DOCUMENTATIE TEMA 3

Order Management

Pop Alin

Grupa 30227

# Cuprins

1. Obiectivul temei . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
2. Analiza problemei, modelare, scenarii . . . . . . . . . . . . . . 4
3. Proiectare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
4. Implementare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
5. Concluzii si posibile dezvoltari ulterioare . . . . . . . . . . . . 7

# Cerinte fundamentale

## Obiectivul principal

Creeare unui program care sa acceseze o baza de date creeata de noi , care sa poata face operatii de insert , delete , update , select si select\* pentru datele din baza de date.

## Obiective secundare

* Dezvoltarea de use case-uri si scenarii
* Alegerea structurilor de date
* Impartirea pe clase
* Dezvoltarea algoritmilor
* Implementarea solutiei
* Testare

# 1.Analiza problemei

Utilizatorul trebuie sa comunice cu interfata creeata de noi pentru a introduce datele necesare rularii programului . In functie de ce operatie doreste sa execute datele pe care le poate introduce utilizatorul sunt : idclient , numeclient , prenumeClient , varstaClient , idProdus , stoc , pret , numeProdus si cantitatea pe care doreste sa o comande dintr-un produs .

Scenariul in care programul va rula cu success :

* Utilizatorul introduce datele necesare operatiei pe care doreste sa o execute
* Utilizatorul apasa pe butonul destinat operatiei pe care doreste sa o execute
* Baza de date este actualizata
* Datele introduce de utilizator sunt valide

Scenarii in care programul nu va rula cu success :

* Utilizatorul nu va introduce date valide pentru inceperea programului

# Proiectare

Proiectul contine 14 clase , acestea fiin impartite in 7 pachete .

Pachete pe care le regasim in proiect sunt :

- connection : este pachetul in care regasim functia cu care facem legatura dintre proietul java si baza de date

- model : este pachetul in care regasim clasele care mapeaza tabelele din baza de date

- dao : este pachetul in care regasim clasele care acceseaza datele din baza de date sub forma de obiecte

- bll.validators : este clasa in care regasim o interfata si o clasa care implementeaza aceasta interfata , cu scopul de a valida datele pe care le folosim

- bll : este pachetul in care regasim calsele care acceseaza metodele din DAO

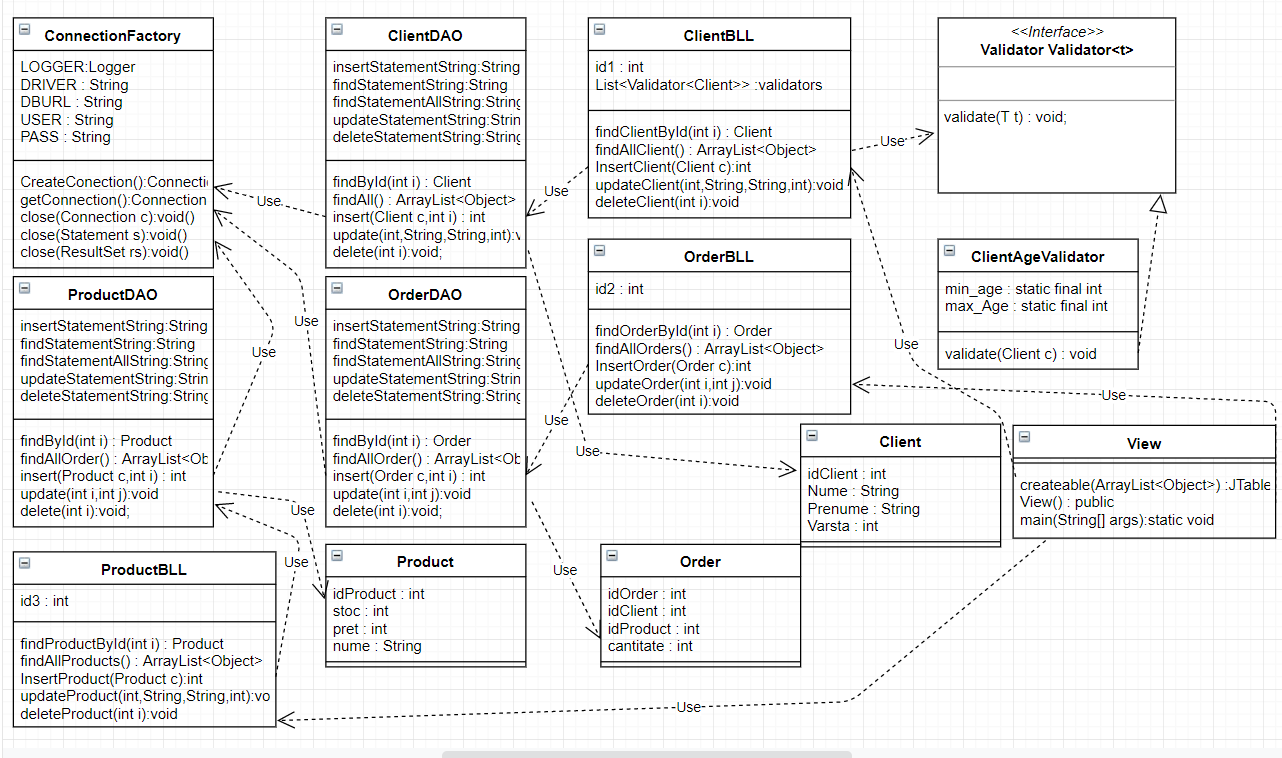
- presentation : este pachetul in care este implementata interfata cu utilizatorul

- start : este clasa din care se porneste programul

Clasele pe care le regasim in proiect sunt :

* ConnectionFactory
* Client
* Product
* Order
* ClientDAO
* OrderDAO
* ProductDAO
* ClientBLL
* OrderBLL
* ProductBLL
* Validator
* ClientAgeValid
* View
* Start

Diagrama UML :



# Implementare

## ConnectionFactory

Aceasta clasa are rolul de a face conexiunea cu baza de date . In aceasta clasa regasim urmatoarele variabile : LOGGER , DRIVER , DBURL , USEr , PASS , toate fiind de tipus private static final String , rolul lor fiind Acela de a stabili conexiunea cu abza de date .

Metoda createConnection ( ) are rolul de a creea conexiunea cu baza de date.

Metoda getConnection ( ) este cea care returneaza conexiunea cu baza de date fiind folosita in toate operatille ( select , insert , delete , update ) .

Metodele de close ( ) au rolul de a inchide conexiunea cu baza de date .

## Client

Aceasta clasa are rolul de a mapa bazele din tabela client pe care o regasim in baza de date . Atributele pe care le poseda aceasta clasa sunt : int idClient , String Nume , String Prenume , int Varsta . Aceste attribute reprezinte numele coloanelor din tabela Client existente in baza de date .

Aceasta clasa are trei constructori , unul fara parametric , unul cu parametric Client(String n , String p , int v) care este folosit la inserare , deoarece se face inserare cu in id aleator , si un constructor cu parametrii int id , String n ,String p , int v , folosit pentru afisarea tuturor datelor din tabela Client.

Dupa aceasta avem gettere si settere pentru toate campurile din clasa , folosite pentru a seta respective obtine valorile campurilor .

Ultima metoda este suprascriereea metodei toString() pentru a afisa Clientii intr-o maniera mai placuta.

## Order

Aceasta clasa are rolul de a mapa bazele din tabela client pe care o regasim in baza de date . Atributele pe care le poseda aceasta clasa sunt : int idOrder , int idClient , int idProduct , int cantitate . Aceste attribute reprezinte numele coloanelor din tabela Client existente in baza de date .

Aceasta clasa are trei constructori , unul fara parametric , unul cu parametric Order(int idc , int idp , int cantitate) care este folosit la inserare , deoarece se face inserare cu in id aleator , si un constructor cu parametrii int id , int idc , int idp , int cantitate , folosit pentru afisarea tuturor datelor din tabela Order.

Dupa aceasta avem gettere si settere pentru toate campurile din clasa , folosite pentru a seta respective obtine valorile campurilor .

Ultima metoda este suprascriereea metodei toString() pentru a afisa Orderele intr-o maniera mai placuta.

## Product

Aceasta clasa are rolul de a mapa bazele din tabela client pe care o regasim in baza de date . Atributele pe care le poseda aceasta clasa sunt : int idProduct , int pret , int stoc , String nume . Aceste attribute reprezinte numele coloanelor din tabela command existente in baza de date .

Aceasta clasa are trei constructori , unul fara parametric , unul cu parametric Product(int stoc , int pret , String nume) care este folosit la inserare , deoarece se face inserare cu in id aleator , si un constructor cu parametrii int idP , int pret , int stoc , String nume , folosit pentru afisarea tuturor datelor din tabela Order.

Dupa aceasta avem gettere si settere pentru toate campurile din clasa , folosite pentru a seta respective obtine valorile campurilor .

Ultima metoda este suprascriereea metodei toString() pentru a afisa produsele intr-o maniera mai placuta.

## ClientDAO

In aceasta clasa se realizeaza accesul la date prin obiecte ( DAO = data acces object ) . Variabilele pe care le regasim in aceasta clasa sunt insertStatementString , findStatementString , findStatementAllString , si deleteStatementString . Toate aceste variabile sunt de tip String , iar in ele se retirn queryurile care se executa asupra bazei de date , adica insert , delete , update , si select.

Metoda findById are un parametru de tip int , clientId , si returneaza clientul din baza de date care are id-ul egal cu parametrul functiei . La inceput se face conexiunea cu baza de date , dupa care prin ResulSet se executa comanda din findStatementString , se retin numele , prenumele , si varsta pentru client , iar la final daca este un client cu id-ul respective acesta este returnat.

Metoda findAll() nu are nici un parametru si are rolul de a return un ArrayList de obiecte , in care se introduce toate datele din tabela Client .

Metoda insert , care are ca parametric un client si un id , insereaza in tabela un nou client cu caracteristicile clientului transmis ca parametru , si cu id-ul clientului egal cu intregul transmis ca parametru in functie .

Metoda update are ca parametric doi intregi si doua Stringuri . Aceasta face un update in baza de date , astfel ca , Clientului cu id-ul egal cu primul intreg transmis ca parametru ii sunt modificate numele , prenumele si varsta cu valorile variabilelor transmise ca parametric functiei .

Metoda delete are ca parametru un intreg , aceasta stergand din baza de date clientul care are id-ul egal cu valoarea intregului transmis ca si parametru fucntiei .

## OrderDAO

In aceasta clasa se realizeaza accesul la date prin obiecte ( DAO = data acces object ) . Variabilele pe care le regasim in aceasta clasa sunt insertStatementString , findStatementString , findStatementAllString , si deleteStatementString . Toate aceste variabile sunt de tip String , iar in ele se retirn queryurile care se executa asupra bazei de date , adica insert , delete , update , si select.

Metoda findById are un parametru de tip int , ordertId , si returneaza orderul din baza de date care are id-ul egal cu parametrul functiei . La inceput se face conexiunea cu baza de date , dupa care prin ResulSet se executa comanda din findStatementString , se retin idClient , idProduct , si cantitatea pentru order , iar la final daca este un order cu id-ul respective acesta este returnat.

Metoda findAll() nu are nici un parametru si are rolul de a return un ArrayList de obiecte , in care se introduce toate datele din tabela Order .

Metoda insert , care are ca parametric un order si un id , insereaza in tabela un nou order cu caracteristicile orderului transmis ca parametru , si cu id-ul orderului egal cu intregul transmis ca parametru in functie .

Metoda update are ca parametric doi intregi. Aceasta face un update in baza de date , astfel ca , orderului cu id-ul egal cu primul intreg transmis ca parametru ii este modificata cantitatea cu valoarea variabilelor transmise ca parametri functiei .

Metoda delete are ca parametru un intreg , aceasta stergand din baza de date orderul care are id-ul egal cu valoarea intregului transmis ca si parametru fucntiei .

## ProductDAO

In aceasta clasa se realizeaza accesul la date prin obiecte ( DAO = data acces object ) . Variabilele pe care le regasim in aceasta clasa sunt insertStatementString , findStatementString , findStatementAllString , si deleteStatementString . Toate aceste variabile sunt de tip String , iar in ele se retirn queryurile care se executa asupra bazei de date , adica insert , delete , update , si select.

Metoda findById are un parametru de tip int , producttId , si returneaza productul din baza de date care are id-ul egal cu parametrul functiei . La inceput se face conexiunea cu baza de date , dupa care prin ResulSet se executa comanda din findStatementString , se retin pretul , stocul si numele pentru product , iar la final daca este un product cu id-ul respective acesta este returnat.

Metoda findAll() nu are nici un parametru si are rolul de a return un ArrayList de obiecte , in care se introduce toate datele din tabela Product .

Metoda insert , care are ca parametric un product si un id , insereaza in tabela un nou product cu caracteristicile productului transmis ca parametru , si cu id-ul productului egal cu intregul transmis ca parametru in functie .

Metoda update are ca parametric doi intregi. Aceasta face un update in baza de date , astfel ca , productul cu id-ul egal cu primul intreg transmis ca parametru ii este modificat stocul cu valoarea variabilelor transmise ca parametri functiei .

Metoda delete are ca parametru un intreg , aceasta stergand din baza de date productului care are id-ul egal cu valoarea intregului transmis ca si parametru fucntiei .

## Validator

Interfata Validator are o singura metoda validate(T t) , aceasta fiin implementata in clasa ClientAgeValidator.

## ClientAgeValidator

Aceasta clase are o singura metoda validate(Client t) care verifica daca varsya unui client este intre 5 si 60 , in caz contrar arunca o exceptie.

## ClientBLL

Metoda findClientById are un parametru , un intreg id , si returneaza clientul cu id-ul egal cu cel al parametrului , folosind metoda findById din ClientDAO .

Metoda findAllAclients returneaza un arrayList de obiecte cu toti clientii din tabela Client , folosind metoda findAll() din ClientDAO .

Metoda insertClient , insereaza Clientul transmis ca parametru , doar in cazul in care varsta acestuia este cuprinsa intre 5 si 60 , cu primul id liber .

Metoda updateClient face un update in baza de date folosind metoda update din ClientDAO .

Metoda deleteClient face un delete in baza de date folosind metoda delete din ClientDAO .

## 

## 

## OrderBLL

Metoda findOrderById are un parametru , un intreg id , si returneaza orderul cu id-ul egal cu cel al parametrului , folosind metoda findById din OrderDAO .

Metoda findAllOrders returneaza un arrayList de obiecte cu toti clientii din tabela Order , folosind metoda findAll() din OrderDAO .

Metoda insertOrder , insereaza Orderul transmis ca parametru cu primul id liber .

Metoda updateOrder face un update in baza de date folosind metoda update din OrderDAO .

Metoda deleteOrder face un delete in baza de date folosind metoda delete din OrderDAO .

## ProductBLL

Metoda findProductById are un parametru , un intreg id , si returneaza productul cu id-ul egal cu cel al parametrului , folosind metoda findById din ProductDAO .

Metoda findAllProducts returneaza un arrayList de obiecte cu toti clientii din tabela Product , folosind metoda findAll() din ProductDAO .

Metoda insertProducts , insereaza Productul transmis ca parametru cu primul id liber .

Metoda updateProduct face un update in baza de date folosind metoda update din ProductDAO .

Metoda deleteProduct face un delete in baza de date folosind metoda delete din ProductDAO .

## View

In clasa View este implementata interfata grafica . Aici regasim toate butoanele , labelurile si textFieldurile necesare pentru ca programul sa indeplineasca toate cerintele .

In constructorul acestei clase sunt implementate listenerurile pentru butoane , fiecare buton apeland functiile de care are nevoie pentru a-si indeplini saricna .

Metoda createTable primeste ca parametru un array list de obiecte , prin reflective isi da seama de ce tip este obiectul transmis ca parametru , ii ia toate campurile si le pune intr-un JTable care apare in interfata grafica .

# 4.Cocluzii si posibile dezvoltari ulterioare

Aceasta tema a fost foarte utila pentru a invata sa lucram cu o baza de date .Ea poate fi dezvoltata prin marirea numarului de tabele pentru baza de date , se pot face mai multe operatii cu baza de date . Una din marile imbunatatiri care pot fi aduse acestui proiect este folosirea reflectiei in momentul in care vrem sa afcem operatiile de update , delete , insert si select .