



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

DOCUMENTAȚIE

TEMA 3

Management Magazin Online

NUME STUDENT: Boloș Andrei Nicolae
GRUPA: 30227

CUPRINS

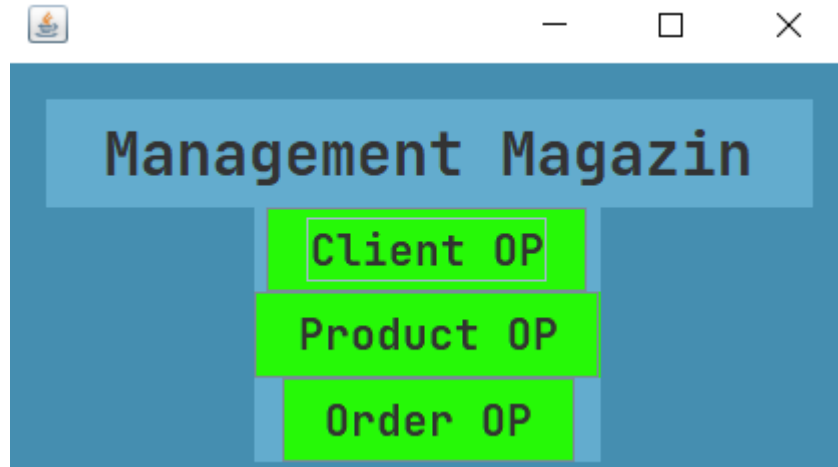
<i>CUPRINS</i>	2
1. <i>Obiectivul temei</i>	3
2. <i>Analiza problemei, modelare</i>	4
3. <i>Proiectare</i>	5
4. <i>Implementare</i>	7
5. <i>Concluzii</i>	13
6. <i>Bibliografie</i>	13

1. Obiectivul temei

Obiectivul principal al temei reprezintă proiectarea și implementarea unui program de organizare pentru un magazin.

Un magazin ar trebui să poată aibă în baza sa de date un tabel cu clienți, unul cu produsele stocate și unul cu comenzile pe care le depun clienții.

Utilizatorul va interacționa cu această aplicație folosind o interfață grafică interactivă:

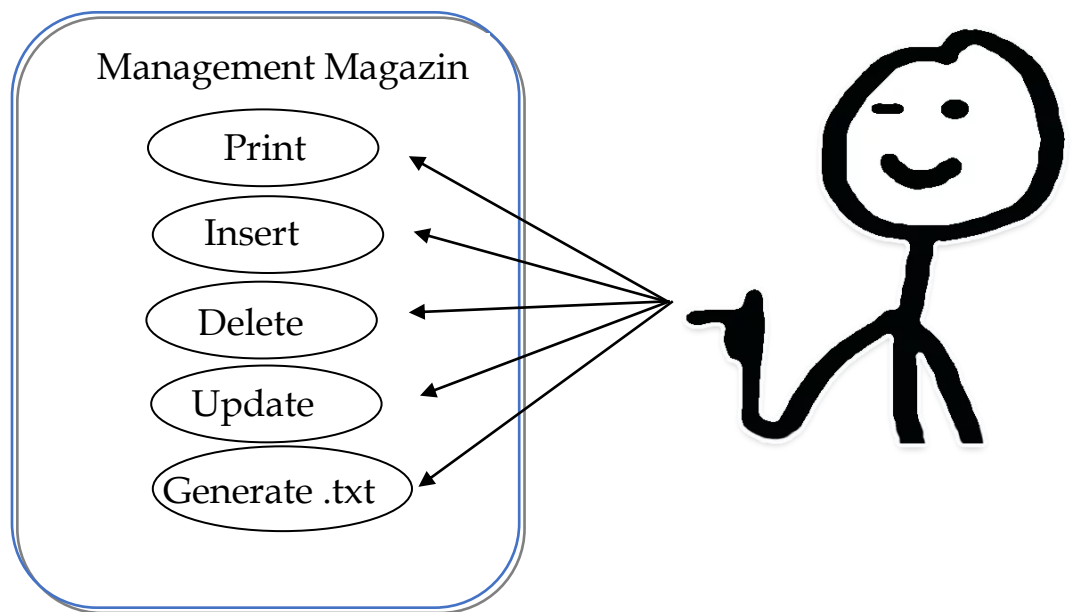


2. Analiza problemei, modelare

Calculatorul are disponibile următoarele operații:

- ➔ Afișarea obiectelor;
- ➔ Inserarea în baza de date a unui obiect;
- ➔ Ștergerea unui obiect din baza de date;
- ➔ Modificarea unui obiect din baza de date;
- ➔ Generarea unui fișier de tip text pentru afișarea unui obiect;

Se vor afișa mai multe ferestre care permit utilizatorului să efectueze operații asupra tabelor *Client*, *Product*, *Orders*.



unde:

Print – este operația de Afișare obiectelor;

Insert – este operația de Inserare în baza de date a unui obiect;

Delete - este operația de Ștergere unui obiect din baza de date;

Update - este operația de Modificare unui obiect din baza de date;

Generate .txt - este operația de Generarea unui fișier de tip text pentru afișarea unui obiect.

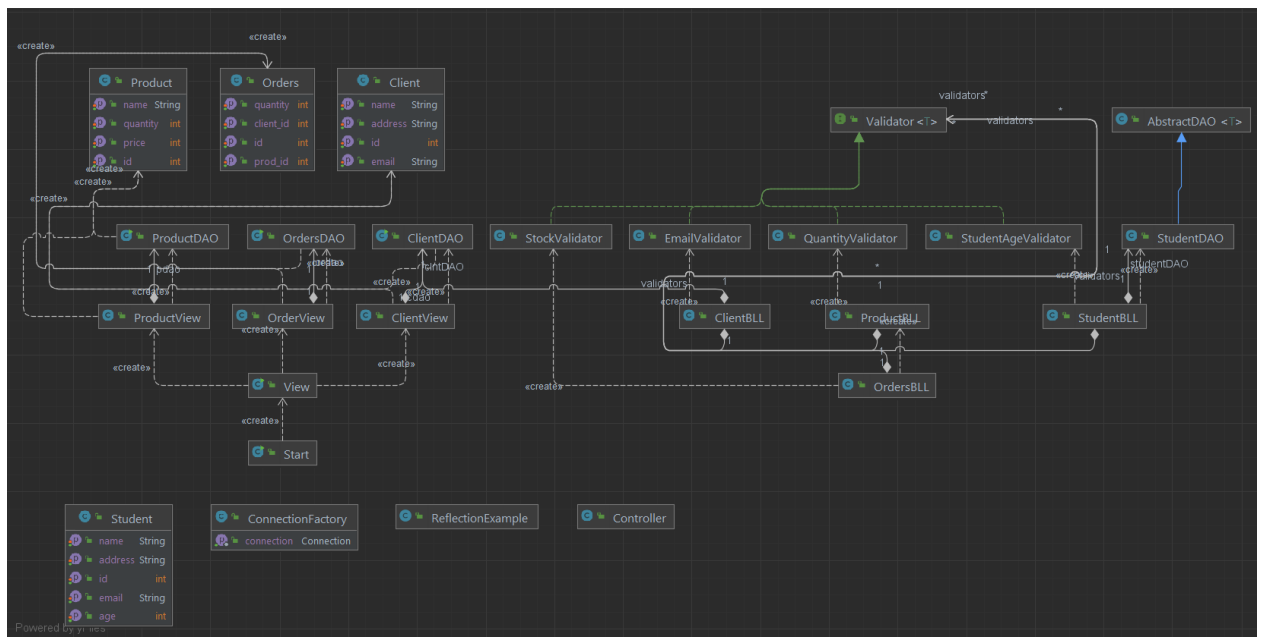
3. Proiectare

Proiectul este organizat pe nivele (*layers*) și se folosește tehnica reflexiei (*Reflection*).

Diagrama UML (Unified Modeling Language) reprezintă structura logică a proiectului. Specifică împărțirea în pachete, clase și relații.

Ea poate conține și atributele, constructorii și metodele claselor.

Diagrama UML:



Proiectul va conține 6 Pachete pentru implementare:

Pachetul *bll* conține pachetul *validators* și clasele:

- *ClientBLL;*
- *ProductBLL;*
- *OrdersBLL;*
- *StudentBLL;*(exemplu sursă)

Pachetul *validators* conține clasele:

- *EmailValidator;*
- *QuantityValidator;*
- *StockValidator;*
- *Operations;*
- *Validator* (exemplu sursă);
- *StudentAgeValidator* (exemplu sursă);

Pachetul *connection* conține clasa:

- *ConnectionFactory;*

Pachetul *dao* conține clasele:

- *AbstractDAO;*(exemplu sursă)
- *ClientDAO;*
- *ProductDAO;*
- *OrdersDAO;*
- *StudentDAO;*(exemplu sursă)

Pachetul *model* conține clasele:

- *Client;*
- *Product;*
- *Orders;*
- *Student;*(exemplu sursă)

Pachetul *presentation* conține clasele:

- *View;*
- *ClientView;*
- *ProductView;*
- *OrderView;*
- *Controller;*(neimplementat)

Pachetul *start* conține clasa:

- *ReflectionExample;*
- *Start.*

4. Implementare

Pachetul *bll*:

Clase:

- ***ClientBLL***

Clasa ClientBLL se ocupă de partea de encapsularea operațiilor logice ale aplicației referitoare la entitatea *Client*. Aici se apelează validatorul de *email* la crearea unui nou *Client*, astfel încât să nu se permită introducerea unui client cu o adresă de email invalidă.

- ***ProductBLL***

Clasa ProductBLL se ocupă de partea de encapsularea operațiilor logice ale aplicației referitoare la entitatea *Produs*. În această clasă se apelează validatorul de *cantitate* al produsului, astfel încât să nu se permită introducerea unui nou produs care are o cantitate negativă, altfel spus: invalidă.

- ***OrdersBLL***

Clasa OrdersBLL se ocupă de partea de encapsularea operațiilor logice ale aplicației referitoare la entitatea *Comandă*. În această clasă se apelează validatorul de verificare a stockului produsului, astfel încât să nu se permită plasarea unei noi comenzi a unui produs din care nu există suficient stock de scăzut odata cu comanda, rezultând ca produsul ulterior să își actualizeze cantitatea ca fiind una negativă, ceea ce nu este permis.

Pachetul *validators*:

Clase:

- ***EmailValidator***

Această clasă are rolul de a nu permite inserarea unui email care nu este conform cu șablonul standard al unei adrese de email.

- ***QuantityValidator***

Această clasă are rolul de a nu permite inserarea unui produs nou dacă acesta nu are o cantitate pozitivă.

- ***StockValidator***

Această clasă are rolul de a nu permite plasarea unei noi comenzi a unui produs dacă nu există suficient stock disponibil.

Pachetul *connection*:

Clasa:

- ***ConnectionFactory***

Clasa ce stabilește conexiunea cu baza de date creată și întreținută de aplicația MySQL este clasa ***ConnectionFactory***.

Pachetul *dao*:

Clase:

- ***ClientDAO***

Această clasă este responsabilă de acțiunile legate de accesul la baza de date și de operațiile ce se fac pe aceasta. Aceste operații sunt de tipul *Create*, *Read*, *Update*, *Delete*. Operațiile disponibile ale acestei aplicații sunt: de afișare a conținutului tabelului aferent, respectiv tabelul *Client*, din baza de date, inserarea unui rând nou, alterarea/modificarea unui rând existent sau ștergerea unui rând existent.

Se formează un șir de caractere ce reprezintă interogarea tabelului din baza de date:

```
private final static String findStatementString = "SELECT * FROM client WHERE id = ?";
```

Similar se procedează și în cazul inserării unui nou rând, a modificării unui existent sau a ștergerii unui existent.

- ***ProductDAO***

Această clasă este responsabilă de acțiunile legate de accesul la baza de date și de operațiile ce se fac pe aceasta. Operațiile disponibile ale acestei aplicații sunt: de afișare a conținutului tabelului aferent, respectiv tabelul *Produs*, din baza de date, inserarea unui rând nou, alterarea/modificarea unui rând existent sau ștergerea unui rând din baza de date. Aceste operații sunt de tipul *Create*, *Read*, *Update*, *Delete*.

Se formează un șir de caractere ce reprezintă interogarea tabelului din baza de date:

```
private final static String findStatementString = "SELECT * FROM product WHERE id = ?";
```

Similar se procedează și în cazul inserării unui nou rând, a modificării unui existent sau a ștergerii unui existent.

La inserarea unui nou produs se verifică dacă acesta are o cantitate pozitivă. Dacă acesta este conform cu cerința, atunci noul produs se inserează cu succes în baza de date, altfel se transmite un mesaj explicativ.

- **OrdersDAO**

Această clasă este responsabilă de acțiunile legate de accesul la baza de date și de operațiile ce se fac pe aceasta. Aceste operații sunt de tipul *Create*, *Read*, *Update*, *Delete*. Operațiile disponibile ale acestei aplicații sunt: de afișare a conținutului tabelului aferent, respectiv tabelul *Comandă*, din baza de date, inserarea unui rând nou, alterarea/modificarea unui rând existent sau ștergerea unui rând din baza de date.

Se formează un șir de caractere ce reprezintă interogarea tabelului din baza de date:

```
private final static String findStatementString = "SELECT * FROM orders WHERE id = ?";
```

Similar se procedează și în cazul inserării unui nou rând, a modificării unui existent sau a ștergerii unui existent.

La inserarea unei noi comenzi se verifică dacă există suficient stock din produsul respectiv. Dacă există suficient stock, atunci comanda se plasează cu succes, altfel se transmite un mesaj explicativ.

Pachetul *model*:

Clase:

- **Client**

Clasa aceasta reprezintă unul din obiectele principale ale aplicației, și anume *Clientul*. Are câmpurile *id*, *name*, *address*, *email*. Câmpul *id* este un identificator unic al acestui obiect în baza de date. Câmpul *name* este de forma String și reprezintă numele clientului. Câmpul *address* este de forma String și reprezintă adresa clientului. Câmpul *email* este de forma String și reprezintă adresa de email a clientului.

- **Product**

Clasa aceasta reprezintă unul din obiectele principale ale aplicației, și anume *Produsul*. Are câmpurile *id*, *name*, *price*, *quantity*. Câmpul *id* este un identificator unic al acestui obiect în baza de date. Câmpul *name* este de forma String și reprezintă numele produsului. Câmpul *price* este de tip *int* și reprezintă prețul produsului clientului. Câmpul *quantity* este de tip *int* și reprezintă cantitatea totală disponibilă a produsului.

- **Orders**

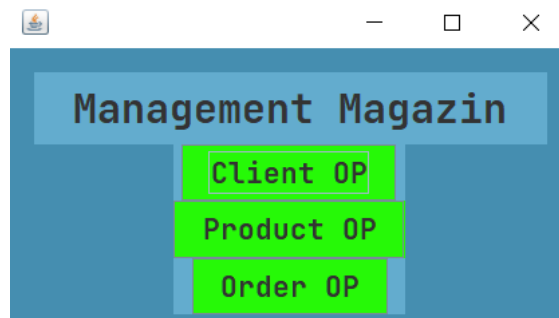
Clasa aceasta reprezintă unul din obiectele principale ale aplicației, și anume *Comanda*. Are câmpurile *id*, *prod_id*, *client_id*, *quantity*. Câmpul *id* este un identificator unic al acestui obiect în baza de date. Câmpul *prod_id* este de tip *int* și reprezintă id-ul produsului cumpărat de către client. Câmpul *client_id* este de tip *int* și reprezintă id-ul clientului ce cumpără respectivul produs. Câmpul *quantity* este de tip *int* și reprezintă cantitatea produsului ce dorește a fi cumpărat de către client.

Pachetul *presentation*:

Clase:

- *View*

În această clasă se creează o fereastră de vizionare și manipulare a datelor celor 3 tabele. Conține 3 butoane reprezentative operațiilor pe cele 3 tabele. La apăsarea unui buton dintre cele 3, se face invizibilă fereastra curentă și se deschide una nouă reprezentativă operațiilor posibil de efectuat pe tabelul aferent.



- *OrderView*

Clasa aceasta constă în reprezentarea vizuală a operațiilor disponibile pe tabelul aferent *Comenzilor*: Print – afișarea conținutului tabelului; Insert – inserarea unui nou rând în tabel; Delete – ștergerea după *id* unui rând existent în tabel; Update – modificarea unui element din tabel; PDF – exportarea conținutului tabelului sub forma unui fișier .txt.

Id	Name	Adress	Email
1	1	1	2
2	1	2	50

- **ClientView**

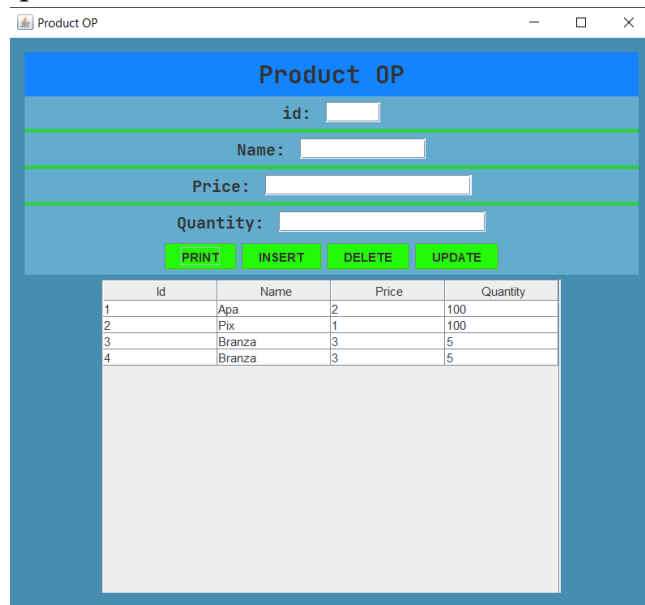
Clasa aceasta constă în reprezentarea vizuală a operațiilor disponibile pe tabelul aferent **Clientilor**: Print – afișarea conținutului tabelului; Insert – inserarea unui nou rând în tabel; Delete – ștergerea după id unui rând existent în tabel; Update – modificarea unui element din tabel.

Id	Name	Adress	Email
1	Ilie	home1	ilie@gmail.com
2	Nelu	home2	nelu@gmail.com
14	Clark	home3	Clark@gmail.com
15	Dorel	home55	Dorel@gmail.com
16	Relu	homeAlone2	relu@yahoo.com

- **ProductView**

Clasa aceasta constă în reprezentarea vizuală a operațiilor disponibile pe tabelul aferent **Produselor**: Print – afișarea conținutului tabelului; Insert –

inserarea unui nou rând în tabel; Delete – ștergerea după id unui rând existent în tabel; Update – modificarea unui element din tabel.



Id	Name	Price	Quantity
1	Apa	2	100
2	Pix	1	100
3	Branza	3	5
4	Branza	3	5

Pachetul *start*:

Clase:

- *Start*

În clasa aceasta se instanțiază un obiect de tip *View*.

- *ReflectionExample*

În această clasă se află două funcții care exprimă metoda *Reflection* din acest proiect. Cele două funcții se numesc *retrieveColumnTitles*, respectiv *retrieveProperties*. Ele sunt folosite pentru a determina proprietăți unui obiect: Numele coloanelor și datele din celulele rândurilor din tabel. Acestea sunt transmise mai departe în consolă spre afișare. Afișarea nu se face standard, ci este modificată pentru a avea un anumit format/aspect:

```
=====
id      name      address  email
-----
1       1         1        1
2       2         2        2
=====
```

5. Concluzii & Rezultate

Acest proiect este o implementare a unui sistem de gestionare a unui magazin. Poate fi util în domeniul comercial.

Această implementare poate fi îmbunătățită atât vizual cât și din punct de vedere al performanței.

Din punct de vedere al rezultatelor, acestea pot fi obținute intuitiv folosind interfața grafică și apăsând butonul "PRINT" pentru a actualiza afișarea tabelului cu datele ce sunt în acel moment în baza de date.

6. Bibliografie

Link Resurse:

https://dsrl.eu/courses/pt/materials/PT2023_A3_S1.pdf

https://dsrl.eu/courses/pt/materials/PT2023_A3_S2.pdf

<https://stackoverflow.com/questions/10566823/list-table-creation-in-java>

<https://stackoverflow.com/questions/2745206/output-in-a-table-format-in-javas-system-out>

<https://docs.oracle.com/javase/tutorial/uiswing/components/table.html>

<https://docs.oracle.com/javase/8/docs/api/javax/swing/JTable.html#setAutoResizeMode-int->