**DOCUMENTATIE TEMA 3**

**ADMINISTRAREA COMENZILOR**

**UNUI DEPOZIT**

**Andrei Sasaran**

**Grupa 30221**

**Profesor Laborator: Dorin Moldovan**

Cuprins:

1. Cerinte functionale …………………………………………………………………………....................... 3

2. Obiective ……………………………………………………………………………………...................... 3

2.1 Obiectiv principal ………………………………………………………………………...……….….. 3

2.2 Obiective secundare …………………………………………………………………………….….… 3

3. Analiza problemei …………………………………………………………………………........................ 4

Use case-uri / scenarii ………………………………………………………………………………... 4

4.Proiectare ……………………………………………………………………………………………......… 4

4.1 Structuri de date........................................................................................................................................4

4.2 Diagrama de clase..................................................................................................................................5

4.3 Algoritmi ................................................................................................................................................6

5. Implementare.................................................................................................................................................7

Testare..........................................................................................................................................................8

6. Concluzii si Dezvoltari Ulterioare...............................................................................................................51

7. Bibliografie..................................................................................................................................................52

# Cerinte Functionale

( cerintele problemei )

Conceperea unui sistem de gestione a clienților și produselor unui depozit.

O aplicație conectată la o bază de date, structurată pe o arhitectură cu clase de tip: model, bussiness-logic, presentation și data-acces.

Aceasta aplicație primește comenzi prin intermediul unui fișier de tip text și execută diferite tipuri de operații asupra bazei de date. Aceste informații sunt legate de clienții si produsele depozitului respectiv.

De asemenea, necesită implementarea unei metode care validează tipurile de comenzi accesibile.

Comenzile valide sunt:

* Insert client
* Delete client
* Report client
* Insert product
* Delete product
* Report product
* Order
* Report order

# Obiective

## Obiectiv Principal:

(obiectivul principal al proiectului)

Obiectivul principal al proiectului este de a creea multiple clase care să modeleze cât mai bine realitatea.

Aceste clase trebuie ulterior să relaționeze cu tabelele lor corespondente din baza de date. Astfel, definim clasele:

Client, Product și Order.

## Obiective secundare:

(pasii care conduc catre atingerea obiectivului principal )

|  |  |  |
| --- | --- | --- |
| **Obiectiv Secundar** | **Descriere** | **Capitol** |
| Alegerea structurilor de date | Structurile de date folosite pentru a modela problema. | 4 |
| Impartirea pe clase | Modelarea logic a realitatii. | 4 |
| Dezvoltarea algoritmilor | Descrierea structurilor de date necesare pentru indeplinirea obiectivului principal si algoritmi care sa permita realizarea acestuia. | 4 |
| Implementarea solutiei | Va fi descrisa fiecare clasa cu campurile si metodele specifice. | 5 |

# Analiza Problemei

# Use-case-uri / Scenarii

Utilizator introduce prin intermediul fișierului text comenzile menționate mai sus. Aplicația le execută, generând pdf-uri cu informații legate de informațiile din baza de date. Pentru ca sistemul sa opereze în mod corect, trebuie să asigurăm validitatea sintactică a comenzilor care reprezintă datele de intrare.

( etapele de proiectare )

## Structuri de date

(structurile de date folosite in program)

Pentru crearea clasei Client s-au folosit urmatoarele campuri definitorii :

private int idClient**;**private String name**;**private String address**;**

idClient – id-ul unic al fiecarui client, cheia primară a bazei de date

name – numele clientului

address - adresa clientului

Pentru crearea clasei Product s-au folosit urmatoarele campuri definitorii :

private int idProduct**;**private String name**;**private double quantity**;**private double price**;**

idProduct – id-ul unic al fiecarui produs, cheia primară a bazei de date

name – numele produsului

quantity - cantitatea disponibilă în depozit a produsului

price - prețul produsului

Pentru crearea clasei Order s-au folosit urmatoarele campuri definitorii :

private int idOrder**;**private int idClient**;**private int idProduct**;**private double quantity**;**

idOrder – id-ul unic al fiecare comenzi, cheia primară a bazei de date (a acestui tabel)

idClient – id-ul unic al fiecarui client, cheia primară a bazei de date (a tabelului Client)

idProduct – id-ul unic al fiecarui produs, cheia primară a bazei de date (a tabelului Product)

quantity - cantitatea solicitată de client

și parametrul auxiliar:

private double totalPrice = -**1.0;**

(cu valoarea inițiala -1) – prețul comenzii.

Acesta din urmă este calculat dupa verificarea stocului și reprezizntă suma pe care clientul trebuie să o achite.

Celalte tipuri de clase sunt folosite pentru conexiunea cu baza de date și validarea câmpurilor.

Clasa care se ocupa de conexiunea cu baza de date:

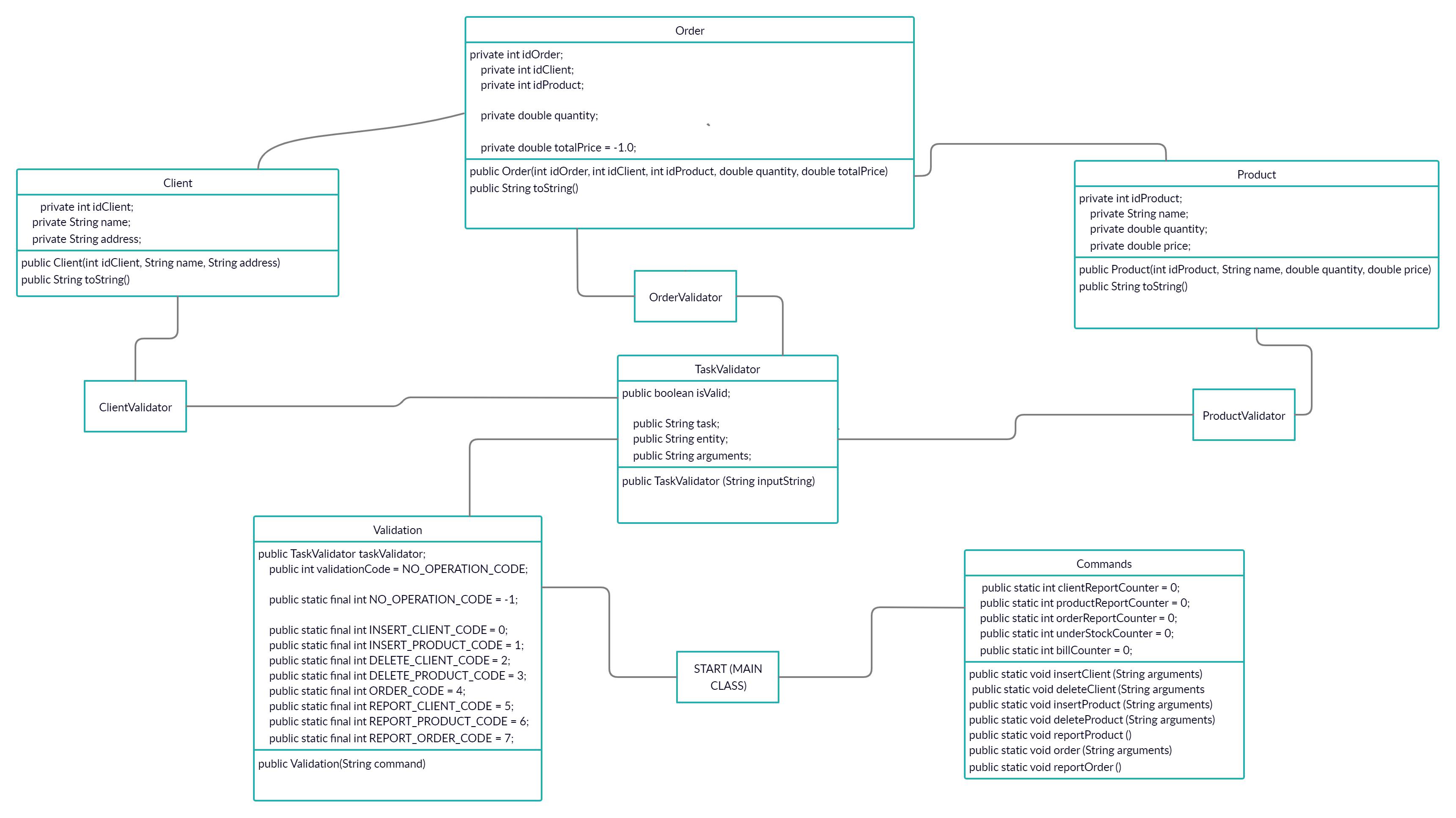
private static final Logger *LOGGER* = Logger.*getLogger*(MyConnection.class.getName())**;**private static final String *DRIVER* = "com.mysql.cj.jdbc.Driver"**;**private static final String *DBURL* = "jdbc:mysql://localhost/ordermanagement"**;**private static final String *USER* = "root"**;**private static final String *PASS* = "123456"**;**private static Connection *connection***;**

O clasă de tip Validator conține aceleași câmpuri ca și clasa pe care o validează cărora li se mai adaugă variabila boolean isValid. Clasa ClientValidator:

public boolean isValid**;**public String name**;**public String address**;**public ClientValidator(String inputString) {  
 int index = inputString.indexOf(",")**;** if (index == -**1**){  
 isValid = false**;** }  
 else {  
 isValid = true**;** name = inputString.substring(**0,** index)**;** address = inputString.substring(index + **2**)**;** }  
}

## Diagrama de clase

(schema UML a proiectului )



Algoritmi

Singurii algortmi folosiți in acest proiect sunt cei care inserează, sterg, updateaza sau returneaza informatii cu privire la baza de date. Acestia apar ori in Java sub forma de if-uri si bucle, ori in MySQL sub forma de proceduri stocate.

Exemplu:

CREATE DEFINER=`root`@`localhost` PROCEDURE `insertClient`(p\_name varchar(45), p\_address varchar(45))

BEGIN

insert into client (name, address)

values (p\_name, p\_address);

END

# Implementare

Clasele au fost prezentate mai sus. Iata cateva metode importane specifice acestora:

Ex: inserarea unui client

public static void insertClient (String arguments){  
  
 ClientValidator clientValidator = new ClientValidator(arguments)**;** if (clientValidator.isValid == true){  
 MyConnection myConnection = new MyConnection()**;** Connection connection = MyConnection.*getConnection*()**;** try {  
 String insertClientString = "{call insertClient(?,?)}"**;** CallableStatement callableStatement = connection.prepareCall(insertClientString)**;** callableStatement.setString(**1,** clientValidator.name)**;** callableStatement.setString(**2,** clientValidator.address)**;** callableStatement.execute()**;** myConnection.*close*(callableStatement)**;** myConnection.*close*(connection)**;** } catch (SQLException e) {  
 e.printStackTrace()**;** }  
 }  
}

Ex: crearea unui raport legat de produse

public static void reportProduct (){  
  
 try {  
 MyConnection myConnection = new MyConnection()**;** Connection connection = MyConnection.*getConnection*()**;** String reportProduct = "select \* from product"**;** PreparedStatement preparedStatement = connection.prepareStatement(reportProduct)**;** ResultSet resultSet = preparedStatement.executeQuery()**;** *productReportCounter*++**;** String documentName = "reportProduct" + "No" + *productReportCounter* + ".pdf"**;** Document document = new Document()**;** PdfWriter writer = PdfWriter.*getInstance*(document**,** new FileOutputStream(documentName))**;** document.open()**;** document.add(new Paragraph("idProduct name quantity price"))**;** while (resultSet.next()){  
 Product product = new Product(resultSet.getInt("idProduct")**,** resultSet.getString("name")**,** resultSet.getDouble("quantity")**,** resultSet.getDouble("price"))**;** Paragraph paragraph = new Paragraph(product.toString())**;** document.add(paragraph)**;** document.add(new Paragraph(" "))**;** }  
 document.close()**;** //writer.close();  
  
  
 myConnection.*close*(preparedStatement)**;** myConnection.*close*(resultSet)**;** myConnection.*close*(connection)**;** } catch (SQLException e) {  
 e.printStackTrace()**;** } catch (FileNotFoundException e) {  
 e.printStackTrace()**;** } catch (DocumentException e) {  
 e.printStackTrace()**;** }  
  
  
}

**Testare**

commands.txt

Insert client: Ion Popescu, Bucuresti  
Insert client: Luca George, Bucuresti  
Report client  
Insert client: Sandu Vasile, Cluj-Napoca  
Report client  
Delete client: Ion Popescu, Bucuresti  
Report client  
Insert product: apple, 20, 1  
Insert product: peach, 50, 2  
Insert product: apple, 20, 1  
Report product  
Delete product: peach  
Insert product: orange, 40, 1.5  
Insert product: lemon, 70, 2  
Report product  
Order: Luca George, apple, 5  
Order: Luca George, lemon, 5  
Order: Sandu Vasile, apple, 100  
Report client  
Report order  
Report product

# Concluzii si Dezvoltari Ulterioare

In urma realizarii acestui proiect am invatat ca totul porneste de la idei. In loc sa stau sa analizez toate scenariile posibile, am trecut direct la implementare, revenind in nenumarate randuri si facand modificari asupra codului. Toate aceste impedimente puteau fii evitate cu putin mai multa rabdare.

Acest proiect poate fii dezvoltat prin introducerea unor noi functionalitati cu privire la statistica depozitulu: inregistrarea numarului de comenzi pe zi, etc.

# Bibliografie

<https://stackoverflow.com/>

<https://www.youtube.com/>

Cursurile de POO / TP

http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/ <http://theopentutorials.com/tutorials/java/jdbc/jdbc-mysql-create-database-example>

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

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

<https://www.baeldung.com/java-pdf-creation>

<https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html>

https://dev.mysql.com/doc/refman/5.7/en/using-mysqldump.html