**Sistem de gestionare a livrarii de mancare**

Documentatie

Cîrciu Mihnea Teodor

Grupa 30227

**Cuprins**

1. Obiectiv

2. Analiza problemei

3. Implementare

3.1 Diagrama UML

3.2 Pachete si clase

3.3 Metode si implementare

3.4 JavaDoc

3.5 GUI

4. Testare

5. Concluzii

6. Bibliografie

**1.Obiectiv**

Obiectivul acestei teme este de a implementa o aplicatie ce reprezinta un sistem de gestionare a comenzilor de mancare pentru o firma de “catering”. Un client are posibilitatea de a comanda produse care sunt in meniul firmei. Aplicatia are trei tipuri de utilizatori care se pot conecta in aplicatie cu un nume de autentificare si o parola: administratorul, angajatul si clientul. Administratorul poate sa importe produse din fisiere pentru a initializa lista de produse din meniu, poate crea, sterge, modifica sau chiar crea meniuri cu produsele de baza. De asemenea el are posibilitatea de a genera anumite rapoarte. Clientul trebuie sa se inregistreze in aplicatie iar apoi poate sa vada meniul, sa caute anumite produse in functie de proprietatile acestora, iar in final sa comande produse sau chiar meniuri create de administrator. Angajatul va fi notificat de fiecare data cand o comanda trebuie procesata.

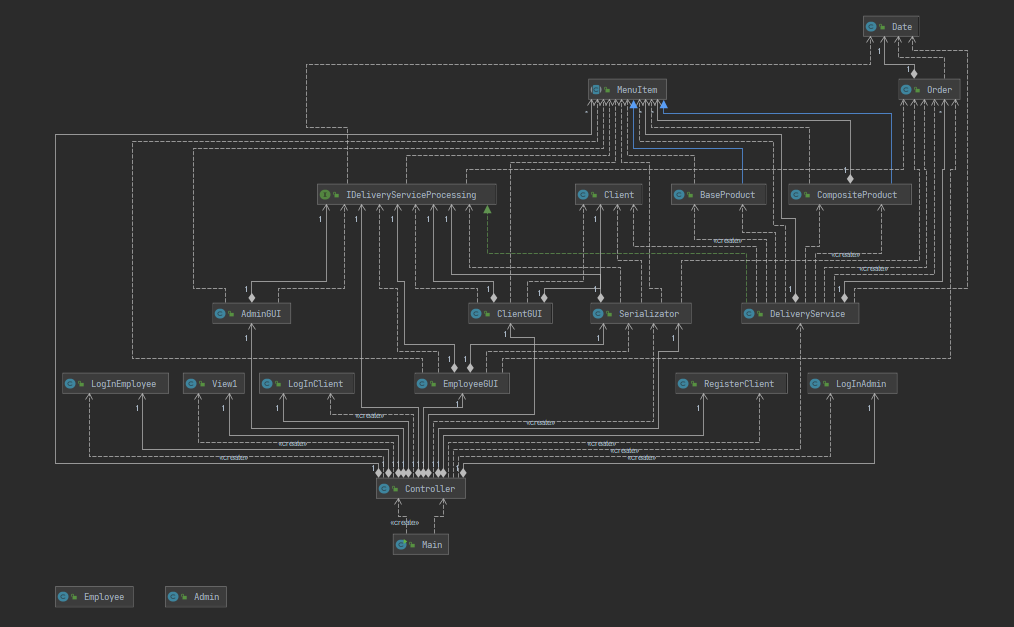
**2.Analiza problemei**

Aplicatia va porni de la diagrama de clase din fisierul pdf care prezinta cerintele temei. Produsele vor fi create folosind tehnica de “Composite Design Pattern”, pornind de la produsele de baza si creeand produse compuse care vor fi adaugate in lista care contine meniul firmei de catering. Metodele din interfata IDeliveryServiceProcessing vor fi implementate in clasa DeliveryService. Aceste metode sunt cele care realizeaza majoritatea cerintelor din aceasta tema. Clasa DeliveryService va contine de asemenea o structura de date care sa poata lega o comanda de lista sa de produse de procesat (HashMap). Sunt implementate interfete grafice pentru: ferestre de autentificare si derestre de utilizare a tuturor metodelor care indeplinesc cerintele temei. Toate informatiile din DeliveryService vor fi salvate in fisiere text pentru a putea fi refolosite dupa ce aplicatia va fi inchisa. De asemenea in functie de cerinta, se vor crea si fisiere text pentru:comenzi procesate (facturi) si rapoarte generate de administrator.

**3.Implementare**

**3.1 Diagrama UML**

Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. Pentru reprezentarea acesteia am folosit generatorul implicit de diagrame UML din Intellij.



**3.2 Pachete si clase**

Pachetul “main” contine clasa “Main” care declara “controller”-ul, iar acesta va deschide prima fereastra din care se poate folosi aplicatia.

Pachetul “presentationLayer” contine toate interfetele grafice ale aplicatiei: fereastra care alege ce fel de utilizator se va conecta la aplicatie (“View1”), fereastra de autentificare pentru administrator (“LogInAdmin”) si fereastra in care acesta poate administra si procesa produsele si rapoartele (“AdminGUI”), fereastra de inregistrare (“RegisterClient”) si autentificare (“LogInClient”) pentru client si fereastra in care clientul poate vizualiza meniul, cauta si comanda produse (“ClientGUI”) , fereastra de autentificare a angajatilor (“LogInEmployee”) si cea de procesare a comenzilor.

Clasa „View1” continte o eticheta in care se sugereaza apasarea unuia din cele trei butoane de interfata grafica: pentru autentificarea ca client, administrator sau agajat.

Clasa „LogInClient” prezinta o fereastra in care clientul se poate autentifica cu numele de utilizator sau parola si se paote conecta. In cazul in care nu detine un cont, se poate inregistra apasand pe butonul sugestiv „Inregistrare”.

Dupa apasarea butonului, se initializeaza obicetul de clasa „RegisterClient” unde acesta isi alege un nume de utilizator(pe care nu-l mai are alt client) si o parola iar apoi se poate autentifica.

Clasa „LogInEmployee” prezinta o fereastra in care angajatii se pot autentifica cu numele de utilizator si parola pe care aplicatia le pune la dispozitie acestora.

Clasa „LogInAdmin” prezinta o fereastra in care administratorul firmei se poate autentifica cu numele de utilizator si parola pe care aplicatia le pune la dispozitia acestuia.

Clasa „EmployeeGUI” contine o caseta text in care vor aparea toate comenzile procesate cat el nu a fost conectat in sesiunea respectiva. El este notificat de comenzi cu ajutorul implementarii metodelor din interfata „Subscriber”.

Clasa „ClientGUI” prezinta o fereastra cu foarte multe variabile instanta (JLabels, JTextFields, JComboBox, JEditorPane si Jbuttons) in care clientul poate vizualiza meniul, poate cauta produse si genera o comanda intr-o anumita zi, la o anumita ora.

Clasa „AdminGUI” prezinta o fereastra, de asemenea cu foarte multe variabile instanta (etichete, casete text si butoane si combobox). In aceasta fereastra administratorul poate importa produsele initiale dintr-un fiser .csv, poate crea, modifica, sterge produse de baza din meniu dar poate si crea, modifica, sterge produse compuse din alta produse existente deja in meniu. Ulterior el poate genera cele patru rapoarte prezentate in cerintele temei.

Pachetul “dataLayer” contine clasa „Serializator” care serializeaza si deserializeaza datele din „DeliveryService”, adica le salveaza in fisiere text pentru a le putea folosi si in alta sesiune de utilizare a aplicatiei.

Pachetul „businessLayer” contine clasele ( si interfata) care realizeaza toate metodele necesare in rezolvarea cerintelor: utilizarea „Composite Design Pattern”-ului pentru a crea produse simple si compuse din alte produse, generarea comenzilor, rapoartelor, cautarii in meniu in functie de anumite criterii, datele de autentificare ale administratorului, clientilor si angajatilor, clasa „Date” care creeaza obiecte speciale pentru reprezentarea zilelor din an alaturi de ore.

Clasa „MenuItem” este una abstracta care contine o singura variabila instanta care reprezinta „de cate ori a fost un produs anume comandat” („timesOrdered”), prezinta mai multe metode care vor fi implementate in clasele care o extind: „BaseProduct” si „CompositeProduct”.

Clasa „BaseProduct” reprezinta produsele de baza de meniu si contine mai multe variabile instanta care sugereaza proprietatile acestor alimente: „title, rating, calories, protein, fat, sodium, price”, dar si o „List<BaseProduct>” in care vor fi salvate produsele importate de administrator. Aceasta implementeaza mai multe metode aflate in clasa abstracta „MenuItem”.

Clasa „CompositeProduct” reprezinta produsele compuse creeate de administratorul firmei. Prezinta o „List<BaseProduct>” care contine lista de produse de baza/compuse dintr-un produs compus. De asemenea contine o variabila instanta pentru numele produsului compus si una pentru pretul total al unui produs compus.

Clasa „Admin” reprezinta administratorul firmei de catering, contine ca variabile instanta doua String-uri: un nume de utilizator si o parola care sunt implicit oferite de aplicatie iar acesta se poate autentifica cu ele.

Clasa „Employee” reprezinta agajatii firmei de catering, contine de asemenea ca variabile instanta doua String-uri: un nume de utilizator si o parola care su implicit oferite de aplicatie iar angajatii se pot autentifica cu ele.

Clasa „Client” reprezinta clientii aplicatiei, contine precum „Admin” si „Employee” doua String-uri pentru nume de utilizator si parola, dar si un ID, un contor de clienti si un numar „de cate ori a comandat clientul respectiv”. De asemenea exista si o lista in care sunt salvati toti clientii inregistrati in aplicatie. Acestia pot sa se autentifice, dar se pot si inregistra in aplicatie in cazul in care nu au deja un cont.

Clasa „Date” reprezinta informatiile legate de data si ora unei comenzi, prezinta patru variabile instanta de tip „int” pentru an, luna, zi si ora.

Clasa „Order” reprezinta informatiile unei comenzi in legatura cu: numarul comenzii, numarul clientului (din care se pot afla numele de autentificare a clientului <in cazuri exceptionale, parola>), data comenzii, pretul total al comenzii, dar si o varibila statica in care se contorizeaza cate comenzi au fost procesate.

Interfata „IDeliveryServiceProcessing” este cea care contine metodele pentru administrator: importarea produselor, administrarea produselor si alimentelor din meniu, generarea rapoartelor, si pentru client: creearea unei noi comenzi, generarea unei facturi si cautarea unor produse in functie de anumite criterii.

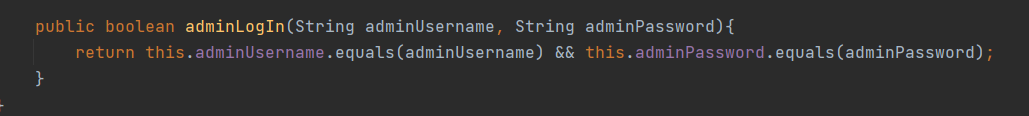
Clasa „DeliveryService” este cea care implementeaza interfata „IDeliveryServiceProcessing” alaturi de metodele sale. Ea prezinta trei liste: una pentru produsele din meniu „List<MenuItem>”, una pentru produsele de baza „List<BaseProduct>” si una pentru comenzi „List<Order>”, dar si un HashMap („HashMap<Order,List<MenuItem>>”) in care vor fi salvate comenzile impreuna cu lista de alimente comandate.

Pachetul „controller” prezinta clasa „Controller” in care se implementeaza toate actiunile aplicatiei la apasarile butoanelor. Este clasa „creier” care gestioneaza legaturile dintre interfetele grafice, metodele si datele care sunt salvate in aplicatie. Contine mai multe sub-clase, fiecare implementand metoda „actionPerformed(ActionEvent e)” din interfata „ActionListener”.

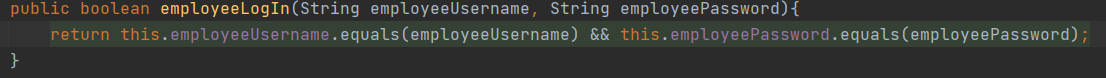
**3.3 Metode si implementare**

Pachetul „businessLayer” contine clasele cu metodele cele mai semnificative:

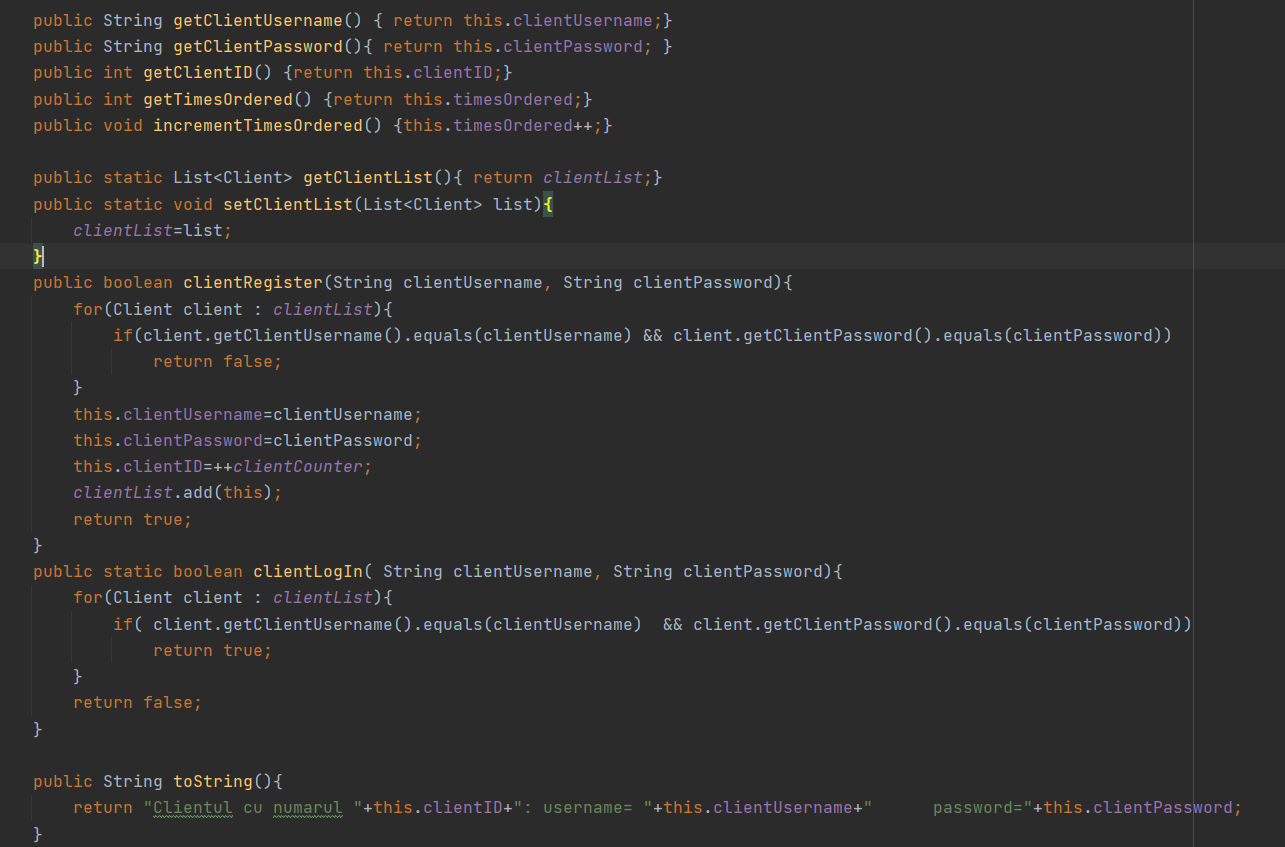
Prima clasa este cea de „Admin”. Metoda sa de „adminLogIn” realizeaza conexiunea intre utlizatorul de tip administrator si aplicatie.



Cea de-a doua clasa este cea de „Employee” a carei metoda, „employeeLogIn”, de asemenea realizeaza conexiunea intre utilizator, de data aceasta de tip angajat si aplicatie.

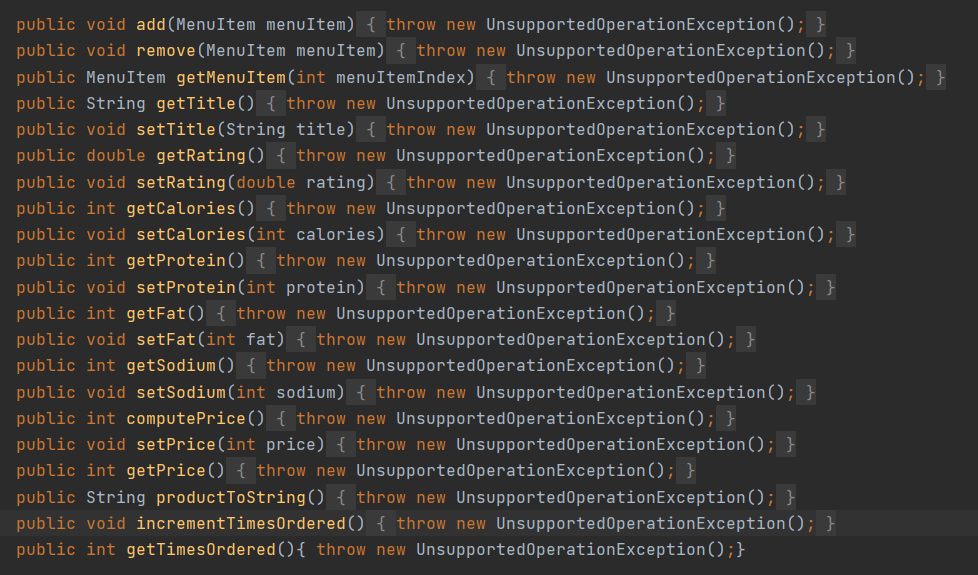


A treia clasa care realizeaza conexiunea la aplicatie este cea de „Client” care contine insa, mai multe metode, printre care si cea de conectare, „clientLogIn”, insa sunt si metode de: inregistrare in aplicatie pe baza unui nume de utilizator si parola, dar si get-ere pentru variabilele instanta ale clasei. De asemenea clasa prezinta metoda care returneaza informatiile clientului sub forma de String dar si o metoda care incrementeaza o variabila initial nula, de fiecare data cand clientul plaseaza o comanda.



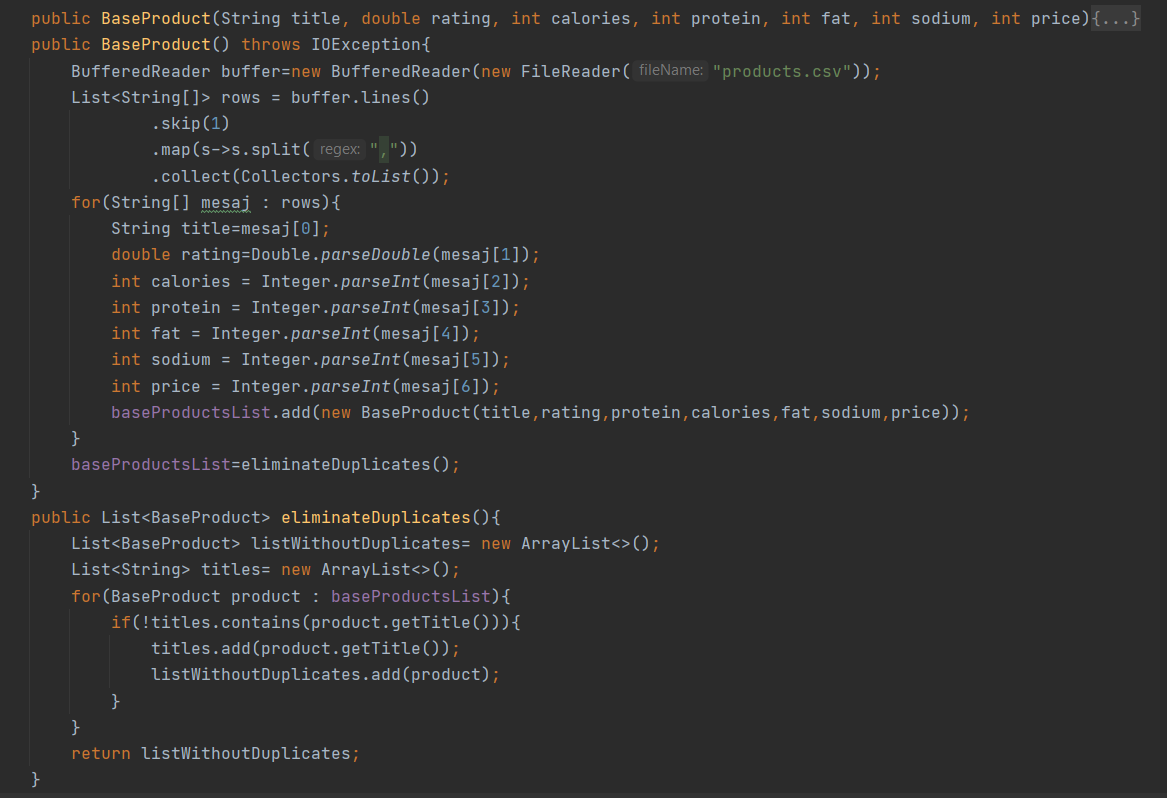
Alte trei clase semnificative ale acestei teme sunt: „MenuItem”, „BaseProduct” si „CompositeProduct” care implementeaza tehnica de „Composite Design Pattern”:

Clasa „MenuItem” este una abstracta si de asemenea implementeaza interfata „Serializable” pentru a putea serializa si deserializa anumite informatii ulterior. Aceasta prezinta mai multe metode neimplementate, utilizate in clasele care o extind.



Urmatoarea clasa, „BaseProduct”, extinde clasa „MenuItem” si implementeaza metodele acesteia, impreuna cu altele. Clasa reprezinta produsul de baza cu proprietatile sale. Aceasta prezinta get-ere si set-ere, metoda care returneaza informatiile produsului de baza sub forma de String.

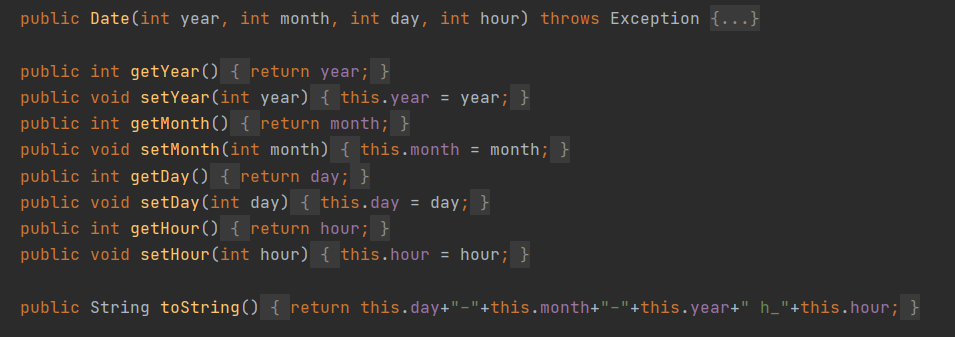


Pe langa aceste metode, clasa „BaseProduct” prezinta un constructor care importa produsele din fisierul „products.csv” si le adauga in lista de produse, implicit si in cea de meniu, iar o metoda extrem de utila este cea care elimina produsele duplicate.

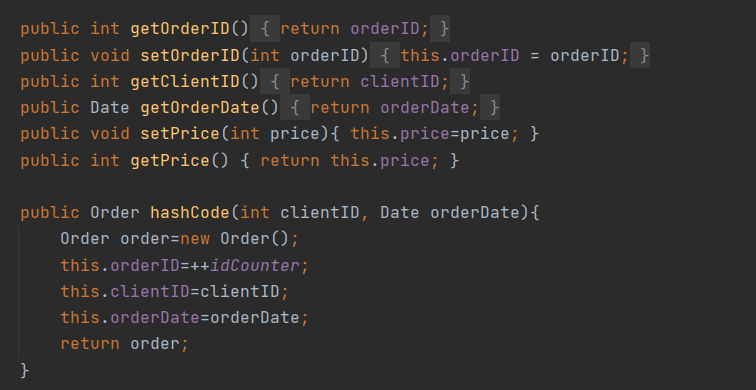
Cea de-a treia clasa care completeaaza „Composite Design Pattern” este „CompositeProduct”: extinde de asemenea clasa „MenuItem” si consta intr-o lista de obiecte de tip „BaseProduct” care sa alcatuiasca un „meniu cu produse de baza”. Aceasta prezinta metode de adaugare si eliminare din lista, get-ere, compunearea pretului produsului final, returnarea informatiilor produsului sub forma de String.



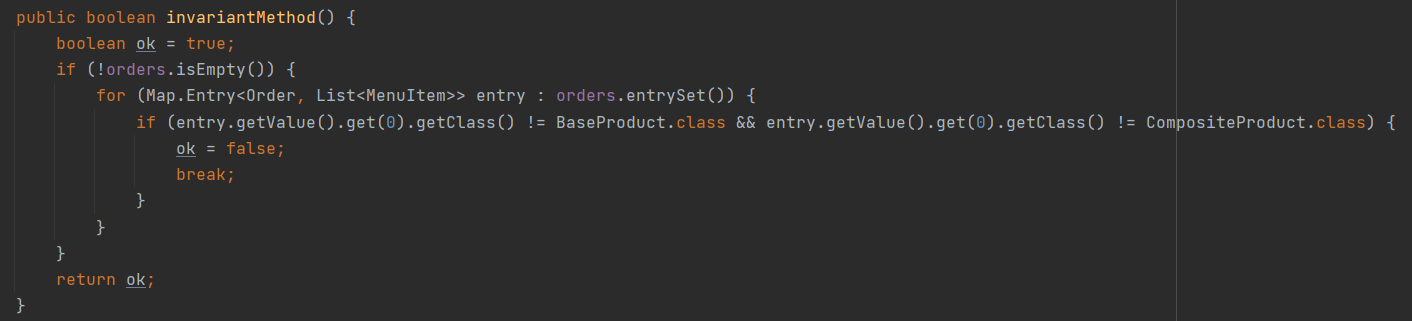
O alta clasa care contribuie la procesarea unei comenzi este clasa „Date” care reprezinta data in care comanda sa fie livrata. Aceasta clasa contine get-ere si set-ere pentru variabilele sale instanta si o metoda care returneaza data sub forma de String.



Clasa ce prezinta informatiile despre o comanda este „Order”, aceasta contine variabile instanta pentru: numarul comenzii, numarul clientului, data in care sa fie livrata comanda (se foloseste de un obiect de clasa „Date”), un pret total si un contor pentru numarul comenzii. Clasa prezinta get-ere si set-ere pentru aceste variabile, dar si o metoda „hashCode” care returneaza comanda creata in functie de valorile variabilelor instanta.



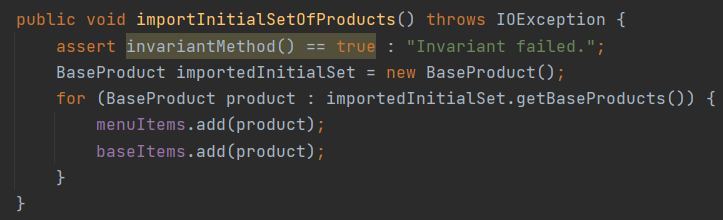
Clasa „DeliveryService” prezinta metodele prin care administratorul si clientii interactioneaza cu datele din aplicatie. Aceasta contine 23 de metode printre care: o metoda de tip „invariant”/”wellFormed” care este verificata la inceputul fiecarei metode din clasa prin „assert” si afiseaza in consola in cazul negativ, mesajul „Invariant failed”, aceasta trebuind sa fie mereu adevarata. Aceasta metoda verifica daca fiecare comanda contine cel putin un produs de tip „BaseProduct”,



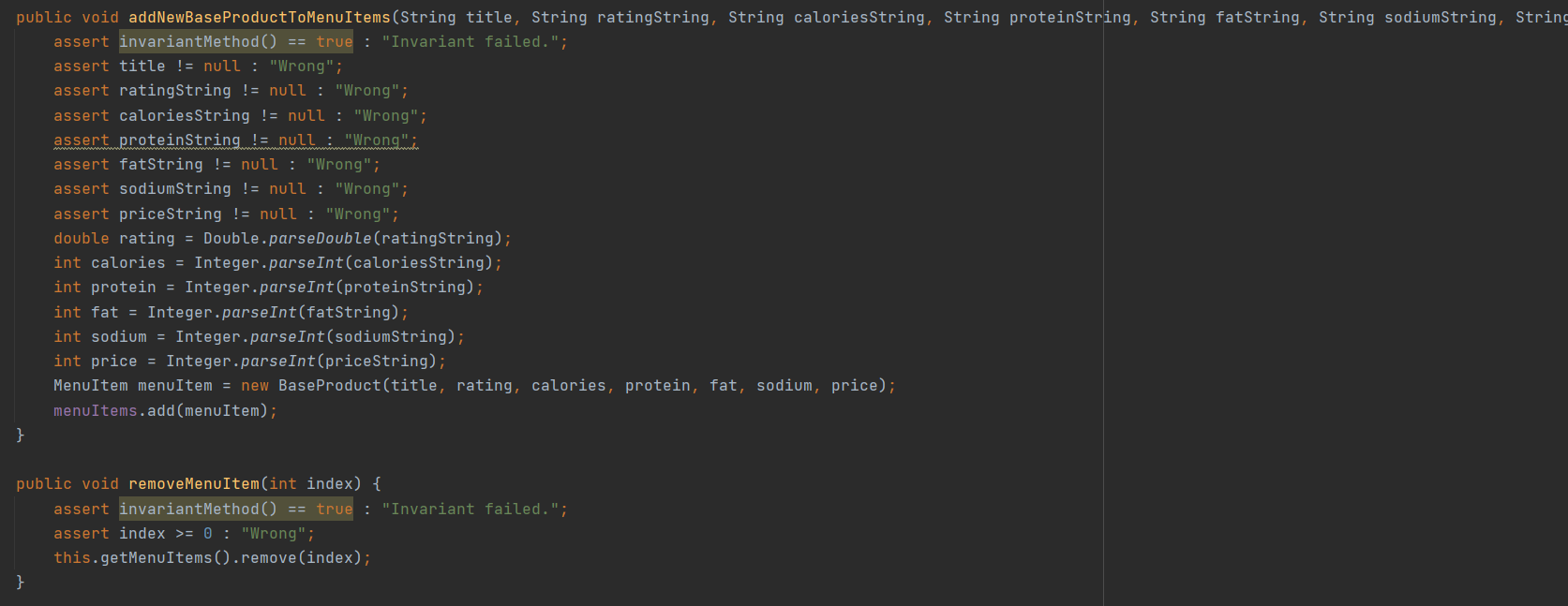
get-ere si set-ere pentru listele si tabelul de dispersie din variabilele instanta ale clasei,

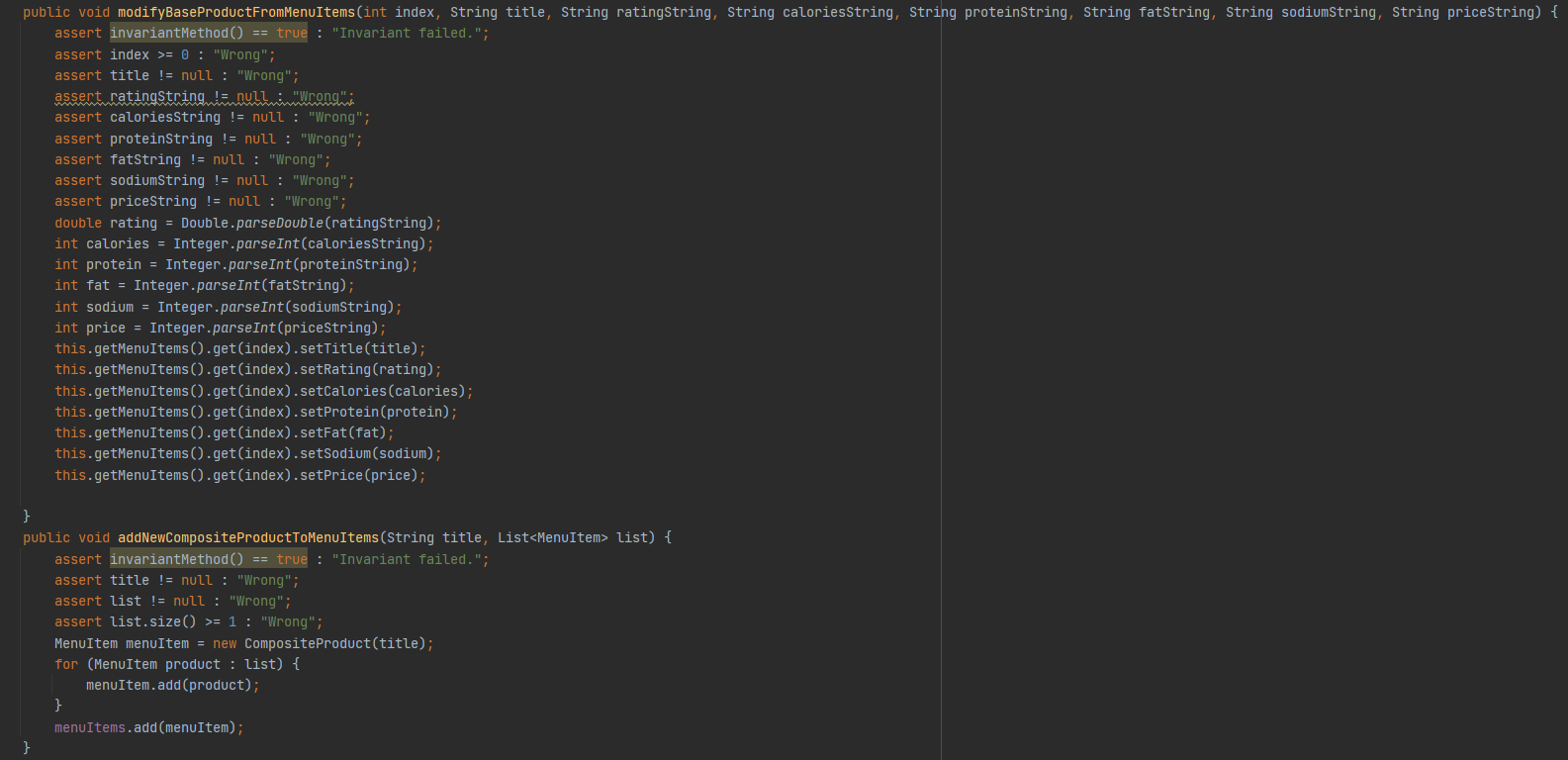


metoda prin care se apeleaza importarea produselor de baza,

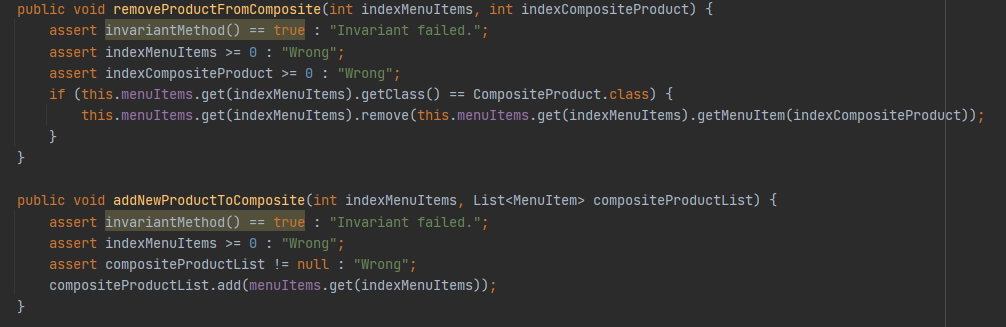


metode de adaugare, eliminare si modificare a produselor din meniu, de tip „BaseProduct” sau „CompositeProduct”

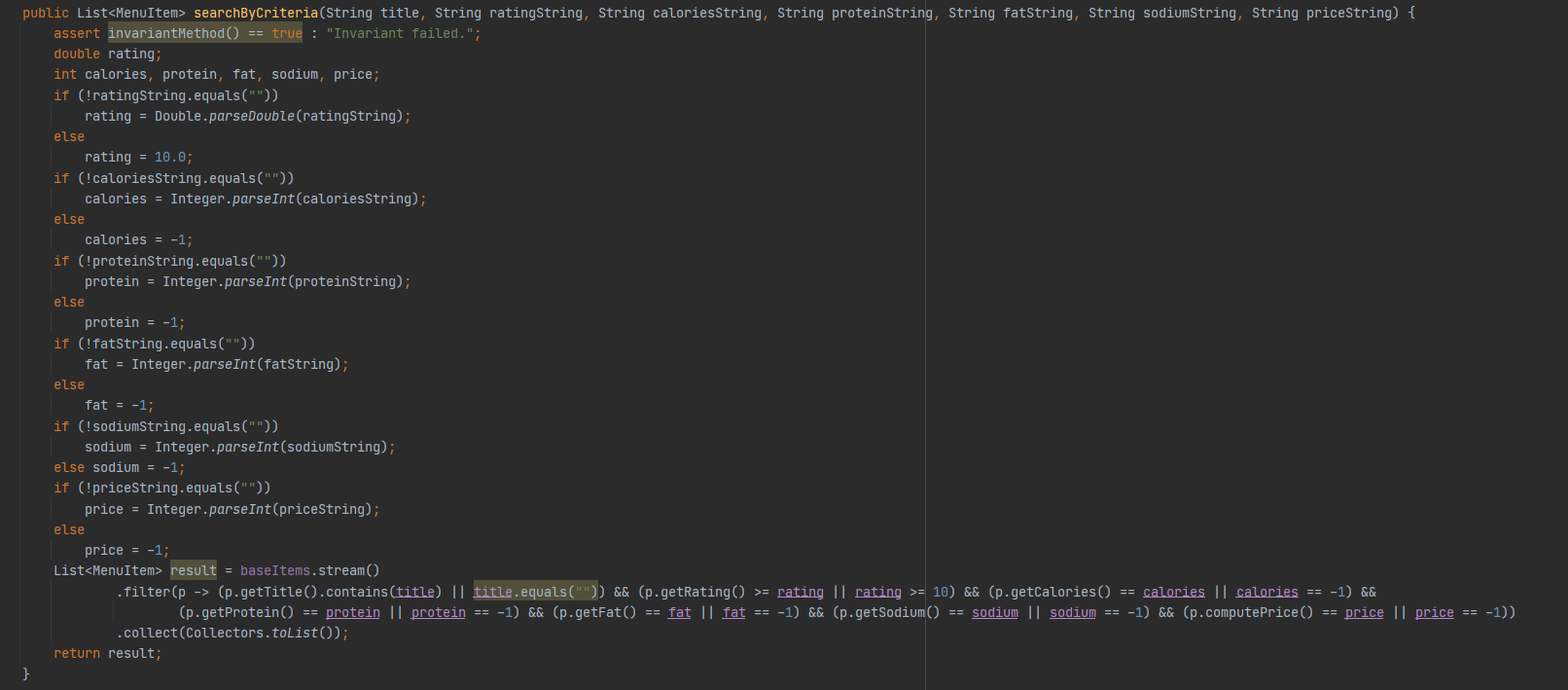




metode de adaugare sau eliminare ale unor produse „BaseProduct” dintr-un produs „CompositeProduct”



metoda prin care clientul poate cauta un anumit produs in functie de diferite criterii, adica proprietatile produsul, „title, rating, calories, proteins, fats, sodium, price”



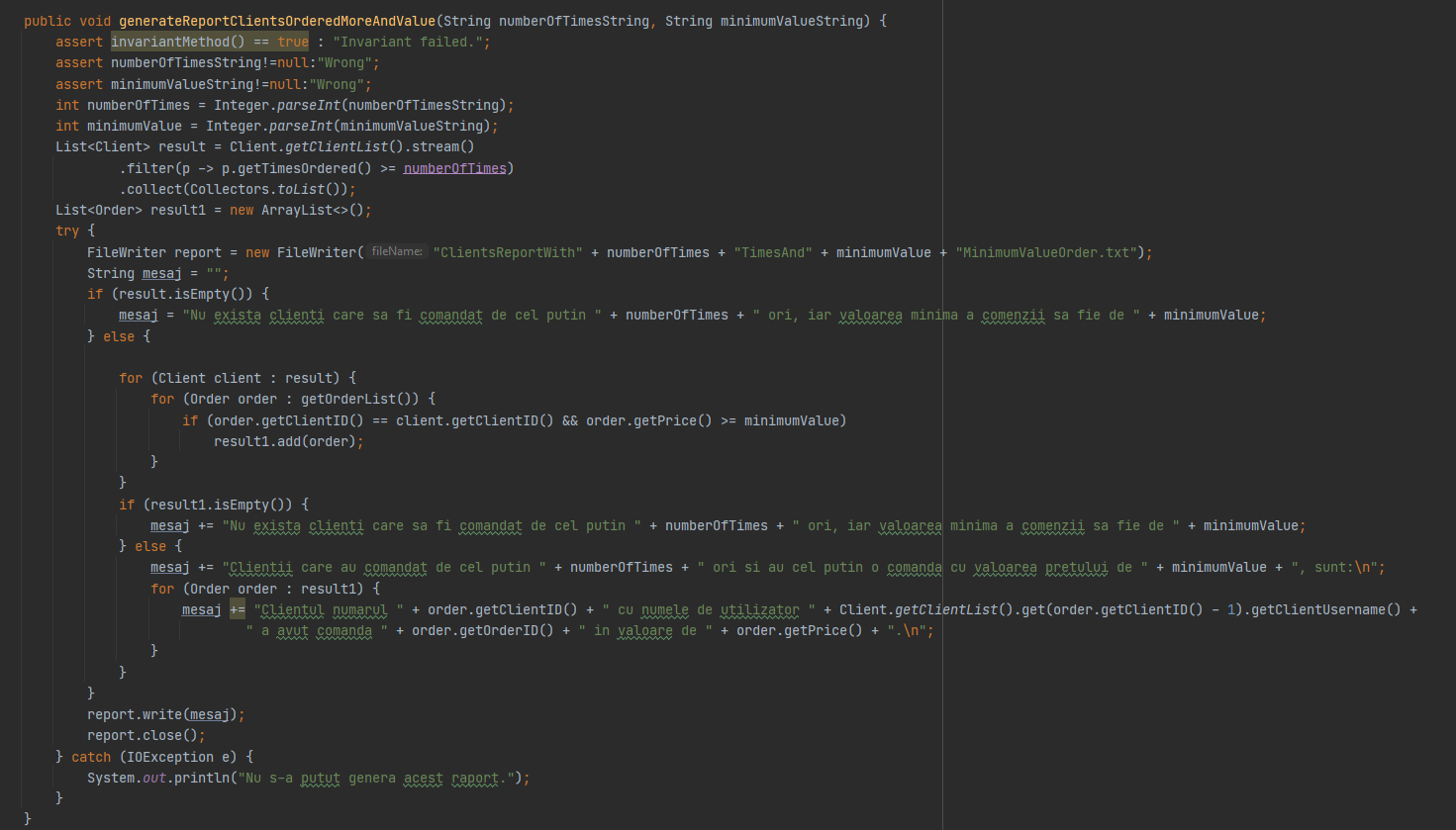
metoda prin care administratorul genereaza primul raport din cerinte „raport in care sa fie afisate comenzile procesate pentru un anumit interval orar”



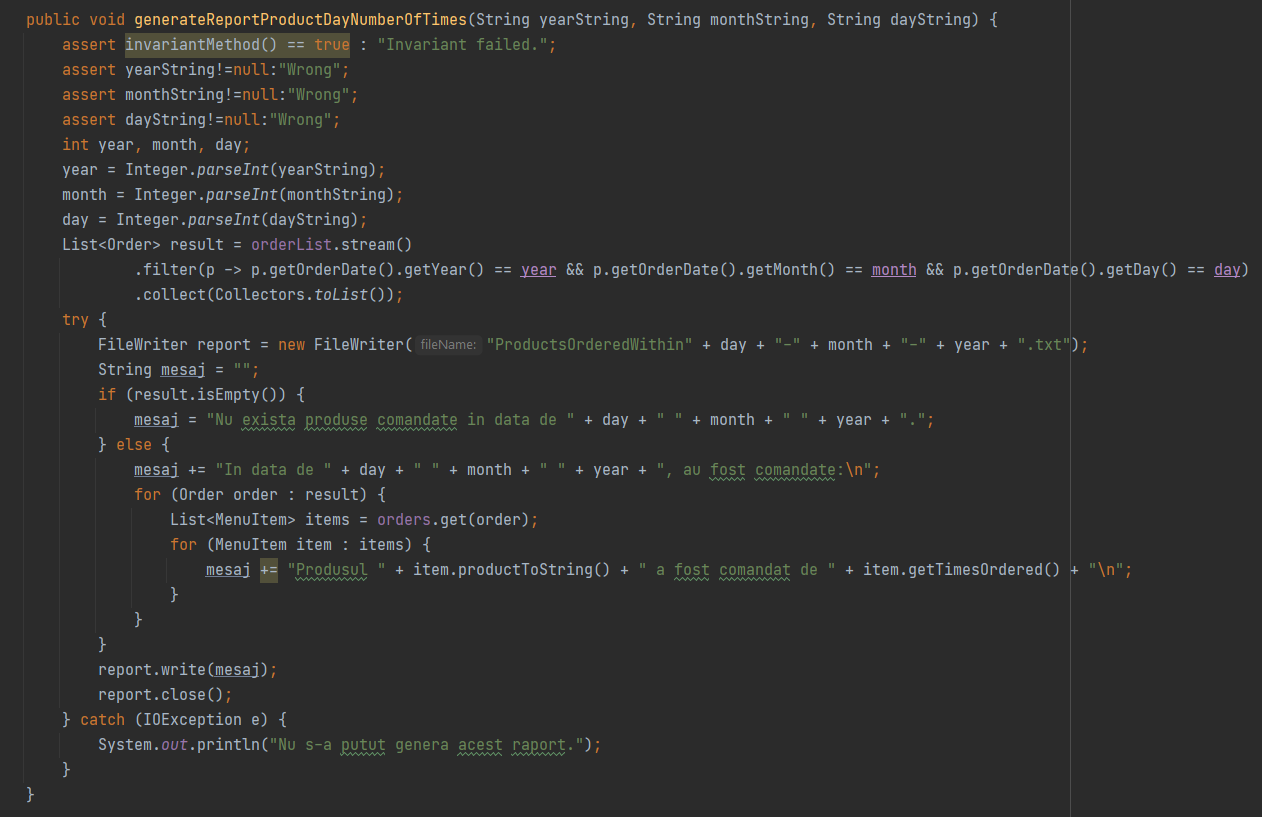
metoda prin care administratorul genereaza al doilea raport din cerinte „raport cu produsele comandate mai mult decat un anumit numar de ori”



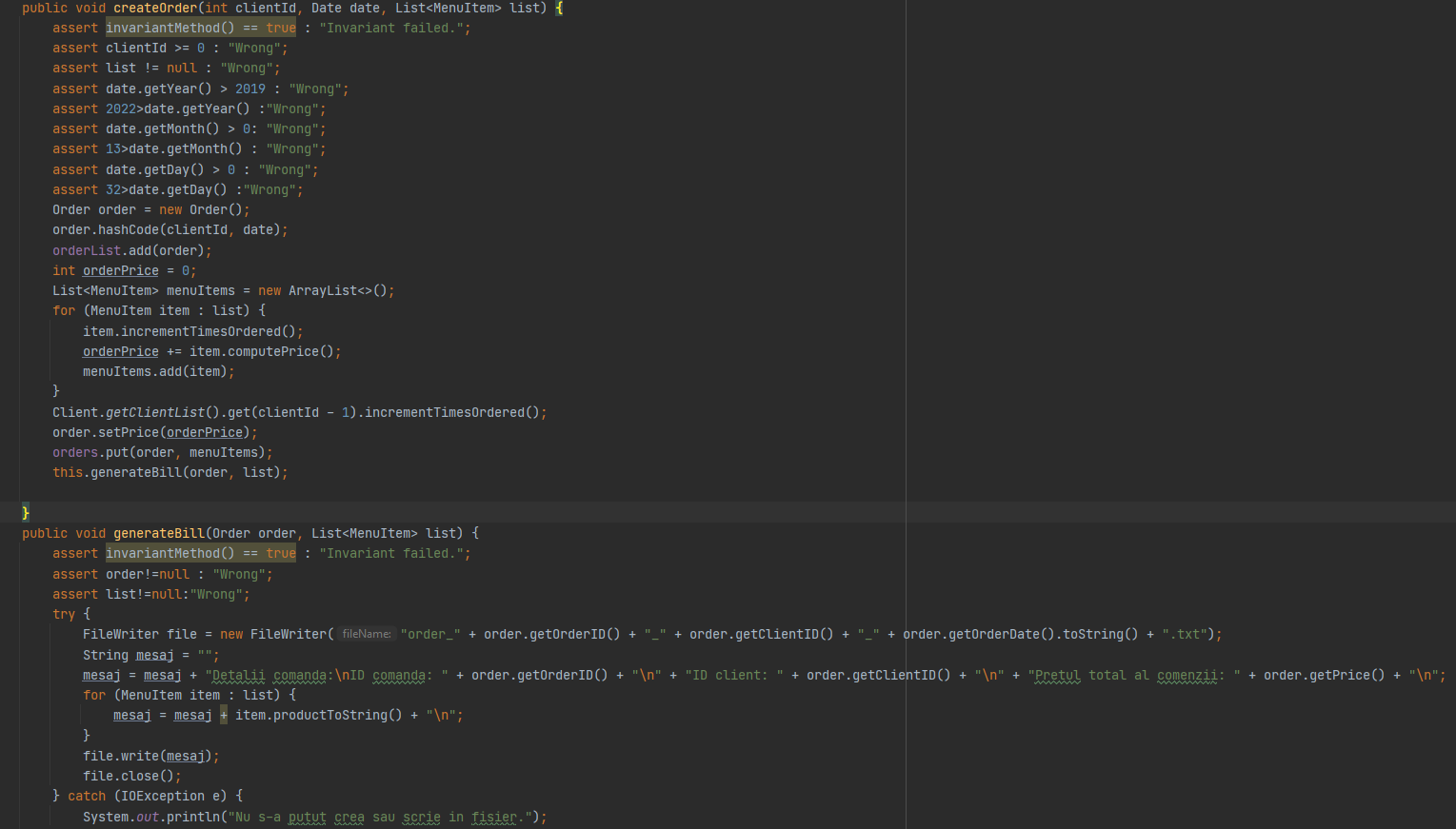
metoda prin care administratorul genereaza al treilea raport din cerinte „raport cu clientii care au comandat mai mult decat un anumit numar de ori, iar pretul comenzii este mai mare decat o anumita valoare”



metoda prin care administratorul genereaza cel de-al patrulea raport din cerinte „produsele comandate intr-o anumita zi si de cate ori au fost ele comandate”

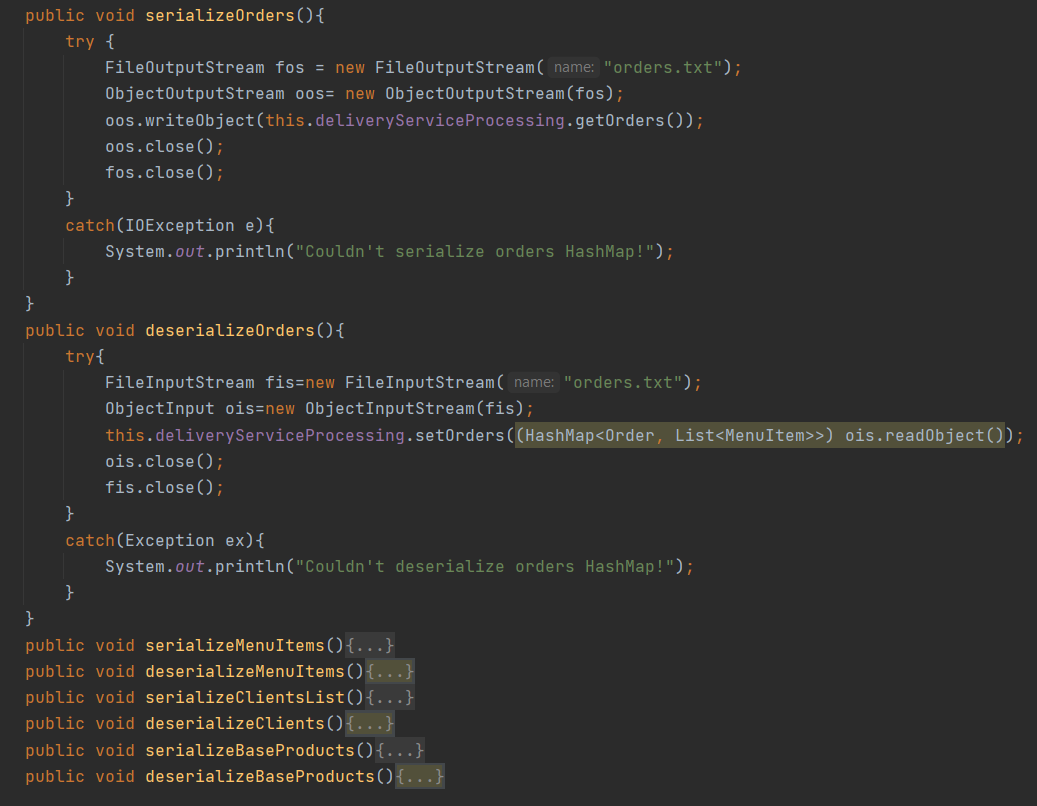


si metode prin care clientul creeaza comanda iar apoi se genereaza o „factura” cu detaliile comenzii

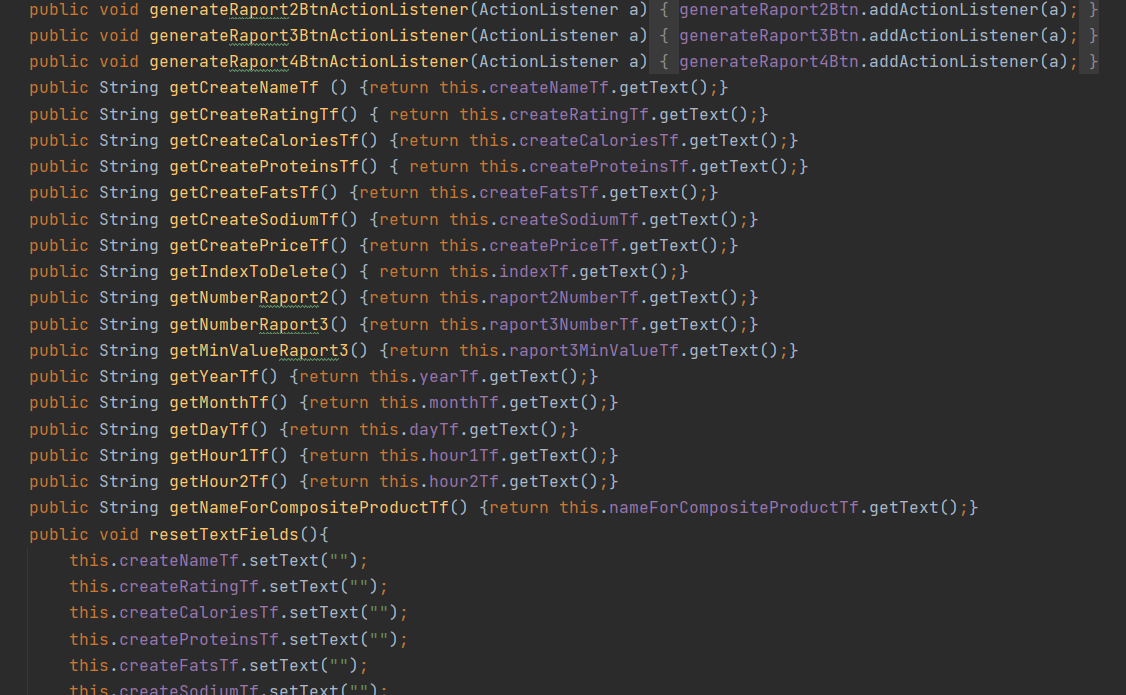


Generarea rapoartelor si facturilor pentru comenzi se face creeandu-se fisiere text (.txt) in care se scriu mesajele corespunzatoare metodelor.

Pachetul „dataLayer” contine o singura clasa, „Serializator”, prin care informatiile care sunt generate si datele aplicatiei legate de clasa DeliveryService sunt salvate in fisiere text (.txt):”orders.txt”, „menuItems.txt”, „clients.txt” si „baseProducts.txt”. Pentru a salva aceste informatii in fisiere text, clasa „Serializator” implementeaza interfata „Serializable”. Clasa contine metode de serializare si deserializare pentru: lista de comenzi, lista de produse din meniu, lista de clienti si lista de produse de baza care sunt importate initial.



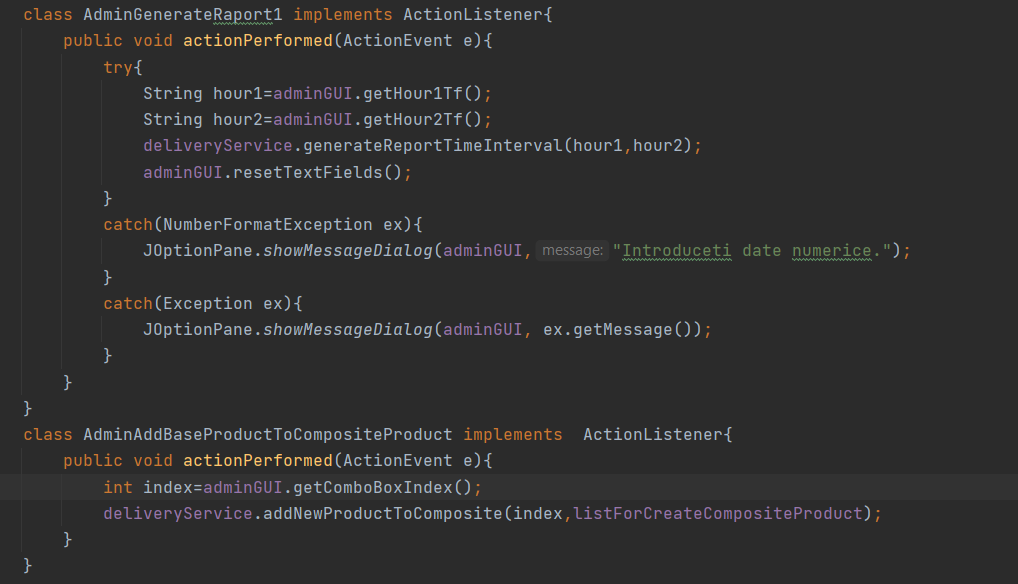
Pachetul „presentationLayer” contine clasele care reprezinta interfetele grafice ale aplicatiei, acestea prezinta metode get si set pentru casetele text si combo box, dar si metode prin care butoanelor li se atribuie actiuni la momentul apasarii, de exemplu:



O clasa care prezinta metode diferite de cele prezentate mai sus este clasa „EmployeeGUI” care, pe langa faptul care deschide o fereastra dupa autentificare angajatului, implementeaza interfata „Subscriber” prin care notifica angajatul atunci cand se proceseaza o comanda, afisand-o intr-un JEditorPane, aceste metode sunt:



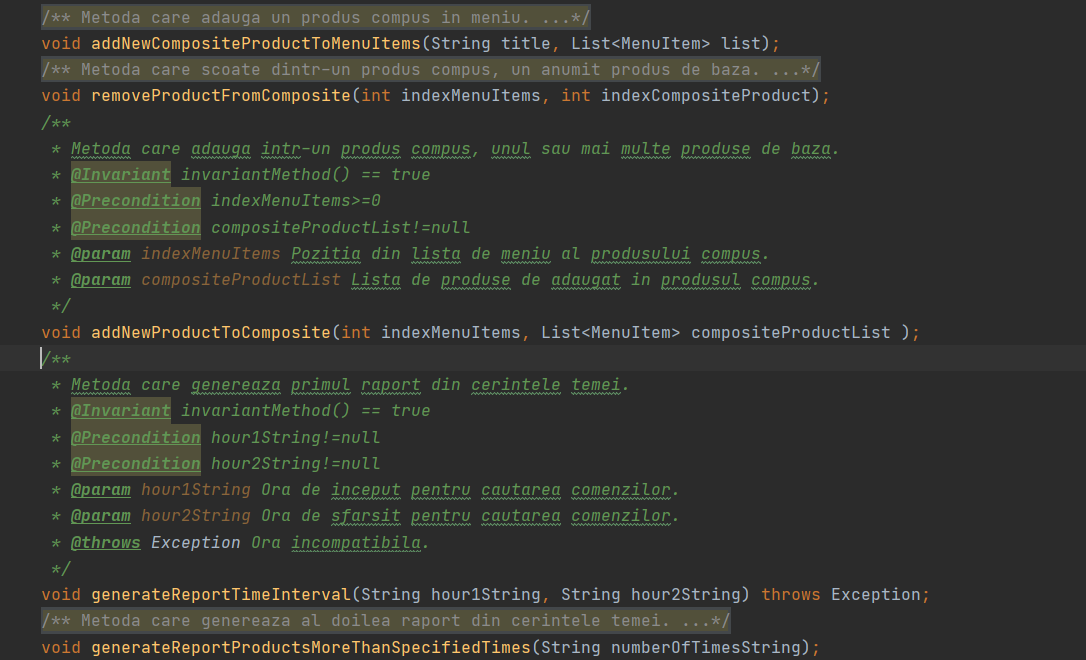
Pachetul „controller” contine o singura clasa „Controller” in care sunt implementate actiunile si raspunsurile aplicatiei atunci ca utilizatorul apasa butoanele din interfetele grafice si afisarea acestora:

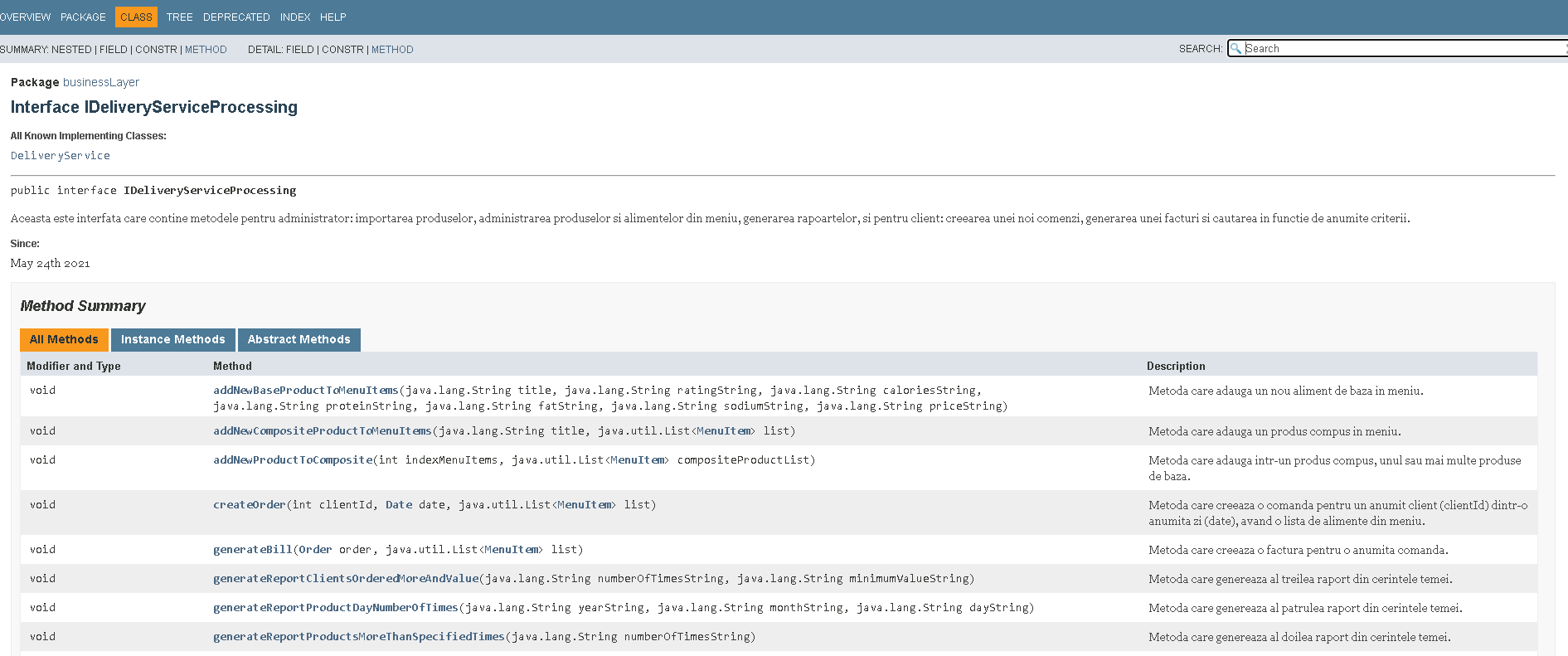


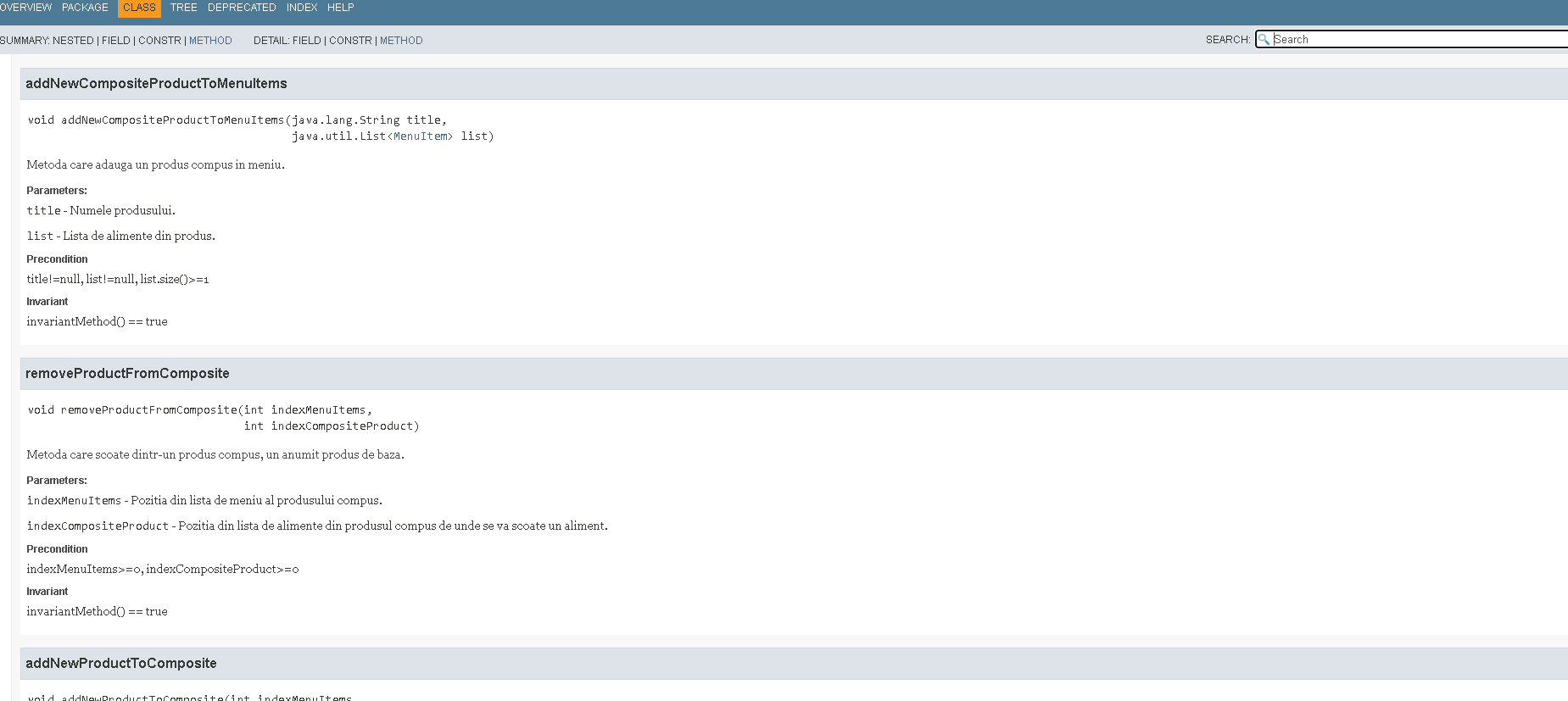


**3.4 JavaDoc**

O alta cerinta a acestei teme este crearea fisierelor JavaDoc pentru interfata „IDeliveryServiceProcessing” folosind tag-uri speciale in care verific metoda de tip „Invariant” si conditiile de tip „Precondition” si „Postcondition” pentru fiecare metoda din interfata, implementate in „DeliveryService”.







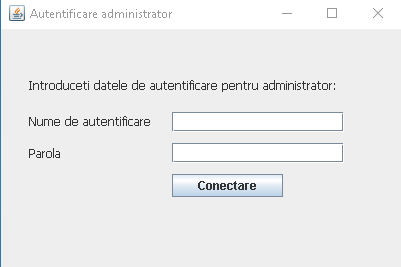
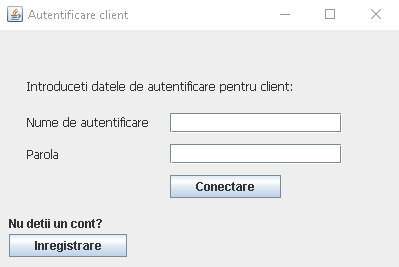
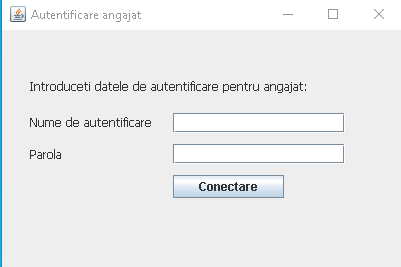
**3.5 GUI**

Aplicatie beneficiaza de interfete grafice pentru:

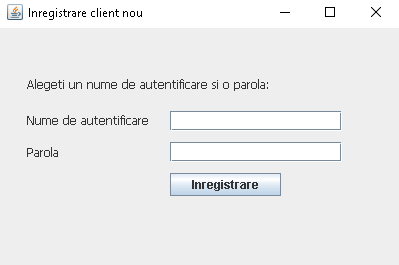
„View1” in care utilizatorul alege fereastra in care sa se autentifice la aplicatie: administrator, client sau angajat.



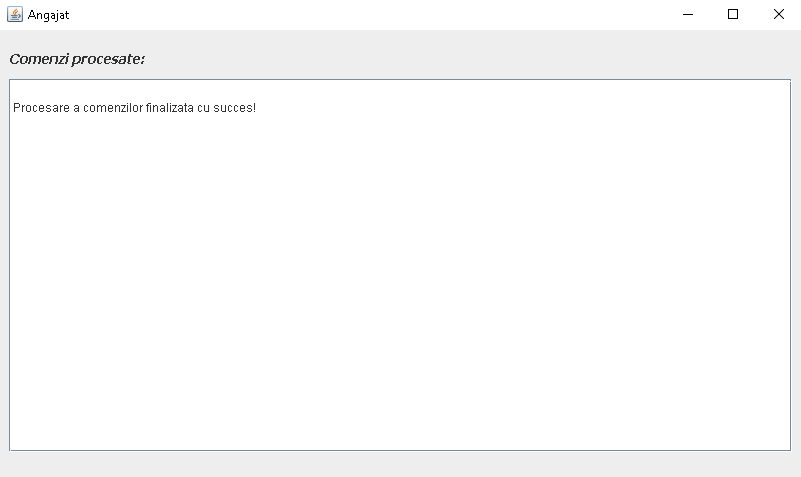
„LogInEmployee” in care utilizatorul se poate autentifica pentru angajat, asemanator si pentru „LogInClient”-client si „LogInAdmin”-administrator .



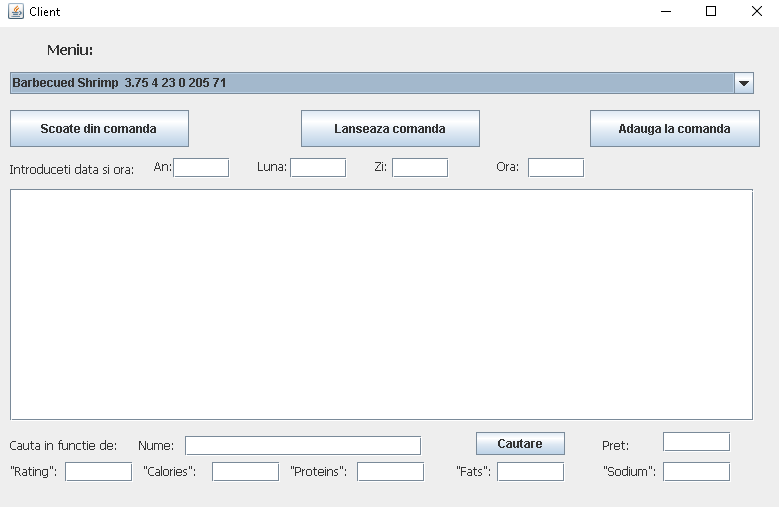
In cazul in care utilizatorul este un client care nu are nume de utilizator si parola, acesta se poate inregistra apasat pe butonul „Inregistrare” din fereastra „Autentificare client” creeata de clasa „RegisterClient” .



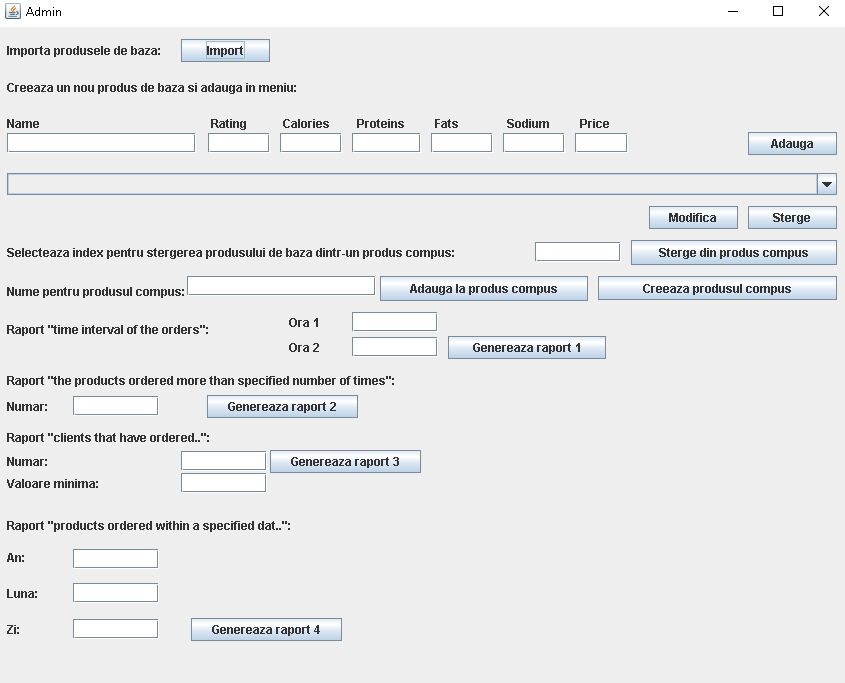
„EmployeeGUI” este clasa care deschide fereastra pentru angajat:



„ClientGUI” este clasa care deschide fereastra pentru client:

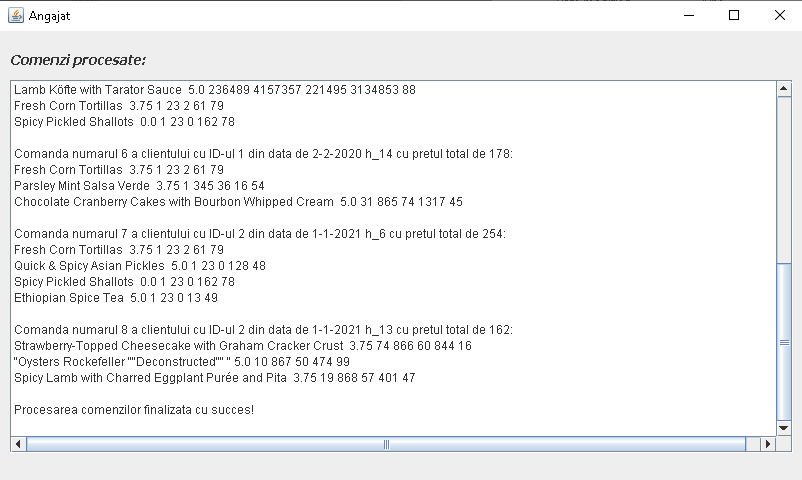


„AdminGUI” este clasa care deschide fereastra pentru administrator:

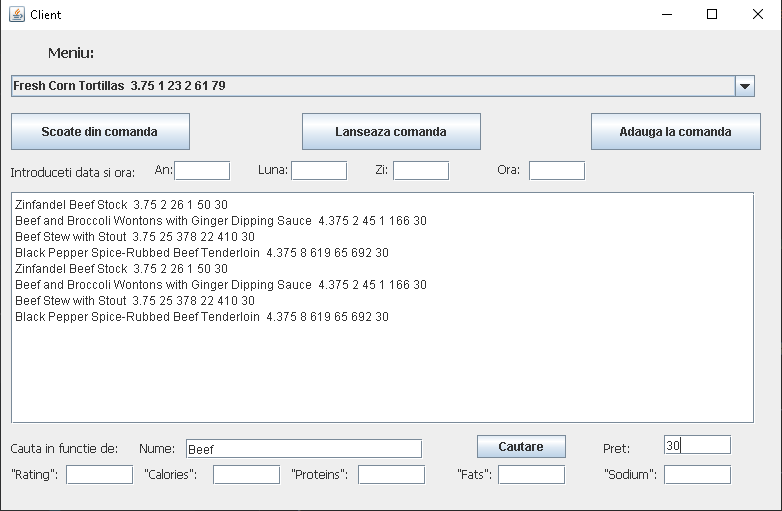


**4. Testare**

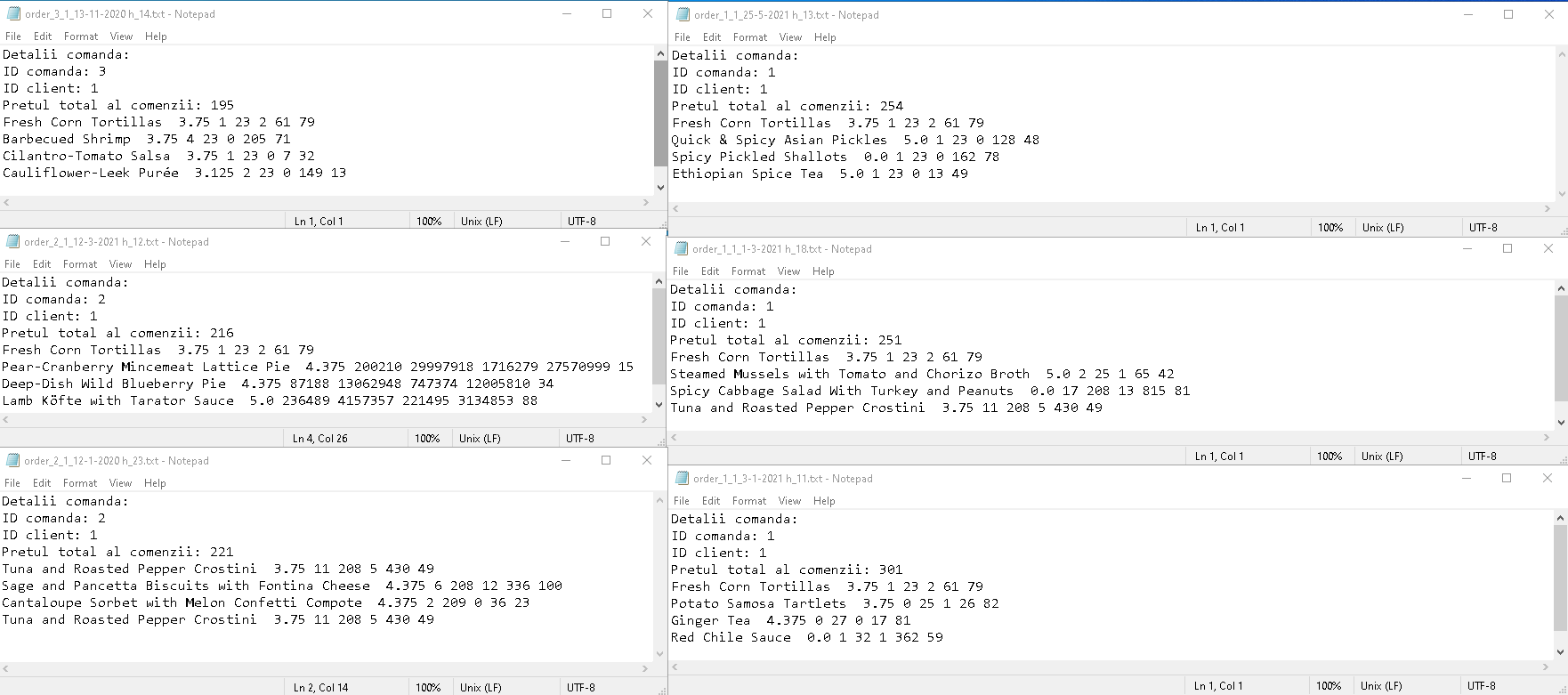
Procesarea comenzilor de catre angajat:



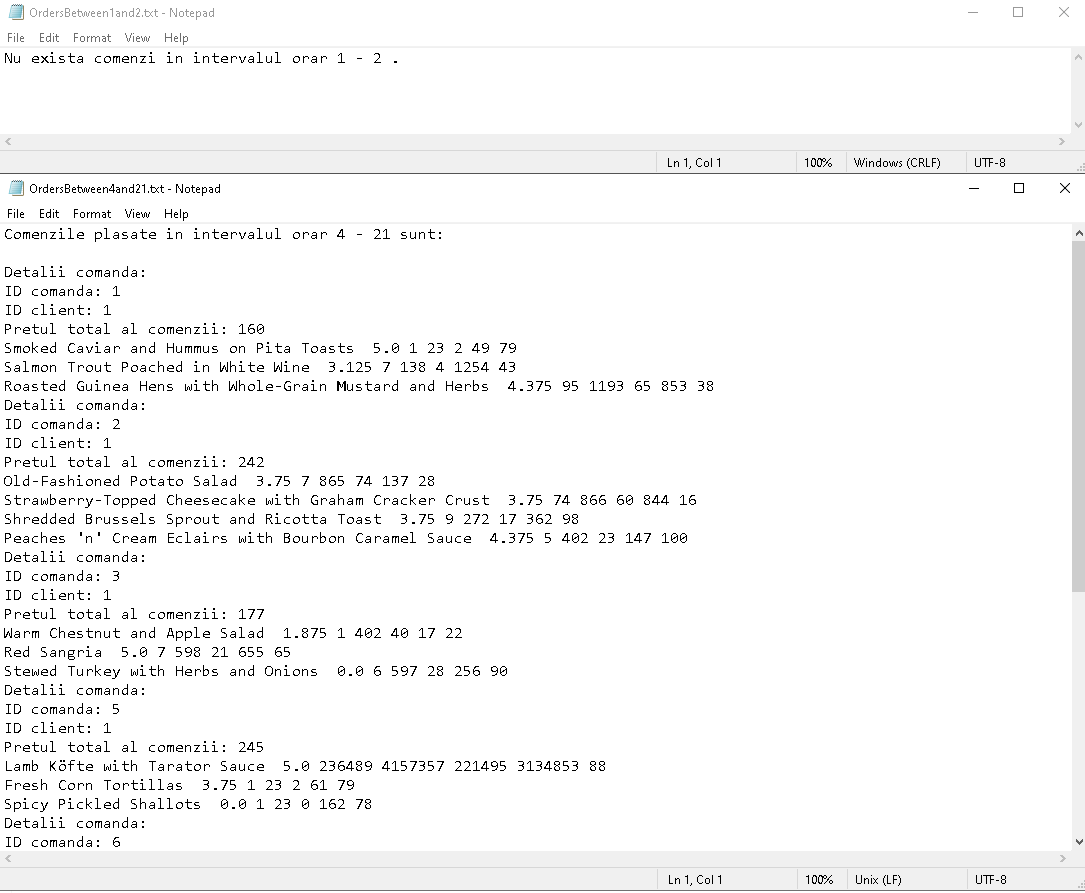
Cautarea produselor in functie de anumite criterii ale clientului.



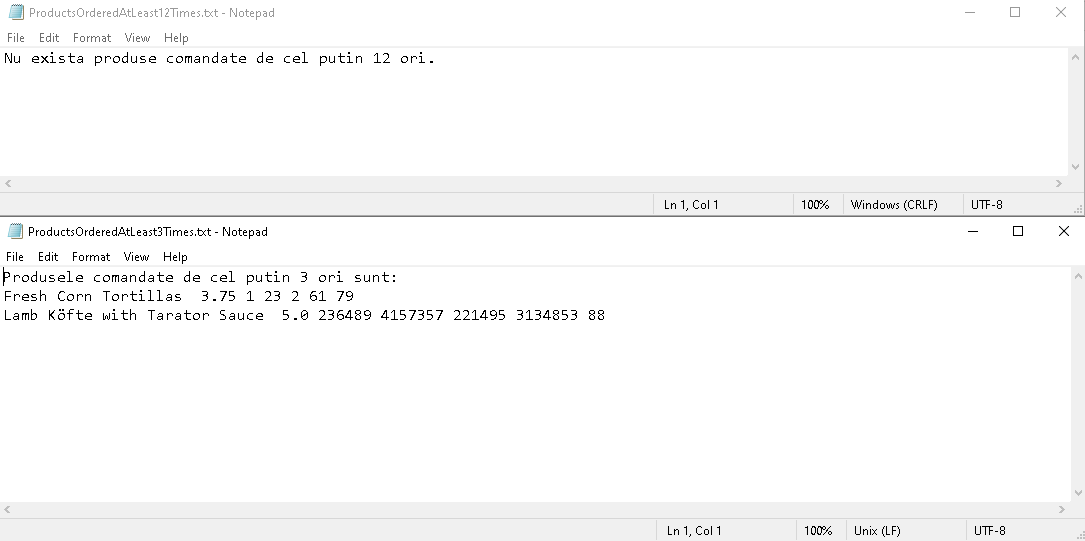
Generarea comenzilor de catre administrator in format .txt:



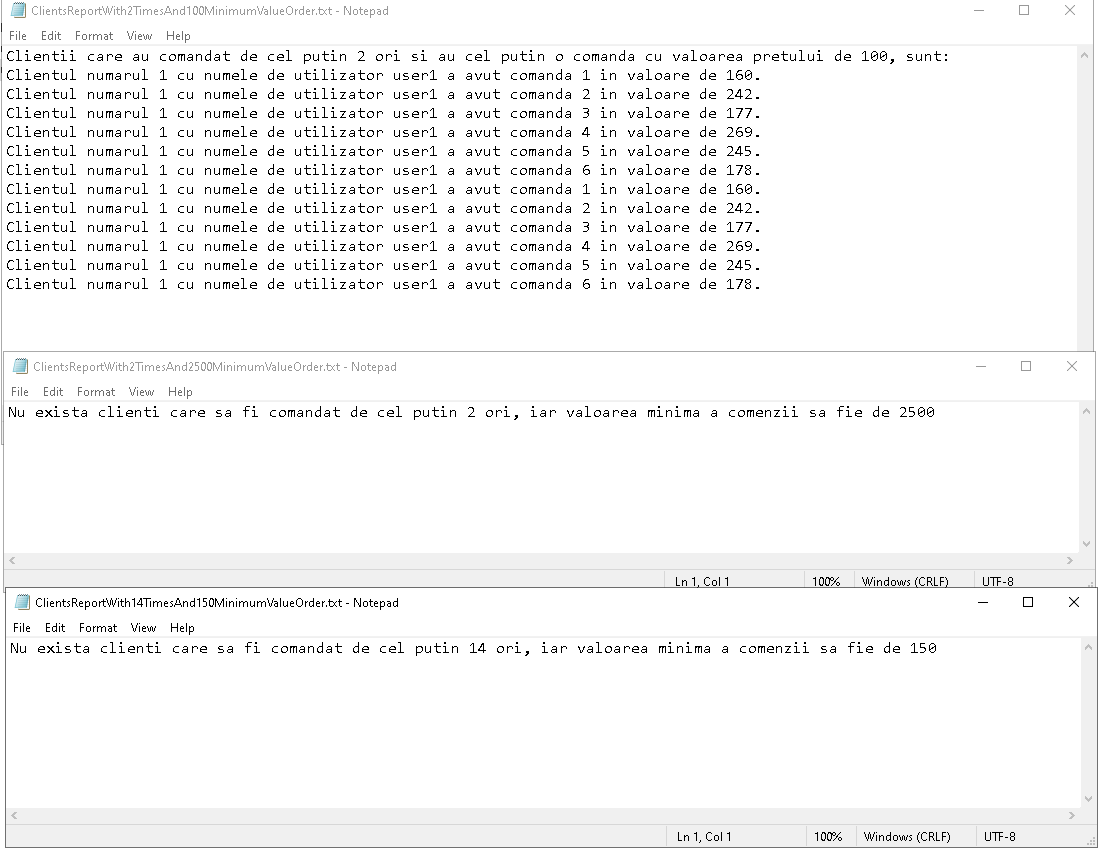
Generarea primului raport de catre administrator:



Generarea celui de-al doilea raport de catre administrator:



Generarea celui de-al treilea raport de catre administrator:



Generarea celui de-al patrulea raport de catre administrator:



**5. Concluzii**

In concluzie, acest proiect m-a ajutat sa aprofundez notiunile de Programare Orientata pe Obiect, implicit Java. Consider ca cele mai importante si complexe aspecte ale acestei teme au fost: importarea datelor dintr-un fisier .csv, utilizarea „labda expressions” si „stream processing” , utilzarea „Composite Design Pattern” –ului, tag-urilor speciale folosind conditiile pre, post si invariant, implementarea interfetei „Subscriber” si serializarea impreuna cu deserializarea. Aplicatia poate beneficia de ulterior de imbunatatiri.

**6. Bibliografie**

https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

http://www.tutorialspoint.com/java/java\_serialization.htm

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

http://javarevisited.blogspot.ro/2012/01/what-is-assertion-in-java-java.html

https://stackoverflow.com/