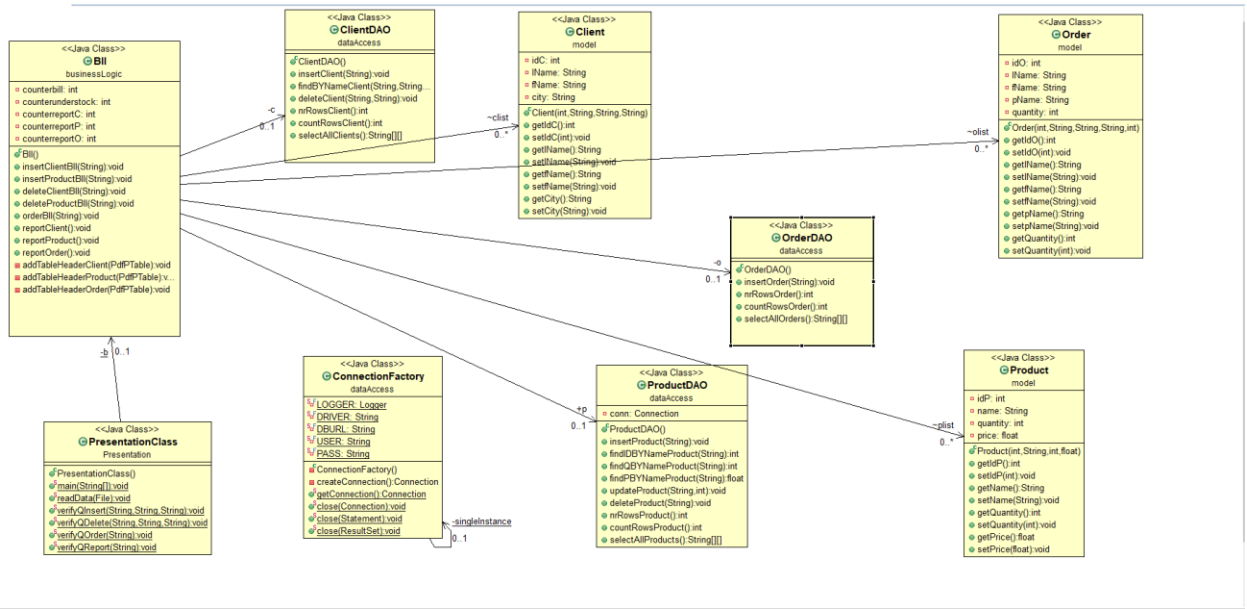


Fig 5.1.1

In figura de mai sus (fig 5.1.1) sunt prezentate diagramele UML a claselor proiectului realizat. Relatia dintre acestea a fost sintetizata de catre programul in care a fost scris si codul, Eclipse Java 2019. In aceasta imagine se evidentiaza atat fiecare clasa in parte (cu varabilele declarate si metodele ce definesc fie constructori, gettere ,settere fie algoritmi utili in realizarea functionalitatilor aplicatiei) cat si legaturile dintre acestea. Aceste diagrame UML ne ajuta la formarea unei imagini in ansamblu a proiectului ajutand totodata si la intelegerea unor concepte si paradigme a programarii orientate pe obiect.

5.2 Descrierea algoritmilor implementati

Fiecare dintre metodele definite si descrise in acesat aplicatie Java, cu exceptia constructorilor sau a celor care seteaza si returneaza valorile variabilelor declarate in clase, descriu cate o sarcina pe care proiectul trebuie sa o indeplineasca folosind un algoritm mai mult sau mai putin riguros.Algoritmii implementati in acest proiect au o saricna foarte bine definita si urmaresc realizarea functionalitatii corecte a cerintelor acestei teme. Descrierea implementarii si functionalizarii fiecare metode si utilitatea fiecarei variabile (aparute in diagram UML a claselor prezentata in capitolul anterior) vor fi prezentate in amanunt in urmatorul capitol.

6. Implementare

In acest capitol se va prezenta fiecare clasa a proiectului si se va explica utilitatea sau scopul fiecarei variabile declarate si metode implementate in aceste proiect avand la baza diagrama UML sintetizata de catre programul in care a fost scris codul.S-au creat 4 pachete diferite, iar fiecare isi indeplineste functionalitatile.

6.1. Clasa Client

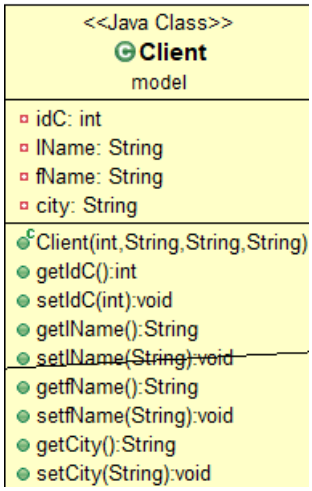


Fig 6.1

In figura alaturata (fig 6.1) este prezentata clasa Client, care reprezinta una dintre cele 3 clase de temelie a proiectului. In aceasta clasa sunt prezentate toate detaliile despre client precum: idC- reprezinta un identificator unic pentru fiecare persoana si se genereaza automat la adaugarea acetuia in baza de date, lName- reprezeinta numele de familie a clientului, iar fName- este prenumele acestuia, city- este orasul in care aceasta locuieste. Metodele definite in aceasta clasa sunt doar cele de baza, adica un constructor, gettere si settere pentru fiecare variabila explicata mai sus.

6.2. Clasa Product

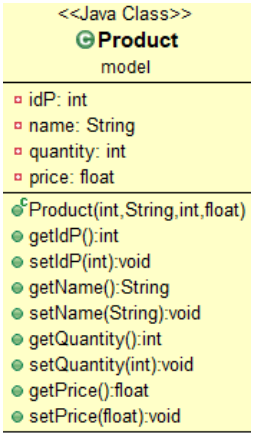


Fig 6.2

In figura alaturata (fig 6.2) este prezentata clasa Product, care reprezinta una dintre cele 3 clase de temelie a proiectului. In aceasta clasa sunt prezentate toate detaliile despre product precum: idP- reprezinta un identificator unic pentru fiecare produs si se genereaza automat la adaugarea acetuia in baza de date, name- denumirea produsului,, quantity- reprezinta cantitatea produsului disponibila in stoc, iar price- reprezinta pretul acestuia. Metodele definite in aceasta clasa sunt doar cele de baza, adica un constructor, gettere si settere pentru fiecare variabila explicata mai sus.

6.3. Clasa Order

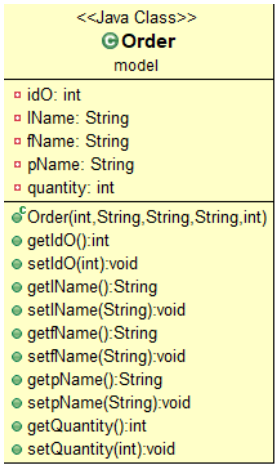


Fig 6.3

In figura alaturata (fig 6.3) este prezentata clasa Order care reprezinta una dintre cele 3 clase de temelie a proiectului. In aceasta clasa sunt prezentate toate detaliile despre comenzile plasate precum: idO- reprezinta un identificator unic pentru fiecare comanda si se genereaza automat la adaugarea acetuia in baza de date, lName, fName- reprezinta numele si prenumele persoanei care a plasat comnad, pName- reprezinta denumirea produslui comanat, iar quantity- reprezinta cantitatea produsului pe care clientul o doreste. Metodele definite in aceasta clasa sunt doar cele de baza, adica un constructor, gettere si settere pentru fiecare variabila explicata mai sus.

6.4. Clasa ConnectionFactory

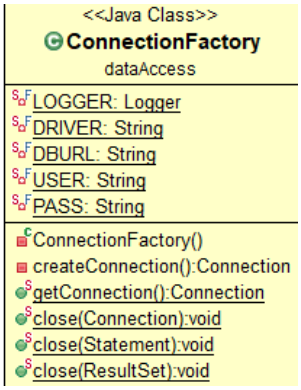
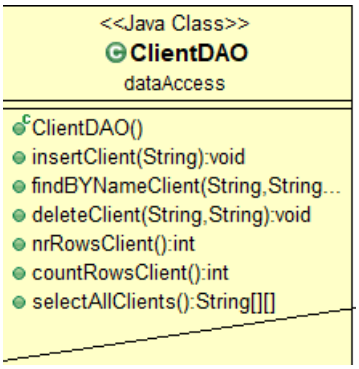


Fig 6.4

In figura 6.4 este prezentata clasa in care aplicatia mea se conecteaza la baza de date MySQL. Variabilele folosite: logger, driver, dburl, user si pass sunt toate necesare pentru a face posibila conectarea. Primele trei metode se refera la conectarea in sine, in timp ce ultimele 3 metode se refera la inchiderea atat a conexiunii, cat si a secventei de cod SQL si rezultatului furnizat de acesta.

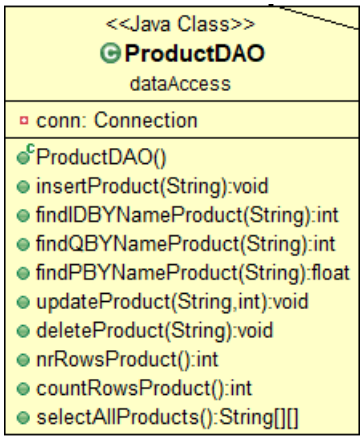
6.5.Clasa ClientDAO



In clasa din figura 6.5 este prezentata clasa ClientDAO, clasa in care sunt implementati toti algoritmii care se vor efectua pe baza de date. Metoda insertClient contine crearea secventei SQL si executarea acesteia pentru adaugarea unui client in tabelul corespunzator. Metoda findByNameClient returneaza identificatorul unic al clientului cu numele si prenumele dat, deleteClient este metoda in care se efectueaza stergerea clientului din baza de date a clientului cu prenumele si numele dat. Metoda nrRowsClient este metoda care returneaza identificatorul ultimului client adaugat pentru a putea genera urmatoarii identificatori cu siguranta ca acestia sunt unici. countRowsClient este metoda care returneaza numarul de linii din tabelul Client, adica numarul de client existenti la un moment in tabel. Ultima metoda creeaza un tabel care contine toate informatiile existente in tabelul Client.

Fig 6.5

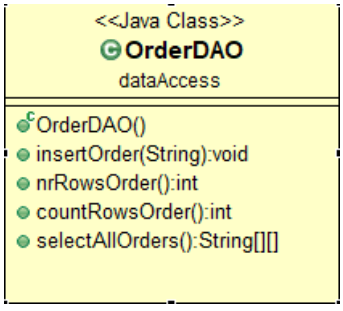
6.6.Clasa ProductDAO



In clasa din figura 6.6 este prezentata clasa ProductDAO, clasa in care sunt implementati toti algoritmii care se vor efectua pe baza de date. Metoda insertProduct contine crearea secventei SQL si executarea acesteia pentru adaugarea unui produs in tabelul corespunzator. Metoda findIDByNameProduct returneaza identificatorul unic al produsului cu denumirea data, metoda findQByNameProduct returneaza cantitatea produsului cu denumirea data, metoda findPByNameProduct returneaza pretul produsului cu denumirea data. Metoda updateProduct este metoda care actualizeaza cantitatea unui produs cu ajutorul metodei findQByName in cazul in care se realizeaza o adaugare in stoc, sau in cazul in care se plaseaza o comanda., deleteProduct este metoda in care se efectueaza stergerea produsului din baza de date a produsului cu denumirea data. Metoda nrRowsProduct este metoda care returneaza identificatorul ultimului produs adaugat pentru a putea genera urmatoarii identificatori cu siguranta ca acestia sunt unici. countRowsProduct este metoda care returneaza numarul de linii din tabelul Product, adica numarul de client existenti la un moment in tabel. Ultima metoda creeaza un tabel care contine toate informatiile existente in tabelul Product.

Fig 6.6

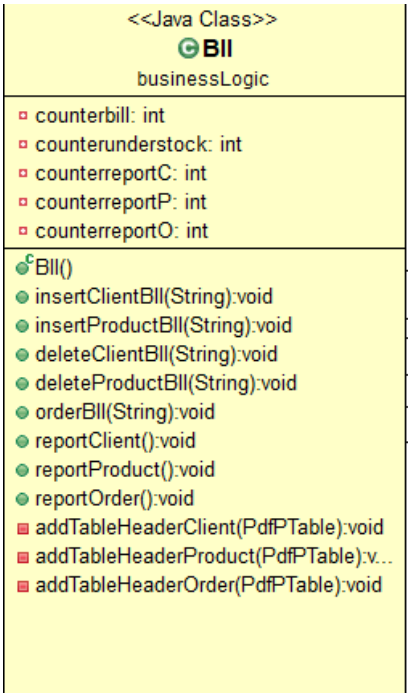
6.7.Clasa OrderDAO



In clasa din figura 6.7 este prezentata clasa OrderDAO, clasa in care sunt implementati toti algoritmii care se vor efectua pe baza de date. Metoda insertOrder contine crearea secventei SQL si executarea acesteia pentru adaugarea unei comenzi in tabelul corespunzator. Metoda nrRowsOrder este metoda care returneaza identificatorul ultimulei comenzi adaugate pentru a putea genera urmatoarii identificatori cu siguranta ca acestia sunt unici. countRowsOrder este metoda care returneaza numarul de linii din tabelul Order, adica numarul de comenzi existente la un moment in tabel. Ultima metoda creeaza un tabel care contine toate informatiile existente in tabelul Order.

Fig 6.7

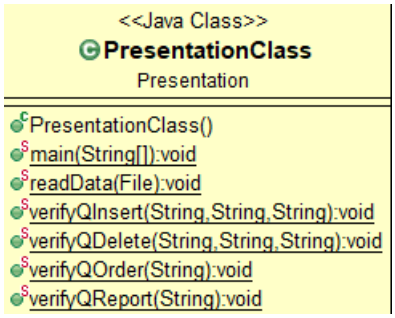
6.8.Clasa BLL



In clasa Bll se realizeaza parte de gestionare a programului. Variabilele prezene in figura din dreapta semnifica niste contoare pentru a putea genera numele documentelor PDF astfel incat sa fie diferte. Metodele insertClientBll, insertProductBll, deleteClientBll, deleteProductBll este apelata din clasa centrala a programului si apeleaza la randul ei metode din clasa prezentate mai sus in functie de onstructiunea introdusa. In metoda orderBll se apeleaza mertoda de adaugare din OrderDAO si genereaza niste documente PDF care contin un fel de facturi cu informatii despre client, produs, cantitatea ceruta si pretul total,sau in cazul in care nu exista disponibilitate in stoc se va afisa un mesaj corespunzator. Metodele reportClient, reportOrder, reportProduct genereaza si ele documente PDF care contin tabele din baza de date la un moment dat, in timp ce metodele addTableHeader* vin in ajutorul metodelor prezentate mai sus si genereaza capul tabelor create acolo.

Fig 6.8

6.9.Clasa PresentationClass



Aceasta clasa contine si metoda main in care se apeleza metoda readData care realizeaza cititrea informatiilor dintr-un fisier text. Celalalte metode apeleaza metodele din clasas de control Bll in functie de instructiunile citite din fisier.

Fig 6.9

7.Rezultate

George Luca a comandat 5 apple-Total= 5.0 RON

Fig 7.1

Vasile Sandu a comandat 100 apple -Stoc epuizat

Fig 7.2

idC	lName	fName	city
1	Popescu	Ion	Bucuresti
2	George	Luca	Bucuresti
3	Vasile	Sandu	Cluj-Napoca

Fig 7.3

In figura 7.1 este prezentat rezultatul obtinut pentru generearea unei facutri. In urmatoarea figura este un exemplu de mesaj pentru un stoc insufficient, in timp ce in ultima figura este unul dintre documentele generate utilizand instructiunea Report.

In aceste figuri se pot verifica si corectitudinile rezultatelor obtinute.

8. Concluzii si posibilitati de dezvoltare ulterioare

In timpul dezvoltarii acestei teme, adica in timpul realizarii unei aplicatii care stocheaza informatii intr-o baza de date legate de client, produse si comenzi si permite gesionarea ei, consider ca am reusit sa aprofundez atat unele paradigme si concepte ale programarii orientate pe obiect cat si imbunatariea abilitatile mele de a scrie cod si/ sau a corecta eventualele erori atat de sintaxa cat si la nivel de gandire aparute pe parcursul descrierii algoritmilor necesare functionalitatilor acestei aplicatii. Pe langa acestea am realizat cat de importanta este citirea cu atentie si intelegerea unei cerinte si capabilitatea de a structura/ imparti problema initiala in mai multe probleme mai mici si inceperea rezolvarii acestora ca un prim pas in indeplinirea sarcinilor unei teme,ca apoi sa asociezi toate rezultatele obtinute din surse pentru a ajunge la un rezultat corect in ansamblu.. Mi-am imbunatatit si abilitatea de a cauta informatii utile si corecte pentru a reusi ducerea proiectului meu intr-o stare mai complexa si mai corecta din punct de vedere a functionalitatii.

O posibilitate de dezvoltare a proiect realizat si descris in acesta documentatie ar putea fi:

- Adaugarea in baza de date a mai multor tabele care pot stoca date suplimentare despre oricare dintre elemente
- mai mult de atat s-ar putea face o revizuire asupra tuturor algoritmilor folositi si realizarea implementarii acestora intr-o maniera mai eficienta din punct de vedere a timpului de executie sau a memoriei folosite sau mai usor de descris (pentru programator) folosind functii deja definite ale pachetelor Java, astfel reusind o reducere a timpului petrecut pentru asigurarea functionarii unui astfel de aplicatii.
- o alta posibilitate de dezvoltare a proiectului ar putea fi imbunatatirea proiectului realizand chiar si o interfata grafica pentru ca utilizatorului sa-I fie cat mai usor intelegerea rezultatelor si a utilizarii aplicatiei

9. Bibliografie

https://conspecte.com/Programare-Java/java-citirea-datelor-de-la-tastatura-si-afisarea-lor.html

https://www.w3schools.com/java/java_arrays.asp

https://mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/

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

https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html

https://ibytecode.com/blog/jdbc-mysql-create-database-example/