**FUNDAMENTAL PROGRAMMING TECHNIQUES**

**ASSIGNMENT 4**

**FOOD DELIVERY MANAGEMENT SYSTEM**

**Moldovan Claudia**

**Grupa 30224**

**An universitar 2021-2022**

Cuprins

[1.Cerinta 3](#_Toc102752762)

[2.Diagrama UML 4](#_Toc102752763)

[3. Implementare 4](#_Toc102752764)

## 1.Cerinta

Proiectați și implementați un sistem de management al livrării de alimente pentru o companie de catering. Clientul poate comanda produse din meniul companiei. Sistemul ar trebui să aibă trei tipuri de utilizatori care se conectează folosind un nume de utilizator și o parolă: administrator, angajat obișnuit 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 – sunt rapoartele care trebuie generate cu comenzile efectuate între o anumită oră de început și o anumită oră 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:

• Înregistrați-vă și utilizați numele de utilizator și parola înregistrate pentru a vă conecta în sistem.

• Vizualizați lista de produse din meniu.

• Căutați produse pe baza unuia sau mai multor criterii, cum ar fi cuvântul cheie (de exemplu, „supă”), evaluarea,

numarul de calorii/proteine/grasimi/sodiu/pret.

• Creați 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 al comenzii.

Angajatul este anunțat de fiecare dată când este efectuată o nouă comandă de către un client pentru ca acesta să poată pregăti livrarea preparatelor comandate.

## 2.Diagrama UML

A screenshot of a computer

Description automatically generated with medium confidence

## 3. Implementare

* Pachetul business:
* Clasa User: lucreaza cu id-ul, numele, parola si tipul specifice fiecarui user (0-client, 1- administrator)
* Clasa Comanda: lucreaza cu id-ul comenzii, id-ul clientului, data in care a fost data comanda si suma specifica fiecarei comenzi
* Clasa ItemMeniu : lucreaza cu numele, nota, caloriile, proteinele, sodiul, grasimile si pretul fiecarui produs
* Clasa BaseProduct: extinde ItemMeniu si lucreaza cu cu aceleasi atribute precum ItemMeniu
* Clasa FormareProdus: extinde ItemMeniu si lucreaza cu cu aceleasi atribute precum ItemMeniu, in plus are un HashSet ce contine produsele care alcatuiesc noul produs
* Clasa ConnectionFactory: implementeaza metodele prin care se face legatura cu baza de date pentru stocarea utilizatorilor
* Clasa DeliveryService: implementeaza toate metodele din interfata IdeliveryService, adica toate functionalitatile aplicatiei
* Interfata IdeliveryServiceProcessing: antelele metodelor functionalitatilor aplicatiei
* Pachetul dataLayer:
* Clasa FileWriterClass: implementeaza metodele de afisare a bonului si istoricului comenzilor intr-un fisier text specific
* Clasa Serializator: implementeaza metodele de serializare si deserializare a datelro
* Pachetul presentation:
* Clasa LogIn: imaginea filei de logare, unde se poate si inregistra un nou client
* Clasa ClientPres: imaginea filei Client unde utilizatorul foloseste aplicatia pentru adauga produse intr-o comanda, a plasa comanda si a cauta produse in functie de o anumita categorie
* Clasa AdministratoriPres: imaginea filei Administrator unde utilizatorul foloseste aplicatia pentru a edita, adauga, sterge produse sau pentru a crea un CompositeProduct pe care ulterior sa il adauge in lista de produse initiala
* Clasa ReportsPres: imaginea filei Reports unde utilizatorul foloseste aplicatia pentru a genera rapoarte legate de clienti, produse, comenzi in functie de ore, zile, totalul comenzii, etc.
* Clasa Angajat: fereastra in care un angajat va fi anuntat la plasarea unei comenzi
* Pachetul start:
* Clasa Start: o putem numi si programul pricipal, prin rularea ei se deschide fereastra aplicatiei unde accesam managerul de comenzi
* Clasa User

Aceasta clasa contine datele care mapeaza tabelul „user”. Contine ca field-uri toate coloanele din tabel, numele acestora fiind identice cu cu numele coloanelor din tabel pentru a se putea folosi thenica reflectiei in crearea obiectelor de tipul respectiv in urma unei interogari. Ca si metode existente in aceasta clasa sunt doar metoda de tipul setter si getter pentru a accesa variabilele de clasa. De asemenea, contine si metoda toString pentru a putea afisa informatiile despre un client.

* Clasa Comanda

Aceasta clasa contine datele ce caracterizeaza o comanda is anume: id, idClient, data si totalul comenzii. Ca si metode existente in aceasta clasa sunt doar metoda de tipul setter si getter pentru a accesa variabilele de clasa. De asemenea, contine si metoda toString pentru a putea afisa informatiile despre o comanda. In plus, mai avem o metoda hashCode() ce returneaza id-ul comenzii.

* Clasa ItemMeniu

Aceasta clasa contine datele despre un produs si anume: nume, nota, calorii, proteine, sodiu, grasimi si pret. Ca si metode existente in aceasta clasa sunt doar metoda de tipul setter si getter pentru a accesa variabilele de clasa. De asemenea, contine si metoda toString pentru a putea afisa informatiile despre un client. In plus, mai avem o metoda numita formarePret() prin care aflam valoarea pretului unui produs. Folosind Composite Design Pattern, aceasta clasa va fi mostenita de catre BaseProduct si FormareProdus

* Clasa BaseProduct

Mostenind clasa ItemMeniu, este caracterizata de aceleasti atribute, iar metodele din clsaa ItemMenu vor fi rescrise, particularizandu-le.

* Clasa FormareProdus

Mostenind clasa ItemMeniu, este caracterizata de aceleasti atribute, in plus avand un HashSet in care pastram produsele ce compun un astfel de produs.Ca si metode, le vom rescrie pe cele din clasa ItemMeniu, particularizandu-le ( exememplu: la calorii, proteine, grasimi, sodiu – le vom adauga intr-o variabila valorile soecifice fiecarui produs, la nota vom realiza media aritmetica, iar la pret, pentru fiecare produs adaugat in comanda, din suma produselor vom scadea 10 lei)

* Clasa Connection

Aceasta clasa are scopul de a returna o conexiune la baza de date si de a inchide unele componente folosite pentru a realiza o interogare. Clasa ConnectionFactory contine urmatoarele metode: createConnection (aceasta metoda este un privata, creaza o conexiune care mai apoi este returnata), getConnection (returneaza o conexiune la baza de date obtinuta prin apelarea metodei prezentate precedent). Aceasta clasa contine si un constructor care verifica existenta driver-ului in conectorul mysql. De asemenea, contine si metodele: close(Connection), close(Statement) si close(ResultSet), 3 metode supraincarcate care au rolul de a inchide obiectul primit ca argument.

* Interfata IDeliveryService

Aceasta interfata contine toate antetele metodelor ce vor fi implemetate ulterior in clasa DeliveryService. Aici, inainte de fiecare metoda avem setate pre si post conditii ce trebuie respectate pentru a se realiza cu bine metoda.

* Clasa DeliveryService

Aceasta clasa implementeaza interfata IdeliveryService, deci vom implementa toate metodele definite aici. Variabilele de clasa folosite sunt: meniu (lista produselor), produseComandate (HashMap in care cheia este comanda, iar valoarea e un HashSet de ItemMeniu) si comenzi ( lista comenzilor).

Metode implementate:

* meniu – punem in lista de produse inregistrarile din fisierul „products.csv”
* refreshCSV – rescriem in CSV noile informatii. Aceasta metoda ne ajuta sa pastram in fisier datele mereu actualizate, astfel incat tabelul dupa o anumita operatie se va actualiza imediat
* stergereMeniu – metoda utilizata de catre administrator cand doreste sa stearga un produs
* adaugareMeniu - metoda utilizata de catre administrator cand doreste sa adauge un produs
* editareMeniu - metoda utilizata de catre administrator cand doreste sa editeze un produs
* adaugareMeniuNou – metoda utilizata de catre administrator cand creaza un nou FormareProdus si il adauga in lista de produse
* comanda – metoda utilizata de client in momentul in care transmite o comnda. In acel moment, se va adauga o comanda la lista de compenzi si produsele din comanda la lista produselor comandate
* gasesteProduse - foloseste lambda expression. Are drept parametrii categoria (in functie de ce parametru cautam) si detaliu (poate fi string sau int, in functie daca cautam dupa nume sa dupa restul atributelor). De exemplu, pentru nume, lambda expression, folosind stream.filter vor gasi elementele ce contin in numele lor cuvantul introdus de noi. La final, vom pune „.collect(Collectors.toList())” pentru a le introduce intr-o lista pe care ulterior o vom afisa in JTable
* generareInterval – are ca parametrii ora de inceput si cea de sfarsit. Se foloseste in aceeasi maniera ca la metoda gaseste produs lambda expression, utilizand un filtru pentru a vedea daca ora comenzii este in intervalul dat de noi.
* generareRaport2 – parcurgem o data comenzile pentru a introduce produsele intr-un ArrayList pe care ulterior, utilizand lambda expression vom numara numarul de aparitii (functioneaza ca un vector de frecventa). Daca numarul de aparitii este mai mare decat atributul dat de noi in antet, vom printa acel produs.
* generareRaport3 - filtram o data comenzile pentru a le lua doar pe cele cu pretul de peste suma introdusa de noi, urmand sa numaram comenzile date de fiecare client, iar in cazul de e mai mare decat numarul dat de noi, vom afisa in raport id-ul clientului cu numarul de comenzi
* generareRaport4 - filtram comenzile penrtu a le obtine pe cele comandate intr-o anumita zi, urmand sa tinem intr0un arrayList produsele pentru a le putea numara ulterior utilizand lambda expression, urmand sa le afisam.
* Clasa FileWriterClass

Aceasta clasa ne ajuta sa printam rezultatele obtinute in fisierele text dorite, dat ca parametru in functia writeToFile(String fisier). Astfel, in momentul in care apasam pe butonul „Finalizare”, in fisierul „Istoric bonuri.txt” va afarea la sfarsit ultima comanda. Totodata, se va crea si fisierul „Bon comanda (idComenzii) (numele Clientului).txt” in care va fi printat bonul comenzii date de catre un anumit client.

* Clasa Serializator

Aceasta metoda contine metodele serialization() si deserialization() cu ajutorul carora luam informatiile stocate anterior in fisierul „file.txt” pentru a nu pierde informatiile in momentul in care inchidem aplicatia. Serializarea o realizam dupa plasarea unei comenzi, iar deserializarea o facem in momentul in care deschidem aplicatia.

* Clasa LogInPres

Aceasta extinde Jframe. Aici sunt definite ca atribute private fiecare buton de care avem nevoie si fereastra propriu-zisa. Cu ajutorul conexiunii cu baza de date, verificam daca este client sau administrator si in momentul in care apasam LogIn ne va duce pe fereastra destinata noua. Daca nu suntem client inca, vom apasa pe Register, urmand sa fim introdusi in sistem drept clienti, urmand apoi sa intram pe fila noastra.

Graphical user interface, application, Word

Description automatically generated

* Clasa AdministratorPres

Aceasta extinde JFrame. Aici sunt definite ca atribute private fiecare buton de care avem nevoie, tabelul in care vor fi afisate datele, filed-urile in care scriem informatii despre un anumit produs, fereastra propriu-zisa, variabila in care calculam totalul unui nou meniu. In functie de butonul accesat, tabelul se va actualiza in acelasi timp. Fiecare buton este utilizabil cu ajutorul apelarii metodei „addActionListener”, in care vom implemnta metoda propriu-zisa pentru butonul respectiv.

Graphical user interface

Description automatically generated

* Clasa ReportsPres

Aceasta extinde JFrame. Aici sunt definite ca atribute private fiecare buton de care avem nevoie, tabelul in care vor fi afisate datele, filed-urile in care scriem informatii despre un anumit raport, fereastra propriu-zisa. In functie de butonul accesat, se va genera raportul respectiv. Fiecare buton este utilizabil cu ajutorul apelarii metodei „addActionListener”, in care vom implemnta metoda propriu-zisa pentru butonul respectiv.

Graphical user interface

Description automatically generated

* Clasa ClientPres

Aceasta extinde JFrame. Aici sunt definite ca atribute private fiecare buton de care avem nevoie, tabelul in care vor fi afisate datele, filed-urile in care scriem informatii despre un anumit produs ce vrem sa il cautam, field-ul in care se screi informatia despre produsul selectat din tabel, fereastra propriu-zisa si comboBox-ul ce ne ajuta in cautarea unui produs in functei de ceva anume. In functie de butonul accesat, se va genera raportul respectiv. Fiecare buton este utilizabil cu ajutorul apelarii metodei „addActionListener”, in care vom implemnta metoda propriu-zisa pentru butonul respectiv.

Table

Description automatically generated

* Clasa Angajat

Aceasta extinde Jframe, insa implementeaza Observable prin care la plasarea unei comenzi de catre un client se afiseaza comanda impreuna cu produsele care vor trebui pregatite. Pentru acest lucru, in metoda comanda, din DeliveryService folosim metoda setChange(0 pentru a introduce schimbarile, urmand sa notificam observer-ul prin metoda notifyObserver().

Graphical user interface, application, Word

Description automatically generated

* Clasa Main

Aceasta clasa instantiaza interfata ( care este realizata in Clasa View), urmata de setarea vizibilitatii aplicatiei. Totodata, apelam si deserializarea pentru a avea din nou in aplicatie datele de inaintea deschiderii aplicatiei.