



# DELIVERY MANAGEMENT SYSTEM

-Documentație-

**Student:**  
Pop Ruxandra Maria

Univeristatea Tehnică din Cluj-Napoca  
Facultatea de Automatică și Calculatoare  
Secția Calculatoare și Tehnologia Informației  
Anul 2, Grupa 30226

## **Cuprins**

1.Obiectivul temei .....	3
2.Analiza problemei .....	4
3.Proiectare .....	9
4.Implementare .....	17
5.Rezultate .....	21
6.Concluzii .....	22
7.Bibliografie .....	22

# 1. Obiectivul temei

## *1.1. Obiectiv principal*

Obiectivul principal al acestei teme a fost realizarea unei aplicații desktop cu interfață grafică pentru gestionarea unui restaurant care oferă posibilitatea livrării comenzi la domiciliu. Aplicația permite logarea a 3 tipuri de utilizatori, în funcție de rolul pe care îl indeplinesc: client, administrator, angajat. Fiecare utilizator poate efectua diferite operații, în funcție de rolul lui. Astfel administratorul poate importa produsele, să gestioneze produsele (să steargă, să adauge, să editeze), să genereze rapoarte. Clientul poate să vizualizeze meniul, să caute un produs în meniu pe baza unui criteriu/criterii, să plaseze o comandă, să solicite bonul. Angajatul primește o notificare de fiecare dată când este plasată o comandă de către client. Aplicația respectă paradigmele programării orientate pe obiect, și totodată respectă Design Pattern-ul pentru implementarea claselor MenuItem, BaseProduct și CompositeProduct.

## *1.2. Obiective secundare*

**Reprezintă pasii care trebuie urmați pentru atingerea obiectivului principal.**

Obiectiv Secundar	Descriere	Capitol
<b>Impărțirea pe clase</b>	Pentru împărțirea pe clase am respectat design-ul impus de temă	3
<b>Generare JavaDoc</b>	A fost necesar să generez javaDoc-ul pentru interfață IDeliveryServiceProcessing, care să cuprindă precondiții, postcondiții, invariant	4
<b>Lucrul cu fisierele</b>	Pentru realizarea acestei teme, a fost necesar să generez fisiere txt, care să cuprindă atât rapoartele, cât și bill-ul comenzi.	3
<b>Implementarea soluției</b>	Am folosit parada OOP, cu multiple clase și metode, care vor fi prezentate mai jos, care respectă modelul arhitectural.	4

Obiectiv Secundar	Descriere	Capitol
<b>Serializarea/deserializarea</b>	A trebuit să folosesc serializarea pentru salvarea informațiilor referitoare la aplicație.După care am folosit deserializarea pentru încărcarea acestor date la o nouă rulare.	4
<b>Folosire HashMap</b>	În realizarea acestei teme a fost eficientă utilizarea hashmap-ului ,deoarece complexitatea operațiilor de căutare este de $O(1)-O(n)$ (worst case).	3

## 2.Analiza problemei

În zilele noastre,nevoia de dezvoltare a tehnologiei este în creștere datorită apariției noului virus,iar restaurantele trebuie să se adapteze noilor restricții ,astfel este nevoie de o gestionare mai bună a sistemului de delivery.Această aplicație oferă suportul necesar atât staff-ului (administratorul și angajații),deoarece reprezintă o modalitate mai usoară de comunicare între ei,cât și clienților care vor să consulte meniul și să plaseze comenzi direct de pe telefon,fără să fie nevoie să se deplaseze.

Aplicația trebuie să indeplinească toate cerințele ,astfel încât angajații firmei să primească imediat detaliile despre comanda plasată ,și să o pregratească în cel mai scurt timp posibil.Totodată aplicația trebuie să usureze și munca administratorului,generând anumite rapoarte ,doar prin apasarea unui buton.

Pentru ca aplicația să funcționeze corect,trebuie să se respecte urmatoarele condiții:

-user-ul și parola să coincidă cu cele din fisierele .txt ,specifice pentru fiecare tip de utilizator

-câmpurile utilizate în efectuarea unei cerințe trebuie să fie completeate corespunzător

-prețul unui produs trebuie să fie mai mare ca 0



# Cerințe de funcționare

**Identificarea cerințelor de funcționare constituie un element critic în dezvoltarea unui sistem software.**

Înainte să ne apucăm de proiectarea și implementarea propriu zisă a sistemului , este necesar să cunoaștem ce cerințe trebuie acesta să indeplinească ,pentru a descoperii ce trebuie să facă sistemul ,cum trebuie să funcționeze ,pentru a stii ce obiective dorim să atingem la sfârșitul implementarii.

În continuare voi prezenta ,sub forma unui tabel ,cerințele sistemului și constrângările,pe care le-am identificat în urma analizei problemei.

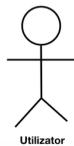
	<b>Descriere</b>
<b>Cerințe funcționale</b>	<ul style="list-style-type: none"><li>-Aplicația trebuie să permită, logarea cu user și parolă a administratorului ,a clientului,a angajatului.</li><li>-Aplicația trebuie să permită administratorului ,să adauge ,să steargă,să modifice elemente din meniu,să adauge la produsele compuse,produse de baza care se găsesc în meniu .De asemenea poate să genereze diferite rapoarte referitoare la comenzi,și tot odată să importe produse dintr-ul fisier .csv.</li><li>-Aplicația trebuie să permită clientului ,să vizualizeze meniu,să caute produse din meniu pe baza unui criteriu/criterii.Să creeze o comandă care conține diferite produse din meniu,și să genereze factură comenzi.</li><li>-Aplicația trebuie să trimită angajatului o notificare de fiecare dată când este creată o nouă comandă,urmând ca acesta să o pregătească</li></ul>
<b>Cerințe non-funcționale</b>	<ul style="list-style-type: none"><li>-Aplicația trebuie să fie intuitiva și ușor de folosit</li><li>-Aplicația trebuie să afiseze rezultatul într-un mod cât mai interactiv și ușor de citit și înțeles</li><li>-Aplicația trebuie să permită logarea tipului de utilizator dorit</li></ul>
<b>Constrângerile</b>	<ul style="list-style-type: none"><li>-Aplicația nu permite logarea clientului și administratorului simultan</li><li>-Prețul comenzi trebuie să fie mai mare ca 0</li></ul>

## **Diagrama de use case**

**Diagrama de use case prezintă o colecție de cazuri de utilizare și actori.**

**În cazul acestei teme :**

- **actorul este utilizatorul/clientul/administratorul/angajarul**
- **cazurile de utilizare :**



Utilizator

**În cazul administratorului:**

**● În caz de succes:**

- User-ul și parola sunt corecte și preluate din fisierul manager.txt,și se selectează login as admin.
- Crează un nou produs de tip Base sau Composite și apasă butonul Create.
- Adaugă elemente din meniu ,la un produs compus după care apasă butonul Add item to component.
- Selectează produsul pe care dorește să-l modifice,iar în funcție de tipul produsului poate modifica anumite câmpuri, sau poate să steargă produsul selectat prin apăsarea butonului Delete.
- Selecteaza produsul compus pe care il dorește,după care alege produsul pe care vrea să-l elime din compoziția acestuia.
- Introduce datele dorite în câmpuri ,după care apasă pe buton,în funcție de ce raport vrea să genereze.

**● În caz de nesucces:**

- User-ul si parola nu sunt corecte,nu se găsesc in fisierul manager.txt.
- Nu completează toate câmpurile prezente.
- Nu selectează produsul compus, sau produsul pe care vrea să-l adauge în compoziția produsului compus.
- Nu selectează produsul pe care dorește să-l modifice
- Nu introduce date în toate câmpurile necesare pentru generarea raportului.
- În toate situațiile se vor genera erorile/avertizările necesare prin interfața grafică

## În cazul clientului:

### ● În caz de succes:

- User-ul și parola sunt corecte și preluate din fisierul client.txt, nu se selectează login as admin.
- Apasă pe butonul Show menu, și astfel are posibilitatea să vadă produsele din meniu. Acestea sunt separate prin 2 criterii: Base Product și Composite Product. Prin butoanele care se găsesc în partea de jos clientul poate alege ce produse dorește să vizualizeze.
- Apasă pe butonul Search menu, și introduce în câmpurile disponibile criteriu/criteriile pe baza căruia vrea să caute un produs în meniu, după care apasă pe Search și în partea din dreapta îi va apărea o listă cu produsele căutate.
- Apasă butonul Order management. În acest moment poate să creeze o nouă comandă sau să genereze bill-ul pentru o comandă existentă. Dacă dorește să genereze factură, trebuie să selecteze o comandă existentă în tabelul de mai sus, după care să apeleze pe butonul generate bill. Dacă dorește să creeze o nouă comandă, apasă pe butonul New, și se va deschide o nouă panou, unde în partea din dreapta poate să selecteze produsul pe care dorește să-l adauge în comandă, după care apasă pe Add și astfel noua comandă va conține produsul selectat.

### ● În caz de nesucces:

- User-ul și parola nu sunt corecte, nu se găsesc în fisierul client.txt.
- Pentru a vedea produsele importate din fisierul products.csv, prima dată trebuie să se logheze administratorul și apoi clientul.
- Nu introduce nici o valoare, în nici un câmp disponibil
- Încearcă să genereze un bill, fară să selecteze comandă
- În toate situațiile se vor genera erorile/avertizările necesare prin interfața grafică

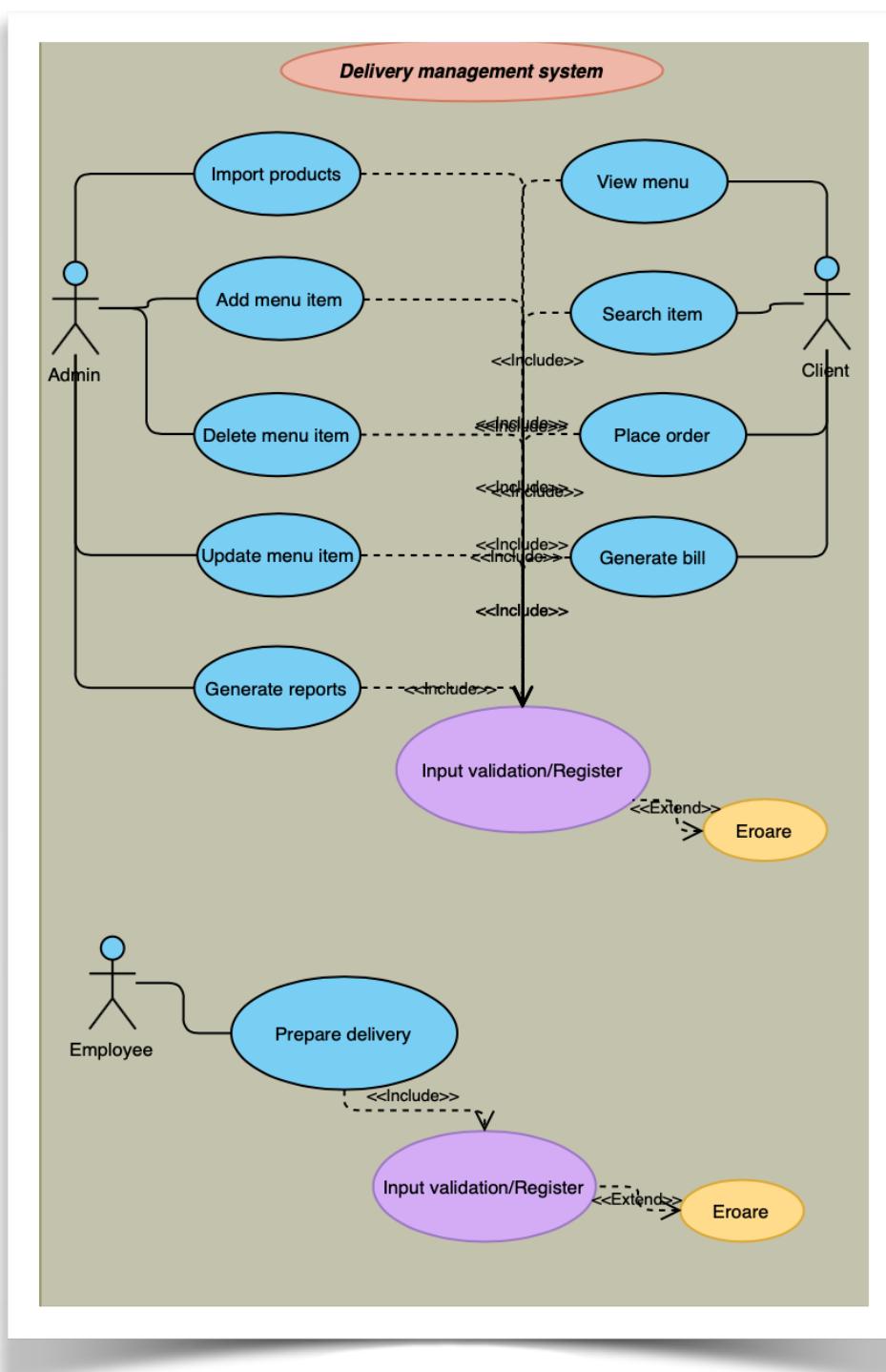
## În cazul angajatului:

### • În caz de succes:

- User-ul și parola sunt corecte și preluate din fisierul employee.txt.
- Angajatul primește notificări atunci când se crează o nouă comandă.

### • În caz de nesucces:

- User-ul și parola nu sunt corecte, nu se găsesc în fisierul employee.txt.
- Angajatul nu se logă



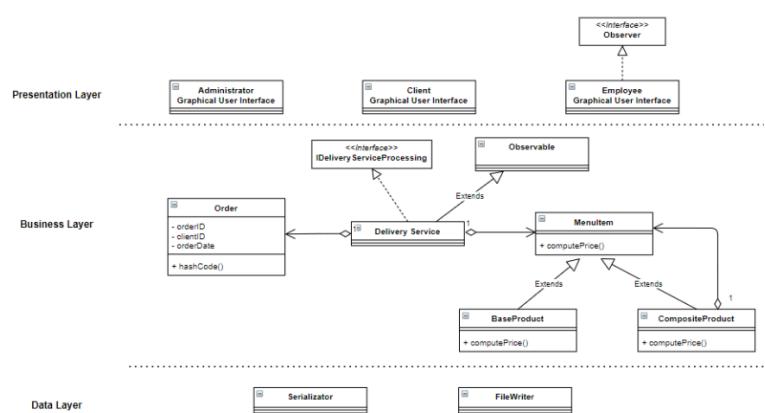
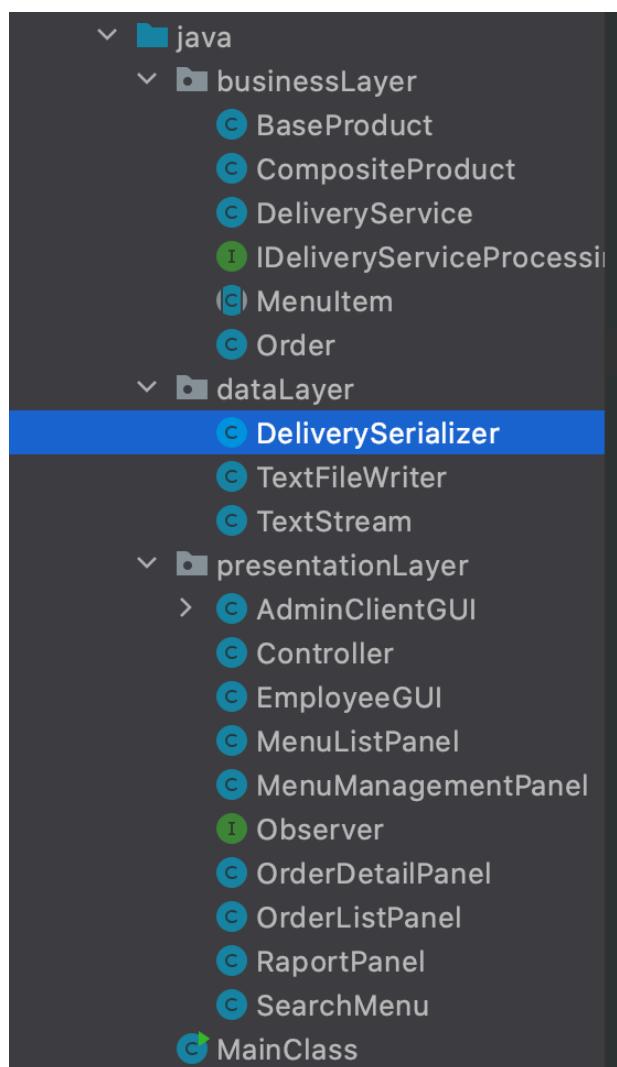
# 3. Proiectarea

## 3.1. Etapa de design

Proiectarea claselor a fost facută ca respectând modelul pe nivele prezentat în prezentarea suport a temei. Astfel fiecare nivel ,are un rol prestabilit, și am implementat toate nivele ,fără să evit vre-unul.

Am structurat aplicația conform modelului arhitectural Layers, ce presupune împărțirea responsabilităților pe mai multe nivele, lucru care facilitează înțelegerea și dezvoltarea ulterioară a aplicației.

Am împărțit aplicația în 3 pachete după cum se poate observa :

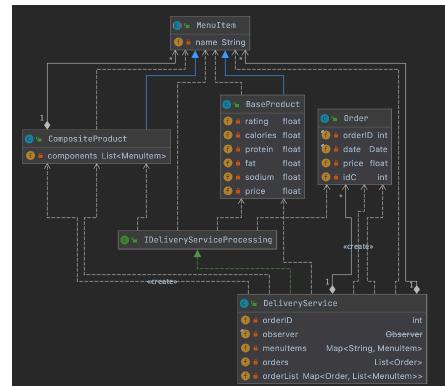


Încă de acum se poate observa că am respectat structura impusă de laborator,cu mici modificări și adăugări de noi clase, care să îmi usureze implementarea aplicației.  
În următorul capitol se vor prezenta detaliat fiecare clasă și fiecare metodă din ea, astfel se va putea observa mai detaliat respectarea design-ului impus.

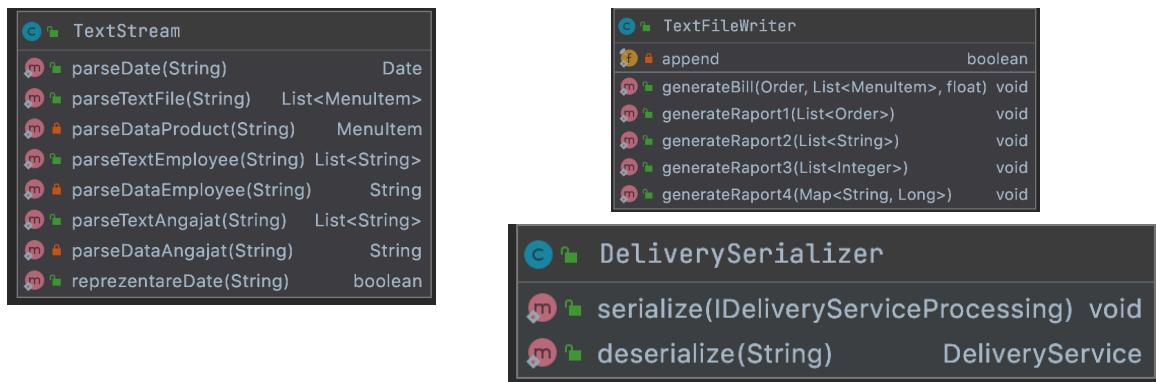
În continuare voi prezenta fiecare pachet pe rând:

**1.Pachetul businessLayer**-Conține clasele ce descriu modelul de date al aplicației.Precum conține și o interfață care definește niste metode care urmează să fie implementate în clasă DeliveryService.

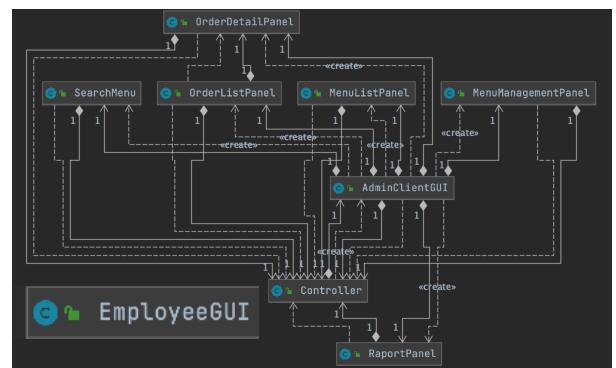
Conține întreaga logică a aplicației.



**2.Pachetul dataLayer**-Conține clase destinate lucrului cu fișiere text,de generare a rapoartelor ,a facturii unei comenzi.Precum conține și clase care au rol în serializare respectiv deserializare.Astfel prin serializare va genera un fisier.txt care conține toate datele referitoare la delivery service ,iar prin deserializare va citi și interpreta datele referitoare la delivery service ,date citite dintr-un fisier dat de tipul txt.Prezintă și o clasa care are rolul de a citi produsele importate de administrator dintr-un fisier .csv prin streams și tot prin această tehnică de streams și split ,citește datele de logare a utilizatorilor din fisierele corespunzătoare fiecarui tip de utilizator.

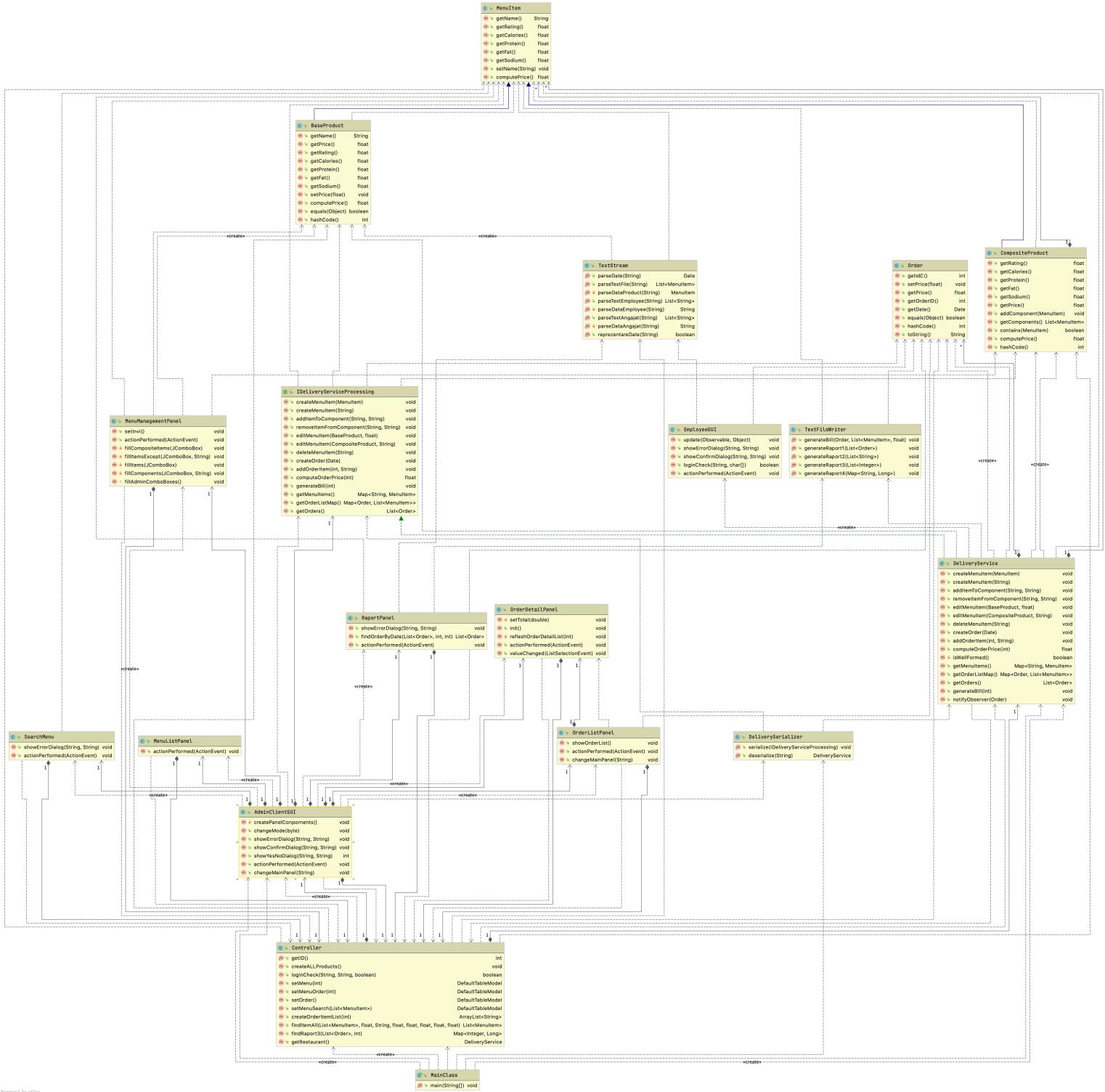


**3.Pachetul presentationLayer**\_ - aici se întâlnesc clasele destinate interacțiunii cu utilizatorul .Conține mai multe clase , dintre care una este Controller, care are rolul de trasmite datele, primite prin interfață grafică, modelului.Celalalte clase sunt destinate modelării interfeței grafice ,care va fi prezentată mai detaliat în următorul subcapitol.



# UML-Unified Modeling Language.

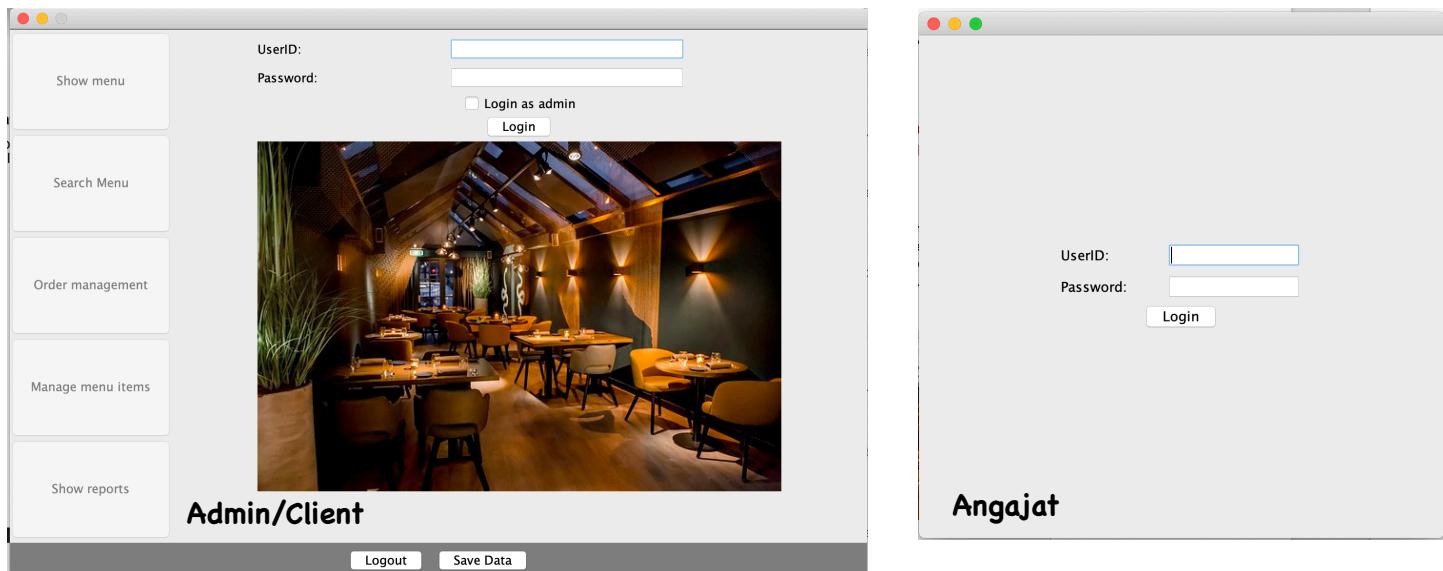
**Reprezintă un set de clase, interfețe, colaborări și alte relații.**



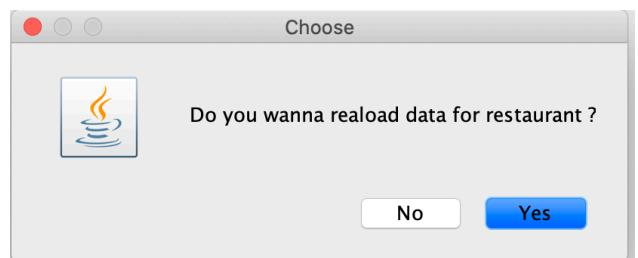
### 3.3. Interfață grafică (Graphical User Interface)

Reprezintă un mecanism prietenos pentru interacțiunea utilizatorului cu programul.

Folosim o interfață User-Friendly pentru a permite utilizatorilor să se simtă mai familiarizați cu programul chiar înainte de a-l fi utilizat. Printr-o interfață utilizatorul poate înțelege mai bine cum funcționează programul, poate învăța mai repede modul de utilizare a acestui calculator. Aceasta interfață cuprinde 2 frame-uri(windows), unul principal pentru employee, și unul pentru client/admin.



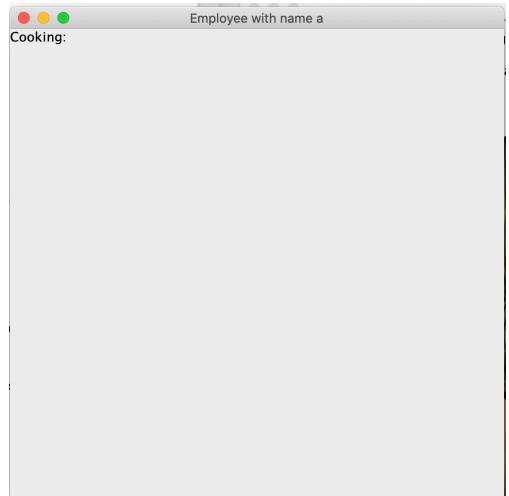
Înainte să se deschidă interfața grafică, se apelează un JOptionPane care ne întrebă dacă vrem să reluăm datele salvate în urma serializării. Dacă se alege yes se vor importa datele dintr-un fisier delivery.txt prin deserializarea, dacă se alege no se formează un nou restaurant.



## Frame-ul employee

Conține :

- 2 JLabel-uri și totodată 2 JTextField-uri ,pentru UserID și pentru Password
- Un buton de Login
- După logare se deschide o nouă fereastră unde se vor afisa în timp real comenziile plasate ,prin intermediul Observable care așteaptă să fie procesate



## Frame-ul admin/client

Prezintă un panou principal care conține 6 panouri secundare :

### ★MenuListPanel -accesat prin butonul Show Menu

	Title	Rating	Calories	Protein	Fat	Sodium	Price
Show menu	Croq	0.0	172.0	1.0	0.0	3.0	28.0
	Chicken Fricas...	2.5	678.0	60.0	39.0	936.0	20.0
	Baked Apples ...	0.0	386.0	3.0	11.0	21.0	50.0
	No-Bake Choc...	3.75	520.0	5.0	34.0	53.0	34.0
	Boulevardier	0.0	231.0	0.0	0.0	2.0	95.0
	Pork Fricassee ...	3.75	727.0	29.0	28.0	374.0	63.0
	Soy & Peanut ...	3.75	119.0	19.0	37.0	201.0	80.0
	Peanut Butter	3.75	320.0	9.0	25.0	146.0	94.0
Search Menu	Puerto Rican B...	3.125	502.0	33.0	13.0	645.0	28.0
	Thai Spices Sh...	3.75	644.0	8.0	28.0	953.0	67.0
	Alternative Oat...	4.375	107.0	1.0	5.0	48.0	51.0
	Chocolate Fud...	0.0	256.0	8.0	18.0	60.0	86.0
	Lemon Caramell...	0.0	229.0	6.0	10.0	68.0	98.0
	Surfer's Granola	2.5	301.0	8.0	16.0	157.0	24.0
	Braised Shortri...	3.75	1347.0	183.0	70.0	500.0	72.0
Order management	Triple-Chocola...	4.375	156.0	2.0	8.0	51.0	31.0
	Roast Turkey ...	4.375	1859.0	58.0	163.0	854.0	35.0
	Pear Cake with...	4.375	839.0	59.0	16.0	932.0	11.0
	Apple Pancake...	4.375	146.0	2.0	9.0	106.0	50.0
	Marbled Sweet...	2.5	672.0	6.0	26.0	444.0	33.0
	Croque-Monsi...	0.0	794.0	63.0	40.0	534.0	58.0
	Chicken Satay ...	4.375	1012.0	86.0	37.0	1994.0	31.0
Manage menu items	Cheese Steak...	4.375	261.0	9.0	11.0	240.0	43.0
	Melissa Hottek...	4.375	337.0	10.0	17.0	7.0	45.0
	Thai Red Curry...	5.0	125.0	5.0	1.0	625.0	99.0
	Almond Cookies	3.75	98.0	3.0	7.0	7.0	38.0
	Roasted Guine...	4.375	1193.0	95.0	65.0	853.0	96.0
	Chinese Caboba...	3.125	62.0	3.0	0.0	737.0	71.0
Generate reports	Apricot and Pis...	3.75	239.0	4.0	6.0	1442.0	1215.0
	Port-Currant S...	0.0	1215.0	18.0	4.0	5302.0	17.0

Sunt prezente și 3 butoane :-All

-Base products

-Composite products

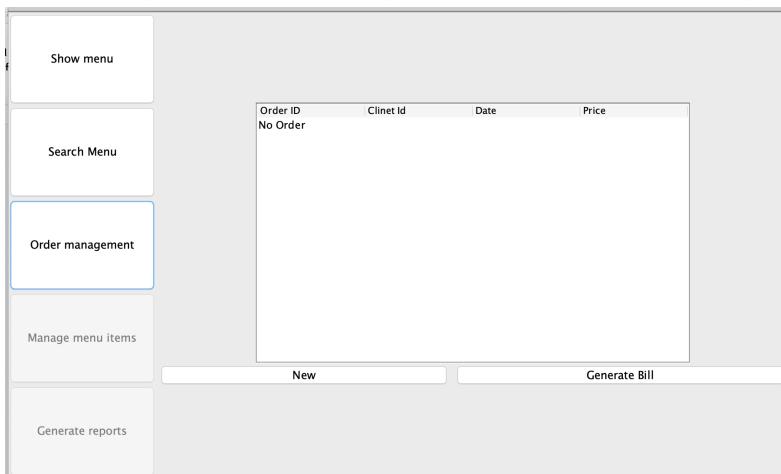
ACEsteau rolul de al lăsa pe client/admin să clasifice produsele în funcție de categoria dorită.

### ★SearchMenu- accesat prin butonul Search menu

Show menu	Menu list	
Search Menu	<input style="width: 100px; height: 20px; margin-right: 10px;" type="text" value="Title: "/> <input style="width: 100px; height: 20px; margin-right: 10px;" type="text" value="Rating: "/> <input style="width: 100px; height: 20px; margin-right: 10px;" type="text" value="Calories: "/> <input style="width: 100px; height: 20px; margin-right: 10px;" type="text" value="Protein: "/> <input style="width: 100px; height: 20px; margin-right: 10px;" type="text" value="Fat: "/> <input style="width: 100px; height: 20px; margin-right: 10px;" type="text" value="Sodium: "/> <input style="width: 100px; height: 20px; margin-right: 10px;" type="text" value="Price: "/>	
Order management	<input style="width: 100px; height: 20px;" type="button" value="Search"/>	
Manage menu items		
Generate reports		

Se găsesc numeroase campuri,unde clientul introduce valorile după care vrea să filtreze un produs.  
După ce a introdus valorile dorite apăsa butonul Search,iar în tabelul din partea dreaptă se vor afisa produsele găsite.

## ★OrderListPanel - accesat prin butonul Order management

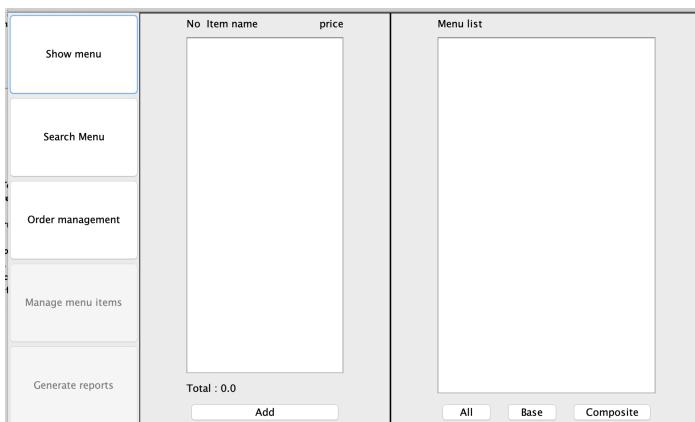


Se găseste un JTable unde vor aparea comenziile plasate.

Două butoane :

- New care va forma o nouă comandă și care va deschide subpanoul pentru adăugarea de produse la comandă.
- Generate Bill va genera o factură de tip .txt care va conține datele referitoare la comanda selectată

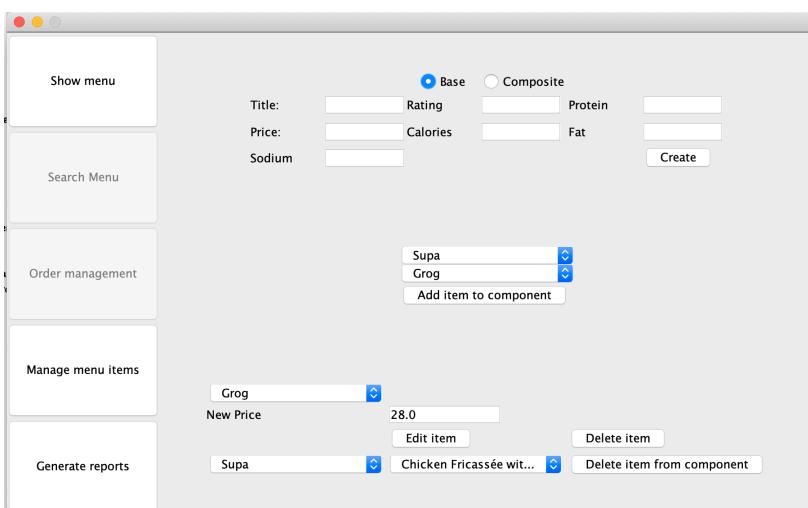
-conține un subpanou OrderDetail Panel



In partea din dreapă se gaseste un JTable unde vor fi filtrate produsele din meniu prin apasarea uneia din cele 3 butoane prezente.

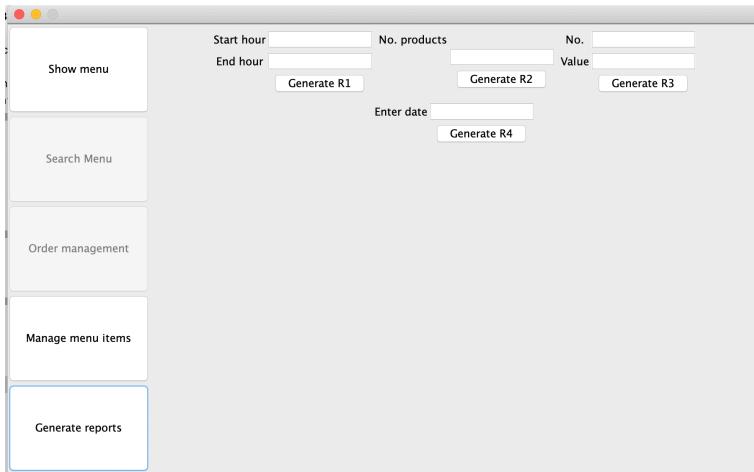
Iar în partea din stângă se vor adăuga ,produsele selectate din Jtable ,prin apasarea butonului Add .Produsele vor fi adăugate la comanda curentă.

## ★MenuManagementPanel – accesat prin butonul Manage menu items



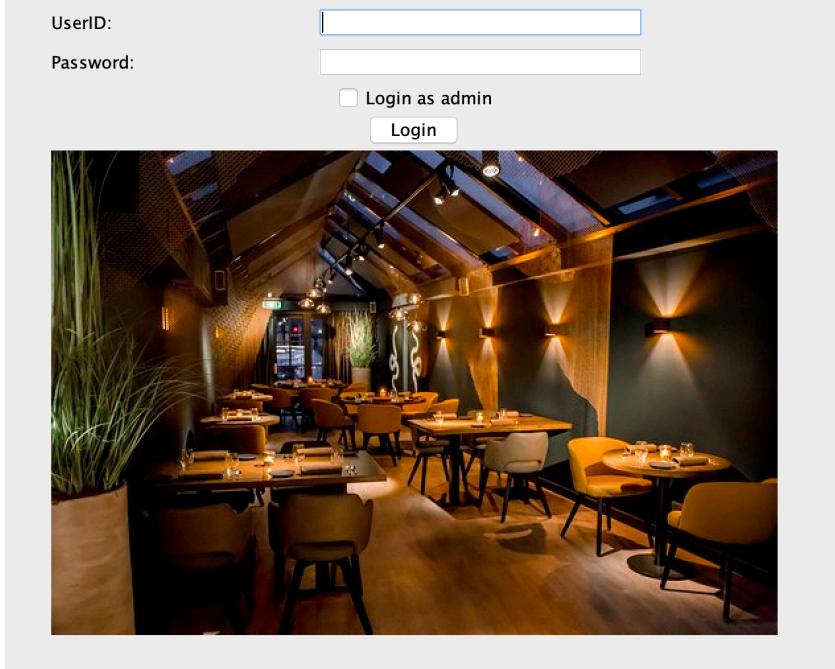
Prezintă numeroase JTextField-uri,JTextLabel,JComboBox ,JButton , care să ajute la adăugarea unui produs base/composite,la adăugarea intr-un produs composite a unui produs de tip base ,la editarea unui produs ,la stergerea unui produs, precum la stergerea unui produs dintr-o componentă.

## ★RaportPanel-accesat prin butonul Generate reports



Prezintă numeroase JTextField-uri, JLabel, JButton , care să ajute la generarea unor rapoarte. Admin-ul completează campurile necesare raportului pe care vrea să-l genereze ,iar după completarea lor apasă butonul corespunzător.

La începutul aplicației se deschide panoul de Login.



Prezintă un JCheckBox pentru a selecta dacă se loghează un admin. 2Jlabel-uri și totodată 2 JTextField-uri ,pentru UserID și password

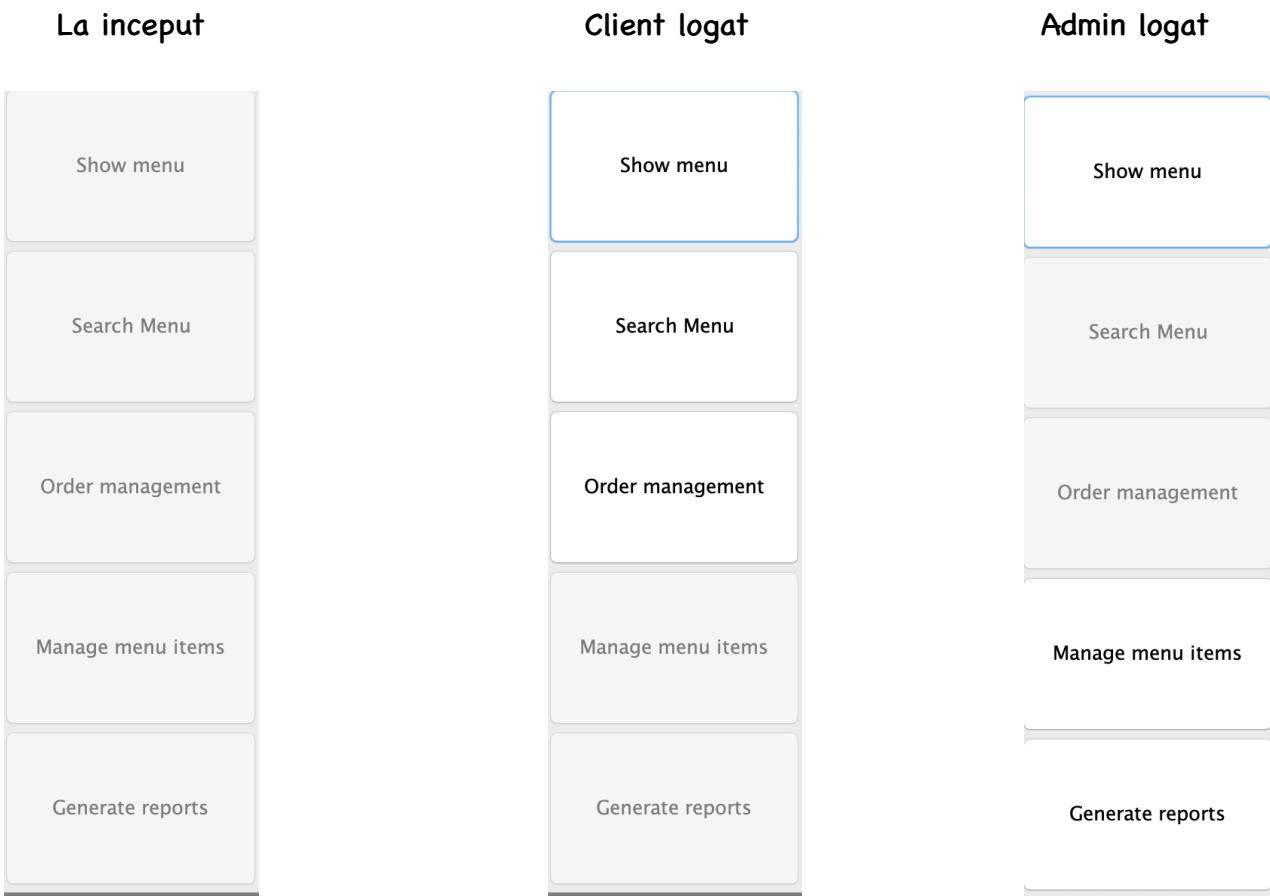
Pe tot parcursul aplicației sunt prezente aceste două butoane.

Dacă se apasă butonul Logout se deconectează utilizatorul curent, și se redeschide panoul de login.

Dacă se apasă butonul Save Data se salvează datele restaurantului într-un fisier.



Cele 5 butoane de pe margine care fac trecerea de la un panou la altul sunt disponibile în funcție de stadiul în care se află aplicația.



### 3.4. Structuri de date

Ca și structuri de date s-au folosit `HashMap`-urile pentru reținerea produselor din `DeliveryService`, și pentru reținea comenzi, deoarece funcția lor de căutare prezintă o complexitate foarte bună ( $O(1)$ ) în caz favorabil și  $O(n)$  în caz nefavorabil), iar la realizarea acestei aplicații ne bazăm foarte mult pe căutarea rapidă și eficientă a produselor. Totodată am ales folosirea `HashMap`-urile deoarece am considerat ca nu pot există două obiecte cu același nume.  
S-au mai folosit și `List` pentru reținerea diferitelor obiecte pe parcursul dezvoltării aplicației.

```
public DeliveryService() {
    menuItems = new HashMap<>();
    orders = new ArrayList<>();
    orderList = new HashMap<>();
    this.observer = new EmployeeGUI();
    assert this.observer != null;
}
```

# 4.Implementarea

## 1.MenuItem

Este o clasă abstractă ,care conține scheletul pentru clasele

BaseProduct și CompositeProduct. Conține numeroase metode care urmează să fie implementate de cele două clase care o mostenesc.

Aici se găsesc metodele de get pentru fiecare atribut.Prezintă un singur constructor, cu un singur atribut (name) de tip String care este necesar pentru ambele clase copii.Totodată aceasta clasa implementează interfața Serializable.

```
public abstract class MenuItem implements Serializable {
    private String name;
    public MenuItem(String name) {
        this.name = name;
    }
}
```

## 2.BaseProduct

Această clasă extinde clasa MenuItem, implementând astfel fiecare metodă abstractă întâlnită în clasa părinte. Este clasa ce modelează produsele de bază.Un astfel de produs prezintă numeroase atribute după cum se poate observa din structura constructorului.Name-ul este mostenit de la clasa super.Am suprascris metoda equals ,pentru a ajuta la compararea a două obiecte,aici mă folosesc și de metoda hashCode.Metoda aceasta este folosită în clasa CompositeProduct pentru a vedea dacă un produs compus conține un anumit produs de bază.

```
public BaseProduct(String name, float rating, int calories, float protein, float fat, float sodium, float price) {
    super(name);
    this.rating = rating;
    this.calories = calories;
    this.protein = protein;
    this.fat = fat;
    this.sodium = sodium;
    this.price = price;
}
```

## 2.CompositeProduct

Această clasă extinde clasa MenuItem, implementând astfel fiecare metodă abstractă întâlnită în clasa părinte. Este clasa ce modelează produsele de compuse întâlnite în restaurant.Un astfel de produs este descris de numele său mostenit de la clasa parinte prin apelul super(name) și de o lista de produse de bază pe care le conține.Conține o metoda de adăugare a produselor de bază în listă produsului compus.Prezintă și o metoda contains care verifică dacă un anumit item se află în produsul compus.Metoda suprascrisă din clasa părinte computePrice calculează pretul total al produsului compus ,în funcție de elementele pe care le conține

```
public class CompositeProduct extends MenuItem {
    private List<BaseProduct> items;
    public CompositeProduct(String name) {
        super(name);
        items = new ArrayList<>();
    }
}
```

## 4.Order

Acestă clasă modelează comenziile pe care le va plasa clientul prin intermediul interfeței grafice. După cum se poate observa, clasa implementează interfața Serializable, necesară la serializarea / deserializarea datelor. O comandă este caracterizată de orderId, de data la care a fost plasată, idC - id-ul clientului care a plasat comanda, și prețul final al comenzi. Sunt întâlnite multiple metode de getter(getIdC, getPrice..) care vor returna valoarea anumitor attribute, și o metodă de setPrice. Am suprascris și aici metoda equals pentru a vedea dacă două comenzi sunt egale. Este suprascrisă și metoda toString care ne va ajuta la scrierea mai frumoasă a comenzi, la generarea facturi de către client.

```
public Order(int orderId, int idC, Date date, float price)
    this.idC = idC;
    this.orderId = orderId;
    this.date = date;
    this.price = price;
}
```

## 5.IDeliveryServiceProcessing

Este interfața prin care exteriorul poate interacționa cu obiectul DeliveryService. Definește toate metodele care se pot efectua de către un administrator sau client asupra aplicației de delivery. Pentru această clasă să genereze și JavaDoc-ul, astfel se pot observa anotații pentru custom tags, @param, @post, @invariant, @throws, @return și o scurtă descriere pentru fiecare metoda.

## 6.DeliveryService

Este clasa ce modelează întreaga structură a aplicației de delivery. Conține toate datele necesare pentru a modela o firmă de catering, un restaurant. Deci această clasa, este clasa principală a programului. Conține o lista de comenzi, fiind de tip ArrayList, și care conține toate comenziile plasate de către clienți, un HashMap care are drept key un String, iar valoarea este un obiect de tipul MenuItem (poate fi BaseProduct sau CompositeProduct), am ales să folosesc acestă colecție deoarece am considerat că nu pot exista două produse cu același nume. OrderList este tot o colecție de tip HashMap, care conține o lista de produse specifice pentru o anumită comandă, am presupus că nu poate exista aceiași comandă de două ori.

```
public DeliveryService() {
    menuItems = new HashMap<>();
    orders = new ArrayList<>();
    orderList = new HashMap<>();
    this.observer = new EmployeeGUI();
    assert this.observer != null;
}
```

Totodată această clasă implementează interfața descrisă mai sus ,deci aici se vor suprascrie fiecare metodă întărită în interfață . Aceste metode au fost implementate cu ajutorul instrucțiuni de assert.Precum mosteneste și clasa Observable ,implementând metoda notifyObserver care are ca scop să anunțe ,în timp real ,angajatul că s-a plasat o comandă nouă.

!Pentru mai multe detali referitoare la ce face fiecare metodă ,se recomandă citirea JavaDoc-ului, prezent in folder-ul javadoc.



javadoc

## 7.DeliverySerializer

Aplicația se bazează pe tehnica de serialization.Această clasă implementând cele două metode necesare pentru această tehnică, metoda de serializare ,respectiv deserializare. Metoda serialize() - are ca scop salvarea unei instanțe de tip DeliveryService, deci salvează toate datele conținute de aplicație ,intr-un fisier denumit "delivery.txt". Metoda deserialize()- are ca scop returnarea unei instanțe de tip DeliveryService,prin deserializarea conținutului fisierului dat ca argument.

```
public static void serialize(IDeliveryServiceProcessing app) {  
    try {  
        FileOutputStream file = new FileOutputStream( name: "del"  
        ObjectOutputStream out = new ObjectOutputStream(file);  
        out.writeObject(app);  
        out.close();  
        file.close();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}  
  
public static DeliveryService deserialize(String path) {  
  
    try {  
        FileInputStream file = new FileInputStream(path);  
        ObjectInputStream in = new ObjectInputStream(file);  
        return (DeliveryService) in.readObject();  
    } catch (IOException | ClassNotFoundException e) {  
        e.printStackTrace();  
    }  
    return null;  
}
```

## 8.TextFileWriter

Această clasă implementează diferite metode care au ca scop crearea/scrierea intr-un fisier de tip .txt.Aici se găsesc metodele pentru crearea rapoartelor ,și metoda pentru generarea facturi unei anumite comenzi transmisă ca argument.Se scrie în fisier prin intermediul PrintWriter.printf().

```
FileWriter fileWriter = new FileWriter(bill.getCanonical  
PrintWriter printWriter = new PrintWriter(fileWriter);  
printWriter.printf("%s%n", order.toString());
```

## 9.TextStream

Această clasă implementează tehnica de stream and split.Aștfel toate metodele prezente aici au ca scop citirea unor date dintr-un fisier .txt sau .csv și parsarea lor sub forma unei colecții de tip List.Pentru a reuși să citeșc produsele din fisierul products.csv a trebuit să împart fiecare linie din fisier în 7 părți separate prin regex-ul „,” după care am creat produsul și l-am adăugat într-o lista. La fel am facut și în cazul utilizatorilor pentru logarea cu user și parolă.

```
String[] parts = data.split( regex: ",");  
float c, f, p, s, pri, r;  
String title = parts[0].trim();  
String calories = parts[2].trim();  
String rating = parts[1].trim();  
String protein = parts[3].trim();  
String fat = parts[4].trim();  
String sodium = parts[5].trim();  
String price = parts[6].trim();
```

## 10.Pachetul presentationLayer

Intregul pachet prezintă același scop și anume crearea interfeței grafice prezentate în capitolul ulterior.Nu am să descriu fiecare clasă detaliat ,decât cele care se diferențiază de restul prin metodele implementate/prin scopul îndeplinit.

- **Clasa Controller**

Este clasa care face legătura între utilizator și interfața grafică.Aici se instantiază obiectul de tip AdminClientGUI,precum și instanța de DeliveryService.Prezintă metodele pentru crearea tabelelor întâlnite în interfață AdminClientGUI,precum și metoda de search a unui produs care are la bază implementarea cu lambda expressions și streams.Metoda de adaugare a produselor importate de către admin din fisierul products.csv.Prezintă și o metoda care verifică dacă user-ul și parola scrise în interfață grafică sunt corecte

```
public Controller(DeliveryService restaurant) {  
    this.restaurant = restaurant;  
    view = new AdminClientGUI(restaurant, controller: this);  
}
```

- **Clasa EmployeeGUI**

Implementează interfața grafică a angajatului.După cum se poate observa această clasă implementează interfața Observer,întrucât angajatul trebuie să stie când a fost plasată o comandă pentru a o putea pregăti.Se întâlnește metoda update în acest scop.

Restul claselor prezente aici, au fost prezentate în capitolul ulterior(Cel cu interfața grafică),deci nu mai are rost dezbaterea lor încă o dată deoarece nu conțin metode speciale/specifice.

## 5.Rezultate

Pentru a demonstra corectitudinea programului voi genera un bill pentru o anumită comandă plasată prin intermediul interfeței grafice. Aplicația este foarte usor de utilizat și de înțeles ,nu ar trebui să întâmpinam dificultăți la utilizarea ei.

Aceasta este comanda plasată de către client.Se pot observa produsele pe care le conține ,și pretul total al comenzi.

The screenshot shows a software interface with two main sections. On the left, a 'Cart' window displays a list of items with their names and prices:

No	Item name	price
1	Melissa Hotek's Granola	43,00
2	Roast Turkey with Corn Bread Stuffing and Giblet Gravy	35,00

Total : 78.0

Below the cart is an 'Add' button.

On the right, a 'Menu list' window shows a list of food items with their titles and prices:

Title	Price
Grog	28.0
Chicken Fricassée w...	20.0
Baked Apples with ...	50.0
No-Bake Chocolate ...	34.0
Boulevardier	95.0
Pork Fricassée with ...	63.0
Soy and Sesame Sho...	80.0
Peanut Butter Chees...	94.0
Puerto Rican Beef St...	28.0
Thai Spiced Shrimp ...	67.0
Alternative Oatmeal...	51.0
Chocolate Fudge-Al...	86.0
Lemon Caramel Cus...	98.0
Surfer's Granola	24.0
Braised Lamb Shank...	73.0
Triple-Chocolate Cr...	31.0
Roast Turkey with C...	35.0
Pear Cake with Lem...	11.0
Apple Pancakes wit...	50.0
Marbled Sweet-Pota...	33.0
Croque-Monsieur	58.0
Chicken Sauté with S...	31.0
Cheese Blintzes with...	79.0
Melissa Hotek's Gra...	43.0
Thai Red Curry Paste	45.0
Almond Cookies	99.0
Roasted Guinea He...	38.0
Chinese Cabbage S...	71.0

Below the menu list are three buttons: All, Base, and Composite. The 'Roast Turkey with C...' item is highlighted with a blue selection bar.

The screenshot shows a software interface for generating a bill. At the top, there is a table with columns: Order ID, Client Id, Date, and Price. One row is visible with values: 0, 10, Thu May 20 02:23..., 78.0.

Order ID	Client Id	Date	Price
0	10	Thu May 20 02:23...	78.0

At the bottom of the interface are two buttons: 'New' and 'Generate Bill'.

Aici se poate observa Id-ul comenzi ,precum și id-ul clientului care a efectuat comanda ,data la care a fost plasată ,și din nou prețul total.

După ce s-a apăsat butonul Generate Bill,s-a creat un fisier cu numele bill+idOrder+.txt, care prezintă urmatorul format:

- Pe prima linie sunt detaliile comenzi (id, data)
- Pe urmatoarele linii sunt afișate produsele comandate împreună cu prețul lor
- Pe ultima linie apare prețul total al comenzii

```
Order 0    date:Thu May 20 02:23:01 BST 2021
Melissa Hotek's Granola 43.0
Roast Turkey with Corn Bread Stuffing and Giblet Gravy 35.0
Total 78.0
```

În concluzie ,aplicația funcționează corect ,fără erori sau greselii.Toate erorile întâlnite de-a lungul implementării aplicației au fost tratate și rezolvate.

## 6.Concluzie

În urma realizării acestei teme, am dobândit cunoștințe cu privire la folosirea tehnici de serializare a obiectelor pentru a aduce aplicația într-o stare anterioară cunoscută.De asemenea, am aprofundat cunoștințe cu privire la citirea și scrierea din fișiere de tip txt.

Ca și dezvoltare ulterioară,partea de interfață grafică ar putea fi imbunătățită, astfel încăt să fie mai user-friendly, fisierele .txt să fie mai frumos aranjate și structurate.Si nu în ultimul rând ,aplicația să lucreze cu baze de date complexe,de unde să preia datele despre utilizatori,și nu din fisiere ca în cazul curent.

## 7.Bibliografie

1.<https://ro.wikipedia.org>

2.<https://stackoverflow.com>

