

Food Delivery

Documentatie

Andrei Burian

Grupa 30229

Cuprins

1. Obiectiv ............................................3
2. Studiul problemei .............................3
3. Proiectare .........................................8
4. Implementare...................................8
5. Imbunatatiri ...................................11
6. Concluzii .........................................11
7. Bibliografie ....................................11
8. Obiectiv

Obiectivul acestei lucrari de laborator este de a proiecta, implementa si testa o aplicatie cu interfata grafica (UI) care gestioneaza comenzile si meniul unei firme de catering.

Interfata grafica are rolul de a face interactiunea dintre utilizator si aplicatie, cat mai simpla si intuitiva. Aceasta aplicatie va avea 4 GUI-uri, una pentru autentificarea utilizatorului si restul de 3 sunt pentru fiecare utilizator in parte: administrator, angajat si client.

De fiecare data cand este plasata o comanda este creat un fisier .txt in care sunt specificate detaliile comenzii.

* 1. Obiective secundare

Primul obiectiv este acela de a crea clasele produselor corespunzatoare, si anume o clasa abstracta care este extinsa de cele 2 tipuri de produse: produs de baza si produs compus.

Al doilea obiectiv a fost crearea unei clase pentru comenzi (Order) unde sunt stocate datele unei comenzi.

Al treilea obiectiv a fost conceperea clasei pentru utilizator (User) si unei clase de enum pentru fiecare tip de utilizator (admin, employee si customer).

Al patrulea obiectiv este implementarea interfetei IDeliveryServiceProcessing prin intermediul clasei DeliveryService, care gestioneaza toate operatiile aplicatiei.

Al cincilea obiectiv a fost implementarea interfetelor grafice, care au fost 4 in total, unele mai complexe (cea de administrator) si altele mai putin complexe (cea de employee).

Al saselea si ultimul obiectiv a fost realizarea si implementarea clasei de serializare, care salveaza modificarile adusei bazei de date de fiecare data cand se inchide aplicatia.

1. Studiul problemei

2.1 Analiza problemei

Cea mai ampla problema a fost implementarea propriu-zisa, faza de proiectare si modelare a aplicatiei fiind necostisitoare din punctul de vedere al timpului. Cele mai ample (sau dificile) probleme la implementare au fost realizarea clasei DeliveryService, unde ajutandu-ne de stream-uri trebuiau implementate opratiile cu comezi, utilizatori sau cu meniul in sine, dar si realizarea claselor pentru produsele din meniu . Un alt obstacol a fost realizarea interfetelor grafice, care afiseaza tabelele in timp real, dar pot si sa le modifice. Fiind si 4 la numar si unele destul de complexe a fost dificil sa se faca functionabile fiind compuse din atatea elemente. Pe langa faptul ca trebuie sa afiseze tabele in timp real, in interfata de administrator trebuie sa se poata afisa mai multe tipuri de tabele, cu coloane si date diferite.

2.2 Modelare

Clasele din pachetul BusinessLayer reprezinta datele aplicatiei, adica ultilizatorii, comenzile, produsele dar si clasa DeliveryService, care face legatura intre toate acestea si le gestioneaza.

In pachetul DataLayer avem clasa Serializator care face serializarea datelor astfel incat atunci cand redeschidem aplicatia sa se incarce toate datele obtinute anterior.

Pachetul PresentationLayer contine toate clasele aferente interfetelor grafice, si anume cate o clasa de view pentru fiecare GUI si cate o clasa de controller, tot asa, pentru fiecare GUI.

In cele din urma, mai este pachetul start care contine clasa Start. Acesta este practic main-ul aplicatiei, din aceasta clasa pornindu-se toata aplicatia.

2.3 Use case

Cand se porneste aplicatia va aparea o singura interfata grafica, cea de autentificare. Aici, se selecteaza tipul de utilizator care doreste sa foloseasca aplicatia si se autentifica sau in cazul clientilor pot sa isi creeze un cont daca nu au mai folosit aplicatia si pe urma se pot autentifica.

In interfata corespunzatoare clientului se pot realiza 2 mari operatii: cautarea de produse si plasarea unei comenzi. Pentru a cauta produse sunt mai multe campuri unde se pot introduce valorile dorite si dupa aceea se apasa butonul „Search”. Daca se doreste resetarea criteriilor de cautare se apasa „Reset filters” si campurile cautarii se vor goli, iar tabelul va aparea complet, nefiltrat. Pentru realizare unei comenzi trebuie sa se adauge pe rand id-ul produselor in caseta corespunzatoare si sa se apese „Add to cart” pentru adaugarea in cos (doar un produs poate fi adaugat in cos o data). Cand s-au adaugat produsele dorite se apasa butonul „Place order” si comanda va fi plasata.

In GUI-ul pentru administrator mai intai trebuie sa se selecteze operatia dorita: importarea de produse, manipularea produselor sau generarea de rapoarte. In primul caz se apasa pe butonul de import daca se doreste importarea produselor dintr-un fisier .csv. Daca se doreste manipularea de produse atunci trebuie selectata din nou ce fel de operatie dorim sa se execute pe produse: adaugarea de produs nou, modificarea unui produs existent, stergerea unui produs sau adaugarea unui produs compus. In functie de alegere vor aparea si disparea campuri astfel incat sa se afiseze doar campurile necesare operatiei selectate. Pentru a adauga un produs se completeaza toate datele produsului in casutele corespunzatoare si se apasa butonul „Add product”. Pentru a modifica un produs se scrie id-ul produsului pe care dorim sa il modificam, selectam campul pe care vrem sa il schimbam si scriem noua valoare. Pe urma se apasa butonul „Edit” si produsul se modifica in tabel. Daca se doreste stergerea unui produs se scrie id-ul produsului pe care vrem sa il stergem si se apasa butonul „Delete”. La adaugarea unui produs compus trebuie sa se specifice pe rand id-urile produselor pe care vrem sa le adaugam la noul produs. De fiecare data se apasa „Add product” (la fel cum se plasa o comanda si interfata clientului), iar cand s-au adaugat produsele dorite se scrie un titlu si se apasa „Create composite product”.

Administratorul mai poate sa genereze rapoarte. Pentru a fi afisate aceste rapoarte se completeaza campurile corespunzatoare raportului care se doreste a fi generat si se apasa butonul de dedesubt de campuri pentru a se efectua cautarea.

Interfata angajatului este foarte minimalista, acesta fiind doar notificat de fiecare data cand un client pune o comanda.

Datele de intrare din interfete trebuie sa respecte cateva cerinte, cum ar fi:

* Trebuie sa coincida cu tipul de date care se asteapta, adica daca trebuie introduse numere, nu este voie sa se introduca litere si viceversa (chiar daca cifrele se pot insera ca text in tabel)
* Nu este permisa introducere de spatii inainte sau dupa numere
* Numerele trebuie sa respecte logica aplicatiei, de exemplu cand modificam un camp id-ul produsului pe care dorim sa il modificam trebuie sa se regaseasca in tabela.

Totusi, in ciuda acestor restrictii aplicatia ofera flexibilitate in cazul unei erori din partea utilizatorului (de exemplu introducerea de spatii sau litere in loc de cifre), aplicatia nu va porni, utilizatorul avand ocazia sa reintroduca corect datele, dupa care aplicatia va porni. Mai mult, daca la autentificare apar probleme acestea vor fi semnalate in interfata (de exemplu daca se incearca autentificare cu date gresite sau inregistrarea cu un username deja folosit) sau daca atunci cand cream un produs compus incercam sa adaugam un produs care nu exista in baza de date se va afisa un mesaj de eroare.

Aceasta este interfata de autentificare:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, table

Description automatically generated with medium confidenceIn acest moment utilizatorul trebuie sa se autentifice in aplicatie sau sa isi creeze un cont inainte, in cazul clientilor. Dupa autentificare va aparea una din urmatoarele ferestre in functie de tipul de utilizator:

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, table

Description automatically generatedGraphical user interface

Description automatically generated with medium confidenceGraphical user interface, table

Description automatically generatedInterfata administratorului poate lua mai multe forme, in functie de operatia selectata:

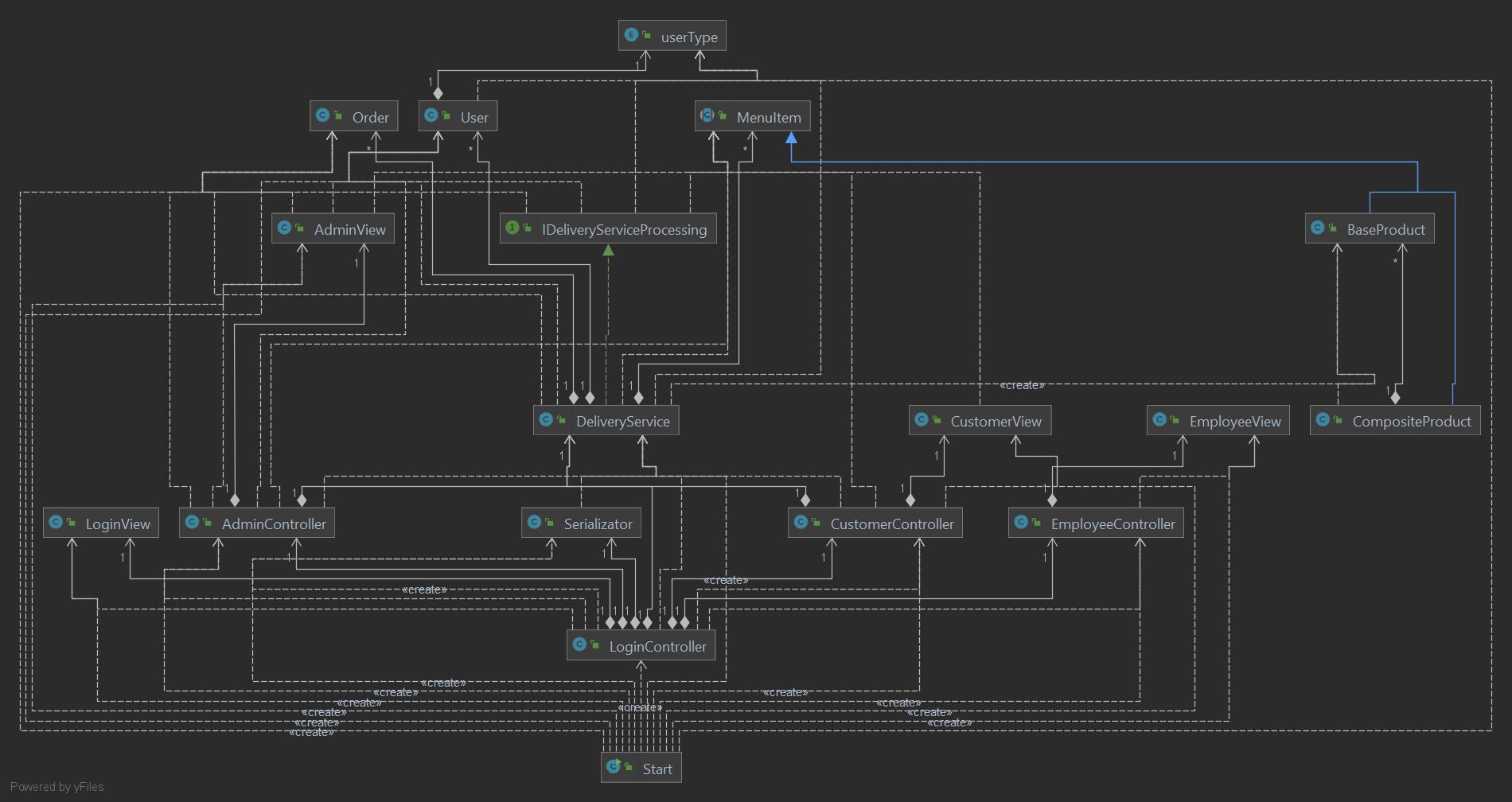
In acest moment, se pot efectua operatiile prezentate mai sus dupa pasii specificati.

2.4 Cazuri de utilizare

Aceasta aplicatie are 3 (de fapt 2) cazuri de utilizare in functie de tipul de utilizator, dupa ce se realizaeaza autentificarea:

1. Admin: se selecteaza operatia dorita si se importa sau se completeaza campurile corespunaztoare si se apasa un buton pentru a se ajunge la rezultatul dorit.
2. Angajat: acesta practic doar asteapta ca un client sa puna o comanda.
3. Client: poate cauta produse dupa anumite criterii, pe care le poate adauga in cos. Cand a terminat de adaugat in cos poate plasa comanda.
4. Proiectare

Proictul a fost realizat folosind o arhitectura de tip layered, avand pachetele BusinessLayer, PresentationLayer, DataLayer si Start.

 Mai jos este diagrama UML a proiectului:

1. Implementare

4.1 MenuItem

Este o clasa abstracta care contine campurile id, idCount, orderedFor. Aceasta clasa mai contine prototipurile functiilor care trebuie implementate in clasele care extind aceasta clasa. Mai contine si getter-i pentru campurile mentionate mai sus dar si niste functii pentru setarea id-ului si incrementarea variabilei orderedFor, pentru a tine cont de cate ori a fost comandat acel produs.

4.2 BaseProduct

Extinde clasa MenuItem si contine campurile title, rating, calories, protein, fat, sodium si price. Exista getter-e si setter-e pentru fiecare camp dar si implementarea metodei equals si hashCode pentru functia distinct(), pentru a putea filtra si elimina produsele care au acelasi titlu.

4.3 Composite product

Contine aceleasi campuri cu clasa BaseProduct, dar mai are in plus o lista de BaseProduct unde sunt memorate produsele din care este compus. In constructor primeste ca parametru o lista de BaseProduct, aici calculandu-se toate atributele, rating-ul facand o medie aritmetica, iar celelalte se aduna doar. De asemenea, si aici se implementeaza metodele equals si hashCode pentru buna functionare a functiei distinct().

4.4 Order

Aceasta clasa este compusa din campurile clientUsername, orderDate, price, idCount si id. Ultimele 2 sunt folosite pentru asignarea id-ului corespunzator. Primul este username-ul clientului care a plasat aceasta comanda, a doua este data si al treilea este pretul comenzii. Si aici se implementeaza metoda equals si cea de hashCode pentru maparea corespunzatoare intr-un hashMap, acolo unde un obiect de tip Order este key-ul map-ului.

4.5 User si userType

Clasa user contine doua String-uri, username si password, un obiect de tip userType, care este un enum de fapt, si un int care tine minte pentru clienti de cate ori a comandat acel client (timesOrdered). userType este un enum care contine campurile ADMIN, EMPLOYEE si CLIENT. Ne folosim de acest enum pentru a putea identifica fiecare client ce tip de utilizator este. Metodele de aici sunt getter-i pentru fiecare camp si o metoda care incrementeaza variabila timesOrdered, doar pentru utilizatorii clienti care pun o comanda.

4.6 DeliveryService

Aceasta clasa este inima acestui proiect, pentru ca aici se executa toate operatiile legate de comenzi, meniu si utilizatori. Campurile de aici sunt menuItems, un array de MenuItem in care se tin minte practic toate produsele din meniu, o lista de Useri si un hashMap care are ca si key un obiect Order si value o lista de MenuItem.

De asemenea, aceasta clasa implementeaza interfata IDeliveryServiceProcessing.

O metoda este readCSV care citeste dintr-un fisier .csv. Mai intai se creaza o lista de MenuItems care este filtrata folosind functia distinct() pentru a scoate elementele care au acelasi titlu, pentru ca sunt duplicate. Dupa aceea se sorteaza lista dupa titlu, se seteaza id-ul produselor si la final concatenam lista rezultata cu lista curenta de MenuItems din clasa si filtram elementele cu acelasi titlu.

Mai avem metoda de adaugare de produs in lista de produse. Mai intai se verifica sa nu existe un produs cu acelasi nume, pe urma se adauga in lista si se sorteaza lista din nou dupa titlu.

Metoda de adaugare de comanda primeste ca parametru un obiect de tip Order si o lista de MenuItems. Se adauga in hashMap un entry cu key-ul Order si valoarea lista primita ca paramtru. Se incrementeaza numarul comenzilor plasate de catre client si pe urma se scrie intr-un fisier .txt chitanta comenzii. In acelasi timp se si incrementeaza pentru fiecare produs din lista numarul de dati cat a fost comandat.

4.7 LoginView/AdminView/EmployeeView/CustomerView

Aceste clase reprezinta interfetele grafice ale acestei aplicatii. Au fost realizate folosindu-ma de bibliotecile java.awt si java.swing.

Toate 3 isi creeaza elementele interfetei la inceput ca si campuri ale clasei. Iar pe urma, in constructor se atribuie elemente JPanel-urilor care pe urma sunt adaugate in JFrame. Mai trebuie amintit si faptul ca elemente cheie din interfata au getter-e si setter-e, cum ar fi butoanele de selectie a operatiei, text field-urile, tabelul si panel-urile.

Mai avem si metode care populeaza un JTabel cu datele primite si returneaza acel tabel.

4.8 LoginController

Aceasta clasa este controller-ul clasei LoginView, practic este logica din spatele ferestrei de login. Campurile clasei sunt un LoginView, un DeliveryService si cate un controller pentru view-urile utilizatorilor (admin, employee si customer). Se creeaza si se implementeaza cate un action listener pentru radio buttons dar si pentru butoanele de login si register. De asemenea mai avem si un WindowListener pentru frame-ul view-ului. Cand acesta este inchis aplicatia se va termina si folosindu-ne de acest WindowListener serializam datele din obiectul DeliveryService si le scriem intr-un fisier pentru a fi accesat mai tarziu.

4.9 AdminController/EmployeeController/CustomerController

Aceste clase implementeaza logica din spatele ferestrelor de admin, employee si customer. In AdminController si in CustomerController avem action listener-i pentru fiecare buton pentru a putea realiza modificarea interfetei sau a operatiilor care trebuie efectuate pe baza de date (DeliveryService).

Mai mult, in AdminController si in CustomerController avem metode care actualizeaza tabelele din interfetele aferente apeland functiile de populare de tabele din view-uri si pe urma modificand interfata si integrand tabelul obtinut din functie.

4.10 Serializator

Nu contine niciun camp, doar doua metode: serialize si deserialize. Prima meotda se apeleaza atunci cand se inchide fereastra si se salveaza practic continutul bazei de date (DeliveryService). Functia de deserialize se apeleaza in main de fiecare data cnad se porneste aplicatia si inceraca sa citeasca un obiect de tip DeliveryService dintr-un fisier.

1. Imbunatatiri

Acest proiect este departe de a fi perfect, dar intotdeauna exista loc de imbunatatire. O imbunatatire considerabila ar fi semnalarea tuturor erorilor generate de introducerea de date gresite in interfete. O alta imbunatatire ar fi cosmetizarea interfetelor grafice astfel incat sa arate mai bine.

1. Concluzii

Acest proiect m-a ajutat sa inteleg cum sa lucrez pe baza unei arhitecturi de tipul layered architecture, dar si cum sa fac o aplicatie cu login.

1. Bibliografie

* <https://www.tutorialspoint.com/index.htm>
* <https://www.w3schools.com/>
* <https://www.javatpoint.com/>
* <https://howtodoinjava.com/>
* <https://beginnersbook.com/>