DOCUMENTATIE

TEMA *4*

NUME STUDENT: Tîrcă Mihnea

GRUPA: 30224

# 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 acestei teme este de a implementa o aplicatie cu interfata grafica pentru o companie de livrat mancare. Clientii pot comanda din meniu, administratorul poate adauga, sterge sau modifica produse. El poate de asemenea sa genereze rapoarte cu diferiti parametri despre comenzi, produse sau clienti. Angajatul este notificat cand o comanda este efectuata pentru a putea-o livra.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

**Analiza:**

Au trebuit implementate:

Interfata grafica

Sistem de memorare a conturilor, a comenzilor si a produselor din meniu

Adaugarea de produse de baza, adaugarea de produse compuse, modificarea de produse, stergerea de produse

Efectuarea comenzilor

Livrarea comenzilor

Cautarea produselor pe baza unor parametri

Incarcarea produselor se face dintr-un fisier .csv. Incarcarea se face folosing stream-uri si expresii lambda.

Stocarea informatiilor despre utilizatori, comenzi si meniu, se face utilizand serializarea.

**Modelare:**

Structura de date „Product”, care este o clasa abstracta, extinsa de „BaseProduct” si „CompositeProduct”

Structura de date „Order”, care reprezinta comenzile

Structura de date „User”, care reprezinta utilizatorii

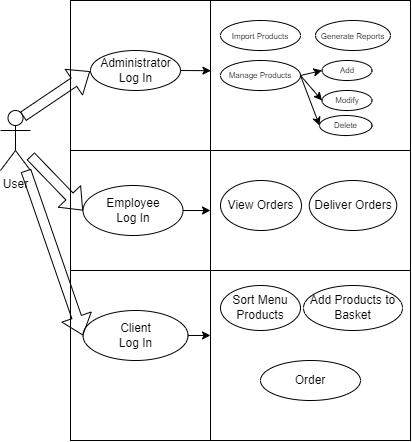
Structura de date „DeliveryService”, care tine minte produsele din meniu si comenzile

**Scenarii:**

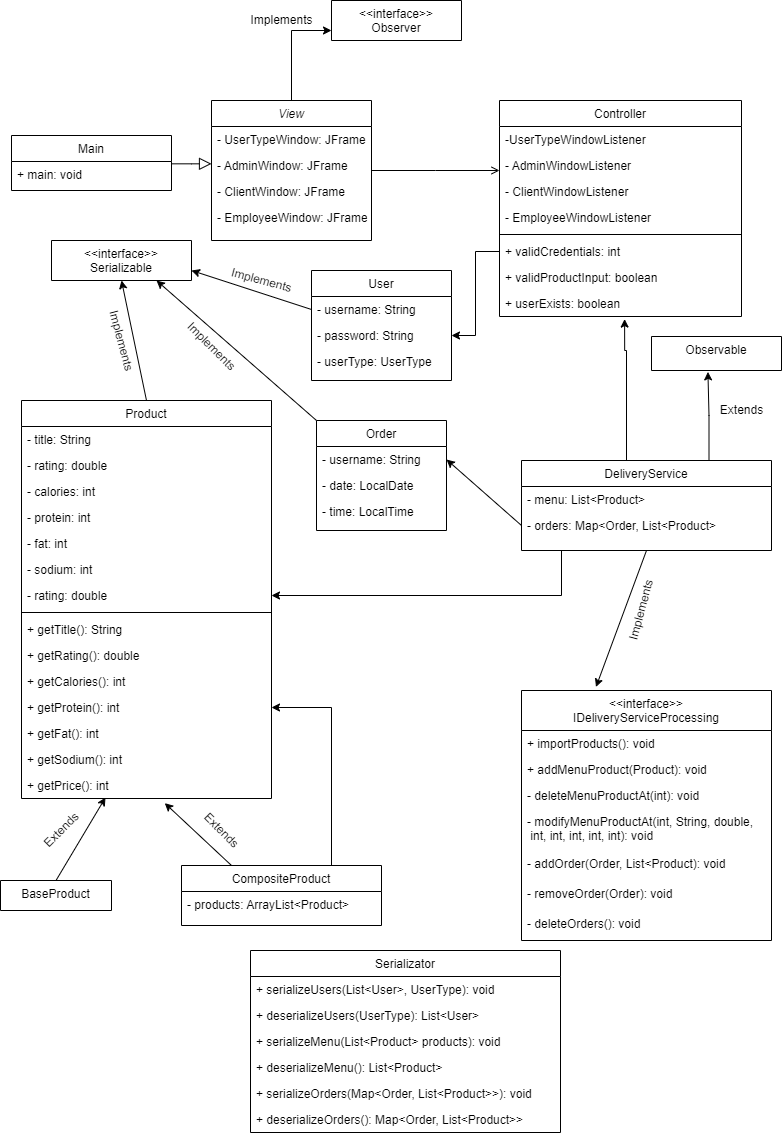
Administratorul nu are voie sa modfice in mod direct un produs compus. Daca vrea sa modifice un produs de baza din produsul compus, trebuie modificat produsul de baza si automat se va modifica si produsul compus.

In orice caz al unor intrari nevalide in oricare dintre ferestre, programul nu lasa utilizatorul sa efectueze actiunile respective, prin dezactivarea diferitelor campuri de text / butoane, sau prin atentionarea utilizatorului in scris.

**Cazuri de utilizare**

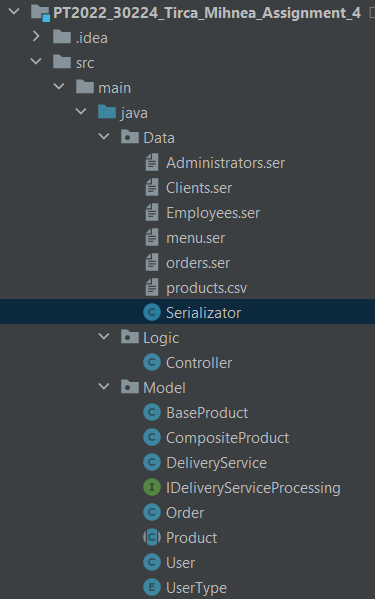


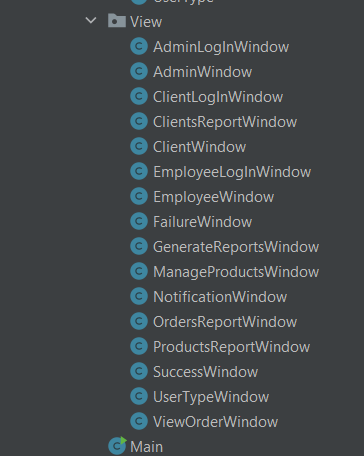
# Proiectare



Proiectul este ierarhizat intuitiv, pe baza pattern-ului Model-View-Controller. Un package este de asemenea responsabil cu stocarea datelor.

Ierarhia de fisiere a proiectului:





# Implementare

**Clasa UserTypeWindow:**

Reprezinta fereastra deschisa la rularea programului. Aici utilizatorul isi alege tipul de user care il reprezinta.

**Clasa AdminLogInWindow:**

Fereastra de log in al adminului.

**Clasa AdminWindow:**

Fereastra principala a adminului. Aici el are optiunile:

* Import products
* Manage Products
* Generate Products

**Clasa ManageProductsWindow:**

Fereastra unde adminul poate modifica meniul: adaugarea de produse, stergerea de produse, modificarea datelor produselor.

**Clasa GenerateReportsWindow:**

Fereastra de generare de rapoarte. Aici adminul poate vedea:

* comenzi facute intr-un interval de timp
* clienti care au comandat de mai mult de un anumit numar de ori
* clienti care au facut comenzi cu pret mai mare de un anumit numar
* produse care au fost comandate de mai mult de un anumit numar de ori
* produse comandate intr-o anumita zi

**Clasa OrdersReportWindow:**

Fereastra cu comenzile filtrate din clasa „GenerateReportsWindow”. Aici administratorul poate apasa pe fiecare comanda in parte pentru a vedea detalii.

**Clasa ViewOrderWindow:**

Fereastra cu detalii despre comanda selectata.

**Clasa ClientsReportWindow:**

Fereastra cu clientii filtrati din clasa „GenerateReportsWindow”.

**Clasa ProductsReportWindow:**

Fereastra cu produsele filtrate din clasa „GenerateReportsWindow”

**Clasa ClientLogInWindow:**

Fereastra de autentificare a clientului. El are optiunea de a isi creea un cont nou si de a se loga.

**Clasa ClientWindow:**

Fereastra principala a clientului. Aici este prezentat meniul. Clientul poate sorta meniul dupa titlu, rating, calorii, proteina, grasime, sodiu si pret. El poate adauga produse in cos si sterge produse din cos. El poate in final sa plaseze comanda, moment in care se va genera intr-o fereastra .txt lista cu produsele comandate si cu pretul total. O fereastra de notificare este deschisa pentru a informa clientul ca a fost plasata cu succes comanda.

**Clasa EmployeeLogInWindow:**

Fereastra de autentificare a angajatului. El are optiunea de a se loga.

**Clasa EmployeeWindow:**

Fereastra principala a angajatului. Aici el poate vedea toate comenzile efectuate si nelivrate. El poate selecta o comanda pentru a o vizualiza in detaliu.

**Clasa ViewOrderWindow:**

Fereastra pentru vizualizarea unei comenzi in detaliu. Aici sunt listate toate produsele din comanda. Angajatul poate livra comanda, ea apoi fiind stearsa din fereastra cu comenzile nelivrate din „EmployeeWindow”.

**Clasa User:**

Clasa care reprezinta utilizatorul. Ea contine urmatoarele campuri:

public String username;  
public String password;  
public UserType userType;

username: numele de logare

password: parola

userType: tipul de utilizator (Client, Admin, Angajat)

**Clasa Product:**

Clasa care reprezinta produsul. Ea este o clasa abstracta, continand urmatoarele campuri:

protected String title;  
protected double rating;  
protected int calories;  
protected int protein;  
protected int fat;  
protected int sodium;  
protected int price;

si metode abstracte:

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();

**Clasa BaseProduct:**

Clasa care repreinta un produs de baza. Ea extinde clasa „Product”, si implicit metodele si campurile ei.

**Clasa CompositeProduct:**

Clasa care reprezinta un produs compus. Ea extinde clasa „Product”, si implicit metodele si campurile ei. Aceasta clasa contine o lista de produse „Product”:

public ArrayList<Product> products;

Implementarea metodelor clasei „Product” implica, pentru:

* Titlu: concatenarea tuturor produselor din lista
* Rating: media aritmetica a tuturor produselor din lista
* Calorii: suma caloriilor tuturor produselor din lista
* Proteina: suma proteinii tuturor produselor din lista
* Grasimi: suma grasimilor tuturor produselor din lista
* Sodiu: suma de sodiu a tuturor produselor din lista
* Pret: suma preturilor tuturor produselor din lista

**Clasa Order:**

Clasa care reprezinta o comanda. Campurile sunt:

public String username;  
public LocalDate date;  
public LocalTime time;

**Interfata IDeliveryServiceProcessing:**

Interfata cu metode care reprezinta diferite actiuni ale administratorului.

**Clasa DeliveryService:**

Clasa responsabila cu stocarea meniului si a comenzilor, cat si gestionarea lor. Ea implementeaza interfata „IDeliveryServiceProcessing”. Campurile sunt:

public List<Product> menu;  
public Map<Order, List<Product>> orders;

**Clasa Serializator:**

Clasa cu diferite metode de serializare:

* a utilizatorilor
* a comenzilor
* a meniului

# Rezultate

Prin testarile facute, nu s-a observat niciun caz particular de nefunctionare a programului. Programul nu lasa utilizatorul sa introduca date nevalide, precum:

* sa se inregistreze cu un nume deja existent
* sa se logheze cu date nevalide
* sa introduca un anumit tip de date intr-un camp de alt tip
* etc.

# Concluzii *.*

Cu acest proiect am invatat sa lucrez cu stream-uri, expresii lambda si sa folosesc serializarea. Am lucrat de asemenea cu interfete si cu clase abstracte, specifice OOP.

# Bibliografie

<https://docs.oracle.com/javase/7/docs/>

<https://geeksforgeeks.org/>

https://stackoverflow.com/