DOCUMENTATIE

TEMA *NUMARUL 4*

NUME STUDENT: Modrea Elena Alexandra

GRUPA: 30223

PROFESOR COORDONATOR : Alexandru Rancea

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 3](#_Toc95297887)

[4. Implementare 3](#_Toc95297888)

[5. Rezultate 3](#_Toc95297889)

[6. Concluzii 3](#_Toc95297890)

[7. Bibliografie 3](#_Toc95297891)

# Obiectivul temei

Obiectivul temei numarul 4 este de a implementa si proiecta o aplicatie ce are ca scop gestionarea comenzilor de mancare ale unei companii de catering.

Ca obiective secundare tema 4 le prezinta pe urmatoarele:

* Analiza problemei si identificarea cerintelor;
* Proiectarea unei aplicatii de gestiune a comenziloe de mancare;
* Implementarea aplicatiei de gestiune a comenzilor de mancare;
* Testarea aplicatiei de gestiune a comenzilor de mancare;

# Analiza problemei, modelare, scenarii, cazuri de utilizare

* + Aplicatia de gestionare ar trebui sa permita adminului sa adauge produse.
  + Aplicatia de gestionare ar trebui sa permita adminului sa editeze produse si atributele acestora.
  + Aplicatia de gestionare ar trebui sa permita adminului sa vizualize produsele si atributele acestora.
  + Aplicatia de gestionare ar trebui sa permita adminului sa stearga produsele selectate.
  + Aplicatia de gestionare ar trebui sa permita adminului sa creeze meniuri compuse din mai multe produse de baza.
  + Aplicatia de gestionare ar trebui sa permita adminului sa importe setul initial de produse si de useri
  + Aplicatia de gestionare ar trebui sa permita adminului sa genereze cele 4 tipuri de raporturi si sa le descarce in format .txt
  + Aplicatia de gestionare ar trebui sa permita adminului sa se logheze in aplicatie folosind usernameul si parola corespunzatoare.
  + Aplicatia de gestionare ar trebui sa permita clientului sa comande produse din lista de produse a firmei de catering.
  + Aplicatia de gestionare ar trebui sa permita clientului sa descarce factura comenzii plasate in format .txt
  + Aplicatia de gestionare ar trebui sa permita clientului sa se logheze in aplicatie folosind usernameul si parola corespunzatoare.
  + Aplicatia de gestionare ar trebui sa permita clientului sa se poata inregistra cu o parola si un username pe care le poate folosi ulterior pentru a se loga.

# Proiectare

Proiectul este impartit in trei pachete: BusinessLayer, Interface, Serialization.

Pachetul BusinessLayer contine 7 clase: BaseProduct, CompositeProduct, DeliveryService, MenuItem, Order, ReadFromFile, ReportsMethods, User. De asemenea acest pachet contine si o interfata IDeliveryServiceProcessing. Contine si o enumeratie TypeUser.

Pachetul Interface are in componenta un numar de 5 clase care au rolul de a defini interfata grafica cu utilizatorul. Clasele din componenta sunt: AdministratorInterface, ClientInterface, EmployeeInterface, LogInInterface, Reports. De asemenea acest pachet contine si 2 clase ce au metoda main: Test si TestInterface.

Pachetul Serialization are in componenta o singura clasa si anume clasa Serializator.

# Implementare*.*

**Pachetul BusinessLayer:**

**Clasa BaseProduct:** Aceasta clasa extinde clasa MenuItem si implementeaza Serializable.

Aceasta este formata din 7 atribute ce vor fi declarate de tip private: String title, double rating, int calories, int protein, int fat, int sodium, int price.

De asemenea are un constructor cu parametrii: **public BaseProduct**( String title, double rating, int calories, int protein, int fat, int sodium, int price ) ce initializeaza toti parametrii.

Ca si metode implementate avem metoda **public String toString()** pentru a putea afisa mai usor informatiile din aceasta clasa.

De asemenea avem metode de tip getter si setter pentru fiecare atribut in parte:

**public String getTitle()** – esteun getter pentru atributul title ce o sa returneze atributul title deoarece este de tip private.

**public void setTitle(String title)** – este un setter pentru atributul title ce o seteze atributul title cu variabila title data ca parametru, deoarece este de tip private

public

**public double getRating()** – este un getter pentru atributul rating ce o sa returneze atributul rating deoarece este de tip private.

**Public void setRating(double rating)** – este un setter pentru atributul rating ce o sa seteze acest atribut cu variabila rating data ca parametru, deoarece este de tip private

**Public int getCalories()** – este un getter pentru atributul calories ce o sa returneze atributul calories deoarece este de tip private.

**Public void setCalories(int calories)** – este un setter pentru atributul calories ce o sa seteze acest atribut cu variabila calories data ca parametru deoarece este de tip private.

**Public int getProtein()** – este un getter pentru atributul protein ce o sa returneze atributul protein deoarece este de tip private

**Public void setProtein(int protein)** – este un setter pentru atributul protein ce o sa seteze acest atribut cu variabila protein data ca parametru deoarece este de tip private.

**Public int getFat()** – este un getter pentru fat ce o sa returneze atributul fat deoarece este de tip private.

**Public void setFat(int fat** ) – este un setter pentru atributul fat ce o sa seteze acest atribut cu variabila fat data ca parametru deoarece este de tip private

**Public int getSodium()** – este un getter pentru atributul sodium ce o sa returneze atributul sodium pentru ca este de tip private

**Public void setSodium(int sodium)** – este un setter pentru atributul sodium ce o sa seteze acest atribut cu variabila sodium data ca parametru deoarece este de tip private.

**Public int getPrice()** – este un getter pentru atributul price ce o sa returneze atributul price pentru ca este de tip private.

**Public void setPrice(int price)** – este un setter pentru atributul price ce o sa seteze atributul price cu variabila price data ca parametru deoarece este de tip private.

**Clasa CompositeProduct:** aceasta clasa extinde MenuItem si implementeaza Serializable.

Are 2 atribute de tip private: String name si List<BaseProduct> products.

Are de asemenea atat un constructor ce seteaza cei 2 atributi, cat si gettere pentru toate atributele din BaseProduct dar care reprezinta de fapt suma atributelor din lista de produse.

**Clasa MenuItems** : aceasta clasa implementeaza Serializable.

Este o clasa abstracta care are ca metode urmatoarele:

public abstract String getTitle();

public abstract double getRating();

public abstract int getCalories();

public abstract int getProtein();

public abstract int getFat();

public abstract int getSodium();

public abstract int getPrice();

Aceste metode vor fi implementate de clasa BaseProduct si CompositeProduct.

**Clasa Order**: aceasta clasa implementeaza Serializable. Contine urmatoarele attribute de tip private: int idOrder, int idClient, Date date.

De asemena aceasta are in componenta sa si un constructor ce are ca rol sa initializeze atributele, precum si gettere si settere pentru fiecare atribut in parte.

**Clasa User**: aceasta clasa implementeaza Serializable. Contine urmatorele atribute de tip private: int userId, String username, String typeuser, String password.

De asemenea aceasta are in componenta sa si un constructor ce are rol sa initializeze atributele, precum si gettere si settere pentru fiecare atribut in parte.

**Enumeratia TypeUser:** este o enumeratie ce ne permite sa identificam tipurile de utilizatori gasiti in aceasta interfata: administrator, client, employee.

Clasa ReadFromFile: este o clasa in care am implementat 2 metode pentru a putea citii din fisier:

**Metoda readFromFile()** – cu ajutorul acestei metode vom citii din fisierul products.csv, iar cu ajutorul Stream-urilor acest lucru este posibil. Urmatorul pas este sa eliminam duplicatele din acest fisier, urmand ca la final sa punem intr-o lista elementele citite din fisier.

**Metoda readUserFromFile()-** returneaza o lista de tip User. Aceasta metoda ne ajuta sa citim userii deja existenti in fisier pentru a ne putea folosii de ei, acest lucru fiind posibil datorita stream-urilor. La final vom pune elementele citite intr-o lista.

**Clasa DeliveryService**: aceasta clasa implementeaza Serializable. Aceasta clasa continue majoritatea metodelor utilizate pentru ca aceasta aplicatie sa functioneze:

Prima metoda, metoda **DefaultTableModel tableModel()-** este o metoda folosita pentru a crea tabelul cu produsele initiale. Acest lucru se realizeaza astfel: am deserializat lista cu MenuItem , am create coloanele cu antetele necesare, dupa care am citit din lista elementele pe care le am adaugat intr un vector de obiecte pentru a le putea adauga la final in tabel sub forma unei linii.

Urmatoare metoda este metoda **modify()** – aceasta metoda este folosita pentru a putea modifica orice atribut al unui produs aflat in lista de produse. Acest lucru se realizeaza astfel: deserializam lista de produse, iar apoi verificam daca variabilele date ca parametru sunt nule, iar daca sunt inseamna ca acel atribut nu trebuie modificat si deci o sa il pastram pe cel vechi. Dupa aceasta verificare cream un obiect de tipul MenuItem pe care il vom adauga in lista. Ultimul pas este de a serializa lista pentru a se pastra modificarile facute.

Urmatoarea metoda este metoda **delete()** – aceasta metoda este folosita pentru a putea sterge un anumit produs din lista de produse. Acest lucru se realizeaza astfel: se deserializeaza lista de produse, apoi se elimina produsul dorit din lista, urmand ca apoi sa se serializeze din nou lista pentru a se mentine modificarile facute.

Urmatoarea metoda este **metoda add()** – aceasta metoda este folosita pentru a putea adauga un anumit produs in lista de produse. Acest lucru se realizeaza astfel: se deserializeaza lista de produse, apoi se convertesc variabilele date ca si parametru in tipurile necesare atributelor, se creeaza un nou obiect de tipul MenuItem care se adauga in lista. Ultimul pas este de a serialize lista pentru a putea mentine modificarile facute.

Urmatoarea metoda este metoda **userInit()** – aceasta metoda este folosita pentru a putea crea noi useri ce se vor putea loga in interfata noastra. Acest lucru se realizeaza astfel: deserializam lista de useri , adaugam userul dat ca si parametru, iar la final vom serializa lista pentru a mentine modificarile facute.

Urmatoarea metoda este metoda **addToComposedList()** – aceasta metoda este folosita pentru a creea o lista cu produsele selectate de administrator pe care doreste a le transforma intr-un meniu. Acest lucru se realizeaza astfel: deserializam lista de produse, cautam produsul selectat de administrator, il adaugam in lista data ca si parametru, iar la final serializam lista pentru a se mentine modifcarile facute.

Urmatoarea metoda este metoda **addToOrderList()** - aceasta metoda este folosita pentru a creea o lista cu produsele selectate de client pe care doreste a le cumpara. Acest lucru se realizeaza astfel: deserializam lista de produse, cautam produsul selectat de client, il adaugam in lista data ca si parametru, iar la final serializam lista pentru a se mentine modifcarile facute.

Urmatoarea metoda este metoda **createOrder()** – aceasta metoda este folosita pentru a creea o comanda. Acest lucru este posibil astfel: deserializam map-ul ce continue informatii despre comenzi, adaugam in el order si menuItems trimisi ca parametrii, unde MenuItems reprezinta de fapt lista create cu ajutorul metodei de mai sus, urmand ca la final sa serializam mapul pentru a putea mentine toate modifcarile facute.

Urmatoarea metoda este **totalPrice()** – aceasta metoda este utilizata pentru a calcula pretul total al unei comenzi cu ajutorul listei date ca parametru astfel: Se parcurge lista si se insumeaza pretul fiecarui produs.

Urmatoarea metoda este metoda **createBill()** – aceasta metoda este utilizata pentru a creea un fisier txt care va reprezenta facture comenzii. Acest lucru este posibil astfel: se creeaza un string in care sunt adaugate toate informatiile necesare comenzii, informatii ce sunt date ca si parametru. Apoi se scrie acest string intr-un fisier.

Urmatoarea metoda este metoda **searchByKeyword()** – aceasta metoda este folosita pentru a putea gasii produse in functie de Keyword-ul dorit. Acest lucru este posibil astfel: deserializam lista de produse , urmand ca apoi sa cream un tabel similar cu prima metoda insa diferenta facand faptul ca vor exista in tabel numai produsele in caror nume se gaseste keywordul dorit.

Urmatoarea metoda este metoda **searchByPrice()** – aceasta metoda este folosita pentru a putea gasii produse ce au un anumit pret. Acest lucru este posibil astfel: deserializam lista de produse, urmand ca apoi sa cream un tabel similar cu prima metoda insa diferenta facand faptul ca vor exista in tabel numai produsele cu pretul echivalent cu cel dorit.

Urmatoarele metode: **searchByCalories(), searchByFats(), searchByProtein(), searchByRating() si searchBySodium()** sunt similar cu metoda **searchByPrice(**), diferenta constand in conditia pentru a ajunge un produs in tabel.

**Clasa ReportsMethods** contine cate doua metode pentru fiecare raport in parte: unul pentru crearea datelor ce vor fi scrise in raport, si alta pentru scrierea efectiva in fisier sub forma unui raport.

Pachetul Interface contine clasele necesare pentru a realiza partea de interfata grafica, fiecare fereastra fiind dedicate fiecarui tip de user. In aceste clase, pe langa metodele folosite pentru a modela interfata grafica s-au folosit toate metodele implementate in pachettul BusinessLayer.

# 5.Rezultate

Pentru aceasta tema nu s-a folosit niciun framework de testare.

# 6.Concluzii *.*

*Aceasta aplicatie pentru gestionarea comenziilor companiei de catering poate fi imbunatatita astfel:*

*Prin crearea interfetei de employee cu ObserverPattern.*

*Consider ca din aceasta tema am invatat la ce este utila Serializarea, dar am si imbunatatit lucrurl cu stream-uri si lambda expressions.*

# 7.Bibliografie

* http://www.tutorialspoint.com/java/java\_serialization.htm
* https://www.baeldung.com/java-serialization
* https://www.geeksforgeeks.org/serialization-in-java/
* https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html
* http://javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.html