

FUNDAMENTAL PROGRAMMING TECHNIQUE

DELIVERY SERVICE – ASSIGNEMENT 4

DIACONU ANDREI , 30227

CUPRINS

| | |
|---------------------------------|----|
| Introducere..... | 3 |
| Analiza Problemei | 4 |
| Proiectare si Implementare | |
| Package Presentatjon Layer..... | 5 |
| Package Business Layer..... | 6 |
| Concluzii..... | 12 |
| Bibliografie..... | 13 |

INTRODUCERE

Daca tema anterioara ne introducea in lumea bazelor de date si lucrul cu acestea, de data aceasta nu folosim baze de date pentru la fiecare rulare noua a proiectului datele nu se pastreaza . Prin date ne referim la useri , comenzi , produse , rapoarte

De data aceasta atentie si focusul ni se va indrepta in mai multe directii . Fiind o tema complexa vom invata despre HashMap-uri ,Set-uri , Liste, etc. Fiecare structura de date va retine una din piesel de baza ale aceste probleme : Useri , Meniu sau Comenzi

La o privire de ansamblu ne gandim la HashMap ca la un HashTable care mapaeaza cheile cu valorile. In aceeasi ordine de idei se va folosi o hashcode-ul pentru a determina pozitia in astfel de structura a unei valori propriu zise.

Ideea de hashing se refera la distribuirea perechilor key-value , insa cerinta cea mai importanta este gasirea unei functii hashcode capabile de o distribuire cat mai uniforma. O distribuire neuniforma cauzeaza multe coliziuni si automat creste costul operatiilor .

La o privire mai obiectiva asupra problemei vom construi un astfel de HashMap care va conveni prin un obiect de tip Order ca si key (cu ClientID, orderID,Data)iar ca valoare o lista (mai exact un Set) de obiecte de tip MenuItem care vor reprezenta parcat impreuna comanda si ce continue aceasta.

Cum am amintit si mai sus este foarte important modul de aflare a hashcodului car in cazul de fata suma dintre idClientului , idComenzii(care este unic) si hashCode-ul datei actuale(la care s-a facut comanda) . Vor exista astfel 0 coliziuni

Legat de celelalte tipuri de structuri de date ii vom retine intr o lista inlantuita iar produsele intr-un simplu Map(BaseProduct).

Obiectivul temei este de a realiza un sistem de gestiune al comenzilor pentru un restaurant si livrarea acestora. Sistemul va beneficia de 3 functii : Admin, Client si Employee fiecare avand cate un rol specific in povestea aceasta

Clientul este cel ce poate comanda produse dupa ce studiaza meniul , sau il filtreaza fiind capabil prin functiile de search (afterTitle si afterPrice). Odata comandat intra in actiune Employee

Employee doar asteapta logat in aplicatie sa fie notificat prin Observer de plasarea unei comenzi

Admin-ul este cel ce populeaza meniul ce l vede clientul. Si care maim ult de atat poate adauga produse noi , sau sa le modifice pe cele existente sau chiar sa le stearaga. Un composite product este un produs create strict de admin si continue mai multe produse simple din meniu. Referitor la rapoarte ce se pot afla intr o frima de delivery acesta le poatea afla foarte usor

Exista si obiective secundare care prevad incarcarea produselor intiale , salvarea datelor de lucru in Set, HashMap-uri si Liste, JavaDoc ,Stream-uri si lucrul cu Observer/Observable

ANALIZA PROBLEMEI

Pentru o implementare fluida a acestui proiect am ales sa structurez clasele de a lungul a 3 pachete . Business continue logia de lucru si mecanismul problemei , Presentation este fromat din clase care construiesc Graphic user interfaceul si tot odata control butoanelor afalate pe aceste interfete. Chiar daca nu am folosit nimic din pachetul 3 , acesta continue Clasele FileWritrer si Serializare care permit lucru cu fisiere.

In interiorul lui Business am definit ca Interface IDeliveryService care continue functiile de baza ale utilizatorilor aplicatiei , si urmeaza a fi implementate in detaliu in DeliveryService .

Am folosit Composite Design Pattern pentru clasele MenuItem, BaseProduct si CompositeProduct, iar pentru a notifica employee am folosit Observer Design Pattern care ii permite angajatului sa pregatesca comanda si sa o trimita spre livrare

Putem totodata sa structuram proiectul in cazurile obligatorii si cazurile optionale.

Cele obligatorii prevad logarea unuia din fiecare categorie de user print o fereastara reprezentativa , imporatrea dintr un fisier excel in JTable, opeartiile pe

produse ale adminului , inclusive adaugarea de produse compuse , search-ul de la client si rapoartele adminului , acestea din urma prezentand conditia de utilizare a streamurilor.

In randul celor optionale se inscriu generare facturii , inregistrarea in aplicatie si serializare datelor

Orice analiza prezinta scenarii de utilizare , asadar pentru inceput ne legam de fereastra de log on, care continue 2 textfield-uri . Daca se introduce contul si parola corespunzatoare corecta adminului/angajatului se permite intrarea in aplicatie. In schimb pentru utilizarea aplicatiei din partea unui client este nevoie de inregistrare ca prim pas , apoi intrarea in aplicatie. In caz contrar la acestea se va afisa un pop-up care anunta ca user-ul si parola sunt gresite.

Odata intrat in aplicatie din postura de admin se poate apasa o singura data butonul de import , deoarece daca este de doua ori apasat se va face stackoverflow , tabelul sustinand deja maximul de elemente posibile.

Legat de adaugare se poate adauga orice element cu conditia ca toate fieldurile din pop up sa fie completate. La update este mai convenabil pentru ca permite obligatoriu doar completarea id-ului restul putand sa ramana nule (cu valoarea initiala din fereastra in cazul nostru aceasta fiind -10).

Aplicatia este construita in asa fel incat sa I fie usor de utilizat unui client. Butonul de view menu nu prezinta niciun caz nefericit , la fel ca si butonul de search care permite cautarea dupa nume sau pret , nume si pret sau niciuna ,primind in ultimul caz un mesaj informativ. La crearea comenzii se va scrie doar titlul produsului .

PROIECATRE SI IMPLEMENTARE

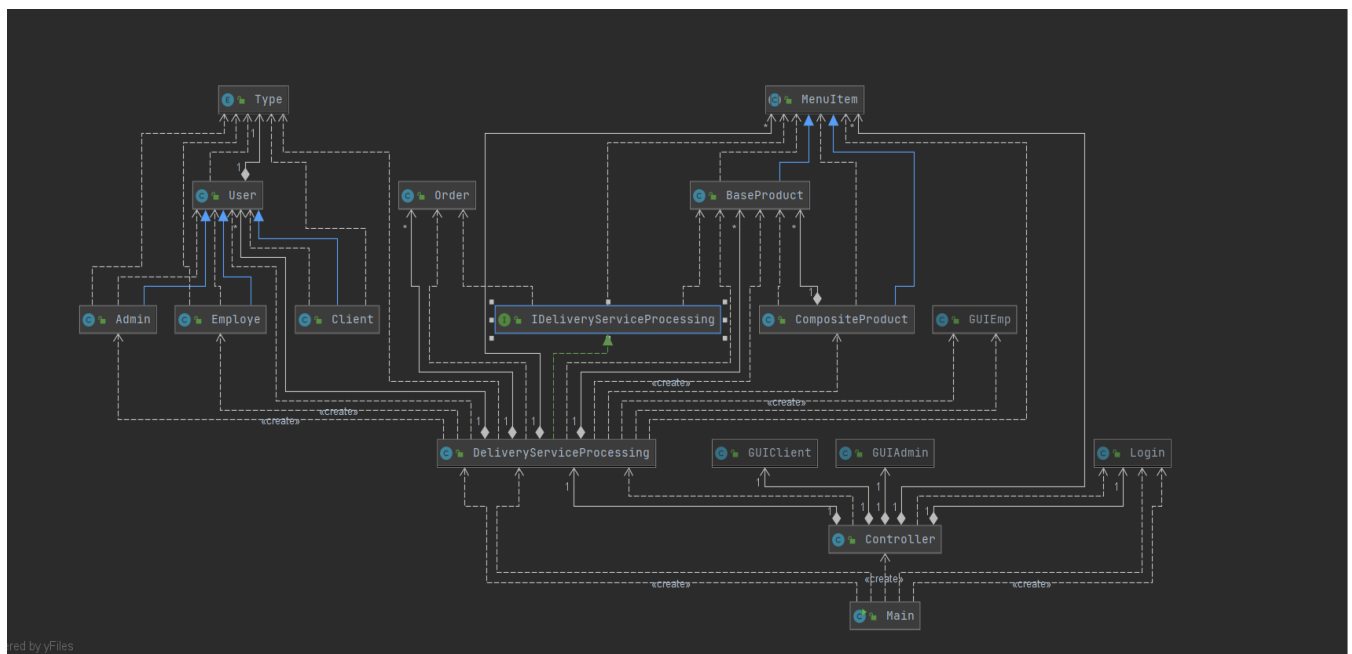
Divizarea proiectului in pachete si ce contina acestea a fost descrisa mai sus

Ca un prim pas la problema inainte de incepe implementareae propriu zisa a diagramei prezentate in enunt, am studiat demersul logic al problemei care contine logarea unui admin si importarea produselor din .csv. Prin importare se intelege si populare tabelului cu acestea nu doar retinerea lor intr un Set.

Urmeaza fie testarea unor aduagari/stergeri/modificari/creari a produselor din meniu , fie direct inregistrarea a catorva client care verifica elemntele din meniu si le filtreaza prin primsa conditiei de search.

Doar la final clientul face produsule , iar adminul vizulizeaza rapoartele acestea generandu-se in cate un fisier txt fiecare cu nu nume sugestiv

Diagrama UML arata in felul urmator:



Odata prezentata si implementarea grafica a aplicatiei putem trece la descrierea fiecărei clase în parte și algoritmii implementați la fiecare pas.

Clasa User

Este clasa care conține user, parola și tipul de utilizator : Client , Admin și Employee . Prezintă totodată și getter-le și setter-le necesare pe parcursul problemei . Aceasta clasă va fi moștenită de fiecare tip de utilizator în parte

Clasa Admin, Employee si Client

Toate din cele 3 prezinta un constructor obisnuit care apeleaza `super()` mostenind cum am spus clasa `User`. Vor fi identificati fiecare dintre acestia pe baza unui id care va fi unic.

In cadrul acestor clase vreau sa specific ca fiecare user are un client unic. Id-urile nu sunt pentru fiecare `Role` in parte sunt pentru fiecare user asdar niciodata un admin nu va avea acelasi id cu un client.

Clasa MenuItem

Am ales sa discut acum despre aceasta clasa deoarece este in tema cu mostenirea , aceasta fiind mostenita de `Base` si `Composite Product`.

Este o clasa care contine toate fieldurile necesare pentru scrierea in meniu , care de altfel se afla si in documentul atasat (.csv) . Este o clasa abstracta cu o metoda abstracta `CompositeProduct` care urmeaza sa fie implementata in detaliu de clasele care o mostenesc

Pe langa acestea prezinta `Getterele` si `Setterele` standard

Clasa BaseProduct si CompositeProduct

Cu referire la acestea 2 care mostenesc `MenuItem` apeleaza `super()` cu parametrii necesari. Desi `BaseProduct` este asemanatoare cu `MenuItem` , trebuie in plus sa avem grija la `computePrice` care trebuie obligatoriu implementata , desi este un singur produs si corpul funciei ei contine un simplu getter al pretului

In ceea ce priveste `CompositeProduct` , este putin mai special singurele fielduri sunt `final price` care ne ajuta sa calculam `ComputePrice` si un `Set` de `menuItems`. Setul de `menuItems` reprezinta o serie de mai multe produse normale , iar scopul acestei clase este de a obtine un singur produs pe baza lor.

Interfata IDeliveryService

Implementarea este fix cea din enunt. Aici nu se afla antetul tuturor operatiilor ce pot fi executate de un utilizator , doar cele specificate care reprezinta cele de baza de fapt.

Le-am impartit in doua : Cele de admin si cele de Client

Cele de Client reprezentau computePrice care practic reprezinta cerea notei de plate si poate fi executata de un client, newOrder adica plasarea unei comenzi si searchInMenu dupa cum spune si numele sugestiv al metodei poate cauta in intreg meniu dupa 2 criterii (pret si titlu).

Cele de Admin reprezentau toate rapoartele referitoare la produse, clienti , un interval orar sau chiar o zi a lunii, si importarea de produse si modificarea acestor prin adaugari, stergeri, creeri de meniu sau chiar edicare de pret la unele.

La toate aceste metode am adaugat javaDoc Documentation cu pre si post conditii si tot odata un invariant despre care vom discuta in cele ce urmeaza. Pre si post conditiile reprezinta niste asserturi care vor fi testate in functie de tipul lor la sfarsit sau la inceput. Am avut grila pentru o intelege cat mai usoara a utilizarii acestei platforme in cazul in care un assert nu este indeplinit mesajul sa fie un corespunzator si inteligibil , clientul sa-si dea seama de eroare si sa inerce o noua comanda/cautare

Clasa DeliveryService

Face Override la toate metodele din Interfata cu acelasi nume , iar in plus mai adauga cateva metode pentru o functionare fluanta a platformei

Comenzile sunt stocate in Map < Order, HashSet<MenuItem>>, menuItems care reprezinta meniul propriu zis este stocat in HashSet<MenuItem>, reprezentand valoarea comenzii numai ca mult mai filtrat, iar userii stocati in LinkedList<User>. Pentru fiecare dintre acestia am facut un getter care poate fi utilizat de Controllerul interfetei atunci cand este nevoie .

Constructorul este utilizat strict pentru hard-codarea unui admin si unui employee deoarece acestia trebuie deja sa fie inregistrati pe platforma nu ca si un client care trebuie sa foloseasca functia de register. Referitor la aceasta metoda pur si simplu dupa creare lui cu new User este introdus in lista initializata cu add.

Invariantul despre care va promiteam ca vorbesc este de fapt functia booleana `isWellFormed()` si cand o utilizam ea trebuie sa fie neaparat adevarata. Avand in vedere acest lucru m-am gandit ca invariantul si implicit aceasta functie sa determine tipul de user care acceseaza interfata . Asadar un Admin nu va putea comanda sau cauta in meniu. Cel care foloseste aplicatia si utilitatile de client va fi doar Client-ul, nu adminul sau angajatul

`ComputePrice` ca si cel din `CompositeProduct` calculeaza pretul total facand suma fiecarui pret dintre elementele aflate intr o lista trimisa ca parametru. Aceasta metoda este folosita in special la generare facturii la scrierea pretului final

Metoda pop up este observer-ul nostru care notifica angajatul cu un numar intreg reprezentand id-ul noii comenzi .Aceasta este apelata din `newOrder` care dupa ce verifica pre acontia introduce in lista de comenzi toate elementele comandate de acel client la key data de `Order order`. Hash-codeul acestuia este unic , fiind prezentat la inceputul documentatiei. Tot in cadrul acestei metode `newOrder` se genereaza si facture care contine numele itemelor selectate din meniu, data la care a fost facuta si pretul total.

`PrintOrders` a fost folosita in scopul testarii pe parcursul implementarii platofrmei daca comenzile sunt luate aferent si corespunzator.

La `SearchInMenu` intervine una din conditiile principale ale acestei teme. Folosirea stream-urilor si a lambdei expression. Logica este una simpla referindu-ne la cei 2 parametrii care pot fi nuli ambii sau doar unul sau niciunu. Metoda este astfel gandita inca daca nu este selectat niciunul va afisa intreg meniul asa cum este normal. Iar in functie de ce este selectat aceea se va intampla.

Mentionez ca price din `SearchInMenu` este pretul maxim, si lista care va fi returnata va contine toate elementele din meniu cu pretul mai mic decat cel introdus. Streamurile sunt folosite pe o lista auxiliara care ia elementele din meniu si le parcurge una cate una prin lambda expression si filter care impune conditiile necesare de filtrare iar apoi este introdus in colectia stabilita la inceput in functie de tipul ei (`toSet` la seturi si `toList` la liste in cazul utilizatorilor).

`ImportProducts` este o alta metoda care beneficiaza de lucrul cu streamuri fiind splituite toate elementele din acel excel in cate 7 bucati reprezentand cate un criteriu de introdus pe cate o coloana in functie de elemental la care ne aflam in `JTable`. Elementele sunt initial introduse in `theProducts` iar mai apoi dupa ace mai

sunt odata filtrate sa elimine toate duplicatele in meniu. Orice s-ar afla in meniu se sterge si se adauga produsele filtrate . Filtrarea a avut loc pe baza metodei statice `distinctByTitle` , implementata cu ajutorul unor documente citite de pe `StackOverflow`.

`WriteInTheTable` si `AddTheTable` sunt metode simple care se ocupa strict de interfata , de tabelul menu mai exact. Acestea ajuta la interactiunea clientilor cu meniul . Practic transpun mai exact ceea ce se intampla in `HashMap` si `HashSet`, dupa adaugarea/editarea/stergerea sau chiar importarea initiala de produse

Referitor la adaugare un produs se adauga cu toate fieldurile obligatorii. In momentul in care adminul introduce in textfields trebuie sa aiba grija sa fie introduse toate , sa nu ramana niciun field necompletat. In cele ce urmeaza acestea pot fi modificate dar nu pot ramane nule , cu exceptia unui composite product care contine doar titlul si pretul . `AddTheTable` spre deosebire de `writeInTheTable` se ocupa doar de ultimul element al tabelului , ultimul element cu care s-a lucrat

Cum am spus la editare id-ul este obligatoriu in rest daca nimic nu este selectat nimic nu este modificat. Daca doar un field este modificat acel se va modifica. Nedorirea modificarii unui field implica lasarea acelui textfield -10, programul detecteaza ca ramane neschimbat si se va ocupa de acest lucru. La delte ca si aici obligatoriu este ca id-ul sa nu fie null , sa se stie ce sa se stearga , iar mai apoi este updatat tabelul suprapunandu-se asupra lui alte elemente ce urmeaza dupa el. In caz ca este ultimul se va complete fiecare celula cu “”.

Rapoartele sunt cele mai complexe parti ale problemei din punctul meu de vedere. In cadrul raportului de timp pornim la lucru dupa ce este verificat faptul ca timpul de start este neaparat mai mic decat timpul de finish. Cream o lista de comenzi care va contine keyset-ul comenzilor care s-au comandat intre intervalul orar specificat de client prin parametrii metodei . Pentru scrierea in fisier se va folosi un stream care conine `forEach`.

Pentru al doilea raport referitor la produsele cele mai comandate cream un map nou de comenzi pe baza valorilor din hashmap initial de comenzi. Ideea de baza este ca punem valoarea 1 daca cheia nu a mai fost inalnita (daca produsul nu a mai fost intalnit) sau incrementam valoarea daca exista deja cheia (produsul a mai fost comandat de cineva in urma)

Acest Map mai apoi ii studiem valorile si afisam cheia, numele cheii adica numele produsului daca indeplineste conditia de filter ca valoarea de la key sa fie mai mare decat paramterul specificat in antetul functiei.

Al treilea raport se refera atat client , pentru ca ei vor fi monitorizati cat si la produsele pe care acestia le-au comandat. Ca sa selectam doar comenzile cu valoarea notei de plata mai mare decat una specificata vom folosi un Set auxiliar, iar pe baza aceluia avand odata separate toate comenzile de pret minim putem crea un nou hashmap de Integer , Integer. Primul integer se refera la codul clientului iar al doilea la cate comenzi a facut acel client din cele selectate.

Parcurgem cu stream Set-ul de comenzi cu pretul minim specificat si retinem in hashmap valoarea id-ului clientului pe baza key hashmapului initial.

Ultimul raport se refera strict la o zi din luna de aceea assert-ul initial prevede incadrarea acesteia intre 1 si 31. Se aseamana cu penultimul trebuind sa selectam din intreaga lista de comenzi , comenzile care au avut loc in ziua selectata. Mai apoi cream un HashMap insa de data asta cu tipurile MenuItem si Integer. Sa stim fiecare element de cate ori a fost comandat.

Nu avem nicio conditie de impus aici , pur si simplu pentru fiecare produs verificam daca a mai fost comandata sa stim daca l incrementam sau este primul produs si vom seta valoarea key doar 1.

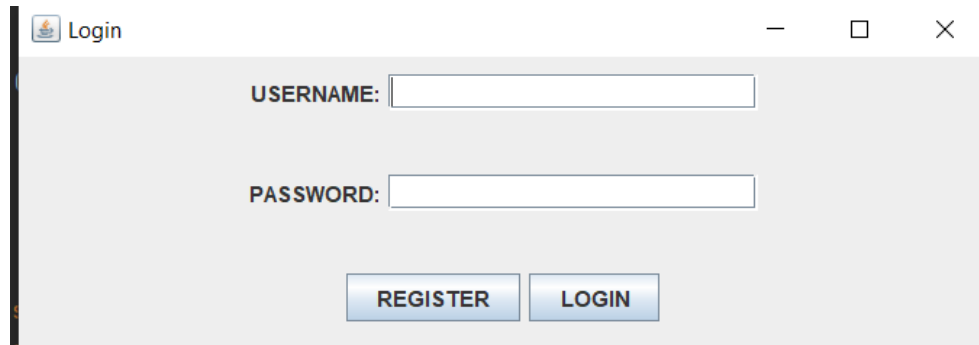
La sfarsit ca si la toate rapoartele printr-un foreach am scris in fisiere. Scriere in fisere s-a facut folosind FileWriter si PrintWriter. Initial trebuie golit fisierul sa nu se suprapuna unele peste altele , asadar vom scrie un “” inainte sa incepem. Selectam fisierul.txt si append-ul true pentru ca se va scrie pe parcurs in el , nu totul odata. Pentru a se salva trebuie sa inchidem neaprat atat printwriter-ul ca sti filewrite-ul.

La toate functiile suprscrise exista ca si in Interfata de DeliveryService @pre, @post @invariantul unde e cazul si paramaterii descrisi , pentru o coerenta generare a javadocurilor.

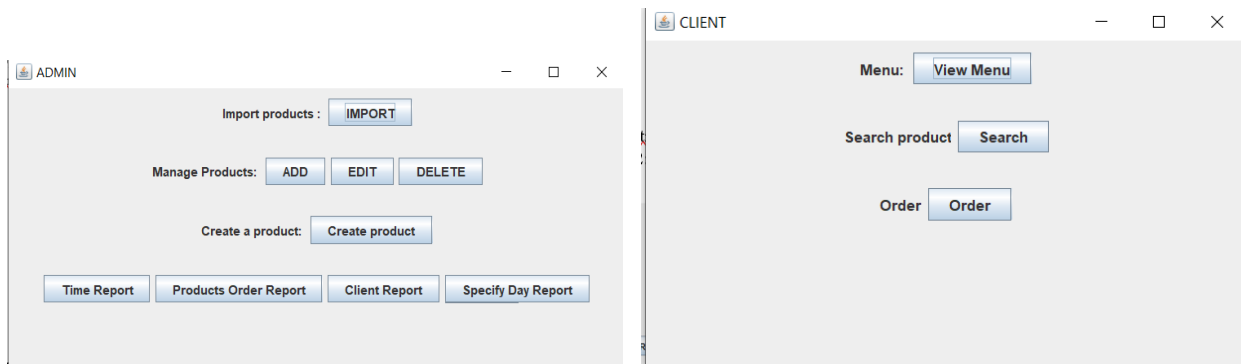
Pachetul PresentationLayer

Contine clasele pentru fiecare fereastră de lucru si Controllerul care gestioneaza butoanele si actiunile din spatele ferestrei , adica toate metodele implementate si discutate inainte.

Login permite registrarea pe baza unui username si parola si logarea cu acelasi date. Daca acestea nu sunt introduse corect se va afisa un pop care specifica ca una din ele este gresita. Parola este in planemode.

A screenshot of a 'Login' window. It has a title bar with a small icon and the text 'Login'. The window contains two text input fields: 'USERNAME:' and 'PASSWORD:'. Below these fields are two buttons: 'REGISTER' and 'LOGIN'.

Odata intrat in aplicatie depinde de tipul utilizatorului introdus, insa puteti intalni una din cazurile acestea 2:

Two side-by-side screenshots of application windows. The left window is titled 'ADMIN' and contains several buttons: 'Import products : IMPORT', 'Manage Products: ADD EDIT DELETE', 'Create a product: Create product', and a row of four buttons: 'Time Report', 'Products Order Report', 'Client Report', and 'Specify Day Report'. The right window is titled 'CLIENT' and contains three buttons: 'Menu: View Menu', 'Search product Search', and 'Order Order'.

CONCLUZII

Consider ca tema aceasta a fost cea mai complexa insa si una foarte utila , invand lamda expression si folosinta streamurilor, lucruri care sunt folosite si de companii , dupa cum am aflat in scurtul meu internship.

Legat de aceasta platforma problema pe care o gasesc eu la nivelul aplicatiei mele este notificare la fiecare comanda , adica daca am 5 comenzi voi primi 5 notificari pana cand o comanda nu este livrata sa poate sa fie stearsa.

Legat de imbunatatiri as putea spune ca la Securitate am ai fi de lucrat deoarece nu sunt foarte sigur retinute parolele mai ales ca sunt scrise in plane mode. Se poate face criptarea lor si stocarea lor intr o baza de date securizata si ea.

Este utila si poate fi abordata de orice companie de delivery mai ales prin prisma rapoartelor care ti permit studierea afacerii :

- Sa-ti dai seama care este ora de varf cu cele mai multe comenzi
- Sa-ti dai seama care produse sunt cele mai comandate
- Sa-ti dai seama care sunt clientii fideli , si care fac o consumatie ce ti aduce maim ult profit
- Sa-ti dai seama de zilele dintr o luna care a fost mai benefica sis a incepi sa ti pui intrebari de ce..

BIBLIOGRAFIE

<https://stackoverflow.com/questions/23699371/java-8-distinct-by-property>

<https://www.baeldung.com/java-observer-pattern>

https://www.w3schools.com/java/java_hashmap.asp

<https://stackoverflow.com/questions/13650142/hide-frame-instead-of-close-when-user-click-on-close-button-of-the-window>

<https://www.geeksforgeeks.org/stream-in-java/>