

FOOD DELVERY MANAGEMENT SYSTEM

Documentație

Pentru: Tehnici de Programare

Studenți: Luncian Lidia, grupa 30227

Cuprins:

1. Obiectivul temei
2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie
8. Obiectivul temei

Obiectivul acestei teme a fost proiectarea și implementarea unui sistem de livrări pentru o firmă de catering cu o interfață ”User-Friendly. Această aplicație permite efectuarea de operații de către administrator și client. În interfața angajatului se vor vedea doar numele produselor care au fost comandate, și care trebuie livrate. Există o interfață principală de LogIn. Utilizatorul trebuie să introducă username-ul și parola, iar apoi să apese butonul ce referă poziția sa în sistem. Dacă este client și nu are cont acesta își poate crea cont de client.

Aplicația a fost proiectată conform paradigmelor de programare orientată pe obiecte, folosind limbajul de programare Java. Am folosit modelul arhitectural Layers pentru modelarea pachetelor și a claselor.

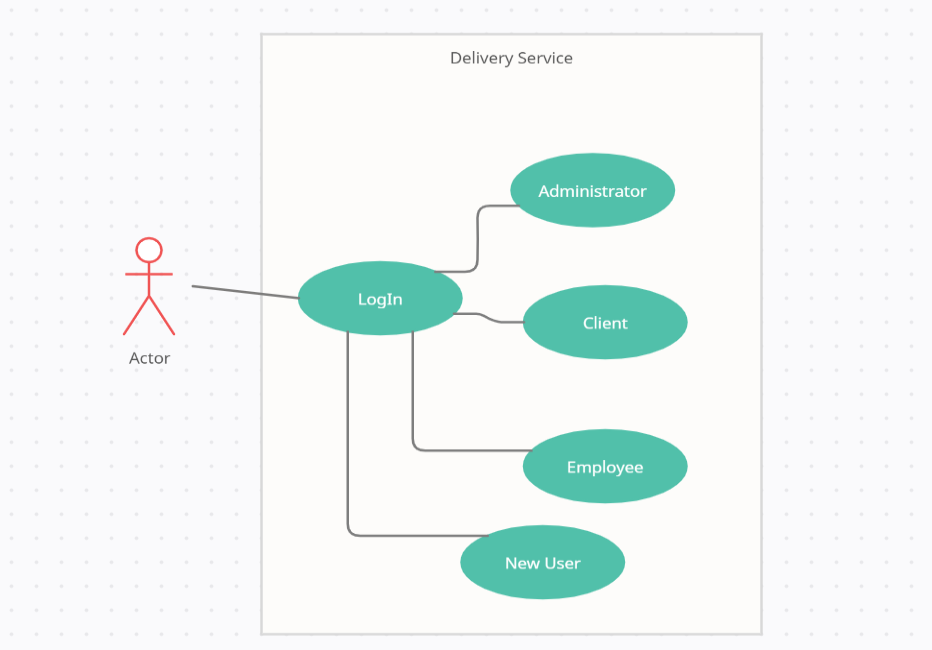
1. Analiza problemei

În această secțiune voi prezenta funcțiile fiecărui utilizator, raportat la interfața grafică. Această aplicație este utilizată de trei useri diferiți: Administratorul, Clientul și Angajatul.

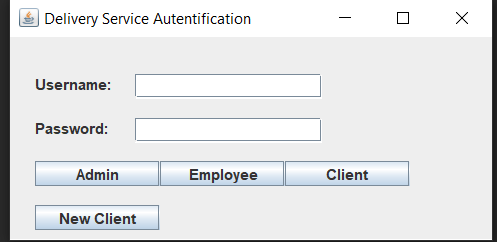
Mai jos este prezentat use-caseul temei.

Acesta prezintă relația utilizatorului cu aplicația prin interfața de LogIn din care poate ajunge în cea destinată lui, în funcție de ce fel de utilizator este.

În interfața LogIn utilizatorul este rugat să introducă username-ul și parola cu care și-au creat contul. Dacă acesta este un client nou el poate apăsa butonul New Client. Prin apăsarea butonului apare o noua fereastră unde utilizatorul își poate crea contul, sau să renunțe și sa apese butonul Back sau X. Odată ce și-a introdus datele, acesta este redirecționat la fereastra principală unde este rugat să își introducă din nou datele și apoi poate apăsa butonul Client.



Interfața LogIn și NewCient:



Graphical user interface, application

Description automatically generated

Dacă user-ul este Administrator acesta trebuie neapărat să își cunoască username-ul și parola , deoarece nu poate fi creat cont de Administrator, ci doar de client. Odată ce și-a completat datele și a apăsat butonul Administrator, va apărea fereastra asignată acestuia.

Graphical user interface, application

Description automatically generated

Administratorul poate face 7 lucruri în această interfață. El poate crea un produs de bază sau un produs compus. Pentru crearea produsului de bază, toate câmpurile Title, Rating, Calories,Proteins, Fats, Sodium și Price trebuie să fie completate, apoi se apasă butonul BaseProduct. Pentru crearea unui produs compus trebuie completat doar câmpul Title. Acesta va crea doar numele produsului compus. Pentru a adăuga componente unui produs compus, administratorul merge în secțiunea Add Base Product to Composite. Folosește primul ComboBox pentru a alege produsul compus pentru care vrea să creeze lista de produse de bază. În cel de-al doilea ComboBox se alege produsul pe care vrea să îl adauge în produsul compus și apasă Add. Poate adăuga oricâte produse dorește.

În următoarea secțiune Edit Base Product administratorul poate schimba datele unui produs de bază. In comboBox se alege produsul care se dorește a fi schimbat, iar în căsuțe se completează noile specificații. Acestea nu pot rămane goale sau se va adăuga zero unde nu a fost completat. Dacă apasă butonul Edit, datele vor fi actualizate, iar dacă seapasă butonul Delete produsul este șters.

Următoarea secțiune se referă la produsele compuse. Administratorul poate șterge componente ale produselor compuse .

Ultima sectiune se refera la generarea rapoartelor. Administratorul trebuie sa introduca datele dorite pentru fiecare raport si apoi sa apese butonul Generate Reports care generează toate cele 4 rapoarte cerute de proiect: primul generează un raport ce conține datele despre comenzile care au fost create într-un anumit interval orar, al doilea generează un raport cu produsele care au fost comandate de mai multe ori decât un număr dat (în acest proiect numărul este deja implementat în cod, nu este date explicit de administrator). Al treilea raport conține clienții care au creat mai multe comenzi decât un număr dat, și care au prețul mai mare decât un număr dat (din nou aceste date, numărul de comenzi și prețul sunt scrise implicit în funcția de apelare, și nu explicit de administrator). Ultimul raport conține numele produselor care au fost comandate într-o anumită zi, precum și de câte ori a fost comandat.

Dacă utilizatorul este un client, după ce a introdus datele corecte și a apăsat butonul Client al intefeței LogIn, apare următoarea fereastră:

Table

Description automatically generated

În primul rând, clientul poate vedea toate produsele disponibile în meniu . El poate căuta un anumit produs în funcție de un cuvânt cheie, preț,calorii... folosind opțiunea search. El va observa rezultatele tot în tabel.

Clientul mai poate crea și o comandă. Prima dată el trebuie să introducă id-ul de client (nu este foarte relevant, dar așa a fost proiectat). Apoi trebuie să apese butonul New Order. Astfel se crează o nouă comandă pentru clientul cu id-ul dat. După aceea clientul poate adăuga ce produse dorește din tabel apăsând butonul Add. Pentru a vedea la ce preț a ajuns cu comanda el poate apăsa butonul Price. La final el poate cere chitanța apăsând butonul Bill.

Odată ce clientul a apăsat butonul New Order, intefața angajatului este atenționată că a fost creată o nouă comandă. În timp ce clientul adaugă în comanda sa, numele produselor apar în fereastra angajatului.

Graphical user interface, application

Description automatically generated

1. Proiectare

Am folosit modelul arhitectural Layers pentru modelarea pachetelor și a claselor. Pachetul Business cuprinde clasele: BaseProduct, CompositeProduct, DeliveryService, Order și interfețele MenuItem și IDeliveryProcessing.

Clasa BaseProduct este clasa care modelează produsele de bază. Un produs de bază este definit de titlu (nume), rating, calories, proteins, fats, sodium, price.

Clasa CompositeProduct este clasa care modelează produsele compuse. Acestea sunt definite de un titlu (nume), și de o listă de produse de bază.

Interfața MenuItem este implementată de clasele BaseProduct și CompositeProduct. Cu aceasta interacționează clasa DeliveryService și este folosită pentru implementarea modelului arhitectural Composite.

Clasa Order este clasa care modelează comenzile create de client. Aceasta este caracterizată de orderId, clientId și orderDate.

Clasa DeliveryService este clasa principală în care sunt implementate toate funcțiile definite în IdeliveryProcessing. Această clasă conține lista comenzilor (orders), orderList – colecția care mapează fiecare componentă a comenzii unui client la comanda sa, am folosit un HashMap deoarece aceasă colecție este foarte eficientă pentru operații de căutare și stocare. Clasa mai conține și o listă cu produse de bază unde sunt importate produsele din fișierul .csv, iar mai apoi acestea sunt adăugate și în colecția meniu, care mapează fiecare produs din meniu, de tip MenuItem(ori BaseProduct, ori CompositeProduct) cu cheia numele produsului și valoarea o listă de MenuItems. Din nou am ales HashMap pentru căutare și adăugare facilă în meniu.

Clasa mai conține și un element de tip Observer, prin care clientul transmite notificări angajatului.

Interfața IDeliveryProcessing definește toate operațiile importante pe care un administrator sau client le poate face.

Pachetul Data conține clasele: FileWriterTxt și Serializator. Clasa FileWriterTxt conține o metodă statică ce generează o chitanță. Clasa Serializator conține 2 metode statice care interpretează fișierul delivery.txt.

Pachetul Presentation conține clasele: Administrator, Client, Employee, LogIn, NewUser și User. În clasa User este modelat utilizatorul, acesta conține un username și o parolă. Clasa Administrator implementează Jframe și creează interfața pentru administrator. Clasa Client implementează interfața pentru client. Clasa Employee implemetează interfața pentru angajat. Clasa LogIn implementează interfața de logare a utilizatorului, iar clasa NewUser implementează interfața pentru un nou utilizator.

Ultimul pachet, default, conține clasa MainClass. În această clas, teoretic se alege dacă se vor importa datele din csv, sau se va deserializa fișierul delivery.txt. Daca se importă atunci se creează și un produs compus pentru a evita ceva erori în intrefața Administrator, la editarea produselor compuse.

Diagrama UML:

Diagram

Description automatically generated

1. Implementare

În clasa BaseProduct pe lângă gettere și settere este implementată metoda toString ce afișează un produs de bază, metoda equals în funcție de titlu, precum și metoda hashCode. Acestea 2 din urmă ajută la importarea elementelor distincte din fișierul products.csv.

Clasa CompositeProduct implementează metode de adăugare și de ștergere în lista cu componente, precum și metoda containsProduct, care verifică dacă un produs este deja în lista produselor sau nu.

În clasa DeliveryService sunt implementate metodele definite în interfața IDeliveryProcessing. Aceste metode sunt folosite mai apoi în interfețele pentru administrator sau pentru client. Metoda importProducts creează o listă de produse de bază folosind stream-uri și expresii lambda. Aceste produse sunt mai apoi adăugate și în lista meniu. Metoda createBaseMenuItem creează un nou produs de bază. Metoda createCompositeMenuItem creează un nou produs compus cu lista de produse nulă. Metodele addComponent și removeComponent adaugă un produs de bază la un produs compus, sau șterge. Metodele editBaseProduct și deleteBaseProduct modifică sau șterg complet un produs de bază din meniu. Apoi urmează metodele legate de comenzile clienților, de creare, createNewOrder, de adăugare de produse, addOrderMenuItem și de calculare a prețului total al comenzii. Urmează metoda generateBill ce apelează metoda statică din clasa FileWriterTxt și generează o chitanță clientului. Metoda isWellFormed, verifică dacă după ce se acționează asupra meniului acesta este în continuare bine format. Metoda notifyEmployee transmite angajatului mesajul că un client a generat o comandă.

În cele din urmă tot aici sunt implemetate metodele pentru cele 4 rapoarte folosind stream-uri și expresii lambda.

Acestă clasă mai conține și o variabilă ce se incrmentează de fiecare dată când un utilizator creează o nouă comandă, astfel ține evidența numărului de comenzi în mod automat. Acesta se incrementează în metoda createNewOrder.

Metoda import:



Raport1:

Text

Description automatically generated

Raport2:

Text

Description automatically generated

Raport3:

Text

Description automatically generated

Raport4:

Text

Description automatically generated

Clasa Order pe lângă gettere și settere mai implemetează și metoda de getHours, aceasta extrage ora la care s-a făcut comanda din variabila orderDate. Este implementată și metoda getDay ce extrage ziua în care a fost făcută comanda, tot din orderDate. Aceste funcții sunt folosite în generarea rapoartelor 1 și 4.

În clasele ce implementează interfețele sunt apelate metodele din clasa DeliveryServie, fiecare corespunzătoare butonului său.

În interfața LogIn am creat o metodă defineWindowListener care, atunci când fereastra se va închide, datele vor fi serializate și astfel salvate în fișierul delivery.txt. Această metodă este apelată în corpul constructorului interfeței.

1. Rezultate

Am introdus date pentru utilizator, acesta poate să creeze produse de bază și compuse, poate să le editeze și să șteargă. Pe scurt merg toate butoanele administratorului. Se generează cu succes rapoartele, desi probabil ar fi fost frumos sa fie mai amanuntite. Clientul poate să creeze comenzi, iar angajatul primește notificarea clientului și începe să proceseze datele date de acesta.

Am folosit stream-uri și expresii lambda pentru crearea rapoartelor și a funcției de importare a produselor din fișierulproducts.csv.

Am testat generarea chitanțelor. Acestea se creează cu succes.

În documentație nu am introdus decât imagini cu funcțiile pe care le-am considerat cele mai importante.

Serializarea se face odata ce se inchide aplicatia, iar deserializarea este apelata in main si se face atunci cand se porneste aplicatia. Datele sunt salvate in fisierul delivery.txt odata ce aplicatia a fost inchisa.

1. Concluzii

În urma acestui proiect am învățat cum să creez o interfață de LogIn, să folosesc stream-uri și expresii lambda pentru a extrage datele dintr-un fișier .csv, pentru a crea rapoarte. Am înțeles conceptul de serializare și deserializare, si am reusit sa il fac sa mearga. Am înțeles nevoia utilizării assert-urilor în implementarea funcțiilor. Acest proiect poate fi bineînțeles îmbunătățit, atât ca interfață cât și ca și conținut. Interfața ar putea fi rafinată. Cand vrea administratorul să editeze un pr odus, în căsuțe să apară datele curente ale produsului, și apoi să ștearga și să scrie ce dorește el, pentru a nu fi nevoit să țină minte toate datele pentru fiecare produs. Pentru client, când acesta dorește să creeze o comandă, să nu introducă id-ul, ci eventual username-ul sau un nume. Idclient nu prea are sens, dar din motive de timp, nu am mai modificat. De asemenea clientul nu poate renunța la produsele deja comandate.

1. Bibliografie
2. <https://www.geeksforgeeks.org/composite-design-pattern/>
3. [The Observer Pattern in Java | Baeldung](https://www.baeldung.com/java-observer-pattern)
4. [swing - Java GUI login screen - Stack Overflow](https://stackoverflow.com/questions/26665538/java-gui-login-screen)
5. [Processing Files With Java 8 Streams](https://reflectoring.io/processing-files-using-java-8-streams/)
6. [Serialization and Deserialization in Java with Example - GeeksforGeeks](https://www.geeksforgeeks.org/serialization-in-java/)
7. [How to serialize and deserialize ArrayList in Java - HowToDoInJava](https://howtodoinjava.com/java/collections/arraylist/serialize-deserialize-arraylist/)
8. [DistinctBy in Java Stream API | Baeldung](https://www.baeldung.com/java-streams-distinct-by)
9. [How to Use Scroll Panes (The Java™ Tutorials > Creating a GUI With JFC/Swing > Using Swing Components)](https://docs.oracle.com/javase/tutorial/uiswing/components/scrollpane.html)
10. [Java Lambda Expressions](https://www.w3schools.com/java/java_lambda.asp)
11. [swing - how to search an element in a JTable java? - Stack Overflow](https://stackoverflow.com/questions/22066387/how-to-search-an-element-in-a-jtable-java)
12. [Filter table by the text in a TextField : JTable Filter « Swing « Java Tutorial](http://www.java2s.com/Tutorial/Java/0240__Swing/FiltertablebythetextinaTextField.htm)
13. [Using Java Assertions | Baeldung](https://www.baeldung.com/java-assert)
14. [Java 8 Stream Tutorial](https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/)
15. [Java HashSet](https://www.w3schools.com/java/java_hashset.asp)