**DOCUMENTATIE TEMA 3**

**ORDER MANAGEMENT**

**Rad Daniel-Cristian**

**Grupa 30223**

Contents

[1.Obiective 3](#_Toc166763653)

[1.1 Obiectiv principal 3](#_Toc166763654)

[1.2 Obiective secundare 3](#_Toc166763655)

[2. Analiza problemei 4](#_Toc166763656)

[3. Proiectare 4](#_Toc166763657)

[3.1. Diagrama UML 4](#_Toc166763658)

[3.2. Diagrama de use case 5](#_Toc166763659)

[3.3. Diagrama de pachete 5](#_Toc166763660)

[4 .Rezultate 7](#_Toc166763661)

[5. Concluzii 9](#_Toc166763662)

[6. Bibliografie 9](#_Toc166763663)

## 1.Obiective

### 1.1 Obiectiv principal

Obiectivul acestei teme este implementarea unei aplicatii OrderManagement care proceseaza comenzile clientilor pentru un depozit. Se folosesc baze de date relationale pentru a stoca produsele, clientii si comenzile. De asemenea aplicatia trebuie sa fie structurata pe pachete folosind o arhitectura stratificata. Si ar trebui sa aiba urmatoarele clase:

* Clasele model – modelele de date ale aplicatiei ;
* Clasele pentru business logic– implementeaza logica aplicatiei ;
* Clasele de prezentare – implementeaza input-ul / output-ul user-ului ;

Clasele pentru acces de date – implementeaza accesul la baza de date ;

### 1.2 Obiective secundare

Obiective secundare*:*

* Dezvoltarea de use-case-uri
* Alegerea corecta a structurilor de date
* Impartirea pe clase
* Dezvoltarea algoritmilor
* Implementarea solutiilor
* Testarea programului

## 2. Analiza problemei

Aplicatia trebuie sa permita procesarea comenzilor, sa efectueze operatiile cerute, sa salveze modificarile efectuate in baza de date si sa genereze rapoarte. Alte clase si pachete pot fi adaugate pentru a implementa functionalitatea completa a aplicatiei.

Operatiile pe care aplicatia le poate procesa:

* Adaugarea unui client in baza de date ;
* Stergerea unui client din baza de date ;
* Adaugarea unui produs in baza de date ;
* Stergerea unui produs din baza de date ;
* Crearea unei comenzi pentru client ;

## 3. Proiectare

In proiectarea acestei aplicatii am folosit principiile OOP si tehnica Reflection. Am creeat mai multe clase care comunica intre ele pentru rezolvarea sarcinilor. In continuare se vor prezenta structurile de date, interfetele, clasele si algoritmii utilizati in proiectarea aplicatiei.

### 3.1. Diagrama UML

A computer screen shot of a computer

Description automatically generated

### 

### 3.2. Diagrama de use case

A diagram of a product order

Description automatically generated

### A picture containing clock, bird Description automatically generated3.3. Diagrama de pachete

In continuare se vor prezenta clasele si pachetele pe care le-am folosit in acest proiect, dar si metodele cele mai importante din fiecare clasa.

ConnectionFactory

ConnectionFactory este o clasă care stabilește legătura cu baza de date, adoptând pattern-ul Singleton. Pattern-ul Singleton este o tehnică de design software ce permite crearea unui singur obiect al unei clase, fiind util când se impune coordonarea acțiunilor în cadrul sistemului printr-un singur obiect. Clasa ConnectionFactory este încadrată în pachetul DataAccess, furnizând conexiunea cu baza de date. Clasa conține driver-ul MySQL pentru baza de date, userul, parola și locația bazei de date. Un atribut major al clasei este singleInstance, care, împreună cu constructorul, asigură accesul la propriul obiect al clasei fără a necesita instantierea.

Model

Pachetul Model contine clasele care reprezintă tabelele din baza de date, mapează toate atributele pe coloanele respectivelor tabele.

Clasa Client include următoarele atribute: id-ul clientului (folosit ca si cheie primară în tabelul Client), numele clientului, si adresa de e-mail.

Clasa Order constă din următoarele atribute: id-ul fiecărei comenzi, id-ul clientului care a plasat comanda, id-ul produsului dorit și cantitatea dorită.

Clasa Product conține următoarele atribute: id-ul produsului (folosit ca si cheie primară în tabelul Products), numele produsului si cantitatea disponibilă în stoc.

BusinessLogic

Pachetul BusinessLogic conține logica aplicației, adica clasele care implementează toate metodele și operațiile pe baza de date. Clasa DAO utilizează tehnica reflection și are ca atribute un logger și un tip generic. În constructor, se preia clasa tipului generic T cu care se creează obiectul de tip DAO.

Pentru fiecare operație necesară asupra bazei de date, clasa conține o metodă privată pentru a returna sub forma de string query-ul SQL corespunzător. Se folosește un obiect de tip StringBuilder și se apelează metoda append() pentru a concatena string-ului datele necesare. De exemplu, metoda createSelectQuery() returnează un obiect de tip string de forma: "select \* from" + "where". Numele tabelului este găsit prin intermediul parametrului type și a apelării metodei getSimpleName().

## 4 .Rezultate

În această secțiune, vom prezenta scenariile de testare și rezultatele obținute în urma rulării aplicației.

Se va rula programul si se va deschide fereastra principala.

A screenshot of a computer

Description automatically generated

Se va merge la fereastra clienti, de unde putem vedea clientii existenti, putem adauga clienti noi, putem edita clientii existenti sau ii putem sterge.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated Dupa aceea vom merge la fereastra de produse. Acolo putem efectua exact aceleasi operatii ca pentru clienti.

Vom merge si la fereastra de comenzi. De acolo putem vedea comenzile existente si putem creea comenzi noi selectand un client si un produs existent.

A screenshot of a computer

Description automatically generated

## 

## 5. Concluzii

In concluzie, aceasta tema m-a ajutat foarte mult sa inteleg legatura programelor Java cu bazele de date. Pe parcursul realizarii temei am invatat cum sa folosesc tehnicile de Reflection. Consider aceasta tema ca fiind una foarte utila pentru dezvoltarea cunostintelor de programare in Java.

## 6. Bibliografie

<https://www.baeldung.com/java-jdbc>

<http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/>

<https://dzone.com/articles/layers-standard-enterprise>

<http://tutorials.jenkov.com/java-reflection/index.html>

<https://www.baeldung.com/javadoc>