DOCUMENTATIE

TEMA 4

**SISTEM DE GESTIONARE A LIVRARII DE ALIMENTE**

NUME STUDENT: Sand Elena - Andreea

GRUPA: 30223

# CUPRINS

[1. Obiectivul temei 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297886)

[3. Proiectare 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297887)

[4. Implementare 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297888)

[5. Rezultate 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297889)

[6. Concluzii 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297890)

[7. Bibliografie 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297891)

# 1.Obiectivul temei

Obiectivul principal al proiectului este de a creea o aplicație care să pună la dispoziție utilizatorului un șistem de management al unui restaurant. Aceste sisteme se folosesc pretutindeni, fiecare site se foloseste de un asemenea sistem pentru vanzari online. Clientul se poate loga, poate vizualiza produsele si poate sa isi adauge in cosul de cumparaturi oricate produse. dupa ce s-a incheiat comanda, clientul primeste o factura pe care trebuie sa o plateasca. Toate aceste date sunt serializate, atat meniul cat si comenzile. Administratorul poate sa adauge produse, sa stearga sau modifice, sa creeze produse compuse, si sa creeze reporturi, si mai exista un employee care este notificat atunci cand se creaza o comanda pentru a o prepara.

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

**2.1 Analiza problemei**

**Cerintele functionale ale acestui proiect sunt:**

* introducerea, editarea si stergerea unui produs;
* crearea unui produs compus;
* importarea unor produse din fisierul csv;
* afisarea listei de produse;
* cautarea unui produs;
* introducerea de comenzi;
* facturarea comenzii.
* crearea rapoartelor;
* observarea creari comenzilor.

**Cerintele non-functionale ale acestui proiect sunt:**

* Realizarea a trei interfete grafice , câte una pentru fiecare tip de utilizator Interfetele au rolul de a realiza interacțiunea dintre utilizator si program, prin prelucrarea datelor introduse de catre utilizator, dar și de a prezenta rezultatul operatiilr de modificare a datelor efectuate de către utilizator.Fiecare interfata conține câte un Jbutton pentru fiecare operație pe care programul o poate realiza în funcție de interesele tipului de utilizator.
* Salvarea datelor utilizate pentru conectarea utilizatorului dar si cele referitoare la comenzile plasate pana in prezent,. Salvarea se realizeaza prin serializare.
* Afisarea intr un fisier .txt a facturilor tuturor comenzilor din fisier. Acestea cuprind id-ul comenzii, al clientului, datele fiecarui produs comandat, data la care a fost plasata comanda, cantitatea de produs comandata, dar si suma pe care trebuie sa o plateasca clientul, aceasta fiind produsul dintre numarul de produse si pretul unui produs.
* Extragerea datelor din fisierele in care au fost salvate cand aplicatia este pornita, realizata prin deserializare.
* Importarea produselor de baza prin citirea caracteristicilor acestora dintr un fisier .csv, utilizand stream si expresii lamda.
* Actualizarea dupa plasarea fiecarei comenzi a numarului de produse din stoc, prin scaderea numarului de produse comandate de catre client.
* Realizarea operatiilor de stergere, adaugare sau modificare a datelor produselor sau adaugarea unui nou produs compus din produse de baza sau din alte produse compuse, dar si afisarea tuturor produselor, in functie de comanda aleasa.
* Implementarea functiilor pentru logarea clientilor sau inregistrarea unui nou utilizator.

**2.2 Modelare:**

**--------------------------------------------------------------------------------------------------------------------------------------------**

**Use case:** crearea unei comenzi

**Actorul principal:** user-ul

**Scenariul principal de success:**

1. User-ul se logheaza in sistem
2. daca nu exista se inregistreaza, apoi se logheaza
3. User-ul adauga la comanda cate produse vrea, putand consulta meniul
4. User-ul termina comanda si primeste bonul/factura;

**Secvența alternativă:** date introduce incorect

Userul introduce în unul dintre câmpurile de date ceva ce nu este un număr, sau lasă gol un câmp. Acesta va fi atenționat cu mesajul necesar.

**Secventa alternativa:**nu exista userul, prima oara se inregistreaza apoi se logheaza

**Secventa alternativa:** nu exista clientul sau produsul introduce: se va primi un mesaj de atentionare.

**Use case:** vizualizarea produselor

**Actor principal**: clientul

**Scenariul principal de succes:**  
 1. Utilizatorul selecteaza prima fereastra (“CLIENT”),

2. Clientul introduce un nume de utilizator si o parola si apasa pe butonul “Autentificare”,  
3. Utilizatorul selecteaza optiunea de afisare a listei de produse prin apasarea primului buton.

4. Se deschide un nou panou care contine un scroll pane in care sunt afisate toate produsele si un buton pentru revenire,

5. Utilizatorul apasa pe butonul cu sageata si revine la meniul prinipal.

**Secvență alternativă**: Date de conectare invalide

Utilizatorul introduce datele pentru autentificare.

a) Datele nu corespund niciunei valori din lista de clienti.

1. Se apasa pe butonul pentru inregistrare .

2. Datele noului client sunt savare in lista de clienti din fisierul .ser.

3. Clientul se poate autentifica cu succes cu datele salvate anterior.

4. Se revine la scenariul de succes

b) Datele corespund contului unui alt utilizator.

1. Se apasa pe butonul pentru revenire (“<=”).

2. Se reintroduc datele corecte sau se parcurg pasii de la punctul a) pentru clientii noi.

3. Se revine in scenariul de succes.

--------------------------------------------------------------------------------------------------------------------------------------------

### **Use case:** cautarea unui produs

**Actorul principal:** clientul

**Scenariul principal de succes:**

1. Clientul introduce un nume de utilizator si o parola si apasa pe butonul “Log in”  
2. Utilizatorul introduce datele dupa care se face cautarea in cele sase TextField - uri

3.Utilizatorul apasa butonul “Search”

3. In scroll pane sunt afisate toate produsele care corespund datelor introduse

4. Clientul introduce datele pentru fiecare criteriu dupa care face cautarea.

5. Utilizatorul apasa butonul din dreapta criteriului selectat.

6. Rezultatele cautarii sunt afisate in interfata grafica.

7. Utilizatorul apasa pe butonul cu sageata si revine la meniul prinipal.

**Secvență alternativă**:

1. Datele introduse sunt invalide
2. Utilizatorul introduce datele pentru autentificare.
3. Datele pentru autentificare nu corespund niciunei valori din lista de clienti.
   * Se apasa pe butonul pentru inregistrare .
   * Datele noului client sunt savate in lista de clienti din fisier
   * Clientul se poate autentifica cu succes cu datele salvate anterior.
   * Se revine la scenariul de succes
4. In casutele corespunzatoare unor valori numerice se introduc caractere nepermise:
   * Apare o exceptie de tipul format number exception .
   * Se asteapta introducerea unor date valide

-Se revine la scenariul de succes.

1. Datele introduse nu corespund niciunei valori din lista de produse

--------------------------------------------------------------------------------------------------------------------------------------------

### **Use case:** administrarea produselor

**Actor principal**: administratorul

**Scenariul principal de succes:**  
1.Utilizatorul cu rolul de admin se logheaza in aplicatie

2.Utilizatorul poate selecta un produs din table in cazul in care doreste sa il stearga sau sa il editeze (prin apasarea butonului corespunzator)

3.Utilizatorul poate apasa butonul “Add element”

-se deschide o noua fereastra in care utilizatorul introduce datele elementului pe care doreste sa il adauge in meniu

-utilizatorul apasa butonul “Add to the menu”

4.Utilizatorul poate apasa butonul “CreateComposite”

-se deschide o noua fereastra in care utilizatorul introduce numele meniului creat

-utilizatorul poate selecta produse din table si il poate adauga la meniu prin apasarea butonului “Add item”

-utilizatorul poate finaliza meniul prin apsarea butonului “Complete menu”

**Secvență alternativă**: Date invalide

1. Utilizatorul introduce in casutele pentru numere caractere invalide

- Apare o exceptie de tipul format number exception .

- Se asteapta introducerea unor date valide

-Se revine la scenariul de succes.

2. Datele introduse nu corespund niciunei valori din lista de produse

- Scroll pane-ul din interfata grafica ramane gol pana la introducerea unor date din lista

# 3.Proiectare

**Divizarea in subsisteme/pachete**

Am decis sa folosesc o arhitectura de tip Layered care imparte sistemul in 3 layers. Layerul pentru dataLayers unde se retin clasele care definesc obiectele concrete cu care se lucreaza, structurile de date. Layerul de prezentare unde sunt clasele ce implementeaza interfata grafica. Business Logic layer ce continue clasele care se ocupa de logica aplicatiei. Este un mod ușor și intuitiv de implementare și împărtire în pachete, și nu există confuzii în legătură cu interacționarea dintre pachete.

**Impartirea in clase:**

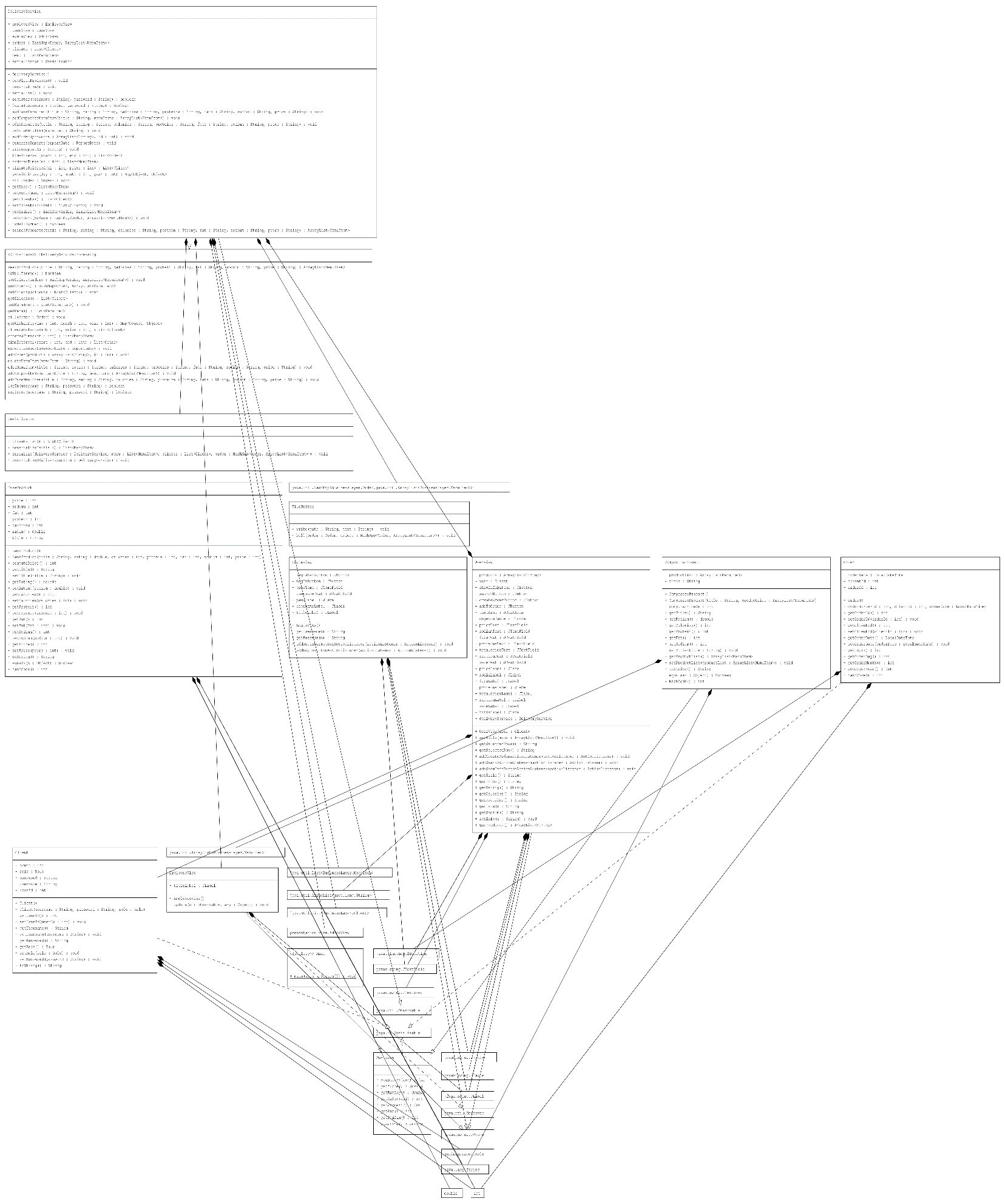
**Pachetul businessLogic** retine toata logica din spatele operatiei, creaza toata operatiile cu ajutorul claselor de access la date(serialzatoarelor) si valideaza inputurile. Acest pachet este responsabil de toate operatiile pe date. Acest pachet contine clasele: Client, Order, BaseProduct, CompositeProduct, DeliveryService, MenuItem , ReportDate; Mai contine interfata IDeliveryServiceProcessing si enumerarea Role care contine rolurile utilizatorilor: client, administrator sau employee

**Pachetul dataLayers** va reține clasele care definesc modelele de date și care le vor putea stoca persistent astfel avem urmatoarele structure: FileWriter, Serializator

**Pachetul presentation** contine doua subpachete: view si controller. Pachetul view contine clasele: AdminView, BaseView, CompositeView, EmployeeView, GenerateReport, LogInView, UserView. Pachetul controller contine clasele: AdminController, BaseController, CompositeController,GenerateReportController, UserControllor. Presentation este pachetul ce se ocupa cu interactionarea cu utilizatorul, abstractizeaza datele si lasa utilizatorul sa utilizeze metodele in mod corect.

**Diagrama de clase:**

Unified Modeling Language (prescurtat UML) este un limbaj standard pentru descrierea de modele și specificații software. Diagrama de clase UML este folosită pentru reprezentarea vizuală a claselor și a interdependențelor, taxonomiei și a relațiilor de multiplicitate dintre ele. Diagramele de clasă sunt folosite și pentru reprezentarea concretă a unor instanțe de clasă, așadar obiecte și a legăturilor concrete dintre acestea.



**Structuri de date:**

Structurile de date predefinite pe care le-am folosit sunt: ArrayList, List, HashMap, HashSet, Map. Toate acestea sunt folosite pentru retinerea datelor.

HashMap e implementarea bazată pe tabelul Hash a interfeței Map. Această implementare oferă toate operațiunile de hartă opționale și permite valori nule și cheia nulă. (Clasa HashMap este aproximativ echivalentă cu Hashtable, cu excepția faptului că este nesincronizată și permite valori nule.) Această clasă nu oferă garanții cu privire la ordinea hărții; în special, nu garantează că comanda va rămâne constantă în timp.

De asemenea, clasele din BusinessLogic pot fi văzte ca niste structure de date ce definesc atributele importante pentru fiecare obiect in parte, deoarece clasele din model definesc clasele concrete, palpabile, care exista si in realitate.

**Interfete folosite:**

Observatorul este un model comportamental de proiectare. Specifică comunicarea între obiecte: observabil și observator. Un observabil este un obiect care notifică observatorii despre schimbările în starea sa.

Mi-am definit de asemenea propria interfata, IDeliveryService care este un contract pentru functionalitatile sistemului, si este implementata de clasa DeliveryService.

# 4.Implementare

## **Clase**

Clasele sunt impartite in mai multe pachete in functie de rolul pe care il au in cadrul proiectului:

* Pachetul “Presentation Layer” care contine clasele ce implementeaza interfetele grafice
* Pachetul “Business” care contine clasele cu functiile realizate de program
* Pachetul “Data” care contine clase folosite pentru scrierea datelor in fisiere sau citirea acestora.

### **Interfata grafica**

Proiectul contine trei interfete grafice principale; cate una pentru fiecare categorie de utilizatori.

**Interfata pentru CLIENT** care poarta acelasi nume, contine o fereastra principala pentru autentificare, o fereastra pentru meniul principal de unde se poate selecta operatia dorita si cate o fereastra pentru fiecare actiune efectuata.Pagina pentru autentificare contine doua text fielduri in care se introduc numele de utiliztor si parola si doua butoane: unul pemtru autentificare pentru clientii ale caror date au fost deja salvate, si un buton pentru inregistrarea unui nou client. Pagina pentru cautarea unui produs contine text fielduri pentru fiecare criteriu in functie de care se poate face cautarea, cate un buton pentru fiecare categorie care este apasat pentru fiecare cautare, si un panou pentru afisarea rezultatelor. Fereastra pentru plasarea unei comenzi care contine un tabel pentru slectarea produselor dorite, un buton pentru adaugare si unul pentru finalizarea comenzii.

**Interfata pentru ANGAJAT** care contine doar notificarea primita.

**Interfata pentru ADMIN** contine operatiile pentru citirea produselor, administrarea acestora, adaugarea unui produs compus si generarea rapoartelor. Fereastra pentru administrarea produselor contine text fielduri, si butoane pentru adaugarea, stergerea sau modificarea unui produs . Fereastra pentru crearea unui produs compus contine un text field, un buton pentru adaugarea produselor si unul pentru adaugarea produsului finalizat. Fereastra pentru generarea rapoartelor cintine text fieldurile pentru datele necesare pentru fiecare raport si un buton pentru generarea rapoartelor.

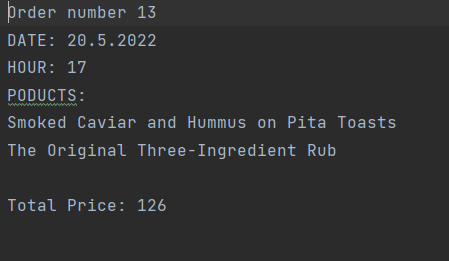
### **Clasa Delivery Service**

Clasa Delivery Service este clasa de baza a proiectului si cuprinde metodele ce implementeaza operatiile realizare de catre client si manager. Pe langa implementarea operatiilor, aceasta clasa contine si un constructor in care sunt apelare si functiile pentru deserializarea datelor si initializarea acestora. Clasa implementeaza interfata IDelivery Service Processing , utilizand Design by Contract Pattern si definind pre si postconditii dar si un invariant al clasei penttu care a fost implementata o metoda de tipul “Well formed” care verifica conditiile care trebuie sa fie indeplinite pentru apelarea fiecarei metode. Pe langa aceste metode; in aceasta clasa mai sunt implementate si metode care seteaza si returneaza valoarea unei variabile a clasei.

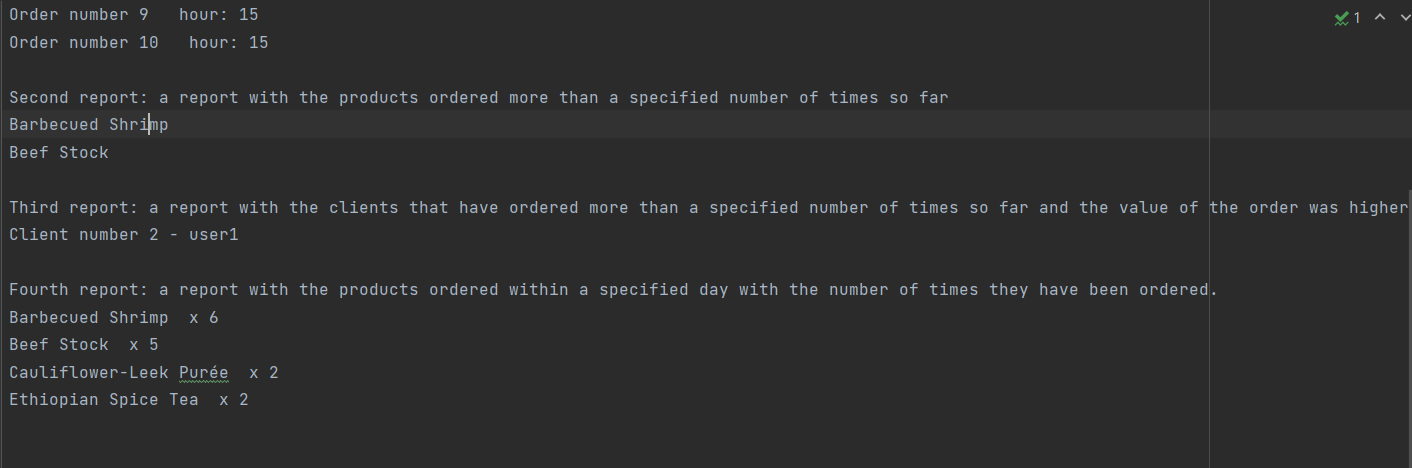
Clasele **Base Product** si **Composite product** extind clasa abstracta **Menu Item**, implementand Composite Design Pattern.

# 5.Rezultate

Asa arata un bon:



Asa arata fisierul cu rapoarte:



# 6.Concluzii

În concluzie realizarea acestui proiect a fost foarte utilă , amintind de o mare parte din principalele concepte ale programării orientate pe obiecte învățate semestrul trecut.

Am învățat să lucrez cu steamuri, sa folosesc Java Serializable, Lamda expressions si interfetele Observer si Observable.

O dezvoltare ulterioară ar fi adaugarea unui email clientului plus un validator pentru email, sa se poata updata produsele prin incrementarea cantitatii, si sa se creeze un pop-up cu factura. De asemenea se mai poate adauga un cont bancar pentru clienti si sa poata face comanda numai daca au destui bani in cont.

# 7.Bibliografie

* <https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>
* <https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html>
* <https://howtodoinjava.com/java8/java-stream-distinct-examples/>
* <https://www.geeksforgeeks.org/serialization-in-java/>
* http://javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.html