UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Tehnici de programare fundamentale
Tema 1

Catering Management

- Documentație -

Realizat de:
Ducai David Alex – grupa 30223

Cuprins

1. Obiectivele temei

Obiectivul principal al temei este realizarea unei aplicații funcționale de management al unei firme de catering. Această aplicație trebuie să aibă o interfață grafică intuitivă pentru ficare tip de utilizator (administator, client oarecare, angajat).

Obiectivele secundare ale temei sunt următoarele:

- Analizarea problemei și identificarea resurselor necesare realizării temei propuse – *capitolul 2*
- Realizarea design-ului aplicației – *capitolele 3 și 4*
- Implementarea aplicației – *capitolul 4*
- Testarea aplicației – *capitolul 5*

2. Analiza problemei

2.1. Cerințe funcționale

Aplicația trebuie să-i permită utilizatorului obișnuit (clientul) să se înregistreze cu un nume de utilizator și o parolă, să se conecteze cu acestea, să adauge produse și meniuri în coșul de cumpărături, să le caute în lista de produse disponibile și să plaseze comanda la final.

Aplicația trebuie să-i permită administratorului să adauge produse noi, să creeze noi meniuri pe baza produselor existente, să modifice (editeze/șteargă) produse existente, să importeze date externe și să genereze rapoarte pe baza comenzilor existente.

Aplicația trebuie să-i afișeze în timp real angajatului notificări pentru comenzi noi.

2.2. Cerințe non-funcționale

Aplicația trebuie să vină însoțită de o interfață intuitivă, ușor de folosit de către utilizator, care să ofere informații în caz de nevoie și să indice eventualele erori de utilizare a acesteia. De asemenea, aplicația trebuie să păstreze modificările aduse asupra meniurilor și comenzilor.

2.3 Cazuri de utilizare

1. ***Caz de utilizare**: adăugarea produselor de bază (actor: administratorul)*

Scenariu principal (operație realizată cu succes):

1. Administratorul se loghează în caseta de login cu credențialele sale
2. Administratorul apasă pe butonul pentru adăugarea unui produs nou
3. Administratorul introduce în câmpurile disponibile informațiile necesare și apasă pe butonul pentru finalizare
4. Administratorul poate vedea în lista produselor produsul adăugat

Scenariu secundar (Câmpuri necompletate):

- Administratorul nu a completat toate câmpurile disponibile
- Se deschide o nouă fereastră de atenționare
- Se revine în scenariul principal la pasul 3.

Scenariu secundar (Valori incorecte):

- Administratorul a completat greșit câmpurile disponibile
- Se deschide o nouă fereastră de atenționare
- Se revine în scenariul principal la pasul 3.

2. ***Caz de utilizare**: modificarea unui produs existent*

Scenariu principal (operație realizată cu succes):

1. Administratorul se loghează în caseta de login cu credențialele sale
2. Administratorul selectează un produs din listă și apasă pe butonul pentru modificarea unui produs
3. Administratorul modifică valorile din câmpurile disponibile și apasă pe butonul pentru finalizare
4. Administratorul poate vedea în lista produselor produsul modificat

Scenariile secundare pentru acest caz de utilizare sunt identice cu cele alea cazului 1.

3. ***Caz de utilizare**: ștergerea unui produs existent*

Scenariu principal:

1. Administratorul se loghează în caseta de login cu credențialele sale
2. Administratorul selectează un produs din listă și apasă pe butonul pentru ștergerea produsului
3. Administratorul poate vedea rezultatul în lista produselor

4. ***Caz de utilizare**: adăugarea unui produs compus*

Scenariu principal:

1. Administratorul se loghează în caseta de login cu credențialele sale
2. Administratorul selectează un produs din listă și apasă pe butonul pentru adăugarea lui într-un meniu nou

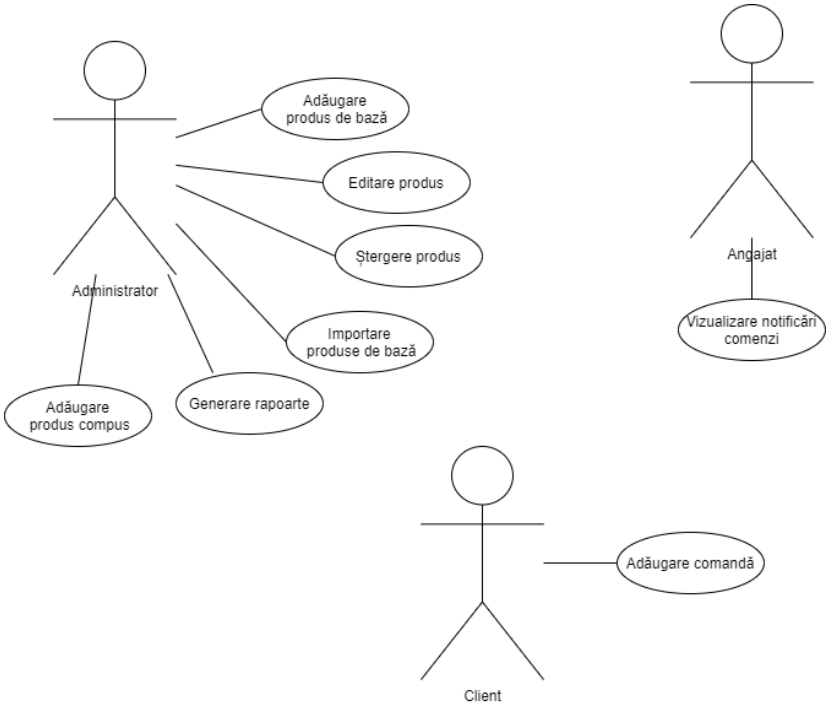
3. Administratorul poate repeta pasul 2. de câte ori dorește
4. Administratorul apasă pe butonul de finalizare
5. Administratorul introduce în câmpurile disponibile informațiile necesare și apasă pe butonul pentru adăugare în listă

5. ***Caz de utilizare**: generarea rapoartelor*
Scenariu principal:
1. Administratorul se loghează în caseta de login cu credențialele sale
2. Administratorul apasă butonul care duce la pgina pentru rapoarte
3. Administratorul introduce informații în casetele disponibile pentru raportul dorit
4. Administratorul apasă pe butonul de generare al unui raport
5. Se afișează o fereastră de informare

Scenariile secundare sunt asemănătoare cu cele de la cazul 1. cu excepția faptului că se revine în acest scenariu principal la pasul 3.

6. ***Caz de utilizare**: plasarea unei comenzi (actor: clientul)*
Scenariu principal:
1. Clientul se loghează în caseta de login cu credențialele sale
2. Clientul selectează un produs din listă și apasă pe butonul pentru adăugarea lui în coșul de cumpărături
   Clientul poate repeta pasul 2. de câte ori dorește
3. Clientul apasă pe butonul de trecere la următoarea pagină
4. Clientul poate să șteargă produse din coș prin selectarea acestuia și apăsarea butonului de ștergere.
   Clientul poate repeta pasul 4 până la golirea coșului
5. Clientul apasă pe butonul de finalizare. Comanda este trimisă și se generează bonul
6. Clientul poate apăsa pe butonul Comandă pentru a comanda din nou. În acest caz se revine la pasul 2.

7. ***Caz de utilizare**: Vizualizarea notificărilor*
Scenariu principal:
1. Angajatul se loghează în caseta de login cu credențialele sale
2. Se deschide o nouă fereastră în care se pot urmări notificările în timp real



3. Proiectarea temei

Pentru realizarea acestei aplicații am ales modelul arhitectural Three-Tier Architecture. Acest model arhitectural desparte aplicația în trei etaje / straturi (layere):

- Etajul de prezentare (Presentation Layer) – conține clasele care definesc interfața grafică a aplicației (view, controller, etc.)
- Etajul de logică a afacerii (Business Logic Layer) – conține clasele care încapsulează logica aplicației
- Etajul de acces la date (Data Access Layer) – conține clasele care manipulează datele aplicației

Etajele modelului arhitectural vor fi proiectate în pachetele: presentation, businesslogic și data.

Vom începe cu proiectarea pachetului *businesslogic*. Primele clase implementate vor fi clasele care reprezintă produse din meniul firmei de catering. Produsele din meniu pot fi simple, reprezentând un singul fel de mâncare, sau compuse, care conțin mai multe produse din meniu (ex. meniul zilei, etc.). Clasele care rezultă din această proiectare sunt: *BaseProduct* și *CompositeProduct*. Ambele tipuri de produse vor avea un nume, un rating și un preț calculat. Pentru produsul de bază vom adăuga informații nutriționale ca atribute (calorii, proteine, grăsime, sodiu). Pentru produsul compus, vom avea nevoie în plus doar de o listă cu produsele pe care le conține. Nu are sens să implementăm informații nutriționale pentru întregul produs compus. Această listă trebuie să permită introducerea de obiecte din ambele clase menționate. Pentru realizarea acestui plan, vom utiliza un *Composite Design Pattern*.

Vom defini interfața *MenuItem*, care va conține metode specifice pentru un produs din meniu: returnarea denumirii, ratingului, detaliilor și calculul prețului produsului. Cele două clase menționate mai sus vor implementa această interfață. Prin urmare, *BaseProduct* și *CompositeProduct* devin amândouă și *MenuItem*. Astfel, se vor putea adăuga produse din ambele clase în lista unui obiect *CompositeProduct*.

Mai trebuie definită o clasă pentru datele comenzii: *Order*. Va avea ca atribute: un ID pentru comandă, un ID pentru client (este suficient să considerăm numele de utilizator pentru accesarea aplicației ca ID-ul clientului) și data la care s-a înregistrat comanda. Cele trei clase menționate, împreună cu interfața, vor alcătui subpachetul *model* al acestui etaj arhitectural.

Urmează declararea unei interfețe care va conține metode specifice pentru aplicația firmei de catering. Metodele sunt reprezentate de operațiuni pe care utilizatorii le pot realiza în interfața grafică: adăugare/editare/ștergere de produse, căutarea de produse, generarea de rapoarte, adăugarea unei comenzi, etc.

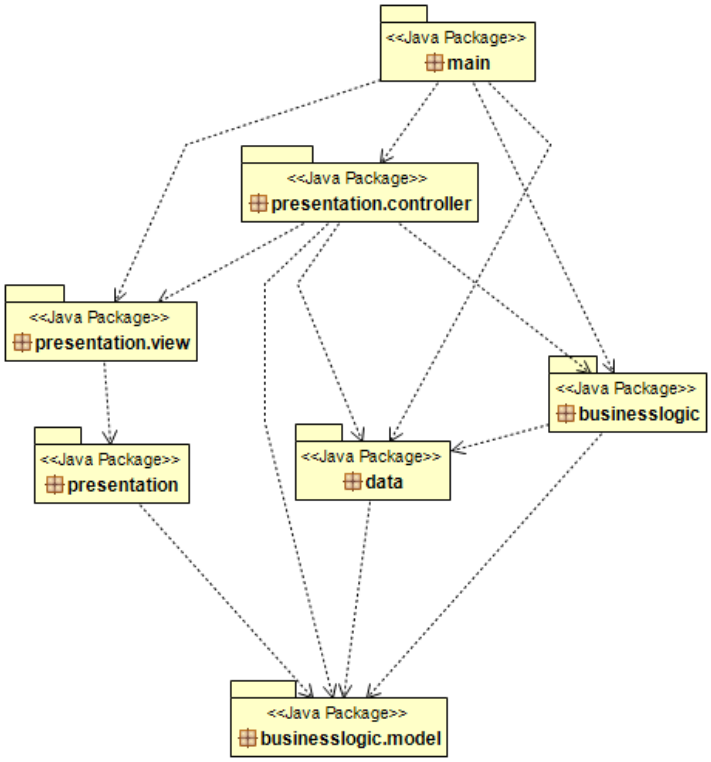
Clasa principală a aplicației este clasa care implementează aceste operațiuni: *DeliveryService*. Această clasa va conține, pe lângă metodele descrise mai sus, lista meniu a firmei de catering, care conține toate produsele (simple și compuse), și o colecție, care reprezintă un istoric al comenzilor. Pentru implementarea listei meniu vom utiliza un *HashSet*. Acest tip de colecție este util, deoarece are proprietăți de care avem nevoie: nu permite adăugarea duplicatelor și este eficient în realizarea operațiilor de căutare. Pentru implementarea istoricului comenzilor vom utiliza un *HashMap*, care va avea ca și cheie un obiect *Order* și va avea ca valoare coșul de cumpărături corespunzător comenzii respective. Utilizăm obiectul *comandă* ca și cheie pentru reducerea coliziunilor (combinația atributelor sale este aproape unică).

Următorul pachet care va fi implementat va fi pachetul *data*. Acest pachet va conține clase care *comunică* cu exteriorul aplicației: generarea de rapoarte, generarea de bonuri, citirea credențialelor pentru logare, serializarea și deserializarea datelor, înregistrarea de clienți noi, etc. Vom proiecta 3 clase fără atribute și doar cu metode statice: *DataWriter*(pentru serilizare/deserializare), *User*(pentru logare), *FileGenerator*(pentru generarea de fișiere).

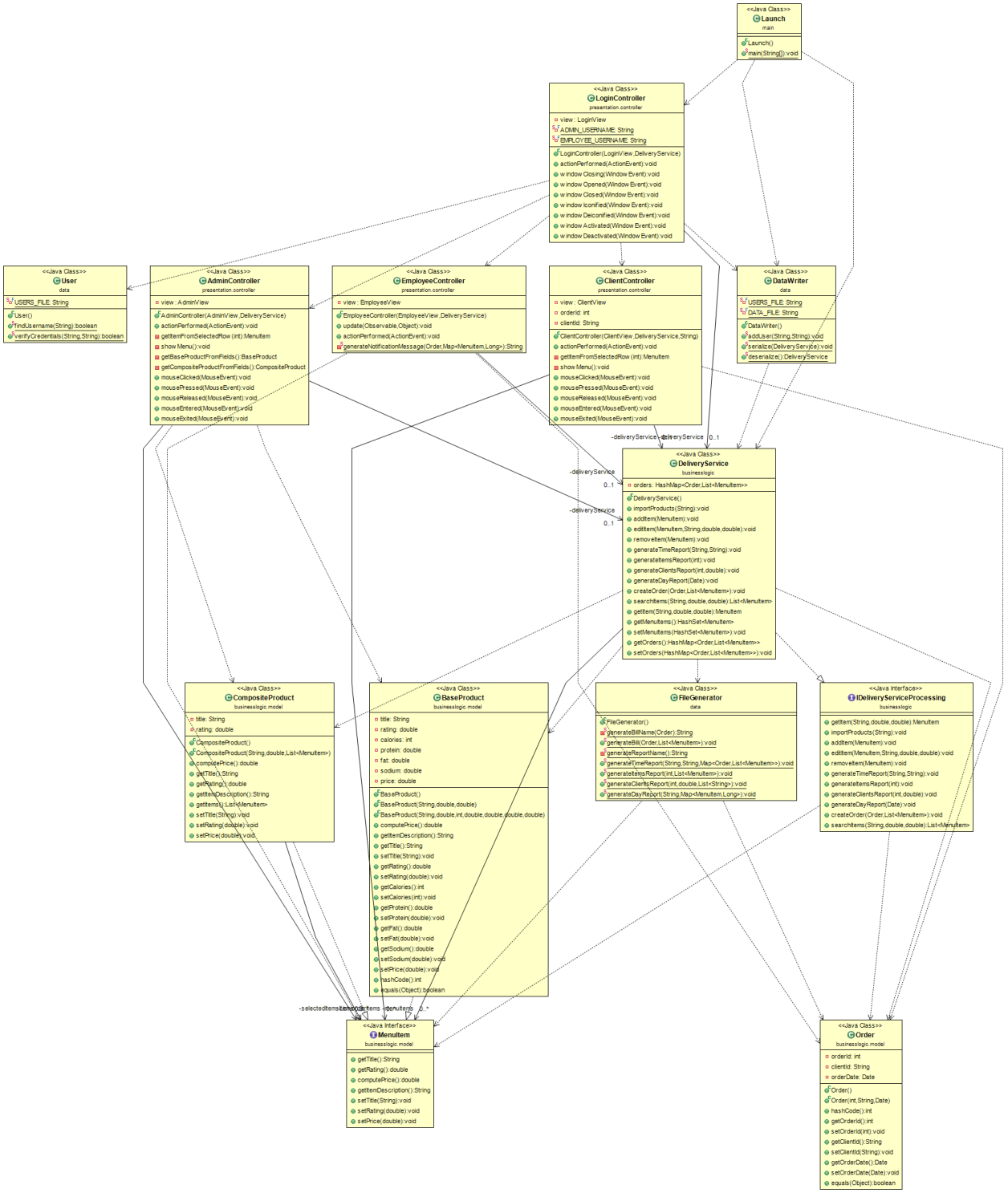
Ultimul pachet implementat va fi cel pentru interfața grafică, *presentation*. Acesta va conține view-urile și controller-ele pentru ferestrele administratorului, clientului, angajatului și autentificării.

Diagrame UML:

- diagrama de pachete



- diagrama de clase



4. Implementarea temei

În acest capitol vor fi descrise toate clasele din aplicație împreună cu atributele și metodele cele mai importante.

4.1. Pachetul *businesslogic*

Subpachetul ***model***

Interfața **MenuItem –** deține metodele pentru produse din lista meniu

Metode

- getTitle() – returnează denumirea
- getRating() – returnează ratingul
- computePrice(Monomial m, Monomial n) – returnează prețul produsului corespunzător astfel (se descriu implementările din clasele *BaseProduct* și *CompositeProduct*):
  - Dacă produsul e de bază, se returnează prețul
  - Dacă produsul e compus, se calculează suma tuturor prețurilor produselor din listă și se aplică o reducere de 15%
- getItemDescription() – returnează detalii despre produs sub formă de String
- metode de *set* pentru titlu, rating și preț

Clasa **BasicProduct –** reține datele unui produs de bază, implementează ***MenuItem***

Atribute

- title – denumire
- rating
- calories – cantitatea de calorii (întreg)
- protein – cantitatea de proteine (double)
- fat – cantitatea de grăsimi (double)
- sodium – cantitatea de sodiu (double)
- price – prețul produsului

Clasa **CompositeProduct –** reține datele unui produs compus, implementează ***MenuItem***

Atribute

- title – denumire
- rating
- items – produsele conținute (*ArrayList de MenuItem*)

Clasa **Order –** reține datele unei comenzi

Atribute

- orderId – ID-ul comenzii – va fi capacitatea incrementată a HashMap-ului care conține istoricul comenzilor
- clientId – ID-ul clientului – va fi numele de utilizator al clientului, care este unic
- orderDate – data la care s-a preluat comanda

Metode

- hashCode() – generează un HashCode pentru introducerea în istoric (metodă suprascrisă)

Restul pachetului

Interfața **IDeliveryServiceProcessing –** definește semnăturile metodelor specifice pentru aplicație

Metode (explicate aici, implementate în clasa DeliveryService, definită mai jos)

- getItem(String title, double rating, double price) – returnează produsul care are ca valori pentru atribute valorile parametrilor metodei
- importProducts() – importă produsele unei liste meniu externă în memoria aplicației
- addItem(MenuItem item) – inserează în lista meniu un produs nou
- editItem(MenuItem item, String title, double rating, double price) – actualizează valorile atributelor produsului *item* cu valorile celorlalți parametrilor (se șterge produsul inițial și se adaugă un produs cu atributele modificate)
- removeItem(MenuItem item) – șterge un produs din lista meniu
- generateTimesReport(String startTime, String endTime) – generează un raport, care conține comenzile care au fost preluate într-un interval orar indiferent de zi

- generateItemsReport(int numberOfTimes) – generează un raport, care conține produsele comandate mai mult de *numberOfTimes* ori.
- generateClientsReport(int numberOfTimes, double value) – generează un raport, care conține clienții care au comandat mai mult de *numberOfTimes* ori și valoarea comenzii este mai mare decât *value*
- generateDayReport(Date date) – generează un raport, care conține produsele comandate în ziua *date* și numărul de bucăți din fiecare
- createOrder(Order order, List<MenuItem> items) – adaugă în istoricul comenzilor o nouă comandă
- searchItems(String title, double rating, double price) – returnează o listă de produse în funcție de anumite filtre, care pot fi conținute(sau nu) în parametrii metodei

  Clientul poate căuta produse după nume, rating și preț. Valorile introduse de către acesta vor fi reținute în parametrii metodei. Valorile inițiale pentru acești parametri sunt null,-1,-1. Metoda utilizează *Stream*-uri pentru selectarea produselor. Dacă nu s-a introdus o anumită categorie, metoda va ști că nu trebuie aplicat filtrul corespunzător.

Clasa **DeliveryService –** implementează **IDeliveryServiceProcessing**, **Serializable**

Atribute

- menuItems (HashSet<MenuItem>) – conține lista meniu
- orders (HashMap<Order,List<MenuItem>>) – conține istoricul comenzilor

Metode

Se implementează metodele descrise mai sus, în interfața *IDeliveryServiceProcessing*. Metodele vor lucra cu cele două atribute ale *DeliveryService*.

4.2. Pachetul *data*

Conține clase care comunică cu exteriorul aplicației.

Clasa **DataWriter –** conține metode pentru scrierea datelor aplicației în fișiere (adăugare utilizatori, serializare. deserializare)

Metode

- addUser(String username, String password) – adaugă un nou utilizator în fișierul cu credențialele utilizatorilor aplicației
- serialize(DeliveryService deliveryService) – se salvează datele obiectului *deliveryService* în fișierul *data.txt* prin serializare
- deserialize() – se aduc în memoria aplicației, prin deserializare, datele obiectului *deliveryService* salvat în fișierul *data.txt*

Clasa **User –** metode pentru verificarea credențialelor utilizatorilor

Metode

- findUsername(String username) – verifică dacă există deja un utilizator cu numele din *username*
- verifyCredentials(String username, String password) – verifică dacă există un utilizator *username* cu parola *password*

Clasa **FileGenerator –** conține metode pentru generarea de rapoarte și bonuri

Metode

- generateBill(Order order, List MenuItems items) – generează factura unei comenzi introduse în istoric

Metodele de generare a fișierelor corespunzătoare metodelor omonime din **DeliveryService**

- generateTimeReport
- genrateItemReport
- generateClientReport
- generateDayReport

4.3. Pachetul *presentation*

Conține clasele care alcătuiesc interfața grafică a aplicației.

Clasa **LoginView –** view-ul pentru fereastra de autentificare

Clasa **LoginController**

Clasa **AdminView –** view-ul pentru fereastra administratorului

Clasa **AdminController**

Clasa **ClientView –** view-ul pentru fereastra clientului

Clasa **ClientController**

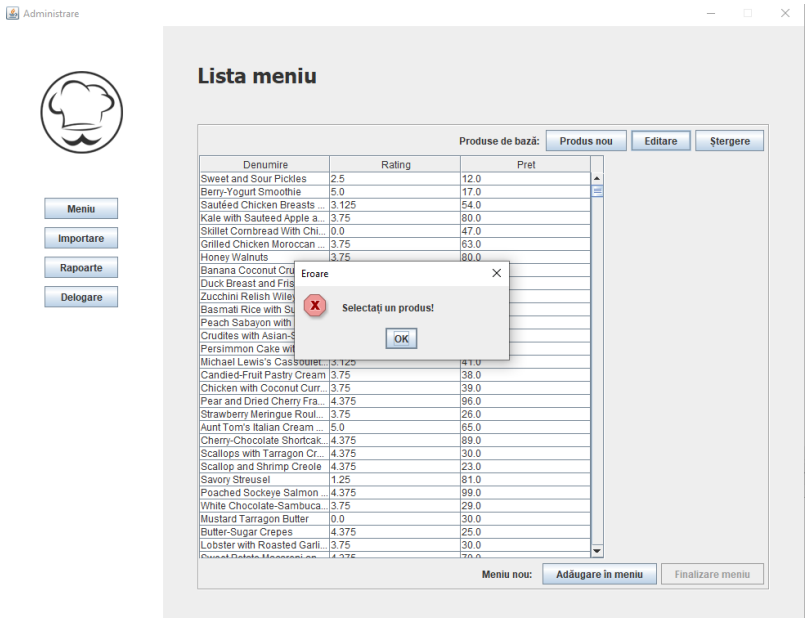
Clasa **EmployeeView –** view-ul pentru fereastra angajatului
Clasa **EmployeeController**
Clasa **Table –** conține tabelul, modelul de tabel și metodele de populare ale tabelului cu produse din lista meniu

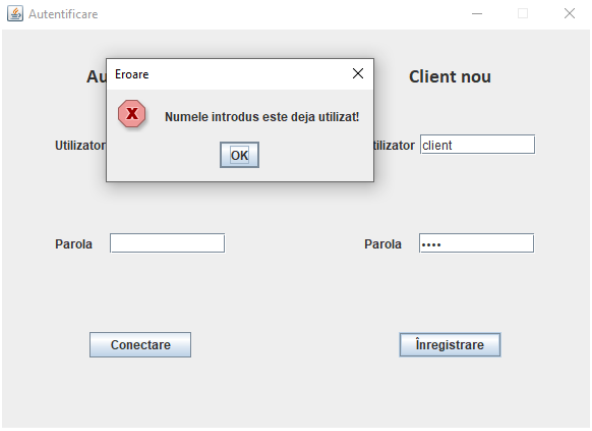
4.4. Pachetul *main*

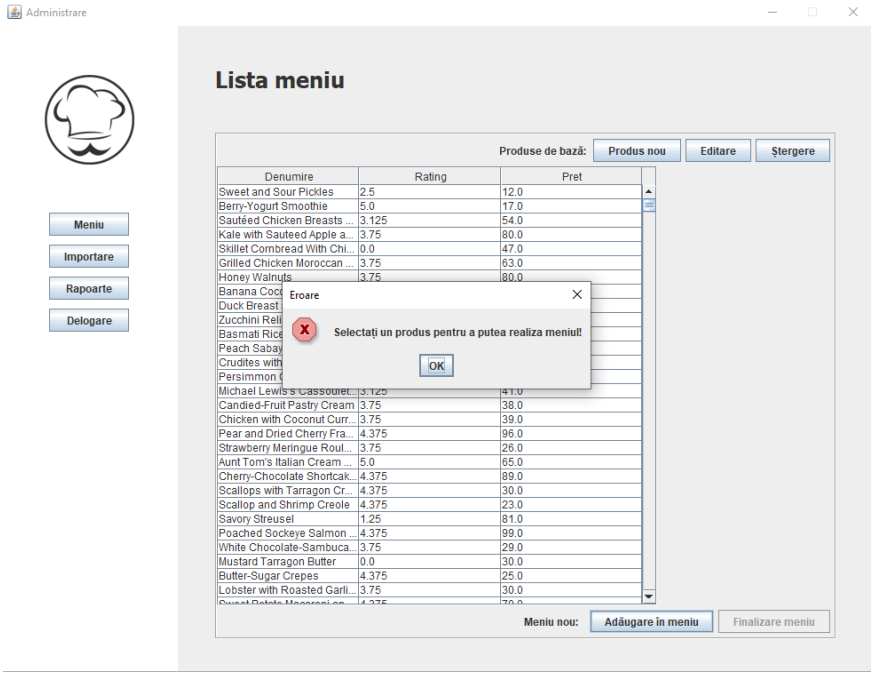
Clasa **Launch –** se execută aplicația

5. Testarea temei

Testări în aplicație ale excepțiior:







6. Concluzii și dezvoltări ulterioare

Realizând această temă, am învățat:

- să utilizez *Stream-urile*
- mai multe metode utile pentru colecții
- să lucrez mai bine și mai eficient cu fișierele externe
- conceptele de serializare și deserializare
- *Composite Design Pattern*
- *Observer Design Pattern*

În versiuni ulterioare ale aplicației, aș dori să perfecționez partea de *User Experience* a aplicației (prin implementarea a mai multor mesaje și avertismente în interfață, în plus față de cele existente) și interfața pentru angajat.

7. Webografie

1. winterbe.com, “Java 8 Stream Tutorial” - https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/
2. GeeksforGeeks, “Serialization and Deserialization in Java with Example” - https://www.geeksforgeeks.org/serialization-in-java/
3. w3schools.com, “Java Lambda Expressions” - https://www.w3schools.com/java/java_lambda.asp
4. journaldev.com, “Java SimpleDateFormat – Java Date Format” - https://www.journaldev.com/17899/java-simpledateformat-java-date-format
5. GeeksforGeeks, “Composite Design Pattern” - https://www.geeksforgeeks.org/composite-design-pattern/
6. Baeldung.com, “The Observer Pattern in Java” - https://www.baeldung.com/java-observer-pattern