

DOCUMENTATIE TEMA 4

CATERING

Cociubei Mihai-Ioan
Grupa 30227
Profesor Laborator Assist Mitrea Dan

Cuprins

1. Obiective.....	2
1.1. Obiectiv Principal	2
1.2. Obiectiv Secundar	3
2. Analiza problemei.....	3
3. Cazuri de utilizare	3
4. Proiectare	4
5. Interfata grafica.....	6
6. Implementare	8
7. Concluzii.....	10
8. Rezultate	10
9. Bibliografie	10

1. Obiective

1.1. Obiectiv Principal

Obiectivul acestui proiect a fost de a realiza simularea unui sistem de livrare a mancarii catering de la un restaurant. Clientul poate comanda produse din meniul companiei. Sistemul are implementati 3 categorii de utilizatori administrator (Admin), angajat (employee) si client.

1.2. Obiectiv Secundar

Administratorul poate incarca setul initial de produse. Poate salva utilizatorii sau modifica produsele puse la vanzare, plus un lucru foarte important poate genera anumite rapoarte prin care sa analizeze produsele cele mai vandute intr-un anumit interval de timp, poate sa vada clientii fideli si multe altele.

2. Analiza problemei

Pentru o structurare mai buna a codului l-am impartit in 3 pachete. Pachetul de prezentare care contine atat view cat si partea de controller pentru toate operatiile. Business logic adica partea de model unde este scrisa partea de algoritmi si cel de-al 3 nivel data layer care realizeaza salvarea si incarcarea informatiilor trecut salvate.

Am definit interfata `IdeliveryServiceProcessing` care contine operatiile principale pe care le vor realiza atat administratorul cat si clientul. Administratorul poate importa produse, modifica produse din meniu, si nu in ultimul rand de a genera anumite rapoarte. Clientul poate pune o comanda noua, dupa care va primi un bon in format .txt, poate cauta produse dupa anumite criterii

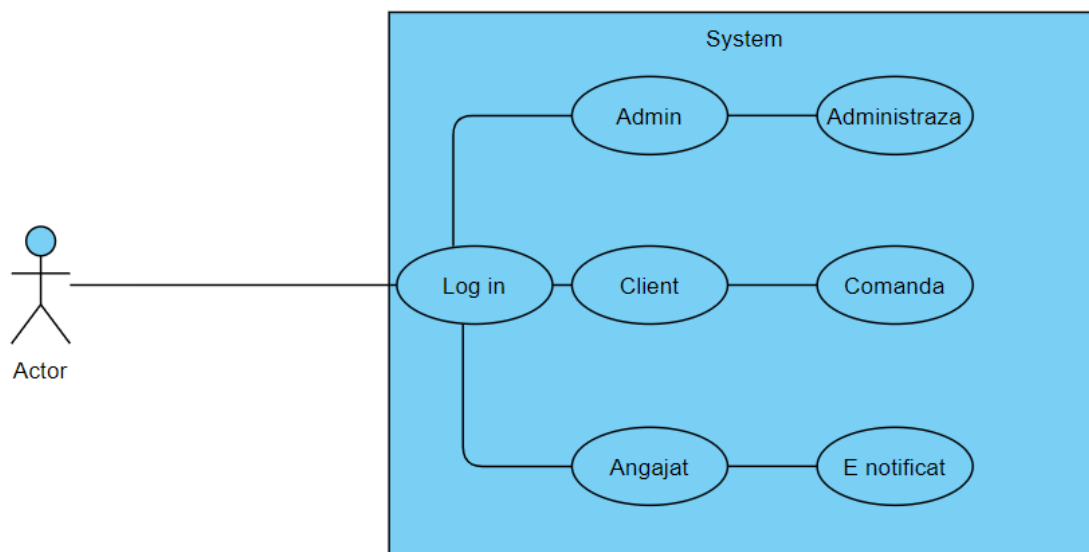
Pentru a putea adauga atat un singur element cat si o combinatie de mai multe produse am utilizat Composite Design Pattern pentru clasele `MenuItem`, `BaseProduct` and `CompositeProduct`. Iar pentru a notifica angajatii am folosit Observer Design Pattern in momentul in care apare o comanda angajatul conectat la magazin va fi notificat in momentul in care clientul a pus comanda si despre datele comenzii pentru a o putea prepara si livra.

1. BLL. Acesta contine clasele de baza, `MenuItem`, `BaseProduct`, `CompositeProduct`, `User`, `Role`, `Admin`, `Employee` `Client` `Observable` si multe altele

2. `DataLayer`. Care contine clasa `FileWriter`.

3. `PresentationLayer`. Aici se gasesc interfetele grafice pentru administrator cat si pentru angajati si clienti.

3. Cazuri de utilizare



Operatiile se efectueaza in aceasta ordine pentru a primi rezultatul dorit si anume intai orice utilizator trebuie sa se logheze la sistem dupa care automat dupa tipul lui i va apare o noua interfata.

Prima data se adauga produsele de baza in meniu, avand asociat un pret. La fiecare adaugare va apare in tabel elementul adaugat iar daca clientul doreste sa il stearga are aceasta posibilitate. Cand este satisfacut de produsele din cos poate apasa pe plasare comanda care va inregistra comanda pusa de catre utilizator

Daca se doreste, se poate schimba pretul unui produs existent prin specificarea numelui acestuia si noului pret sau cantitatile de proteina sau numele orice camp mai exact.

Se introduce un produs complex, iar apoi se adauga in compozitia lui produse de baza, la fiecare adaugare in compozitia produsului complex, va fi recalculat pretul acestuia si va fi afisat din nou intregul meniu exact ca si in cazul punerii unei comenzi de catre client.

Daca se doreste, se pot sterge produse din meniu specificand numele produsului. In orice moment puteti vedea meniul complet apasand pe butonul "View all" din stanga pentru a putea comanda produsul dorit ferestrei. Acesta va popula tabelul cu toate produsele existente. Sau daca doreste clientul sa caute dupa un anumit criteriu poate realiza si aceasta operatie

4. Proiectare

Proiectarea claselor si a pachetelor a fost facuta dupa modelul dat, astfel incat nu se va sari peste nivele, fiecare clasa dintr-un pachet avand "voie" sa apeleze doar metode din clase aflate la un nivel inferior ei.

Buseniss level

Am definit interfata `IdeliveryServiceProcessing` care contine operatiile principale pe care le vor realiza atat administratorul cat si clientul. Administratorul poate importa produse, modifica produse din meniu, si nu in ultimul rand de a genera anumite rapoarte. Clientul poate pune o comanda noua, dupa care va primi un bon in format .txt, poate cauta produse dupa anumite criterii

Pentru a putea adauga atat un singur element cat si o combinatie de mai multe produse am utilizat Composite Design Pattern pentru clasele MenuItem, BaseProduct and CompositeProduct. Iar pentru a notifica angajatii am folosit Observer Design Pattern in momentul in care apare o comanda angajatul conectat la magazin va fi notificat in momentul in care clientul a pus comanda si despre datele comenzii pentru a o putea prepara si livra.

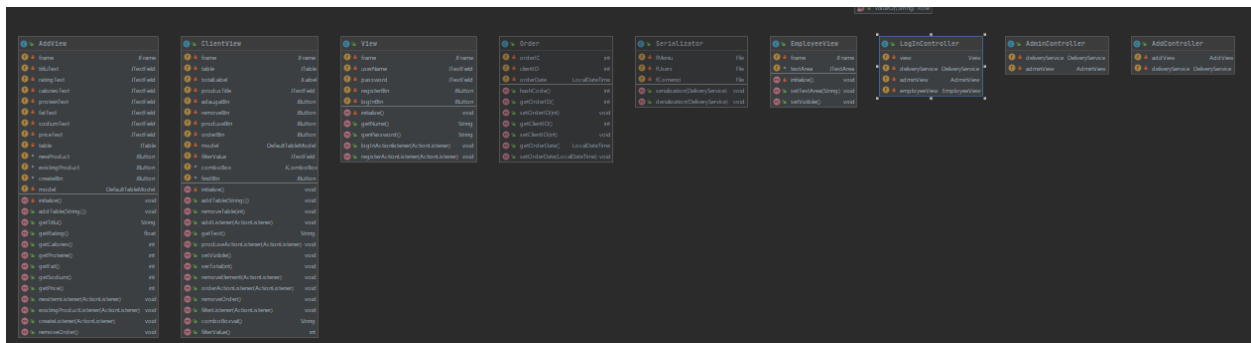
Clasa `DeliveryService` realizeaza partea de algoritmi si adica am creat o interfata cu toate operatiile de care am nevoie dupa care am implementat aceasta interfata cu toate operatiile aferente. Aici este implementat popularea meniului. Dupa care pentru a stii fiecare utilizator ce rol are am creat un enum cu campurile `ADMIN`, `EMPLOYEE`, `CLIENT`. Am hardcodat unul din fiecare pentru a avea de fiecare data. Dar un nou utilizator poate fi introdus in orice moment.

Pachetul DataLayer:

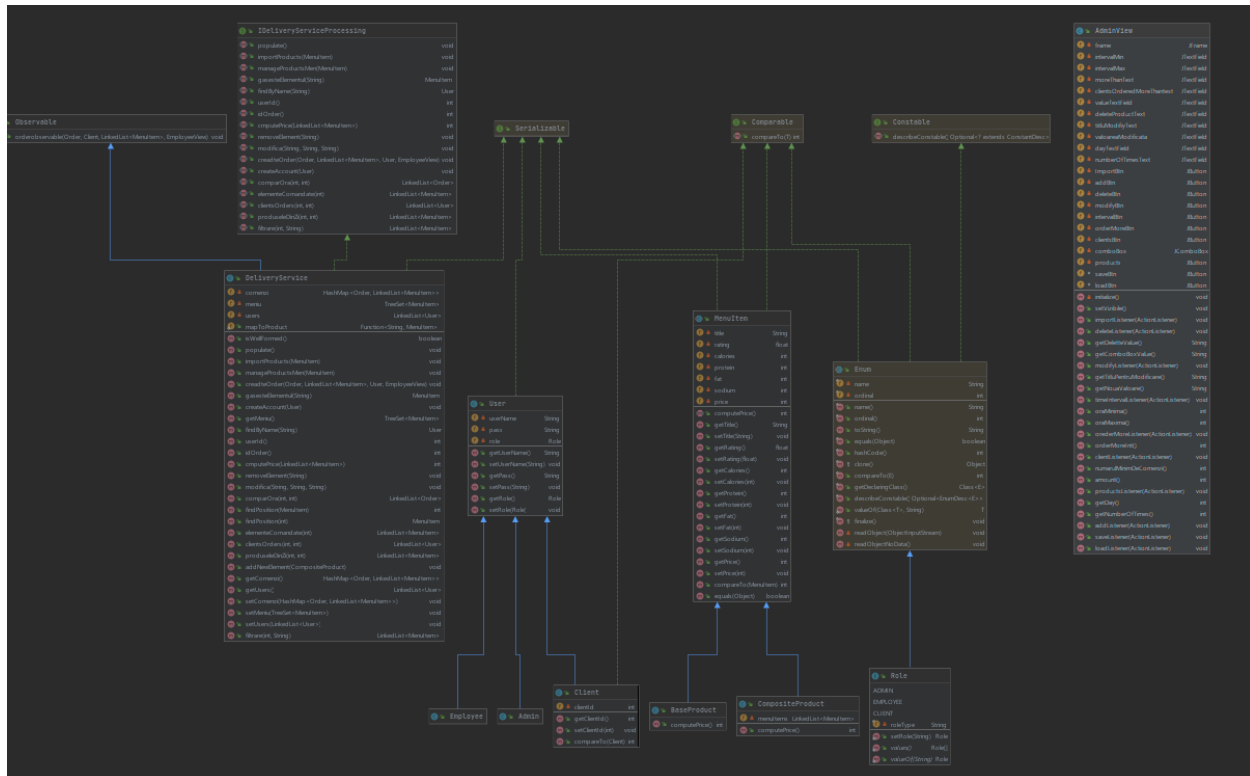
Clasa FileWriter. Aceasta clasa nu a fost folosita deloc in implementarea proiectului. Dar incarcarea resurselor salvate anterior a fost utilizata. Doarece trebuie sa salvam modificarile facute atat de Administrator cat si noile conturi inregistrate in sistem.

Pachetul presentation layer

În cadrul acestui pachet am implementat toate view-urile pe care un utilizator le poate vedea, cât și partea care conectează partea de model cu view și anume controller, dându-le nume sugestive.



Aici se poate observa partea din pachetul presentation unde sunt prezente toate view urile cat si partea de controller.



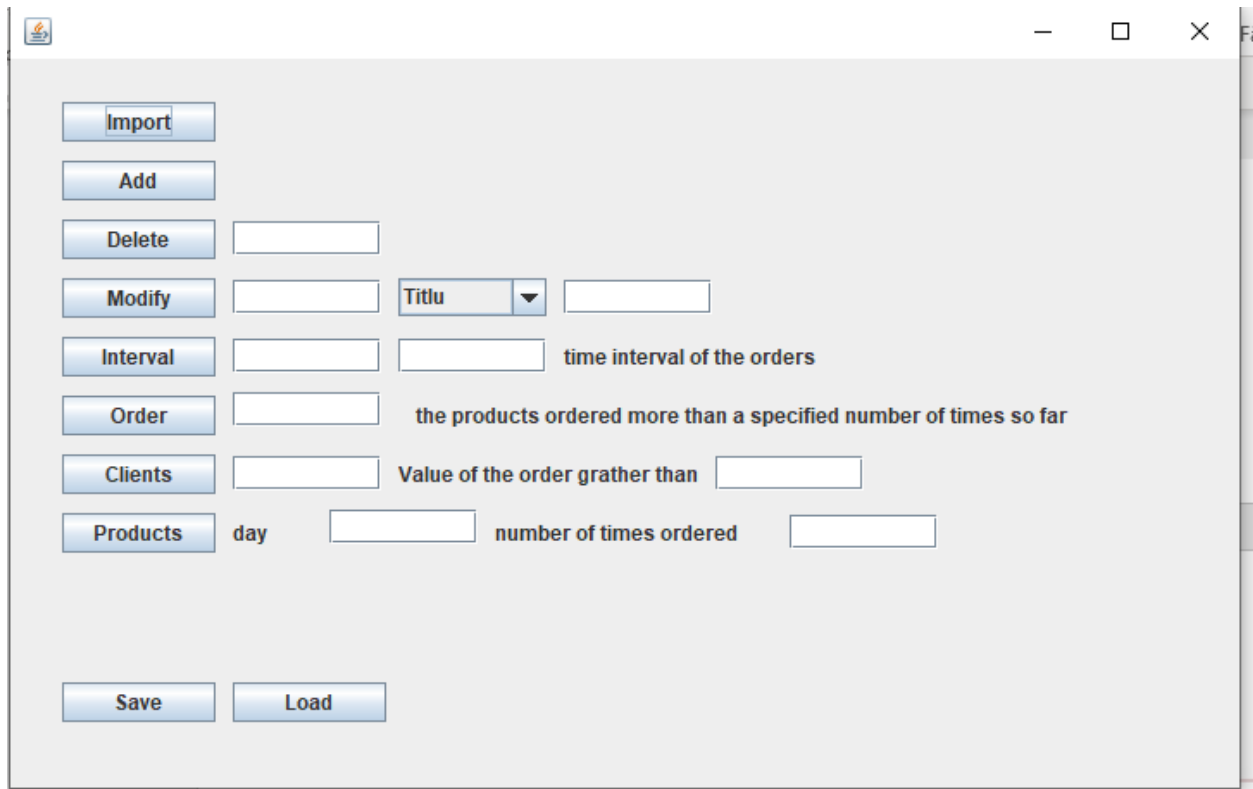
Iar aici este partea de buseniss logic cat si data logit se poate observa faptul ca sunt numeroase legaturi intre elemente si o multime de metode pe care le va apela partea de controller.

5. Interfata grafica

The screenshot shows a web application interface with a light gray background. It features a login and registration section with the following elements:

- User name:** A text input field.
- Password:** A text input field.
- Log in:** A blue button with white text.
- Register:** A blue button with white text.

Fiecarui utilizator I va aparea aceasta pagina in care va trebui ori sa isi introduca numele si parola ori sa isi creeze un cont nou dar acest cont poate fi doar de catre client. Dupa care in cazul in care cel logat este admin I va aparea.



Import

Add

Delete

Modify

Interval

Order

Clients

Products

Save

Load

Poate importa produsele din fisierul .csv sa adauge noi produse care va deschide o noua interfata. Adminul mai poate sterge anumite produse introducand numele produsului. Poate modifica parametrii din produse introducand numele selstcand ce camp doreste sa fie modificat si noua valoare pentru acest camp.

Poate afla toate comenzile dintr-un anumit interval orar. Intr-o companie este important sa vezi care sunt clientii fideli sau produsele cel mai bine vandute. Un alt aspect important pentru a realiza statistici si a nu fi luat prin surprindere este important sa stim in fiecare zi cate comenzi am avut. Iar la sfarsit inainte de a inchide aplicatia Adminul are posibilitatea de a salva tot ce s-a modificat pe parcursul unei zile.

Title	Rating	Calories	Protein	Fat	Sodium	Price
-------	--------	----------	---------	-----	--------	-------

Total:

Rating

Un client va vedea aceasta interfata unde are posibilitatea de a adauga produse in cos sau de a le scoate. In cazul in care nu stie numele produselor poate vedea lista cu toate produsele selectand **Produce**. Daca este satisfacut de produsele din cos poate plasa comanda. Sau daca doreste sa caute un produs dupa anumite criterii poate realiza acest lucru filtrand produsele dupa caracteristicile dorite.

6. Implementare

Pachetul business

Clasa **Role** are un singur camp **String** **role type** care reprezinta unul din cele 3 roluri si anume **ADMIN** **EMPLOYEE** **CLIENT**.

Clasa **User** are nume si parola pentru fiecare user cat si o instanta de tipul **Role** pentru a putea stii fiecare utilizator ce rol are. Iar ca metode am constructor **gettere** si **settere**.

Clasele **Admin** si **Client** extind clasa **User** si modeleaza un client sau un admin. Iar ca metode am constructor **gettere** si **settere**.

Clasa MenuItem reprezintă un item din meniu și toate campurile aferente și anume (Titlu, rating, calori, fat, protein, sodium, price) După care utilizând tehnica Composite Design Pattern pentru clasele MenuItem, BaseProduct and CompositeProduct. Iar pentru a notifica angajații am folosit Observer Design Pattern în momentul în care apare o comandă angajatul conectat la magazin va fi notificat în momentul în care clientul a pus comanda și despre datele comenzii pentru a o putea prepara și livra.

Design pattern-urile reprezintă soluții generale și reutilizabile ale unei probleme comune în design-ul software. Un design pattern este o descriere a soluției sau un template ce poate fi aplicat pentru rezolvarea problemei, nu o bucată de cod ce poate fi aplicată direct. În general pattern-urile orientate pe obiect arată relațiile și interacțiunile dintre clase sau obiecte, fără a specifica însă forma finală a claselor sau a obiectelor implicate.

Patternurile de tip Factory sunt folosite pentru obiecte care generează instanțe de clase înrudite (implementează aceeași interfață, moștenesc aceeași clasă abstractă). Acestea sunt utilizate atunci când dorim să izolăm obiectul care are nevoie de o instanță de un anumit tip, de crearea efectivă acestuia. În plus clasa care va folosi instanța nici nu are nevoie să specifice exact subclasa obiectului ce urmează a fi creat, deci nu trebuie să cunoască toate implementările acelui tip, ci doar ce caracteristici trebuie să aibă obiectul creat. Din acest motiv, Factory face parte din categoria *Creational Patterns*, deoarece oferă o soluție legată de crearea obiectelor.

Interfața DeliveryServiceProcessing are declarate principalele operații pe care le poate realiza atât clientul cât și adminul. Urmand ca și clasa DeliveryService să implementeze toate aceste metode. Explicații amanuntite se poate observa în cadrul aceste clase unde am explicat ce doresc să returnez ce elemente primesc cât și cum aș dori să fie aceste elemente.

Pachetul data

Acesta conține 2 clase FileWriter dar în implementarea mea nu am utilizat această clasă. Serializator care îmi va salva starea elementelor și o va încărca. Această clasă am legat-o la view în cadrul adminului care va putea salva toate operațiile făcute dar și după repornire să încarce toate datele salvate anterior. Salvarea elementelor se face în 3 fișiere diferite deoarece dacă se compromite unul să nu se compromită toate. Utilizatori, produse, comenzi toate acestea le salvez în 3 fișiere .txt diferite.

În cadrul acestui proiect am folosit pentru prima dată streamuri. The addition of the *Stream* was one of the major features added to Java 8. This in-depth tutorial is an introduction to the many functionalities supported by streams, with a focus on simple, practical examples. Simply put, streams are wrappers around a data source, allowing us to operate with that data source and making bulk processing convenient and fast.

Tot pentru prima dată am folosit serializarea care permite salvarea stării unui obiect pentru a putea fi încărcat după repornirea sistemului. Java provides a mechanism, called object serialization where an object can be represented as a sequence of bytes that includes the object's data as well as information about the object's type and the types of data stored in the object. After a serialized object has been written into a file, it can be read from the file and deserialized that is, the type information and bytes that represent the object and its data can be used to recreate the object in memory.

Most impressive is that the entire process is JVM independent, meaning an object can be serialized on one platform and deserialized on an entirely different platform.

Pachetul presentation

Interfata principala care apare tuturor este View unde ne este prezentat modalitatea de logare in aplicatie. Dupa care am definit clasa LoginController care va realiza instante de interfate in functie de ce utilizator se logheaza.

Pentru a modela mai frumos am realizat o noua clasa care modeleaza operatiile Adminului si in cadrul acestei clase am implementat operatiile pe care le realizeaza acesta. In cadrul acesui pachet nu am intampinat dificultati deoarece totul a venit de la sine dupa ce am avut o implementare corecta a logicii din spate.

7. Concluzii

Acest proiect a aprofundat lucrul cu interfetele grafice. Acum fiind mult mai bine intelese si mai usor de utilizat. Iar acum pentru prima data dupa ce aau fost prezentate aceste notiuni in cadrul cursului le-am folosit intr-un proiect. Si anume am folosit streamuri serializator si tehnici noi de programare. Acest proiect a fost unul din cele mai complexe proiecte pe care le-am realizat a trebuit sa ma documentez din diferite surse pentru a duce la bun sfarsit cerinta.

8. Rezultate

Am incercat sa testez toate cazurile de utilizare, unele cazuri nu le-am tratat cum ar fi introducerea unui string in loc de int dar aceasta modificare ar trebui facuta in cazul in care un astfel de proiect s-ar folosi intr-o companie cu adevarat. Dar cele mai multe erori identificate legate de logica algoritmului le-ma tratat pentru a ma asigura de corectitudinea algoritmului. Pentru a corecta unele erori am afisat mesaje in consola dar pana la urma toate aceste erori au fost rezolvate.

9. Bibliografie

<http://elf.cs.pub.ro/poo/laboratoare/design-patterns>