



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

4/18/2021

Secția Calculatoare și Tehnologia
Informației

TEMA 3
ORDER MANAGEMENT
DOCUMENTAȚIE

Sirghi Paula
GRUPA 30223

Table of Contents

1. Obiectivul temei.....	2
2. Analiza problemei, modelare, scenarii, cazuri de utilizare	2
.....	3
.....	3
3. Proiectare.....	3
4. Implementare	4
• Clasa Start.....	4
• Clasa MainView	4
• Clasa ClientView	5
• Clasa ProducView	5
• Clasa OrderView	6
• Clasa Eroare	7
• Clasa Succes.....	7
• Clasa Controller	7
• Clasa Client	7
• Clasa Produs	7
• Clasa Coamnda	7
• Clasa AbstractDAO.....	7
• Clasa ClientDAO.....	8
• Clasa ProdusDAO.....	8
• Clasa ComandaDAO.....	9
• Clasa ConnectionFactory	9
• Clasa ClientBLL.....	9
• Clasa ComandaBLL.....	9
• Clasa ProdusBLL.....	9
• Clasele Validator.....	9
• Interfața Validator	9
5. Rezultate	9
6. Concluzii	10
7. Bibilografie.....	10

1. Obiectivul temei

Obiectivul acestei teme este reprezentat de implementarea unei aplicații ce are ca scop gestionarea unor comenzi pentru a procesa comenzile unor clienți de la un depozit. Bazele de date relaționale sunt folosite pentru a stoca produsele, clienții și comenzile. Aplicația este structurată în pachete folosind arhitectura pe nivele, având clasele de bază de tip Model, Business Logic, Presentation și Data access. Conform cerințelor problemei, comenzile sunt plasate în funcție de selecția unui produs din stoc, a unui client existent în baza de date și a unei cantități, iar dacă aceasta nu depășește stocul disponibil, comanda se va realiza cu succes și implicit și factura, urmând să se reactualizeze stocul disponibil al aceluși produs în baza de date.

Modul de citire al datelor introduse se face simplu, urmând ca textul introdus să fie convertit conform unui șablon în concordanță cu cerințele temei, iar rezultatul va fi scris în baza de date, iar în cazul generării comenzilor, se va face o factură pentru fiecare comandă care va fi scrisă într-un fișier. Astfel, se vor modifica datele din tabelele SQL pentru operații de tipul inserare, ștergere, update sau se va scrie un nou fișier pentru operația de plasare a unei comenzi.

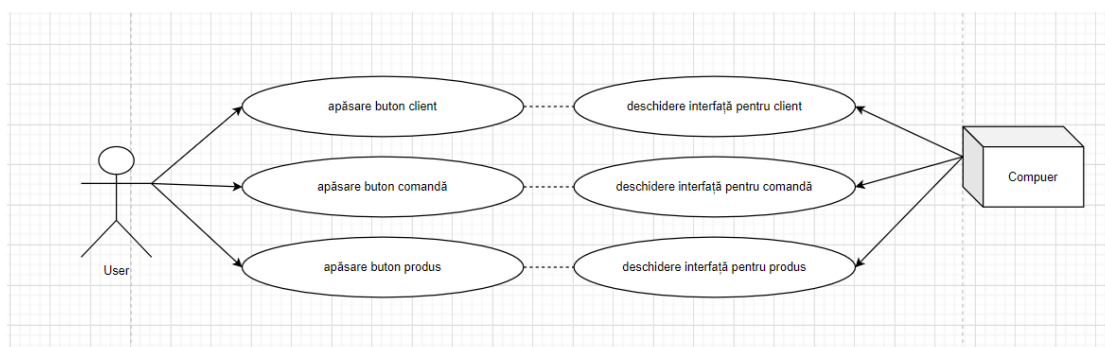
2. Analiza problemei, modelare, scenarii, cazuri de utilizare

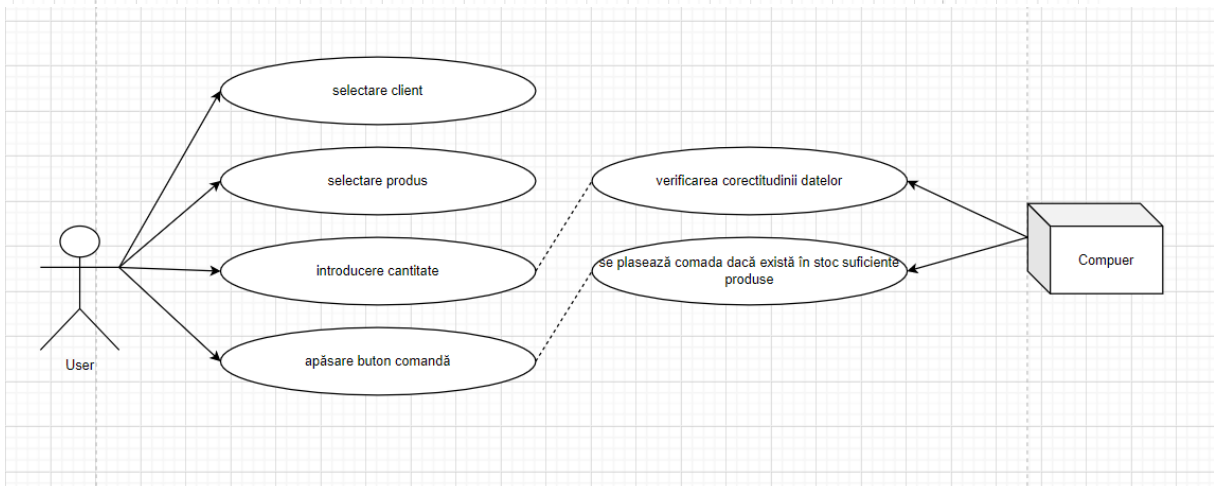
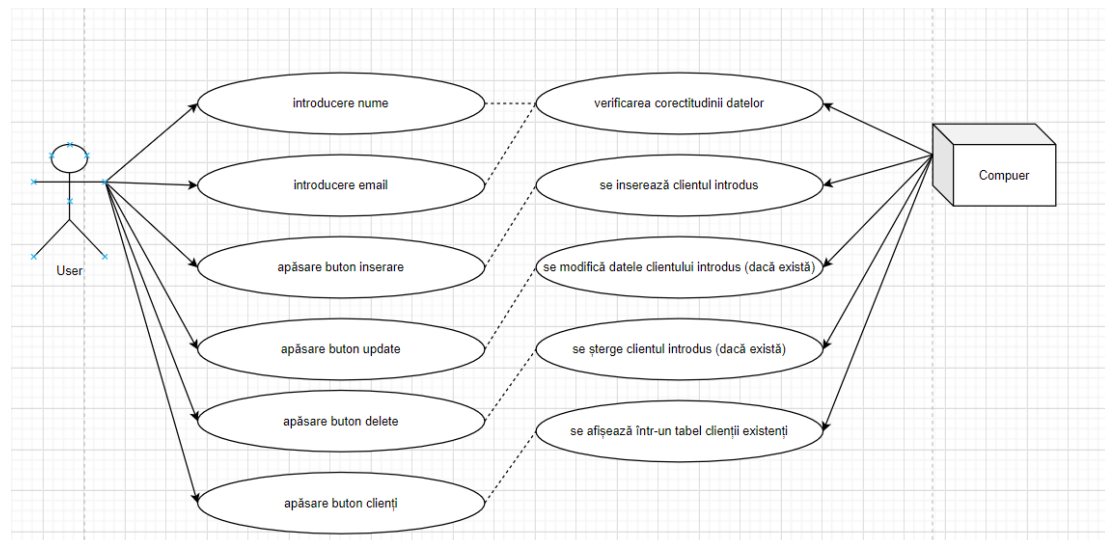
Analiza problemei constă, inițial, în înțelegerea cerinței problemei și implicit implementarea unui plan inițial pentru rezolvarea sa care, după cum am menționat mai sus, necesită implementarea operațiilor cerinței, operații de tipul: insert, update, delete, select și generare factură. Ca prim pas, vom alege clasele necesare (substantive) și metodele reprezentative pentru fiecare (verbe) pentru a putea realiza o aplicație funcțională și ușor de înțeles pentru orice cunoscător de java.

O altă etapă reprezentativă constă în stabilirea intrărilor și ieșirilor aplicației noastre. În cazul de față intrările depind de interfața grafică în care ne aflăm. Dacă suntem în interfața pentru client avem 4 intrări: 2 nume și 2 email-uri, în interfața pentru produs avem 4 intrări, 1 nume, 1 preț și două cantități, iar în interfața pentru comandă avem 3 intrări, un client, un produs și o cantitate.

Am luat în calcul eventualele erori de introducere a unor date care nu sunt în conformitate cu intrările dorite, iar ca urmare utilizatorul va primi un mesaj de eroare în acest caz, aceste erori fiind tratate, în principal, în clasele Validator.

Use case-urile de mai jos evidențiază pașii ce sunt parcurși atât de utilizator, cât și de computer pentru a se obține rezultatul așteptat în momentul execuției programului.

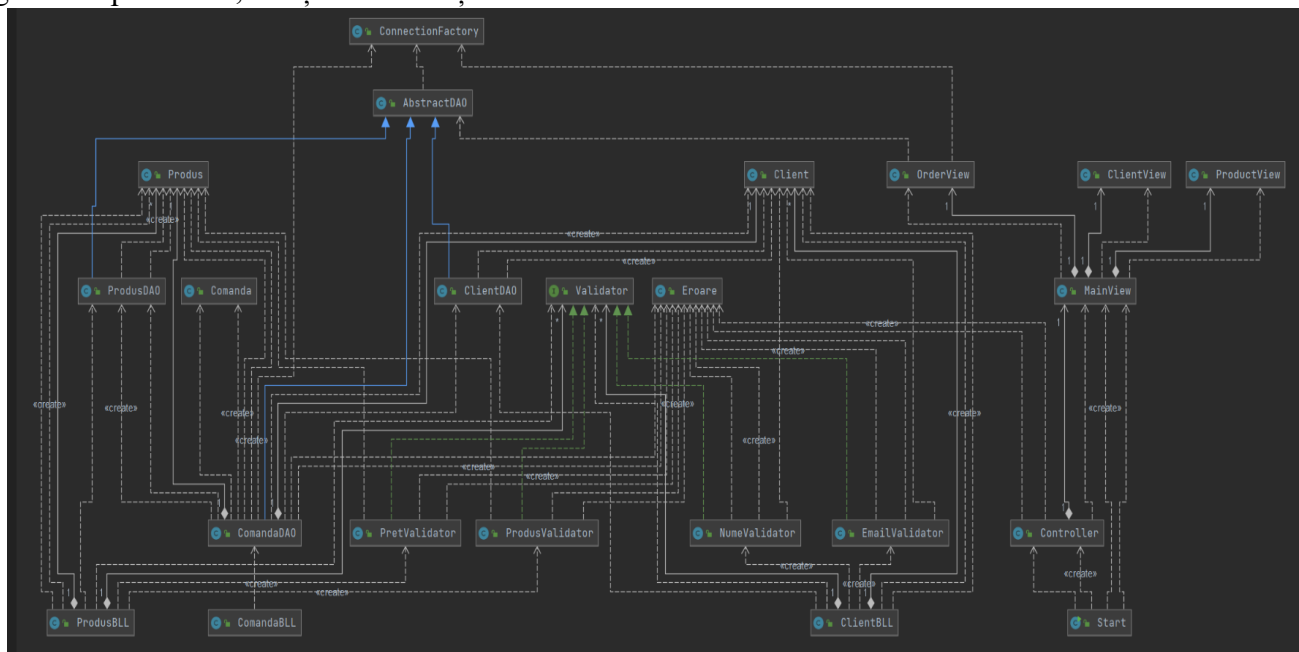




3. Proiectare

Pentru reprezentarea temei am ales o arhitectură structurată. Pachetul bll conține clasele ClientBLL, ComandaBLL și ProdusBLL, dar și pachetul valisator care conține clasele și interfața folosite pentru validarea datelor de intrare. Pachetul connection conține clasa care realizează conexiunea cu baza de date. Pachetul dao conține clasele DAO. Pachetul model


Diagrama UML este reprezentată mai jos, iar în ea putem observa clasele proiectului cu legăturile specificate, dar și metodele și structurile de date folosite.



Descrierea claselor se va face pe baza diagramei UML de clase reprezentată mai sus.

- Această clasă reprezintă clasa de bază a proiectului, clasa de unde se începe simularea, fiind clasa ce conține metoda statica main care conține instanțierea unui obiect MainView care reprezintă interfața grafică de bază a proiectului și a controller-ului.

- Reprezentativă pentru GUI-ul inițial în care utilizatorul selectează meniul în care vrea să lucreze: client, produs sau comandă.



A screenshot of a web browser window. The address bar shows a URL starting with 'http://'. The page content includes three input fields with labels 'client', 'comanda', and 'produs' below them. The 'client' field is currently selected.

Constructorul clasei realizează interfața grafică pe care o vede utilizatorul în momentul rulării programului, precum se vede în poza de mai sus.

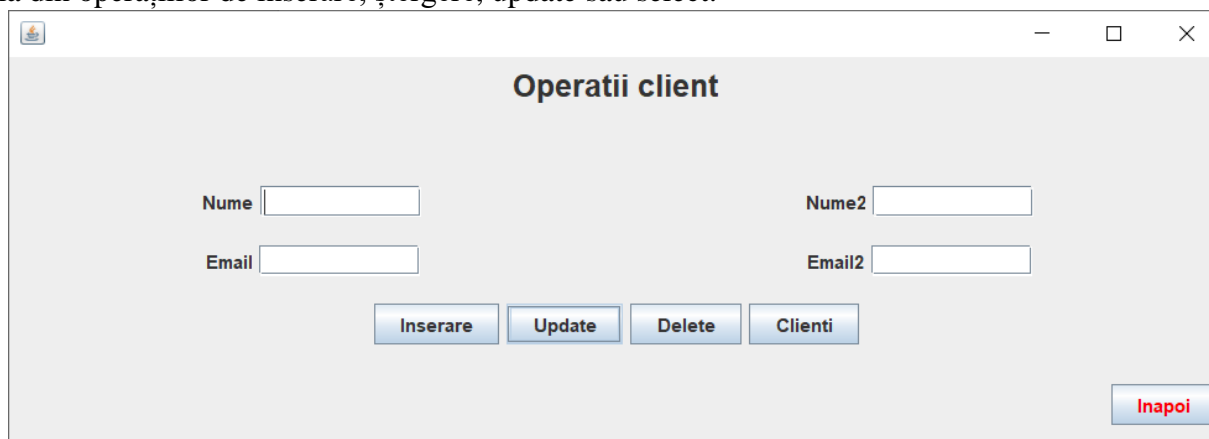
Metodele client, produs și comandă sunt folosite pentru ca cele trei butoane să implementeze ActionListener-I, fiind folosite în controller și detaliate mai bine.

Pe lângă metodele menționate mai sus mai avem și gettere și settere necesare datorită folosirii tehnicii încapsulării datelor.

- **Clasa ClientView**

Reprezentativă pentru GUI-ul clientului în care utilizatorul introduce date pentru realizarea operațiilor reprezentative pentru un obiect de tipul client, respectiv pentru tabela client din baza de date.

Aceasta conține 4 JTextField-uri, 5 JLabel-uri și 5 Jbuttons, fapt care poate fi observat în poza de mai jos a interfeței grafice în care utilizatorul introduce noi date pentru o nouă execuție uneia din operațiilor de inserare, ștergere, update sau select.



În constructorul clasei se realizează apelul metodelor de tipul JPanel care sunt folosite la aranjarea panourilor folosite în interfața grafică. Tot aici se realizează și alcătuirea Jpanelului principal pentru această interfață grafică care va avea aspectul de mai sus.

Metodele ins, upd, delete, listare și înapoi sunt folosite pentru a adăuga ActionListeners butoanelor interfeței, funcțiile lor fiind stabilite în clasa Controller.

Alte metode ale acestei clase sunt getterele getTfNume, getTfEmail, getTfNume2, getTfEmail2, metode necesare datorită folosirii încapsulării datelor.

- **Clasa ProductView**

Reprezentativă pentru GUI-ul pentru produs în care utilizatorul introduce date pentru realizarea operațiilor reprezentative pentru un obiect de tipul produs, respectiv pentru tabela produs din baza de date.

Aceasta conține 4 JTextField-uri, 5 JLabel-uri și 5 Jbuttons, fapt care poate fi observat în poza de mai jos a interfeței grafice în care utilizatorul introduce noi date pentru o nouă execuție uneia din operațiilor de inserare, ștergere, update sau select.

În constructorul clasei se realizează apelul metodelor de tipul JPanel care sunt folosite la aranjarea panourilor folosite în interfața grafică. Tot aici se realizează și alcătuirea Jpanelului principal pentru această interfață grafică care va avea aspectul de mai sus.

Metodele ins, upd, delete, listare și înapoi sunt folosite pentru a adăuga ActionListeners butoanelor interfeței, funcțiile lor fiind stabilite în clasa Controller.

Alte metode ale acestei clase sunt getterele getTfNume, getTfCantitate, getTfNume2, getTfPret, metode necesare datorită folosirii încapsulării datelor.

- [Clasa OrderView](#)

Reprezentativă pentru GUI-ul pentru comandă în care utilizatorul introduce date pentru realizarea operațiilor reprezentative pentru un obiect de tipul comandă, respectiv pentru tabela comandă din baza de date.

Aceasta conține 1 JTextField, 2 JComboBox-uri și 2 Jbutton, fapt care poate fi observat în poza de mai jos a interfeței grafice în care utilizatorul introduce noi date pentru o nouă execuție a operației de adaugare a unei comenzi.

În constructorul clasei se realizează apelul metodelor de tipul JPanel care sunt folosite la aranjarea panourilor folosite în interfața grafică. Tot aici se realizează și alcătuirea Jpanelului principal pentru această interfață grafică care va avea aspectul de mai sus. Apelurile metodelor pr și cl sunt realizate tot în constructor.

Metoda pr realizează un query de select al numelor tuturor produselor prezente în tabela produs din baza de date, urmând să pună rezultatul în Jcombobox-ul corespunzător produselor.

Metoda `cl` realizează un query de select al numelor tuturor clienților prezenți în tabela client din baza de date, urmând să pună rezultatul în JcomboBox-ul corespunzător clienților.

Metodele `ins` și `înapoi` sunt folosite pentru a adăuga `ActionListeners` butoanelor interfeței, funcțiile lor fiind stabilite în clasa `Controller`.

Alte metode ale acestei clase sunt getterele `getTfProdus`, `getTfCantitate`, `getClient`, și setterele `setClient` și `setProdus`, metode necesare datorită folosirii tehnicii încapsulării datelor.

- [Clasa Eroare](#)

Această clasă este folosită pentru a afișa un mesaj de eroare utilizatorului în cazul întâmpinării unui caz defavorabil. Constructorul ei este folosit pentru a instanția această interfață grafică.

- [Clasa Succes](#)

Această clasă este folosită pentru a afișa un mesaj de succes utilizatorului în cazul în care operația dorită s-a realizat cu succes. Constructorul ei este folosit pentru a instanția această interfață grafică.

- [Clasa Controller](#)

Această clasă realizează legătura între interfețele grafice ale programului, conținând clase pentru fiecare buton din acestea, clase în care se vor apela metodele corespunzătoare din clasele DAO menționate mai jos.

- [Clasa Client](#)

Această clasă este reprezentativă pentru tabela client din baza de date. Pe lângă constructorul ei care are scopul de a instanția un obiect de tipul `Client`, mai avem și gettere și settere necesare datorită folosirii încapsulării datelor.

- [Clasa Produs](#)

Această clasă este reprezentativă pentru tabela produs din baza de date. Pe lângă constructorul ei care are scopul de a instanția un obiect de tipul `Produs`, mai avem și gettere și settere necesare datorită folosirii încapsulării datelor.

- [Clasa Comanda](#)

Această clasă este reprezentativă pentru tabela comanda din baza de date. Pe lângă constructorul ei care are scopul de a instanția un obiect de tipul `Comanda`, mai avem și gettere și settere necesare datorită folosirii încapsulării datelor.

- [Clasa AbstractDAO](#)

Această clasă este reprezentativă pentru tehnica reflexiei, conținând metodele necesare realizării operațiilor pe fiecare tabelă din baza noastră de date.

Metoda `createFindQuery` realizează conținutul unui `String` pe care îl va returna, acesta conținând un `SELECT` după un câmp dat dintr-un tabel oarecare.

Metoda `createDeleteQuery` realizează conținutul unui `String` pe care îl va returna, acesta conținând un `DELETE` după un câmp al parametrului obiect de tipul `T`, interogare realizată pentru un tabel oarecare, în funcție de obiectul ce o apelează.

Metoda `createUpdateQuery` realizează conținutul unui `String` pe care îl va returna, acesta conținând un `UPDATE` după un câmp al parametrului obiect de tipul `T`, modificările fiind luate

din parametrul object2 de tipul T, interogare realizată pentru un tabel oarecare, în funcție de obiectul ce o apelează. Clauza where a acestui update se realizează în metoda apelată createWhereClause, iar clauza set în metoda createSetClaus.

Metoda createInsertQuery realizează conținutul unui String pe care îl va returna, acesta conținând un INSERT a unui obiect obținut din parametrul object de tipul T, interogare realizată pentru un tabel oarecare, în funcție de obiectul ce o apelează.

Metoda createViewAllQuery realizează conținutul unui String pe care îl va returna, acesta conținând un SELECT a elementelor unui tabel dat de clasa ce o apelează.

Metoda ViewAll realizează apelul interogării create în metoda de mai sus, rezultatul apelului fiind returnat pentru a putea fi folosit în metoda createTable.

Metoda createTable creează un Jtable pe care îl și returnează în funcție de rezultatul obținut de metoda de mai sus și de numele coloanelor obținute în columnNames.

Metoda add realizează apelul interogării returnate de createInsertQuery, fapt care are ca urmare introducerea unui nou obiect într-o tabelă în funcție de clasa ce o apelează.

Metoda remove realizează apelul interogării returnate de createDeleteQuery, fapt care are ca urmare ștergerea unui obiect dintr-o tabelă în funcție de clasa ce o apelează.

Metoda update realizează apelul interogării returnate de createUpdateQuery, fapt care are ca urmare modificarea câmpurilor unui obiect într-o tabelă în funcție de clasa ce o apelează.

Metoda findById realizează apelul interogării returnate de createFindQuery, fapt care are ca urmare returnarea câmpurilor dintr-o tabelă în funcție de clasa ce o apelează și de id-ul specificat ca parametru.

Metoda createObject realizează adăugarea într-o listă a obiectelor formate în urma unei interogări, rezultatul fiind memorat în parametrul resultSet.

Metoda getCols returnează coloanele unui tabel în funcție de obiectul dat ca parametru, obiectul fiind de tipul T.

- [Clasa ClientDAO](#)

Metoda view realizează afișarea unui Jtable format în urma apelului metodei ViewAll din clasa AbstractDAO.

Metoda addC realizează adăugarea unui client în baza de date, apelând metoda add din clasa AbstractDAO.

Metoda removeC realizează ștergerea unui client din baza de date, apelând metoda remove din clasa AbstractDAO.

Metoda updateC realizează modificarea unui client din baza de date, apelând metoda update din clasa AbstractDAO.

- [Clasa ProdusDAO](#)

Metoda view realizează afișarea unui Jtable format în urma apelului metodei ViewAll din clasa AbstractDAO.

Metoda addP realizează adăugarea unui produs în baza de date, apelând metoda add din clasa AbstractDAO.

Metoda removeP realizează ștergerea unui produs din baza de date, apelând metoda remove din clasa AbstractDAO.

Metoda updateP realizează modificarea unui produs din baza de date, apelând metoda update din clasa AbstractDAO.

- [Clasa ComandaDAO](#)

Metoda pr realizează selecția unui produs după nume, instanțiind obiectul p cu datele obținute.

Metoda pc realizează selecția unui client după nume, instanțiind obiectul c cu datele obținute.

Metoda fisier realizează crearea unui fișier text în care se va scrie factura, fișierul având numere în formatul numeClient_numeProdus.txt.

Metoda addC realizează adăugarea unei comenzi în baza de date, apelând metoda add din clasa AbstractDAO. Tot aici sunt apelate toate metodele clasei menționate mai sus.

- [Clasa ConnectionFactory](#)

Această clasă este folosită pentru a realiza conexiunea cu baza de date.

- [Clasa ClientBLL](#)

Această clasă este folosită pentru a verifica datele introduse de utilizator și pentru a apela metodele clasei ClientDAO.

- [Clasa ComandaBLL](#)

Această clasă este folosită pentru a verifica datele introduse de utilizator și pentru a apela metodele clasei ComandaDAO.

- [Clasa ProdusBLL](#)

Această clasă este folosită pentru a verifica datele introduse de utilizator și pentru a apela metodele clasei ProdusDAO.

- [Clasele Validator](#)

Aceste clase sunt folosite pentru a verifica corectitudinea datelor introduse de utilizator, implementând interfața Validator.

- [Interfața Validator](#)

Această interfață prezintă metoda validate care va fi suprascrisă de clasele ProdusValidator, PretValidator, NumeValidator, EmailValidator pentru a verifica corectitudinea datelor introduse de utilizator.

5. Rezultate

Rezultatele operațiilor au fost fie afișate în baza de date în tabelele corespunzătoare, fie în fișierele generate pentru fiecare factură, aceasta conținând numele clientului ce a plasat comanda, numele produsului comandat și prețul total al comenzii.

6. Concluzii

Din punctul meu de vedere, această temă a fost destul de interesantă și diferită față de celelalte două, de data aceasta fiind nevoiți să lucrăm și cu baze de date relaționale. În ciuda acestui fapt a fost o temă frumoasă deși parte cu reflexia a fost o noutate pentru mulți dintre studenți, nemaifolosind-o până acum.

7. Bibilografie

Resursele bibliografice sunt reprezentate de:

- suportul de laborator;
- suportul de curs de semestrul trecut de la materia Programare Orientată pe Obiect;
- tutotiale pe youtube pentru documentația cu javadoc
- tutoriale pentru reflexie;
- tutoriale pentru sqlDump;