FOOD DELIVERY MANAGEMENT SYSTEM

Documentatie

Nume: Ungurianu Daniel

Grupa 30228 An 2 semestrul al doilea

Disciplina: Tehnici de programare

Profesor indrumator: Cristina Bianca Pop

Cuprins

[1.Obiective 3](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588731)

[1.1 Obiectiv principal 3](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588732)

[1.2 Obiective secundare 3](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588733)

[2.Analiza problemei,modelare,scenarii,cazuri de utilizare 3](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588734)

[2.1 Analiza 3](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588735)

[2.2 Cerințe funcționale 3](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588736)

[2.3 Cerințe non-funcționale 3](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588737)

[2.4 Diagrama Use-Case 4](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588738)

[2.5 Scenarii de funcționare 4](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588739)

[3.Proiectare 5](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588740)

[3.1 Decizii de proiectare 5](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588741)

[3.2 Diagrame de pachete și clase 6](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588742)

[3.2.1. Diagrama de pachete 6](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588743)

[3.2.2.Diagrama de clase 7](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588744)

[3.3 Structuri de date și algoritmi 7](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588746)

[3.4.Interfețe și pachete 7](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588747)

[3.5 Interfața utilizator 8](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588748)

[4.Implementarea 8](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588749)

[4.1 Clasa BaseProduct 8](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588750)

[4.2 Clasa CompositeProduct 8](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588751)

[4.3 Clasa DeliveryService 8](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588752)

[4.4 Clasa Order 12](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588755)

[4.5 Clasa User 12](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588756)

[4.6 Clasa ExistingUserException 12](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588757)

[4.7 Clasa InvalidCredentials 13](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588758)

4.8 Clasa MenuItemNotFound……………………………………………………………………………………………………. 14

4.9 Clasa Role ……………………………………………………………………………………………………………………………. 14

4.10 Clasa Controller…………………………………………………………………………………………….……………………. 14

4.11 Clasa Serializator…………………………………………………………………………………………………………………. 14

4.12 Class Writer…………………………………………………………………………………………………………………………. 14

4.13 Clasa GUI1………………………………………………………………………………………………………………..…………. 14

4.14 Clasa GUI2…………………………………………..……….………………………………………………………..……………. 14

4.15 Clasa GUI3………………………………………………………………………………………………………………………………. 14

4.16 Clasa GUI4………………………………………………………………………………………………………………………………. 14 4.17 Clasa GUI5………………………………………………………………………………………………………………………………. 14

4.18 Clasa Start………………………………………………………………………………………………………………………………. 14

[5.Javadoc 14](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588759)

[6.Concluzii 14](file:///C:\Users\danie\Downloads\Order_manager_VATAFU_30228.docx#_Toc70588760)

**1.Obiective**

**1.1 Obiectiv principal**

Obiectivul principal al acestei teme este proiectarea si implementarea unei aplicatii ce simuleaza un system de management de livrare de alimente/produse pentru o firma de catering.

**1.2 Obiective secundare**

Printre obiectivele secundare ale acestei teme, se enumera:

Analiza problemei si modelarea ce are la baza, cazuri de utilizare, diagrame Use-Case, Proiectarea ce consta in decizii de proiectare, diagrame UML, clase, interfete, structure de date, GUI, algoritmi, packages, implementarea ce se caracterizeaza prin descrierea claselor si a metodelor.

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

**2.1 Analiza**

Pentru ducerea la bun sfarsit a acestei teme si implementarea corespunzatoare a aplicatiei, aceasta ar trebui sa permita inregistrarea a 3 tipuri de users: admin, employee si client, fiecare avand anumite sarcini:

admin:-sa importe seturile de produse

-sa poata realiza operatiile de adaugare,stergere si modificare a produselor si crearea de noi produse prin combinarea altor produse

-generarea de rapoarte legate de comenzile plasate in functie de niste criterii

Client:-sa se inregistreze cu username si parola cu care sa se poata loga

-sa vada produsele din menu

-sa caute produse pe baza mai unor criterii

-sa plaseze o comanda ce consta din mai multe produse

Employee:-va fi notificat de fiecare data cand un client plaseaza o comanda noua pentru a putea pregati livrarea comenzii

**2.2 Cerinte functionale**

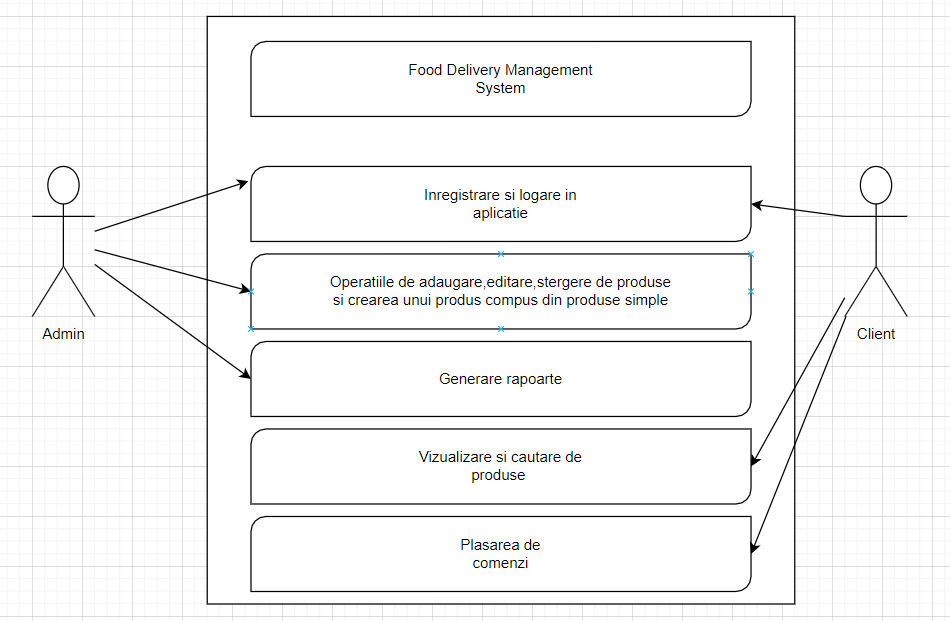
Este necesar ca utilizatorul sa aiba posibilitatea de a:

* Se inregistra folosind un rol, nume si parola
* Se loga cu numele si parola cu care s-a inregistrat
* Efectua operatiile de adaugare,editare si stergere de produse (admin)
* Crea un produs compus din mai multe produse simple (admin)
* Genera rapoarte (admin)
* Importe produsele (admin)
* Cauta un produs dupa niste criterii (client)
* Vizualiza produsele (client)
* Vizualiza bonul in format text

**2.3 Cerinte non-functionale**

Este necesar sa fie implementata o interfata intuitiva si usor de folosit de catre utilizator.

**2.4 Diagrama Use-Case**



Aceasta diagrama use-case evidentiaza principalele facilitate puse la dispozitie user-ului (fie el admin sau client).

**2.5 Scenarii de functionare**

2.5.1) Pentru inceput, user-ul va fi nevoit sa se inregistreze. Mai intai isi va alege rolul cu care doreste sa se logheze, apoi isi va introduce numele si parola.

Pentru fiecare rol ales, exista 2 scenarii posibile:

1. Scenariul cu succes

Datele introduse de user pentru a se inregistra sunt folosite si la logare, deci sunt valide.

Daca produsul cautat de client se afla in lista de alimente, se poate realiza comanda.

Daca produsul cautat pentru editare,stergere sau creare de produs compus exista, atunci operatiile vor fi realizate cu success.

Daca datele introduse pentru generarea de rapoarte sunt valide si exista comenzi care sa respecte cerintele user-ului atunci acestea vor fi afisate.

1. Scenariul esuat

Valorile introduce de user(fie ca e admin sau client sau employee) nu sunt valide,atunci logarea nu se poate realiza, asadar, nici operatiile corespunzatoare fiecarui tip de user nu vor putea fi realizate si se va genera o eroare cand se incearca logarea.

**3. Proiectare**

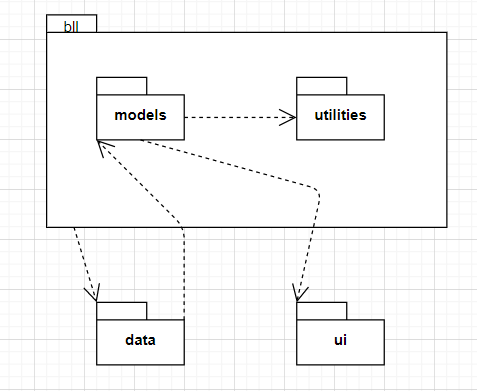
**3.1 Decizii de proiectare**

Pentur realizarea sistemului de management al serviciului de livrare de alimente la o firma de catering a fost necesara folosirea serializarii(care are rolul de a transforma un obiect java in format binar si scris intr-un fisier .txt), a deserializarii(ce presupune preluarea datelor din acel fisier .txt si transformarea lor inapoi in obiecte java) si a modului de implementare Composit Design pentru a crea posibilitatea de a putea realiza produse compuse de tip meniu, ce sunt formate din mai multe produse simple.

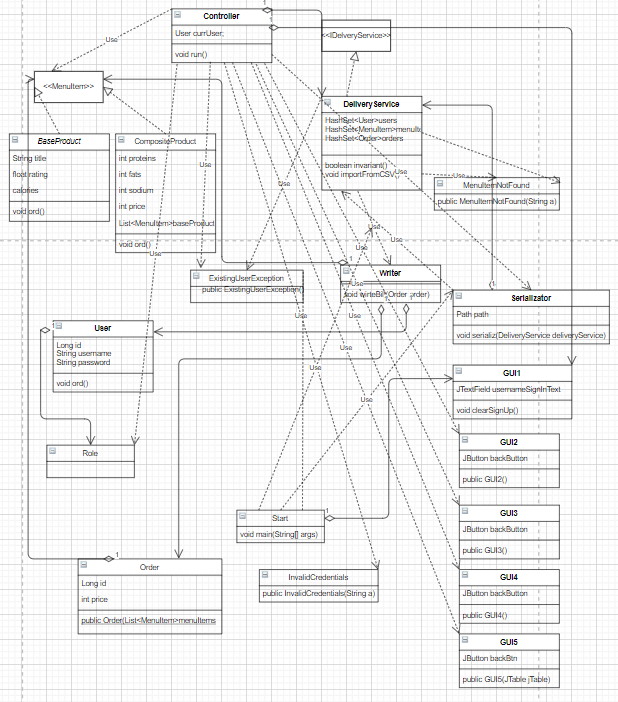
Inainte ca user-ul sa se poata loga sau de a se putea realiza vreo operatie , aplicatia are un system de validare a datelor ExistingUserException, InvalidCredentials, MenuItemNotFound care asigura user-ul ca nu va fi lasat sa se inregistreze daca a facut-o deja, sa introduca date eronate la incercarea de a se loga sau de a cauta un produs ce nu exista in lista de produse.

**3.2 Diagrame de pachete si clase**

**3.2.1 Diagrama de pachete**



**3.2.2 Diagrama de clase**



**3.3 Structuri de date si algoritmi**

In realizarea acestei aplicatii, am folosit liste (ce mi-au fost folositoare pentru stocarea menu items-urilor), HashSet-urilor pentru stocarea userilor, menuItems urilor si comenzilor(sunt niste structuri de date foarte practice intrucat elimina duplicatele automat).

**3.4 Interfete si pachete**

Principalele interfete folosite sunt Serializable, IDeliveryService,MenuItem.

Pentru a avea posibilitatea de a lucra cu fisiere, am avut nevoie de java.io.FileWriter.

**3.5 Interfata utilizator**

Pentru interfata grafica am folosit un design simplu, usor de intuit.

Utilizatorului i se pun la dispozitie o serie de campuri, ce permin introducerea datelor, si butoane ce ofera posibilitatea realizarii operatiilor folosind datele introduce, cu ajutorul carora se vor crea comenzi, se vor adauga/sterge produse, se vor crea produse compuse.

**4. Implementarea**

**4.1 Clasa BaseProduct**

Mosteneste clasa MenuItem si contine date ce descriu un produs simplu din lista de produse.

In aceasta clasa am folosit un stil nou de implementare a getterelor, setterelor, metodei equals din Comparable si metoda de calculare a hashCode-ului ce consta in adnotari. A fost posibil acest lucru apeland la lombok.

**4.2 Clasa CompositeProduct**

Mosteneste clasa MenuItem si contine date ce descriu un produs simplu din lista de produse. In plus fata de clasa BaseProduct, aceasta clasa mai contine un field: o lista de BaseProduct-uri ce constituie un produs compus. Am folosit asadar Composite Design.

In aceasta clasa am folosit un stil nou de implementare a getterelor, setterelor, metodei equals din Comparable si metoda de calculare a hashCode-ului ce permite gasirea elementelor stocate in acel hashset. A fost posibil acest lucru apeland la lombok.

**4.3 Clasa DeliveryService**

Aceasta clasa este una de baza in realizarea temei intrucat sunt implementate toate operatiile posibil realizabile de catre oricare tip de user (inregistrarea si logarea fiind necesare in prima instanta, inainte ca celelalte operatii sa poata fi realizate), fie el admin (importarea produselor, operatii cu produsele: adaugare, editare, stergere, creare de produse compuse, generarea rapoartelor pentru comenzi) sau client (cautare de produse, posibilitatea de a le vizualiza, plasarea de comenzi).

Se observa si aparitia invariantului cat si a utilizarii claselor folosite cu scopul validarii datelor introduse sau a operatiilor incercate de user.

Pentru stocarea de useri, menuItems si comenzi au fost folosite hashSet-uri.

In aceasta clasa am folosit un stil nou de implementare a getterelor, setterelor, a unui constructor fara parametri. A fost posibil acest lucru apeland la lombok.

@NoArgsConstructor  
@Setter  
@Getter  
public class DeliveryService implements IDeliveryService{  
 private HashSet<User> users = new HashSet<>();  
 private HashSet<MenuItem> menuItems = new HashSet<>();  
 private HashSet<Order> orders = new HashSet<>();  
  
  
 public boolean invariant()  
 {  
 return menuItems != null;  
 }  
  
 public void importFromCSV()  
 {  
 Path path = Paths.*get*("src\\main\\resources\\products.csv");  
 try (Stream<String> lines = Files.*lines*(path.toAbsolutePath()).skip(1)) {  
 List<List<String>> values = lines.map(line -> Arrays.*asList*(line.split(","))).collect(Collectors.*toList*());  
 for (List<String> value : values) {  
 menuItems.add(new BaseProduct(value.get(0), Float.*parseFloat*(value.get(1)), Integer.*parseInt*(value.get(2)), Integer.*parseInt*(value.get(3)),  
 Integer.*parseInt*(value.get(4)), Integer.*parseInt*(value.get(5)), Integer.*parseInt*(value.get(6))));  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }

public GUI5 showAll()  
{  
 String[] top = {"Title","Rating","Calories","Proteins","Fats","Sodium","Price"};  
 Object[][] content = new Object[menuItems.size()][top.length];  
  
 int i=0;  
 for(MenuItem j:menuItems)  
 {  
 content[i][0] = j.getTitle();  
 content[i][1] = j.getRating();  
 content[i][2] = j.getCalories();  
 content[i][3] = j.getProteins();  
 content[i][4] = j.getFats();  
 content[i][5] = j.getSodium();  
 content[i][6] = j.getPrice();  
  
 i++;  
 }  
  
 GUI5 gui5 = new GUI5(new JTable(content,top));  
 assert gui5 != null;  
  
 return gui5;  
}

**4.4 Clasa Order**

Aceasta clasa are ca field-uri id(pentru numerotarea si diferentierea comenzilor), data(field ce ne va ajuta la generarea unor rapoarte), pret(field ce ma va ajuta la generarea unor rapoarte si intocmirea bonului) si o lista de menuItems intrucat intr-o comanda pot fi incluse mai multe produse.

In aceasta clasa am folosit un stil nou de implementare a getterelor, setterelor, metodei equals din Comparable si metoda de calculare a hashCode-ului ce permite gasirea elementelor stocate in acel hashset, a unui constructor fara parametri. A fost posibil acest lucru apeland la lombok.

**4.5 Clasa User**

In componenta acestei clase intra la socoteala field-urile ce tin de construirea userilor: username si parola, necesare la logare la care se mai adauga rolul(admin, client sau employee) necesar inregistrarii acestuia.

La acestea se mai adauga un field: o lista de comenzi in care se vor stoca toate comenzile plasate de un client.

In aceasta clasa am folosit un stil nou de implementare a getterelor, setterelor, metodei equals din Comparable si metoda de calculare a hashCode-ului ce permite gasirea elementelor stocate in acel hashset, a unui constructor fara parametri. A fost posibil acest lucru apeland la lombok.

**4.6 Clasa ExistingUserException**

Aceasta clasa este folosita pentru validarea datelor introduse de user in fereastra de inregistrare in modul urmator: daca datele introduse de user in fereastra de inregistrare au mai fost folosite inainte pentru o alta inregistrare, atunci se va genera o exceptie.

**4.7 Clasa InvalidCredentials**

Aceasta clasa este folosita pentru validarea datelor introduse de user in fereastra de logare in modul urmator: daca datele introduse de user in fereastra de inregistrare sunt diferite fata de cele introduse in fereastra de logare, atunci se va genera o exceptie.

**4.8 Clasa MenuItemNotFound**

Aceasta clasa este folosita pentru validarea datelor introduse de user in fereastra clientului atunci cand incearca sa caute un produs in lista sau in fereastra adminului atunci cand incearca sa faca o operatie de stergere, editare sau creare de produs compus in modul urmator: daca datele introduse de user in fereastra adminului legate de numele produsului nu se regasesc in lista de produse, se va genera o exceptie; daca datele introduse de user in fereastra clientului legate de numele produsului nu se regasesc in lista de produse, se va genera o exceptie.

**4.9 Clasa Role**

Aceasta clasa este o enumeratie de 3 elemente( admin, client si employee) si va ajuta la diferentierea userilor in momentul in care acestia incearca sa isi faca inregistrarea.

**4.10 Clasa Controller**

Aceasta clasa este cea mai stufoasa din acest proiect intrucat contine toti ascultatorii pentru fiecare buton atat din fereastra dedicata adminului cat si cea a clientului.

Este asadar o clasa de o importanta majora in proiect cat si pentru ca aici se realizeaza serializarea datelor.

**4.11 Clasa Serializator**

In aceasta clasa sunt implementate operatiile de serializare si deserializare ce constau in transformarea obiectelor java in date cu format binar si scrierea lor intr-un fisier .txt, respectiv in transformarea datelor cu format binar preluate din acel fisier .txt in obiecte java.

public static void serializ(DeliveryService deliveryService)  
{  
 try {  
 FileOutputStream fileOut =  
 new FileOutputStream(*path*.toAbsolutePath().toString());  
 ObjectOutputStream out = new ObjectOutputStream(fileOut);  
 out.writeObject(deliveryService);  
 out.close();  
 fileOut.close();  
 } catch (IOException i) {  
 i.printStackTrace();  
 }  
}

public static DeliveryService deserializ()  
 {  
 DeliveryService deliveryService = null;  
  
 try {  
 FileInputStream fileIn = new FileInputStream(*path*.toAbsolutePath().toString());  
 ObjectInputStream in = new ObjectInputStream(fileIn);  
 deliveryService = (DeliveryService) in.readObject();  
 in.close();  
 fileIn.close();  
 } catch (IOException i) {  
 i.printStackTrace();  
 return null;  
 } catch (ClassNotFoundException c) {  
 System.*out*.println("Class not found");  
 c.printStackTrace();  
 return null;  
 }  
  
 return deliveryService;  
 }  
}

**4.12 Clasa Writer**

Aceasta clasa se ocupa cu generarea bonului cat si afisarea rapoartelor in alte fisiere .txt.

public static void wirteBill(Order order)  
{  
 Path path = Paths.*get*("src\\main\\resources\\bill.txt");  
 try {  
 FileWriter myWriter = new FileWriter(path.toAbsolutePath().toString());  
 myWriter.write("ID: "+order.getId()+" \n");  
 myWriter.write("Date: "+order.getDate()+"\n");  
 myWriter.write("Price: "+order.getPrice());  
 myWriter.close();  
 } catch (IOException e) {  
 System.*out*.println("An error occurred.");  
 e.printStackTrace();  
 }  
}

**4.13 Clasa GUI1**

Se ocupa de generarea ferestrei de logare si inregistrare.

**4.14 Clasa GUI2**

Aceasta clasa genereaza fereastra corespunzatoare administratorului ce contine toate operatiile pe care acesta le poate realiza.

**4.15 Clasa GUI3**

Aceasta clasa genereaza fereasta corespunzatoare clientului ce contine toate operatiile pe care acesta le poate realiza.

**4.16 Clasa GUI4**

Aceasta clasa genereaza fereastra corespunzatoare employee-ului.

**4.17 Clasa GUI5**

Aceasta clasa se ocupa de fereastra pentru afisarea produselor pentru client.

**4.18 Clasa Start**

Aceasta clasa include metoda main ce da start aplicatiei.

**5. Javadoc**

Pentru aceasta tema am generat documentatia Javadoc. Acest lucru a fost posibil cu ajutorul unei structuri de comentarii speciale.

**6. Concluzii**

Realizarea acestei teme m-a pus in dificultate intrucat am avut de implementat concepte noi: serializare si deserializare, composite design, invariant, iar pentru ca acest lucru sa devina posibil, a fost nevoie de documentare.

**7. Surse de documentare**

-lambda expressions and stream processing

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

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

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

-custom tags in javadoc

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

-composite design pattern

<https://www.baeldung.com/java-composite-pattern>

-<https://users.utcluj.ro/~igiosan/teaching_pc.html>

- prezentarile propuse