DOCUMENTATIE

TEMA 4

NUME STUDENT: Ungureanu Iulia Iarina

GRUPA: 30225

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

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

[3. Proiectare 3](#_Toc95297887)

[4. Implementare 3](#_Toc95297888)

[5. Rezultate 3](#_Toc95297889)

[6. Concluzii 3](#_Toc95297890)

[7. Bibliografie 3](#_Toc95297891)

# Obiectivul temei

Proiectați și implementați un sistem de management al livrării de alimente pentru o companie de catering. Clientul poate comanda produse din meniul companiei. Sistemul trebuie să aibă trei tipuri de utilizatori care se conectează folosind un nume de utilizator și o parolă: administrator, angajat și client.

Administratorul poate:

• Importați setul inițial de produse care vor popula meniul dintr-un fișier .csv.

• Gestionați produsele din meniu: adăugați/ștergeți/modificați produse și creați produse noi

compus din mai multe produse din meniu (un exemplu de produs compus ar putea fi

numit „meniu zilnic 1” compus dintr-o supă, o friptură, o garnitură și un desert).

• Generați rapoarte despre comenzile efectuate luând în considerare următoarele criterii:

o intervalul de timp al comenzilor – trebuie generat un raport cu comenzile efectuate

între o anumită oră de început și o dată de sfârșit, indiferent de dată.

o produsele comandate de mai mult de un anumit număr de ori până acum.

o clienții care au comandat mai mult de un anumit număr de ori până acum și

valoarea comenzii a fost mai mare decât o sumă specificată.

o produsele comandate într-o zi specificată cu numărul de ori pe care le au

fost comandat.

Clientul poate:

• sa se inregistreze cu nume de utilizator și parola si apoi se poate conecta cu ele în sistem.

• Vizualiza lista de produse din meniu.

• Căutați produse pe baza unuia sau mai multor criterii, cum ar fi cuvântul, evaluarea, numărul de calorii/proteine/grasimi/sodiu/pret.

• Crea o comandă formată din mai multe produse – pentru fiecare comandă va fi data și ora

a persistat și va fi generată o factură care va enumera produsele comandate și prețul total

a ordinului.

Angajatul este anunțat de fiecare dată când o nouă comandă este efectuată de către un client pentru a putea pregăti livrarea preparatelor comandate.

Sub-obiective:

* Definiți interfața IDeliveryServiceProcessing care conține principalele operațiuni care pot
* fie executat de administrator și client ( 1 )
* implementarea claselor MenuItem, BaseProduct și CompositeProduct ( 2 )
* implememntarea modelul de proiectare Observer pentru a notifica angajatul
* Implementarea aplicației de simulare ( 4 )
* Testați aplicația de simulare ( 5 )

# Analiza problemei, modelare, scenarii, cazuri de utilizare

*Un administrator poate sa aleaga setul initial de produse care vor fi adaugate la meniu. Poate sa editeze produsele: add/delete/modify (poate sa adauge base products) si sa creezee produse noi din produsele deja existente.Poate sa genereze rapoarte.*

*Un client se poate inregistra folosind un nume si o parola si apoi le poate flosi pentru a intra in aplicatie.*

*Poate sa vada lista produselor din meniu (exista un meniu). Poate sa caute pe baza unui cuvant, sau dupa raiting, sau dupa numarul de calorii, proteine, grasimii, pret. Poate sa creeze o comanda cu mai multe produse. El v-a primii o chitanta cu toate produsele, pretul lor, data actuala si pretul total.*

*Angajatul poata sa vada toate comenzile.*

Use Case: ștergere produs

Primary Actor: user

Main Success Scenario:

1. Utilizatorul alege produsul dorit

2. Utilizatorul apasă butonul delete.

3. Aplicația actualizează tabela de produse.

Alternative Sequence: Utilizatorul nu a selectat nici un produs => mesaj de eroare

Use Case: adăugare produs

Primary Actor: user

Main Success Scenario:

1. Utilizatorul introduce de la tastatura datele produsului : nume, raiting, numarul de calorii, proteine, grasimii si pret.

2. Utilizatorul apasă butonul New Produs.

3. Aplicația actualizează tabela de produse.

Alternative Sequence: Utilizatorul nu a introdus toate informațiile, sau a introdus date incorecte (Ex: raiting, numarul de calorii, proteine, grasimii, pret negative sau un sir de caractere) => mesaj de eroare

Use Case: editare produs

Primary Actor: user

Main Success Scenario:

1. Utilizatorul alege câmpul pe care își dorește sa îl editeze
2. Utilizatorul introduce noua data in câmpul respectiv.
3. Utilizatorul apasă butonul Edit Produs.
4. Aplicația actualizează tabela de produse.

Alternative Sequence:

* Utilizatorul nu a selectat nici un produs => mesaj de eroare
* Utilizatorul nu a introdus toate informațiile, sau a introdus date incorecte (Ex raiting, numarul de calorii, proteine, grasimii, pret negative sau un sir de caractere) => mesaj de eroare

Use Case: adăugare produs compus

Primary Actor: user

Main Success Scenario:

1. Utilizatorul introduce de la tastatura numele produsului

2. Utilizatorul apasă butonul Create Product.

3. Aplicația actualizează tabela de produse.

4. Utilizatorul alege un produs simplu pentru a fi adaugat la produsul compus;

5. Aplicația actualizează tabela de produse.

6. Se repeta de la pasul 4

Alternative Sequence: Utilizatorul nu a introdus toate informațiile, sau a introdus date incorecte (Ex: raiting, numarul de calorii, proteine, grasimii, pret negative sau un sir de caractere) => mesaj de eroare

Use Case: adăugare comanda

Primary Actor: user

Main Success Scenario:

1. Utilizatorul alege produsul pe care vrea sa il comande.
2. Apasa butonul add order;
3. Se repeta de la pasul 1 pana au adaugat toate produsele.
4. Utilizatorul apasă butonul new Order.
5. Aplicația actualizează comenzile si elibereaza o comanda.

Alternative Sequence:

* Utilizatorul nu a selectat nici un produs => mesaj de eroare

Cerințe funcționale:

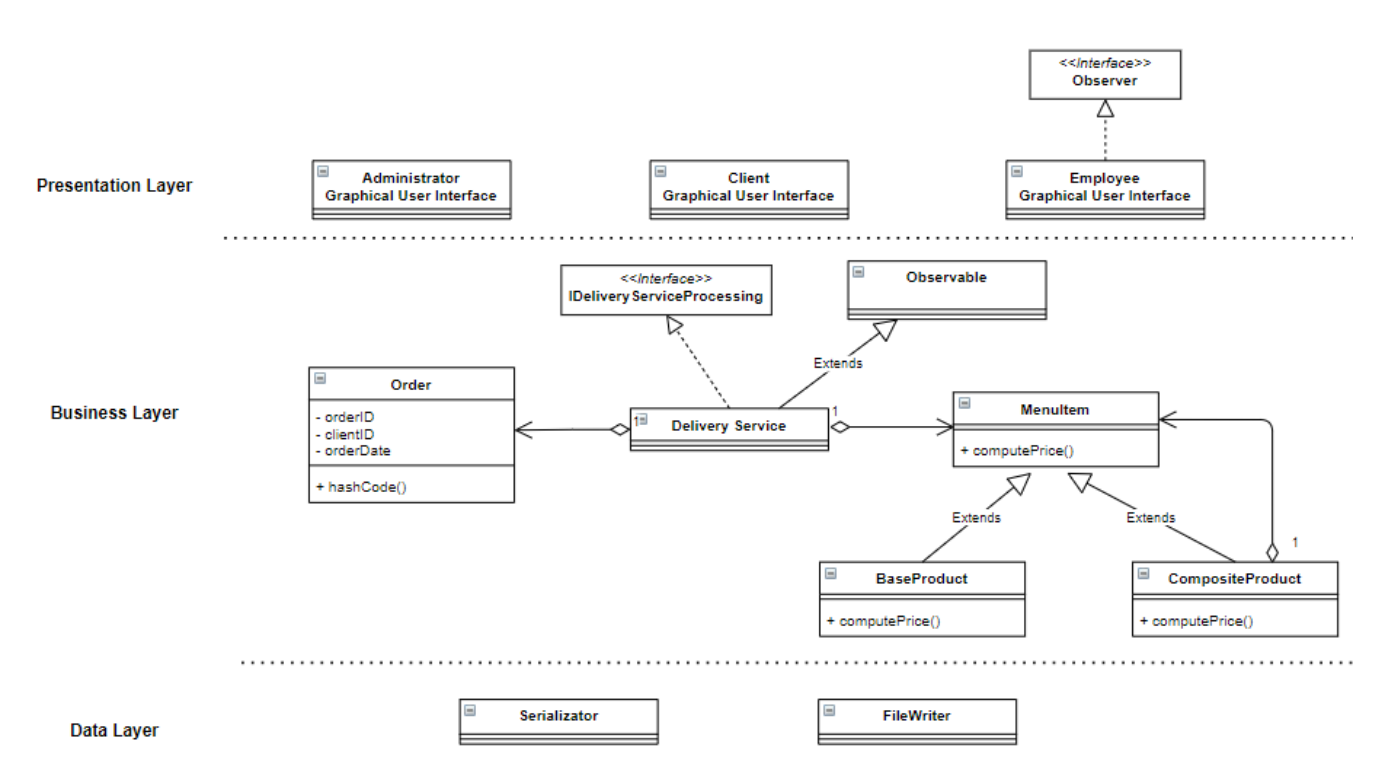
Aplicația trebuie sa afișeze toate produsele, existente.

Aplicația trebuie sa lase utilizatorul sa adauge noi client, produse si comenzi;

Aplicația trebuie sa lase utilizatorul sa editeze produsele si clienții existenți.

Aplicația trebuie sa lase utilizatorul sa șteargă produsele si clienții existenți.

# Proiectare

* 1. Diagramele UML de clase si de pachete:

Presentation Layer contine toate interfetele necesare aplicatiei:

1. Administratorul – contine un Jtabel care afiseaza toate produsele (MenuItem) prin intermediul clasei DeliveryService. Ea contine 3 butoane pentru operatile pe produse : add product (datele introduse de utilizator sunt luate prin intermediul JTabel.get() si apoi este creat un obiect de tipul MenuItem cu ele care este adaugat in products cu metoda .addMenuItem), delete product (randul selectat de utilizator este preluat cu Jtable.getRow(), in functi de pozitia acestiua poate fi gasit in ArrayList<>Products si eliminat cu .remove), edit product ( cu ajutorul Jtabel putem selecta data modificata si obiectul respectiv si modifica in lista de produse). Aici se pot si importa produse dintr-un fisier de tip .cvs aflat in folderul proiectului si se pot genera diferite rapoarte bazate pe datele introduse de utilizator.
2. PageReport este o subPagina a paginii administrator care contine 4 zone de completare pentru generarea rapoartelor: numai dupa ce toate zonele au fost completate se poate apaza butonul generate pentru a crea cele patru fisiere respective: prima zona contine doua label-uri pentru ora se start si ora de final intre care trebuie gasite comenzile si afisate. A doua zona contine un camp pentru introducerea unui numar de produse. In functe de acest numar trebuie afisate in al doilea fisier toate produsele care au fost comandate de mai mult decat nr introdus. In zona a treia trebuie introdus un numar de clienti si un pret minim de comanda pe baza caroar se va genera al 3-lea fisier. In zona a patra se va introduce o data pentru care se vor afisa toate produsele comandate in acea zi impreuna cu numarul de comanzi in care apar.
3. Login – este prima pagina a aplicatiei contine doua butone : primul pentru inregistrare: prima oara trebuie introdus un nume de utilizator si apoi parola si apoi se apasa butonul, al doilea este pentru logare trebuie introdus un nume de utilizator si apoi parola si apoi apasat butonul respectiv.
4. Client – interfata contine doua JTabel – ul , unul pentru produse, si unul pentru comanda. Acesta poate adauga in comanda obiectele afisate in primul Jtabel prin intermediul butonului add product si apoi poate termina comanda cu ajutorul butonului add order.

Data Layer

1. FilesWriter – contine doua metode : una pentru scrierea chitantelor. Aceasta primeste un text generat in DEliveryService si il adauga intr-un fiser care il denumeste in functie de data si ora actuala. A doua metoda este pentru generarea unui raport, aceasta primeste tipul raportului :1,2,3,4 si continutul acestulia si genereaza un fisier cu numele RAPORTtypeDATA.
2. Serialization – contine metode pentru serializarea a trei tipuri de date : MenuItem, Users ,Orders.

Fiecare dintre acestea are cate doua metode :serialize ( care salveaza datele trimise in fisierul respectiv utilizand metodele: ObjectOutputStream out = new ObjectOutputStream(fileOut); out.writeObject(class);) si getAll (care returneaza un ArrayList/ HashMap pentru clasa respectiva cu datele din fisier) : orders=( HashMap <Order, ArrayList <MenuItem> >) in.readObject();

Business Layer:

BaseProduct – reprezinta un produs simplu si extinde MenuItem si implementeaza serializable. Contine urmatoarele campuri si o metoda de calculare a pretului.

private String title ;

private double rating ;

private int calories ;

private int protein ;

private int fat ;

private int sodium ;

private int price ;

CompositeProduct exitde MenuItem si implementeaza Serializable si reprezinta un produs comus din mai multe produse simple, de aceea are ca campuri ArrayList<MenuItem> menuItems pentru salvarea produselor simple si String title pentru salvarea titlului si o metoda de calculare a pretului.

DeliveryService implementeaza interfata DeliveryServiceIntreface si extinde clasa Observable pentru a notifica angajatul de fiecare dată când o nouă comandă este creata. Are ca campuri HashMap<Order, ArrayList<MenuItem>> orders si ArrayList<MenuItem> menuItems pentru stocarea momentana a comenzilor si a produselor pana la serializare. Ea implementeaza toate operatile de pe produse ,generarea de rapoarte, crearea comenzilor si search -ul pe produse. Java oferă un mecanism, numit serializare obiect, în care un obiect poate fi reprezentat ca o secvență de octeți care include datele obiectului, precum și informații despre tipul obiectului și tipurile de date stocate în obiect. După ce un obiect serializat a fost scris într-un fișier, acesta poate fi citit din fișier și deserializat, adică informațiile de tip și octeții care reprezintă obiectul și datele acestuia pot fi utilizați pentru a recrea obiectul în memorie. Elementele de meniu, comenzile efectuate și informațiile despre utilizator vor fi continuate folosind serializare astfel încât să fie disponibilă la viitoarele execuții ale sistemului prin intermediul deserializării.

MenuItem este o clasa abstract care contine doar campul titlu si metoda computePrice();

Order reprezinta o comanda si are doua campuri: private int clientID; private Date date; Data comenzii este utilizata pentru generarea rapoartelor in functe de ora comenzii, ziua, anul, luna , pretul sau obiectele care au fost comandata utilizand streamuri si expresii lambda in interiorul clasei DeliveryService. Acestea pot fi folosite pentru luara comenzilor din hashMap:

List<Order> numbers = new ArrayList<Order>();

orders.forEach((k,v) -> {

numbers.add(k);

});

--- pentru filtrarea pe baza unor inforrmatii:

List<Order> sortedOrders =

numbers.stream()

.filter(n -> {

if(n.getDate().getHours() >= startHour && n.getDate().getHours() <= endHour){

return true;

}

return false;

})

.collect(toList());

Filtrarea elementelor repetate de minNrOrderP:

Set<MenuItem> produseRep = products // am creat un set cu produsele care se repeta de

//cel putin minNrOrderP ori

.stream()

.filter(n -> products // ele se filtreaza astfel: n este adaugat daca

.stream() // in acesr stream interior avem un numar

.filter(x -> x.equals(n)) // corect de elemente la final

.count() >= minNrOrderP)

.collect(Collectors.toSet());

--- sau pentru eliminarea unor obiecte care se numara de n ori cu diferite conditii:

Set<Order> filtru = comenzi

.stream()

.filter(n -> comenzi

.stream()

.filter(x -> {

if(x.getClientID() == n.getClientID() && getPriceOfOrder(x)>minPriceC){

return true;

}

return false;

})

.count() >= minNrOrderC)

.collect(Collectors.toSet());

Type reprezinta tipul de utilizator : Adimn , user , client;

User reprezinta utilizatorul si implementeaza clasa Serializable. Contine urmatoarele campuri:

private static int max ;

private String username ;

private String password ;

private int id ;

private Type type ;

Contine metoda setMax care genereaza automat id-ul utilizatorilor prin incrementare succesiva a maximului actual.

# Concluzii

*Din aceasta tema am învățat sa folosesc expresii lambda si procesarea cu streamuri. Folosirea disanului Composite Pattern pentru modelarea claselor si design by contract (utilizarea preconsitilor si postconditilor) definirea unui invariant pentru clasa. Folosirea interfetei Observer Design Pattern pentru notificarea unei clase „observator” asupra unei alte clase „observate”. Utilizarea serializarii pentru salvarea datelor in fisiere binare pentru pastrarea informatiei la urmatoarele acesarii.*

# 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-serializationo
* 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.html o
* http://stackoverflow.com/questions/11415160/how-to-enable-the-java-keywordassert-in-eclipse-program-wise