DOCUMENTATIE

TEMA NUMARUL 3

NUME STUDENT: Modrea Elena Alexandra

GRUPA: 30223

Profesor coordonator : Alexandru Rancea

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 3](#_Toc95297887)

[4. Implementare 3](#_Toc95297888)

[5. Rezultate 3](#_Toc95297889)

[6. Concluzii 3](#_Toc95297890)

[7. Bibliografie 3](#_Toc95297891)

# Obiectivul temei

Obiectivul temei este de a implementa si proiecta o aplicatie pentru a gestiona comenzile clientilor .

Ca si obiective secundare avem :

* Analiza problemei si identificarea sarcinilor ;
* Proiectarea aplicatiei de gestiune a comenzilor clientilor ;
* Implementarea aplicatiei de gestiune a comenzilor clientilor ;
* Testarea aplicatiei de gestiune a comenzilor clientilor ;

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Va trebui implementata o aplicatie care sa permita angajatului sa selecteze optiunea de a adauga un nou produs . Aplicatia va afisa un form in care detaliile produsului sa poata fi inserate . Angajatul trebuie sa poata insera numele produsului , pretul sau stocul current .

Angajatul sa poata apasa pe butonul de add .

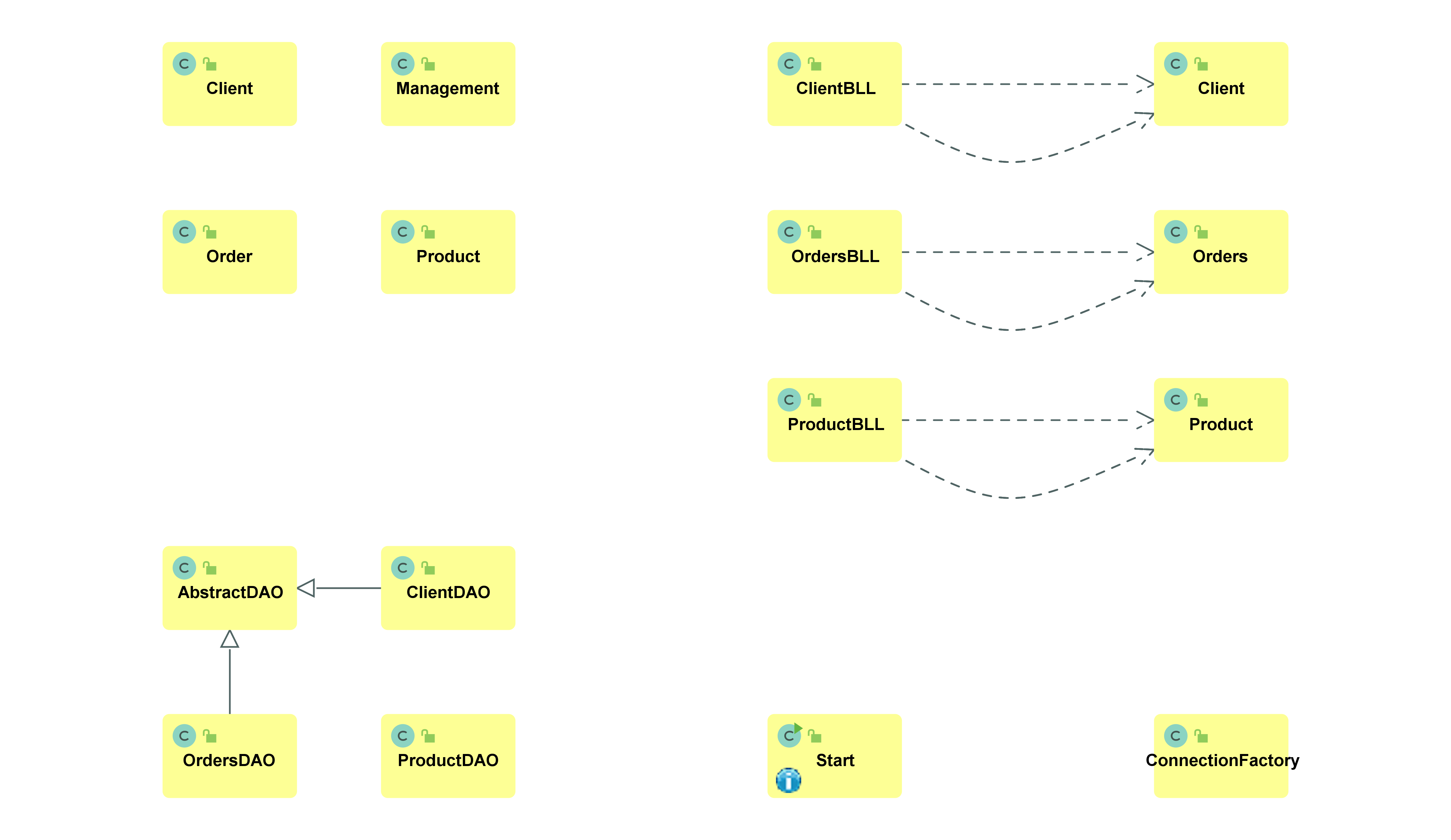
Aplicatia trebuie sa poata salva in baza de date datele despre produs si sa afiseze un mesaj pe ecran

Aplicatia ar trebui de asemenea sa permita unui angajat sa adauge un nou client.

Aplicatia ar trebui sa permita adauarea unui nou produs.

Aplicatia ar trebui sa i ofere posibilittea angajatului sa poata descarca o facture daca acesta doreste .

# Proiectare



Pentru implementare vom avea 6 pachete cu clasele prevazute si in diagrama UML :

Pachetul bll , pachetul connection , pachetul dao , pachetul presentation , pachetul start .

Pachetul bll contine urmatoarele clase : ClientBLL , OrdersBLL , ProductBLL .

Pachetul connection contine o singura clasa : ConnectionFactory

Pachetul dao contine urmatoarele clase : AbstractDAO , ClientDAO , OrdersDAO , ProductDAO .

Pachetul model contine urmatoarele clase : Client , Product , Orders .

Pachetul presentation continue urmatoarele clase : Client , Product , Order .

# Implementare

*.*Clasele implementate in acest proiect sunt : ClientBLL , OrdersBLL , ProductBLL , ConnectionFactory , AbstractDAO , ClientDAO , OrdersDAO , ProductDAO , Client , Product , Orders , Client , Product , Order .

**Clasa Client** din pachetul model :

Aceasta clasa este echivalentul tabelei Client din baza de date . Aceasta clasa continue atributele id , name , email , address ce au aceeasi denumire ca si coloanele din tabela din baza de date .

Aceasta clasa contine urmatoarele metode :

Un constructor gol public Client ( ) ;

Un constructor cu parametrii public Client ( String name , String email , String address ) care initializeaza toate atributele acestei clase pentru a ne fi usor pentru baza de date ;

Un getter pentru id getId() ;

Un setter pentru id setId( int id ) ;

Un getter pentru name getName() ;

Un setter pentru name setName( String name ) ;

Un getter pentru email getEmail( ) ;

Un setter pentru email setEmail ( String email ) .

**Clasa Orders** din pachetul model :

Aceasta clasa este echivalentul tabelei Orders din baza de date . Aceasta clasa contine atributele id , idClient , idProduct , numClient , numeProdus , cantitate ; ce au aceeasi denumire ca si coloanele din tabela din baza de date .

Aceasta clasa contine urmatoarele metode :

Un constructor gol public Orders () ;

Un constructor cu parametrii public Orders ( idClient , idProduct , numClient , numeProdus , cantitate ) care initializeaza toate atributele acestei clase pentru a n fi usor pentru baza de date ;

Un getter pentru id getId ( ) ;

Un setter pentru id setId ( id ) ;

Un getter pentru idClient getIdClient ( ) ;

Un setter pentru idClient setIdClient ( idClient ) ;

Un getter pentru idProduct getIdProduct ( ) ;

Un setter pentru idProduct setIdProduct ( idProduct ) ;

Un getter pentru numClient getNumClient ( ) ;

Un setter pentru numClient setNumClient ( numClient ) ;

Un getter pentru numeProdus getNumeProdus () ;

Un setter pentru numeProdus setNumeProdus ( numeProdus ) ;

Un getter pentru cantitate getCantitate ( ) ;

Un setter pentru cantitate setCantitate ( cantitate ) .

**Clasa Product** din pachetul model :

Aceasta clasa este echivalentul tabelei Product din baza de date . Aceasta clasa continue atributele id , name , cantitate , pret , ce au aceeasi denumire ca si coloanele din tabela din baza de date .

Aceasta clasa continue urmatoarele metode :

Un constructor gol public Product ( ) ;

Un constructor cu parametrii public Product ( name , cantitate , pret ) , care initializeaza toate atributele acestei clase pentru a ne fi usor pentru baza de date ;

Un getter pentru id getId ( ) ;

Un setter pentru id setId ( id ) ;

Un getter pentru name getName ( ) ;

Un setter pentru name setName ( name ) ;

Un getter pentru pret getPret ( ) ;

Un setter pentru pret setPret ( pret ) ;

Un getter pentru cantitate getCantitate ( ) ;

Un setter pentru cantitate setCantitate ( cantitate ) .

**Clasa AbstractDAO** din pachetul dao :

Aceasta clasa este una din cele mai importante clase pentru implementare deoarece toate metodele sunt facute pe un tip de date abstract pentru a le putea utiliza pe orice tip date dorim noi . Astfel , acest lucru ne ofera si un program mai curat si usor de inteles , este si mai usor pentru debug si de asemenea folisim reflexia .

Metodele ce se regasesc in clasa AbstractDAO sunt :

**private String createSelectQuery** (String field ) . Aceasta metoda este una private deoarece o folosim doar in aceasta clasa si deci nu este nevoie sa fie publica . In aceasta metoda se urmareste crearea unui string care sa contina query-ul de tip select din mySql : “ SELECT \* FROM table\_name WHERE field = ? ” . Pentru a a realiza acest lucru m-am folosit de stringBuilder si am concatenate folosind functia append la string pentru a crea query ul droit.

Metoda returneaza stringul creat .

**Private String createSelectAllQuery ( ) .** Aceasta metoda este una private deoarece o folosim doar in aceata clasa si deci nu este nevoie sa fie publica . In aceasta metoda se urmareste crearea unui string care sa contina query-ul de tip select all sin mySql : “ SELECT \* FROM table name ” . Pentru a realiza acest lucru m-am folosit de stringBuilder si am concatenat folosind functia append la string pentru a crea query ul dorit. Metoda returneaza stringul creat .

**Private String createInsertQuery ( T t ) throws IllegalAccesException** . Aceasta metoda este una private deoarece o folisim doar in aceasta clasa si nu este nevoie sa fie publica . In aceasta metoda se urmareste crearea unui string care sa contina query-ul de tip insert din mySql : “ INSERT INTO table\_name ( column1 , column2 , column3 , .. ) VALUES ( value1 , value2 , value3 , … ) ” . Pentru a realiza acest lucru am folosit un string si am create prima oara insert into table\_name ; am aflat table\_name-ul cu ajutorul functiei type.getSimpleName().

Am luat un vector de fields pentru putea retine coloanele . Apoi am adaugat denumirile coloanelor acest lucru fiind posibil cu ajutorul metodei fields[i].getName ( ) si le-am concatenat la stringul initial. Apoi trebuiau luate si valorile din tabela acest lucru fiind posibil cu ajutorul functiei fields.get(t) urmand sa concatenam apoi la stringul initial . Metoda returneaza stringul creat .

**Private String createDeleteQuery( int id)** . Aceasta metoda este una private deoarece o folisim doar in aceasta clasa si nu este nevoie sa fie publica . In aceasta metoda se urmareste crearea unui string care sa contina query-ul de tip insert din mySql : “ DELETE FROM table\_name WHERE id = id ” . Pentru a realiza acest lucru m-am folosit de string si am concatenat la string pentru a crea query ul dorit. Metoda returneaza stringul creat .

Private String createUpdateQuery( T t , int id ) throws IllegalAccesException . . Aceasta metoda este una private deoarece o folosim doar in aceasta clasa si nu este nevoie sa fie publica . In aceasta metoda se urmareste crearea unui string care sa contina query-ul de tip insert din mySql : “ UPDATE table\_name SET column1 = ‘value1’ , column2 = ‘value2’ , column3=’value2’ WHERE id = id” . Pentru a realiza acest lucru m-am folosit de string si am concatenat la string pentru a crea query ul dorit astfel : am aflat numele tabelei cu ajutorul functiei type.getSimpleName ( ) , apoi am salvat intr un vector de fields-uri coloanele urmand ca apoi sa parcurgem acest vector pentru a concatena numele coloanelor dar si elementele din coloana . Metoda returneaza stringul creat .

**Public List <T> findAll () .** Aceasta metoda urmareste pastrarea tututor elementelor dintr o tabela astfel : apelam query ul create la cu ajutorul functiei createSelectAllQuery pentru a putea gasii toate atributele in baza de date , dupa care apelam functia createObjects care ne va creea o lista cu toate obiectele de tip care s au gasit in urma interogarii query ului . Functia returneaza lista returnata de metoda createObjects .

**Pubic T findById ( int id )** . Aceasta metoda urmareste pastrarea atributelor unui obiect dintr-o tabela astfel : apelam query ul createSelectQuery pentru a putea gasii toate atributele obiectului cu id-ul id din baza de date , dupa care apelam functia createObjects care ne va creea o lista cu toaye obiectele gasite insa tinand cont ca ii dam cautare dupa id, vom avea nevoie doar de primul element al listei. Metoda returneaza primul element al listei returnate de metoda creatObjects .

**Private List<T > create Objects ( Resultset resultset )** . Aceasta metoda este una private deoarece o folisim doar in aceasta clasa si nu este nevoie sa fie publica . Aceasta metoda urmareste extragerea atributelor dintr o tabela cu ajutorul resultset si creearea unui obiect de tipul T cu acele atribute . Metoda returneaza o lista de obiecte de tipul T .

**Public void insert ( T t ) throws IllegalAccessException .** Aceasta metoda urmareste realizarea query ului insert din mySql atfel : avem nevoie de conxiunea cu baza de date pentru a putea insera in baza de date anumite linii . Dupa care vom apela prepareStatement ( query ) care sa ne pregateasca query ul pentru baza de date , iar la final executam query ul cu metoda executeUpdate () .

**Public void update ( T t , int id ) throws IllegalAccesException** . Aceasta metoda urmareste realizarea query ului update din mySql atfel : avem nevoie de conxiunea cu baza de date pentru a putea actualiza din baza de date anumite linii . Dupa care vom apela prepareStatement ( query ) care sa ne pregateasca query ul pentru baza de date , iar la final executam query ul cu metoda executeUpdate () .

**Public void delete ( int id )** . Aceasta metoda urmareste realizarea query ului delete din mySql atfel : avem nevoie de conxiunea cu baza de date pentru a putea sterge din baza de date anumite linii . Dupa care vom apela prepareStatement ( query ) care sa ne pregateasca query ul pentru baza de date , iar la final executam query ul cu metoda executeUpdate () .

**Public DefaultTableModel tableModel ( List <T> list )** **throws IllegalAccessException** . Aceasta metoda urmareste crearea unui tabel pentru a JTable astfef ; vom crea in primul rand headerul tabelului cu ajutorul metodei type.getDeclaredFields si fields.getName() pe care le vom adauga cu ajutorul metodei addColomn . Apoi vom crea liniile cu ajutorul metodelor fields[i]. get(t) pe care le vom adauga in tabel cu ajutorul metodei addRow . Metoda returneaza tabelul creat .

Clasele ClientDAO si OrdersDAO nu contin nicio metoda diferita, ele sunt folosite doar pentru a extinde clasa AbstractDAO pentru a putea instanta obiectul .

Clasa ProductDAO extinde si ea clasa AbstractDAO insa mai contine si 2 metode: verificareCantitate care returneaza cantitatea finala si actualizeazaCantit care actualizeaza in baza de date cantitatea finala.

Clasele din pachetul BLL sunt echivalente . Acestea instanteaza functiile din AbstractDAO cu cu obiectele de tip Client, Product si respectiv Orders.

Pachetul start continue doar o clasa main in care apelam interfata.

Pachetul presentation contine clasele pentru interfata dar si formul.

# Rezultate

In urma acestui proiect am reusit sa implementez interfata grafica care sa ii permita angajatului operatiile CRUD pe client si product dar si sa poata crea o comanda.

# Concluzii

In urma acestui proiect am invatat sa folosesc reflexia care este de ajutor in lucrul cu baza de date, mi-am reamintit lucrul cu baza de date legata de java si am reusit realizarea acestui proiect.

# Bibliografie

<https://dev.to/scottshipp/two-ways-to-integrate-javadoc-into-intellij-idea-leb>

<https://www.javatpoint.com/example-to-connect-to-the-mysql-database>

https://www.oracle.com/technical-resources/articles/java/javareflection.html