

**FUNDAMENTAL PROGRAMMING TECHNIQUES**

**ASSIGNMENT 4**

**FOOD DELIVERY MANAGEMENT SYSTEM**

**STUDENT: Balcanu Vlad-Andrei**

**Grupa: 30221**

**CUPRINS**

1. Obiectivul temei :…………………………………………………………3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare:……………....3
3. Proiectare (decizii de proiectare, diagrame UML, structure de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator):………………………………….3
4. Implementare:……………………………………………………………..5
5. Rezultate:………………………………………………………………….7
6. Concluzii:………………………………………………………………….13
7. Bibliografie:………………………………………………………………..13

**1.Obiectivul Temei**

Obiectivul principal al acestei teme a fost proiectarea si implementarea unei aplicatii de management al unui sistem de livrare a unor comenzi de mancare prin care administratorul poate importa produsele stocate in fisiere , poate crea produse noi compuse din produsele anterioare , modifica produse , sterge produse si poate genera rapoarte cu statistici pentru anmite cerinte. Clientul se poate inregistra , autentifica , sa caute produse dupa un criteriu sau mai multe criterii . plasa comenzi si crearea unei chitante la fiecare comanda . Angajatul va fi informat de fiecare data cand se plaseaza o camanda noua.

Obiectivile principale ale temei sunt :

* Folosirea unui design de programare orientat pe obiect;
* Clase cu cel mult 300 de linii de cod;
* Metode cu cel mult 30 de linii de cod;
* Folosirea conventiilor de denumire specifice Java;
* Implementarea diagramei de clase prezentata in assignment;
* Alegerea unor structure apropiate pentru salvarea comenzilor si a produselor;
* Definirea clasei BaseProduct cu urmatoarele atribute:title, rating, calories , protein ,fat , sodium ,price si crearea unei liste de obiecte de tip BaseProduct.
* Citirea produselor din fisierul products.csv (fara duplicate);
* Folosirea Stream-urilor si a expresiilor lamda pentru generarea raporturilor facute de administrator;
* Folosirea Stream-urilor si a expresiilor lamda pentru operatiile de cautare ce le poate realiza clientul
* Documentatie buna;

Obiectivile secundare ale temei sunt:

* Folosirea unui Stil de design de tipul Composite pentru modelarea claselor MenuItem , BaseProduct si CompositeProduct;
* Crearea unei chitante de tipul .txt pentru fiecare comanda efectuata;
* Preconditions si Postconditions la metodele din DeliveryService;
* Definirea unui invariant in clasa DeliveryService;
* Generarea fisierelor JavaDoc specific;
* O fereastra pentru angajat ce foloseste Design pattern-ul de tip Observer pentru a fi notificat la fiecare comanda noua;
* Salvarea informatiilor din DeliveryService intr-un fisier txt prin Serializare;

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

Serviciile oferite de aceasta aplicatie sunt acelea de a realiza management-ul in timp real al unui sistem de livrare a mancarii care are clienti , produse , administrator , angajati si comenzi. Toate acestea sunt realizate prin intermediul unei interfete grafice foarte simpla si utila pentru a face folosirea aplicatiei cat mai usoara si placuta atat pentru administrator cat si pentru clienti. Datele despre clienti , produse si comenzi trebuie sa fie cat mai usor de introdus in aplicatie de catre administrator iar rezultatele sa fie cat mai concise si inteligibile pentru a putea fi analizate cu success.

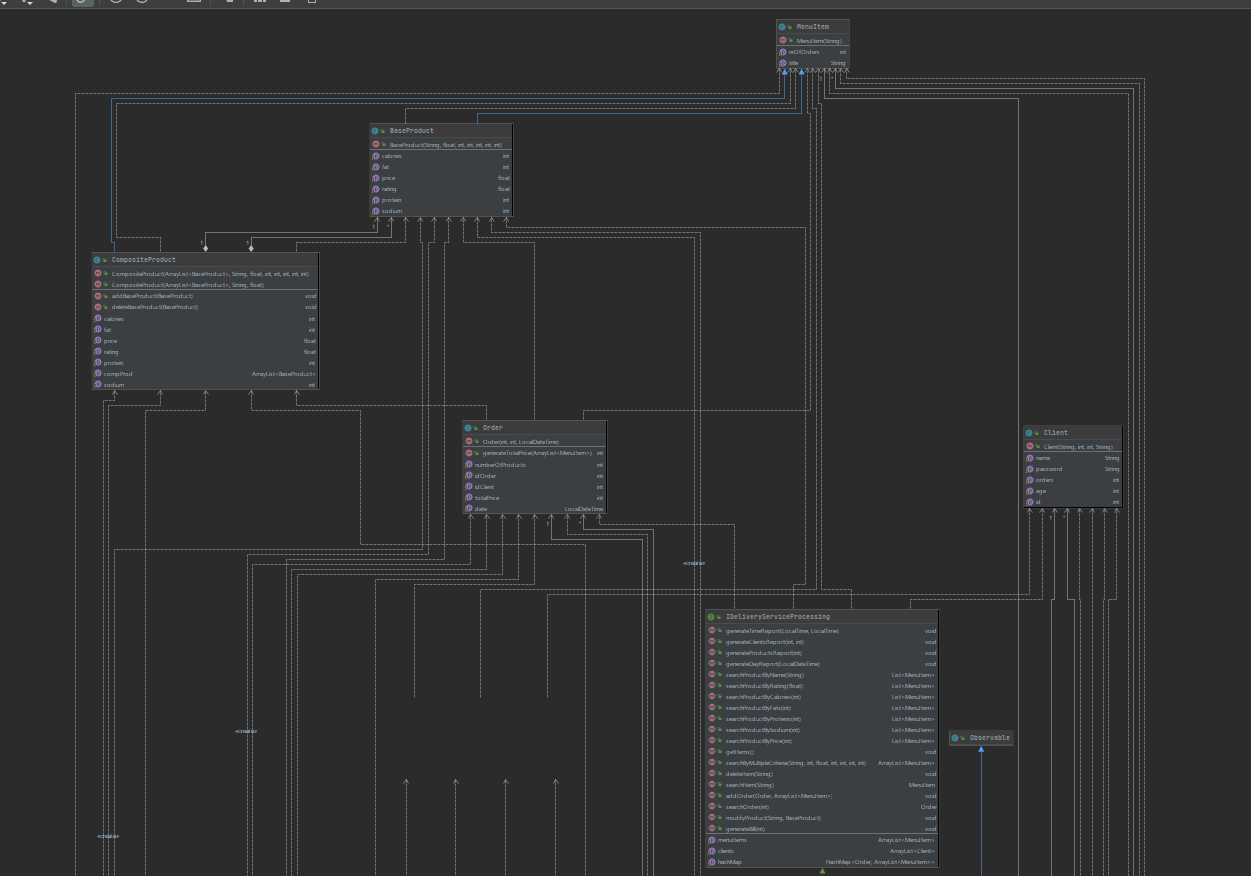
Pasii pe care utilizatorul trebuie sa ii urmeze pentru a folosi aceasta aplicatie sunt : Lansarea aplicatie care se va deschide cu fereastra de LogIn/SingUp. Utilizatorul isi poate crea un cont de client sau sa se logheze cu un cont de client deja existent sau ca si administrator. Daca se logheaza ca si client acesta va vedea fereastra de client ce contine un table cu toate produsele din meniu si butoane functionale. Acesta poate vedea produsele , sa le caute dupa anumite criterii , sa adauge produsul la comanda sis a plaseze comanda. In momentul in care acesta plaseaza comanda , fereastra de angajat se va deschide sa ne avertizeze ca s-a facut o noua comanda.

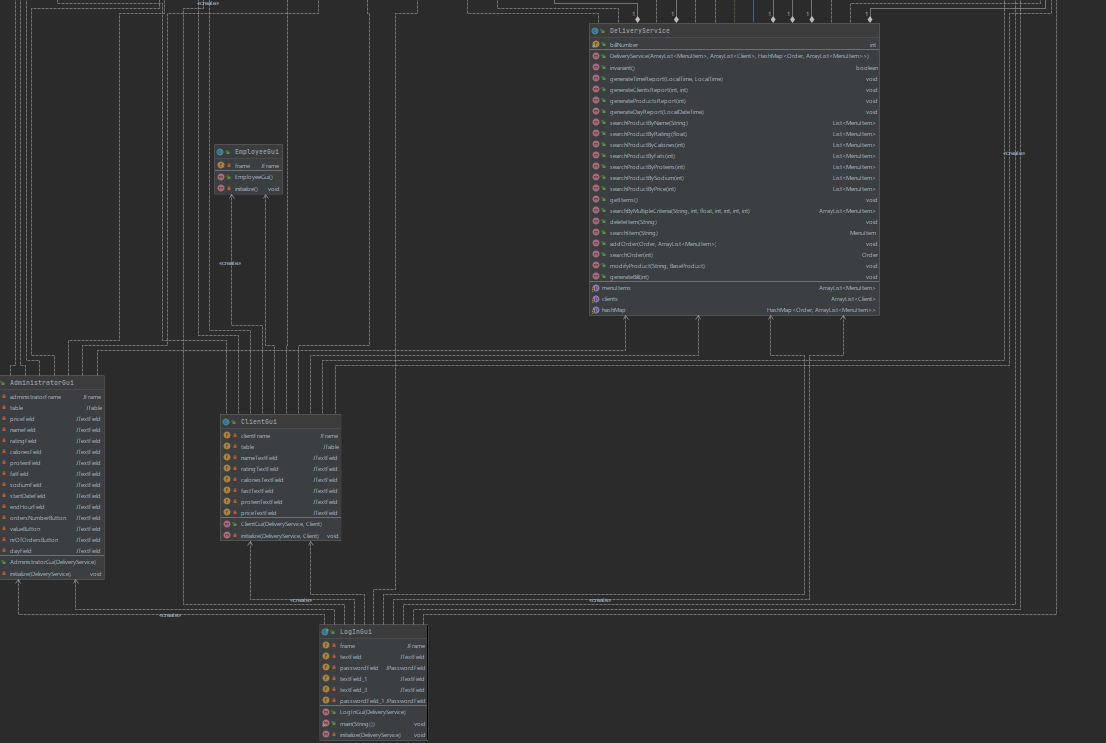
Daca se autentifica ca si administrator , se va deschide fereastra de administrator care contine un tabel cu toate produsele si butoane functionale. Acesta va putea sa importe produsele , sa le afiseze , sa le modifice , sa le stearga , sa adauge produse de baza , sa creeze produse compuse si sa genereze diferite raport-uri ce contin date folositoare pentru administrator.

**3.Proiectare (decizii de proiectare , diagrama UML , structure de date, proiectare clase , interfete , relatii , packages , algoritmi, interfata utilizator)**

Pentru proiectarea acestei aplicatii s-a utilizat procedeul de design Layered Architecture ce separa aplciatia in 3 parti mari : Data Layer , ce se ocupa cu serializarea informatiilor in fisiere pentru a putea fi stocate si extrase cand este nevoie .In acest pachet se gaseste clasa Serializator si contine metodele de serializare a produselor , clientilor si a comenzilor din aplicatie; Business Layer ce contine toate clasele si metodele ce realizeaza actiuni in aplicatie . Avem clasele BaseProduct CompositeProduct si MenuItem ce reprezinta Clasele cu ajutorul carora de creem produsele. Avem clasa de Client cu ajutorul caruia ne creem clientii din aplicatie si clasa Order ce creaza comenzile din aplicatie. Mai avem clasa DelvieryService ce implementeaza interfata IDeliveryServiceProcessing in care se afla majoritatea metodelor ce realizeaza functionaliatile aplicatiei.

In pachetul PresentationLayer avem cele 4 interfete grafice . Interfata de LogIn in care putem sa ne inregistram si sa ne autentificam in aplicatie. Interfata de Client in care clientul poate realiza operatiile dorite. Interfata de administrator in care administratorul poate face management la produse si genera raporturi si interfata de angajat care ne genereaza un mesaj la fiecare comanda noua.

**DIAGRAMA DE CLASE UML:**  




Structurile de date utilizate au fost : Liste si ArrayList-uri pentru stocarea produselor , clientilor si a comenzilor .Am mai folosit structura de date de tip HashMap<k,v> pentru a pastra comenzile impreuna cu toate produsele din acea comanda.

Au fost folosite pachetele necesare pentru implemtarea interfetei grafice (import java.awt.EventQueue; import javax.swing.JFrame ;import java.awt.CardLayout; import javax.swing.JPanel; import javax.swing.JTextField; import javax.swing.JButton;import java.awt.event.ActionListener; import java.awt.event.ActionEvent; import java.sql.SQLException; import java.util.List; import javax.swing.JTable; import javax.swing.table.DefaultTableModel;) si pachete din Java.DateTime pentru stocarea de date pentru comenzi , java.stream pentru a folosi stream-uri si lamda-uri.

**4.Implementare**

In acest capitol vor fi explicate in amanunt aspectele si implementarile pentru fiecare clasa in parte.

In pachetul BusinessLayer avem clasele BaseProduct , Client , CompositeProduct , DeliveryService , IDeliveryService, MenuItem , Observable si Order .

In clasa BaseProduct avem atributele specificate in cerinta impreuna cu gettere si setttere pentru fiecare atribut si de un constructor.

In clasa Client avem atributele String name , int age , int id , String password , int orders (in care tinem minte cate comenzi a efectuat clientul) si avem gettere si settere pentru fiecare atribut impreuna cu constructorul clasei.

In clasa CompositeProduct avem aceleasi atribute ca in clasa BaseProduct si in plus un ArrayList<BaseProduct> in care stocam produsele de baza care vor forma produsul compus. Pe langa getterele si setterele fiecarui atribut si constructorul clasei mai avem metoda de addaBaseProduct prin care adaugam un produs de baza la cel compus si adunam toate caracteristicile acestuia si metoda de deleteBaseProduct prin care stergem un produs de baza din produsul compus si scadem din atributele produsului compus , atributele produsului de baza sters.

In clasa MenuItem avem doar atributul de title sic el nrOfOrders , produsele de baza si cele compuse fiind salvate sub forma de MenuItem in listele aplicatiei. In aceasta metoda avem doar getterele si setterele clasei impreuna cu constructorul clasei.

In clasa Order avem atributele int idClient , int idOrder , int totalPrice , int numberOfProducts si LocalDateTime date pentru a stii data la care a fost realizata comanda. Pe langa gettere settere si constructorul clasei mai avem metoda de generateTotalPrice care ne va calcula valoarea intreaga a comenzii cand este efectuata .

In clasa DeliveryService avem un ArrayList de tip MenuItem in care ne stocam toate produsele , un ArrayList de tipul Client in care ne stocam toti clientii si un HashMap <Order , ArrayList<MenuItem>> in care ne vom stoca toate comenzile ca si chei iar ca valoarea arraylist-urile de produse alea comenzilor . In aceasta clasa se regasec , pe langa gettere settere si constructorul clasei , majoritatea metodelor ce realizeaza functionalitatile aplicatiei.

Metoda invariant ce returneaza true daca este indeplinita conditia ceruta .

Metoda public void generateTimeReport(LocalTime startDate, LocalTime endDate) throws IOException genereaza un raport sub forma unui fisier text cu toate comenzile realizate intre 2 ore trimise ca si parametru si le afiseaza in acest fisier text. Functionalitatea este realizata cu Stream() si lamda.

Metoda public void generateClientsReport(int nrComenzi , int suma) throws IOException genereaza un raport sub forma unui fisier text cu toti clientii ce au realizat mai multe comenzi decat numarul de comenzi trimis ca si parametru si care au macar o comanda ce are valoarea mai mare decat parametrul trimis. Functionalitatea este realizata cu Stream() si lamda.

Metoda public void generateProductsReport(int nrComenzi) throws IOException genereaza un raport sub forma unui fisier text cu toate produsele ce au fost achizitionate pana in momentul de fata de un numar de ori mai mare decat numarul trimis ca si parametru. Functionalitatea este realizata cu Stream() si lamda.

Metoda public void generateDayReport(LocalDateTime date) throws IOException genereaza un raport sub forma unui fisier text cu toate produsele ce au fost cumparate in data trimisa ca si parametru si de cate ori au fost acestea cumparate . Functionalitatea este realizata cu Stream() si lamda.

Metoda public List<MenuItem> searchProductByName(String name) realizeaza cautarea produselor din arraylist ce contin cuvantul trimis ca si parametru si imi creaza o lista cu aceste produse pe care mi-o returneaza. Functionalitatea este realizata cu Stream() si lamda.

Metoda public List<MenuItem> searchProductByRating(float rating) realizeaza cautarea produselor din arraylist ce au rating-ul egal cu cel trimis ca si parametru si imi creaza o lista cu aceste produse pe care mi-o returneaza. Functionalitatea este realizata cu Stream() si lamda.

Metoda public List<MenuItem> searchProductByCalories(int calories) realizeaza cautarea produselor din arraylist ce au numarul de calorii egal cu cel trimis ca si parametru si imi creaza o lista cu aceste produse pe care mi-o returneaza. Functionalitatea este realizata cu Stream() si lamda.

Metoda public List<MenuItem> searchProductByFats(int fats) realizeaza cautarea produselor din arraylist ce au numarul de grasimi egal cu cel trimis ca si parametru si imi creaza o lista cu aceste produse pe care mi-o returneaza. Functionalitatea este realizata cu Stream() si lamda.

Metoda public List<MenuItem> searchProductByProteins(int proteins) realizeaza cautarea produselor din arraylist ce au numarul de proteine egal cu cel trimis ca si parametru si imi creaza o lista cu aceste produse pe care mi-o returneaza. Functionalitatea este realizata cu Stream() si lamda.

Metoda public List<MenuItem> searchProductBySodium(int sodium) realizeaza cautarea produselor din arraylist ce au sodiul egal cu cel trimis ca si parametru si imi creaza o lista cu aceste produse pe care mi-o returneaza. Functionalitatea este realizata cu Stream() si lamda.

Metoda public List<MenuItem> searchProductByPrice(int price) realizeaza cautarea produselor din arraylist ce au pretul egal cu cel trimis ca si parametru si imi creaza o lista cu aceste produse pe care mi-o returneaza. Functionalitatea este realizata cu Stream() si lamda.

Metoda public void getItems() throws FileNotFoundException ce imi deschide fisierul products.csv si imi importa din acesta toate produsele si le introduce in arrayList-ul de menuItems.

Metoda public ArrayList<MenuItem> searchByMultipleCriteria(String name , int price , float rating , int protein ,int sodium , int fat, int calories ) realizeaza cautarea produselor din arraylist ce au anumite criterii egale cu parametrii trimisi si imi creaza o lista cu aceste produse pe care mi-o returneaza. Functionalitatea este realizata cu Stream() si lamda.

Metoda public void deleteItem(String name) realizeaza stergerea unui element din lista de produse ce are numele precizat.

Metoda public MenuItem searchItem(String name) realizeaza cautarea unui anumit produs din lista de produse si mi-l returneaza .

Metoda public void addOrder(Order order, ArrayList<MenuItem> menuItem) throws IOException realizeaza adaugarea unei comenzi noi in sistem.

Metoda public Order searchOrder(int id) realizeaza cautarea unei anumite comenzi dupa id-ul trimis ca si parametru.

Metoda public void modifyProduct(String name,BaseProduct item) modificarea produsului cu numele trimis ca si parametru cu produsul trimis ca si parametru.

Metoda public void generateBill(int id) throws IOException realizeaza scrierea chitantei sub forma unui fisier txt la fiecare comanda efectuata de catre clienti.

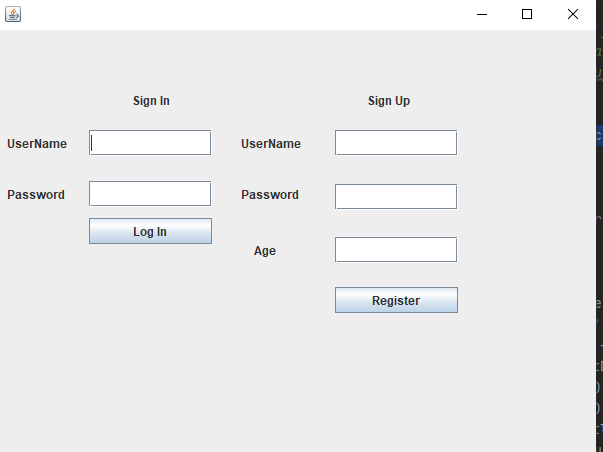
In pachetul DataLayer avem clasa Serializator ce contine doar metode statice ce realizeaza serializarea si deserializatea produselor , clientilor si hashMap-ului cu comenzile efectuate si listele de produse cumparate la fiecare order.

In pachetul PresentationLayer avem clasele de interfata grafica ce contin elementele specifice unei interfete grafice.

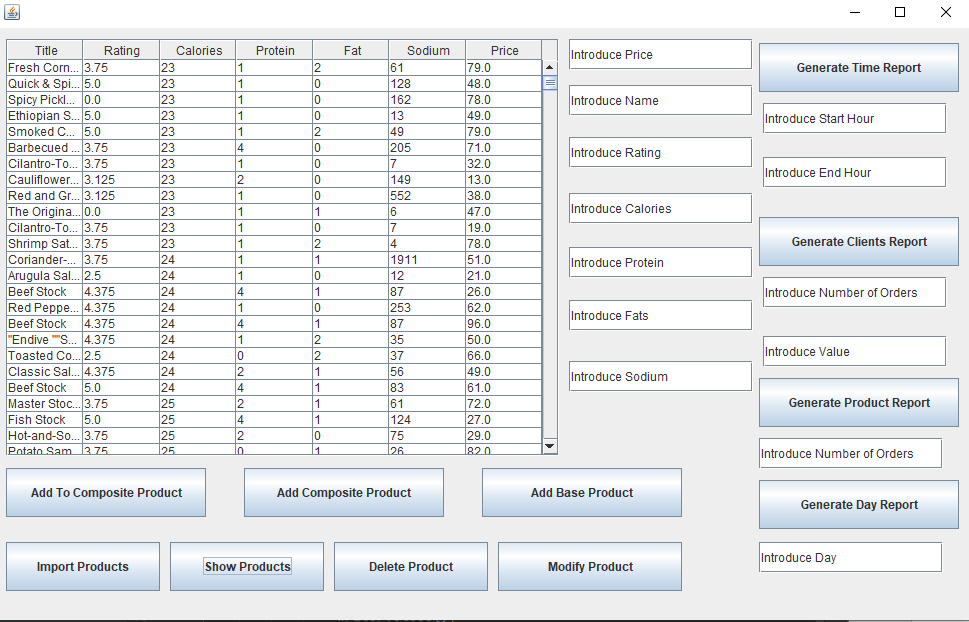
**5.Rezultate**

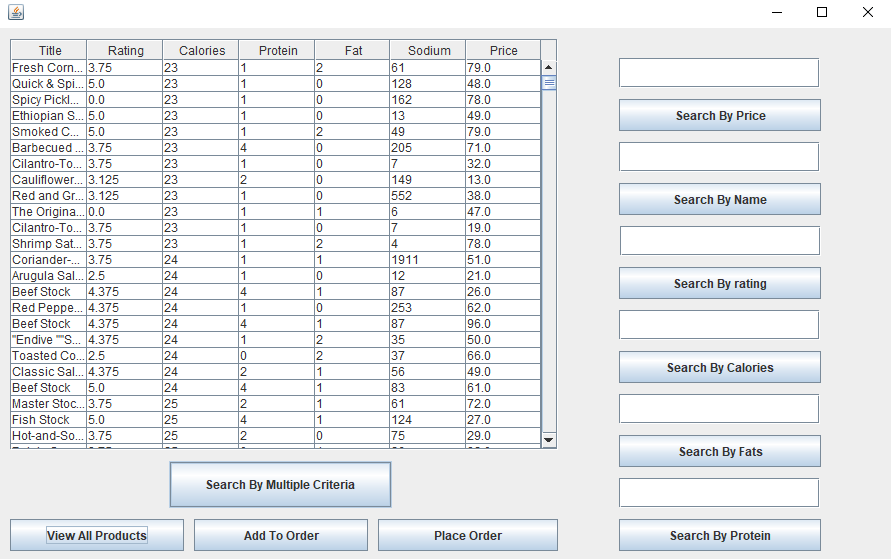
In interfata grafica se pot observa rezultatele aplicatiei si in rapoartele efectuate.

Interfata grafica la pornire:

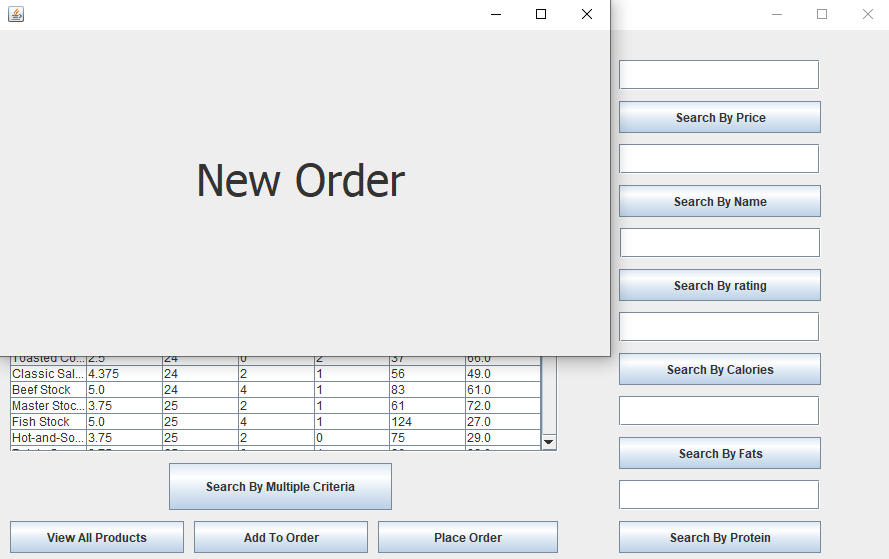


Interfata grafica pentru administrator:

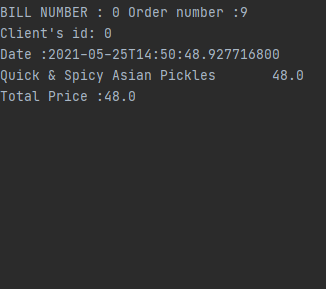


Interfata grafica pentur clienti:  


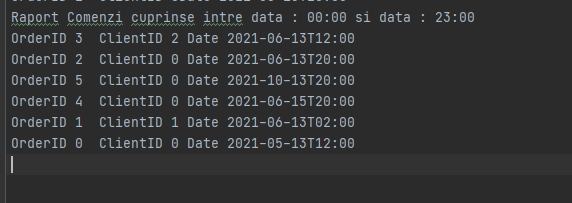
Interfata grafica cand se efectueaza o camanda noua



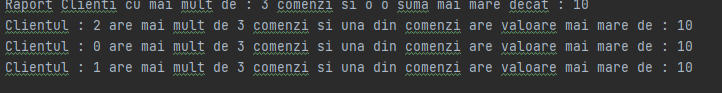
Chitanta pentru o comanda noua :



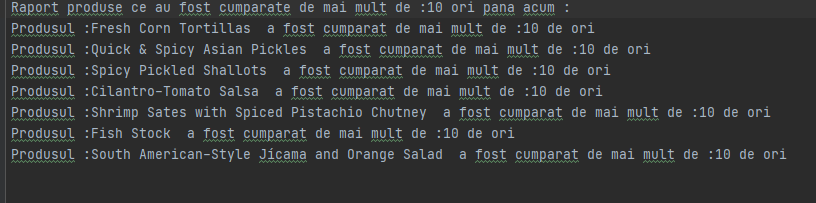
Raport-ul de Timp (inte oradestart si oradefinish):



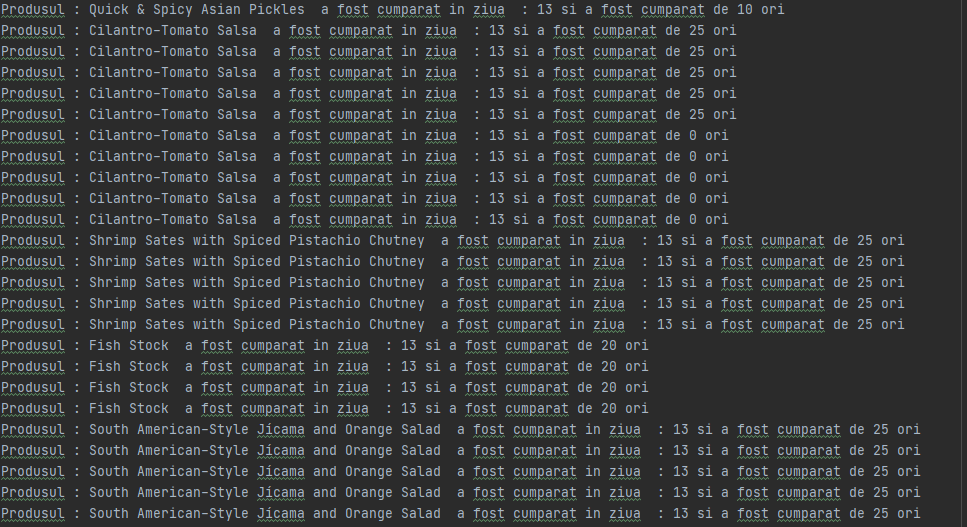
Raport-ul de clienti:



Raport-ul de produse:



Raportul de zi:



**6.Concluzii**

Dupa realizarea acestei teme am invatat cum sa realizez o aplicatie ce retine date fara a le stoca intr-o baza de date , prin serializare si am invatat concepte foarte importante despre folosirea streamurilor in java si a expresiilor de tip lamda , serializare si pre si post conditions.

Dezvoltari ulterioare :

* Adaugarea de acituni suplimentare pentru administrator cum ar fi vizualizarea tuturor comenzilor si a clientilor precum si functionalitati legate de aceste 2 seturi de date pentru a face aplicatia mai performanta si mai actuala;
* Adaugarea de rapoarte cu diferite statistici pentru a putea observa mai usor datele stocate in aplicatie;
* Cautarea dupa anumite informatii legate de produs cum ar fi : proVegan , pentru vegetarieni , intoleranti la lactoza etc;
* Adaugarea de functionalitati pentru angajati si clienti;

**7.Bibliografie**

Lambda expressions and stream processing :

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

o https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html

o https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-

streams.html

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

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

Java serialization :

o http://www.tutorialspoint.com/java/java\_serialization.htm

o https://www.baeldung.com/java-serialization

o https://www.geeksforgeeks.org/serialization-in-java/

o <https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html>

Java HashMap :

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

Java assert :

o http://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html

o http://javarevisited.blogspot.ro/2012/01/what-is-assertion-in-java-java.htmlo http://stackoverflow.com/questions/11415160/how-to-enable-the-java-keywordassert-in-eclipse-program-wise

o https://intellij-support.jetbrains.com/hc/en-us/community/posts/207014815-How-toenable-assert

Adding custom tags to Javadoc

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