**DOCUMENTATIE TEMA 4**

**FOOD DELIVERY MANAGEMENT SYSTEM**

**Nume prenume: Pașca Maria**

**Grupa 30229**

**Profesor Laborator Assist Antal Marcel**

Contents

[1. Cerinte Functionale 3](#_Toc476131445)

[2. Obiective 3](#_Toc476131446)

[2.1. Obiectiv Principal: 3](#_Toc476131447)

[2.2. Obective Secundare: 3](#_Toc476131448)

[3. Analiza Problemei 3](#_Toc476131449)

[4. Proiectare 3](#_Toc476131450)

[4.1. Structuri de date 3](#_Toc476131451)

[4.2. Diagrama de clase 3](#_Toc476131452)

[4.3. Algoritmi 3](#_Toc476131453)

[5. Implementare 4](#_Toc476131454)

[6. Testare 4](#_Toc476131455)

[7. Concluzii si Dezvoltari Ulterioare 4](#_Toc476131456)

[8. Bibliografie 4](#_Toc476131457)

# Cerinte Functionale

Cerința a fost dezvoltarea unei platforme de comandat mancare cu interfata grafica pentru login, signup, si separate pentru administrator, client si angajat. Astfel administratorul poate adauga, sterge, modifica produse, poate importa produse din fisierul products.csv si poate genera rapoarte in functie de criteriile pe care le doreste, clientul poate sa isi comande mancare si sa caute dupa un cuvant cheie produsele. In cazul clientului, pentru a plasa produse in cos, este de ajuns sa selecteze id-ul produsului din combo box-ul din dreptul labelului cu Id, iar pentru plasarea comenzii trebuie doar sa apese butonul de Finish. Asemenea pentru generarea rapoartelor de catre administrator, trebuie doar sa selecteze din combo box optiunea pentru raport pe care o doreste.

# Obiective

## Obiectiv Principal:

Obiectivul principal este proiectarea și implementarea unei aplicatii de comandat mancare, cu interfata de logare, de signup, si privilegii diferite pentru administrator, client si angajat.

## Obective Secundare:

|  |  |  |
| --- | --- | --- |
| **Obiectiv Secundar** | **Descriere** | **Capitol** |
| Dezvoltarea de use case-uri si scenarii | Descrierea modului de funcționare a programului în diferite situații | 3 |
| Alegerea structurilor de date | Prezentarea pe scurt a structurilor de date folosite | 4 |
| Diagrama de clase | Diagrama UML a proiectului | 4 |
| Implementarea solutiei | Descrierea programului pe larg, descrierea pachetelor, claselor, metodelor și relației dintre clase și pachete | 5 |
| Testare |  | 6 |

# Analiza Problemei

Use Case Name: Crearea unui cont nou

Actori:

* Utilizatorul
* Program
* Baza de date

Trigger:

* Utilizatorul a dat click pe butonul de Sign up

Precondition:

* Utilizatorul a introdus în câmpuri datele cerute corect

Normal Flow:

1. Utilizatorul apasa pe butonul de signup din fereastra de login, care il duce la o noua fereastra.
2. Utilizatorul introduce corect datele în textFieldurile aferente din fereastra de Signup.
3. Utilizatorul apasă butonul de Signup.
4. Listenerul butonului de Signup din SignupController apelează metoda checkInput din SingupController.
5. În checkInput se verifică dacă textfieldurile sunt goale sau nu, după care se revine in listenerul butonului de signup
6. In listener se apeleaza metoda createUser din clasa DeliveryService.
7. In metoda createUser din clasa DeliveryService se apeleaza metodele de validate.
8. Daca informatia introdusa respecta forma necesara, iar in cazul in care trece de aceste teste, noul client se adauga in lista de clienti si i se atribuie un index.
9. Dupa aceea, se revine in listenerul butonului de Signup, unde se verifica existenta clientului.
10. Daca clientul a fost creat, va afisa pe ecran un popup cu mesajul user created sucesfully.
11. Ulterior, se va reveni in fereastra de login, pentru logare.

Alternate Flow:

5A. Utilizatorul lasă câmpuri goale.

1. Va apărea pe ecran un popup care va avertiza utilizatorul asupra campurilor goale sau completate gresit.
2. Utilizatorul apasă ok și introduce și informațiile lipsă.

8A. Numarul de telefon nu respecta formatul cerut.

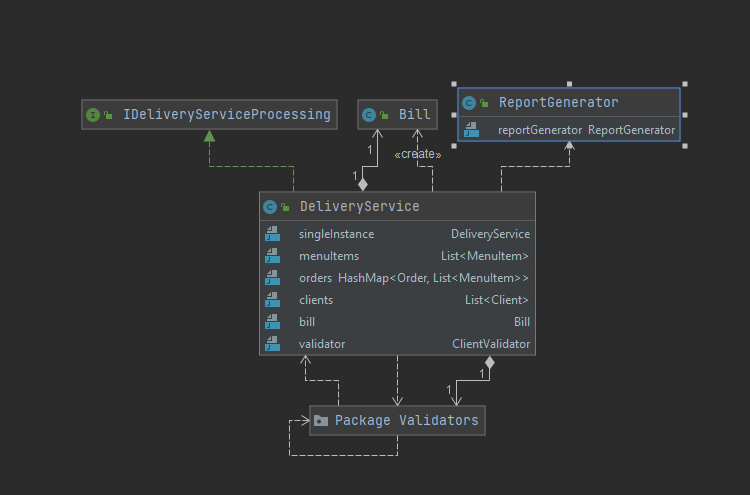
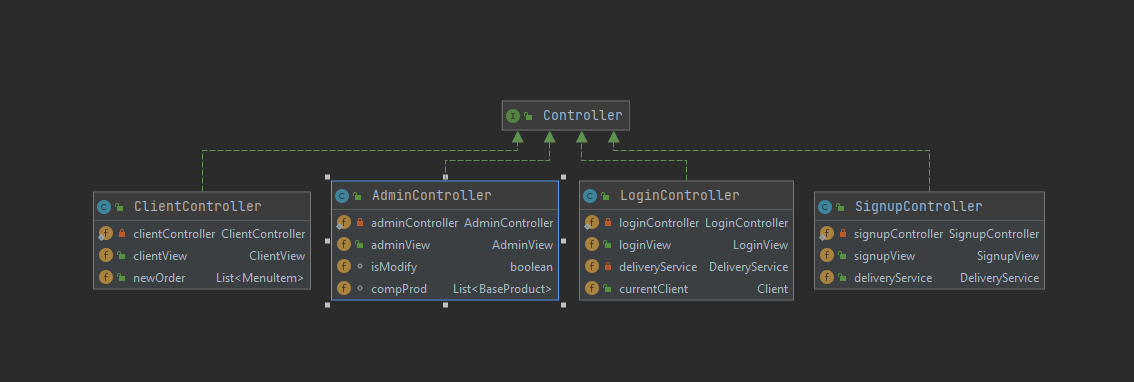
1. Va apărea în consolă eroarea ”Not a valid telephone number!”
2. Utilizatorul va putea reintroduce informatia in campuri.

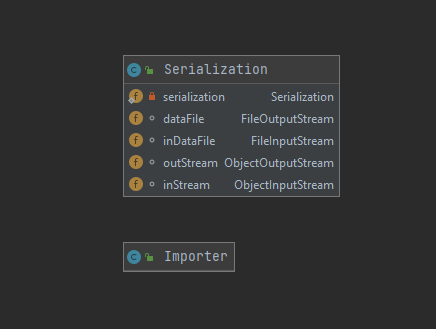
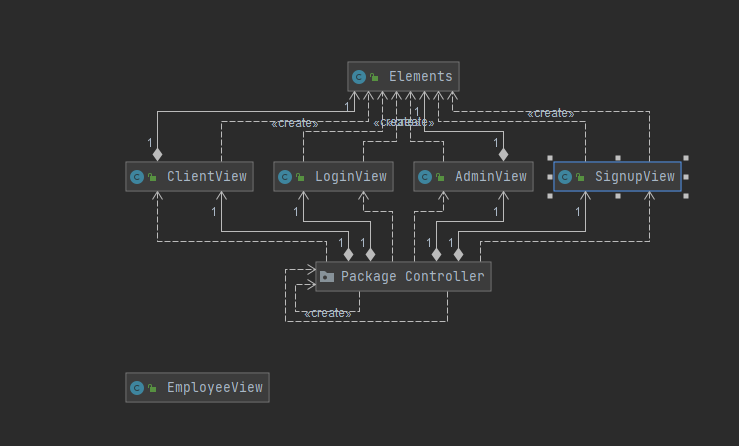
# Proiectare

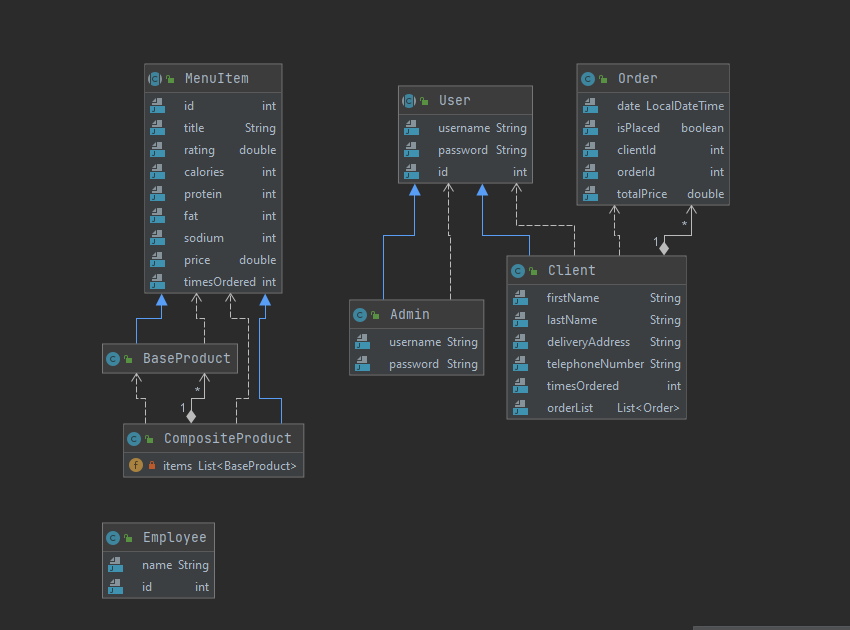
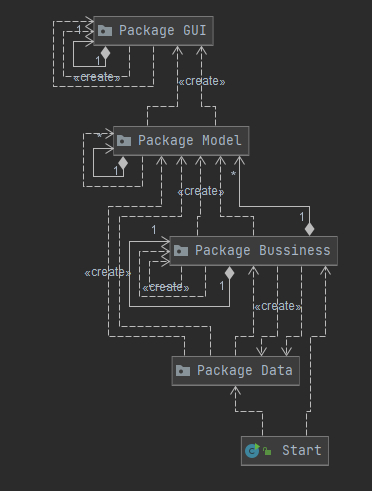
## Structuri de date

Structura de date cea mai răspândită în proiect este Singly-LinkedList-ul. Singly-LinkedLists sunt folosiți în acest proiect atât pentru a stoca și returna clienții, produsele și comenzile atunci când se apelează metodele de returnare a tuturor obiectelor si pentru stocarea produselor, userilor si comenzilor. De asemenea, s-a mai folosit si un HashMap in care sunt stocate listele de produse dintr-o comanda si a caror index este chiar obiectul Order atribuit listei de produse.

## Diagrama de clase

# Implementare

Pentru acest proiect am folosit layered architecture, în cadrul căreia am pachetele Bussiness, care conține logica de lucru a aplicației, Validators (în cadrul pachetului Bussiness), care conține validatorii prin care programul se asigură că informația introdusă este corectă și respectă formatul cerut, anume interfata Validator si clasele Telephone Validator, Username Validator si Client Validator, Data, care contine clasa Importer si clasa Serialization.

Pachetul Model conține clasele abstracte User si Menu Item, alaturi de clasele care le extind, Admin, Client si Employee, respectiv Base Product si Composite Product si clasa Order. Pe urma, pachetul GUI contine toate viewurile, Admin View, Client View, Employee View, Login View si Signup View, dar si clasa Elements si pachetul Controller, care tine interfata Controller, si clasele care o implementeaza, Admin Controller, Client Controller, Login Controller si Signup Controller.

In pachetul Data sunt pastrate clasele Importer si Serialization. Clasa importer se ocupa cu importarea produselor din fisierul .csv furnizat si sortarea acestuia, astfel incat sa nu apara duplicate in lista rezultata. Clasa Serialization se ocupa cu serializarea si deserializarea datelor stocate in program. Aceasta are metodele exportData, pentru Serializare, care deschide si citeste din fisierul data.ser, si importData pentru Deserializare, care suprascrie informatia in data.ser.

Pachetul Model tine toate obiectele care s-au folosit in acest program. Clasa abstracta user este folosita pentru a stoca informatiile generale despre un utilizator, fie el Client, Admin sau Employee, anume id, username si password, cu getterele si setterele si constructorul lor. Clasa Client are si prenumele si numele, adresa de livrare si numarul de telefon, dar si numarul de comenzi si lista cu comenzile pe care le-a efectuat. Clientul are metoda de incrementare a numarului de comenzi efectuate, care se apeleaza in cazul plasarii unei noi comenzi, si getterele campurilor sale, dar si metoda ToString, folosita in cadrul generarii rapoartelor. Clasa Admin contine constantele username si password, cu valorile „admin” si „admin”, valori cunoscute deja care nu pot fi modificate de catre utilizatori.

Clasa abstracta MenuItem este folosita pentru a stoca informatiile generale despre BaseProduct si CompositeProduct, anume id-ul, numele, ratingul, caloriile, proteinele, grasimile, sodiul si pretul, dar si numarul de dati de cate ori a fost comandat. Aceasta are in principal gettere, dar si o metoda incTimesOrdered, identica cu cea din Client, prin care este incrementat numarul de dati de cate ori a fost comandat produsul. Clasa CompositeProduct are o lista de BaseProducts din care este alcatuit meniul. Metoda de baza din aceasta clasa este AddItem, prin care nu numai ca este adaugat un Base Product listei de componente a meniului, insa este se adauga numarului de calorii, proteine, grasimi si sodiu, valorile acestor campuri din produsul nou adaugat. De asemenea, in metoda se mai regaseste un toString, folosit atat pentru chitanta, cat si pentru generarea rapoartelor. BaseProduct are la randul lui o metoda toString, de asemenea folosita pentru scrierea chitantelor si generarea rapoartelor.

In pachetul Bussiness se regasesc clasele Bill, Delivery Service cu interfata IDeliveryServiceProcessing si Report Generator si pachetul de validatori. Clasa Bill este cea in care se construieste si se scrie chitanta intr-un fisier de tip txt. Numele fisierului chitantei va fi chitanta #id-ul comenzii si va contine data si ora la care a fost efectuata comanda, id-ul si datele clientului care a plasat comanda si produsele comandate si pretul total al comenzii. Clasa ReportGenerator este cea in care se scriu rapoartele cerute de admin si are ca metode generateReportBy TimeInterval / TimesOrdered / DateAndTimesOrdered si generate Clients report. Acestea primesc ca parametru titlul fisierului si pe o lista cu elementele care corespund parametrilor raportului.

Pachetul Validators contine validatorii pentru crearea unui client nou. Telephone Validator verifica daca numarul de telefon corespunde patternului fixat printr-un regex. Validatorul UsernameValidator verifica ca username-ul ales de noul utilizator sa nu fie cumva admin, username care este rezervat pentru administrator, sau ca username-ul ales sa nu fi fost deja in uz, astfel trecand prin toata lista de clienti si comparand usernameurile.

Clasa DeliveryService este cea in care se intampla tot practic. Aici sunt stocate produsele, in lista menuItems, in menuItems, comenzile, in HashMap-ul orders, si clientii, in lista clients. Aici este apelata metoda importProducts, in care se instantiaza importer, a carui rezultat este stocat in menuItems. De asemenea are metoda addProduct, prin care este adaugat un nou produs in lista cu produse. Aici este apelata si functia create composite product, care ia ca parametrii o lista de produse, numele noului produs si pretul, creeaza produsul si il si adauga in lista de produse. Aici sunt apelate metodele de generate reports, care iau ca parametrii, parametrii raportului, si se folosesc de stream-uri pentru a filtra informatia in functie de parametrii alesi. Rezultatul acestora, pana sa fie returnat, este trimis in Report generator, unde urmeaza sa fie scris intr-un fisier. Metoda newOrder creeaza o comanda noua si ia ca parametri clientul care a facut comanda si lista de produse comandate. Comanda este adaugata in HashMap-ul cu comenzi, chitatanta este scrisa in fisierul text si returneaza id-ul comenzii. Metoda createUser verifica, in primul rand, daca userul nou creat este client, caz in care este trecut prin toti validatorii, ca mai apoi sa fie adaugat in lista de clienti si sa ii fie atribuit un id. In plus, mai sunt metode de fiind By Id pentru produse, fiind client by id pentru client si metoda is well formed, care verifica ca id-urile produselor, comenzilor si comenzilor sa nu aiba duplicate, astfel sa nu apara probleme mai departe in utilizarea programului.

Clasa GUI contine view-urile pentru admin, client, employee, pentru Login si pentru signup. View-ul pentru Login are un mainPanel, care contine titlul si panelul pentru credentiale, care la randul sau contine panelul pentru username si parola, care fiecare au un label si un text box, si butonul de login. In plus, mai contine si un panel de signup, cu un label cu mesaj de informare pentru utilizator, in cazul in care nu e logat, si cu un buton care ii deschide fereastra de sign up. Acesta este frame-ul cu care se incepe programul. Panelul de signup contine panelul pentru username, parola, prenume, nume, numar de telefon, adresa de livrare, fiecare continand un label si un textbox. Aici trebuie completate toate campurile ca, in urma apasarii butonului de signup, utilizatorul sa se poata inscrie in platforma fara probleme. Platforma clientului are un panelClient care contine toate panelurile, un panel main, unde sunt tinute panelurile pentru adaugarea produselor in cos si pentru cautarea produselor in lista. In jumatatea dreapta a ferestrei se afla tabelul cu produsele din platforma, iar daca utilizatorul doreste sa caute dupa un anumit cuvant cheie, tabelul din dreapta se va updata si va afisa doar produsele care contin acel cuvant cheie. Daca utilizatorul apasa pe refresh, tabelul va primi un refresh care sterge criteriile de cautare ale produselor. Butonul de logout delogheaza utilizatorul si il aduce din nou la fereastra de login. Fereastra Admin are un meniu principal, din care utilizatorul poate alege daca vrea sa manage-uiasca produsele, sa creeze un produs compus nou, sa genereze rapoarte sau sa se delogheze. Daca utilizatorul apasa pe butonul Manage, se va deschide un meniu nou din care utilizatorul poate selecta daca vrea sa importe produse, sa adauge, sa modifice sau sa stearga produse. Butonul Report va deschide un nou panel din care utilizatorul poate selecta ce raport vrea sa genereze si, daca va apasa butonul de generate, va aparea un nou raport sub forma de fisier text in folder. Butonul Meniu va deschide un nou panel in care utilizatorul poate crea un meniu nou, selectand produse prin combobox.

Pachetul controller contine controllerele interfetelor grafice, toate implementand interfata Controller in care este suprascrisa metoda de close a frameurilor. Interfata LoginController are metoda de check input in care verifica daca userul este administratorul sau este client, caz in care verifica in lista de clienti daca exista sau nu clientul respectiv si daca parola este corecta sau nu. In Signup controller, metoda checkInput verifica daca campurile in care trebuiau incarcate datele noului utilizator sunt goale.

# Concluzii si Dezvoltari Ulterioare

Programul reușește să realizeze toate funcționalitățile dorite, anume să insereze, să updateze, să șteargă sau să afișeze clienții, produsele sau comenzile, în funcție de dorințele utilizatorului, de asemenea reușind și să genereze chitanța pentru fiecare nouă comandă plasată și să o salveze local sub forma de text file.

Dezvoltarile pe care le vad facute ulterior sunt, in primul rand, posibilitatea de a sterge produse din cos daca utilizatorul se razgandeste, posibilitatea de a sterge produse din lista pentru crearea produsului compus, ascunderea parolei la logare si inscriere, posibilitatea cautarii dupa mai multe criterii, crearea unor validatori pentru campurile de price, fat, proteins, sodium si calories, astfel ca sa nu existe sansa sa se introduca si alte caractere in afara de cele numerice.

În afară de aceste aspecte tehnice, prin acest proiect am reușit să stăpânesc mai bine limbajul Java, serializarea si folosirea hashmap-urilor.

# Bibliografie

<http://javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.html>

<https://howtodoinjava.com/java8/java-stream-distinct-examples>

<http://www.tutorialspoint.com/java/java_serialization.htm>