**Food Delivery**

Documentatie

Munteanu Cezar-Lucian

Grupa 30226

**Cuprins :**

1. Cerinte Functionale
2. Obiective
   1. Obiectiv principal
   2. Obiectiv secundar
3. Analiza problemei
4. Proiectare
   1. Diagrama de clase
   2. Structuri de date
   3. Algoritmi
5. Implementare
6. GUI
7. Concluzii si posibilitati de dezvoltare
8. Bibliografie
9. **Cerinte Functionale**

Se cere crearea si implementarea unui sistem de livrare a mancarii pentru o firma de catering . In care clientul poate sa isi aleaga produse din meniul companiei . De asemenea trebuie sa se implementeze si un sistem de login pentru client , administrator si angajat .

1. **Obiective**
   1. **Obiectiv principal :**

Obiectivul principal al acestei teme este cel de a proiecta si implamenta o aplicatie pentru o companie de catering unde administratorul poate importa produce din fisiere .csv , sa poata modifica produsele din menu ( se adauge , sa stearga , sau sa modifice ) , sa genereze raporturi despre anumite produse ,clienti , intervalul de ora in care s-au comandat produsele . Clientii pot vizualiza menu sis a plaseze o comanda , care va fi interceptata de catre un angajat .

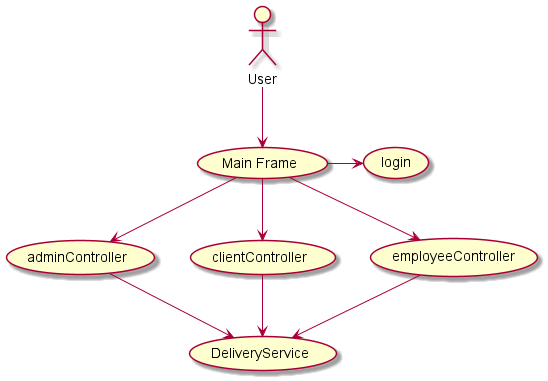
* 1. **Obiectiv secunar :**

Pentru a putea ajunge la obiectivul final trebuie ca problema sa fie impartita pe probleme mai mici , respective obiective mai mici . Asadar , obiecivele secundare sunt :

* **Dezvoltarea de use case- uri si scenarii :** reprezinta scenariile in care programul merge , respective cazurile in care merge si modul in care trebuie utilizat programul pentru ca utilizatorul sa nu aiba un rezultat nedorit . In cazul cirent , pentru ca un utilizator al aplicatiei sa aiba acces , el trebuie sa aiba un cont facut , altfel va aparea un mesaj de eroare . In cazul in care se intampla sa avem aceeasi cheie si pentru a evita suprascrierea intr-un hashMap trebuie sa tratam cu atentie modul in care adaugam intr-un map .
* **Alegerea structurilor de date :** Avem nevoie de anumite structure de date pentru a realiza operatiile asupra comenzilor , salvarii utilizatorilor si asa mai departe . Ca de exemplu , pentru a putea stii detaliile unei comenzi , vom avea o clasa si ca sa stim ce produse s-au comandat vom folosi un ArrayList de MenuItems, iar acestea vor fi mapate intr-un HashMap , ca sa avem o legatura intre cele doua obiecte si ca sa putem realiza serializarea , respective deserializarea .
* **Dezvoltarea algoritmilor :** pentru a putrea implementa operatiile corespunzatoare de adaugare intr-un Map sau de a modifica continutul pe care il are menu-ul trebuie sa folosim structure de date , algoritmi si pattern-uri de urmat si stabilit . De exemplu Composition pattern , il vom folosi pentru a putea face operatii atat cu base product cat si cu composite product , fara a fi nevoiti sa verificam la fiecare pas cu ce fel de clasa lucram .
* **Testarea solutiei :** Dupa finalizarea implementarii solutiei , trebuie sa testam cat maim ult solutia pentru a ne asigura ca este cat mai corecta si eficienta .

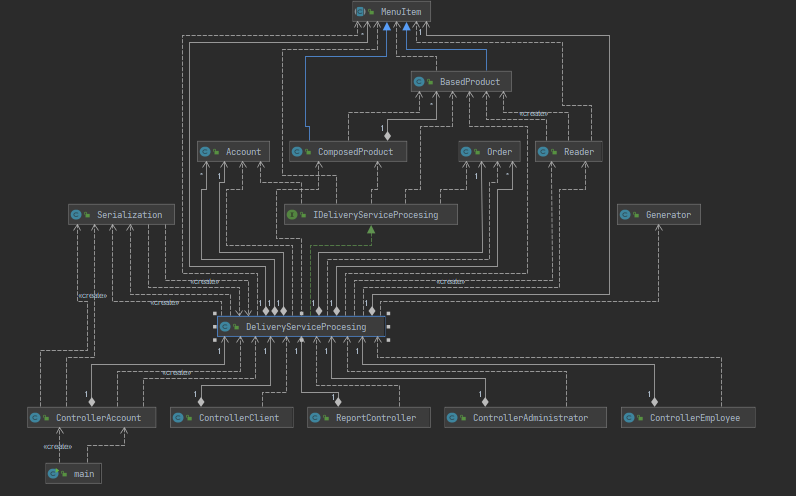
1. **Analiza problemei**

Diagrama USE – CASE reprezinta cea mai simpla reprezentare si a interactiunii unui utilizator cu sistemul / programul , astfel intelegand procesul de interactiune cu mult mai usor . In cazul acestei aplicatii , utilizatorul va avea acces la un menu principal in care se va loga in functie de ce reprezinta el , client , administrator sau angajat . Pentru un client , el va putea vizualiza produsele apasand pe butonul view products , iar apoi va putea adauga produse intr-o lista m scriind numele produselor alese si apasand pe Order . In momentul in care a adaugat tot ce si-a droit , apasa pe Order . In acest moment se creaza o noua comanda , iar adminisitratorul poate sa importe produsele dintr-un fisier , sa adauge, sa stearga sau sa modifice produse , apasand butoanele corespunzatoare . Poate de asemenea sa genereze un nou raport , la apasarea butonului va aparea pe ecran fereastra unde va scrie ce vrea el sa afle de la raportul current . Angajatul va fi anuntat de fiecare data cand se va plasa o noua comanda .



1. **Proiectare**
   1. **Diagrama de clase**

Diagrama de clase UML este o structura de diagrama statica care descrie structura unui sistem afisand clasele , atributele si metodele implementate si relatia dintre obiectele proiectului . Mai jos avem reprezentata o diagrama UML simplificata pentru a putea urmarii mai usor relatiile dintre obiecte .



Aici avem o reprezentare mai detaliata a diagramei UML , la nivelul claselor proiectului .

* 1. **Structuri de date**

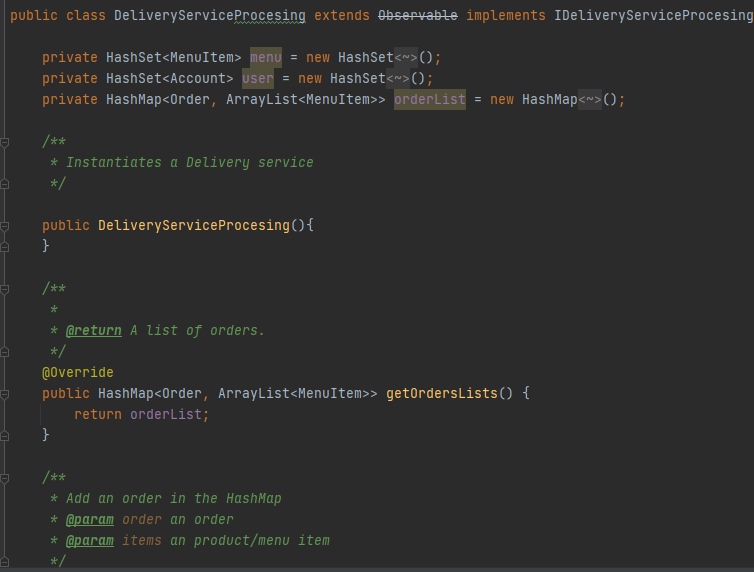
In acest proiect a fost nevoie de mai multe tipuri de structure de date . Se cere folosirea unei coletii de tip HashMap<Order , ArrayList<MenuItem>> , pentru a putea face legatura dintre o comanda si produsele selectate de catre client . Aceasta strucutra am instantiate-o la un HashMap , pentru a putea salva in combinatie , cu ajutorul metodei put() . Pentru a putea salva produsele din fisierul .csv am folosit un HashMap care va contine tot ce trebuie ca citirea sa se faca direct si va fi returnat sub aceeasi forma pentru a simplica problema .

* 1. **Algoritmi**

Pentru acest proiect a trebuit sa implementez niste algoritmi pentru a realiza operatiile specific fiecarei categorii de utilizator . Pentru rapoartele administratorului si criteriile de cautare ale unui client am folosit functii lambda si expresii stream . Astfel , am putut itera prin listele corespunzatoare intr-un mod efficient . La fel a trebui dezvoltata o metoda pentru login-ul utilizatorilor unde ne folosim de un hash set si operatiile specific . Am folosit diverse design pattern-uri pentru a putea compacta codul . De exemplu am folosit Composition pentru a nu scrie cod diferit in cazul in care avem de implementat mai multe tipuri de produse . De asemenea am folosit Design pattern-ul Observer pentru a anunta fiecare angajat atunci cand un client efectueaza comanda .

1. **Implementare**

* **Clasa DeliveryServiceProccesing**



Am ales sa incep cu aceasta clasa pentru ca ea este cea mai importanta din tot proiectul , deoarece implementeaza functiile si leaga clasele astfel incat sa asigure rularea corespunzatoare a programului .

Aceasta clasa implementeaza interfata IDeliveryServiceProcesing , unde sunt implementate toate functiile necesare unui administrator sau client . Mosteneste clasa Observable pentru a realizalegatura dintre momentul in care se adauga o noua cerere si notificarea unui angajat . O explicare pe scurt a metodelor de implementare.

addBasedProduct() – aceasta metoda adauga un base product in lista noastra de menu items;

addComposedProduct() – aceasta metoda adauga un composed product in lista noastra de menu items;

addOrder() – aceasta metoda preia imputul de la client prin intermediul unui controller si va primi un obiect de tip Order si un Array list de tim menu item pentru a putea realiza adaugarea unui nou order .

removeOrder() – aceasta metoda are rolul de a sterge un anumit order din map-ul nostrum, ea primeste ca argument un obiect de tip order ce va trebuii a fi sters ;

getMenu() – aceasta metoda returneaza cel mai important lucru si anume menu-ul nostru .

addMenuItem() – aceasta metoda are rolul de a adauga in menu un produs nou, ea va primi ca argument un obiect de tip menu item pe care il va adauga la hash set-ul mare printr-o instructiune basica de add() ; Tot aici mai avem functii asemanatoare cum ar fi cea de deleteMenuItem() si editMenuItem() ;

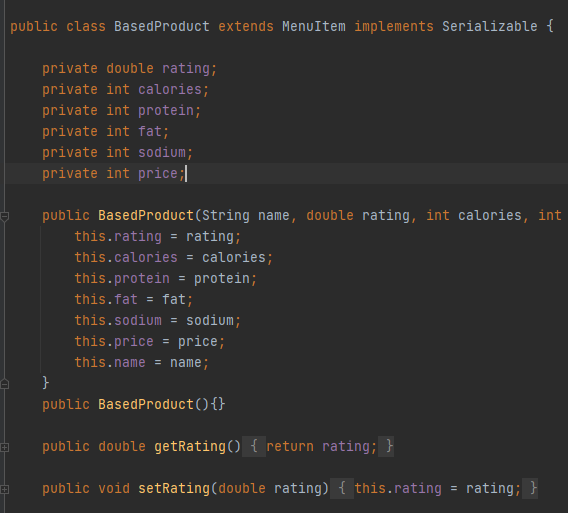
getUsers() – aceasta metoda are rolul de returna toate conturile existente in aplicatia nostra .

addUser () – aceasta metoda are rolul de adauga un user nou in aplicatie , aceasta primeste prin intermediul unui parametru un argument de tip Account pentru a putea adauga noul user .

Report() – aceastametoda genereaza un raport pentru un administrator privind anumite detalii ale timpului comenzilor , numarul de selectari ale unui produs si de asemenea clientii care au peste un anumit numar de comenzi . Aceasta metoda genereaza o lista pentru fiecare tip de cerere . Aici ne folosim de expresii stream si lambda pentru a genera listele corespunzatoare . Listele obtinute vor fi scrise intr-un fisier numit report.txt .

importProducts() – aceasta metoda creaza o noua lista de base product pentru care va complenta prin intermediul clasei Reader (aceasta clasa realizeaza citirea din fisierul .csv) . Asadar aici vom adauga in lista mare de produse .

* **Clasa BasedProduct**



Aceasta clasa modeleaza componentele unui base product . Observam ca implementeaza interfata ManuItem , acest lucru se datoreaza desig-ului pattern Composition . Avem constructorul care ia datele din fisierul sau de la utilizator si creaza un nou obiect cu proprietatile dorite .

* **MenuItem**

Aceasta intefata modeleaza comportamentul unui menu item . Realizand un pattern de Composition , daca vrem sa adaugam ulterior un nou tip de produs , aceasta abordare ne permite implementarea fara a modifica codul nostru .

* **Serialization**

Aceasta clasa este reponsabila de serializarea si respective deserializarea unei clase. Practic convertim untr-un sir de bytes. Metoda de update() se va apela atunci cand terminam executia proramului , si metoda de read() va fi executata la inceputul programului, la istantierea clasei ControllerAccount . Aici avem impementate condiitle ca atunci cand fisierul menu.txt este fol sa se instatieze correct.

* **Reader**

Aceasa clasa contine path-ul fisierului .csv pe care vrem sa il citim . Citirea se va realiza cu ajutorul stream-urilor . Mai intai fiecare linie va fi transpusa intr-un String, iar apoi acel string il vom transforma intr-un obiect de tip base product . Functia read va rreturna un hash map corespunzator datelor din fisierul .csv . De asemenea se evita existenta elementelor duplicate.

* **ControllerAccount**

Aceasta clasa contine un view specific pentru user, pe care programul il afiseaza prima data la rularea sa. Aici facem instantierea unui DeliveryServiceProcesing prin intermediul serializarii . Aceasta interfata contine un user si un field de password, pe care le va procesa pentru a realiza login-ul si a alege ce fel de user doreste sa acceze aplicatia . Daca creditentialele sunt introduce correct atunci vom fi directionati catre o alta interfata, altfel se va afisa un mesaj in care suntem rugate sa ne cream un cont .

* **ControllerAdministrator**

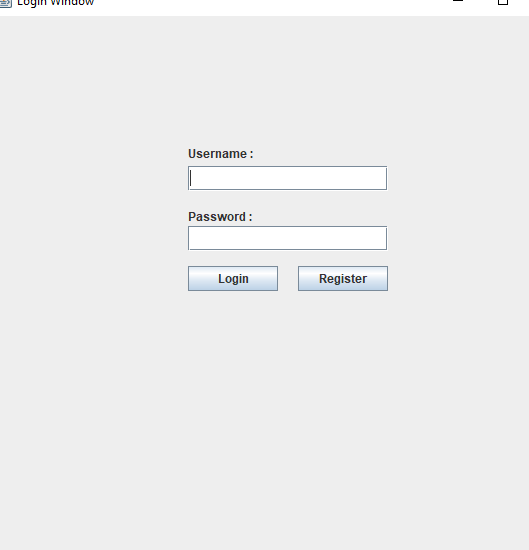
Aceasta clasa contine un view specific pentru administrator . Aici implementam fiecare actiune care poate fi executata de un administrator . De exemplu importarea produsele in menu , adaugarea sau stergerea unora deja existente .

Analog intalnim aceleasi functionalitati si la celelate clase de Controller. Fiecare avand un view specific pentru functionalitatile sale.

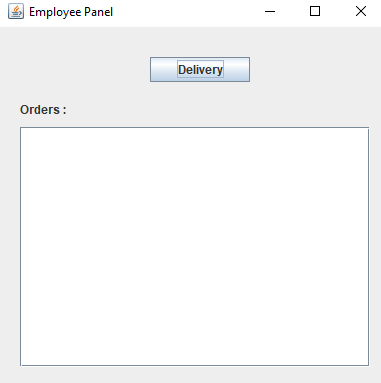
Totodata, deliveryService- ul se transmite mai departe de la un controller la altul pentru a ne asigura ca lucram pe acelasi menu si aceasi utilizatori in acelasi timp.

1. **GUI**

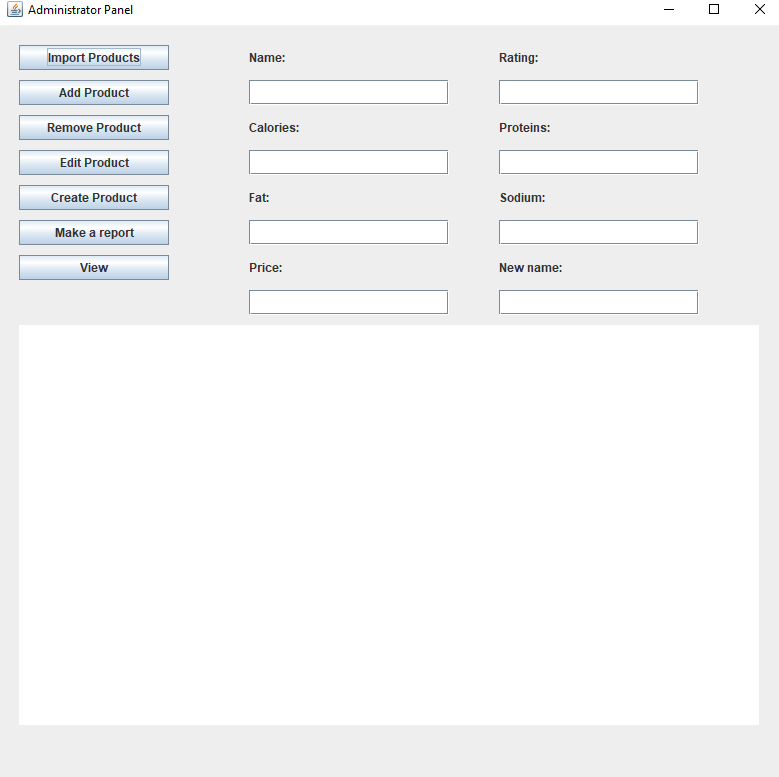
Pentru reakuzarea interfetei grafice am folosit pachetul java swing . Am folosit un frame in care am introdus un panel pe care ulterior, utilizand functionalitatile librariei date , am ales componente pe care sa le asez dupa bunul plac . La inceperea executie programul va afisa pe ecran o fereasta noua de login, unde utilizatorul isi va pune creditentialele in functie de pozitia pe care o are .



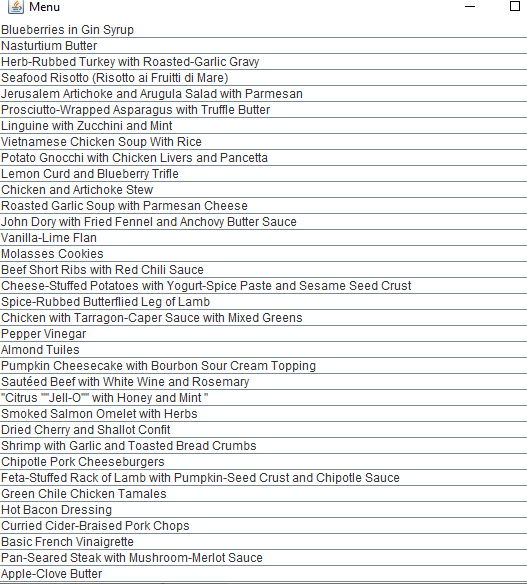
In cazul in care ne logam cu un cont de Employee, urmatoarea fereastra va aparea:



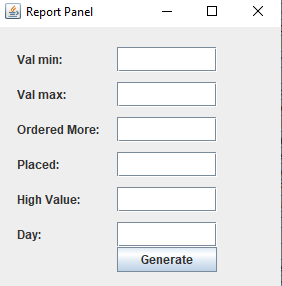
In cazul in care ne logam cu un cont de Administrator , ne aparea aceasta fereastra in care administratorul are toate comenzile .



In cazul in care un utilizator doreste sa dea un view asupra produselor acesta va primi urmatoarea fereastra:



Iar daca acesta se hotaraste sa faca un raport in legatura cu comenzile primate, acestuia I se va afisa urmatoarea fereastra :



1. **Concluzii si posibilitati de dezvoltare**

Prin intermediul acestui proiect am reusit sa invat tehnici de design pattern nou . Cum ar fi cel de composition pentru clasele Base si composite product si observer pentru a notifica de fiecaredata cand se plaseaza o comanda . De asemenea serializarea a fost un lucru nou pe care nu l-am mai folost niciodata. Ca posibilitati de dezovltare se poate face conexiunea la o baza de date care sa tina produsele firmei. Sisteml de login poate fi imbunatatit tot asa, cu o baza de date in care sa ne salvam conturile. La partea de interfata se pot aduce multe imbunatatiri atat de grafica cat si de design si functionalitate .

1. **Bibliografie**

<https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>

<https://www.geeksforgeeks.org/serialization-in-java/>

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

<http://stackoverflow.com/questions/11415160/how-to-enable-the-java-keyword-assert-in-eclipse-program-wise>

<https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html#tag>