Text

Description automatically generated

DOCUMENTAȚIE

TEMA 4-Food Delivery Service App

Buruian Cătălina

Grupa 30221

# 

**CUPRINS**

1. [Obiectivul temei 2](#_Toc97418559)

1. [Analiza problemei, modelare, scenarii, cazuri de utilizare 2](#_Toc97418560)
2. [Proiectare 2](#_Toc97418561)
3. [Implementare 2](#_Toc97418562)
4. [Rezultate 2](#_Toc97418563)
5. [Concluzii 2](#_Toc97418564)
6. [Bibliografie 2](#_Toc97418565)

**Capitolul 1:Obiectivul temei**

Obiectivul principal

Obiectivul principal al acestei teme a constat in implementarea unei aplicatii de Food Delivery Service care sa permita logarea a trei tipuri de utilizatori (admin, client, angajat) cu username si parola. Utilizatorii de tip admin pot gestiona produsele din meniu si pot genera rapoarte, utilizatorii de tip client pot filtra produsele din meniu si pot adauga o comanda, iar utilizatorii de tip angajat vor fi notificati cand se inregistreaza o comanda noua.

Obiective secundare

1. Respectarea conceptelor de programare OOP
2. Implementarea diagramei de clase prezentate
3. Citirea din products.csv a produselor din meniu
4. Realizarea unei interfețe grafice pentru fiecare tip de utilizator
5. Utilizarea de lambda expressions și stream processing pentru implementarea funcționalităților
6. Serializarea datelor aplicației
7. Generarea de javaDocs
8. Realizarea unei facturi pentru fiecare comanda

**Capitolul 2:Analiza problemei, modelare, scenarii, cazuri de utilizare**

Cerinte functionale:

1. Aplicatia de Food Delivery Service ar trebui sa permita utilizatorilor sa introduca datele de conectare (username-ul, parola) in fereastra de login, sau in cazul in care utilizatorul nu are cont sa isi creeze unul
2. Aplicatia ar trebui sa deschida fereastra specifica tipului de utilizator
3. Utilizatorul ar trebui sa poata efectua diferite actiuni in functie de rolul sau (adminul sa adauge, stearga sau modifice produse din meniu, sa creeze un produs compus sau sa genereze rapoarte bazate pe anumite criterii, clientul sa poata filtra produsele, iar angajatul sa poata vedea in timp real cand o comanda a fost procesata de sistem).

Use case: crearea unui cont

Actorul principal: Utilizator

Principalul scenariu de succes:

1. Un utilizator de orice tip apasa pe butonul specific crearii unui nou cont
2. Utilizatorul adauga informatii legate de nume, prenume, adresa, email, parola, username, cat si tipul de utilizator cu care isi creeaza un nou cont.
3. Utilizatorul apasa pe butonul de „Creare cont”, iar pe ecran apare o fereastra cu un mesaj de succes in cazul in care datele au fost introduse corect

Scenariu alternativ: Date invalide

* Utilizatorul introduce valori invalide pentru numar de telefon, email
* Aplicatia afiseaza un mesaj de eroare si cere utilizatorului sa introduca valori valide

Use case1: logarea ca admin

Actorul principal: Adminul

Principalul scenariu de succes:

1. Un utilizator de tip admin se logheaza cu username si parola
2. Utilizatorul apasa pe butonul de „Conectare” pentru a se deschide pagina specifica adminului
3. Adminul are posibilitatea de a vedea meniul, de a realiza modificari asupra lui si de a genera rapoarte

Scenariu alternativ: Utilizatorul nu exista

* Username-ul si parola nu corespund unui cont existent
* Aplicatia afiseaza un mesaj de eroare si sugereaza utilizatorului sa isi creeze un cont

Scenariu alternativ: Parola introdusa gresit

* Utilizatorul introduce valori invalide pentru parola
* Aplicatia afiseaza un mesaj de eroare si cere utilizatorului sa incerce alta parola
* Scenariul se intoarce la pasul 1.

Use case2: logarea drept client

Actorul principal: Clientul

Principalul scenariu de succes:

1. Un utilizator de tip client se logheaza cu username si parola
2. Utilizatorul apasa pe butonul de „Conectare” pentru a se deschide pagina specifica clientului
3. Clientul are posibilitatea de a vedea meniul, de a cauta produse in functie de niste filtre legate de valori minime si maxime sau keywords, de adauga/sterge produse in/din cos si de a efectua o comanda.

Scenariu alternativ: Utilizatorul nu exista

* Username-ul si parola nu corespund unui cont existent
* Aplicatia afiseaza un mesaj de eroare si sugereaza utilizatorului sa isi creeze un cont

Scenariu alternativ: Parola introdusa gresit

* Utilizatorul introduce valori invalide pentru parola
* Aplicatia afiseaza un mesaj de eroare si cere utilizatorului sa incerce alta parola
* Scenariul se intoarce la pasul 1.

Use case3: logarea drept angajat

Actorul principal: Angajatul

Principalul scenariu de succes:

1. Un utilizator de tip angajat se logheaza cu username si parola
2. Utilizatorul apasa pe butonul de „Conectare” pentru a se deschide pagina specifica angajatului
3. Angajatul are posibilitatea de a vedea in timp real cum o comanda este realizata de catre un client (si fereastra client trebuie sa ramana deschisa).

Scenariu alternativ: Utilizatorul nu exista

* Username-ul si parola nu corespund unui cont existent
* Aplicatia afiseaza un mesaj de eroare si sugereaza utilizatorului sa isi creeze un cont

Scenariu alternativ: Parola introdusa gresit

* Utilizatorul introduce valori invalide pentru parola
* Aplicatia afiseaza un mesaj de eroare si cere utilizatorului sa incerce alta parola
* Scenariul se intoarce la pasul 1.

**Capitolul 3: Proiectarea**

Design-ul arhitectural al aplicatiei respecta Composite Design Pattern. Acest aspect se reflecta in special la definirea claselor specifice produselor din meniu: MenuItem, BaseProduct respectiv CompositeProduct.

De asemenea, pentru a notifica angajatul când o nouă comandă este plasată am utilizat Observer Design Pattern.

Aplicația este structurată pe pachete, astfel:

* business
  + Admin
  + BaseProduct
  + Bill
  + Client
  + CompositeProduct
  + DeliveryService
  + Employee
  + IDeliveryProcessingService
  + MenuItem
  + Order
  + ROLE
  + User
* data
  + FileWriter
  + Serializator
* presentation
* AdminController
* AdminView
* ClientController
* ClientView
* EmployeeController
* EmployeeView
* LoginController
* LoginPage
* RegisterView
* RegisterViewController
* ReportController
* ReportView
* TableModel
* Start
* Main

Pachetul **business** conține: clasele și interfața ce reprezintă produsele: Clasa MenuItem și clasele BaseProduct și CompositeProduct – ce definesc tipurile de produse care pot sa existe (simple sau compuse din mai multe produse simple, ambele mostenesc clasa MenuItem si cu proprietatile sale, CompositeProduct avand si alte atribute suplimentare), DeliveryService și IDeliveryProcessingService, care modelează operațiile de adăugare, modificare, ștergere a produselor din meniu și de creare a rapoartelor; clasa Order care modelează comenzile; enumeratie ROLE pentru cele trei tipuri de utilizator: client, admin, angajat; clasele Admin, Client, Employee care implementeaza clasa User si ii mostenesc atributele; clasa Bill care defineste o chitanta.

Pachetul **data** cuprinde clasele FileWriter pentru crearea și scrierea în fișiere a rapoartelor, respectiv Serialization pentru serializarea și deserializarea datelor din aplicație: produse, utilizatori și comenzi.

Pachetul **presentation** cuprinde controllere si interfete grafice pentru fiecare fereastra a aplicatiei: de login, de înregistrare, pentru administrator, angajat, client, raport.

Pachetul **start** cuprinde clasa Main care permite rularea aplicatiei.

**Diagrama UML de clase**

Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora ( numite și obiecte ). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte, pentru business Process Design etc.

Graphical user interface, application

Description automatically generated

**Structuri de date folosite**

**Liste**

O lista reprezinta o colectie ordonata de elemente. Intr-o lista sunt permise elementele duplicate. Permite un control mult mai precis asupra pozitiei unui element prin intermediul indexului acelui element.

Interfata List descrie liste de elemente indexate si are disponibile implementari precum cea utilizata in cadrul acestei teme, si anume ArrayList.

ArrayList: permite acces aleator la elementele listei; obtinearea oricarui element din lista se va face in timp constant; in schimb, adaugarea sau stergerea unui element in interiorul listei necesita deplasarea elementelor care se gasesc dupa pozitia la care se executa operatia.

**HashMap<K,V>**

HashMap<K, V> face parte din colecția Java începând cu Java 1.2. Această clasă se găsește în pachetul java.util. Acesta oferă implementarea de bază a interfeței Map din Java. Stochează datele în perechi (cheie, valoare) și le puteți accesa printr-un index de alt tip (de exemplu, un număr întreg). Un obiect este folosit ca cheie (index) pentru un alt obiect (valoare). Dacă încercați să introduceți cheia duplicată, aceasta va înlocui elementul cheii corespunzătoare.

HashMap este similar cu HashTable, dar este nesincronizat. Permite și stocarea cheilor nule, dar ar trebui să existe un singur obiect cheie nulă și poate exista orice număr de valori nule. Această clasă nu oferă garanții cu privire la ordinea map-ului. Pentru a utiliza această clasă și metodele sale, trebuie să importați pachetul java.util.HashMap sau superclasa acestuia.

**Capitolul 4: Implementarea**

Descrierea claselor

- Clasa MenuItem conține signaturile metodelor ce trebuie implementate de clasele ce implementează MenuItem, BaseProduct și CompositeProduct, și anume settere, gettere și metoda computePrice(). Clasa mai implementează Serializable, pentru a putea serializa respectiv deserializa produsele din aplicație.

- Clasa BaseProduct mosteneste atributele si metodele clasei abstracte MenuItem si suprascrie metodele de tip get implementate de aceasta. Aceasta modelează produsul de bază din aplicația noastră, care este definit de următoarele atribute: title, rating, calories, proteins, fats, sodium și price. Produsele de bază sunt citite din products.csv sau pot fi adăugate manual de către administratorul aplicației.

- Clasa CompositeProduct mosteneste atributele si metodele clasei abstracte MenuItem si suprascrie metodele de tip get implementate de aceasta, si in plus mai contine si o lista de produse de baza. Aceasta modelează produsul compus din mai multe produse de bază, și este definită de un titlu, rating, respectiv de un ArrayList de MenuItems.

- Clasa Order care implementează Serializable, și modelează o comandă, definită prin orderId, clientId, data la care s-a plasat comanda și lista de produse comandate. Clasa conține settere și gettere și un constructor, care are ca parametri toate atributele comenzii.

- Interfața IDeliveryProcessingService conține signaturile metodelor pentru importarea produselor: chooseFile() care alege ori din fisierul csv produsele ori din cel creat prin serializare, pentru adăugarea respectiv ștergerea unui produs din meniu: addProduct(), removeProduct(), pentru editarea unui produs: modifyProduct(), pentru fiecare raport în parte, pentru adăugarea unei noi comenzi, respectiv pentru filtrare produse.

- Clasa DeliveryService extinde clasa Observable și implementează IDeliveryProcessingService. Aceasta contine implementarea efectiva e metodelor din interfata:

void addCompositeProduct(List<String> titles, String newTitle, float newRating) throws IOException;  
  
void addProduct(BaseProduct baseProduct) throws IOException;  
  
void removeProduct(BaseProduct baseProduct) throws IOException;  
  
void modifyProduct(BaseProduct baseProduct, String newTitle, float newPrice, float newCalories, float newProteins, float newSodium, float newFats, float newRating) throws IOException;  
  
HashMap<String, MenuItem> filterProducts(String title, float minPrice, float maxPrice, float minRating, float maxRating, float minCalories, float maxCalories, float minFat, float maxFat, float minSodium, float maxSodium, float minProtein, float maxProtein);  
  
void createOrder(Order order) throws IOException, ClassNotFoundException;  
  
void reportOne(int startHour, int endHour) throws IOException, ClassNotFoundException;  
  
void reportTwo(int times) throws IOException, ClassNotFoundException;  
  
void reportThree(int minNumber, int minValue);  
  
void reportFour(int year, int month, int day);

* + reportOne – generează un raport privind produsele comandate într-un interval orar.
  + reportTwo – generează un raport privind produsele comandate de un număr mai mare sau egal de ori decât un număr dat.
  + reportThree – generează un raport privind clienții care au comandat de un număr mai mare de ori decât un număr dat, și plasat comenzi cu o valoare mai mare decât o valoare dată.
  + reportFour – generează un raport privind produsele și de câte ori au fost comandate într-o anumită zi.
  + Pentru rapoarte se folosesc lambda expressions.
  + filterProducts - care filtrează un HashMap<String, MenuItem>, care reprezinta produsele din meniu, în funcție de titlu, respectiv atributele date ca parametru functiei. Se folosesc din nou expresii lambda și stream processing.
  + createOrder -care creeaza comanda, genereaza chitanta si notifica angajatul ca s-a efectuat o comanda (sunt apelate metodele de setChanged() și notifyObservers()).
  + addProduct, modifyProduct, removeProduct, addCompositeProduct – modifica direct HashMap-ul cu meniul initial si la fiecare modificare acesta este serializat si deserializat.

- Clasa FileWriter conține o metodă de citire din fișier si de punere a continutului intr-o lista (putInList) si o metoda care transforma lista in HashMap (getItemHashMap)

- Clasa Serialization conține metodele de serializare/deserializare pentru fisierele cu produse, clienti, comenzi.

- Clasele Administrator/Client/Employee care extind clasa User, modelează utilizatori și atributele acestora, username, parolă, nume, adresă, numar de telefon, email și rol.

- Enum-ul Role conține rolurile pe care le pot avea utilizatorii: Administrator, Client, Employee.

- De asemenea sunt implementate clase care descriu GUI pentru fiecare fereastra si controllere asociate.

Descrierea GUI

Fereastra de Login: utilizatorul se poate loga folosind un username si o parola valide.

In cazul in care utilizatorul nu dispune de un cont, acesta are optiunea de a crea un nou cont, apasand butonul „Create Account”.

Graphical user interface

Description automatically generated

Fereastra de Register: utilizatorul se poate inregistra completand campurile de nume, prenume, adresa, email, numar de telefon, username, parola cu date valide si alegand rolul sau (client, admin, angajat).

Apasand butonul „Create Account” va aparea un mesaj de succes care va anunta utilizatorul ca a fost creat contul.

Graphical user interface, table

Description automatically generated

Fereastra de Admin:

Contine butoane pentru operatiile pe care le poate efectua un admin: adaugare produs, stergere, editare produs, adaugare produs simplu, creare produs compus, generare rapoarte. In textFieldurile de mai jos apar datele despre produsele selectate din tabel, sau adminul poate alege sa le completeze astfel incat sa creeze un nou produs simplu sau compus.

Table

Description automatically generated

Fereastra de Client:

Clientul poate filtra produse, adauga produse in cos, sterge un produs din cos sau goli cosul. Prin apasarea butonului Create Order acesta plaseaza comanda formata din produsele din cos.

Graphical user interface, application

Description automatically generated

Fereastra Employee:

Dupa logarea utilizatorului de tip employee, se va deschide fereastra cu notificari legate de o comanda realizata de un client. Aceasta notificare se realizeaza in timp real, astfel incat ambele ferestre, de client si employee trebuie sa fie deschise simultan pentru a se observa cum a fost efectuata comanda.

Graphical user interface, text, application, email

Description automatically generated

Fereastra pentru Reports:

Aceasta se deschide la apasarea butonului Generate Reports de catre admin. Adminul are optiunea de a genera 4 tipuri diferite de rapoarte in functie de criterii legate de: intervalul orar in care au fost procesate comenzi, produsele comandate de mai mult de un numar specificat de ori, clientii care au comandat produse de mai mult de un nr specificat de ori si cu valoarea produselor mai mare de un anumit nr, produsele comandate intr-o anumita zi specificata si numarul de ori in care au fost comandate.

Graphical user interface

Description automatically generated

**Capitolul 5: Rezultate**

Mai jos sunt prezentate cateva dintre functionalitatile aplicatiei:

Procesul de creare a unui CompositeProduct prin adaugarea mai multor produse simple si denumirea acestuia si atribuire unui rating:

Table

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated

Filtrarea produselor de catre client:

Graphical user interface, application

Description automatically generated

Adaugarea produselor in cos:

Graphical user interface, application

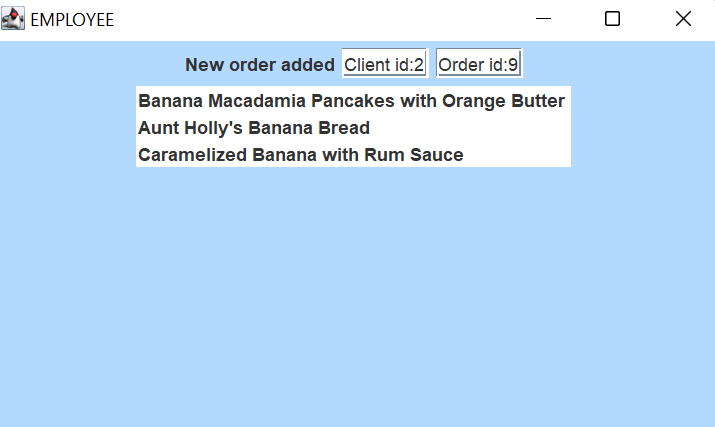
Description automatically generated

Crearea unei comenzi:

Graphical user interface, application

Description automatically generated

Notificarea angajatului ca o comanda a fost inregistrata:



Chitanta generata:

Text

Description automatically generated

Raport cu produsele comandate in intervalul orar 01:00-02:00:

Report 1: time interval of the orders:[1,2]  
  
OrderID: 3  
ClientID: 2  
Products: Savory Streusel , Chicken with Coconut Curry Sauce , Red Lion , Date: 2022-05-23T01:56:47.273491800  
  
OrderID: 7  
ClientID: 5  
Products: Butter-Sugar Crepes , Sweet Potato Macaroni and Cheese , Bacon and Cashew Caramel Corn , Chicken Burgers with Lemon and Tarragon , Date: 2022-05-23T02:00:11.064171800  
  
OrderID: 5  
ClientID: 2  
Products: Sweet and Sour Pickles , Date: 2022-05-23T01:58:41.938938100  
  
OrderID: 6  
ClientID: 5  
Products: Butter-Sugar Crepes , Sweet Potato Macaroni and Cheese , Bacon and Cashew Caramel Corn , Date: 2022-05-23T01:59:47.246010  
  
OrderID: 1  
ClientID: 2  
Products: Brandy Alexander II , Crudites with Asian-Style Dip , Napa Cabbage Salad with Buttermilk Dressing , Date: 2022-05-23T01:56:05.097026300  
  
OrderID: 4  
ClientID: 2  
Products: Savory Streusel , Date: 2022-05-23T01:57:20.470541500  
  
OrderID: 2  
ClientID: 2  
Products: Savory Streusel , Sautéed Chicken Breasts With Country Ham and Sage Sauce , Date: 2022-05-23T01:56:32.605902100

**Capitolul 6: Concluzii**

In concluzie, prin rezolvarea acestei teme am ajuns sa ma familiarizez cu HashMaps, am invatat cum sa folosesc filtrarea, streamurile, si in principal am descoperit ce inseamna serializarea si deserializarea fisierelor si la ce sunt utile acestea. De asemenea,

am acumulat cunoștințe noi despre Composit Design Pattern și Observer Design Pattern și consider că mi-am îmbunătățit modul de scriere și organizare a codului. O dezvoltare ulterioară ar putea fi ca clientul sa primeasca pe mail chitanta si un mail de confirmare a order-ului.

**Capitolul 7: Bibliografie**

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

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

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

<https://www.javatpoint.com/java-jlist>

<https://dsrl.eu/courses/pt/materials/PT2021-2022_Assignment_4.pdf>

<https://dsrl.eu/courses/pt/materials/A4_Support_Presentation.pdf>