FOOD DELIVERY MANAGEMENT SYSTEM

Nita Iosif-Gabriel

Grupa 30223

# CUPRINS

[1. Obiectivul temei 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297886)

[3. Proiectare 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297887)

[4. Implementare 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297888)

[5. Rezultate 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297889)

[6. Concluzii 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297890)

[7. Bibliografie 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297891)

Obiectivul temei

Obiectivul principal al acestei teme este proiectarea si implementarea unei aplicatii pentru gestionarea livariilor de mancare pentru un restaurant. Clienti pot comanda diverse produse din meniul resaturantului. Sistemul implementat are trei tipuri de utilizatori si anume: administrator, angajat obisnuit si client.

Administratorul poate incarca toate produsele din meniul companiei dintr-un fisier de tipul .csv.

Acesta poate sa modifice produsele din meniul companiei: sa adauge, sa stearga, sa modifice detaliile produselor. De asemenea, acesta poate sa genereze raporturi depsre comenzile plasate de catre clienti dupa diferite criterii cum ar fi:

1. Intervalul de timp al comenzilor: comezile plasate intre o ora minima de inceput si o ora maxima de terminat.
2. Produsele comandate de mai mult de de un anumit numar de ori.
3. Clientii care au comandat de mai mult de un anumit numar de ori si a caror comanda a fost mai mare decat un anumit pret dat.
4. Toate produsele care au fost comandate intr-o anumite zi impreuna cu frecventa lor.

Clientul iti poate crea propriul cont si sa se logheze in aplicatei folosind numele de utilizator si parola alese. Acesta poate, de asemenea sa vada toate produsele din meniul companiei. Totodata, poate sa caute un anumit produs din meniu dupa diferite criterii: titlu, rating, cantitatea de calorii, cantitatea de proteine, cantitatea de grasimi, cantitatea de sodiu si pretul total al comenzii.

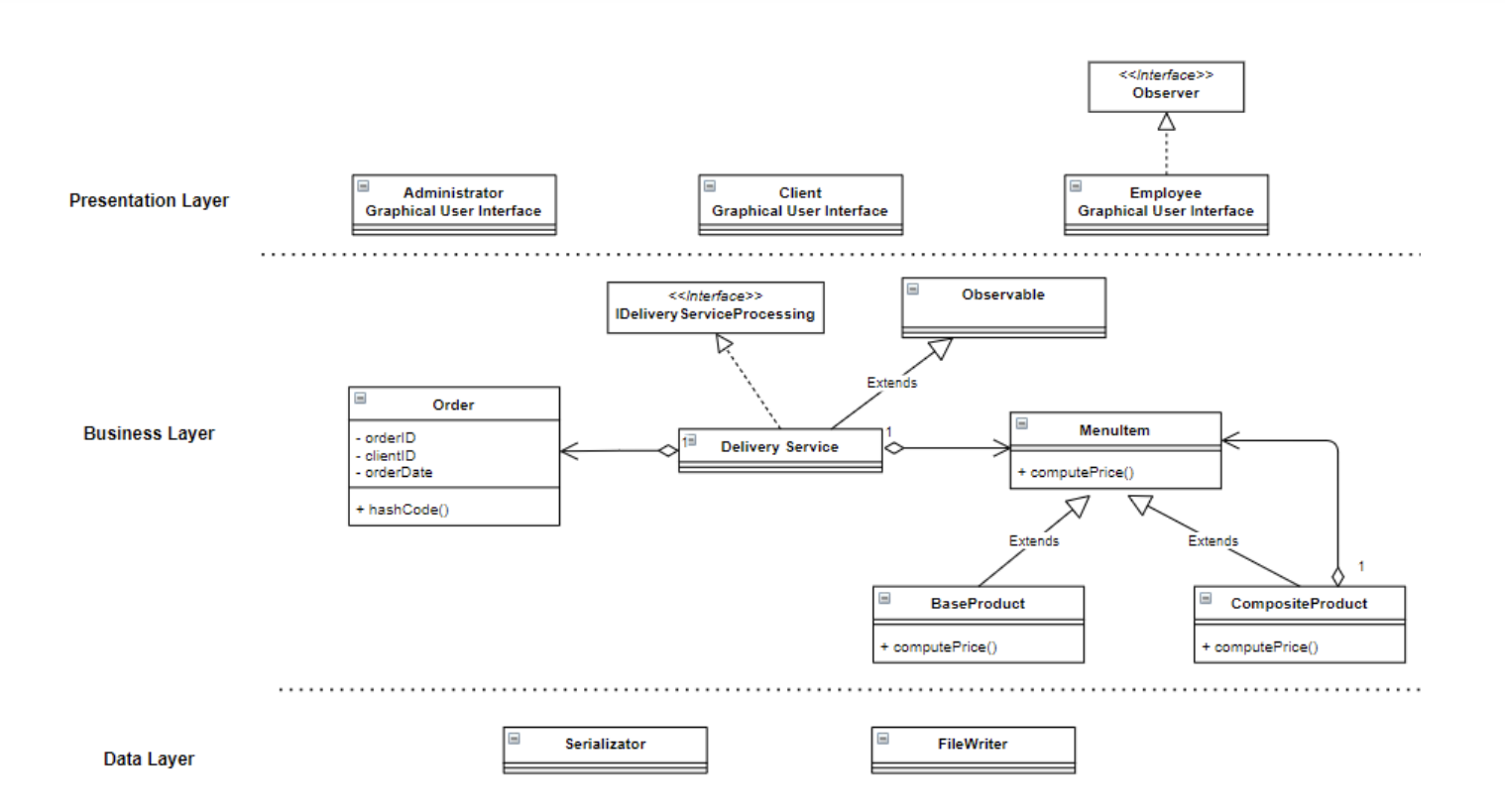
Angajatul va fi notificat de fiecare data cand o noua comanda este plasata pentru a pregati produsele comandate de catre client.

* Dezvoltarea unor scenarii in care aplicatia ve fi dezvoltata
* Alegerea structuriilor de date, structura proiectului trebuie sa fie una eficienta
* Impartirea corespunzatoare a claselor si a rolurilor acestora
* Implementarea si dezvoltarea algoritmilor corespunzatori pentru toate cazurile
* Implementarea solutiilor, combinarea algoritmilor
* Testarea si gasirea cazurilor care pot cauza probleme aplicatie, remedierea pe cate se poate a acestora.

Analiza problemei, modelare, scenario, cazuri de utilizare

Pentru a implementa sistemul de gestionare a restaurantului trebuie sa cream o interfata grafica care va contine operatiile necesare pe care administratorul, angajatul si clientii le pot efectua. Administartorul poate incarca produsele, le poate modifica, adauga, sterge crea diferite tipuri de rapoarte. Clientul poate crea o noua comanda care prespune calcularea pretului si generarea unei chitantei cu formatul .txt, poate cauta produse dupa diferite preferinte.

Diagrama din resursele ajutatoare a fost de mare ajutor pentru ca m-am putut inspira pentru a implementa acesata aplicatie



Am definit si implementat clasele din diagrama prezentata. Am folostie Composite Design Pattern pentru definirea claselor pentru produsele de baza, produsele compuse si pentru meniu. Am folosit Observer Design Pattern pentru a notifica angajatul de fiecare data cand o noua comanda este plasata.

Am implementat o clasa care suprascire toate metodele pentru operatiile administratorului si clientilor utilizant o colectie predefinita de tip HashTable. Cheia este generate automat, fiind bazata pe clasa corespunazatoare a comenzilor avand asociata mai multe produse din meniu.

Produsele de bază utilizate initial pentru popularea meniului vor fi incarcate din fisierul .csv dat, utilizand expresii lambda si procesarea stream. (Administratorul poate, de asemenea, sa adauge noi produse de baza.)

Produsele din meniu, comenzile si utilizatori var fi salvati utilizant serializarea pentru a putea fi disponibili pentru viitoare utilizari ale aplicatiei utilizand deserializarea.

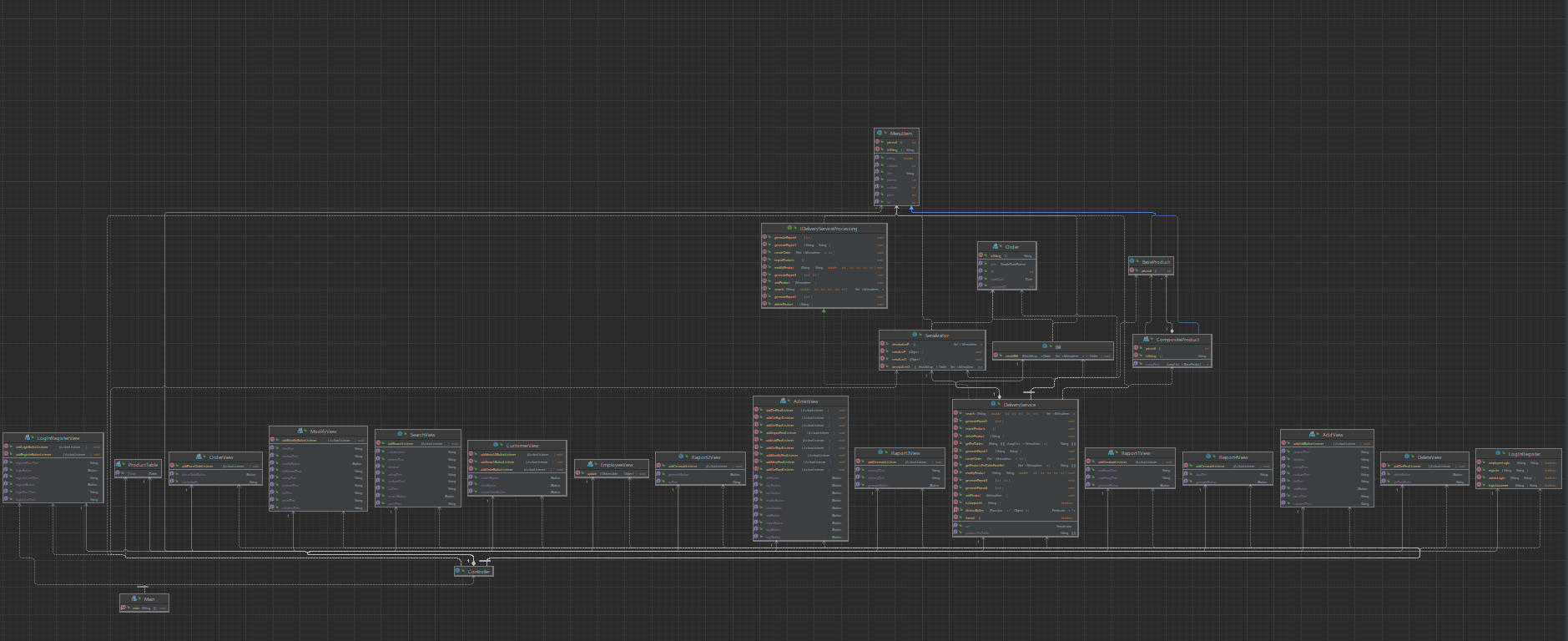
# Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfețe, relații, packages, algoritmi, interfața utilizator)

## Package -> Class

|  |  |
| --- | --- |
| **Package** | **Class** |
| main -> java -> **BuisinessLogic** | BaseProduct  CompositeProduct  DeliveryService  IDeliveryServiceProcessing (interface)  LogInRegister  MenuItem  Order |
| main -> java -> **DataLayer** | Bill  Serializator |
| main -> java -> **View** | AddView  AdminView  Controller  CustomerView  DeleteView  EmployeeView  LogInRegisterView  ModifyView  OrderView  ProductTableView  Report1View  Report2View  Report3View  Report4View  SearchView |
| main -> **java** | MainClass |

|  |  |
| --- | --- |
| **Class** | **Descriere** |
| BaseProduct | Clasa care extinde clasa MenuItem si care conține un constructor cu toti parametrii corespunzatori unui produs de baza si o metoda care calculeaza si returneaza pretul produsului. |
| CompositeProduct | Clasa care extinde clasa MenuItem si care are un singur parametru, o listă de produse de baza. Clasa contine si un constructor cu doi parametrii, titlul produsului compus si lista de produse de baza pe care trebuie sa o contina. In cadrul constructorului setez titlul si calculez restul parametrilor aferenti. Adun rateing-ul, caloriile, proteinele, grasimile si sodiul tuturor produselor din lista primita ca parametru in variabilele aferente. Pretul il calculez cu ajutorul metodei abstractă suprascrisă din clasa MenuItem care aduna pretul fiecarui produs din lista de produse. Rating-ul il setez ca fiind suma rating-urilor calculate mai sus impartita la cate produse sunt in total in lista de produse. Restul parametrilor ii setez cu suma calculata in variabilele aferente de mai sus. |
| DeliveryService | Clasa care contine metodele suprascrise din interfata IDeliveryServiceProcessing pe care o impementeaza. Are doi parametrii, un obiect de tipul Serializator utilizat pentru serializare si deserializare si un obiect de tipul Bill utilizat pentru crearea fisierelor asociate chitantelor comenzilor. Cele zece metode sunt suprascrise aferent operațiilor pe care trebuie sa le execute. Clasa mai conține si metoda formed care returneaza false numai daca listele de produse ale meniului ramane goale. Mai am si doua metode care ajuta la popularea tabelul afisat in interfaaa grafica, o metoda care returnează true dacă titlul unui produs primit ca parametru este produs compus și o metoda care ne ajută sa alegem doar produsele distincte din fișierul .csv în funcție de titlu. |
| IDeliveryServiceProcessing | Interfata care contine declararea metodelor pentru actiunile administratorului si ale clientilor. |
| LogInRegister | Clasa asociata inregistrarii si logarii clienților. Contine doua metode, una pentru register si una pentru login. Metoda pentru register verifica in fisierul care contine toate numele de utilizatori si parolele asociate din cadrul companiei daca numele de utilizator pe care clientul vrea sa il foloseasca este deja utilizat sau nu. Daca este utilizat returneaza false si true in cazul contrar. Metoda de login cauta in fisierul de care am spus mai sus daca numele de utilizator introdus exista si daca parola introdusa este cea aferenta. Daca totul coresunde, returneza true, altfel false. |
| MenuItem | Clasa abstracta care implementeaza interfata Serializable. Clasa are toti parametrii asociati unui produs din meniu: titlu, rating, calorii, proteine, grasimi, sodiu si pret. Avem metoda abstracta care calculeaza pretul si metoda suprascrisa toString. |
| Order | Clasa aferenta comenzilor. Avem trei parametrii importanti, ID-ul comenzii, ID-ul clientului si data la care a fost plasata comanda. ID-ul comenzilor este incrementat automat. Avem si metoda suprascrisa de hashCode care calculeaza cheia unde trebuie sa fie asociata comanda si setul de produse comandate. |
| Bill | Clasa care contine metoda care scrie în fisierul corespunzator factura comenzii plasate. |
| Serializator | Clasa care contine metodele de serializare si deserializare pentru produsele din meniu si comenzile plasate. |
| AddView | Clasa care proiecteaza fereastra grafica aferenta adaugarii unui produs in meniul restaurantului. Pentru a adauga un produs de baza vom completa toate campurile aferente si vom apasa butonul de adaugare. |
| AdminView | Clasa care proiecteaza fereastra grafica cu butoanele aferente fiecarei operatii pe care o poate face administratorul si ascultatorii. |
| CustomerView | Clasa care proiectează fereastra grafică cu butoanele aferente fiecărei operații pe care o poate face clientul și ascultătorii aferenți. |
| LogInRegisterView | Clasa care proiecteaza fereastra grafica pentru inregistrarea si logarea clientilor in aplicatie impreuna cu metodele si ascultatorii aferenti. |
| Controller | Clasa care implementează fiecare metodă ascultătoare din toate ferestrele corespunzătoare. |
| DeleteView | Clasa care proiecteaza fereastra grafica asociata stergerii unui produs din tabel. Selectam un produs din tabel, apasam butonul pentru titlul care pune titlul intr-o eticheta si apoi apasam butonul de delete care va sterge produsul cu titlul selectat. |
| EmployeeView | Clasa care proiecteaza fereastra grafica pentru angajat. Atunci cand se plaseaza o noua comanda, aceasta va aparea dupa comenzile deja plasate inaintea ei sau chiar ea va fi prima. |
| ProductTableView | Clasa care proiecteaza fereastra grafica care contine tabelul folosit de fiecare operatie care are nevoie de afiaare sub forma de tabel. |
| ModifyView | Clasa care proiecteaza fereastra grafica aferenta modificarii unui produs din tabel. Se selecteaza produsul corespunzator, se apasa butonul de titlu pentru scrierea in eticheta corespunzatoare a titlului produsului selectat si apoi se completeaza campurile care se vor schimbate si se apasa butonul de modificare. |
| OrderView | Clasa care proiecteaza fereastra grafica corespunzatoare plasarii unei comenzi. Se selecteaza produsele care se vor comandate, se introduce ID-ul clientului si se plaseaza comanda. |
| Report1View | Clasa care proiecteaza fereastra grafica pentru generarea primului raport. |
| Report2View | Clasa care proiecteaza fereastra grafica pentru generarea celui de-al doilea raport. |
| Report3View | Clasa care proiecteaza fereastra grafica pentru generarea celui de-al treilea raport. |
| Report4View | Clasa care proiecteaza fereastra grafica pentru generarea celui de-al patrulea raport. |
| SearchView | Clasa care proiecteaza fereastra grafică aferentă căutării unui produs de către clientul corespunzător. |
| Main | Clasa care contine metoda main si care porneste aplicatia. |

Diagrama UML:



Interfata:

Administrator:

O imagine care conține text, captură de ecran, interior, dulap

Descriere generată automat

Client:

O imagine care conține text, captură de ecran, interior, computer

Descriere generată automat

Angajat: O imagine care conține text

Descriere generată automat

Implementare

Interfata IDeliveryServiceProcessing: interfata contine declararea tuturor metodelor operatiilor pe care le poate executa administaratorul respectiv clientul am adaugat si comentariile JavaDoc corespunzatoare. O imagine care conține text

Descriere generată automat

Clasa DeliveryService impreuna cu parametrii si toate metodele suprascriese din interfata prezentata mai sus

O imagine care conține text

Descriere generată automat

Metoda de cautare a unui produs din meniu am implementat-o in functie de fiecare camp al produselor din meniu. Daca valoarea primita ca parametru este diferita de -1, atunci cu ajutorul expresiilor lambda caut in meniu produsul care are acele atribute.

O imagine care conține text

Descriere generată automat

Metoda care genereaza raportul 2, aceasta metoda afiseaza produsele care au fost comandate de mai multe ori decat un numar specificat. Pentru fiecare comanda din HashMap-ul destinat comenzilor adaug toate produsele comandate intr-o lista de produse. Pentru fiecare produs din lista ii este calculate frecventa cu ajutorul metodei frequency din clasa Collections. Apoi parcurg vectorul de frecvente si setez valoare 0 pe pozitiile unde se gaseste un duplicat pentru a ramane numai frecventele unice pentru produse unice cu valori diferite de 0. Apoi vectorul de frecventa este parcurs si daca frecventa este mai mare sau egala cu numarul dat ca parametru se afiseaza produsul corespunzator. Cam asa au fost implementate si restul raporturilor, nu o sa le adaug pentru nu a ingreuna fara motiv documentatia O imagine care conține text

Descriere generată automat

Clasa Controller

Clasa Controller impreuna cu parametrii si metodele implementate care ajuta la comunicare dintre utilizator si masina.

O imagine care conține text

Descriere generată automat

In constructor adaug ascultatorii pentru butoanele interfete si creez obiecte noi pentru fiecare operatie realizata de Admin, angajat si client.

O imagine care conține text

Descriere generată automat

Subclasa care implementeaza ActionListener pentru butoanele din interfata grafica.

O imagine care conține text

Descriere generată automat

Clasa care implementeaza ascultatorul pentru metoda de modificare, daca campul din fereastra este gol acesta sa ramana -1 variabila auxiliara, altfel sa se schimbe cu valoarea dorite de catre admin.

O imagine care conține text

Descriere generată automat

Clasa AdminView

Clasa AdminView cu atributele necesare creeari unei interfete usor de folosit si de catre o persoana nespecializata. Restul interfetelor sunt structurate asemanator cu aceasta.

O imagine care conține text

Descriere generată automat

Constructorul clasei in care am instantiate fiecare componenta.

O imagine care conține text

Descriere generată automat

Metodele get si cele de adaugare a ascultatorilor necesare pentru implementarea si functionare corecta a unei interfete grafice.

O imagine care conține text

Descriere generată automat

Rezultate (testare)

Toate rezultatele (testele) asociate acestei aplicatii sunt afisate de catre interfetele grafice, de catre fisierele .txt aflate la resources, sau in consola IDE-ul (si anume rapoartele, nu a fost specificat unde trebuiau salvate si am ales cea mai usoara metoda pentru a scurta proiectarea si implementarea aplicatiei).

Concluzii

In urma implementarii acestei aplicatii, pot sa declar ca mi-am insusit mai bine cunostintele din semestrul anterior referitoare la Java si am acumulat noi informatii, cum ar fi implementarea expresiilor lambda si procesarea informatiilor cu stream-uri, diverse design pattern-uri si JavaDoc. Dezvoltarea treptata a aplicatiei m-a ajutat sa vin cu idei cat mai diverse creative si sper sa cred eficiente pentru rezolvarea problemei. Ca si dezvoltari ulterioare as mai finisa interfetele grafice deoarece au fost facute putin in graba, as mai adauga mesaje pentru a fi mai usor ca aplicatia sa fie folosite si de persoane care nu au cunostintele necesare navigarii unei astfel de aplicatii.

Bibliografie

Nu am avut nici o sursa din care sa ma inspir la redactarea acestei documentatii. Acestea sunt ideile mele care au venit pe parcursul proiectarii si implemetarii proiectului si anumite ideei care mi-au venit pe moment. La implementare am folosit sursele prezentate la laborator si cand intampinam o eroare cautam pe google pentru a gasi o rezolvare corespunzatoare.