DOCUMENTATIE

TEMA *4*

NUME STUDENT: Florica Stefania

GRUPA: 30228

# 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

Obiectivul acestei teme consta in implementarea unei aplicatii care se ocupa cu operatiuni specifice unei firme de food delivery, simuland, totodata, interactiunea clientului cu un astfel de tip de aplicatie: acesta poate sa isi creeze un cont, poate sa se logheze, sa vada o lista cu meniul, sa filtreze produsele dupa anumite criterii alese de el si sa plaseze comenzi. Pentru fiecare comanda plasata se va genera un fisier .txt care reprezinta bonul fiscal cu informatii legate de pretul total al comenzii, clientul care a plasat comanda, data si ora plasarii comenzii, dar si o lista cu produsele selectate. Angajatii vor fi notificati de fiecare data cand un client plaseaza o comanda. Aministratorii au dispun de o functionalitate diferita. Ei pot sa introduca produse noi, sa le modifice pe cele deja existente sau sa le stearga din meniu pe cele care nu mai sunt disponibile pentru comanda. De asemenea, acestia pot crea produse compuse din mai multe alte produse in scopul introducerii unui produs de tipul “Meniul zilei”.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

* ***Analiza problemei***

Printre cerintele aplicatiei de food delivery se numara:

1. **Cerinte functionale:**
   1. Aplicatia trebuie sa permita clientului sa se logheze si sa creeze un cont nou in cazul in care nu este inregistrat deja.
   2. Aplicatia trebuie sa permita clientului sa filtreze produsele dupa numite criterii precum un cuvant-cheie, rating minim/maxim, numarul de calorii, de proteine, de grasimi, dupa cantitatea de sodiu, dar si dupa un interval de pret, [pret minim, pret maxim].
   3. Aplicatia trebuie sa permita plasarea unei comenzi de catre clientul care s-a logat cu succes in aplicatie, sa genereze un bon fiscal si sa trimita o notificare angajatilor cand comanda a fost plasata, pentru a incepe pregatirea acesteia.
   4. Aplicatia trebuie sa imprementeze functionalitatea pentru rolul de administrator al serviciului de catering; acesta trebuie sa poate sa adauge produse noi in meniu, sa le modifice pe cele existente, sa le stearga pe cele care nu mai sunt relevante, dar si sa creeze produse compuse.
   5. Aplicatia trebuie sa permita generarea de a patru tipuri de rapoarte din meniul administratorului, care sa ofere informatii legate de anumite statistici pentru comenzile plasate.
   6. Aplicatia trebuie sa avertizeze utilizatorul in cazul in care inputul introdus este invalid, dar si sa in informeze ca operatiunea selectata a fost executata cu succes.
2. **Cerinte non-functionale:**
   1. Aplicatia trebuie sa fie intuitiva si usor de folosit pentru orice tip de utilizator, fie ca dispune de o pregatire anterioara sau nu.
   2. Este de preferat ca aplicatia sa aiba o interfata cu utilizatorul cat de cat atractiva, care permite interactiunea eficienta si corecta cu functionalitatile acesteia.

* ***Scenarii de utilizare***

Acestea se pot fi impartite in doua categorii: scenarii uzuale de utilizare, car in care totul functioneaza conform planului, si scenarii critice, in care ceva nu decurge conform asteptarilor.

* **Scenariul uzual**:

Din prima interfata, utilizatorul poate alege daca este client sau administrator, urmand sa fie intampinat de o fereastra de logic, in cazul clientului, respectiv de o fereastra in care trebuie sa fie introdusa parola pentru accesarea meniului administrativ in cazul utilizatorului cu rol de administrator. Fereastra cu sumarul comenzilor plasate corespunzatoare angajatului va fi accesata doar atunci cand un client va plasa o comanda. Daca utilizatorul este un client si s-a logat cu succes, poate sa plaseze o comanda si sa utilizeze meniul de filtrare a produselor. Administratorul poate sa isi acceseze meniul daca introduce corect parola, ulterior putand genera rapoarte sau modifica meniul restaurantului.

De asemenea, este posibila executarea unui nou ciclu de utilizare/logare fara a reporni aplicatia.

* **Scenarii critice**:

Utilizatorul poate introduce date invalide de la tastatura, precum un sir de caractere in loc de numere, un numar negativ pentru valoarea unuia dintre field urile unui produs, un rating pentru un produs care nu este cuprins in intervalul [0,5]; in toate aceste situatii utilizatorul este avertizat ca inputul sau este incorect si i se va cere sa incerce introducerea corecta a datelor de cate ori este nevoie, pana cand vor fi indeplinite cerintele.

# Proiectare

In proiectarea aplicatiei, am avut in vedere respectarea conceptelor programarii orientate pe obiect, punand accentul pe abstractizare, mostenire, incapsulare si polimorfism. Fiecare clasa implementata este cat mai specifica si implementeaza cat mai putine functionalitati, cum intalnim, de exemplu, in cazul claselor de implementeaza design- pattern-ul Observer, MenuItem, BaseProduct si CompositeProduct.

Pentru structurarea proiectului, am folosit o arhitectura de tip Layered, care se refera la divizarea claselor in pachete corespunzatoare functionalitatilor lor: pachetul data contine clase care se ocupa cu operatiile de serializare, deserializare, scriere si citire din fisiere .csv; pachetul bll( Business Logic Layer) cuprinde clase ce se ocupa de functionalitatea aplicatiei si validarea datelor, iar in pachetul presenation se pot observa clasele care se ocupa de realizarea interefetelor cu utilizatorul si controlul acestora.

Cu scopul de a sustine si mai mult principiile programarii orientate pe obiect, am definit si o serie de interfete, pentru a defini functionalitatile fiecarui tip de utilizator. De exemplu, in pachetul bll se pot gasi 2 interfete, ClientOperations si AdminOperations, care definesc functionalitatile care vor trebui sa fie implementate in clasa DeliveryService, care se angajeaza sa ofere functionalitate concreta a acestor metode prin implementarea interfetei IDeliveryService, o interfata ce extinde cele doua interfete mentionate anterior.

De asemenea, pentru clasa DeliveryService s-a folosit paradigma Design By Contract, prin definirea invariantului validData(), metoda supraincarcata, folosit diferite combinatii de parametrii, in fucntie de metoda in care este apelata in cadrul unei asertiuni. Totodata, am folosit preconditii si postconditii pentru fiecare metoda unde acestea au fost necesare si relevante.

In ceea ce priveste structurile de date folosite, pentru stocarea clientilor si a produselor am folosit cate un HashSet pentru a asigura unicitatea elementelor ( am suprascris metoda hash() in fiecare din clasele din care au fost instantiate obiecte pentru popularea acestor HashSet uri pentru a asigura generearea unui hashcode corect pentru fiecare instanta).

Pentru stocarea instoricului de comenzi am folosit un HashMap<Order,ArrayList<MenuItem>>, unde cheia unui entry este reprezentata de un obiect de tipul Order, care contine informatii precum id-ul comenzii, id-ul cleintului care a plasat comanda, pretul total si data la care a fost plasata comanda. Valoarea entry-ului este data de un ArrayList care contine instante ale clasei MenuItem (BaseProduct sau CompositeProduct, clasa MenuItem fiind abstracta).

Pentru extragerea produselor din fisierul .csv s-au folosit Stream-uri. Acstea au fost introduse incepand cu Java 8 si permit programatorului sa efectueze operatii de tipul map, filter, reduce asupra obiectelor de tip Collection intr-un mod mai rapid si eficient.

# Implementare

Dupa cum am mentionat in sectiunile anterioare, logica interna a aplicatiei este implementata in cadrul claselor din pachetul bll. Astfel, clasa DeliveryService este cea mai complexa, ingloband cea mai mare parte a functionalitatii aplicatiei. La crearea unei instante din aceasta clasa, se apeleaza constructorul, in care se creeaza un obiect din clasa Serializer, dupa care se incarca in aplicatie datele referitoare la clienti, produse si istoricul comenzilor, prin deserializare. De asemenea, implementeaza interfata Subject (asa-zisa Observable) si are ca variabila-instanta o lista de obiecte de tipul Observer, instante ale clasei EmployeeView, care vor fi notificate de fiecare data cand a fost plasata o noua comanda. Design Patter-ul Observer ne permite sa comunicam din bll cu elemente din pachetul presentation prin intermediul interfetei Observer, compiler-ul neputand observa in mod direct acest lucru. Aceeasi idee am folosit-o si in cazul aparitiei pe ecran a unor pop-up-uri atunci cand utilizatorul introduce date gresite.

Tot in cadrul acestei clase am implementat si metodele pentru filtrarea produselor, operatiuni specifice clientului. Acestea au fost implementate folosind functionalitatile introduse odata cu Java 8, si anumte Stream-uri si expresii lambda. Am folosit operatii de tip filter, map, reduce pentru lucrul mai rapid asupra colectiilor mari de obiecte.

# Rezultate

In scopul testarii functionalitatilor aplicatiei, am efectuat numeroase teste, am adaugat comenzi in diferite zile din prisma a diferiti clienti pentru a testa generarea rapoartelor de catre administrator. In plus, la plasarea fiecarei comenzi se genereaza un fisier .txt cu rol de bon fiscal, putandu-se observa faptul ca fiecare comanda a fost plasata in mod corect. De asemenea, la fiecare comanda plasata, apare pe ecran pop-up-ul cu comezile in pregatire

# Concluzii

In urma implementarii acestui proiect, am deprins abilitati de a lucra o serie de concepte noi, cum ar fi serializarea si deserializarea, programarea functionala folosid Stream uri si expresii lambda. De asemenea, am avut ocazia sa inteleg conceptul de Design By Contract, dar si sa integrez anumite Design Pattern-uri in proiectul meu, precum Observable si Composite.

In ceea ce priveste interfata grafica cu utilizatorul, am intampinat o serie de asa-zise provocari peste care am reusit sa trec, cum ar fi comunicarea intre mai multe interfete, actualizarea instanta a tabelelor dupa modificarea, inserarea sau stergerea unui produs. Partea de GUI a fost mult mai complexa, dar am avut ocazia sa imi dezvolt abilitatile de a crea un front-end atractiv si usor de folosit pentru utilizator.

In ceea ce priveste dezvoltarile ulterioare, aplicatia poate merge in foarte multe directii. Printre acestea se numara:

* Introducerea unei optiuni de a alege cantitatea dorita dintr-un anumit produs selectat.
* O alta imbunatatire ar putea fi implementata in meniul administratorului, in cazul modificarii informatiilor despre anumite produse, sa nu fie nevoie sa se apese butonul Edit pentru fiecare item editat, ci sa se poata procesa dintr-o data toate modificarile efectuate asupra mai multor produse la apasarea butonului Edit.
* O alta dezvoltare ar putea consta in abilitatea clientului de a putea vedea produsele componente dintr-un CompositeProduct, nu doar prin concatenarea numelor in titlu, putand fi analizate si valorile nutritionale ale fiecarui produs in parte, nu doar per total.
* Se poate modifica logica aplicatiei, renunand la fereastra cu combobox prin care se face distinctia intre client si administrator, prima fereastra pe care o vede utilizatorul sa fie cea de Login. Astfel, administratorii si clientii ar putea fi modelati printr-o clasa User, care ar putea sa aiba un camp care sa aiba valori diferite in functie de rolul fiecaruia.

# Bibliografie

<https://www.youtube.com/watch?v=wiQdrH2YpT4>

<https://www.tutorialspoint.com/design_pattern/observer_pattern.htm>

<https://www.youtube.com/watch?v=2HUnoKyC9l0>

<https://www.youtube.com/watch?v=MskSpxNl-K8>