

Tehnici de programare

Laborator – Assignment 4

**Food Delivery Management System**

Teacher Assistant: Dr. Marcel Antal

Student: Mihaila Gabriel

Group: 30229

# Assignment Objective

Obiectivul principal al assignment-ului este de a servi ca o modalitate pentru restaurante de a-si gestiona clientii, administratorii si angajatii. Permite gestionarea comenzilor alaturi de angajati si permite angajatilor sa fie la curent cu data si cu ce produse trebuie facute. Administratorii trebuie, de asemenea, sa poata adauga produse sau meniuri noi compuse din mai multe entitati.

Toate datele utilizate de aplicație (clienți, angajați și administratori) trebuie să fie persistente.

Obiectivele secundare ale misiunii sunt, dar nu se limiteaza la:

* Implementarea unei anumite diagrame de clasa in PDF.
* Avand un GUI independent pentru fiecare tip de utilizator.
* Utilizarea expresiilor lambda pentru generarea rapoartelor.
* Folosirea expresiilor lambda pentru generarea functionalitatilor de cautare.
* Utilizarea modelului de design compozit pentru elementele de meniu.
* Crearea unei facturi pentru fiecare comanda.
* Utilizarea metodei Design by Contract.
* Utilizarea modelului de proiectare a observatorului pentru notificarea fiecarei comenzi.
* Salvarea informatiilor folosind serializarea.
* Utilizarea de JavaDoc pentru documentarea claselor si metodelor.

# Problem analysis, modeling, scenarios, use case

## Problem analysis and modeling

Problema necesita modelarea datelor intr-un mod care poate fi procesat in Java. Astfel, l-am impartit in mai multe parti de date: cont, produs de baza, produs compozit si comanda. Intrarea este data ca produse, comenzi sau rapoarte care pot fi create de un utilizator conectat cu contul corect. Rezultatul este dat ca mai multe tabele reprezentand rapoartele si lista articolelor si un fisier txt pentru fiecare comanda.

Modelul pentru fiecare dintre clase este modelat ca:

Account

* id - Integer
* username - String
* password - String
* type – AccountType

BaseProduct

* title – String
* rating – Double
* calories – Integer
* fats – Integer
* sodium – Integer
* price - Integer

CompositeProdct

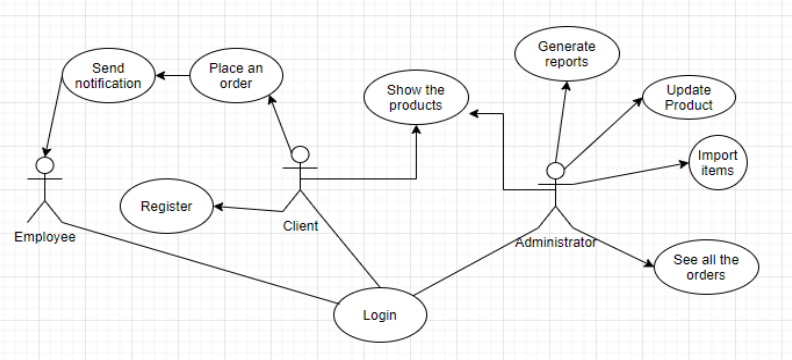
title – String

## Scenarios and use case diagram

Cazurile de utilizare sunt definite de imaginea de mai jos, actorul fiind utilizatorul care doreste sa isi proceseze datele folosind aplicatia.

O explicatie detaliata a functionarii poate fi data ca:

Pot exista 3 actori diferiti:

Employee

* Asteapta o notificare care ii spune ce ordine sa pregateasca.

Client

* Este permisa inregistrarea
* Poate plasa o comanda
* Pot vedea toate produsele disponibile, cu filtre date

Administrator.

* Poate vedea toate produsele
* Poate genera rapoarte ce se bazeaza pe order-uri
* Poate vedea toate order-urile
* Poate actualiza orice produs si sa il plaseze in lista de produse
* Poate importa liste de produse din fiserul products.csv

# Design (design decisions, UML diagrams, data structures, class design, interfaces, relationship packages, algorithms, user interface)

## Design decisions

Dupa cum este impus de cerintele atribuirii, codul foloseste o Arhitectura stratificata care imparte totul in 3 pachete de baza: businessLayer, dataLayer si presentationLayer, alaturi de un pachet de start suplimentar care contine doar clasa principala.

## Relationship Packages

Fiecare dintre pachete imparte clasele dupa cum urmeaza:

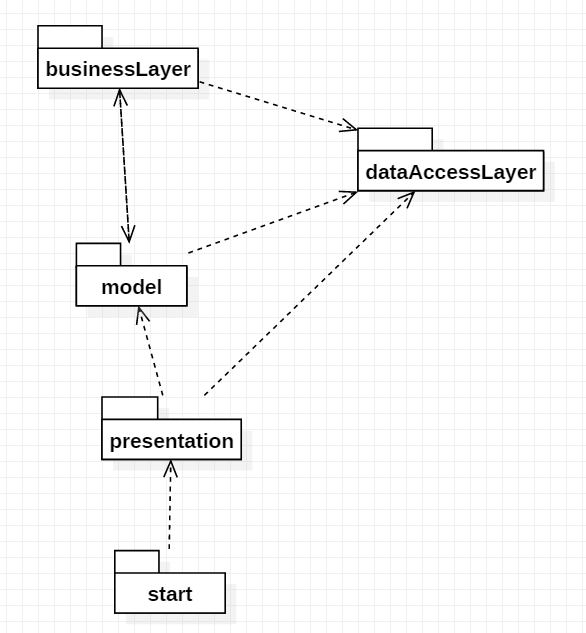
* dataAccessLayer -> Este implicat in clasele care permit citirea si scrierea datelor in baza de date intr-o maniera generica.
* businessLayer -> Este implicat cu unele superclase peste modele care permit accesul si validarea mai usoara a datelor.
* model -> Modeleaza datele intr-un mod care poate fi utilizat de limbajul de programare.
* presentation -> Este implicat in clasele care genereaza interfata grafica si iau intrarea care va fi procesata de dataAccessLayer.
* start -> Conține doar clasa principală care începe acțiunea pentru proiect.
* Vizualizarea care are toate clasele necesare pentru a afisa informatiile pe ecran si, de asemenea, citeste intrarea din primul GUI.

Diagrama arata o reprezentare mai usor de inteles a claselor si a relatiilor acestora.

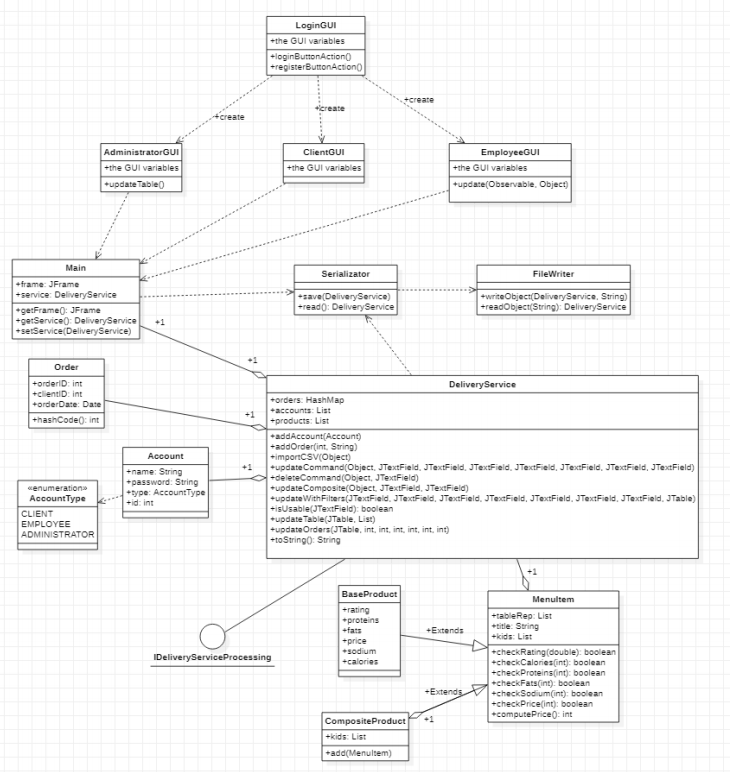
## Data structures

Datorita naturii sarcinii, nu exista multe structuri de date notabile, unele dintre cele mai interesante ar putea fi:

* HashMap<Order, List<MenuItem> -> care este folosit pentru a pastra persistent toate comenzile cu articolele corespunzatoare.
* Date -> stocheaza data pentru o comanda data.

## Class design and UML diagrams

Voi incepe cu o diagrama care prezinta toate clasele si relatiile dintre ele ca un UML



Principalele clase necesare pentru proiect pot fi impartite in

* Account -> Care stocheaza conturile necesare pentru conectare.
* AccountType -> Este o enumere care specifica ce tip este un cont: CLIENT, ANGAJAT, ADMINISTRATOR.
* FileWriter -> Scrie datele persistente intr-un fisier dat.
* Serializator -> Foloseste clasa FileWriter pentru a scrie datele mai general.
* AdministratorGUI -> Clasa utilizata pentru GUI-ul administratorului.
* ClientGUI -> Clasa utilizata pentru GUI-ul clientului.
* EmployeeGUI -> Clasa utilizata pentru GUI-ul angajatului..
* LoginGUI -> Primul GUI care permite utilizatorului sa se conecteze sau sa se inregistreze.
* Main -> Clasa principala a aplicatiei.
* BaseProduct -> Stocheaza datele pentru un produs de baza care poate fi utilizat in produse mai complexe.
* CompositeProduct -> Combina mai multe elemente de meniu intr-un singur produs
* DeliveryService -> Clasa centrala a aplicatiei de program care implementeaza multe dintre functii necesara pentru functionalitate, tinand totodata evidenta persistentei necesare.
* Order -> Modeleaza datele necesare unei comenzi.

## Interfaces

Exista 2 interfete prezente in aplicatie.

IDeliveryServiceProcessing -> Defineste functiile care trebuie implementate in DeliveryService.

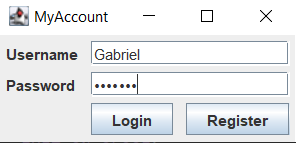
MenuItem -> O interfata pentru un produs vandut la restaurant.

## Algorithms

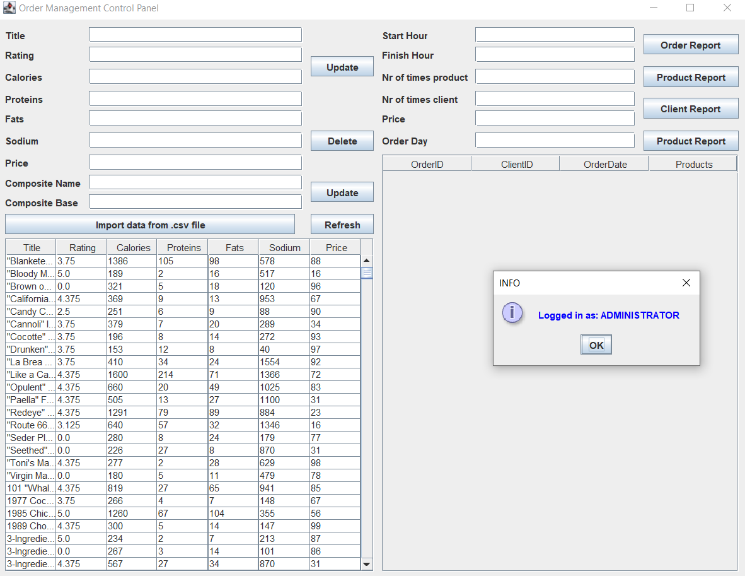
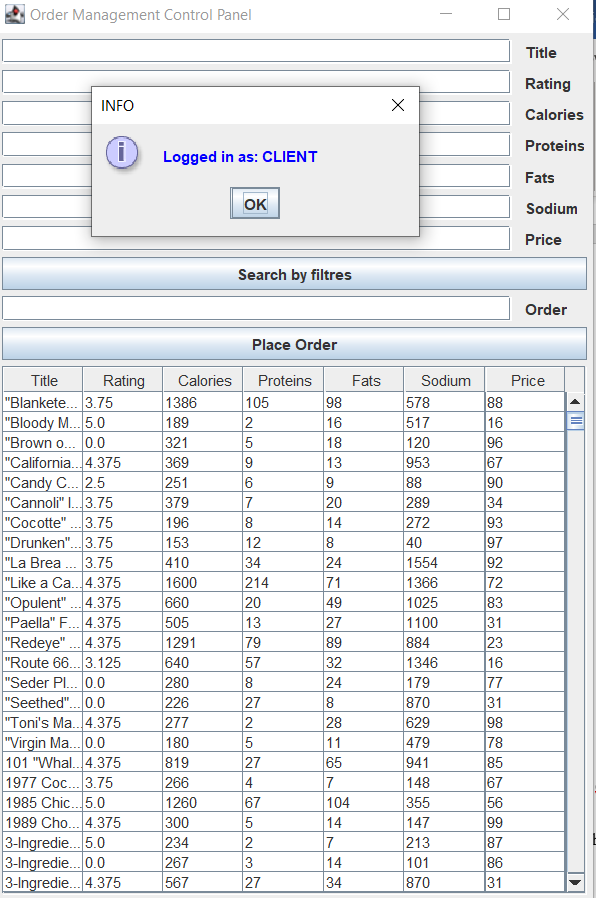
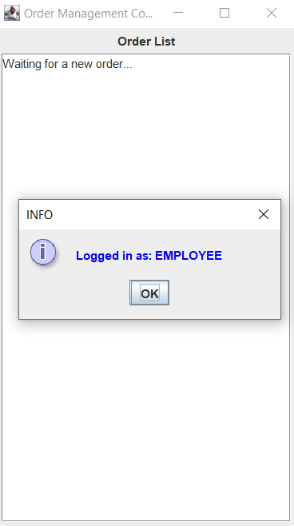
Natura sarcinii nu a necesitat utilizarea unor algoritmi specifici pentru un bun implementare, altele decat utilizarea hashului, chiar daca acestea nu sunt implementate direct in cod.

## User interface

Prima interfata cu care este prezentat utilizatorul este GUI-ul de conectare, care ii permite sa se conecteze fie ca: client, angajat, administrator sau ii permite sa creeze un cont de client nou.

Interfata arata dupa cum urmeaza:

Dupa conectare, interfata se separa in 3 cazuri:



Client

● Are un buton care permite utilizatorului sa vada toate produsele filtrate de multe dintre filtrele prezentate mai sus dupa cum cere el.

● Are un buton care permite utilizatorului sa plaseze comanda specificata in JTextField-ul de mai sus.

● Are un tabel JT care permite utilizatorului sa vada articolele curente de vanzare.

Employee

● Asteapta pana cand se plaseaza o comanda si apoi arata angajatului ca cel mai tarziu ordinea care a fost plasata ca iesire.

Administrator

● Are un buton care permite utilizatorului sa import produse din un fisier prefabricat .csv.

● Are un buton care iti permite sa adaugati / actualizati o baza produs.

● Are un buton care va permite sa stergeti un produs.

● Are un buton care va permite sa adaugati un element la un compozit produs.

● Are un buton care va permite sa reimprospatati produsele si comenzile afisate pe GUI.

● Are un buton care va permite sa creati o comanda bazata pe timp.

● Are un buton care va permite sa creati un raport bazat pe produse vandute

● Are un buton care va permite sa creati un raport pe baza clientilor care au comandat mancare.

● Are un buton care va permite sa creati un raport pe baza produselor vandute intr-o anumita zi.

● Are un tabel JT care arata toate produsele disponibile.

● Are un tabel JT care arata toate comenzile curente disponibile sau rezultatul raportului daca a fost un raport solicitat.

# Implementation

Vom lua toate clasele pe rand si vom analiza toate partile importante.

## Start

Singurul rol al clasei este de a initializa clasele necesare programului si de a-l porni.

## Account

Gestioneaza datele necesare procesarii si stocarii conturilor.

Functia calculeaza ce date ar trebui sa fie afisate in JTable avand tipul de date selectat din JTabbedPane ca parametru 

Mai intai converteste indexul selectat in clasa corecta si acceseaza dataAccessLayer pentru a prelua datele necesare care sunt reprezentate ca campuri private care trebuie setate ca accesibile.

Iesirea finala este o lista de liste de siruri, dintre care prima lista de siruri reprezinta antetul tabelului si urmatoarele reprezinta datele care ar trebui afisate.

## AccountType

Enumera tipul de cont posibil: CLIENT, ANGAJAT, ADMINISTRATOR..

## Product

Creeaza o noua instanta de produs cu urmatoarele field-uri: id, nume, cantitate si pret.

## BaseProduct

Modifica din meniu datele necesare pentru procesarea si stocarea unui produs de baza. Returneaza copiii instantei ca fiind doar instanta in sine

## CompositeProduct

Modeleaza datele necesare procesarii si stocarii unui produs compus din mai multe articole.



Returneaza formatul necesar pentru tiparirea unui articol intr-un JTable, un produs combinat ar trebui sa fie recursiv reprezentat ca linii multiple.

## DeliveryService

Ofera principalele functii necesare procesarii datelor necesare pentru modelare si are, de asemenea, datele

care va fi salvat persistent pentru executii viitoare

Adauga functionalitate suplimentara pentru o data, care permite extragerea zilei din luna cand data are

a fost salvat.

Actualizeaza un JTable, avand ca parametri JTable, o lista a titlurilor antetului si o lista care specifica

toate randurile care vor fi prezente.

Este utilizat pentru a genera rapoartele solicitate de un anumit administrator.

Comanda poate fi separata in 5 cazuri, iar parametrii utilizati din functie depind de cazul solicitat. ei

poate fi separat ca:

● Caz 0 -> Afiseaza toate comenzile stocate in serviciu.

● Cazul 1 -> Afiseaza toate comenzile care au fost date intr-o ora de inceput si de finalizare.

● Cazul 2 -> Afiseaza toate produsele care au fost comandate de mai multe ori.

● Cazul 3 -> Afiseaza toti clientii, de cate ori au comandat peste un anumit pret.

● Cazul 4 -> Afiseaza produsele cu numarul de ori comandat intr-o anumita zi a lunii

## IDeliveryServiceProcessing

Defineste functiile care trebuie implementate in DeliveryService..

## MenuItem

Defineste toate atributele de baza si functiile care trebuie implementate intr-unul din cele 2 produse posibile

tipuri.

## FileWriter

Folosit pentru a citi si scrie datele din aplicatie intr-un fisier.



Scrie o instanta DeliveryService intr-un fisier cu numele dat.

## Serializator

## Permite o utilizare mai usoara a clasei FileWriter.

## AdministratorGUI

Are datele si functiile necesare pentru utilizarea GUI a administratorului.

ClientGUI

Are datele si functiile necesare pentru utilizarea GUI a clientului.

EmployeeGUI

Are datele si functiile necesare pentru utilizarea GUI a angajatilor. Asculta trimiterea unei comenzi de catre un client si apoi foloseste datele trimise pentru a le afisa pentru

angajati.

LoginGUI

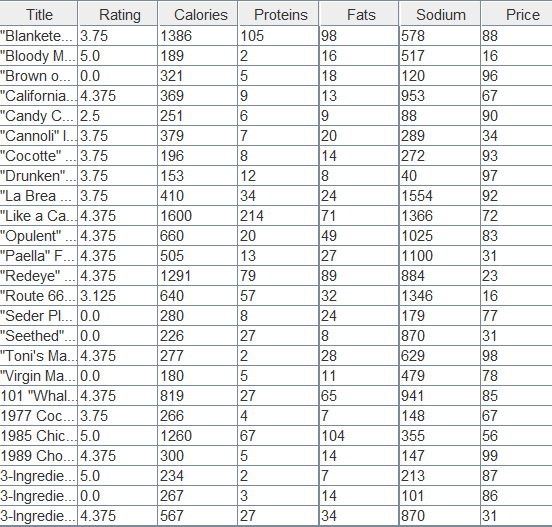
Are datele si functiile necesare pentru utilizarea GUI-ului de conectare. Creeaza un cont CLIENT nou cand este apasat butonul de inregistrare.

# Results

Rezultatele aplicatiei pot fi vazute in mai multe moduri diferite. Lista produselor care pot fi vazute de administrator si pot fi vazute si filtrate de catre un utilizator. Comenzile care au fost plasate de clienti, actionand totodata ca o modalitate de a vedea cele 4 tipuri diferite de rapoarte in acelasi JTable.

O interfata care permite angajatilor sa fie notificat ori de cate ori a fost efectuata o noua comanda plasata, pentru a se pregati sa o faca.

Un fisier text de factura care permite pentru ca toate ordinele sa fie stocate pe disc.



# Conclusion

In concluzie, assignment-ul a servit ca o modalitate excelenta de a invata despre multe tehnici Java puternice, cum ar fi serializarea si afirmare.a Cred ca serializarea poate fi utilizata ca un mod mai rapid si mai flexibil de stocare a datelor in comparatie cu bazele de date, chiar daca acestea pot fi mai rapide in unele scenarii.

In acelasi timp, a servit ca un exemplu foarte real al unei aplicatii care ar putea de fapt sa fie folosita de

un adevarat restaurant. Cred ca o mare cantitate de modele de proiectare care au fost necesare in proiect vor permite acest lucru si mai bine sa fie scrise orice idei in viitor ca si un cod clean.

In viitor, unele dintre imbunatatirile aduse aplicatiei ar putea fi printre:

● Extinderea datelor care pot fi modelate prin adaugarea de clase suplimentare, care pot fi usor integrate

de sistemul de serializare deja existent.

● Adaugarea suportului de retea care permite utilizarea aplicatiei intre mai multe computere.

● Permite inregistrarea tuturor tipurilor de utilizatori posibili.

● Furnizarea unei interfete mai usor de utilizat.

● Utilizarea aplicatiei pentru o companie din lumea reala care are nevoie de ea.

● Permite angajatilor sa vada si orice comenzi care au fost plasate in trecut, in cazul in care o fac

nu reusiti sa faceti ultima comanda la timp.

● Crearea unei versiuni web a aplicatiei care poate fi utilizata mai usor fara a fi nevoie

instalati-l si Java pe toate masinile.

● Realizarea unei versiuni mai generale care poate fi utilizata de alte companii, cum ar fi o librarie sau

magazin de benzi desenate.

# Bibliography

* [Lambda Expressions (The Java™ Tutorials > Learning the Java Language > Classes and Objects) (oracle.com)](https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html)
* [Java 8 Stream Tutorial (winterbe.com)](https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/)
* [Java Stream distinct() with Examples - HowToDoInJava](https://howtodoinjava.com/java8/java-stream-distinct-examples/)
* <https://dzone.com/articles/layers-standard-enterprise>
* [How HashMap works in Java (javarevisited.blogspot.com)](https://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html)