**Logo

Description automatically generated**

**Documentație pentru tema 4 la materia**

**Tehnici de programare fundamentale**

Student: Pop Adrian

grupa:30223

seria:A

An universitar:2020-2021

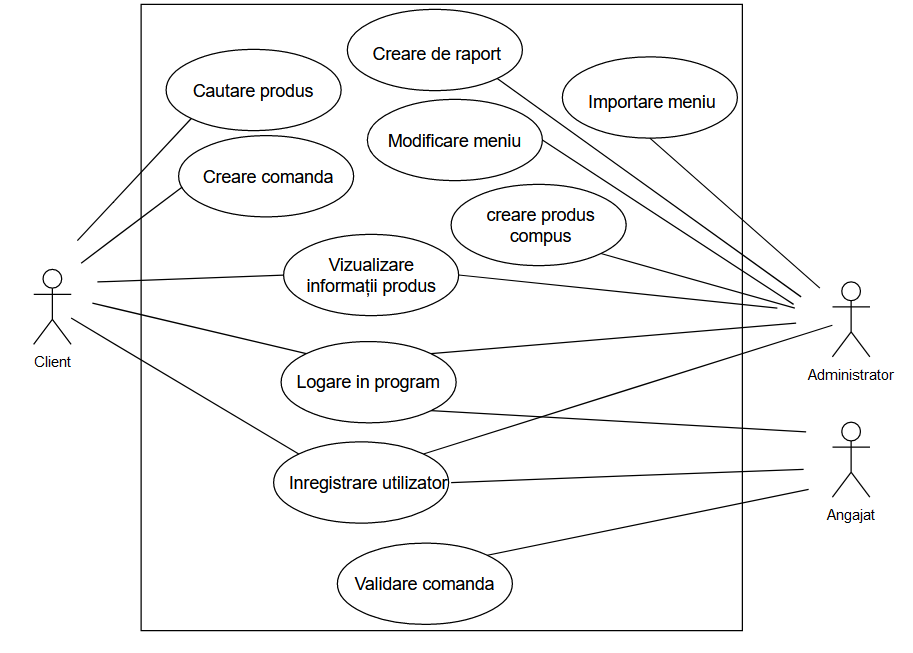
**1.Obiectivul temei:**

Obiectivul principal al acestei teme constă în elaborarea unui program de management pentru o companie de catering. Prin aceasta clienții pot da comenzi de alimente sau meniuri, administratorii pot modifica meniul obține rapoarte si angajații pot procesa comenzile clienților.

Obiective secundare:

1. Analiza problemei și a algoritmilor necesari de implementat. Se analizează cerințele programului și se notează funcționalitățile acestuia si interacțiunea cu utilizatorul (2. Analiza problemei)
2. Crearea unei diagrame UML pentru elaborarea unei implementări pentru clase si a interacțiunii dintre acestea. Desenarea claselor împreună cu relațiile dintre acestea permit la o înțelegere mai bună a structurării programului și a logicii din acesta. (3. Proiectare)
3. Creearea unei clase ce va pemite deserealizarea și serializarea datelor din program într-un fișier text. Aceasta va fi folosiă pentru păstrarea meniului și a comenzilor.(4. Implementare)
4. Creearea unor clase pentru reprezentarea modelelor de date. Acestea vor fi făcute în concordanță cu tipul de produse ce se regăsesc în meniu sau cu comenzile date de clienți. (4.Implementare)
5. Creearea unor clase prin care orice tip de utilizator pot să interacționeze cu datele serializate. Acest tip de clase se fac pentru clienți, angajați și administratori (4. Implementare)
6. Creearea unei clase și interfețe ce va conține metode pentru manipularea datelor serializate. Aceasta va conține metode pentru adaugarea, modificarea și ștergerea de produse din meniu, de adăugare de comenzi noi și de preluare a informației.(4.Implementare)

**2.Analiza problemei:**



Interacțiunea cu programul se face prin GUI. Inițial utilizatorul poate să se logheze sau să se înregistreze. Dacă utilizatorul introduce informație ce nu se găsește în niciunul dintre fișierele cu clienți, angajați sau administratori, atunci se va dechide o fereastra pop-up. Utilizatorul se poate înregistra, introducând un username, o parolă și alegând ce tip de utilizator este. Un client poate vizualiza produse din meniu, poate să caute în funcție de un keyword sau de un atribut, poate să adauge și să șteargă produse la comandă și poate să aplice o comandă. Administratorul poate să vizualizeze produsele din meniu, să creeze, modifice și șterge produsele din meniu, să importeze un mediu și să creeze un raport pentru comenzile existente. Un angajat poate să valideze comenzile puse de clienți.

Use Case: Înregistrare

Actor principal: Client, Angajat, Administrator

Scenariu in caz de success:

1. Utilizatorul introduce un username și o parolă
2. Utilizatorul bifează unul din butoanele ce îi desemnează rolul
3. Se apasă butonul „Save”.
4. Se închide fereastra.

Scenariu in caz de eșec:

1. Utilizatorul nu introduce un nume si prenume la username sau nu bifează unul dintre radio-buttons,
2. Fereastra rămâne activă.

Use Case: Logare

Actor principal: Client, Angajat, Administrator

Scenariu în caz de success:

1. Utilizatorul introduce un username și o parola.
2. Utilizatorul apasă butonul “Accept”.
3. Se deschide una din ferestrele corespunzătoare tipului de utilizator și se închide fereastra de logare.

Scenariu in caz de eșec:

1. Utilizatorul introduce un username sau o parolă ce nu se potrivește cu niciunul din utilizatorii înregistrați.
2. Se deschide o fereastră PopUp cu un mesaj corespunzător.

Use Case: Vizualizare informații produs

Actor principal: Client, Administrator

Scenariu în caz de success:

1. Se alege un produs din meniul drop-down.
2. Sub acesta în spațiul gol vor apărea informațiile legate de acel produs.

Scenariu în caz de eșec:

1. Nu se selectează niciun produs
2. Spațiul cu informații rămâne gol.

Use Case: Cautare produs

Actor principal: Clienți

Scenariu în caz de success:

1. Se deschide o nouă fereastră de căutare
2. Se scrie un keyword și/sau se alege un atribut din meniul drop-down
3. Se deschide un JTable cu produse ce conțin keyword-ul și care sunt ordonate în mod crescător în funcție de atributul ales

Use Case: Creare comandă

Actor principal: Clienți

Scenariu în caz de success:

1. Se adaugă un produs la comandă, conținutul comenzii fiind arătate în caseta text; se pot șterge produse din comandă
2. Se apasă pe butonul ”Place Order”
3. Se plasează comanda, se generează chitanța și se șterge conținutul din caseta text pentru conținutul comenzii

Scenariu în caz de eșec:

1. Nu se adaugă niciun produs în comandă
2. Se deschide o fereastră PopUp cu un mesaj corespunzător

Use Case: Validare comandă

Actor principal: Angajat

Scenariu în caz de success:

1. Angajatul este atenționat de noua comandă
2. Angajatul procesează comanda

Use Case: Modificare meniu

Actor principal: Administrator

Scenariu în caz de success:

1. Se scriu informații în casetele text
2. Se apasă butonul ”Add product” pentru adăugarea în meniu a unui nou produs cu informațiile din casetele text
3. Se apasă butonul ”Modify product” pentru modificarea produsului desemnat de meniul Drop-down cu informația din casetele text
4. Se apasă butonul ”Delete product” pentru a șterge produsul selectat din meniul Drop-down
5. Meniul drop-down revine la mesajul ”Select product”.

Scenariu în caz de eșec:

1. Nu s-a scris informație potrivită în casetele text sau meniul Drop-down are selectat varianta ”Select product”
2. Meniul nu revine la varianta ”Select product” și informația care nu este corectă rămâne neștearsă în casetele text

Use Case: Importare meniu

Actor principal: Administrator

Scenariu în caz de success:

1. Se apasă pe butonul ”Import products”
2. Informația din fișierul ”product.csv” este folosită pentru popularea meniului cu profuele care nu sunt prezente

Scenariu în caz de eșec:

1. Fișierul ”product.csv” nu există
2. Meniul rămâne gol

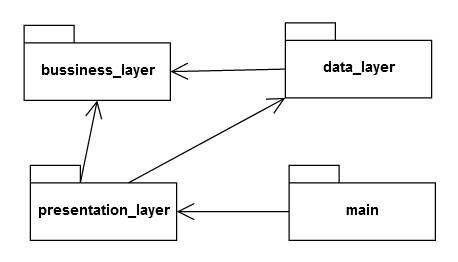
Use Case: Generare raport

Actor principal: Administrator

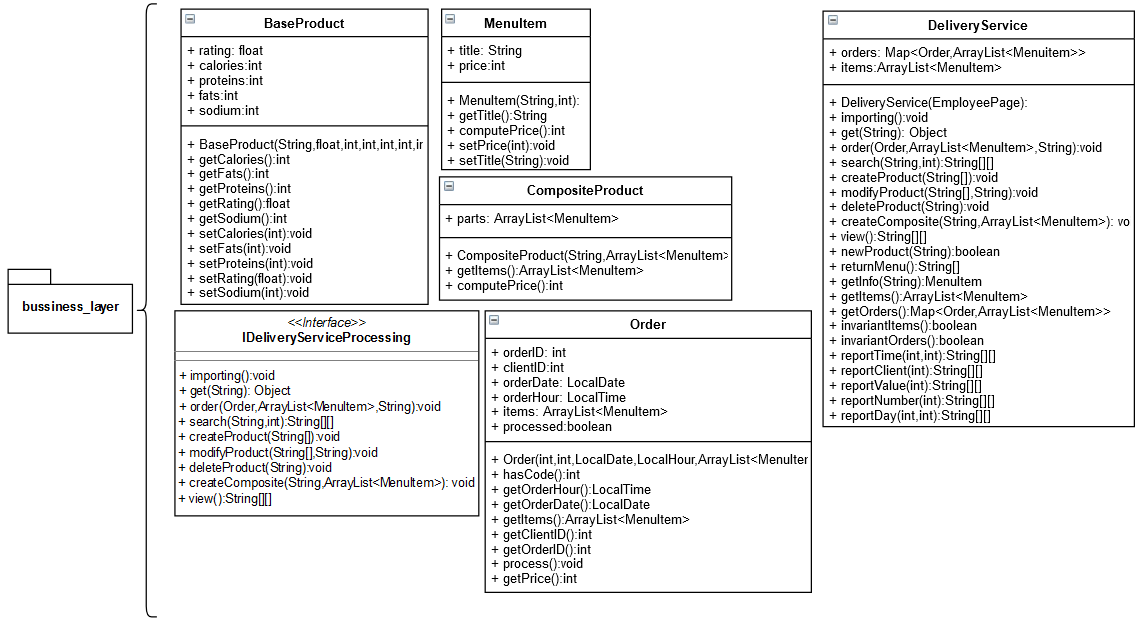
Scenariu în caz de success:

1. Se apasă butonul ”Report”
2. Se generează un nou fișier cu informația relevantă pentru comenzile existente

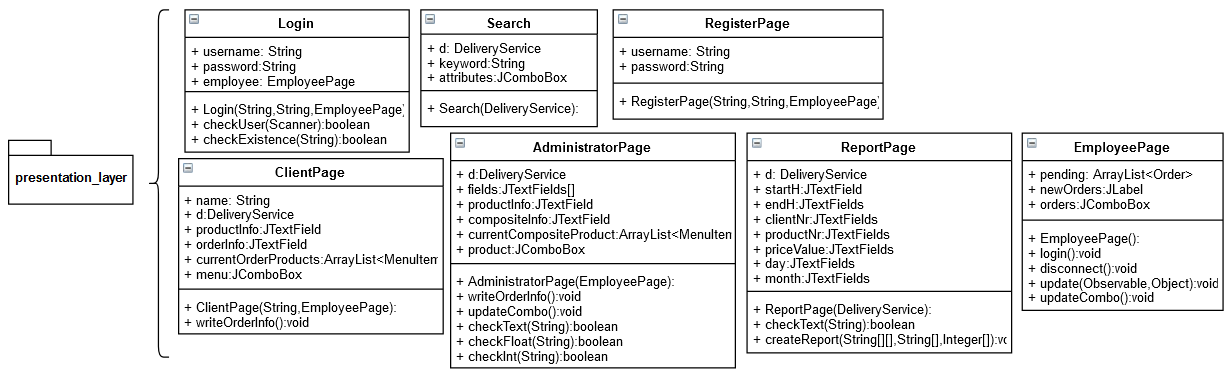
**3.Proiectare:**



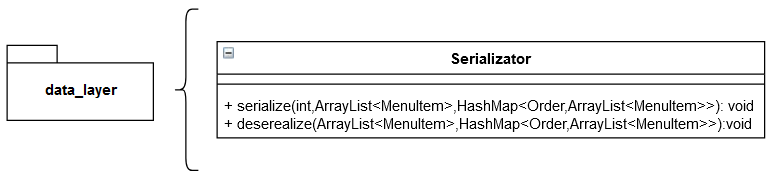
Programul este format din 4 pachete. Pachetul main conține clasa din care se pornește programul în sine. Pachetul business\_layer conține clase cu modelele de date folosite în cadrul programului precum MenuItem, dar și clase folosite pentru stocarea și manipularea datelor precum DeliveryService. Pachetul data\_layer conține clasa Serializator prin care se realizează serializarea și deserializarea datelor. Pachetul presentation\_layer conține clasele pentru GUI.



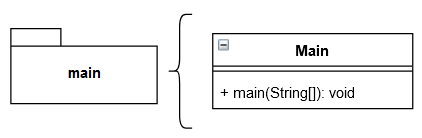
Pachetul business\_layer conține toate clasele folosite pentru manipularea modelelor de date din program. Clasa Order este folosită pentru a reprezenta comenziile clienților. DeliveryService extinde interfața IdeliveryServiceProcessing, aceasta implementând metode de modificare și creeare a produselor din meniu și a comenzilor create de clienți.Clasa MenuItem este folosită pentru reprezentare a produselor din meniu. BaseProduct extinde MenuItem și este folosit pentru a arăta un produs simplu din meniu. CompositeProduct extinde Menu și are rolul de-a reprezenta un produs din meniu format din mai multe produse.



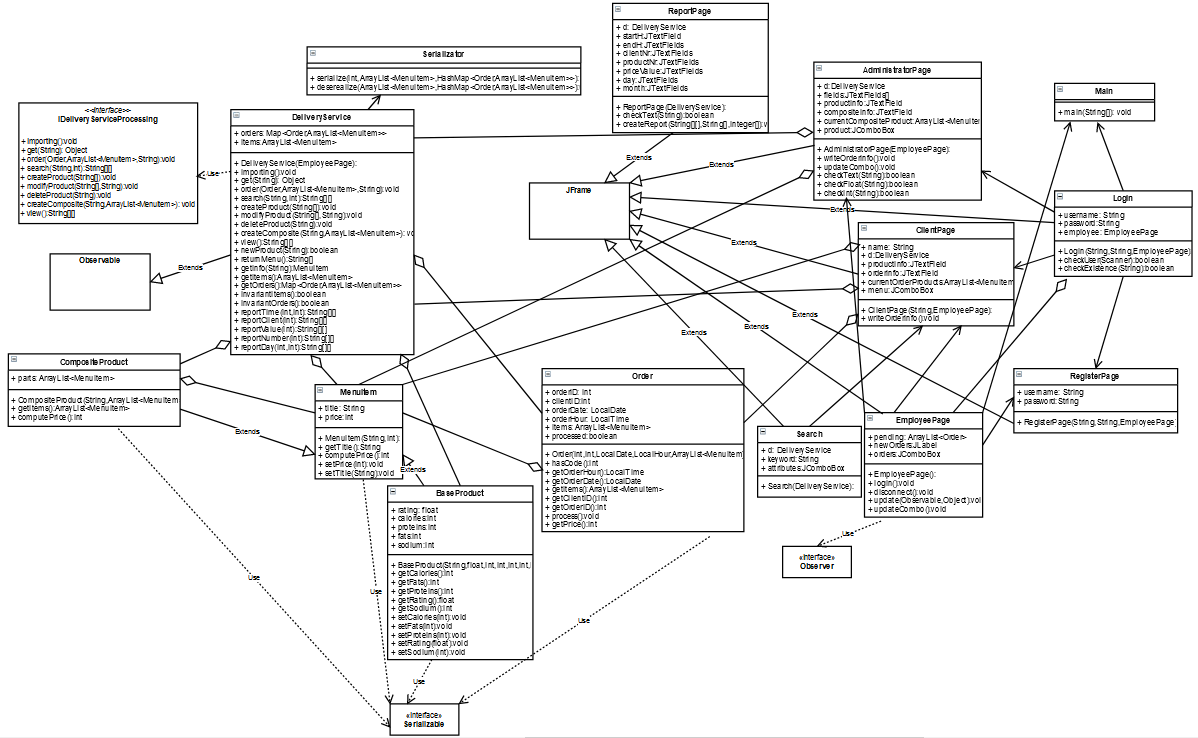
Pachetul presentation\_layer conține clasele ce formează GUI-ul programului. Clasa Login creează fereastra prin care utilizatorul se poate loga. Clasa RegisterPage creează fereastra prin care un utilizator se poate înregistra ca unul dintre cele 3 roluri: client, angajat sau administrator. Clasa ClientPage este folosită pentru a creea fereastra prin care clienții își pot realiza activitățile. Clasa AdministratorPage creează o fereastră prin care a administratorii își pot realiza activitățiile pe datele folosite de program. Clasa Search creează o fereastră folosită pentru căutarea de anumite produse pe baza atributelor sau a unui keyword. Clasa EmployeePage reprezintă fereastra prin care angajatul poate pregătii comenzile introduse. Clasa ReportPage reprezintă o fereastră prin care adminsitratorul poate creea rapoarte pe baza anumitor criterii.



Pachetul data\_layer conține clasa Serializator prin care informația stocată legată de comenzi și produse din meniu este serializată în urma modificărilor și deserealizată la deschiderea ferestrelor ce se folosesc de acestea.



Pachetul main conține clasa Main de la care se pornește funcționarea programului.

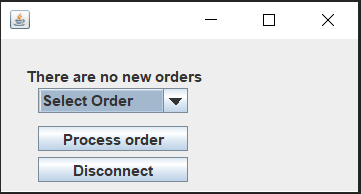


Din diagrama UML se observă toate interacțiunile diverse între clasele programului. Clasele CompositeProduct și BaseProduct extind clasa MenuItem, CompositeProduct agregând deasemenea pe MenuItem. Clasa Order în sine agreghează clasa MenuItem. Toate aceste clase în sine implementează interfața Serializable deoarece acestea reprezintă date stocate prin serializare și sunt agregate în clasa DeliveryService. DeliveryService extinde clasa Observable pentru a notifica obiectele Observer și implementează interfața IdeliveryServiceProcessing ce conține metode de manipulare a datelor, adică a obiectelor din clasele MenuItem și Order. DeliveryService se folosește de metodele statice ale clasei Serializator pentru serializarea și deserealizarea datelor din aceasta. Clasele Login, Search, RegisterPage, EmployeePage, AdministratorPage.ReportPage și ClientPage extind clasa JFrame, acestea având rolul de-a creea GUI-ul programului. EmployeePage implementează interfața Observer pentru a putea fi notificat de către clasa DeliveryService.

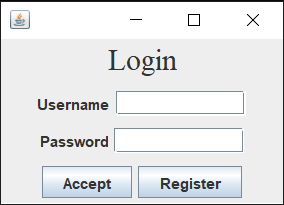
**4.Implementare:**

Clasa Main conține o singură metodă prin care se creează o fereastră reprezentată de clasa Login și un obiect EmployeePage. Aceasta reprezintă locul de unde se pornește funcționarea programului.

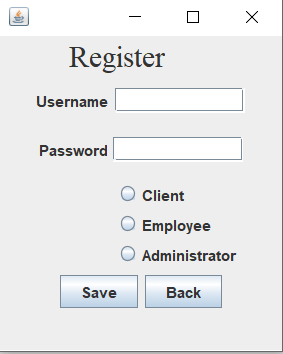
Clasa EmployeePage extinde clasa Jframe și implemenentează interfața Observer. Această clasă reprezintă fereastra prin care un angajat va putea pregătii comenzile depuse de clienți. Are ca și atribute un ArrayList pending de obiecte Order ce reprezintă comenzile ce așteaptă a fi procesate, un obiect JLabel newOrders și un obiect JComboBox orders. Deoarece este implementată interfața Observer, de fiecare dată când obiectul EmployeePage este notificat de obiectul observable legat, atunci va fi apelată metoda update prin care se va adăuga un obiect Order în lista pending, se va modifica conținutul lui orders și se va modifica mesajul lui newOrders. Apăsarea butonului ”Disconnect” va face ca fereastra să se închidă și să se deschidă o fereastră Login. Dacă se apasă butonul ”Process” și este selectat o comandă în JComboBox, atunci obiectul Order respectiv va apela metoda process, se va șterge obiectul din listă și se va modifica conținutul lui orders și newOrders. Metoda login va creea o nouă fereastră Login și va închide fereastra curentă.Un obiect din această clasă este creat la începutul programului și este transmis ca și parametru în constructorii tuturor claselor din pachetul presentationLayer, înafară de Search. Acest lucru este folosit pentru a păstra constantă interacțiunea cu DeliveryService care este obiectul Observable. Fereastra creată de clasă este făcută vizibilă doar atunci când se loghează un angajat.



Clasa Login extinde clasa JFrame, această clasă reprezentând fereastra prin care un utilizator se poate loga în aplicație. Clasa are ca și atribute două instanțe de JTextField username și password în care se vor scrie numele, respectiv parola utilizatorului. Când se va apăsa butonul ”Accept” se va verifica conținutul celor două casete text și se va compara cu informațiile stocate în fișierele ”client.txt”, ”employee.txt” sau ”administrator.txt”. Dacă se găsesc informații ce coincid cu ceea ce este într-unul dintre fișiere, atunci se va dechide o nouă fereastră corespunzătoare utilizatorului găsit, altfel se va deschide o fereastră PopUp cu un mesaj corespunzător. Pentru înregistrare se va apăsa butonul ”Register” ce va creea o nouă instanță a clasei RegisterPage. Metoda checkExistence va căuta în fișierul desemnat de parametrul filename dacă este vreun utilizator ai căror date coincid cu ceea ce este scris în username și password. Această metodă se folosește de metoda checkUser ce returnează true dacă s-a găsit utilizatorul și false în caz contrar.



Clasa RegisterPage este folosită pentru creearea unei ferestre prin care se poate înregistra un nou utilizator. La fel ca și Login, această clasă are două atribute de tip JTextField username și password. Un utilizator poate să-si scrie numele, ce trebuie să fie format din nume și prenume, parola și să bifeze una dintre variantele ce îi desemnează rolul în cadrul programului. Pentru înregistrare se apasă butonul ”Register”, se verifică corectitudinea numelui și butonul bifat, se scrie într-unul din fișierele corespunzătoare informația din casetele text și se deschide o nouă fereastră de logare. Butonul ”Back” creează o nouă fereastră de logare și o închide pe cea de înregistrare fără să creeze un nou utilizator.



Clasa MenuItem reprezintă un produs oarecare din meniul companiei. Acesta are ca atribute un String title ce reprezintă numele produsului și un int price ce reprezintă prețul produsului. Înafară de operații de tip getter și setter, clasa are o metodă computePrice prin care se returnează prețul produsului.

Clasa BaseProduct extinde clasa MenuItem și reprezintă un produs simplu. Acesta prezintă pe lângă atributele moștenite de la clasa MenuItem un float ce reprezintă recenzia produsului și mai multe obiecte de tip int ce reprezintă diferite aspecte alte produsului: calorii, sodiu, proteine și grăsime. Această clasă prezintă numai metode de tip setter și getter. De asemenea, această clasă implementează interfața Serializable.

Clasa CompositeProduct extinde clasa MenuItem și reprezintă un grup de produse ce pot fi cumpărate împreună ca și un meniu.Această clasă are ca și atribute un obiect parts de tip ArrayList de obiecte din clasa MenuItem ce reprezintă produsele ce fac parte din meniu. Constructorul clasei primește o listă de obiecte MenuItem ai căror elemente vor fi adăugate pe rând în atributul parts. Ca și metode clasa prezintă o metoda getItems ce returnează un String format din titlurile fiecărui MenuItem din parts urmat de suma prețurilor acestor produse. Metoda computePrice este Overriden unde, dacă este apelată pentru prima dată, atunci se face suma dintre prețurile obiectelor componente din lista parts și după se returnează suma respectivă ce reprezintă prețul total al meniului. De asemenea, această clasă implementează interfața Serializable. Această clasă, împreună cu MenuItem și BaseProduct urmează modelarea de tip Composite Design Pattern.

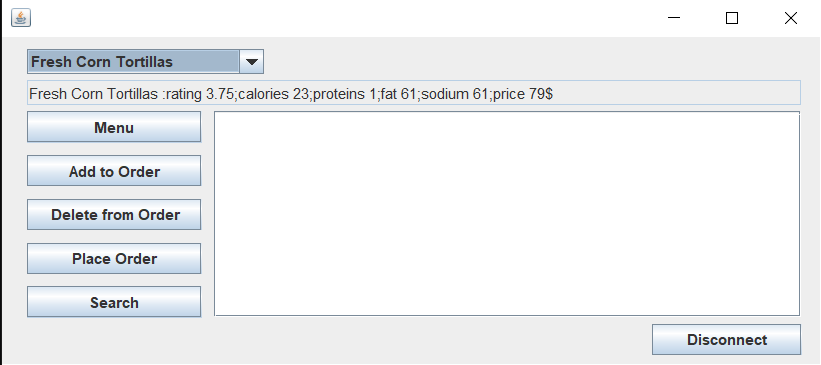
Clasa Order este folosită pentru a reprezenta o comandă făcută de un client. Clasa are ca și atribute două obiecte de tip int cID și oID ce reprezintă ID-urile clientului ce a pus comanda și ID-ul comenzii în sine, adică poziția acestora în fișierul serializat; un obiect din clasa LocalDate ce reprezintă data depunerii comenzii; un obiect din clasa LocalTime ce reprezintă ora depunerii comenzii, un obiect boolean processed ce arată dacă un angajat a validat comanda sau nu și un obiect din ArrayList de obiecte din clasa MenuItem ce reprezintă produsele din comanda respectivă. Pe lângă metode de tip getter, clasa dă Override la metoda HashCode a clasei Object pentru returna o valoare formată pe baza unei operații ce implică atributele clasei, metoda process face ca valoarea atributului processed să devină true și metoda getPrice realizează suma tuturor prețurilor produselor din items și returnează rezultatul respectiv. Această clasă implementează interfața Serializable.

Interfața IDeliveryServiceProcessing este folosită pentru a oferii metode prin care se pot manipula produsele și comenzile din cadrul companiei de catering.

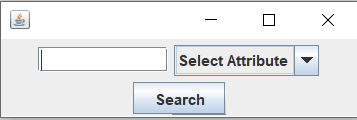
Clasa DeliveryService implementează interfața IDeliveryServiceProcessing și reprezintă serviciul de livrare al companiei. Aceasta are ca și atribute un obiect de tip ArrayList items de obiecte de tip MenuItem și un obiect orders de tip Map ce reprezintă comenzile date de clienți și care folosește obiecte de clasa Order ca și cheie și o listă de obiecte de clasă MenuItem ce reprezintă valoarea. În constructorul clasei valorile sunt adăugate în listă și Map prin intermediul deserializării. Clasa prezintă metodele primite de la interfața IDeliveryServiceProcessing și metode private ce sunt folosite ca și suport pentru metodele de la interfață. Metoda importing este folosită pentru a umple lista de produse cu informații stocate în fișierul ”product.csv”: se citește din fișier câte un rând, se verifică dacă titlul produsului se regăsește deja în meniu și, în caz contrar, se adaugă în listă un nou obiect de tip BaseProduct ce se formează cu informațiile din rândul preluat. Metoda get primește ca și parametru un String s și va căuta un obiect din lista items pe care îl va returna dacă are atributul title identic cu s. Metoda createProduct primește ca și parametru un array de String-uri și va forma un nou obiect BaseProduct ce îl va adăuga în lista items. Metoda modifyProduct primește ca și parametrii un array de String-uri fields și un String title și pentru produsul din items ce are titlul identic cu title, atunci pentru fiecare String din fields care nu este gol se va modifica atributul corespunzător din produsul desemnat. Metoda deleteProduct primește ca și parametru un String s și va căuta produsul cu titlul identic cu s pe care îl va șterge din lista items. Metoda search primește un String s și un obiect de tip int index. Această metodă va căuta inițial toate produsele din lista items ce au în titlu String-ul s ce se vor stoca într-un array de BaseProduct și după, din array-ul de string-uri rezultat, se vor ordona elementele în funcție de atributul desemnat de parametrul index. Din array-ul rezultat se va transmite informația de la fiecare produs într-un 2D array ce va fi returnat. Metoda view va creea și returna un 2D array ce va conține informația fiecărui obiect din lista items. Metoda createComposite va primii ca și parametrii un String title și un ArrayList products cu de obiecte MenuItem. Această metodă va adăuga în items un nou obiect CompositeProduct cu informația furnizată de parametrii. Metoda order va primii ca și parametrii un obiect Order m și un String user. Această metodă va creea un nou fișier text în care vor fi scrise toate informațiile relevante comenzii reprezentate de m și care va adăuga în orders obiectul m. Dacă lista de produse a comenzii este goală atunci se va afișa o fereastră PopUp cu un mesaj corespunzător. Metoda returnMenu va returna un array de String-uri ce va conține String-uri relevante pentru obiectele JComboBox din clasele pentru GUI. Metoda newProduct primește ca și parametru un String s și va returna true dacă este un obiect din lista items ce are titlul egal cu s și false în caz contrar. Metodele getItems și getOrders vor returna atributele clasei. Metodele invariantItems și invariantOrders au rolul de invariante, acestea verificând la începutul fiecărei metode corespunzătoare dacă atributele items și orders nu sunt nule. Metodele reportTime, reportClient, reportDay, reportNumber si reportValue funcționează într-un mod foarte similar: toate returnează un 2D Array de obiecte String ce conțin informații preluate din HashMap-ul orders folosind Java Streams. reportTime va selecta comenzile depuse între orele transmise ca și parametrii, reportClient va selecta clienți care au depus mai multe comenzi decât un număr transmis ca parametru, reportDay preia toate produsele comandate în ziua și luna trimise ca și parametrii, reportValue preia comenzile al căror preț este mai mare ca și valoarea trimisă ca și parametru și reportNumber preia produsele ceau fost comandate mai mult decât un număr trimis ca și parametru.

Clasa Serializator conține metode statice pentru serializarea și deserializarea datelor din DeliveryService. Metoda serialize va primii obiectul ArrayList și obiectul Map din clasa DeliveryService și va scrie în fișierul ”file.txt” conținutul acestora. Metoda deserialize primește aceeași parametrii în care vor fi scrise datele serializate în fișierul ”file.txt”.

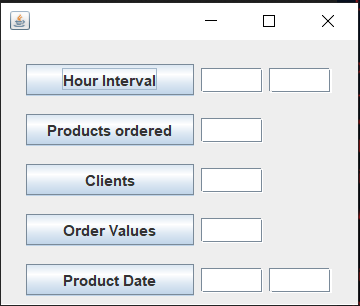
Clasa ClientPage este folosită pentru creearea ferestrei pentru Clienți. Aceasta are ca și atribute un String name, un DeliveryService d, o listă currentOrderProducts de obiecte de tip MenuItem, un JComboBox menu, un JTextField productInfo și un JtextArea orderInfo.În fereastră clientul poate selecta un produs dintr-un drop-down menu, informația produsului fiind arătată în productInfo, poate deschide un JTable cu toate produsele din meniu apăsând butonul ”View”, poate apăsa butonul ”Search” pentru a deschide o fereastră de tip Search și poate revenii la fereastra de logare apăsând pe butonul ”Disconnect”. Dacă apasă pe butonul ”Add to Order”, atunci produsul selectat din JcomboBox se va adăuga la comanda curentă. Prin apăsarea butonului ”Place Order” , se creează un nou obiect de tip Order ce va avea produsele din lista currentOrderProducts, cID care este poziția clientului din fișierul ”client.txt”, oID care reprezintă numărul total de comenzi plus 1 și ora și data curentă și se va apela metoda order din atributul d. Informația legată de produsele din comandă și prețul acestora este scris în orderInfo cu ajutorul metodei updateOrderInfo.



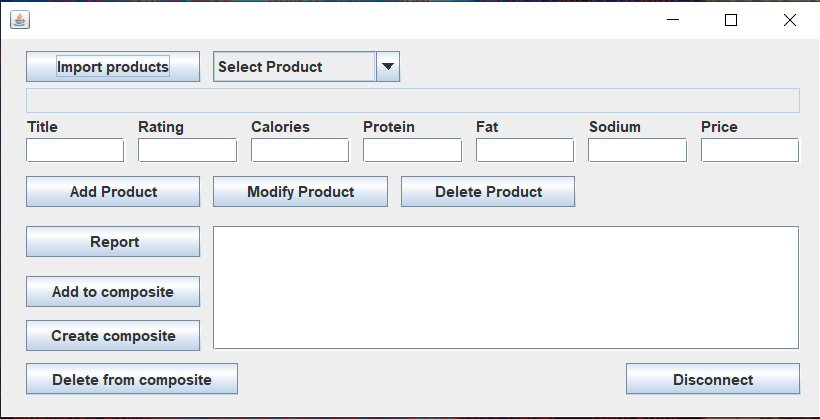
Clasa Search este folosită pentru căutarea și ordonarea produselor din meniu pe baza unui keyword și a unui atribut selectat dintr-un JComboBox. Clasa are ca și atribute un obiect DeliveryService d, un JTextField keyword și un JComboBox attributes. Prin apăsarea butonului ”Search” se va deschide un JTable ce va conține toate produsele ce au textul din caseta keyword în titlu și eventual ordonate crescător în funcție de atributul selectat.



Clasa ReportPage are rolul de-a creea o fereastră prin care administratorul poate creea diferite rapoarte urmând anumite reguli. Clasa are ca și atribute un obiect DeliveryService d și mai multe obiecte JTextField: startH, endH, clientNr, productNr, priceValue, day, month. Fiecare buton are lângă acesta una sau două căsuțe text în care vor fi scrise valori pe baza cărora se preia informație din obiectul DeliveryService. Butonul ”Hour Interval” va deschide o fereastră nouă cu un Jtable ce va conține toate comenzile depuse între două ore scrise în căsuțele text de lângă buton. Butonul ”Product Ordered” creează o nouă fereastră cu un JTable ce va conține produse ce au fost comandate mai mult decât o valoare notată în căsuța text. Butonul ”Clients” creează o nouă fereastră cu un JTable ce va conține numele clienților ce au comandat mai mult decât o valoare notată în căsuța text. Butonul ”Order Values” creează o nouă fereastră cu un JTable ce va conține coemnzi ce au un preț mai mare decât o valoare notată în căsuța text. Butonul ”Product Date” creează o nouă fereastră cu un JTable ce va conține produse ce au fost comandate în ziua și luna notate în căsuțele text.



Clasa AdministratorPage este folosit pentru creearea unei ferestre prin care administratorul poate să realizeze operații de modificare a meniului companiei. Are ca și atribute un obiect DeliveryService d, un array de JTextField fields, un JTextField productInfo, un JtextArea compositeInfo, un JComboBox product și un ArrayList currentCompositeProduct de obiecte MenuItem. Butonul ”Import products” va apela metoda import din d pentru popularea meniului cu obiecte ce folosesc informația din ”product.csv”. Pentru produsele selectate din JComboBox informația acestora va fi prezentată în productInfo. În casetele text ale lui fields se va putea scrie informație pentru crearea de produse noi, sau pentru modificarea de produse existente. Prin apăsarea butonului ”Add Product” se va creea un nou obiect BaseProduct pentru meniu dacă datele din casetele fields sunt corecte. Dacă butonul ”Modify Product” este apăsat, atunci informația pentru produsul selectat din JComboBox se va modifica cu informațiile din casetele fields ce au fost scrise și care sunt corecte. Butonul ”Delete Product” va șterge din meniu produsul selectat din JComboBox. Funcționarea corectă a acestor butoane se observă dacă informația din casetele fields a fost înlăturată și dacă product a revenit la opțiunea ”Select product”. Butonul ”Add to composite” va adăuga produsul selectat din product al currentCompositeProduct. Butonul ”Create composite” va creea un nou obiect CompositeProduct ce va avea titlul textul din fields[0] și ca și produse lista currentCompositeProduct. Dacă nu este nimic scris în fields[0], atunci se va afișa o fereastră PopUp cu un mesaj potrivit. Acesta va fi adăugat în d folosind metoda createComposite. Informația legată de currentCompositeProduct este scrisă în compositeInfo prin metoda writeCompositeInfo. Metoda updateCombo va actualiza JComboBox cu cel mai recent meniu și va serializa informația din d. Metodele checkText, checkFloat și checkInt vor primii fiecare câte un String și vor verifica corectitudinea acestuia. Butonul ”Disconnect” va duce la revenirea la fereastra de logare. Butonul ”Report” va creea un obiect ReportPage ce va deschide o fereastră prin care administratorul va genera rapoarte.



**5.Concluzii:**

Pentru realizarea acestei teme au fost necesare cunoștiințe legate de serializarea datelor, de folosire a Java Streams și de înțelegere a modului de lucru cu Java HashMaps.

Această temă a fost folositoare deoarece a ajutat la înțelegera conceptului de serializare și deserealizare, dar și pentru înțelegerea modului de lucru cu HashMaps, structuri de date pe care nu s-a pus foarte tare accentul la materia Programare Orientată pe Obiect.

Îmbunătățiri posibile la această ar fi eliminare a repetiției din cod și elaborarea unor algoritmi mai scurți pentru realizarea anumitor sarcini, precum creearea de comenzi.

**6.Bibliografie:**

Lambda expressions and stream processing:

<https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html>

Java serialization:

<https://www.baeldung.com/java-serialization>

Java HashMap:

<http://javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.html>

Java assert:

<http://stackoverflow.com/questions/11415160/how-to-enable-the-java-keyword-assert-in-eclipse-program-wise>