MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FOOD DELIVERY MANAGEMENT SYSTEM

Alin Matean

Facultatea de Automatica si Calculatoare

Grupa 30228

Cuprins

Obiectivul temei

Obiectivul acestei teme este proiectarea și implementarea unei aplicații care să simuleze un sistem de management al livrării unor produse / meniuri. Această aplicație ar urma să aibă 3 tipuri de utilizatori și un meniu complet administrat de admin-ul acestui sistem.

Analiza problemei, modelare, scenarii, cazuri de utilizare

Analiza problemei

Problema care se impune este cea de a gestiona nevoile fiecărui tip de utilizator: administrator, client și angajat. Starea aplicației trebuie să fie salvată prin serializare. Adminstratorul ar trebui să poată să importe toate produsele, să poată să adauge un nou produs, să poată să editeze un produs deja existent sau chiar să poată să șteargă un produs. De asemenea, admin – ul ar trebuie să poată să genereze anumite rapoarte bazate pe criterii precum o anumită dată, un anumit preț al comenzii, un anumit număr minim de comenzi sau comenzi efectuate într-un anumit interval orar.

Scenarii

Scenariul de utilizare a aplicației presupune interacțiunea cu interfața grafică creată. Astfel fiecare tip de utilizator are o fereastră specială în funcție de statutul său. Fiecare utilizator trebuie mai întâi să se logheze, astfel se va deschide o pagină corespunzătoare tipului său de utilizator. Dacă este client și nu are un cont încă, se poate înregistra fără nicio problemă.

Scenariu de utilizare:

- se pornește aplicația
- se înregistrează utilizatorul
- dacă este admin aceasta ar trebui să introducă username-ul și parola admin – ului după care aceasta va putea să adauge un produs nou, să editeze un produs deja existent, să șteargă un produs. În tot acest timp el are acces la tabelul care conține toate aceste produse.Pe langă operațiile pe acest tabel cu produse, admin – ul mai poate să genereze rapoarte, mai exact 4 rapoarte. Primul raport va genera într-un fișier text toate comenzile efectuate într-un anumit interval orar, ora de start și de final fiind date de admin. Al doilea raport va genera într-un fișier text produsele comandate mai mult de un anumit număr

de ori, ales de admin. Al treilea raport va genera într-un fișier text clienții care au comandat mai mult de un anumit număr de ori și valoarea comenzii a fost mai mare de un anumit preț, aceste date fiind alese tot de admin, iar ultimul raport va genera într-un fișier text produsele comandate într-o anumită dată aleasă de admin prin lună și an.

- Dacă utilizatorul este client, atunci în caz că nu are deja un cont își va crea unul, altfel se loghează cu contul deja exitent. Fereastra care se va deschide pentru client conține tabelul cu toate produsele. În plus, conține un text Box și un combo Box prin care clientul își poate filtra produsele în funcție de un anumit criteriu specific.Acesta poate adăuga produse la o comandă curentă și poate plasa comanda după ce a ales toate produsele pe care le dorește. Pentru fiecare comandă se va genera un fișier bill.txt (care va fi suprascris de fiecare dată) ce va conține toate produsele din comanda curentă și prețul total al comenzii.
- În același timp în care se loghează un utilizator și i se deschide fereastra, se va deschide și fereastra angajatului, care va fi notificat la fiecare comandă nou apărută. Acesta dispune de un tabel în care poate vedea fiecare comandă efectuată.

Un exemplu valid de utilizare a aplicației ar fi următorul:

- Se loghează administratorul, introducând datele corespunzătoare sau
- Se loghează un client care are deja cont introducând numele de utilizator și parola corecte sau
- Un client își creează un cont și mai apoi se loghează

Cazuri de utilizare – diagrama Use-case

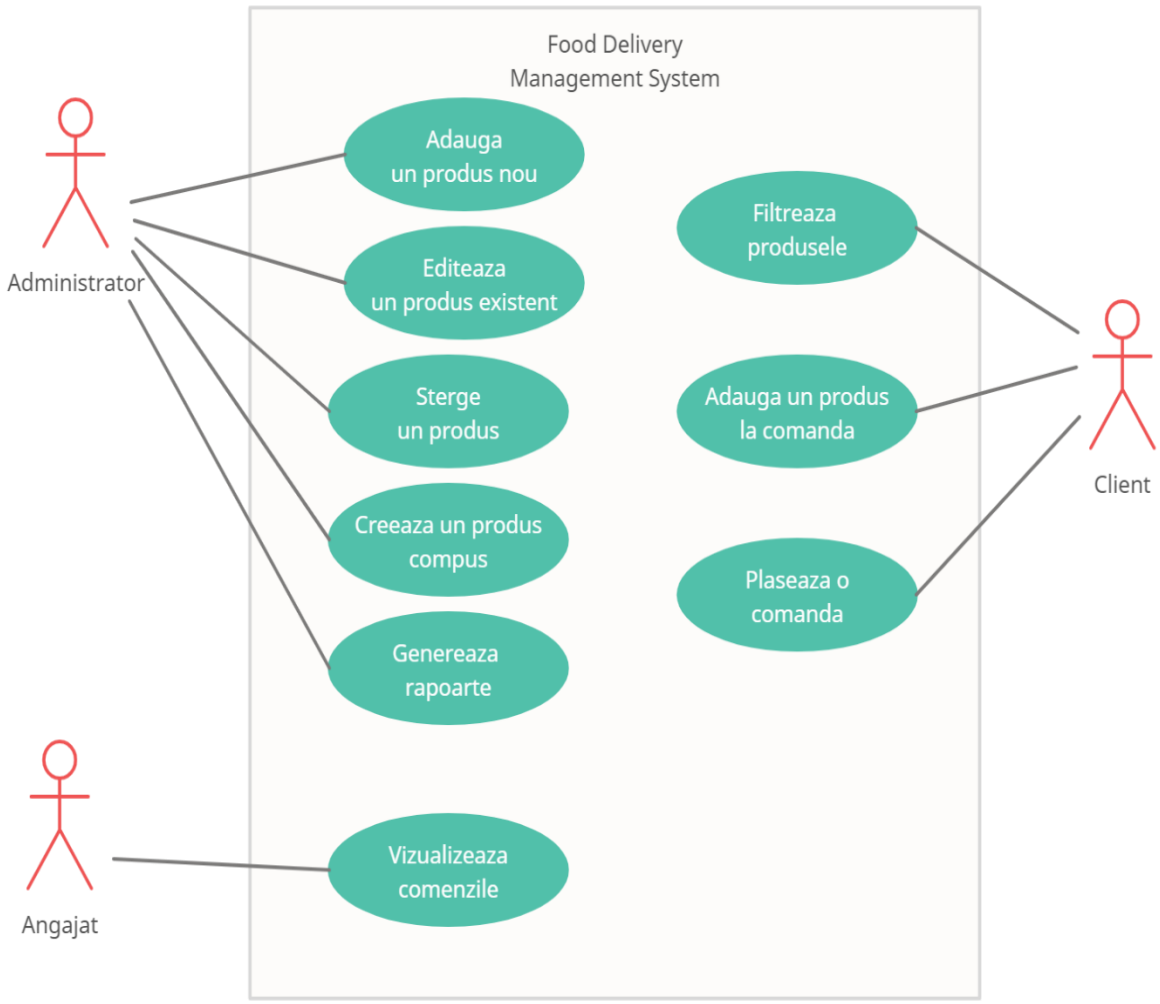


Figura 1. Diagrama use-case

Proiectare

Decizii de proiectare

Aplicația a fost proiectată folosind limbajul de programare Java. Deoarece nu am avut o bază de date, starea curentă a aplicației a fost salvată folosind Serializare si deserializare. Interfața grafică a fost realizată folosind Swing. Mai mult, aplicația este realizată prin layered architercture.

Astfel, am organizat aplicația in 3 pachete mari, , am organizat aplicația in 3 pachete mari. În cadrul pachetului "src" unde se află tot codul necesar funcționării și testării aplcației se găsește directorul "main". În directorul main găsim folder-ul "java" care conține pachetul prinipal și anume "tuc.tp.tema4", în care se află de fapt tot conținutul pentru funcționarea corectă a aplicației. Celelalte pachete sunt:

- Business- aici se încapsulează logica aplicației, sunt implementate clasele BaseProduct, Client, CompositeProduct, IDeliveryServiceProcessing, DeliveryService, MenuItem, Order
- Data – conține clasa necesară scrierii în fișiere FileWriter și cea pentru salvare Serializator
- Presentation – pachetul care conține clasele pentru interfețele grafice astfel avem: Login, EmployeeGUI, ClientGUI, AdministratorGUI
- Clasa App este cea care pornește aplicația și nu se află într-un pachet separat

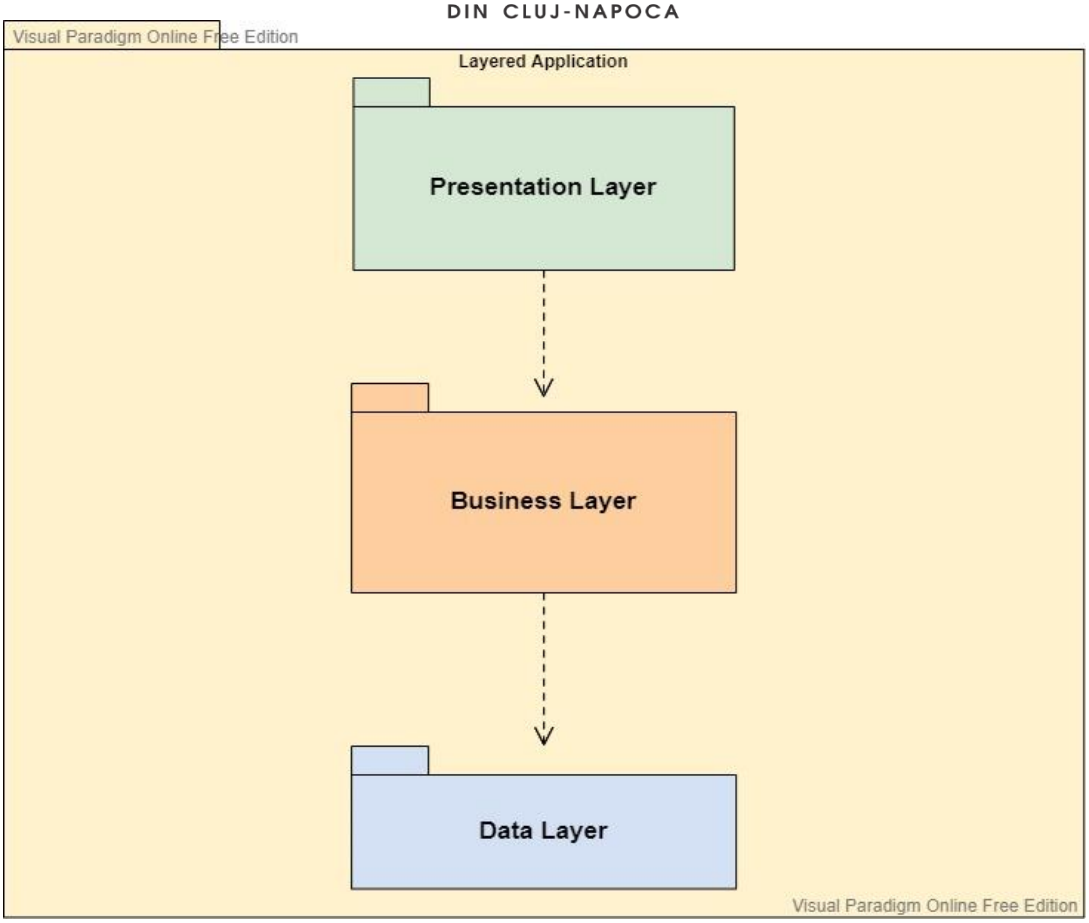


Figura 2. Diagrama de pachete

Diagrama UML de clase

Clasele proiectate:

În pachetul tuc.tp.tema4 se găsesc:

- Clasa App – cea care pornește aplicația
- BaseProduct – pachetul business
- Client – pachetul business
- CompositeProduct – pachetul business
- DeliveryService - pachetul business
- IDeliveryServiceProcessing - pachetul business
- MenuItem - pachetul business
- Order - pachetul business
- FileWriter – pachetul data

- Serializator - pachetul data
- AdministratorGUI - pachetul presentation
- ClientGUI - pachetul business
- EmployeeGUI - pachetul business
- Login - pachetul business

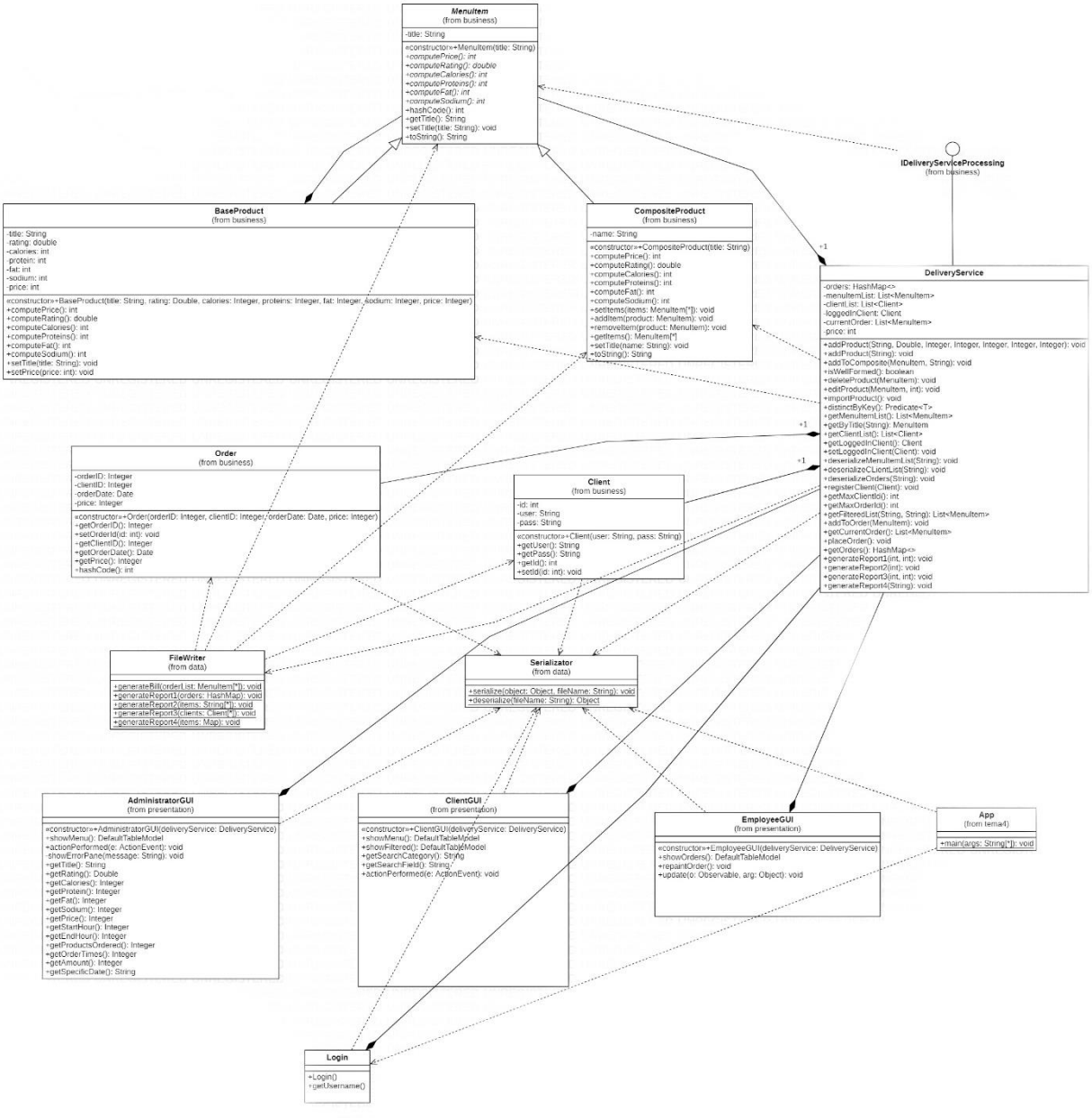
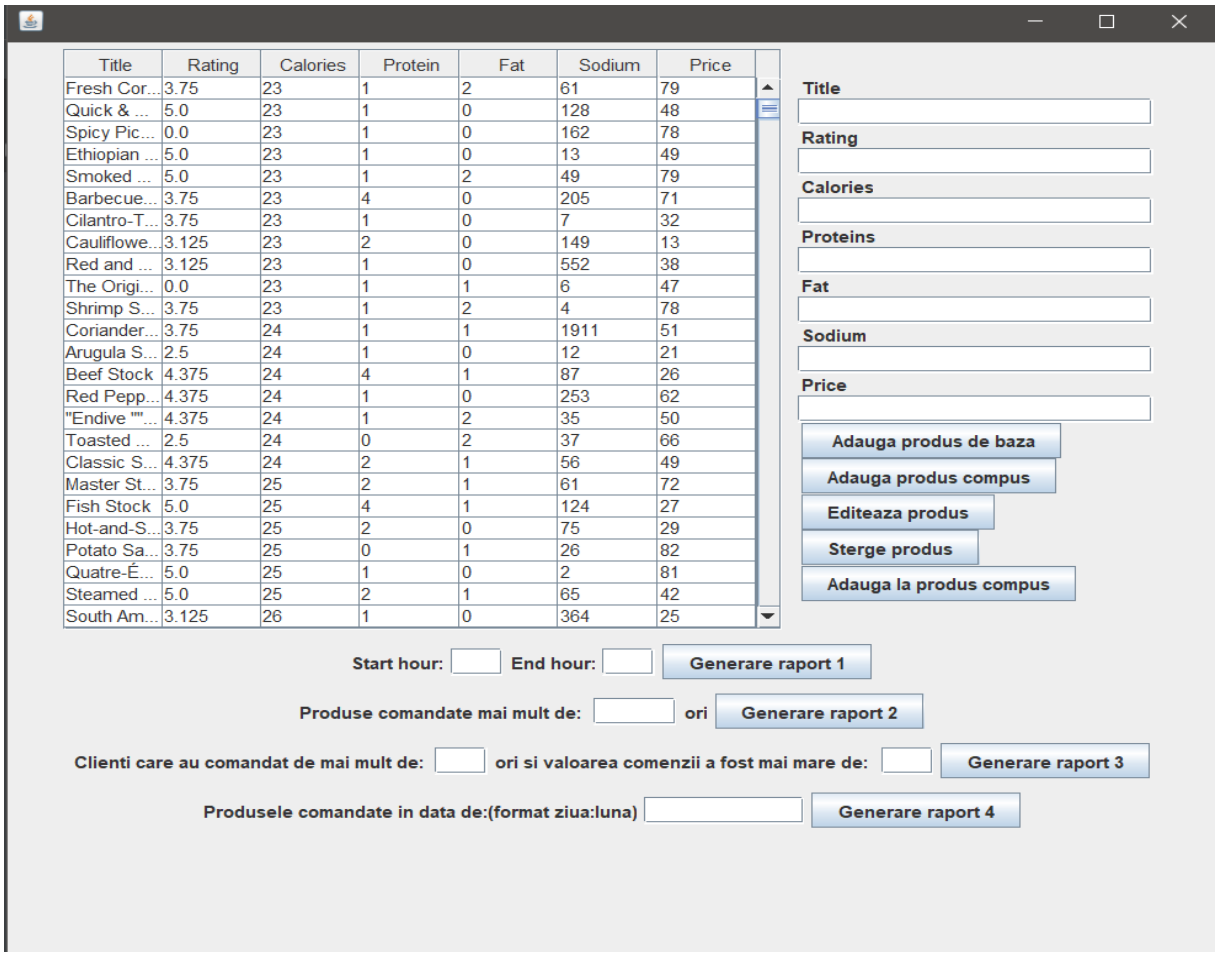
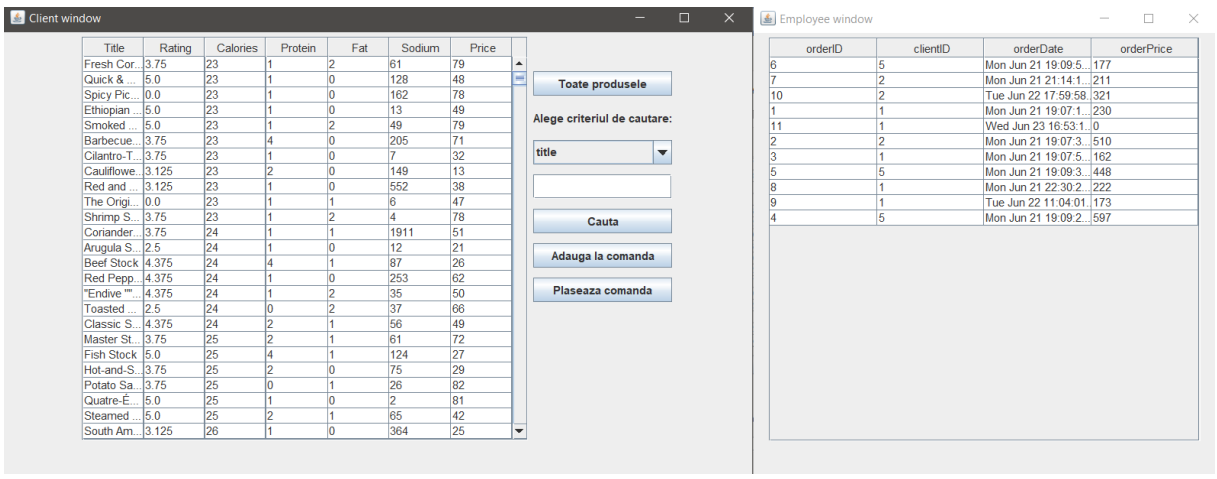


Figura 3. Diagrama UML de clase

Interfața grafică cu utilizatorul:

În funcție de tipul utilizatprului, fiecare are o anumită interfață corespunzătoare. Astfel, avem panoul comun Login unde trebuie sa fie introdus username-ul si parola sau se poate crea un cont de Client. Administratorul are panoul special în care poate să facă modificări asupra meniului, iar clientul poate să-și facă o comandă. Angajatul nu trebuie să se logheze în mod special, panoul său se deschide când se loghează un client, iar aceasta poate să vizualizeze comenzile.





Implementare

Clasele

Pachetul business:

- BaseProduct – modelează un produs de bază. Acesta conțin un titlu, un rating, numărul de calorii, numărul de proteine, numărul de grăsimi, conținutul de sodiu și prețul. În această clasă mai sunt metode de get și set pentru fiecare proprietate.
- Client – modelează un client. Acesta conține un nume de utilizator și o parole, pe langă un id unic. Avem metode de get și set în plus, pentru id, parola, username. În plus, această clasă implementează Serializable pentru a putea fi serializată.
- CompositeProduct – modelează un produs compus . Acesta conține o listă de MenuItem. Produsul compus are doar un titlu pentru identificare, iar restul caracteristicilor precum: rating, numărul de calorii, numărul de proteine, numărul de grăsimi, conținutul de sodiu și prețul se calculează ca suma tuturor elementelor din care este compus.
- DeliveryService – aceasta este cea mai importantă clasă, aici sunt implementate principalele funcționalități. În plus, această clasă implementează Serializable pentru a putea fi serializată și interfața IDeliveryServiceProcessing, pe lângă asta extinde și Observable. În clasa DeliveryService ținem minte: comenzile ca un HashMap<Orders, List<MenuItem>>, o listă de MenuItem cu toate elementele din meniu, și cele din products.csv și cele adăugate de administrator, o listă de clienți, un client loggedInClient, o listă pentru comanda curentă și prețul comenzii inițial fiiind 0. Metodele pe care le suprascrie din inerfața IDeliveryServiceProcessing sunt:
  - Metoda addProduct – care adaugă un produs de bază in meniu
  - Metoda addProduct – care adaugă un produs compus în meniu, inițial doar cu titlu și cu toate celelalte valori 0
  - Metoda addToComposite – care adaugă un produs în cadrul unui produs compus, în lista acestuia, produs care poate să fie ori de bază ori tot compus
  - isWellFormed – invariantul acestei clase care stabilește dacă e bine formată clasa
  - deleteProduct – care șterge un anumit produs din meniu
  - editProduct – care editează prețul unui produs din meniu

  Toate aceste metode au legătură cu interfața admin – ului, deoarece sunt acțiunile specifice lui.

  Celelalte metode care sunt definite în această clasă sunt:

  - importProducts – care realizează importarea produselor din fișierul csv
  - distinctByKey – care are grijă să nu existe duplicate în meniu
  - metodele care apelează deserialize din clasa Serializator și anume: deserializeMenuItemList, deserializeClientList, deserializeOrders
  - registerClient – adaugă un nou client în lista de clienți
  - getMaxClientId și getMaxOrderId – sunt necesare pentru a adăuga un nou client și o nouă comandă fiecare cu un nou id, astfel obținem maximul și adăugam + 1
  - getFilteredList – metoda care se ocupă de filtrarea produselor din meniu în momentul în care un client adaugă un criteriu de filtrare
  - addToOrder – adaugă un produs la comanda curentă și ii adaugă și prețul pentru a se calcula prețul total

- placeOrder – realizează plasarea unei comenzi pe baza datei curente și a comenzii curente, cu toate produsele din ea și cu prețul final calculat. Tot aici se notifică observatorii și se face 0 prețul pentru următoarea comandă ce va urma
- generateReport1 – genereaza raportul de tip1 care afișează într-un fișier text comenzile realizate într-un anumit interval orar
- generateReport2 – genereaza raportul de tip2 care afișează într-un fișier text produsele care au fost comandate de mai mult de un anumit număr de ori
- generateReport3 – genereaza raportul de tip2 care afișează într-un fișier text clienții care au comandat de mai mult de un anumit număr de ori și valoarea comenzii a fost mai mare de un anumit preț
- generateReport4 – genereaza raportul de tip2 care afișează într-un fișier text comenzile dintr - o anumită dată, specificată prin zi și lună.

- Interfața IDeliveryServiceProcessing – conține metode care sunt implementate în clasa DeliveryService
- MenuItem – este clasa care modelează un produs (de bază sau compus) din meniu. Acesta are inițial doar un titlu. În plus, implementează Serializable pentru a putea fi serializată.
- Order – este clasa care modelează o comandă. Aceasta conține un id, un client, o data și un preț. În plus, această clasă implementează Serializable pentru a putea fi serializată.

Pachetul data:

- FileWriter - este clasa în care se realizează scriere în fișiere, Conține 5 metode: una pentru generarea facturii în format text, una pentru generarea raportului de tip 1 în format text, una pentru generarea raportului de tip 2 în format text, una pentru generarea raportului de tip 3 în format text și una pentru generarea raportului de tip 4 în format text
- Serializator – este clasa care se ocupă de serializarea aplicației, mai exact a anumitor obiecte. Această clasă conține două metode, una pentru a serializa și una pentru a deserializa

Pachetul presentation:

- AdministratorGUI - este clasa în care se realizează interfața grafică pentru administrator. Aceast conține un tabel cu toate produsele și câmuri pentru a adăuga un nou produs, sau a edita. Mai mult, admin-ul poate să și șteargă un produs din meniu. Mai sunt câmpurile pentru generarea celor 4 tipuri de rapoarte.
- ClientGUI – este clasa în care se realizează interfața grafică pentru client. Aici un client poate vizualiza meniul, cu toate produsele, poate să filtreze meniul după un anumit criteriu ales și mai mult poate să realizeze o comandă prin adăugarea unui produs și mai apoi plasarea acesteia
- EmployeeGUI – aici exită doar un tabel cu toate comenzile plasate, iar când o nouă comandă e plasată apare o notificare
- Login – este clasa prin care e realizată prima fereastră în momentul în care e pornită aplcația. Aici se poate crea un cont de client sau se poate loga un client deja existent sau administratorul. În funcție de tipul acestui utilizator se va deschide mai apoi fereastra corespunzătoare. Tot aici, se adaugă și observatorul în cazul în care se loghează un client, pentru a putea fi mai apoi notificat angajatul.

Clasa App – este clasa care pornește aplicația, cea care conține metoda main.

Rezultate

Pentru testare am realizat operțiile corespunzătoare ale administratorului: am adăugat unprodus, am editat un produs și am șters un produs. Mai apoi, m-am logat ca și client pentru a putea vedea modificările făcute, care funcționează corespunzător. Ca și client am adăugat o comandă și am verificat fișiserul bill.txt care conține produsele. Mai mult, am verificat ca angajatul să primească o notificare la fiecare comandă nou plasată. Ca și administrator am generat și cele 4 rapoarte specifice și am verificat fișierele text să fie corecte.

Concluzii

Dezvoltări ulterioare:

- S-ar putea dezvolta interfața grafică cu utilizatorul
- Pentru crearea unui cont s-ar putea reține si solicita mai multe informații despre client
- Posibilitatea de a putea vedea din interfață ce conține fiecare produs compus
- Posibilitatea de a putea filtra după mai mult criterii, în calitate de client ca utilizator
- Crearea mai multor angajați, care la rândul lor să se autentifice în funcție de programul pe care l-ar avea la locul de muncă

Concluzie:

Cu ajutorul acestui proiect am învățat multe lucruri noi precum: ce este și cum funcționează serializarea și deserializarea. Am învățat să lucrez cu stream-uri. Am folosit noi design pattern – uri ca observer sau composite.

Bibliografie

[1] https://refactoring.guru/

[2] https://dzone.com/articles/how-to-read-a-big-csv-file-with-java-8-and-stream

[3] https://dzone.com/articles/how-to-read-a-big-csv-file-with-java-8-and-stream

[4] https://www.codota.com/code/java/classes/java.io.ObjectOutputStream

[5] https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/