**TEHNICI DE PROGRAMARE FUNDAMENTALE**



**ASSIGNMENT III:**

**ORDER MANAGEMENT**

**Documentatie**

STUDENT: MARCU MIHAI-ALEXANDRU

GRUPA: 30227

**CUPRINS:**

1. Obiectivul temei;
2. Analiza problemei, modelare, scenarii si cazuri de utilizare;
3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator);
4. Implementare:

Pachete:

* 1. connection:

1. ConnectionFactory;
   1. dataAccessLayer:
      * 1. ClientDao;
        2. ProdusDao;
        3. OrderItemDao;
        4. OrderDao;
   2. model:
2. Client;
3. Produs;
4. OrderItem;
5. Order;
   1. presentation:
6. FileParse;
7. Presentation;
   1. start:
8. MainClass;
9. Concluzii;
10. Bibliografie.
11. Obiectivul temei:

Obiectivul principal al temei este realizarea unei aplicatii care proceseaza comenzile unor clienti pentru un depozit.

Obiectivele secundare ale acestei teme sunt:

1. Utilizarea unei baze de date in care sunt salvate datele despre client, produse si comenzi;
2. Realizarea conexiunii dintre baza de date descrisa mai sus si aplicatie;
3. Realizarea unei Java Archive;
4. Citirea comenzilor dintr-un fisier si implementarea acestora in program dupa standardele SQL;
5. Creearea unor PDF-uri in care sunt salvate informatii;
6. Creearea unui Javadoc.
7. Analiza problemei, modelare, scenarii si cazuri de utilizare:

Comenzile pentru manipularea datelor se citesc din fisierul numit „In.txt” care este dat ca parametru in metoda main. Comenzile sunt urmatoarele:

* Insert client/product: insereaza in tabela respectiva un rand nou cu datele specificate in fisier. Querry-ul SQL este: „INSERT INTO Tabel(coloanele tabelei) + VALUES(datele de inserat)”;
* Delete client/product: sterge din baza de date randul care are numele specificat in fisier. Querry-ul SQL este: „DELETE FROM Tabel where coloanaconditie=?(de exemplu id=1)”;
* Order: insereaza in tabela orideritem o comanda noua cu datele specificate dupa „:” in fisier. Querry-ul SQL este: „INSERT INTO ordertem(coloanele tabelei) + VALUES(datele de inserat)”;
* Report client/product/order: genereaza un PDF sub forma de tabel care contine toti clienti/produse/comenzi cu toate caracteristicile lor de la momentul actual.

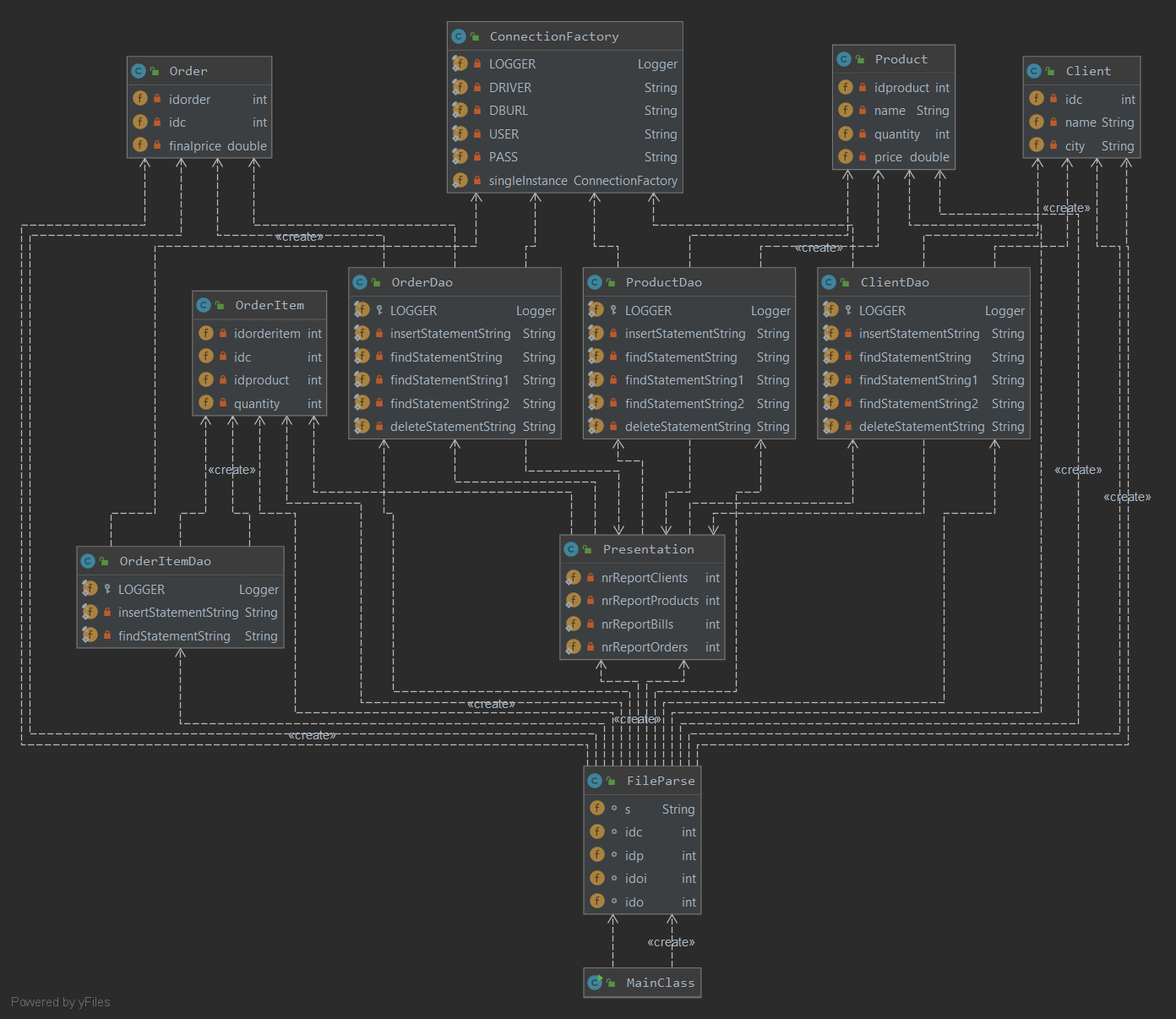
1. Proiectare( decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, pachete, algoritmi, interfata utilizator):

In prima instanta am creeat baza de date in care sunt salvate informatiile despre clienti, produse si comenzi prin intermediul workbench-ului MySQL si am populat aceasta baza de date cu 4 tabele: client, product, orderitem si order. Cheia primara a fiecarui tabel este reprezentata de catre un id specific. Aceasta actiune a fost urmata de creearea clasei care realizeaza conexiunea dintre baza de date si program. Urmatorul pas a fost realizarea claselor din pachetul model ce reprezinta datele din tabele: Client, Product, OrderItem si Order. Dupa aceea am creeat clasele care manipuleaza datele din tabele, cele din pachetul dataAccessLayer. Urmatorul pas a fost realizarea celor 2 clase din pachetul presentation, una in care este realizata parsarea din fisierul dat a comenziilor cerute de manipulare a tabelelor, iar in a doua clasa sunt creeate PDF-urile specifice cerintelor. Dupa aceea am creeat clasa MainClass care contine metoda main si care are ca parametru numele fisierului pentru care se realizeaza functia de parse din clasa precizata mai sus. La final am realizat arhiva .jar specifica proiectului.

Proiectul este structurat in 5 pachete:

* 1. connection: acest pachet contine clasa care realizeaza conexiunea dintre baza de date si aplicatie;
  2. dataAccessLayer: este alcatuit din clasele care manipuleaza datele din tabelele din baza de date, creeind metode pentru interogarile SQL necesare;
  3. model: contine cate o clasa pentru fiecare tabela din baza de date, fiecare clasa avand ca variabile coloanele din tabela specifica;
  4. presentation: acest pachet contine clasa in care se realizeaza parsarea datelor din fisierul ce contine comenziile cerute, dar si o clasa in care sunt realizate PDF-urile cerute;
  5. start: contine calsa MainClass in care se regaseste metoda main.

Diagrama UML a claselor este urmatoarea:



1. Implementare:
2. Pachetul connection:
3. Clasa ConnectionFactory: Aceasta clasa realizeaza conexiunea dintre proiect si baza de date.

Metodele acestei clase sunt:

* createConnection: returneaza un obiect de tipul Connection ce reprezinta conexiunea cu baza de date;
* getConnection: aceasta metoda este un getter pentru obiectul de tip Connection ce reprezinta conexiunea realizata cu baza de date;
* close(Connection connection): aceasta metoda inchide conexiunea specificata ca parametru;
* close(Statement statement): aceasta metoda inchide statement-ul specificat ca parametru;
* close(ResultSet resultSet): aceasta metoda inchide setul de rezultate dat ca parametru.

1. Pachetul model:
2. Clasa Client: In aceasta clasa sunt descrise caracteristicile unui client, fiind salvate in variabile: id-ul clientului: idc, numele: name si orasul in care locuieste: city.

Metodele acestei clase sunt urmatoarele:

* Un constructor care initializeaza un client nou cu datele oferite ca parametrii;
* Metode getter pentru id-ul, numele si orasul clientului;
* O metoda toString care realizeaza transforamrea unui obiect de tipul Client intr-unul de tip String.

1. Clasa Product: In aceasta clasa sunt descrise caracteristicile unui produs, acestea fiind salvate in variabile: id-ul produsului: idproduct, numele acestuia: name, cantitatea disponibila in stoc: quantity si pretul unei bucati: price, care este de tipul double.

Metodele acestei clase sunt urmatoarele:

* Un constructor care initializeaza un produs nou cu datele oferite ca parametrii;
* Metode getter pentru id-ul, numele, cantitatea si pretul produsului;
* Metode setter pentru id-ul si cantitatea produsului;
* O metoda toString care realizeaza transforamrea unui obiect de tipul Product intr-unul de tip String.

1. Clasa OrderItem: In aceasta clasa sunt descrise caracteristicile unei comenzi, acestea fiind salvate in variabile: id-ul comenzii: idorderitem, id-ul clientului care a realizat comanda: idc, id-ul produsului cumparat: idproduct si cantitatea in care acesta a fost solicitat: quantity.

Metodele acestei clase sunt urmatoarele:

* Un constructor care initializeaza o comanda noua cu datele oferite ca parametrii;
* Metode getter pentru id-ul comenzii, id-ul clientului, id-ul produsului cumparat si pentru cantitatea de produs solicitata;
* toString: este o metoda de transformare al unui obiect de tipul OrderItem intr-unul de tip String.

1. Clasa Order: In aceasta clasa sunt descrise caracteristicile unei comenzi finale, acestea fiind salvate in variabile: id-ul comenzii: idorder, id-ul clientului care a realizat comanda: idc si pretul final: finalprice, care este de tipul double.

Metodele acestei clase sunt urmatoarele:

* Un constructor care initializeaza o comanda finala noua cu datele oferite ca parametrii;
* Metode getter pentru id-ul comenzii finale, id-ul clientului si pentru pretul final pe care clientul trebuie sa-l plateasca;
* Metoda setter pentru pretul final de platit de catre client;
* toString: metoda de transformare al unui obiect de tipul Order intr-unul de tip String.

1. Pachetul dataAccessLayer:
2. Clasa ClientDao: Are scopul de a manipula datele din tabela client. Aceasta clasa contine un obiect de tipul Logger specific ei si 5 obiecte de tipul String care sunt utilizate pentru executarea querry-urilor din SQL pentru inserare, cautare si stergere.

Metodele acestei clase sunt urmatoarele:

* findById: realizeaza cautarea in tabela client a unui rand care are idc egal cu cel precizat ca parametru si returneaza obiectul de tip Client cu id-ul respectiv daca acesta a fost gasit sau null in caz contrar. Aceasta cautare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "SELECT \* FROM client where idc = ?";
* findByName: realizeaza cautarea in tabela client a unui rand care are name egal cu cel precizat ca parametru si returneaza obiectul de tip Client cu numele respectiv daca acesta a fost gasit sau null in caz contrar. Aceasta cautare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "SELECT \* FROM client where name = ?";
* findReport: Realizeaza inserarea clientilor din tabela client in tabelul PDF specific clientilor cu ajutorul functiei addRowsClient din clasa Presentation. Aceasta inserare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "SELECT \* FROM client";
* Metoda insert: Insereaza un client dat ca parametru in tabela client si returneaza id-ul acestuia daca inserarea a avut succes sau -1 in caz contrar. Aceasta inserare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "INSERT INTO client (idc,name,city)" + " VALUES (?,?,?)";
* delete: aceasta metoda realizeaza stergerea unui client dat ca parametru din tabela client, folosindu-se de id-ul acestuia si returneaza id-ul clientului sters daca stergerea s-a realizat cu succes sau -1 in caz contrar. Aceasta stergere a fost realizata prin intermediul obiectului de tip String specific querry-ului: " DELETE FROM client WHERE idc = ?";

1. Clasa ProductDao: Are scopul de a manipula datele din tabela product. Aceasta clasa contine un obiect de tipul Logger specific ei si 5 obiecte de tipul String care sunt utilizate pentru executarea querry-urilor din SQL pentru inserare, cautare si stergere.

Metodele acestei clase sunt urmatoarele:

* findById: realizeaza cautarea in tabela product a unui rand care are idproduct egal cu cel precizat ca parametru si returneaza obiectul de tip Product cu id-ul respectiv daca acesta a fost gasit sau null in caz contrar. Aceasta cautare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "SELECT \* FROM product where idproduct = ?";
* findByName: realizeaza cautarea in tabela product a unui rand care are name egal cu cel precizat ca parametru si returneaza obiectul de tip Product cu numele respectiv daca acesta a fost gasit sau null in caz contrar. Aceasta cautare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "SELECT \* FROM product where name = ?";
* findReport: Realizeaza inserarea produselor din tabela product in tabelul PDF specific produselor cu ajutorul functiei addRowsProduct din clasa Presentation. Aceasta inserare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "SELECT \* FROM product";
* Metoda insert: Insereaza un produs dat ca parametru in tabela product si returneaza id-ul acestuia daca inserarea a avut succes sau -1 in caz contrar. Aceasta inserare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "INSERT INTO product (idproduct,name,quantity,price)" + " VALUES (?,?,?,?)";
* delete: aceasta metoda realizeaza stergerea unui produs dat ca parametru din tabela product, folosindu-se de id-ul acestuia si returneaza id-ul produsului sters daca stergerea s-a realizat cu succes sau -1 in caz contrar. Aceasta stergere a fost realizata prin intermediul obiectului de tip String specific querry-ului: " DELETE FROM product WHERE idproduct = ?";

1. Clasa OrderItemDao: Are scopul de a manipula datele din tabela orderitem. Aceasta clasa contine un obiect de tipul Logger specific ei si 2 obiecte de tipul String care sunt utilizate pentru executarea querry-urilor din SQL pentru inserare si cautare.

Metodele acestei clase sunt urmatoarele:

* findById: realizeaza cautarea in tabela orderitem a unui rand care are idorderitem egal cu cel precizat ca parametru si returneaza obiectul de tip OrderItem cu id-ul respectiv daca acesta a fost gasit sau null in caz contrar. Aceasta cautare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "SELECT \* FROM orderitem where idorderitem = ?";
* Metoda insert: Insereaza o comanda primita ca parametru in tabela orderitem si returneaza id-ul acesteia daca inserarea a avut succes sau -1 in caz contrar. Aceasta inserare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "INSERT INTO orderitem (idorderitem,idc,idproduct,quantity)" + " VALUES (?,?,?,?)";

1. Clasa OrderDao: Are scopul de a manipula datele din tabela orders. Aceasta clasa contine un obiect de tipul Logger specific ei si 5 obiecte de tipul String care sunt utilizate pentru executarea querry-urilor din SQL pentru inserare, stergere si cautare.

Metodele acestei clase sunt urmatoarele:

* findById: realizeaza cautarea in tabela orders a unui rand care are idorder egal cu valoarea precizata ca parametru si returneaza obiectul de tip Order cu id-ul respectiv daca acesta a fost gasit sau null in caz contrar. Aceasta cautare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "SELECT \* FROM orders where idorder = ?";
* findById: realizeaza cautarea in tabela orders a unui rand care are idc egal cu valoarea precizata ca parametru si returneaza obiectul de tip Order cu id-ul respectiv daca acesta a fost gasit sau null in caz contrar. Aceasta cautare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "SELECT \* FROM orders where idc = ?";
* findReport: Realizeaza inserarea comenzilor finale din tabela orders in tabelul PDF specific acestora cu ajutorul functiei addRowsOrder din clasa Presentation. Aceasta inserare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "SELECT \* FROM orders";
* Metoda insert: Insereaza o comanda primita ca parametru in tabela orders si returneaza id-ul acesteia daca inserarea a avut succes sau -1 in caz contrar. Aceasta inserare a fost realizata prin intermediul obiectului de tip String specific querry-ului: "INSERT INTO orders (idorder,idc,finalprice)" + " VALUES (?,?,?)";
* delete: aceasta metoda realizeaza stergerea unei comenzi finale date ca parametru din tabela orders, folosindu-se de id-ul acesteia si returneaza id-ul ei daca stergerea s-a realizat cu succes sau -1 in caz contrar. Aceasta stergere a fost realizata prin intermediul obiectului de tip String specific querry-ului: "DELETE FROM orders WHERE idorder = ?";

1. Pachetul presentation:
2. Clasa FileParse: In aceasta clasa sunt citite comenzile din fisier ce sunt utilizate pentru manipularea bazei de date.

Aceasta clasa contine metodele:

* Constructorul clasei, in acesta se initializeaza un sir de caractere, acesta este utilizat pentru citirea din fisier, s1 fiind numele fisierului care este dat ca parametru al metodei main din clasa MainClass si se apeleaza functia readFromFile;
* readFromFile: In aceasta functie am deschis fisierul si cu ajutorul expresiilor regex am impartit continutul acestuia. Tot aici se apeleaza si functiile din alte clase care realizeaza fiecare comanda. Citirea din fisier am realizat-o prin intermediul unui scanner.

1. Clasa Presentation: In aceasta clasa am creeat PDF-urile specifice fiecarui client, produs si fiecarei comenzi si facturi.

Aceasta clasa contine cate 3 metode pentru fiecare tabela. Prima este cea principala numita docPdf plus numele obiectului corespunzator tabelei si in ea este creat un obiect nou de tipul Document si un obiect nou de tipul PdfPTable, dupa care sunt apelate celelalte 2 metode: addTableHeader plus numele obiectului: creeaza header-ul tabelului din PDF si addRows plus numele obiectului: se ocupa cu introducerea datelor pe fiecare rand din tabel.

1. Pachetul start: Contine clasa MainClass care contine la randul ei metoda main in care creeaza un obiect nou de tipul fileParser si ii da ca parametru un string care reprezinta numele fisierului din care trebuie citite comenzile, string care este parametru al metodei main.
2. Concluzii:

Dupa realizare acestui proiect, personal mi-am imbunatatit abilitatile Programarii Orientate pe Obiect si cele in lucrul cu baze de date. Aditional, am invatat sa realizez PDF-uri in program si sa creez un Javadoc.

1. Bibliografie:

* Mare parte din proiect a fost realizata dupa exemplul oferit in sectiunea temei, cel gasit pe contul de bitbucket al utcn\_dsrl: <https://utcn_dsrl@bitbucket.org/utcn_dsrl/pt-layered-architecture.git>;
* https://utcn\_dsrl@bitbucket.org/utcn\_dsrl/pt-reflection-example.git;
* [https://stackoverflow.com](https://stackoverflow.com/questions/10961714/how-to-properly-stop-the-thread-in-java);
* <https://en.wikipedia.org/wiki/Singleton_pattern>;
* Cunostinte dobindite pe parcursul laboratorului de baze de date din anul II semestrul I;
* <https://www.baeldung.com/executable-jar-with-maven>;
* <https://www.baeldung.com/java-pdf-creation>;
* https://www.baeldung.com/javadoc;
* https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html.