DOCUMENTATIE

Tema 4

**FOOD DELIVERY MANAGEMENT SYSTEM**

NUME STUDENT: FLOREAN EUSEBU-IOAN

GRUPA: 30223

# 

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

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

[3. Proiectare 4](#_Toc95297887)

[4. Implementare 6](#_Toc95297888)

[5. Rezultate **Eroare! Marcaj în document nedefinit.**](#_Toc95297889)

[6. Concluzii 8](#_Toc95297890)

[7. Bibliografie 8](#_Toc95297891)

# Obiectivul temei

Obiectivul acestei teme este crearea unei aplicatii numita “FOOD DELIVERY MANAGEMENT SYSTEM” care permite crearea unui cont de logare prin apasarea butonului de Register. Acest cont poate sa fie de client, admin sau angajat. Dupa creearea contului, user ul se poate loga cu credentialele sale in propiul cont. Odata introduse datele, se va deschide fereastra corspunzatoare. In cazul logarii in contul de admin, pe ecran va aparea o fereastra care permite importarea tuturor produselor din fisierul csv, adaugarea produselor in meniu unde vor fi vizibile pentru client, adaugarea unui nou produs, stergerea si editarea acestuia. De asemenea, acest cont permite crearea de meniuri compuse care pot fi vazute de catre client ca un produs normal.

Fereastra de client permite vizualizarea intregului meniu, filtrarea datelor din acesta dupa mai multe criterii, adaugarea produselor in cos si efectuarea unei comenzi.

Proiectuls se bazeaza pe operatia de serializare prin scrierea in fisiere auxiliare a datelor, ceea ce ofera o stocare in siguranta a tuturor modificarilor efectuate de catre client.

Serializarea este o metoda prin care se pot salva, intr-o maniera unitara, datele impreuna cu signatura unui obiect. Folosind aceasta operatie se poate salva intr-un fisier, ca sir de octeti, o instanta a unei clase, in orice moment al executiei. De asemenea, obiectul poate fi restaurat din fisierul in care a fost salvat in urma unei operatii de serializare.

Interfata Serialize nu are nici o metoda, ea da doar posibilitatea de a specifica faptul ca se doreste ca o anumita clasa sa poata fi serializata. Declaratia unei astfel de clase ar fi :class ClasaSerializabila implements Serializable

In urma serializarii obiectele sunt pot fi salvatr intr-un fisier, in acelasi fisier putand fi salvate si mai multe obiecte. Operatiile de intrare iesire la nivelul obicetelor se realizeaza prin intermediul unor fluxuri de obiecte, implementate de clasele ObjectInputStream si ObjectOutputStream.

In analiza problemei pornim de la cele 3 tipuri de utilizatori: client, administrator si angajat

si de la atributiile fiecaruia in parte.

Administatorul poate efectua urmatoarele operatii:

-sa importe setul initial de produse cu care va fi populat meniul dintr-un fisier .csv

- sa gestioneze produsele din meniu prin adaugarea/stergerea/modificarea acestora si crearea

unui meniu nou compus din produsele existente

-sa genereze rapoarte despre comenzi cu urmatoarele criterii:

o Intervalul orar al comenzilor – se vor selecta toate comenzile plasate intr-un

anumit interval orar indiferent de zi

o Produsele comandate de mai multe ori de cat un numar specificat

o Clientii care au comandate de un numar de ori mai mare decat un numar

specificat si valoarea comenzii depaseste o suma specificata

o Produsele comandate intr-o anumita zi si de cate ori au fost comandate

Clientul poare sa:

-se inregistreze in aplicatie sau sa se autentifice folosind un username si o parola

-sa vada lista de produse din meniu

-sa caute produse dupa diverse criterii

-sa plaseze comenzi adaugand diverse produse: pentru fiecare comanda se va specifica data si

ora, pretul total al comenzii si se va genera o factura continand detaliile comenzii

Angajatul

-este notidicat de fiecare data cand se plaseaza o noua comanda de catre un client pentru a putea sa livreze produsele

# Analiza problemei, modelare, scenarii, cazuri de utilizare

In momentul rularii aplicatiei, pe ecran vor aparea fereastra de LOGIN, care este prevazuta cu un buton de Register. La actionarea butonul Register, se va putea crea un cont de admin, client sau angajat.

In aceasta interfata exista butonul de back, care ne va redirectiona in fereastra de login. Dupa introducerea credentialelor se va deschide fereastra corespunzatoare:

CLIENT:

Aceasta fereastra contine 3 mari butoane care permit vizualizarea produselor din meniu, cautarea acestora dupa criterii si efectuarea de comenzi. Pentru a crea o comanda, se va selecta un produs din meniu, se actioneaza addProduct, operatia care va introduce produsul in cos, iar pentru a finaliza comanda se apasa butonul MAKE ORDER. Prin finalizarea comenzii, intelegem inregistrarea comenzii cu toate detaliile, inclusive incrementarea cantitatii vandute pentru fiecare produs. Pentru a ne intoarce la fereastra de login, vom actiona buton de log out.

O imagine care conține masă

Descriere generată automat

ADMINISTRATOR:

Aceasta fereastra prezinta 2 mari tabele in care se vor stoca datele importate din fisierul csv, vizibile doar pentru admin si respective produsele vizibile pentru client, cele din meniu. Pentru a imprta produsele, se introduc indicia din fisierul xcel al produselor dupa care se actioneaza import products.

Pentru a adauga produsul in meniu, se selecteaza un produs din cele importate, dupa care se apasa add Product To Menu. In acest moment, produsul este vizibil atat in tabelul din partea stanga, cat si in meniul din contul de client.

Pentru a adauga un nou produs creat de admin, care nu se gaseste in fisierul csv, sunt prezente 7 text fielduri prin care se preiau informatiile despre acestea, se creeeaza obiectul urmand ca apoi sa fie introdus in meniu. De asemenea, se poate edita fiecare produs prin selectarea acestuia, introducerea datelor dorite in text fieldurile corespunzatoare si actionarea butonului de edit.

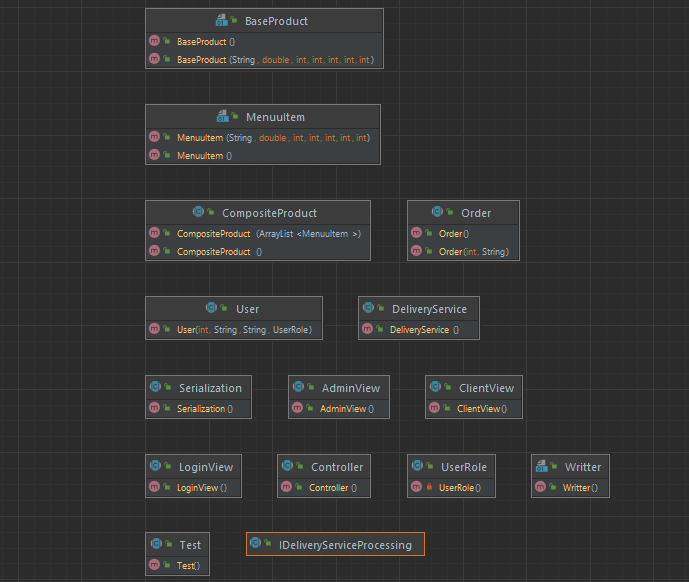
Pentru a crea si implicit adauga un produs compus, se actioneaza butonul de create composed product prin care se vor selecta in mod random 4 produse din fisier care vor fi inserate sub forma unui produs compus in meniu.

O imagine care conține masă

Descriere generată automat

# Proiectare

Aplicatia este creata in InteliJ. In Java, codul este structurat in 4 pacheta: BussinessLayer care contine clasele de baza prin care sunt organizate toate produsele: BaseProduct, MenuuItem, CompositeProduct, DataLayer care contine clasa DeliveryService in care s-au implementat toate metodele, Order, Serialization, User, UserRole, Writter si IDeliveryService, PresentationLayer in care sunt prezente clasele care tin de interfata grafica precum si clasa Controller al carei rol este de a lega toate ferestrele intre ele.



Mai sus este ilustrata o imagine care contine toate clasele din aplicatia creata, impreuna cu constructorii lor. Am evitat ilustrarea tuturor variabilelor si ale metodelor deoarece aplicatie este foarte complexa si schema ar fi devenit foarte greu de inteles.

Swing este un cadru pentru Java , aparținând Java Foundation Classes (JFC) și orientat spre dezvoltarea de interfețe grafice . O parte din clasele de cadru Swing sunt implementări de widget-uri (obiecte grafice), cum ar fi casete de text, butoane, panouri și tabele.

Biblioteca Swing este utilizată ca bibliotecă oficială pentru crearea de interfețe grafice în Java. Este o extensie a setului de instrumente pentru fereastra abstractă anterioară. Principala diferență dintre cele două este că componentele Swing sunt scrise complet în cod Java .

# Implementare

Implementarea intregului proiect se bazeaza pe noua notiune de serializare, dar si de parcurgerea colectiilor prin expresiile lambda, cu ajutorul forEach() method.

Serializarea este o metoda ce permite transformarea unui obiect într-o secventa de octeti din care sa poata fi refacut ulterior obiectul original. Cu alte cuvinte, serializarea permite salvarea într-o maniera unitara a datelor împreuna cu signatura unui obiect pe un mediu de stocare a informatiei extern programului. Procesul invers de citirea a unui obiect serializat pentru a-i reface starea originala se numeste deserializare. Intr-un cadru mai larg, prin serializare se întelege procesul de scriere/citire a obiectelor

Fiecare dezvoltator trebuie să învețe să utilizeze noi caracteristici de programare, mai ales acum că versiunea 8 a atins nivelul maxim de utilizare. Expresiile Lambda sunt o caracteristică nouă și importantă inclusă în Java 8. Acesta oferă o modalitate clară și concisă de a reprezenta o metodă care îmbunătățește colecția bibliotecii prin simplificarea iterației, filtrarea și extragerea datelor. Noile caracteristici de concurrency îmbunătățesc performanța în medii cu mai multe nuclee. În Java, toate colecțiile au o metodă forEach, moștenită de la interfața super-interfață Iterable. Acesta a existat de la Java 5, a fost extins la versiunea 8. Metoda forEach iterează peste toate elementele colecției și aplică funcția fiecărui element.

O expresie lambda este o funcție anonimă care oferă un comportament parametrizat. Se compune dintr-o listă de opțiuni de returnare și de excepții care sunt selectate. O instanță a unei expresii lambda java poate fi atribuită oricărui Iterable care se potrivește cu definiția, cu condiția să fie funcțională.

Clasa Delivery Service contine toate metodele de care am avut nevoie in implementarea proiectului. Acestea sunt:

IMPORT PRODUCTS: Am folosit Stream uri cu ajutorul carora ne accesam fisierul csv present in directorul proiectului nostrum. Se va citi fiecare linie, se va face split dupa virgula pentru a obtine fiecare atribut de care avem nevoie pentru a creea obiectul. Vom adauga obiectul la lista noastra de obiecte dupa care vom serialize lista, urmand ca la urmatorul pasa sa deserializam din nou produsele.

CREATE ORDER: metoda prin care are loc crearea unei noi comenzi. In cadrul acestei metode se creeaza un Map de Order si un array de obiecte ale clase MenuuItem. Pentru fiecare comanda se va actualiza cantitatea vanduta a fiecarui produs, inclusiv numarul de comenzi efectuate de catre fiecare user.

GENERATE REPORT 1: se va genera un raport cu toate comenzile care au fost efectuate intr-o anumita zi si respective in intervalul orar ales de admin.

GENERATE REPORT 2: se va genera un raport cu toate produsele care au fost comandate de un numar de ori mai mare ca si numarul introdus de catre administrator.

GENERATE REPORT 3: se va genera un raport in care vor fi afisati userii care au comandat un numar de comenzi mai mare ca si x si cu o valoare mai mare decat y, unde datele x si y sunt alese de catre user.

GENERATE REPORT 4: se va genera un raport cu toate produsele comandate in ziua aleasa de admin si se va afisa atat numele acestora, cat si cantitatea vanduta.

# Concluzii

In urma dezvoltarii acestei aplicatii numite **FOOD DELIVERY MANAGEMENT SYSTEM**

am deprins o multitudine de informatii in ceea ce priveste atat programarea orientate pe obiecte, cat si limbajul de programare Java. Sunt de parere ca este foarte benefic pentru student sa aiba de realizat astfel de proiecte in primul rand pentru ca acestea reprezinta lucruri practice care te dezvolta si te pregatesc pentru cariera in programare si in al doilea rand intrucat aceasta aplicatie reprezinta un proiect personal care poate fi atasat la CV-ul fiecaruia.

De asemenea, implementarea acestei aplicatii m-a ajutat sa inteleg mai bine cum pot crea aplicatii care pot fi folositoare in viata de zi cu zi si a chiar mi-a dat idei de viitoare aplicatii pe care as putea sa le implementez in viitor.( modificarea acestei aplicatii si crearea posibilitatii de a fi descarcata si rulata pe telefon).

Un alt beneficiu al dezvoltarii acestui preiect este reprezentat de faptul ca am reusit mult mai bine sa inteleg idea dintre partea de front-end si back-end. Aceasta aplicatie a fost create folosind ambele strategii. Front-End-ul este partea aplicației care interacționează în mod direct cu utilizatorul. Cunoscută și sub numele de partea client, aceasta se ocupă de aspectul și funcționalitatea interfeței aplicatiei.Elementele implementate de către un Front-End developer aplicației includ, dar nu sunt limitate la: stilul, paleta de culori, imaginile, butoanele, meniurile de navigație, comportamentul elementelor vizibile de către utilizator. Back-End-ul este partea aplicației care nu interacționează în mod direct cu utilizatorul. Cunoscută și sub numele de partea server, aceasta se ocupă de stocarea și managementul datelor prin intermediul bazelor de date, logica de business a aplicației și scrierea de API-uri – interfețe programatice ce expun date din Back-End către aplicația Front-End, și așadar, către client.

Deoarece aplicația Front-End este expusă în mod direct clientului, marea parte a logicii de business a aplicației este scrisă în Back-End, pentru a putea evita accesarea logicii aplicației de către persoane din exterior. Comunicarea cu bazele de date se face exclusiv în Back-End, pentru a ne feri de atacuri de tip malware la nivelul bazei noastre de date.

Java este un limbaj multifuncțional bazat pe computer care este creat pe ideologia scrierii odată rulat oriunde. A fost inventat de James Gosling și Sun Microsystems în 1991 și la acea vreme singura lor intenție era de a îmbunătăți reutilizarea codului. Java este un limbaj de programare orientat pe obiecte bazat pe limbaje C și C++, cu toate acestea, este un limbaj de programare îmbunătățit și simplificat, cu capacitatea de a gestiona erorile de programare. Java este utilizat în mai multe scopuri, cum ar fi dezvoltarea web, dezvoltarea de aplicații mobile etc.

# Bibliografie

<https://ro.wikipedia.org/wiki/Fir_de_execu%C8%9Bie>

<https://examples.javacodegeeks.com/java-thread-example/>

https://dsrl.eu/courses/pt/

<https://ro.wikipedia.org/wiki/Model-view-controller>