

Documentație laborator tema 4

Restaurant

Tehnici fundamentale de programare

Student: Dunca Denisa Mihaela

Grupa: 30227

1. **Obiectivul temei**

Cerința temei este de a crea și implementa o aplicație care sa reprezinte un restaurant. Aplicația are ca primă fereastră, o fereastră de log in, unde utilizatorul va introduce username-ul si parola. În funcție de rolul pe care îl are utilizatorul în fișierul administrator, client sau angajat. Dacă utilizatorul are rolul de administrator, acesta poate sa importeze produsele aflate in fisierul products.csv, poate sa adauge produse noi in meniu, să modifice produsele și sa le șteargă. De asemenea, acesta poate să creeze un produs nou care sa conțină mai multe produse mai mici, acest lucru se face prin apelarea metodelor composite produsct. În plus, administratorul poate sa accese 4 rapoarte legate de cum s-au desfășurat order-urile.

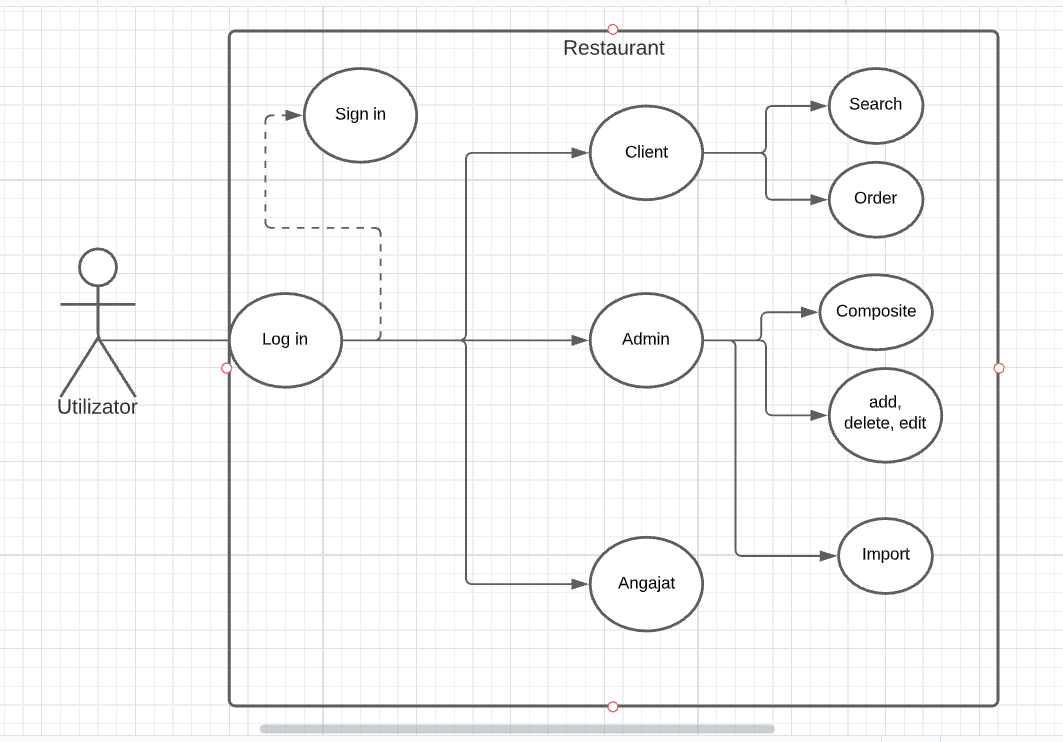
Dacă utilizatorul este client, acesta poate să vadă meniul cu toate produsele disponibile, dar doar după ce administratorul face importul la produse, apoi poate sa de-a search la produse in lista care pot indeplinii diferite criterii. Atunci cand alege produsele pe care le doreste, clientul le adauga în căruciorul de cumparaturi , vede pretul total, si depunde comanda. Aceasta comanda este treansmisa spre angajat, iar atunci cand utilizatorul se logeaza ca si angajat are acces la notificarea legata de comanda.

Obiectivele secundare ale proiectului:

* Analizarea problemei prin înțelegerea corectă a cerinței și modelarea de scenarii și cazuri de utilizare.
* Proiectarea unei soluții care să se plieze pe toate cazurile pe care le introduce utilizatorul.
* Implementarea soluției alese prin scrierea codului implicit și a interfeței utilizator.
* Observarea rezultatelor finale și concluziile.

1. **Analiza problemei**

Analiza problemei a constat prin înțelegerea pe deplin a cerinței și a informațiilor pe care dorim să le primim la generarea unui rezultat final. S-a creat o schemă inițială care trebuie să prezinte în mare ceea ce se urmărește prin acest proiect.

. Un mod prin care se poate scoate în evidență această structură este generarea unei Diagrame USE CASE.

În diagrama de mai sus este prezentată structura logică a proiectului. Sistemul este aplicația în sine prin care utilizatorul poate sa introduca date in cele 3 tabele. Introducerea datelor este dependentă de actor, precum și apăsarea pe butoanele de pe interfata și ieșirea din program. La deschiderea aplicației utilizatorul trebuie să urmarească următorii pași : logare, lucur cu interfata in functie de rol si iesire.

Logare:

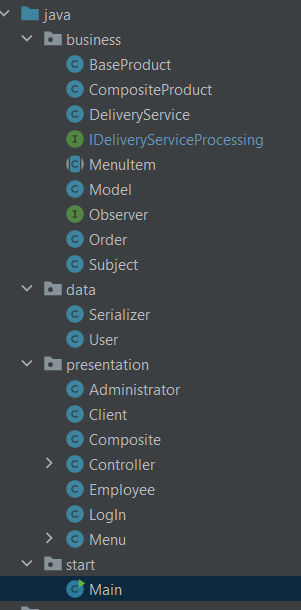
* Utilizatorul introduce un username si o parola, acesta este directioneat spre o fereastra in functie de rolul pe care il are.
* Usernameul este unic, iar daca utilizatorul nu are un cont poate sa isi faca unul , cu rolul de client.

Administrator:

* Poate sa importeze produsele din products.csv
* Sa adauge un produs nou
* Sa modifice orice field dintr-un produs
* Sa stearga un produs
* Sa creeze un produs compus din mai multe produse si sa ii dea un nume
* Sa genereze 4 rapoarte

Clientul:

* + Poate sa vada lista cu produsele, dar doar dupa ce a dat administratorul import
  + Sa dea search la produse in functie de mai multe filtre
  + Sa selectelze din produsele cautate, pe cele pe care doreste sa le cumpere si sa le adauge in cos
  + Sa vada un pret total al comenzii
  + Sa comande



Angajatul:

* Vede comenzile facute de clienti

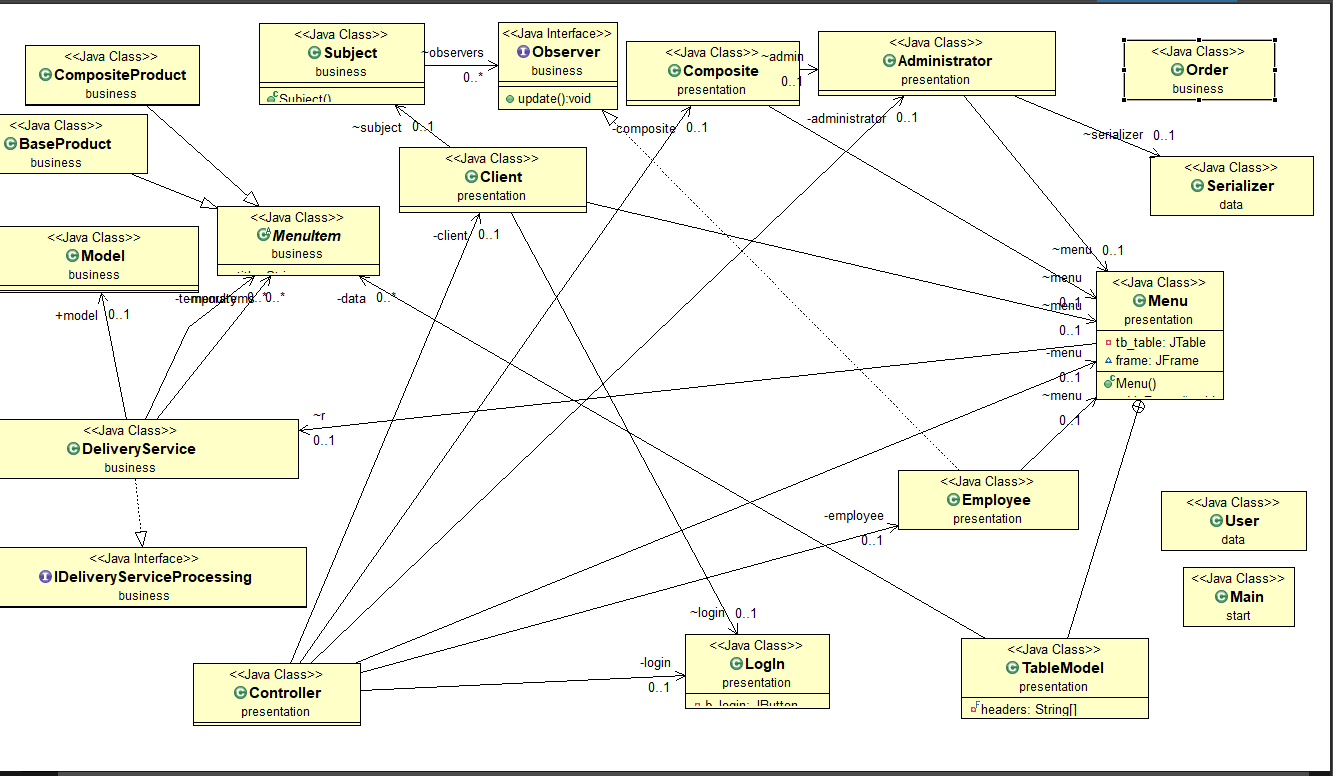
Aplicatia este impartita in mai multe pachete. Pachetul bussines contine in primul rand clasa abstracta MenuItem, cele doua clase care o extind BaseProduct si CompositeProduct, Interfata DeliveryServiceProdeccing si clasa DeliveryService care contine toate metodele folosite in proiect, si clasa Order. Pachetul data contine clasa User care ia utilizatorii salvati in fisierul user.txt si clasa de serializare. Pachetul presentation contine toate interfetele: Administrator, Client, Employee, Composite, Login, Menu si clasa Controller unde se afla toate listenerele la butoane.

1. **Proiectare**

Trecând la partea practică de rezolvare a problemei, și anume proiectarea în limbaj OOP, au fost urmăriți mai mulți pași, primul fiind crearea unei Diagrame de pachete urmărind modelul MVC (Model-View-Controller), dar si modelul Layers folosind Business Layer, Presentation Layer, Model Layer. Controllerul face legătura dintre clasele din Model și View prin intermediul butoanelor apăsate.

Intrând într-o analiză mai amănunțită a proiectului, am generat și diagrama UML. Aici se pot vedea în detaliu legăturile dintre clase, metodele folosite, precum și structura logică a întregului proiect.

Pentru a crea diagrama UML am folosit o extensie din Eclipse. Pe diagrama se observa legaturile dintre clase si modul in care a fost gandit acest proiect.



**Implementare**

Pentru implementarea acestui program a trebui sa implementez mai multe design patternuri precum Composite Design Pattern, Observer Design Pattern, Design by Contract method si o metoda well formed, o fereastra de log in care se verifica cine face operatiile, administratorul, angajatul sau clientul, pentru in fiecare in parte trebuiau implementate diferite operatii folosind streams sau un observer.

**Pachetul bussines**

In primul rand, am implementat o clasa abstracta numita MenuItem care contine toate fieldurile cerute pentru produse: title, raiting, calories, proteins, sodium, fats, price. Aceasta clasa este extinsa de catre BaseProduct, respectiv CompositeProduct. Apoi am si clasa Order care defineset pentru comenzi 3 atribute, data, id-ul orderului si numele clinetului care a facut comanda. In al doilea rand,am implementat si interfata IDeliveryServiceProcessing care contine toate metodele din clasa DeliveryService. Aceasta clasa, DeliveryService are principalele metode folosite pentru a face logica din spatele aplicatiei:

* Metoda importProducts este o metoda folosita pentru a imparti elementele primite dintr-un file in elemente de tipul BaseProduct, cu toate fieldurile din MenuItem. Aceasta clasa este implementata cu streams si expresii lambda.
* Metoda addMenuItem este functia care adauga in lista de produse un nou produs, clasa este chemata in interfata administratorului.
* Metoda deleteMenuItem este functia care sterge un item din lista de produse in functie de numele pe care adimistratorul il introduce. Aceasta metoda parcurge lista de produse si cand gaseste un produs ce este egal cu cel introdus de administrator ii da remove din lista.
* Metoda editMenuItem este folosita de catre administrator pentru a modifica oricare dintre fieldurile unui anumit produs, aceasta modificare se face in functie de titlul produsului.
* Metodele createOrder si addItemToOrder sunt folosite pentru a crea o lista initiala care sa contina datele introduse de client, respectiv pentru a adauga un nou item in order dupa ce a fost deja adaugat unul.
* Metodat sendOrder este folosita pentru a verifica id-urile comenzilor
* Metoda ComputePrice este folosita pentru a genera suma dintre toate preturile produselor adaugate in cosul de cumparaturi
* Metoda doSearch este metoda care foloseste streams si lambda expressions penrtu a filtra produsele din lista de produse in functie de ce introduce clientul
* Metodele temporaryComposite si saveComposite sunt folosite pentru a face produslui composite
* Aici apar si cele patru metode destinate celor patru rapoarte de la administrator, toate sunt implementate folosind stream si indeplinesc urmatoarele: report1 metoda care primeste doua inturi (ore) si care verifica daca sunt comenzi intre orele respective si pune rezultatul in fisier, report2 primeste ca parametru un int si verifica ce produse au fost comandate de mai mult decat numarul dat, report3 primeste ca parametru doua numere, unul pentru numarul de orderuri si al doilea pentru pretul minim si report4 care primeste ca parametru o zi luna sub forma de int si cauta produsele comandate in ziua respectiva si de cate ori au fost comandate.
* Tot aici si afla si metodat generateBill care creaza pentru ultimul order facut un fisier.txt cu informatiile din order.

**Pachetul data**

Pachetul data contine o clasa Serializer care face serializarea si o clasa User care ia elementele din fisierul user.txt, fisier in care se afla datele despre utilizatori, adica rolul, numele si parola. Tot in pachetul data se afla si clasa Serializer care contine cele doua metode de serializare si respectiv de deserializare.

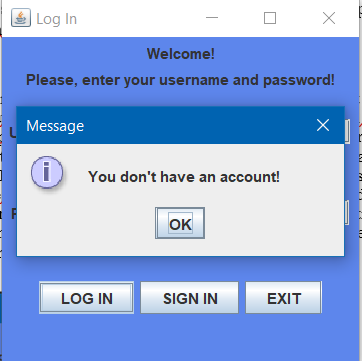
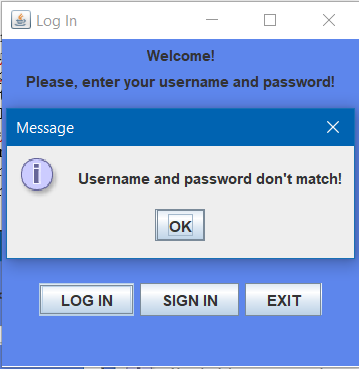
**Pachetul presentation**

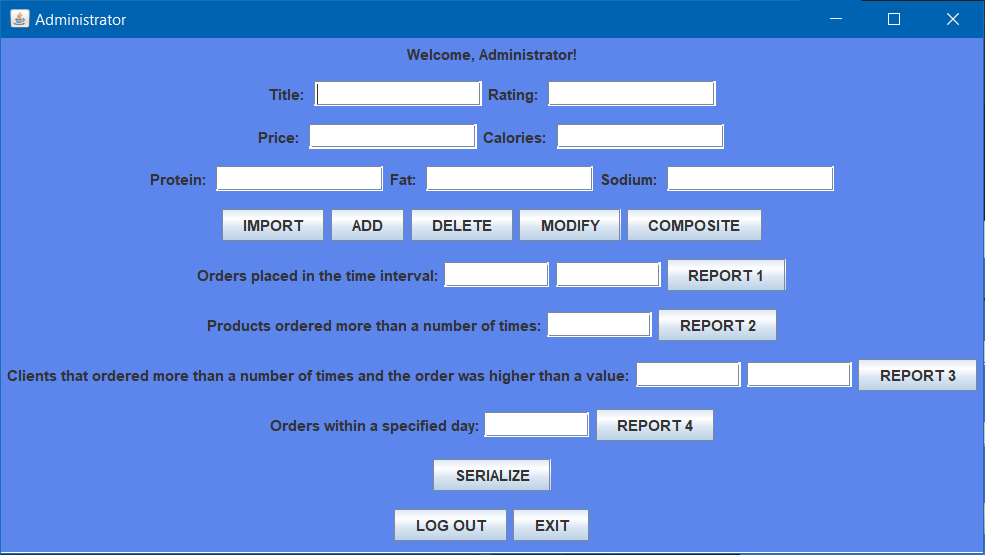
Pachetul presentation contine toate interfetele si controller-ul, interfetele au fiecare un rol important si metode care sunt apelate in controlle penrtu a face controlul de la butoane, interfetele sunt: Adiministrator, Client, Composite, Employee, Login, Menu.

* Clasa Administrator contine pe langa toate elementele specifice JFrame si getterele pentru aceastea, si trei metode care apeleaza metodele din DeliveryService, acestea sunt add, delete, si edit care are si o verificare suplimentara pentru cand textfieldul unui anumit field este gol.
* Clasa Client are si ea cateva metode, o metoda care pune elementele selectate de client in cos, o metoda care creste idul orderului de fiecare data cand se creaza un order nou, o metodat care face pretul total al comenzii, si metoda metru search care filtreza produsele.
* Clasa Composite este folosita pentru a apela metodele din DeliveryService care fac CompositProduct.
* LogIn este folosita pentru a verifica logarea utilizatorului si pentru a genera mesaje de eroare in cazul in carea logarea nu este corect facuta. La fel si pentru a ii oferii utilizatorului nou oportunitatea de a-si crea un cont nou si de a fii adaugat in fisier in cazul in care nu este doar apasand butonul de sign in.
* Clasa Employee este clasa definita pentru angajati, acesta primeste o notificare cu comanda de fiecare data cand se face o comanda prin implementarea unui observer.
* Clasa Menu pune elementele luate din products.csv, apeland metoda din DeliveryService, importProducts, intr-un tabel si il afiseaza pe ecran
* In Controller sunt apelate metodele finale din fiecare clasa de tip gui, si este facut un constructor care sa le contina pe toate.

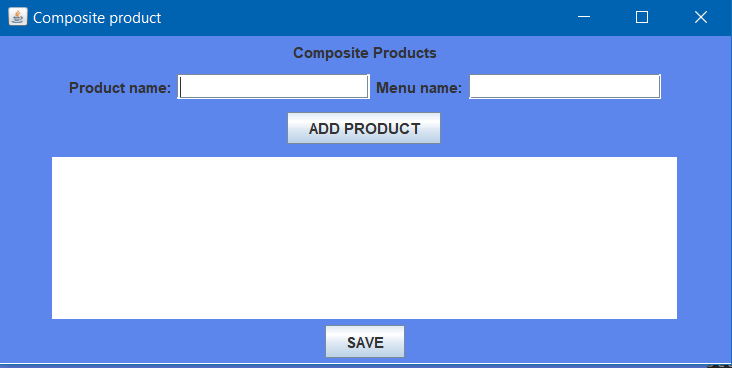
1. **Rezultate**

După rezolvarea problemelor din cod și ajungerea la proiectul final am testat introducand si modificand diferite date de la tastatura. Am facut mai multe teste logandu-ma cu cilenti, administraturi si respectiv angajati si am verificat daca functioneaza fiecare operatie din proiect, am testat in principal fuctionarea rapoartelor pentru diferite tipuri de comenzi in momente diferite introducand o serie de date de la tastatura care sa ma ajute sa observ daca acestea functioneaza sau nu corect. De fiecare data rezultatele returnate de aplicatie au fost puse in fisiere de tip .txt pentru a se putea observa functionare lor. Am creat un fisier txt numit user unde pot sa introduc clienti administratori si angajati care se pot loga, am creat un fiser order.txt unde este adaugata de fiecare data ultima comanda facuta si patru fisere txt penrtu fiecare raport in parte unde sunt puse rezultatele acestora. Cat despre rezultatele din interfata am implementat urmatoarele ferestre:

****Fereastra de login cu cele doua textfielduri pentru nume si parola si mesajele specifice in caz ca nu a fost facuta bine autentifixarea:

Fereastra pentru administrator care are toate fieldurile necesare pentru adaugarea stergerea sau modificarea unui produs, butonul de import de produse din fisierul products .csv si cele 4 rapoarte cu textfiedurile specifice pentru fiecare.

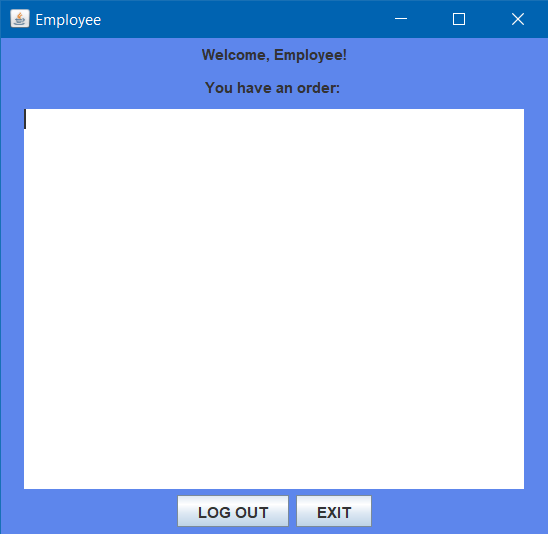
Tot pe fereastra administrator este un buton composite care va deschide o alta fereastra in care administratorul poate sa creeze produse compuse si sa le adauge in meniu:



Fereastra pentru client care contine toate filtrele cu care se poate face cautarea in lista de produse si un cos de cumparaturi unde pot sa isi adauge toate produsele ce vor sa le comande. Tot mai jost se vede si pretul total al comenzii pe care urmeaza sa o faca.

****

Fereastra pentru angajat contine doar un textArea in care de fiecare data cand un client face o comanda, angajatul va primii o notificare cu comanda respectiva.

****

1. **Concluzii**

Per total, întregul proiect este destinat utilizării, și pentru că testărilor și verificărilor făcut aș putea spune că ar trebui să funcționeze pentru orice caz pe care utilizatorul îl introduce.

Pentru acest proiect am reusit sa inteleg cum functioneaza diferite design patternuri cum ar fi: Composite Design Pattern, Observer Design Pattern, Design by Contract method. Am reust sa creez o pagina de login fara a utiliza o baza de date, plus ca am invatat cum functioneaza streams is lambda expresions. A inteles care sunt pasii pentru a face o aplicatie care sa indeplineasca anumite cerinte si cum se utilizeaza serializarea pentru a se salva datele de la o pornire a aplicatiei la alta. A fost un poriect mai difisil cu o multime de taskuri si toate diferite, mult de munca pe interfete si chiar mai multa pe logica din spate a proiectului, dar ma bucur ca am reusit sa indeplinesc fiecare task cerut in comanda.

1. **Bibliografie**

Streams:

<https://www.baeldung.com/rxjava-vs-java-flow-api>

<https://www.baeldung.com/java-9-reactive-streams>

Observer:

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

File Writer:

<https://javarevisited.blogspot.com/2012/01/what-is-assertion-in-java-java.html>

<https://www.geeksforgeeks.org/java-swing-jtable/>

<https://docs.oracle.com/javase/7/docs/api/java/io/BufferedWriter.html>

Search:

<https://stackoverflow.com/questions/49462114/search-for-book-using-multiple-criteria-in-java>

Composite Design Pattern:

<https://docs.oracle.com/javase/7/docs/api/java/awt/MenuItem.html>