Logo, company name

Description automatically generated

**Documentatie tema 3**

**Nume student: Maracine Stefania Maria**

**Profesor coordonator: Rancea Alexandru**

**Grupa: 30223**

# CUPRINS

[1. Obiectivul temei **.**](#_Toc95297885)

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

[3. Proiectare **.**](#_Toc95297887)

[4. Implementare **.**](#_Toc95297888)

[5. Rezultate **.**](#_Toc95297889)

[6. Concluzii **.**](#_Toc95297890)

[7. Bibliografie **.**](#_Toc95297891)

1. **Obiectivul temei**

Obectivul principal al celei de a treia teme de la laboratorul de tehnici de programare este de a implementa si proiecta o aplicație care urmărește gestionarea produselor, clientilor si comenzilor dintr-un depozit. In aceasta aplicatie se vor putea crea noi clienti, noi produse si noi comezi, se vor afisa informatii despre clienti produse si comenzi.

Sub-obiective:

1. Prezentați clar obiectivul principal și subobiectivele necesare pentru a-l atinge.

2. Analizați problema și definiți cerințele funcționale și nefuncționale.

3. Proiectați soluția

4. Implementați soluția

5. Testați soluția

1. **Analiza problemei, modelare, scenario, cazuri de utilizare**

Diagram

Description automatically generatedProblema este una simpla, utilizatorul aplicatiei, adica un angajat, va deschide aplicatia, pagina de pornire a interfetei grafice, si va selecta una dntre cele 3 optiuni pe care le are, client angajat sau comenza. Dupa selectare, se va deschide o noua interfata, corespunzatoare selectiei facute anterior. Aici se vor putea adauga, modifica sau sterge obiecte, in functie de alegerea facuta. Toate aceste lucruri vor fii explicate mai tarziu. Toate modificarile facute prin intermediul interfetei grafice sunt de fatada, schimbarile adevarate se fac prin metode din java care schimba informatiile din tabelele create in baza noastra de date pe care am conectat-o cu proiectul. In acest proiect se va lucra cu MyQSL Workbench.

1. **Poiectare**

In acest capitol este prezentata diagrama UML, clasele existente, pachetele folosite si structurile de date folosite.

Pachetele folosite sunt:

-pachetul BLL care continue clasele de validare: EmailValidator si StudentAgeValidator, exemple date in prezentarea proiectului, ClientEmailValidator si ProductStocValidator. In acest pachet mai avem si clasele ClientBLL, ProductBLL si OrderBLL care fac conexiunea intre interfata grafica si clasele cu metode.

-pachetul DAO care continue clasele ClientDAO, ProductDAO, OrderDAO si AbstractDAO. Aceste clase implementeaza metode asupra bazei de date pentru a modifica tabelele si informatiile corespunzatoare.

-pachetul model continue clasele Client, Product si Order. Aceste clase contin variabile corespunzatoare cu cele din baza de date si contructori gettere si settere pentru a lucre cu ele in metodele din java.

-pachetul presentation continue clasele ClientView, ProductView, OrderView si StartUpPage. Aceste clase implementeaza metodele pentru interfata grafica.

-pachetul Start care continue doar clasa Start unde au fost facute cateva teste inaintea implementarii interfetei si de unde pornim acum interfata grafica.

-pachetul connection care continue doar clasa ConnectionFactory unde se realizeaza conexiunea cu baza de date.

Ca si structuri de date am folosit listele.

1. **Implementare**

In acest capitol se va prezenta mai in detaliu functionalitatea fiecare clase, cu variabilele si metodele lor.

1. Clasa Client

In aceasta clasa avem variabilele necesare, care coincid cu coloanele din tabelul din baza de date. De asemenea aveam gettere si settere pentru a putea accesa informatiile si in celalate clase. De asemenea avem si metoda toString care ne ajuta cu afisarea.

1. Clasa Product

In aceasta clasa avem variabilele necesare, care coincid cu coloanele din tabelul din baza de date. De asemenea aveam gettere si settere pentru a putea accesa informatiile si in celalate clase. De asemenea avem si metoda toString care ne ajuta cu afisarea.

1. Clasa Order

In aceasta clasa avem variabilele necesare, care coincid cu coloanele din tabelul din baza de date. De asemenea aveam gettere si settere pentru a putea accesa informatiile si in celalate clase. De asemenea avem si metoda toString care ne ajuta cu afisarea.

1. AbstractDAO

In aceasta clasa sunt implementate mai multe metode legate de obiectele din tabelele noastre din baza de date. Aceste metode functioneaza pentru obiecte te tip general, nu doar un singur tip de obiect precum Client sau Product.

Primele doua metode, createSelectQuery si createSelectAll, alcatuiesc un string pe care il vom utiliza in urmatoarele metode ca si query pentru a face interogati in baza de date.

Metoda showInformation este o metoda care ia o lista de obiecte si prin functia getDeclaredFields va scoate atributele obiectelor cum ar fii nume id stoc etc si de asemenea ne va adauga in tabele pe coloane si linii informatiile valorile atributelor respective. Pentru implementarea ei m-am folosit de Reflection Techniques Ne luam un vector de string-uri si un contor. Intru-un for, pentru fiecare element, in vectorul declarant mai sus, vom concatena numele getterelor necesare, dupa tipul de atribut pe care il returneaza getDeclaredFields. In functie de cate field-uri vom avea se vor adauga si coloanele. La final ne vom lua un vector de tip Object in care vom pune valoarea atributelor cu ajutorul getterelor pe care le-am salvat in vectorul de string-uri, dupa care le adaugam pe linii in tabela noastra.

Metoda findAll este o metoda care returneaza o lista de obiecte. Metoda cauta in baza noastra de date toate obiectele dintr-o tabela, le pune intr-o lista si le afiseaza. In aceasta metoda ne folosim si de metoda createSelectAll.

Metoda findById primeste ca parametru un int id, si returneaza un obiect al carui id corespunde cu numarul primit ca parametru. In aceasta metoda ne folosim de metoda createSelectQuery.

Metoda createObject este o metoda care creaza obiecte de tip T pe care le pune intr-o lista pe care apoi o returneaza.

1. Clasa ClientDAO

Aceasta clasa extinde clasa AbstractDAO deci ii mosteneste metodele.

Metoda insert adauga un nou client in baza noastra de date. Are ca parametru un obiect de tip client. Ne declaram un string care va fii query-ul nostru de interogare in baza de date. String-ul contine cuvinte cheie din limbajul mySQL dar ne folosim si de gettere pentru a lua informatiile pe care vrem sa le adaugam in baza de date din obiectul primit ca parametru.

Metoda update actualizeaza informatiile despre un client din baza noastra de date. Merge pe aceeasi metoda ca insert. Ne declaram un string care va fii query-ul nostru de interogare in baza de date. String-ul contine cuvinte cheie din limbajul mySQL dar ne folosim si de gettere pentru a lua informatiile pe care vrem sa le modificam in baza de date din obiectul primit ca parametru.

Metoda delete sterge din baza noastra de date un client al carui id corespunde cu numarul primit ca parametru cu ajutorul unui query pe care ni-l declaram in metoda si in care concatenam sintaxa din mySQL si id-ul primit ca parametru.

1. Clasa ProductDAO

Aceasta clasa extinde clasa AbstractDAO deci ii mosteneste metodele.

Metoda insert adauga un nou produs in baza noastra de date. Are ca parametru un obiect de tip client. Ne declaram un string care va fii query-ul nostru de interogare in baza de date. String-ul contine cuvinte cheie din limbajul mySQL dar ne folosim si de gettere pentru a lua informatiile pe care vrem sa le adaugam in baza de date din obiectul primit ca parametru.

Metoda update actualizeaza informatiile despre un produs din baza noastra de date. Merge pe aceeasi metoda ca insert. Ne declaram un string care va fii query-ul nostru de interogare in baza de date. String-ul contine cuvinte cheie din limbajul mySQL dar ne folosim si de gettere pentru a lua informatiile pe care vrem sa le modificam in baza de date din obiectul primit ca parametru.

Metoda delete sterge din baza noastra de date un produs al carui id corespunde cu numarul primit ca parametru cu ajutorul unui query pe care ni-l declaram in metoda si in care concatenam sintaxa din mySQL si id-ul primit ca parametru.

Metoda decrementStock primeste ca parametru un obiect de tip produs si ii decrementeaza stocul cu o unitate, apeland apoi metoda update pentru a actualiza informatia si in baza noastra de date.

1. Clasa OrderDAO

Aceasta clasa extinde clasa AbstractDAO deci ii mosteneste metodele.

Metoda insert adauga o noua comanda in baza noastra de date. Are ca parametru un obiect de tip client. Ne declaram un string care va fii query-ul nostru de interogare in baza de date. String-ul contine cuvinte cheie din limbajul mySQL dar ne folosim si de gettere pentru a lua informatiile pe care vrem sa le adaugam in baza de date din obiectul primit ca parametru.

Metoda update actualizeaza informatiile despre o comanda din baza noastra de date. Merge pe aceeasi metoda ca insert. Ne declaram un string care va fii query-ul nostru de interogare in baza de date. String-ul contine cuvinte cheie din limbajul mySQL dar ne folosim si de gettere pentru a lua informatiile pe care vrem sa le modificam in baza de date din obiectul primit ca parametru.

Metoda delete sterge din baza noastra de date o comanda al carei id corespunde cu numarul primit ca parametru cu ajutorul unui query pe care ni-l declaram in metoda si in care concatenam sintaxa din mySQL si id-ul primit ca parametru.

1. Clasa ClientBLL

Aceasta clasa conecteaza clasa ClientDAO de interfata grafica. In aceasta clasa, toate metodele sunt metode de legatura care ajuta la apelarea metodelor din clasa ClientDAO sau arunca o exceptie daca metoda nu duce la capat ceea ce trebuie sa faca.

Metoda insert primeste ca parametru un client si apeleaza in interiorul ei metoda de insert din clasa ClientDAO. Daca clientul nu a putut fii adaugat arunca o exceptie de tipul NoSuchElementFound.

Metoda update primeste ca parametru un client si apeleaza in interiorul ei metoda de update din clasa ClientDAO. Daca clientul nu a putut fii actualizat arunca o exceptie de tipul NoSuchElementFound.

Metoda delete primeste ca parametru un client si apeleaza in interiorul ei metoda de delete din clasa ClientDAO. Metoda delete din clasa ClientDAO primeste ca parametru id-ul clientului primit ca parametru aici. Daca clientul nu a putut fii sters arunca o exceptie de tipul NoSuchElementFound.

1. Clasa ProductBLL

Aceasta clasa conecteaza clasa ProductDAO de interfata grafica. In aceasta clasa, toate metodele sunt metode de legatura care ajuta la apelarea metodelor din clasa ProductDAO sau arunca o exceptie daca metoda nu duce la capat ceea ce trebuie sa faca.

Metoda insert primeste ca parametru un produs si apeleaza in interiorul ei metoda de insert din clasa ProductDAO. Daca produsul nu a putut fii adaugat arunca o exceptie de tipul NoSuchElementFound.

Metoda update primeste ca parametru un produs si apeleaza in interiorul ei metoda de update din clasa ProductDAO. Daca produsul nu a putut fii actualizat arunca o exceptie de tipul NoSuchElementFound.

Metoda delete primeste ca parametru un produs si apeleaza in interiorul ei metoda de delete din clasa ProductDAO. Metoda delete din clasa ProductDAO primeste ca parametru id-ul produsului primit ca parametru aici. Daca produsul nu a putut fii sters arunca o exceptie de tipul NoSuchElementFound.

1. Clasa OrderBLL

Aceasta clasa conecteaza clasa OrderDAO de interfata grafica. In aceasta clasa, toate metodele sunt metode de legatura care ajuta la apelarea metodelor din clasa OrderDAO sau arunca o exceptie daca metoda nu duce la capat ceea ce trebuie sa faca.

Metoda insert primeste ca parametru o comanda si apeleaza in interiorul ei metoda de insert din clasa OrderDAO. Daca comanda nu a putut fii adaugata arunca o exceptie de tipul NoSuchElementFound.

Metoda update primeste ca parametru o comanda si apeleaza in interiorul ei metoda de update din clasa OrderDAO. Daca comanda nu a putut fii actualizata arunca o exceptie de tipul NoSuchElementFound.

Metoda delete primeste ca parametru o comanda si apeleaza in interiorul ei metoda de delete din clasa OrderDAO. Metoda delete din clasa OrderDAO primeste ca parametru id-ul comenzii primit ca parametru aici. Daca comanda nu a putut fii stearsa arunca o exceptie de tipul NoSuchElementFound.

1. Clasele pentru interfata grafica

Pentru interfata grafica avem 4 clase, StartUpPage, ClientView, ProductView si OrderView. In fiecare clasa avem implementate in constructori butoanele. Fiecarui buton i se atribuie un ActionListener si apoi se implementeaza c ear trebui sa faca fiecare buton in parte. Se apeleaza fiecare metoda necesara. De exemplu in clasa ClientView pentru butonul addButton care are rolul de a adouga un client nou se ia un obiect de tip Client si unul de tip ClientDAO. Din textfiled-urile din interfata grafica se ia informatiile necesare pentru adaugarea clientului in baza de date, se face conversia de la string la int daca este nevoie si se apeleaza metoda de insert client.

Interfata grafica a fost construita cu GUI Form din IntelliJ.

1. **Rezultate**

Nu s-a folosit nicio metoda speciala de testare.

1. **Concluzii**

In concluzie, aceasta aplicatie de gestionare a diferitelor tipuri de obiecte usureaza foarte mult munca unui deposit care ofera spre vanzare produse unor clienti.

1. **Bibliografie**

<https://attacomsian.com/blog/capitalize-first-letter-of-string-java>

<https://gitlab.com/utcn_dsrl/pt-reflection-example>