DOCUMENTATIE

TEMA *4*

NUME STUDENT: Alixandrescu Octavian

GRUPA: 30223

# 

# CUPRINS

[1. Cerințe funcționale 3](#_Toc95297885)

[2. Obiectivul temei 4](#_Toc95297885)

[3. Analiza problemei, modelare, scenarii, cazuri de utilizare 5](#_Toc95297886)

[4. Proiectare 5](#_Toc95297887)

[5. Implementare 7](#_Toc95297888)

[6. Testare 10](#_Toc95297887)

[7. Concluzii 10](#_Toc95297890)

[8. Bibliografie 10](#_Toc95297891)

# Cerințe funcționale

Să se implementeze un sistem de management și livrare al mâncării pentru o companie de catering. Un client poate comanda produse din meniul companiei. Sistemul ar trebui a aibă trei tipuri de useri ce se pot loga folosind un username și o parolă: administrator, angajat obișnuit și client.

Administratorul va putea să:

- Importe un set inițial de produse care să populeze meniul, dintr-un fișier .csv.

- Managementul produselor din meniu: adăugare, ștergere, modificare și creare de noi produse compuse din mai multe produse (de exemplu “daily menu 1” compus din trei produse deja existente în meniu).

- Genereze rapoarte despre comenzile făcute pe următoarele criterii:

• Intervalul de timp al comenzilor – un raport ar trebui generat în care să apară comenzile intre ora de start și ora de finish

• Produsele comandate mai mult decât un număr specificat de ori

• Clienții care au comandat de mai multe ori decât un număr specificat, iar suma unei comenzi a fost mai mare decât o suma specificata

• Produsele comandate într-o zi cu numărul de dați de cate au fost comandate

Clientul va putea să :

- Să se înregistreze în platforma că ulterior să se poată loga în sistem.

- Să vadă o lista cu produsele din meniu.

- Să caute produsele bazate pe criterii multiple: titlu, rating, număr de calorii, proteine, grăsimi, sodium, preț.

- Să creeze o comanda formata din mai multe produse – pentru fiecare comanda, data și ora o să apară într-un bon care vă și genera și lista de produse comandate și prețul total al comenzii.

Angajatul este notificat de fiecare data când o comanda noua este făcută de un client că să o poată pregăti produsele pentru livrare.

Pentru a implementa sistemul, considerați următoarele:

1. Definiți interfață IdeliveryServiceProcessing care să conțină principalele operații ce pot fi executate de administrator și de client:

- Administrator: importarea de produse, managementul produselor din meniu, generarea rapoartelor

- Client: crearea de noi comenzi care să genereze prețul total al unei comenzi și generarea unui bill în format .txt, căutarea de produse după anumite criterii.

2. Definiți și implementați clase precum cele din diagrama prezentata mai sus:

- Folosirea Composite Design Pattern pentru a defini clasele MenuItem, BaseProduct și CompositeProduct.

- Folosirea Observer Design Pattern pentru a notifica angajații mereu când o noua comanda este creata.

3. Implementați clasa DeliveryService folosind elemente din colecția JCF care folosesc structuri de tipul hashtable. Key-ul pentru hashtable o să fie generat pe baza clasei Order care are asociata mai mule MenuItems

- Definiți o structura de tipul Map> pentru a stoca informațiile comenzii pe baza clasei DeliveryService. Key-ul mapului o să fie format din obiecte de tipul Order, pentru hacare metoda de hashCode() o să fie suprascrisa că să poată fi computata cu valorile atributelor din clasa Order (OrderID, date, etc).

- Definiți o structura de tipul Collection care să salveze meniul (toate produsele) oferite de firma de catering. Alegeți colecția potrivita pentru implementarea aleasa.

- Definiți o metoda de tipul “well formed” pentru clasa DeliveryService.

- Implementați clasa folosind tehnica Design by Contract (involving pre și post condiții).

4. Produsele de baza folosite inițial pentru a popula obiectul de tipul DeliveryService vor fi încărcate din products.csv folosind lambda expression și stream processing. Administratorul poate adaugă manual produse de asemenea.

5. Produsele de tipul MenuItem, comenzile făcute și informațiile utilizatorilor vor fi salvate folosind serializarea că să fie ulterior valabile în sistem pentru viitoare execuții folosind deserializarea.

# Obiectivul temei

2.1 Obiectivul principal

Crearea unei aplicații prin care orice utilizator își poate crea un cont și ulterior să se logheze în aplicație și să comande produse după bunul plac. Prin aplicație administratorul poate să insereze, să șteargă și modifice produse în funcție de preferință și să genereze raporturi prin care să verifice frecventa comenzilor.

2.2 Obiective secundare

Pentru a ajunge la obiectivul principal trebuie să fie efectuați o serie de pasi pentru că utilizatorul să poată beneficia de calculatorul de polinoame.

Dezvoltarea de ușe case-uri. – Prezentate în capitolul 3.

Alegerea structurilor de date – Prezentate în capitolul 4.1.

Împărțirea pe clase – Prezentate în capitolul 4.2.

Dezvoltarea algoritmilor – Prezentate în capitolul 4.3.

Implementarea soluției – Prezentata în capitolul 5.

# Analiza problemei

Un caz de use case ar fi acela când un client își face cont pe platforma companiei de catering de pe care își poate cumpără diverse produse / meniuri. Pentru a putea fi înregistrat clientul, acesta trebuie să introducă un username și o parola validă. Tot odată acesta poate vizualiza toate produsele odată ce s-a deschis magazinul (administratorul da import la produsele din documentul products.csv). Acesta mai are opțiunea de a caută produse după preferințele proprii și are la dispoziție șapte criterii după care poate să efectueze căutarea: nume, rating, calorii, proteine, sodium, grăsimi și preț. În funcție de datele introduse de acesta o să îi fie returnata o listă de produse care îndeplinesc cerințele introduse de acesta. Nu în ultimul rand, clientul poate selecta produsele dorite și după să creeze comanda după bunul plac.

Diagram

Description automatically generated

# Proiectare

4.1 Alegerea structurilor de date

Una din structurile de date folosite este HashMap-ul, structura ce extinde HashTable-ul. Implementarea de map oferă și anumite metode extra care ne permit o funcționare corectă a programului. Implementarea de HashMap oferă o performanță constantă în timp pentru operațiile de baza (put și get), asumându-ne că funcția de hash distribuie elementele uniform. Pentru programul scris, HashMapul conține pe câmpul de key Orderurile, iar pe câmpul de value, setul de MenuItems care conține produsele comandate de fiecare client. Am ales structura de HashMap pentru a putea accesa ușor comenzile și produsele selectate și pentru a le afișă atunci când e nevoie.

4.2

Diagrama UML de mai jos este generată după ce proiectul a fost finalizat și se pot observa dependențele dintre clase și se poate deduce funcționalitatea lor si totodată se observă o claritate a structurării codului.

A screenshot of a computer game

Description automatically generated with medium confidence

4.3 Dezvoltarea algoritmilor

Pentru a implementa cu succes programul am folosit o serie de tehnici de programare menite să ajute în implementare și funcționalitate.

Prima tehnica folosita a fost cea a serializarii și deserializarii care se ocupa de preluarea datelor și trasnmiterea acestora într-un fișier în format binar, putând fi luate ulterior de acolo și prelucrate atunci când este nevoie. Am folosit acest procedeu pentru a puea stoca produsele inițiale, comenzile și ușerii pentru logare. Avantajul acestui procedeu este ocuparea a mai putina memorie pentru a stoca toate aceste date și pentru a avea acces la ele.

Următoarea tehnica a fost cea de lucru cu lambda expressions și stream uri pentru a evita blocurile mari de cod și pentru a eficientiza structurarea și preluarea datelor. Pe acestea le-am folosit pentru importarea produselor și pentru reporturile ce trebuiau generate de către administrator la momentele de timp alese de acesta.

Am folosit tehnica Design by Contract pentru a verifica preconditile și postconditiile din interfață IdeliveryServiceProcessing. Acestea asigura că fiecare metoda primește parametri doriți la început și returnează ceea ce ar trebui, pentru a se evita outputurile nedorite.

Încă o tehnica folosita este cea de Observable care trimite notificări către angajați de fiecare data când un client efectuează o comanda noua. Aceasta tehnica transmite o notificare către interfață angajatului în timp real prin intermediul metodelor setChanged() și notifyObservers().

# Implementare

Pentru implementare am folosit 3 pachete care o să fie descrise în cele ce urmează în detaliu pentru a se evita orice ambiguitate cu privire la implementarea soluției.

Pachetul bussinesslayer

În acest pachet se implementează tot ce tine de partea de preluare, prelucrare și implementare a datelor pentru comenzile magazinului. În primul rand folosind Composite Design Pattern am modelat clasele MenuItem, BaseProduct și CompositeProduct. MenuItem fiind super clasa, aceasta conține toate metodele și atributele necesare pentru cele doua clase care o moștenesc. Totodată, aceasta conține și o metoda abstracta computePrice() care calculează prețul în funcție de comanda și de produsele selectate.Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Interfața IdeliveryServiceProcessing conține toate metodele de care avem nevoie pentru implementare, metode ce se găsesc în clasa DeliveryService despre care vom discuta în cele ce urmează.

Metoda csvProducts() importă produsele din products.csv și le serializează în fișierul SerializedProducts.txt de unde le preia mai departe, la momentul dorit, prin deserializare.

Metoda insertMenuItem primește că parametru un MenuItem care să fie adăugat în lista de produse. Acesta poate fi fie Base Product fie Composite Product pentru a putea fi adăugat. Fiecare produs este caracterizat prin șapte caracteristici: titlu, rating, calorii, proteine, grăsimi, sodium și preț. Pentru implementarea metodei de insert am deserializat fișierul într-un set de Menu Items, am adăugat produsul în set, după care am serializat înapoi setul, produsul nou fiind adăugat cu succes în set.

Metoda deleteMenuItem primește că parametru un MenuItem care urmează să fie șters din lista de produse. Aceasta comanda că și cea de insert și cea de modify (despre care urmează să discutam) pot fi executate doar de către administratorul magazinului.

Metoda de modifyMenuItem primește opt parametri și anume: titlul produsul existent, noul titlu, noul rating, noul preț, noua cantitate de calorii, proteine, grăsimi, sodium. Aceasta metoda caută produsul în lista deja importata de către utilizator și daca găsește produsul cu titlul respectiv îl modifica.

Metoda de searchMenuItem primește cei șapte parametri specifici fiecărui produs, mai exact ratingul minim după care să fie afișate produsele și maximul pentru rating, calorii, proteine, grăsimi, sodium și preț. Metoda aceasta returnează un set de produse care îndeplinesc ceintele date de către client și permite căutarea după unul, doua sau mai multe filtre, după alegerea clientului.

Metoda de createOrder primește că parametri un set de MenuItems (produse) și un clientID că să cunoască cine este clientul care a efectuat comanda respectiva și tot prin serializare și deserializare aceasta se încarca în fișier, respectiv program.

Ultimele patru metode din interfață sunt cele pentru reporturi și anume:

Pentru reportul 1 avem ca și parametri intervalul orar între care se efectuează comenzile și anume starting hour și finish hour. Prin lambda expressions și stream processing se verifica că orele comenzilor să fie în intervalul respectiv și se adaugă în fișierul destinat raportului 1.

Pentru raportul 2 avem că parametru un număr întreg care reprezintă treshold-ul pentru a vedea ce produse au fost comandate de cel puțin atâtea ori cât este specificat. Pentru motoda aceasta am preluat produsele folosind lambda expressions și stream processing, după care am verificat frecventa produselor.

Pentru raportul 3 avem doi parametri care să verifice numărul de clienți care au executat un anumit număr de comenzi de o anumită suma per comanda. Tot folosind lambda expressions și stream processing am preluat clienții după care am verificat pentru fiecare client numărul de comenzi, respectiv suma fiecărei comenzi, ulterior fiind afișate în fișier.

Pentru reportul 3 se verifica daca comenzile au fost efectuate în ziua în care a fost specificata că și parametru, rezultatele ulterioare fiind transmise în fișier.

Pachetul datalayer

În acest pachet se află metodele se serialiazare și deserializare și o enumerație prin în care se afla datele Client, Administrator și Employee.

Pachetul presentationlayer

Aici se afla tot ce tine de interfață grafica și de controlul acesteia, metodele asignate butoanelor și preluarea parametrilor conform text fieldurilor asociate acestora.

# Testare

Am testat mai multe insert-uri, search-uri, modify-uri și delete-uri pentru a verifica corectitudinea programului. Din testele făcute am constat că nu exista bug uri nedescoperite.

# Concluzii

Ca și conluzii pot afirma că am învățat multe lucruri, că mi-am lărgit aria de lucru și am reușit să explorez și alte probleme. Am reușit să învăț cum se lucrează cu serializare și deserializare de date, am învățat să lucrez cu lambda expressions și stream processing, am folosit pattern-uri noi și nu numai.

Posibile dezvoltări ulterioare ale proiectului ar fi extinderea lui pe partea de aplicație mobila și web de unde să se primească comenzi în restaurante și să fie la propriu că un catalog de magazine virtuale, că fiecare client să poată comanda de la ce magazin dorește el.

# Bibliografie

* <https://howtodoinjava.com/java8/java-stream-distinct-examples/>
* <http://javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.html>
* <https://www.baeldung.com/java-serialization>
* <https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>
* <https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html>
* <https://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html>
* <https://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html>