**Assignment 4**

**Tehnici de programare**

**Tarta Manuel Vasile**

**Grupa 30225**

**Cuprins**

1. Obiectivul temei

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

3. Proiectare ( decizii de proiectare, diagrame, structuri de date, proiectare clase, relatii, interfata utilizator)

4. Implementare

5. Rezultate

6. Concluzii

7. Bibliografie

1. **Obiectivul temei**

Obiectivul temei 4 este realizarea unei aplicatii care simuleaza un Food Delivery Management System. Aplicatia trebuie sa permita inregistrarea pentru 3 tipuri de useri(administrator, client si angajat), logarea acestor 3 tipuri si in functie de ce tip de user este, trebuie sa i se permita sa faca anumite operatii. Operatiile pentru admin sunt: importul produselor dintr-un fisier csv, adaugarea unui BaseProduct, crearea si adaugarea unui CompositeProduct, editarea sau stergerea unui produs si generarea a 4 tipuri de rapoarte.

Printre obiectivele secundare ale temei se numara folosirea mai multor tipuri de Design Pattern-uri precum: Observable Design Pattern si Composite Design Pattern, generarea unui javadoc care sa contina tag-uri custom pentru pre, post conditii si invarianti, generearea unui bill dupa crearea fiecarei comenzi.

1. **Analiza problemei, modelare, scenarii, cazuri de utilizare**

Printre cerintele functionale se numara:

* Interfata ar trebui sa ii permita utilizatorului sa se inregistreze si/sau sa se logheze;
* Odata logat, utilizatorului ar trebui sa i se deschida o interfata specifica functiei pe care o are (admin, angajat sau client);
* Utilizatorului ar trebui sa i se permita sa execute operatiile specifice functiei pe care o are;
* Interfata ar trebui sa ii ofere utilizatorului un feedback daca operatia s-a realizat sau nu, si daca nu s-a realizat care a fost motivul;
* Daca utilizatorul lasa anumite campuri necompletate care trebuie neaparat sa fie completate, ar trebui sa i se indice acest lucru.

Cerintele non-functionale sunt:

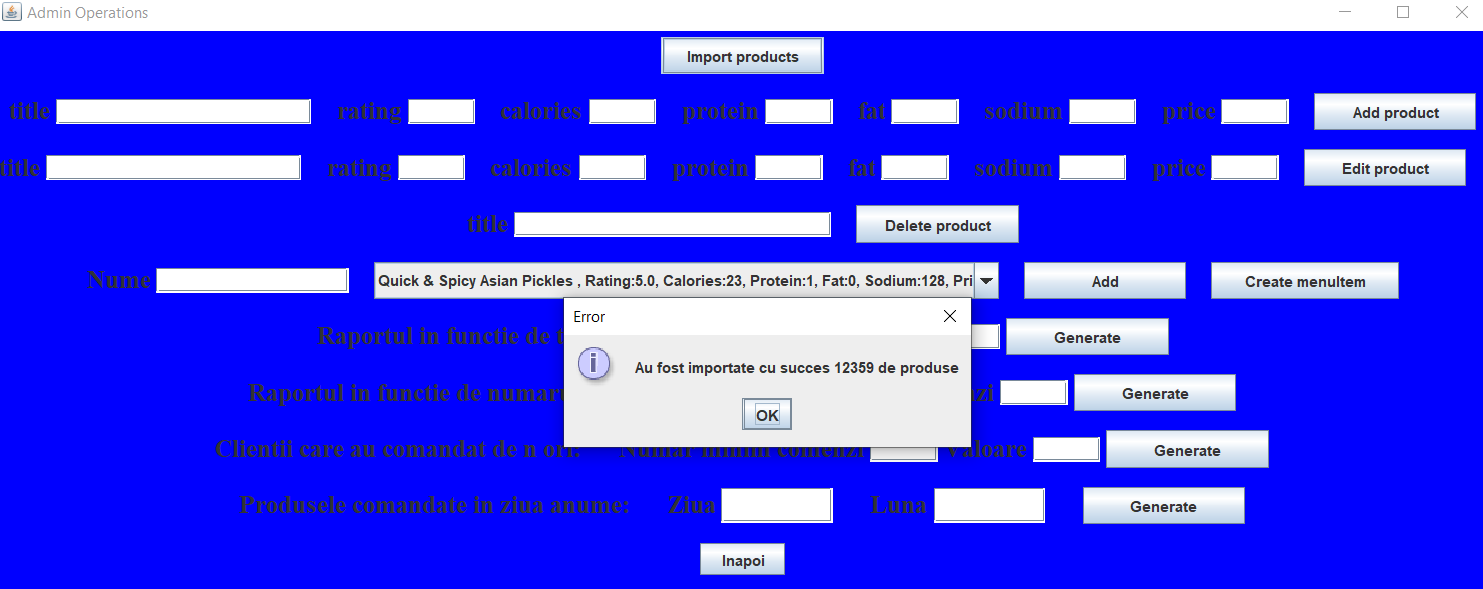
* Interfata ar trebui sa fie intuitiva, astfel incat utilizatorul sa stie ce trebuie date trebuie sa introduca pentru o anumita operatie, pe ce buton trebuie sa apese, sa stie daca operatia s-a realizat cu succes, altfel sa stie de ce operatia nu s-a putut realiza.

Use case: importul produselor din fisierul csv

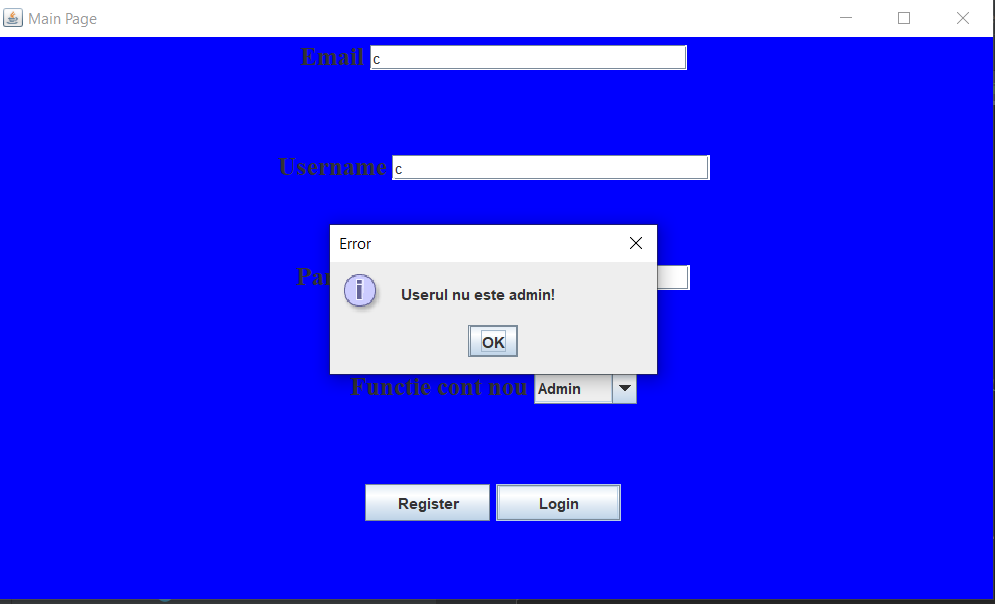
Primary actor: utilizatorul programului;

Scenariul principal de succes:

* Utilizatorul se logheaza avand functia de administrator;
* I se deschide interfata pentru admin si apasa pe butonul de import products;
* Apare mesajul care ii indica ca importul s-a realizat cu succes.



Scenariu alternativ de esec:

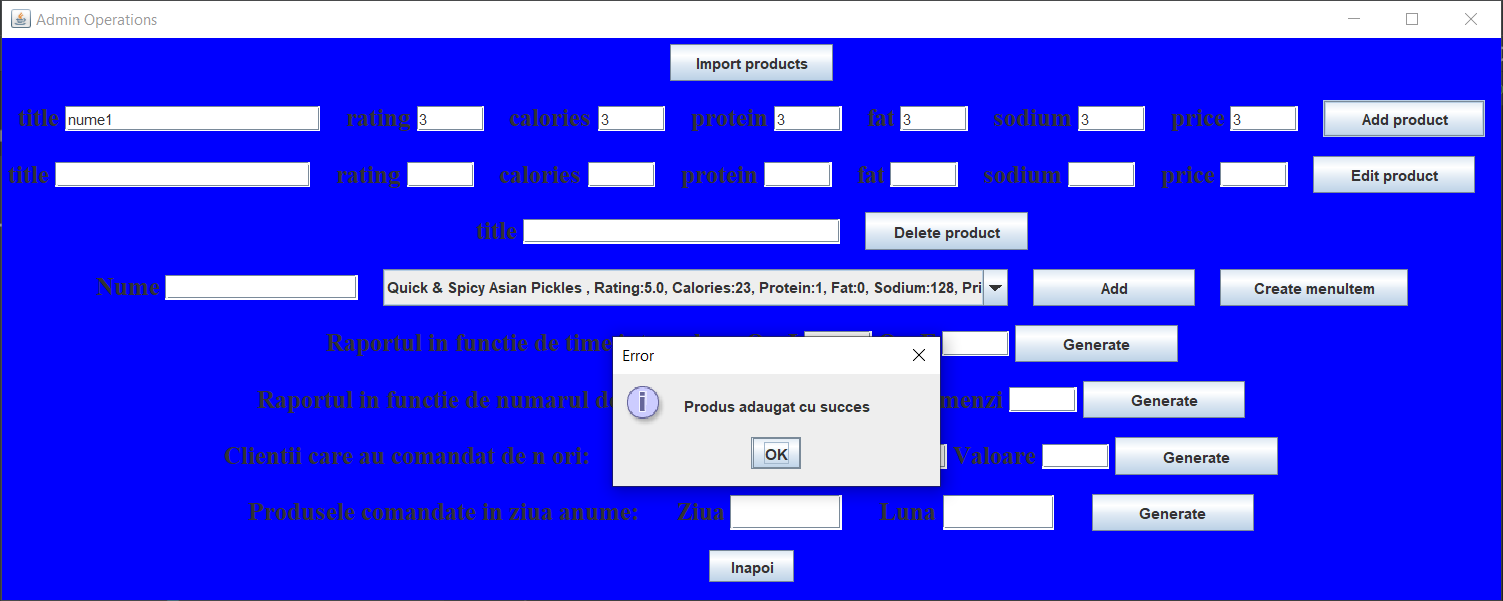
* Utilizatorul care doreste sa se logheze nu este administrator;
* Ii apare mesajul care ii indica ca nu are functia necesara pentru a se loga ca admin.
* 

Use case: adaugarea unui produs nou de catre un admin:

Primary actor: utilizatorul programului

Scenariul principal de succes:

* Utilizatorul se logheaza avand functia de admin;
* I se deschide interfata cu operatii pentru admini;
* Introduce campurile necesare pentru operatia de add si anume: nume, rating, calories, proteine, fats, sodium si pret;
* Utilizatorul apasa pe butonul de add;
* I se afiseaza mesajul care ii indica ca operatia s-a realizat cu succes;



Scenariu alternativ de esec:

* Utilizatorul doreste sa se logheze ca admin, dar acesta nu este admin;
* Ii apare un mesaj care ii indica ca nu are functia necesara.

Scenariu alternativ de esec:

* Utilizatorul se logheaza ca admin cu succes;
* Incearca sa introduca un produs al carui nume exista deja;
* Va primi un mesaj care sa ii indice ca acel produs exista deja.

Scenariu alternativ de esec:

* Utilizatorul se logheaza ca admin;
* Nu introduce toate campurile necesare pentru operatia de add;
* Va primi un mesaj care sa ii indice acest lucru.

Cam aceeasi pasi sunt necesari si pentru operatiile de edit si delete.

Use case: adaugarea unui produs composite

Primary actor: utilizatorul programului

Scenariu principal de succes:

* Utilizatorul se logheaza ca admin;
* Prin apasarea repetata a butonului de AddProduct selecteaza din combo box acele BaseProducts pe care doreste sa le contina acel CompositeProduct;
* Introduce un nume pentru noul Composite Product;
* Apasa pe butonul de Create.

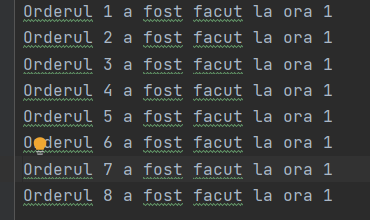
Scenariile alternative de esec constau in incercarea de logare ca admin cand nu are acea functie, neintroducerea a niciunui produs inainte de a apasa pe Create. Aceste scenarii de esec vin cu afisarea unui mesaj de eroare care sa ii sugereze utilizatorului ce trebuie sa faca pentru a reusit sa creeze un Composite Product.

Use case: generarea raportului in functie de ore

Primary actor: utilizatorul programului

Scenariu principal de succes:

* Utilizatorul se logheaza ca admin cu succes;
* Introduce o ora de inceput si o ora de final intre care doreste sa fie orderurile;
* Apasa butonul de generate;
* Se genereaza un fisier txt care contine toate orderurile realizate in acel interval orar.



Scenariile alternative de esec sunt: incercarea de logare cand nu are functia de admin, introducerea unor ore gresite ( mai mici ca 0 sau mai mari de 24). Acestea vor genera un mesaj prin care utilizatorul isi va da seama ce a introdus gresit.

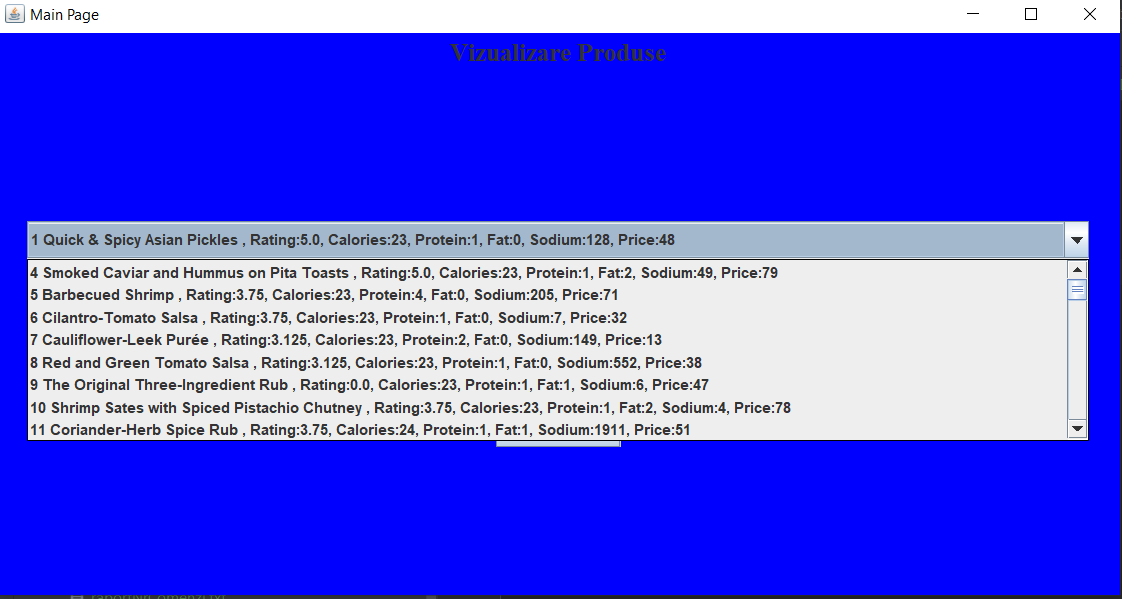
Si celelalte 3 rapoarte functioneaza dupa exact acelasi schelet, trebuie doar introduse alte date.

Use case: vizualizarea tuturor produselor

Primary actor: utilizatorul programului

Scenariu principal de succes:

* Utilizatorul se logheaza cu succes cu functia de client;
* I se deschide interfata specifica clientilor;
* Apasa pe butonul de View Products;
* I se deschide alta interfata care contine un Combo Box cu toate produsele.



Scenariul de esec consta in incercarea de logare cand nu are cont sau nu are functia de client.

Use case: Search Product;

Scenariu principal de succes:

* Utilizatorul se logheaza ca client;
* Alege din Combo Box ce criteriu doreste sa insereze si in combo Box insereaza valoarea pentru acel criteriu. Apasa pe butonul de Add Criteriu. Poate face asta cu cate criterii doreste dintre cele 7;
* Daca doreste, insereaza un keyword pe care trebuie sa il contina numele produselor;
* Apasa pe butonul de Search;
* I se deschide o interfata cu un Combo Box in care sunt toate produsele gasite.

Scenariile alternative de esec constau in incercarea esuata de logare, caz in care trebuie sa se logheze din nou, sau sa isi creeze un cont, introducerea unor valori gresite pentru criterii, cum ar fi numere negative, caz in care ii va aparea un mesaj care ii va sugera sa introduca date corecte.

Use case: Create Order;

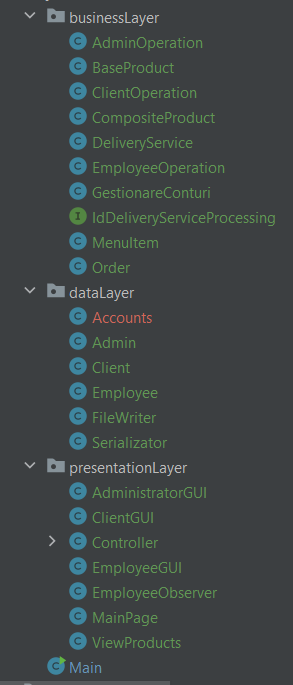
Scenariu principal de succes:

* Utilizatorul se logheaza ca client;
* Alege din Combo Box un produs pe care il doreste in order si apasa butonul de AddProduct;
* Repeta pasul de mai sus de cate ori doreste;
* Apasa pe butonul de AddOrder;
* Ii apare un mesaj de succes si se genereaza un bill in format txt.

Scenariile alternative de esec pot fi esecul in incercarea de logare sau neintroducerea niciunui produs inainte de a apasa pe butonul de Add Order.

1. **Proiectare ( decizii de proiectare, diagrame, structuri de date, proiectare clase, relatii, interfata utilizator)**

Din punct de vedere al proiectarii, am impartit programul in mai multe pachete folosind ideea de Layered Arhitecture.



Am un pachet businessLayer in care am clasele responsabile cu operatiile specifice adminului si clientului, clasele BaseProduct si CompositeProduct care extind clasa MenuItem, clasa GestionareConturi care retine cele 3 liste de useri si clasa Order.

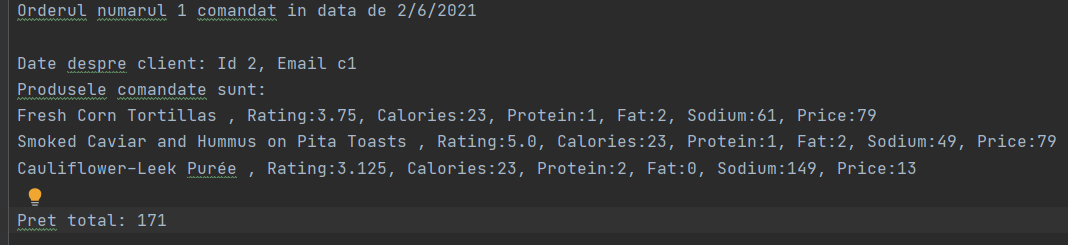
In adminOperation am operatiile specifice adminului, in clientOperation operatiile pentru client, Composite si BaseProduct doar dau super() in constructor pentru a lua cele 3 fielduri din MenuItem. In MenuItem am dat override la metoda de equals si hashCode pentru a putea folosi distinct la streams.

Totodata, in Order am dat override la aceleasi 2 metode pentru a putea face un HashMap cu Orders ca si cheie.

In dataLayer am date personale despre cele 3 tipuri de users, Clasa FileWriter in care am o metoda cu care printez chitanta pentru un order si Clasa Serializable in care am metoda de serializare.

Datele despre users le am in clasa Accounts pentru ca sunt aceleasi date pentru toate cele 3 tipuri de utilizatori. Vreau totusi sa am o separare intre ei pentru a sti cu ce ma loghez. Asa ca am datele comune in clasa Accounts si in cele 3, Admin, Client si Employee dau super() la constructorul din Accounts.

Metoda de printBill din FileWriter primeste ca parametrii Orderul pentru care fac chitanta, produsele din acel order ca o lista, si clientul care a realizat comanda si printeaza intr-un fisier txt date despre cele 3 tipuri de obiecte.



La serializare, serializez cu totul obiectul DeliveryService pentru ca in el am toate datele de care am nevoie: conturile pe care le am, toate cele 3 tipuri, produsele pe care le am si orderurile. Toate le scriu intr-un singur fisier si la deserializare le stochez in obiectul DeliveryService nou creat. Deci singurul parametru pe care il trimit metodei de serializare este obiectul DeliveryService.

In pachetul presentationLayer am interfetele grafice si Controllerul. Am o interfata pentru administrator in care am campurile necesare pentru fiecare operatie (fie ele textFielduri sau ComboBoxuri) si butoanele si am setteri pentru ele.

Mai am o interfata pentru Client care si ea contine textFields sau comboBoxuri si butoanele specifice fiecarei operatii, plus getteri pentru datele de intrare si actionListeneri la butoane.

In interfata pentru Employee am doar un TextArea in care scriu Orderurile care au fost facute si nu au fost vazute inca de niciun client. Ele vor ramane scrise acolo pana cand un client se logheaza, citeste orderurile si apasa pe butonul de Inapoi. Cand apasa Inapot se reseteaza textul din acel TextArea. Aceasta clasa contine o metoda update folosita pentru implementarea Design Pattern-ului Observer. Aceasta metoda primeste un ArrayList de produse (care reprezinta produsele dintr-un order nou creat) si le scrie in acel TextArea. Acea lista de produse este trimisa din clasa DeliveryService, care reprezinta Observable-ul, in momentul in care se apeleaza metoda de addOrder, prin intermediul metodei notifyObservers().

Mai am interfata MainPage in care se poate crea un cont sau sa se logheze prin intermediul unui cont existent. Se alege din ComboBox functia acelui cont(admin,client sau angajat) si se apasa pe butonul de Login. Daca contul exista se va deschide alta interfata in functie de functia acelui cont.

Ultima interfata pe care am creato este ViewProducts. Aceasta contine un ComboBox in care introduc fie toate produsele cand apas butonul ViewProducts din interfata ClientGUI, fie o parte din produse cand apas SearchProducts.

In clasa Controller implementez functionalitatea butoanelor pentru toate operatiile din toate interfetele.

Pentru Login, in functie de functia aleasa, cauta in lista respectiva un user cu datele date pe interfata. Daca il gaseste deschide o alta interfata, altfel se afiseaza un mesaj de eroare.

Pentru Register, se creeaza un nou user care va avea functia pe care o alege din combo box.

La import, apeleaza metoda de import din DeliveryService si afiseaza mesaje de succes, sau de eroare. La fel pentru toate celelalte operatii de genul.

La operatiile care lucreaza cu cele 3 liste din DelvieryService (useri, menuItems si orders), dupa ce am realizat operatia, fac din nou o serializare pentru a salva modificarile pentru viitoarele rulari ale programului.

1. **Implementare**

Implementarea claselor si a operatiilor am discutat-o si in comentariile pentru javadoc asa ca nu o sa vorbesc foarte in detaliu.

Pentru operatia de import, am folosit streams si lambda expressions pentru citirea linie cu linie a datelor din csv, transformarea liniilor in obiecte de tipul BaseProduct, eliminarea duplicatelor cu distinct si apoi adaugarea lor in lista de MenuItems din DeliveryService.

La add, delete si edit e simplu. Doar am luat datele din interfata si am facut operatiile necesare. Daca la delete si edit nu am gasit produsul cu acel nume, am afisat un mesaj de eroare. Daca la add exista deja un produs cu acel nume, am afisat un mesaj de eroare.

La rapoarte am folosit din nou streams si lambda expressions. Am transformat HashMap-ul de orders in stream, am filtrat orderurile in functie de ce aveam nevoie pentru raport( in functie de ore, de numarul de comenzi, de pret, sau de ziua in care a fost realizata comanda) si am extras ce era nevoie ( produsele care indeplinesc criteriul, clientii care au comandat de atatea ori, orderurile dintr-o anumit interval orar, produsele dintr-o anumita zi). Rezultatul fiecarui tip de raport l-am printat intr-un fisier txt diferit.

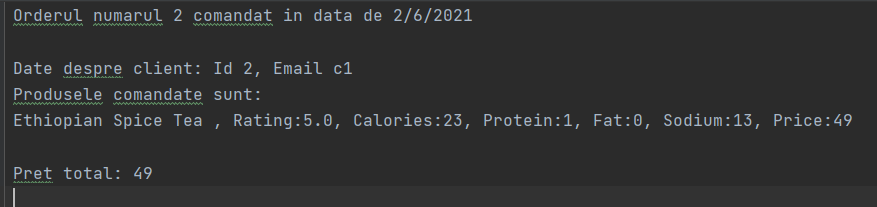
Operatia de ViewProducts este simpla. Am luat fiecare produs din lista de MenuItems din DeliveryService si l-am afisat pe un rand dintr-un ComboBox din interfata ViewProducts.

Pentru Search am un combo box din care aleg criteriul pe care doresc sa il adaug pentru cautare. Langa combo box introduc valoarea pentru acel criteriu si apas butonul de Add Criteriu. Dupa ce am adaugat toate criteriile, apas pe search. Butonul de Search apeleaza in spate metoda de searchProducts care face un filter cu ajutorul streamurilor in functie de criteriile introduse precedent si dupa afiseaza lista de produse gasite in acelasi combo box din interfata ViewProducts.

Add Order este de asemenea destul de simplu. Aleg din Combo Box un produs pe rand, putand alege oricate produse. Dupa ce s-au ales produsele, apesi pe butonul de Add Order care ia lista de produse selectate si creeaza un order, adauga in HashMap la cheia acestui order aceasta lista de produse.

1. **Rezultate**

Ca rezultate, am chitantele pentru fiecare order creat de fiecare client salvate intr-un fisier txt diferit.



Pe langa asta, am in diferite combo box-uri produsele returnate de diferite operatii. De exemplu dupa ce dau import sau add, produsele vor aparea in combo box-ul din interfata ViewProducts, dar si in alte combo boxuri pentru operatii care necesita selectarea de produse existente.

Totodata mai am 4 fisiere txt si pentru rezultatele celor 4 rapoarte.

1. **Concluzii**

A fost o tema destul de lunga, s-a simtit ca si cum sunt doua tema intr-una, dar conceptul a fost unul interesant si actual. Aplicatii precum Food Panda si Glovo au devenit destul de folosite. Si noi am implementat cam ceva de genul. Am incercat sa fac metodele sa functioneze cat de bine am putut si in mare sunt multumit de produsul final. Daca ar fi sa mai fac ceva, ar fi de exemplu ca dupa ce dau search in functie de criterii, sa pot alege din lista de produse gasite. Momentan pot doar sa le vad si stat. Nu pot face nimic cu ele.

Streamurile si lamba sunt destul de usor de implementat dupa ce te obisnuiesti cu ele si chiar fac programul mai usor de citit si fac procesarea unor siruri de date foarte mari mai usoara si mai rapida.

1. **Bibliografie**

https://www.journaldev.com/31860/java-stream-distinct-function

https://howtodoinjava.com/java8/java-stream-distinct-examples/

https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/

<https://dzone.com/articles/how-to-read-a-big-csv-file-with-java-8-and-stream>

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

<https://www.tutorialspoint.com/java/java_serialization.htm>

<https://www.baeldung.com/java-serialization>

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

<https://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html>

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