MINISTERUL EDUCAȚIEI NAȚIONALE
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

DOCUMENTATIE
FOOD DELIVERY MANAGEMENT SYSTEM

BORZA DIANA-CRISTINA
GRUPA 30225 | AN 2 SEMESTRUL 2

Cuprins

1.Obiectivul temei

1.1.Obiectivul principal al temei

Obiectivul principal al temei este acela de a creea o aplicatie care permite logarea intr-un sistem in functie de tipul utilizatorului. Exista trei tipuri de utilizatori : angajat, client si administrator. Fiecare va avea posibilitati specifice functiei, spre exemplu modificarea produselor, crearea unei comenzi, efectuarea unei comenzi.

1.2.Obiective secundare

Obiectivele secundare reprezinta pasii care trebuie urmati pentru a indeplini obiectivul principal, iar ele sunt definite in cele ce urmeaza.

- Analiza problemei si identificarea cerintelor

Programul trebuie sa contina o interfata de logare, cat si una de creare cont. Odata logat in aplicatie un utilizator va putea efectua diferite operatii in functie de functia sa. Clientul va putea vedea lista de produse si de a cauta in aceasta unele produse bazate pe diverse criterii si de a le selecta pentru a le adauga intr-o comanda. La finalizarea comenzii un bon va fi generat, iar comanda va fi transmisa angajatului. Un angajat va putea vizuakiza comenzile curente si va avea posibilitatea sa le onoreze incepand de la cea cu numarul comenzii minim. Administratorul poate importa lista de produse din fisierul .csv, le poate modifica sau sterge si poate adauga noi produse. De asemenea, el poate creea meniuri create din mai multe produse. Totodata, acesta poate genera si rapoarte specifice datelor din aplicatie. Rapoartele se vor selecta si se vor introduce cerintele specifice, iar rapoartele vor fi generate in fisierele specifice. Acest obiectiv va fi prezentat pe larg in capitolul 2.

- Proiectarea sistemului de management

Intregul sistem va avea ca si intrari un username si o parola pe baza careia se vor diferentia tipurile de utilizatori. Datele despre utilizatori si comenzi vor fi pastrate in fisiere obtinute prin serializare. Interfata trebuie sa fie una usoara de inteles si de utilizat. Obiectivul va fi detaliat in capitolul 3.

- Implementarea sistemului de management

Pentru implementarea aplicatiei se va tine cont de faptul ca exista obiecte de tipul Menu Item. Un Menu Item contine un id, pretul si numele iar toate acestea sunt mostenite de Base Product si Composite Product, insa ele vor avea si campuri specifice. Obiectivul va fi explicat amanuntit in capitolul 4.

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

Cerinta functionala a proiectului este aceea de a implementa un sistem care proceseaza produse si comenzi intr-o firma de catering. Operatiile pe care sistemul trebuie sa le efectueze sunt diferentiate in functie d etrei tipuri de utilizatori :

→ Administratorul poate sa :
- Importe produsele initiale din meniu dintr-un fisier csv
- Gestioneze produsele din meniu : sa modifice produse , sa stearga produse , sa adauge produse sau sa creeze produse compuse (meniuri) din mai multe produse de baza
- Genereze rapoarte (Raport 1 : comenzile efectuate intr-un interval orar specificat facand abstractie de la data ; Raport 2 : produsele comandate mai mult de un numar specificar de ori pana in momentul actual ; Raport 3: clientii care au comandat de un numar mai mare de ori decat un numar specificat iar valoarea comenzii a fost mai mare decat un numar specificat ; Raport 4 : produsele care au fost comandate intr-o zi specificata si de cate ori au fost comandate)

→ Clientul poate sa :
- Se logheze in sistem folosind username-ul si parola
- Vizualizeze lista de produse din meniu
- Caute produse in functie de diferite criterii
- Creeze o comanda care contine mai multe produse – pentru fiecare comanda va fi generata o chitanta in care se va afisa pretul total si toate produsele comandate

→ Angajatul poate sa :
- Vizualizeze toate comenzile aflate in asteptare
- Onoreze comenzi (sa le prepare)

- Fie notificat de fiecare data cand este creata o noua comanda

Cazuri de utilizare:

- → Utilizatorul isi introduce username-ul si parola
  - In caz de success se va deschide o interfata corespunzatoare functiei sale.
  - In caz de esec aplicatia se va inchide
- → Client: apasa pe butonul de confirmare comanda
  - in caz de success se va genera un bon si se va deschide interfata de vizualizare in timp real a comenzilor adaugate pentru angajat.
  - In caz de esec : va aparea un mesaj de eroare (nu a fost selectat niciun produs)
- → Client : selecteaza un criteriu de cautare
  - in caz de success se vor afisa produsele ce indeplinesc conditiile.
  - In caz de esec se va afisa un mesaj (nu au fost introduse valorile) sau va aparea o eroare daca valorile introduse sunt alte caractare decat int-uri
- → Administrator : apasa butonul generare raport si selecteaza din interfata rapoartele dorite
  - in caz de success in fisierele txt s evor afisa rapoartele.
  - In caz de esec se va afisa un mesaj (criteriile nu au fost introduse)

3.Proiectare

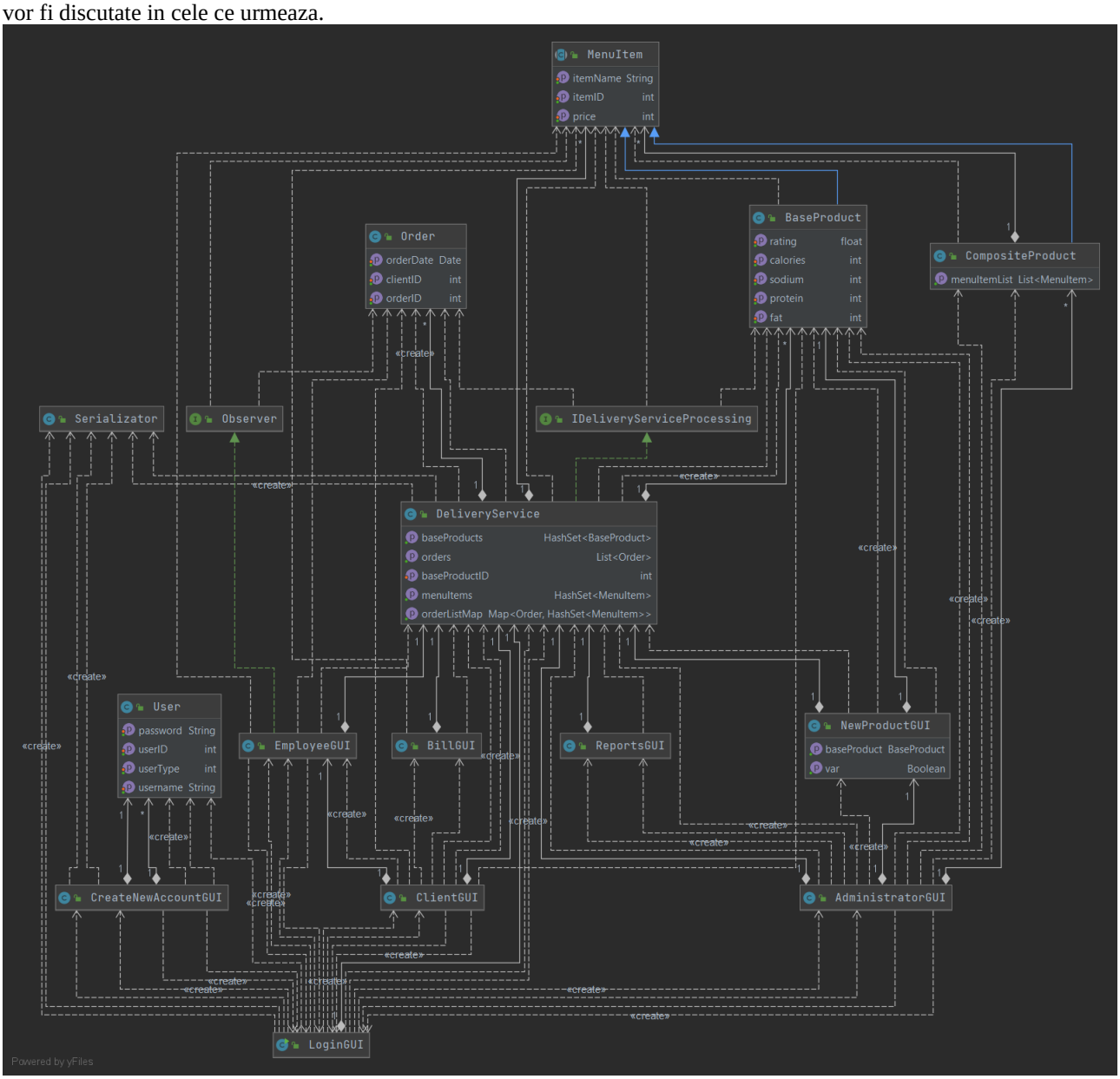
3.1.Decizii de proiectare

Aplicatia va gestiona obiecte de tipul User, Order si MenuItem. Clasa MenuItem este o clasa abstracta iar clasele BaseProduct si CompositeProduct mostenesc aceasta clasa. Un base product este un produs de baza, pe cand un composite product este un “meniu” format din mai multe produse. Ambele sunt produse ale meniului. Clasa Order. In clasa DeliveryService se vor implementa toate metodele corespunzatoare operatiilor efectuate in sistemul de management. Pentru pastrarea datelor la rulari diferite ale aplicatiei se va folosi clasa Serializator unde sunt implementate metodele de serializare si deserializare.

Mai multe ferestre se vor implementa pentru fiecare functionalitate in parte, usor de utilizat de catre un utilizator.

3.2.Diagrama UML

Diagrama specifica aplicatiei este vizibila in imaginea urmatoare, iar componentele acesteia, anume clasele vor fi discutate in cele ce urmeaza.



3.3.Structuri de date

Structurile de date folosite sunt HashSet, HashMap si ArrayList. Am folosit HashSet pentru a stoca setul de produse (base products , menu items) deoarece intr-un set nu putem avea obiecte duplicate iar aceasta conditie era necesara pentru implementarea mea. In ArrayList am pastrat obiecte de tipul Order. In HashMap am pastrat perechile < Order , HashSet<MenuItem> >. Practic, am stocat comenzile intr-o tabela de dispersie unde cheia este calculata pe baza atributelor comenzii iar tabela la cheia respectiva se mapeaza produsele corespunzatoare comenzii.

3.4.Proiectare clase

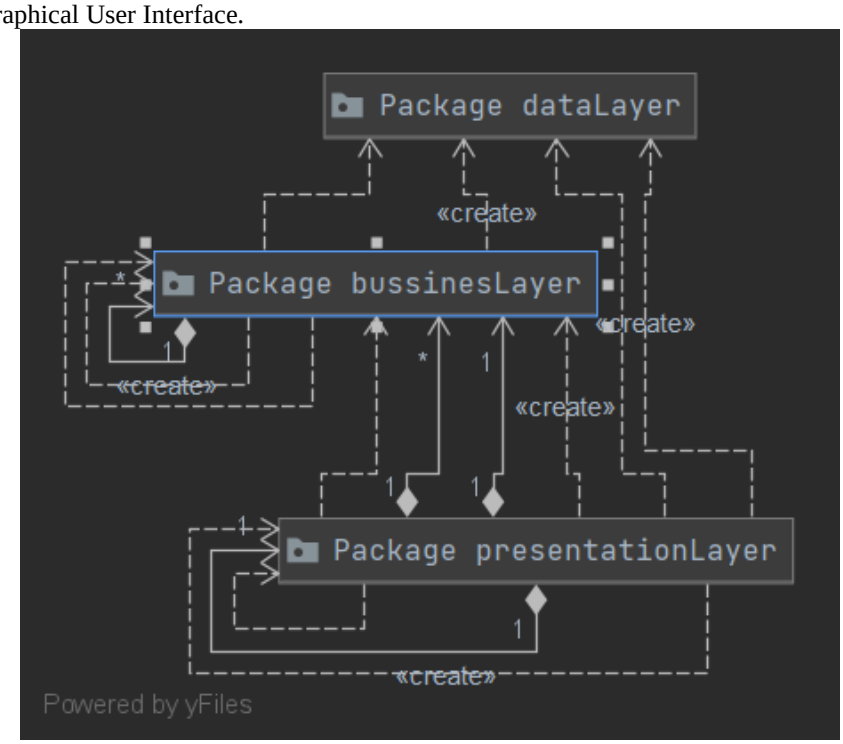
Aplicatia contine cinsprezece clase si doua interfete. Clasele pentru produse sunt : MenuItem , BaseProduct, CompositeProduct. Pentru comanda avem clasa Order iar pentru utilizator clasa User. Metodele pentru efectuarea operatiilor se afla in clasa DeliveryService impreuna cu map-ul corespunzator tuturor comenzilor. Pentru interfata sunt create opt clase, acestea avand la finalul numelui “GUI”.

3.5.Interfete

Aplicatia contine doua interfete, IDeliveryService si Observer. IDeliveryService reprezinta interfata pe care o implementeaza DeliveryService iar Observer interfata pe care o implementeaza fereastra corespunzatoare angajatului pentru notificarea primirii unei noi comenzi. De asemenea in aplicatie sunt folosite si interfetele Serializable si Comparable.

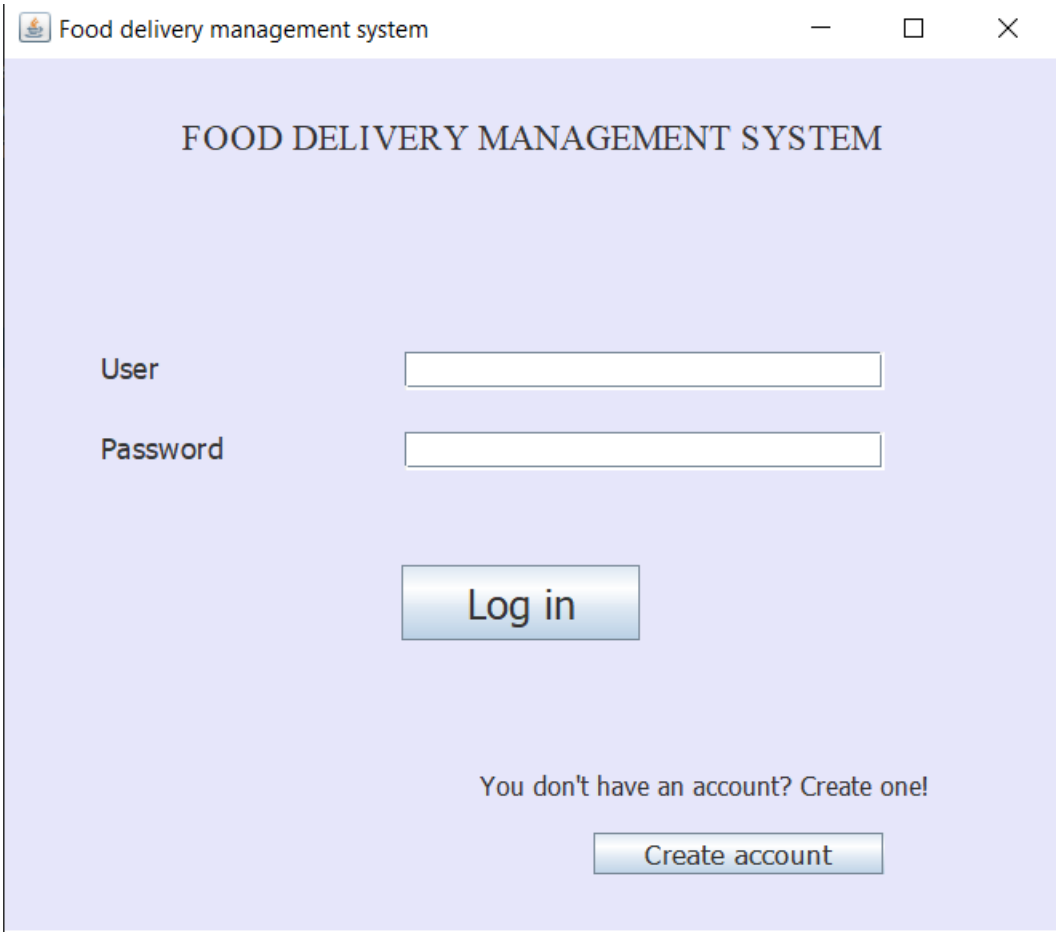
3.6.Pachete

Aplicatia contine trei pachete : bussinesLayer care contine clasele corespunzatoare obiectelor gestionate de interfata, dataLayer care contine clasa Serializator si presentationLayer care contine toate clasele care implementeaza Graphical User Interface.

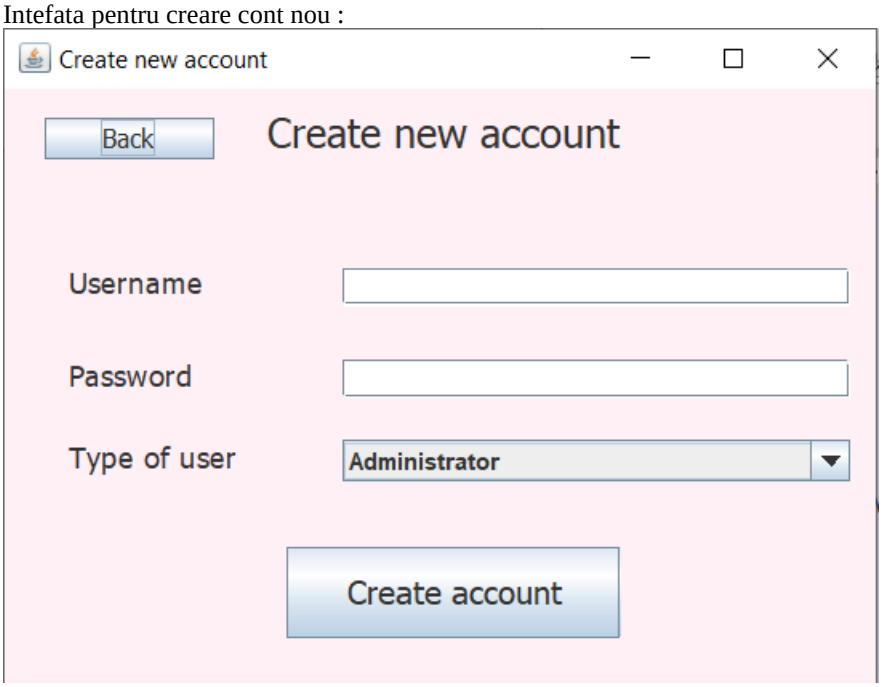


3.7.Interfata utilizator

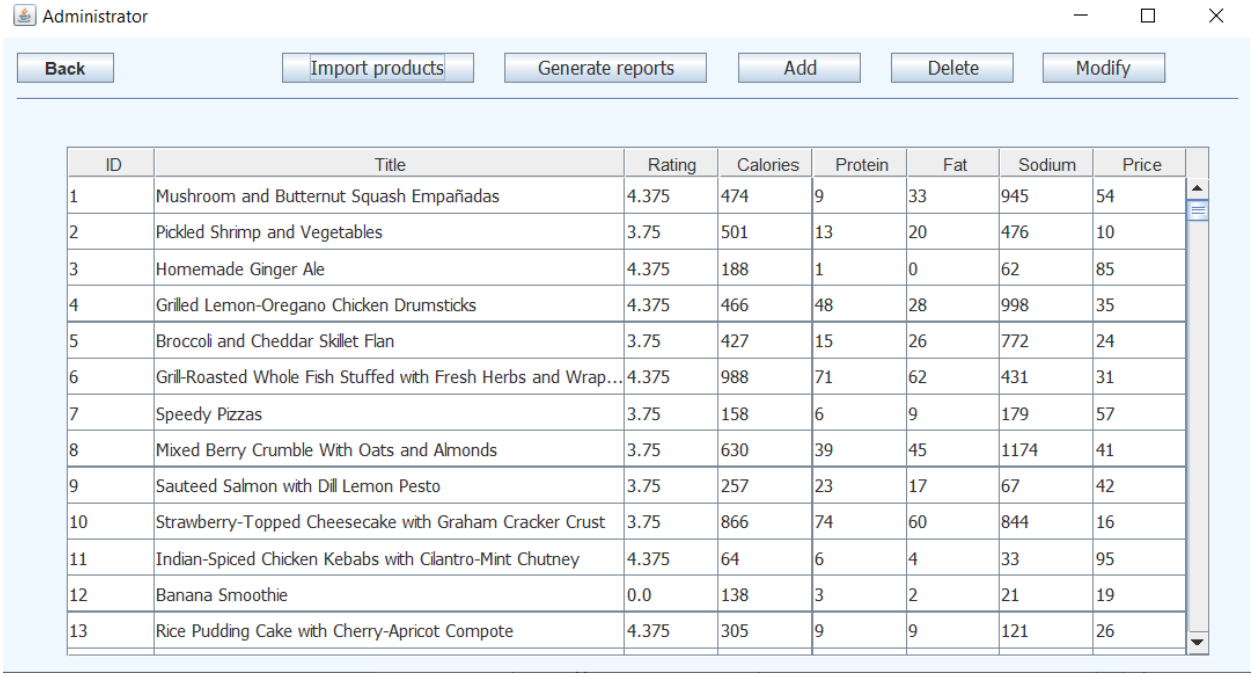
Aplicatia contine mai multe interfete pentru utilizator, toate sugestive si usor de folosit.
Interfata de logare pentru utilizator :



Intefata pentru creare cont nou :

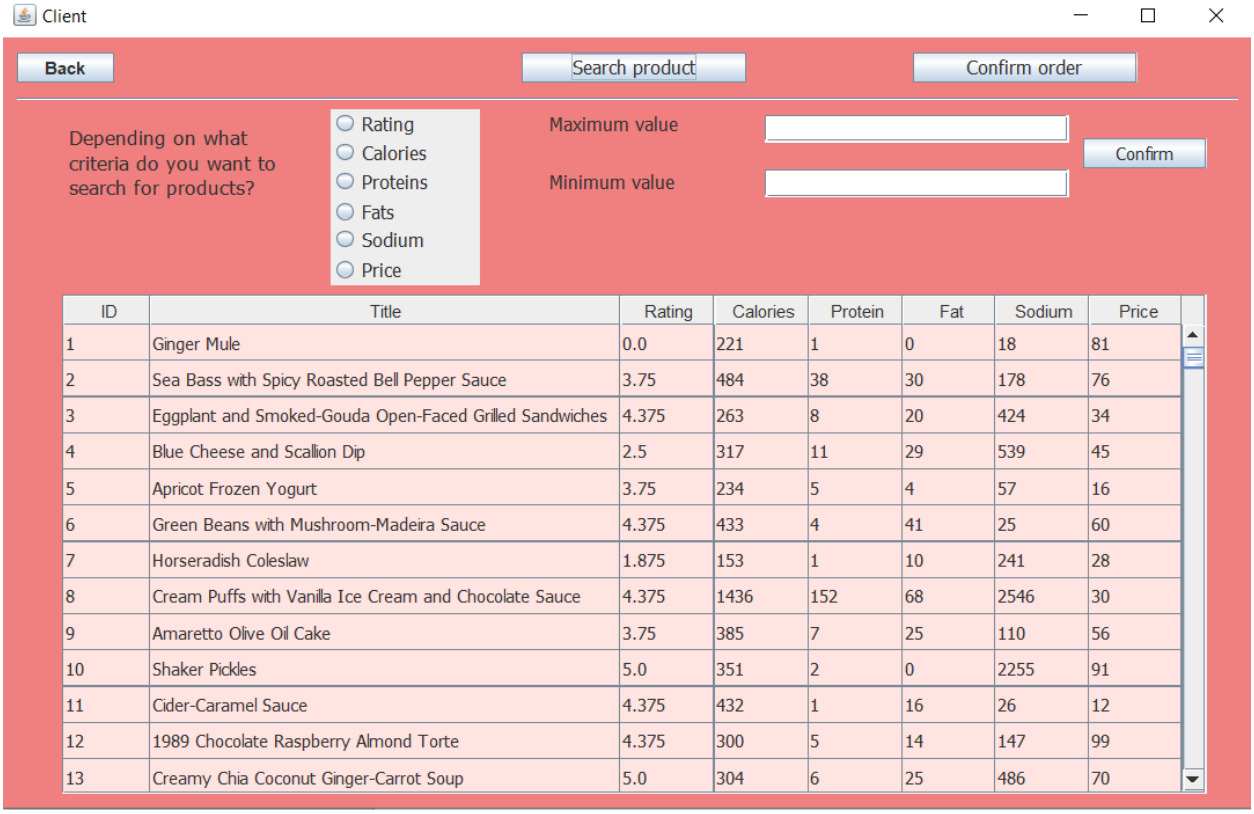


Interfata pentru administrator :



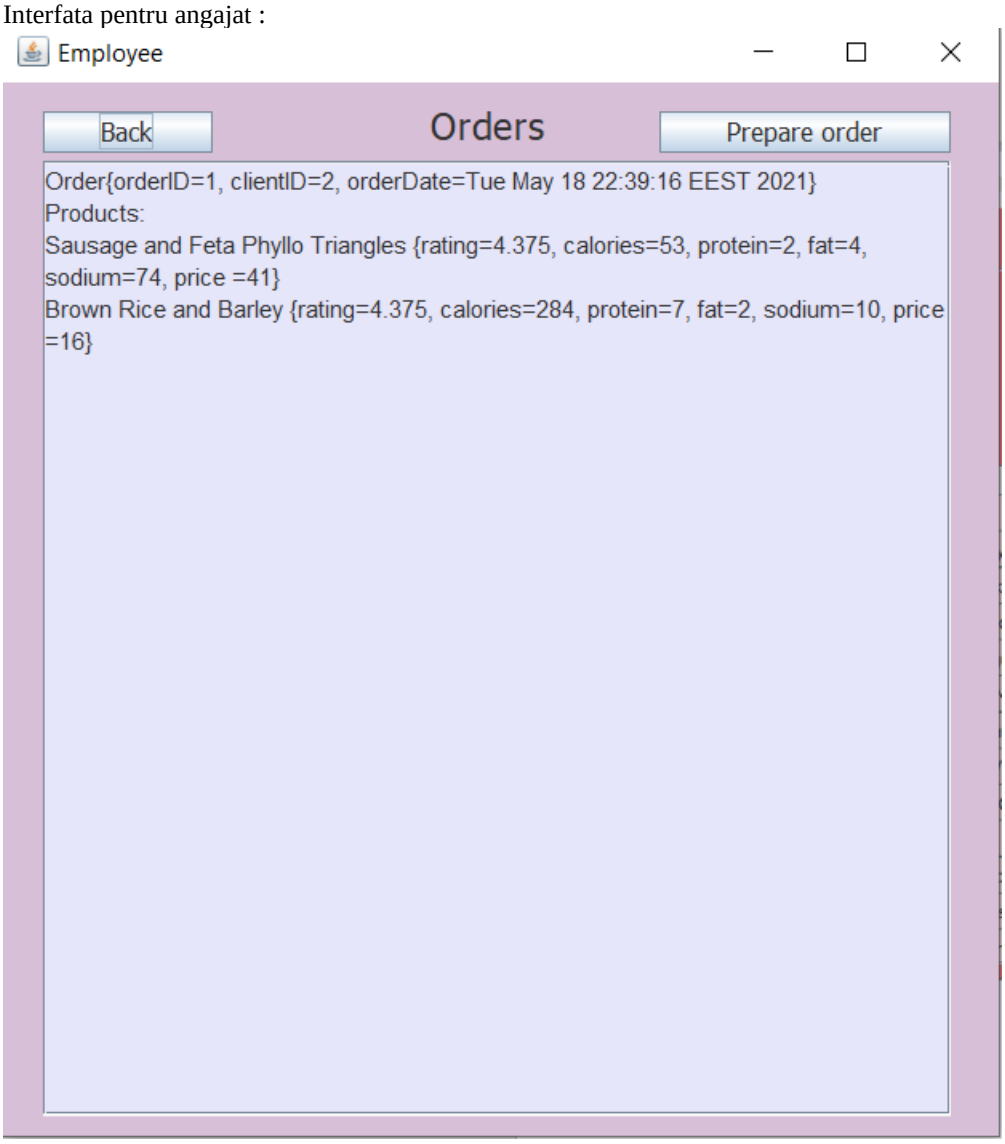
ID	Title	Rating	Calories	Protein	Fat	Sodium	Price
1	Mushroom and Butternut Squash Empañadas	4.375	474	9	33	945	54
2	Pickled Shrimp and Vegetables	3.75	501	13	20	476	10
3	Homemade Ginger Ale	4.375	188	1	0	62	85
4	Grilled Lemon-Oregano Chicken Drumsticks	4.375	466	48	28	998	35
5	Broccoli and Cheddar Skillet Flan	3.75	427	15	26	772	24
6	Grill-Roasted Whole Fish Stuffed with Fresh Herbs and Wrap...	4.375	988	71	62	431	31
7	Speedy Pizzas	3.75	158	6	9	179	57
8	Mixed Berry Crumble With Oats and Almonds	3.75	630	39	45	1174	41
9	Sauteed Salmon with Dill Lemon Pesto	3.75	257	23	17	67	42
10	Strawberry-Topped Cheesecake with Graham Cracker Crust	3.75	866	74	60	844	16
11	Indian-Spiced Chicken Kebabs with Cilantro-Mint Chutney	4.375	64	6	4	33	95
12	Banana Smoothie	0.0	138	3	2	21	19
13	Rice Pudding Cake with Cherry-Apricot Compote	4.375	305	9	9	121	26

Interfata pentru client :



ID	Title	Rating	Calories	Protein	Fat	Sodium	Price
1	Ginger Mule	0.0	221	1	0	18	81
2	Sea Bass with Spicy Roasted Bell Pepper Sauce	3.75	484	38	30	178	76
3	Eggplant and Smoked-Gouda Open-Faced Grilled Sandwiches	4.375	263	8	20	424	34
4	Blue Cheese and Scallion Dip	2.5	317	11	29	539	45
5	Apricot Frozen Yogurt	3.75	234	5	4	57	16
6	Green Beans with Mushroom-Madeira Sauce	4.375	433	4	41	25	60
7	Horseradish Coleslaw	1.875	153	1	10	241	28
8	Cream Puffs with Vanilla Ice Cream and Chocolate Sauce	4.375	1436	152	68	2546	30
9	Amaretto Olive Oil Cake	3.75	385	7	25	110	56
10	Shaker Pickles	5.0	351	2	0	2255	91
11	Cider-Caramel Sauce	4.375	432	1	16	26	12
12	1989 Chocolate Raspberry Almond Torte	4.375	300	5	14	147	99
13	Creamy Chia Coconut Ginger-Carrot Soup	5.0	304	6	25	486	70

Interfata pentru angajat :



4.Implementare

Aplicatia contine 15 clase si 2 intefete structurate astfel :

Clasa **MenuItem** este clasa abstracta cu campurile itemID, itemName si price. Aici sunt definite si functiile pentru set si get a variabilelor si constructorul clasei, dar si o metoda abstracta toString() care va fi implementata de fiecare subclasa si metoda abstracta computePrice(). Clasa implementeaza interfata Serializable.

Clasa **BaseProduct** implementeaza interfata Comparable si suprascrie metoda compareTo in functie de itemID. Contine variabilele de clasa calories, rating, fat, proteins, sodium si implementeaza metode de set si get pentru acestea. Ea extinde clasa MenuItem.

Clasa **CompositeProduct** extinde clasa MenuItem si contine o lista de MenuItem-uri si metoda addItem, dar si metoda computePrice().

Clasa **Order** implementeaza interfata Serializable si defineste un obiect de tipul comanda. Are ca variabile de clasa atributele : orderID, clientID si orderDate si implementeaza metode de get, set si toString. De asemenea se implementeaza si metodele hashCode si equals folositoare maparii in HashMap.

Clasa **User** implementeaza interfata Serializable si defineste un utilizator. Contine variabilele de clasa userID, userType (1-Administrator 2-Client 3-Employee), username si password. Contine metode pentru set si get.

Interfata **IDeliveryService** este interfata in care se implementeaza mai multe metode care se folosesc pentru efectuarea operatiilor necesare rularii conform cerintelor a aplicatiei.

Clasa **DeliveryService** este clasa in care sunt implementate metode pentru diferite functionalitati. Implementeaza interfata IDeliveryService si Serializable. Are ca variabile de clasa baseProducts (lista de produse din csv) , orderListMap (hash table-ul unde se pastreaza comenzile si produsele corespunzatoare) , menuItems (unde se pastreaza produsele adaugate pentru comanda curenta) , orders (unde se pastreaza comenzile) , baseProductID , orderID. Metoda importProducts() pune in baseProducts produsele din csv file (filtrate fara duplicate si dupa nume). Metoda getProductsBasedOnRating() returneaza elementele filtrate in functie de rating. Metoda getProductsBasedOnCalories() returneaza elementele filtrate in functie de calorii. Metoda getProductsBasedOnProteins() returneaza elementele filtrate in functie de proteine. Metoda getProductsBasedOnFats() returneaza elementele filtrate in functie de grasimi. Metoda getProductsBasedOnSodium() returneaza elementele filtrate in functie de sodiu. Metoda getProductsBasedOnPrice() returneaza elementele filtrate in functie de pret. Functia addMenuItems() adauga o lista de produse in menuItems in functie de o lista de id-uri preluate din interfata. Functia addOrder() creeaza o noua comanda si o mapeaza la HashMap. Functia generateBill() efectueaza scrierea bonului. Functia computePrice() returneaza pretul unei comenzi (al unui HashSet de MenuItem). Functia saveInformation() realizeaza serializarea informatiilor din clasa DeliveryService intr-un fisier. Functia deleteProduct() sterge un produs din lista de produse de baza, iar functia addProduct() adauga un nou produs in lista de produse de baza. Functia modifyProduct() modifica un produs din lista de produse de baza. Functia generateReportTimeInterval() genereaza raportul 1 descris la inceput. Functia generateReportClient() genereaza raportul 3. Functia generateReportProductsOrderedMost() genereaza raportul 2, iar functia generateReportProductBasedOnDay() genereaza raportul 4. Functia prepareOrder() sterge comanda cu id-ul cel mai mic (este finalizata).

Clasa **Serializator** implementeaza metodele pentru serializare si deserializare a datelor.

Clasa **LoginGUI** este clasa care implementeaza interfata utililzator pentru logare. Ea detine obiect de tipul DeliveryService ca si variabila a clasei pe care in instantiaza cu valori preluare prin deserializare din fisier.

Clasa **CreateNewAccountGUI** este clasa care implementeaza interfata pentru creare cont nou. Detine un arrayList de User pentru a prelua conturile deja facute prin deserializare si efectueaza serializarea datelor dupa crearea noului cont in fisierul specific.

Clasa **Administrator** este interfata grafica specifica utilzatorului care contine ca si variabila de clasa un obiect de tipul DeliveryService. In aceasta interfata se pot efectua operatiile specifice administratorului : importare produse, generare raport, modificare/ adaugare / stergere produse si creare de meniuri formate dn mai multe produse.

Clasa **NewProductGUI** este clasa care defineste interfata grafica care se va deschide in momentul in care administratorul doreste sa introduca un nou produs .

Clasa **ReportsGUI** este clasa care defineste interfata grafica care se deschide la apasarea butonului Generate Reports din interfata administratorului. Aici se pot selecta rapoartele dorite si se introduc valorile cerute pentru a genera rapoartele in format txt.

Clasa **ClientGUI** este clasa care implementeaza interfata grafica specifica Clientului. Aceasta are ca si variabila de clasa un obiect de tipul DeliveryService si implementeaza operatiile posibile pentru client: vizualizare produse, cautare in functie de anumite criterii si creare comanda. La crearea unei comenzi se va genera o factura in fisierul bill.txt si se va afisa is pe ecran un bon, dar si fereastra angjatului care este notificat de fiecare data cand o comanda este adaugata.

Clasa **BillGUI** este clasa care implementeaza interfata grafica care afiseaza bonul fiscal la apasarea butonului ConfirmOrder din interfata clientului.

Interfata **Observer** este interfata care implemneteaza metoda update() de notificare a unei schimbari.

Clasa **EmployeeGUI** este clasa care implementeaza interfata Observer si implementeaza metoda update() de fiecare data cand o noua comanda este adaugata de catre client. In aceasta interfata se vizualizeaza toate comenzile curente si se finalizeaza apasand butonul penru finalizare incepand cu comanda cu id-ul cel mai mic. Este interfata grafica specifica angajatului.

5.Concluzii

Implementarea programului ilustreaza tehnicile de programare orientate obiect, modul de impartire al claselor, de stabilire a relatiilor, de implementare eficienta si concisa, de pastrare a datelor din diferite procese prin serializare si deserializare, de generare a JavaDoc dar si de implementare a unei interfete accesibila oricarui utilizator. Prin aceasta tema mi-am aprofundat cunostintele in limbajul java si a tehnicilor de programare, cat si a

DesignPattern-urilor care au fost implementate in proiect (Composite Design Pattern, Design by Contract, Observer Design Pattern).

Aplicatia ar putea fi dezvoltata ulterior prin creearea unei interfete mai interactive si prin adaugarea mai multor optiuni pentru tipurile de utilizatori. De asemenea s-ar putea adauga oferte sau reduceri pentru clientii fideli.

6.Bibliografie

https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html#tag
https://objectcomputing.com/resources/publications/sett/september-2011-design-by-contract-in-java-with-google
https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html
https://www.tutorialspoint.com/java/java_serialization.htm
https://en.wikipedia.org/wiki/Observer_pattern
https://en.wikipedia.org/wiki/Composite_pattern