

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

**FUNDAMENTAL PROGRAMMING TECHNIQUES**

**ASSIGNMENT 4**

**FOOD DELIVERY MANAGEMENT SYSTEM**

**Nume si prenume: Stecko Daiana**

**Grupa 302210 | An 2 semestrul 2**

**Profesor laborator: Mitrea Dan**

**Cuprins**

**1.Problema si solutia problemei.**

**2.Obiectivul temei.**

**3.Analiza problemei, modelare, scenarii, cazuri de utilazare.**

**4.Proiectare.**

**5.Implementare.**

**6.Rezultate.**

**7.Concluzii.**

**8.Bibliografie.**

**1.Problema si solutia problemei:**

In cadrul temei 4 trebuie sa proiectam si sa implementam un sustem de gestionare a livrarii alimentelor pentru o companie de caterig. Clientul poate comanda produse din meniul companiei. Sistemul ar trebuie sa aiba trei tipuri de utilizatori care se pot conecta utilizand un nume de utilizator si o parola: administratorul, angajatul obisnuit si clientul.

Administratorul poate avea urmatoarele atribute:

1.Poate importa setul in initial de produse care vor completa meniul dintr-un fisier .csv.

2.Se ocupa de gestionarea produselor din meniu: adauga/ sterge/ modifica produse/ si creeaza produse noi compuse din mai multe produse.

3.Genereaza rapoarte despre comenzile efectuate luand in considerare anumite criterii:

a)Ce order-uri s-au facut intr-un anumit interval dat.

b)Produsele care au fost comandate de multe ori decat un anumit numar specificat.

c)Clientii care au comandat mai mult decat un numar specificat de ori sau care au platit o sa suma mai mare.

d)Produsele care au fost comandate intr-o anumita zi si de cate ori au fost comandate.

Clientul poate sa:

1.Inregistra si utiliza numele de utilizator si parola inregistrate pentru a se putea conecta in sistem.

2.Vizualiza lista de produse din meniu.

3.Cauta produse bazate pe unul sau mai multe criterii, cum ar fi cuvinte cheie(„supa”), evalua, numar de calorii/ proteine/grasimi/ sodiu/ pret.

4.Sa creeze o comanda formata din mai multe produse, iar pentru fiecare comanda data si ora vor fi fixate, si de asemenea se va genera o factura care va enumera produsele comandate si pretul total al comenzii.

Angajatul este anuntat de fiecare data cand o noua comanda este efectutata de un client, astfel incat sa poata pregati livrarea produselor comandate.

Pentru solutionare problemei o sa facem trei „ferestre”, una pentru administrator, una pentru client, si una pentru employees care o sa aiba un punct comun, si anume „Delivery Service”.

**2.Obiectivul temei:**

Obiectivul acestei teme este sa facem o aplicatie pentru mancare, in care sa avem un administrator care se va ocupa de parte de organizare si de decizii importante, angajatii care preiau comenzi si se ocupa de a indeplinii acele comenzi, si mai avem clientii care cu ajutorul unei aplicatii isi aleg mancarea dorita si o pot comanda.

**3.** **Analiza problemei, modelare, scenarii, cazuri de utilizare:**

Programul FOOD DELIVERY MANAGEMENT SYSTEM este folosit pentru a realiza o aplicatie tip restaurant unde clientii pot comanda orice mancare isi doresc fara probleme si angajatii din spatele acestei aplicatii preiau acele comenzii pentru a le realiza.

Problema a fost facuta in mare parte dupa exemplul alaturat prezentarii temei, astfel incat avem administratorul care se ocupa de gestionare, angajatii care se ocupa de preluarea si procesarea comenilor si clintii care realizeaza comenzile doretie.

**4.Proiectare (decizii de proiectare, diagrame UML, structuride date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator):**

Voi incepe pentru inceput cu o descriere simpla a interfetei. In cazul acestei teme interfata a fost mai greu de realizat, ea a necesitat destul de multe butoane si in acelasi timp a fost nevoie sa fac doua ferestre, una pentru admin si una pentru client, plu cea de logare. Am facut ceva usor de inteles si de introdus date. Interfata am realizat-o in javaFx. Nu este ceva complicat, este foarte usor de utilizat. Se introduc mai exact date de la tastatura, in prima ferestra avem datele de conectare pentru adminstrator si client, alturi de o parola dupa care apasam „Login” pentru a putea accesa. Aici se vor deschide doua ferestre bazate pe cele doua criterii si in functie de acest lucru avem casetele si butoanele pentru a da o comanda sau in cazul administratorului a importa, sa a gestiona(adauga, sterge, modifica produse, pentru a crea alte produse, etc), genera rapoarte. In cazul clientului avem logarea lui, vizualizarea listelor, cautarea produselor dupa criterii, pentru crearea de comenzi, etc.

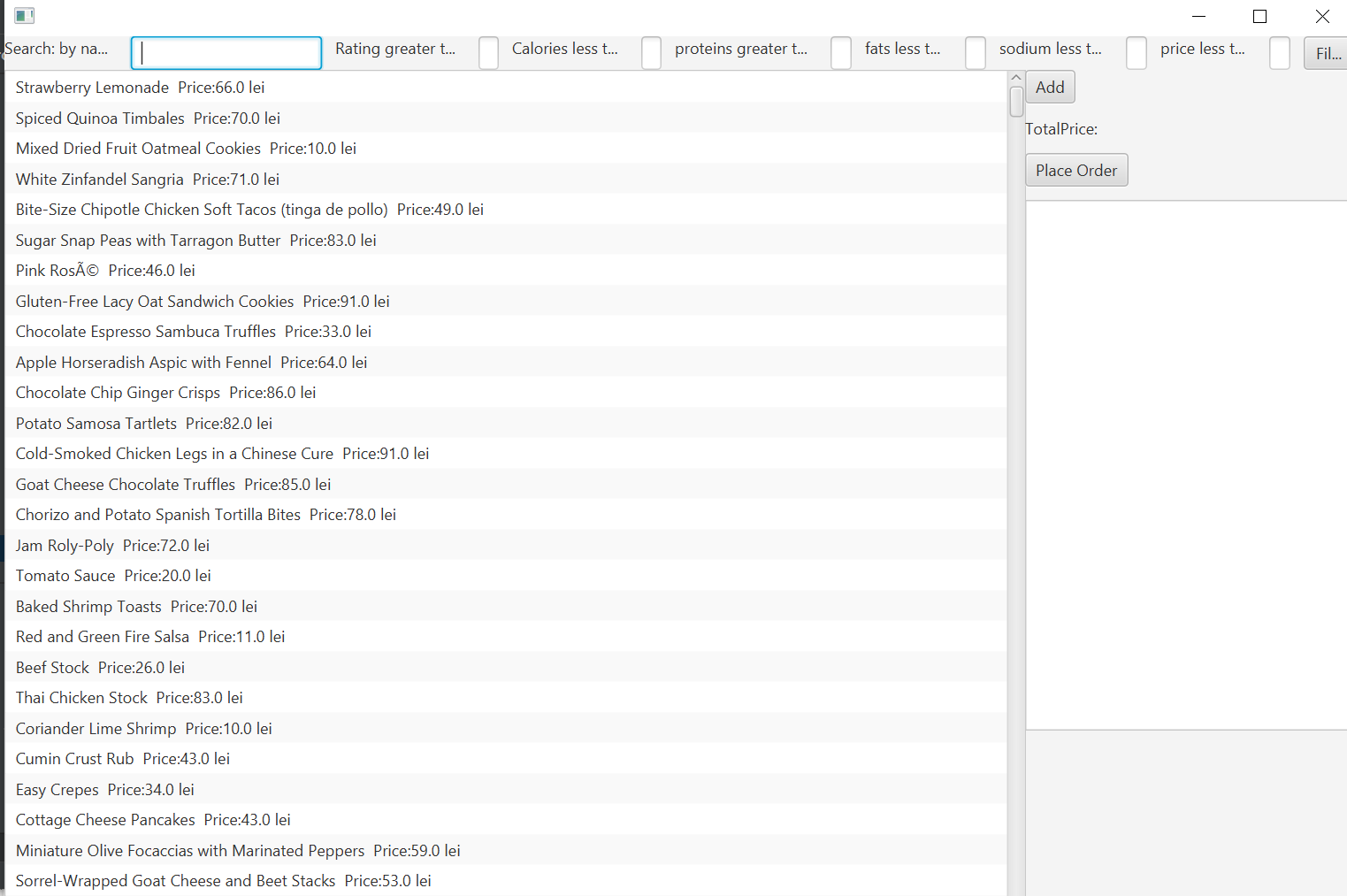
**Interfata:**

**O imagine care conține text

Descriere generată automat**

**O imagine care conține text

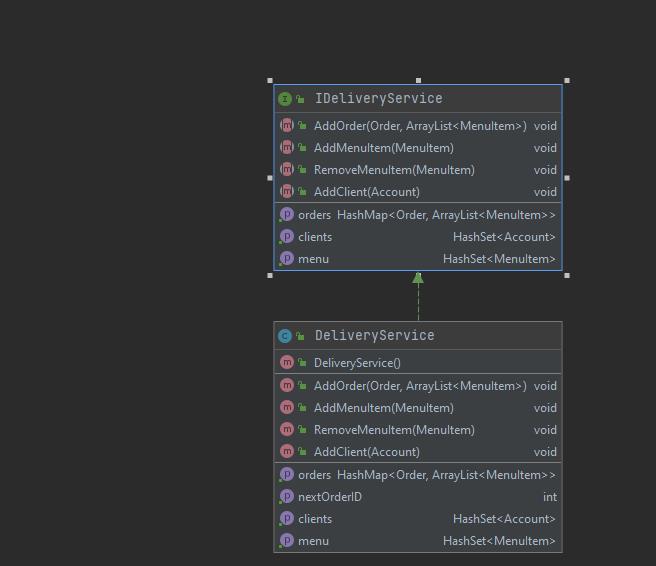
Descriere generată automat**

****

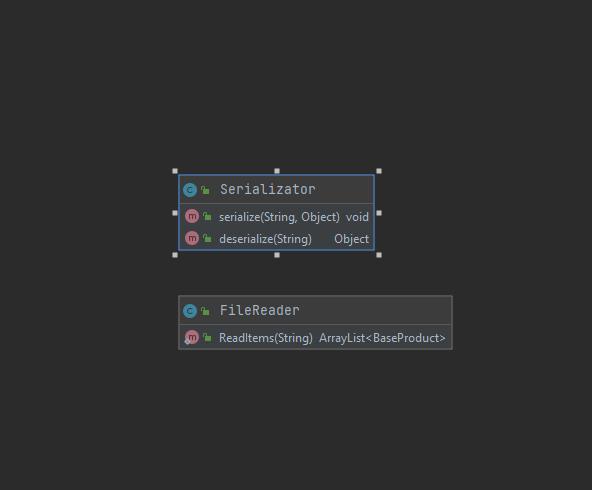
Unified Modeling Language (adica prescurtat UML) este un limbaj standard pentru descrierea de [modele](https://ro.wikipedia.org/w/index.php?title=Modelare_orientat%C4%83_pe_obiect&action=edit&redlink=1) și [specificații](https://ro.wikipedia.org/w/index.php?title=Specifica%C8%9Bie&action=edit&redlink=1) pentru [software](https://ro.wikipedia.org/wiki/Software). UML are la baza dezvoltarea pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora (numite și obiecte).In cazul cestui proiect am folosit diagrama te tip clase, diagramele de clasă sunt folosite pentru reprezentarea concretă a unor instanțe de clasă, așadar obiecte, și a legăturilor concrete dintre acestea.

Am generat o diagrama pentru fiecare pachet in functie legaturile pe care le are.

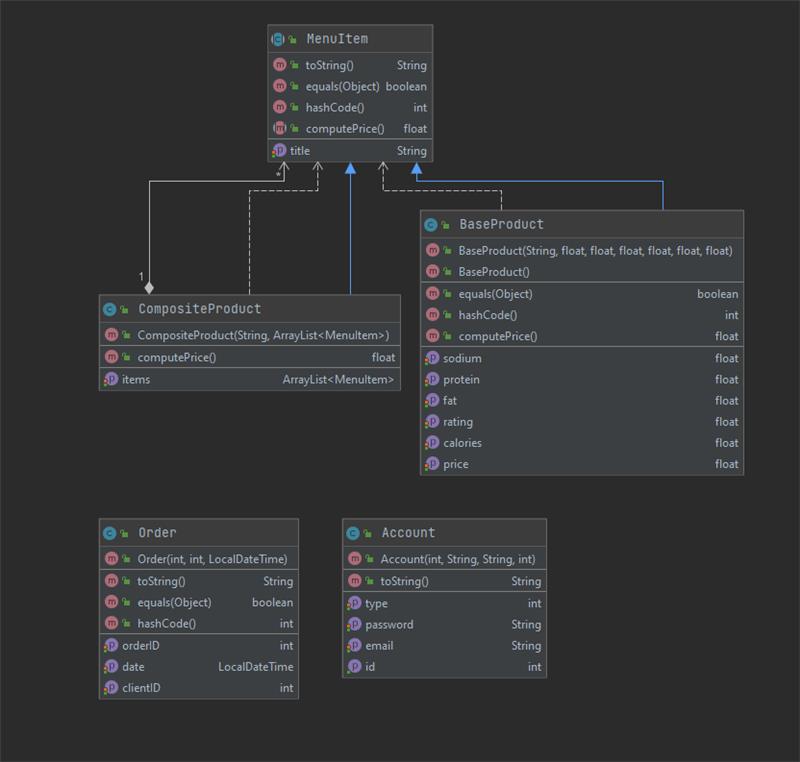
Avem mai jos pachetului BusinessLogic.

****

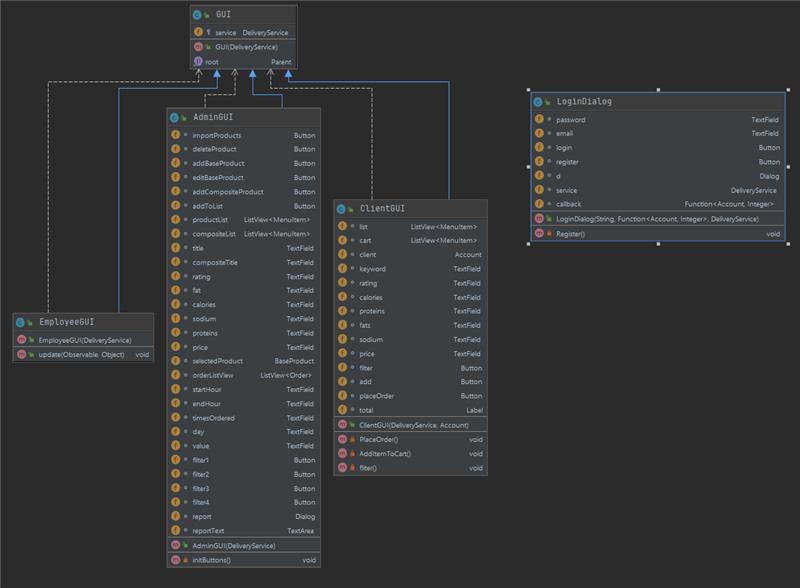
Dupa care am general pachetul DataAccess.

****

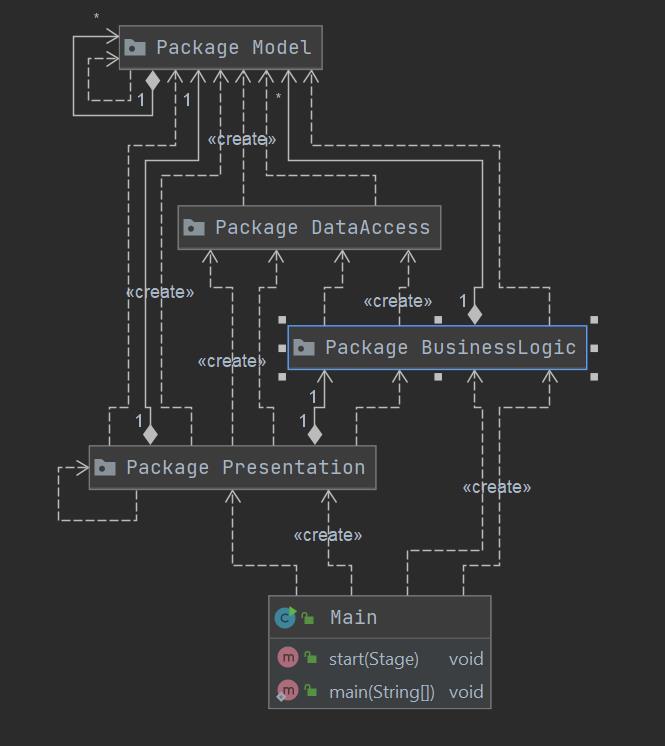
Urmat de pachetul Model.



Urmat de pachetul Presentation.



In final am generat pachetul foodDelivery care contine pachetul BusinessLogic, pachetul DataAccess, pachetul Model, pachetul Presentation si Main-ul.



**5.Implementare:**

In acest proiect am inceput cu o clasa Main, aici mi-am creat un loginDialog ca ia ca paramentru o metoda, iar aceasta metoda va deschide o fereastra in functie de ce user s-a logat. Daca user-ul ii de tip „0”, adica administratorul atunci se va deschide fereastra de admin, daca-i de tip „2” se va deschide fereastra de client, iar daca-i de tip „1” va fi deschisa fereastra de employee. Aceste ferestre inheriteaza din clasa GUI, in aceasta clasa am facut un root, unde se va „injecta” service-ul, adica primeste ca argument service-ul, si anume deliveryService-ul din Main si-l va pune ca un membru in GUI pentru a putea avea acces la el de peste tot.

Pachetul BusinessLogic:

In clasa IDDeliveryService am implementat interfata de deliveryService care expune getClients, getOrders, getMenu, acestea fiind cele trei tipuri de date care au fost necesare, conturile, order-urile si menu item-urile. Prcatic aceasta clasa manage-uieste aceste lucruri.

In clasa DeliveryService, atunci cand aplicatia este deschisa, DeliveryService va fi instantat si toate datele vor fi citite din fisierele noastre, pentru ca a fost serializat in acele fisiere inainte, deci vor fi citite. Dupa ce le-am citit, am facut metode care expun aceste date citite, practic in care expun clientii, order-urile si functiile de add, adica addClient, RemoveMenuItem, addMenuItem, etc. Acest lucru doar ne face treaba mai usoara cand vine vorba de pastrarea datelor.

Pachetul Model:

In clasa Account avem un cont care are un eml, o parola, un tip si un id.

In clasa Order avem un order, care are un order id, un client id, adica acel id dat clientului care a dat acea comanda, si data la care a fost facut.

In clasa MenuItem, aceasta are doua proprietati main, un titlu si o functie de compute price. Aceasta clasa este practic inheritat de clasele de mai jos, adica de progeniturile sale:

Si anume, vom avea BaseProduct care defineste rating, calories, fat, protein, sodium, price, pe langa cele deja definite in MenuItem, aceasta clasa este compusa din alte MenuItem-uri. Din cauza ca ii un menuItem, el poate fi tratat evident ca un alt menuItem, dar din motiv ca are mai multe iteme inautru il va face diferit, acest lucru reprezinta de fapt compositDesign. El poate fi tratat ca orice alt produs, doar ca este de fapt un set de produse.

Pachetul Presentation:

Avem in acest pachet clasa AdminGUI, care contine toate elementele din interfata data. Atunci cand initializam adminGUI setam de fapt Dialog-ul de rapoarte. Dupa ce facem asta vom intitializa functionalitatea butoanelor si le vom aranja in pagina. Dupa care am declarat texte, pe care le-am pus in pagina si am configurat listele(productList, orderListview, compositeList, orderListview). Dupa care mi-am setat filed-urile pentru input, atunci cand trebuie sa modificam sau trebuie sa adaugam produsele, le-am configurat si setat.

Dupa care mai am un productList, care este de fapt lista de produse, practic configuram ca atunci cand vom selecta unul dintre produse din ProductList, atunci atributele produsului sa ni se puna. Iar in functie de ce este acel produsul, fie ca este baseproduct, fie ca este compositeproduct o sa le punem in partea lui specifica. In metoda initButtons, in importProducts pur si simpu vom zice service.getMenu.addAll de fileReader de ReadItems si dupa care ne va parsa din products.csv un produs si ne va genera inteme de acolo. Metoda addBaseProduct va face exact ce zice si numele sau, practic va lua dint text Filed-uri ce i-am bagat noi si va adauga un nou baseproduct. In metoda editBaseProduct, din nou, face exact ce sugereaza numele, o sa editeze un base product. In metoda CompositeProduct, o sa ia toate produsele composite adaugate in lista noastra si ne va crea un compositeproduct, dupa care il va baga din nou in lista noastra de produse, ca un meniu de exemplu. Metoda deleteProduct face exact ce sugereaza numele, o sa stearga un base product.

Acum o sa descriu partea de filtere. Filter 1, se va duce prin toate order-urile noastre si vede care dintre order-urile noastre se afla in intervalul de timp specificat din textBoxurile date si le pune in report.show.

Filter 2, se va uita print toate produsele noastre si va vedea care dintre aceste produse au fost order-uite de mai mult decat un timp specificat, adica se duce prin fiecare produs si vede de cate ori a fost order-uit, si dupa compara acel numar cu numarul specificat si dupa da report.show.

Filter 3, aici vrem toti clientii care au comandat mai mult decat un numar specificat de ori, si a carui valoare e mai mare decat un numar specificat. Asa ca am mers prin toti clientii, o sa vedem care sunt de fapt clientii, adica mergem prin toate countr-urile,si dupa pentru fiecare client o sa vedem de cate ori a fost comandat(adica ne ia toate order-urile care le-a comandat clientul nostru) si dupa din acele order-uri le filtreaza in functie de cati bani a dat pe acel order si dupa le numara. Daca numarul lor este mai mare decat un numar specificat atunci ii ok, merge mai departe si dupa le colecteaza si afiseaza intr-o lista.

Filter 4, aici avem nevoie de produsele order-uite, intr-o zi specificata si de cate ori au fost order-uite. De data aceasta se cer doua date, produsul si de cate ori. Asa ca, in primul rand ne ducem prin meniu si pentru fiecare produs vrem sa luam doar produsele care au fost order-uite in ziua „day”, care match-uiesc produsul nostru. Odata ce avem produsele care indeplinesc acest numar, vrem de fapt sa atasam la datele astea si de cate ori au fost comandate. Asa ca o sa map-uim fiecare produs la un string, care ne va zice numele produsului si de cate ori a fost comandat. O sa mergem prin orders, o sa facem un join si o sa numaram cate produse au fost comandate. Dupa atasam totul cu raport.show.

In clasa clientGUI, avem textField-uri si butoane care ne lasa sa adaugam ordere. Avem layout-ul si le punem in pagina. Dupa care adaugam acea pasrte din stanga care este de fapt cosul de cumparaturi. Cand adaugam ceva, vom lua pur si simplu ce am selectat din lista mare si bagam in lista mica. Cand apelam placeorder se creaza un order nou si la service se adauga acel order impreuna cu itemele din cart, si pe langa asta se genereaza un bill pentru acel order. Am configurat listele si filtere, ne ia toate produsele si le filtreaza in functie de criteriile specificate in textBox-uri. Din toate produsele, ia doar produsele care contin keyword-ul specificat, etc.

In clasa LoginDialog avem text field-uri(password, email) si butoane(login, register) si un dialogpane, si partea de logare.

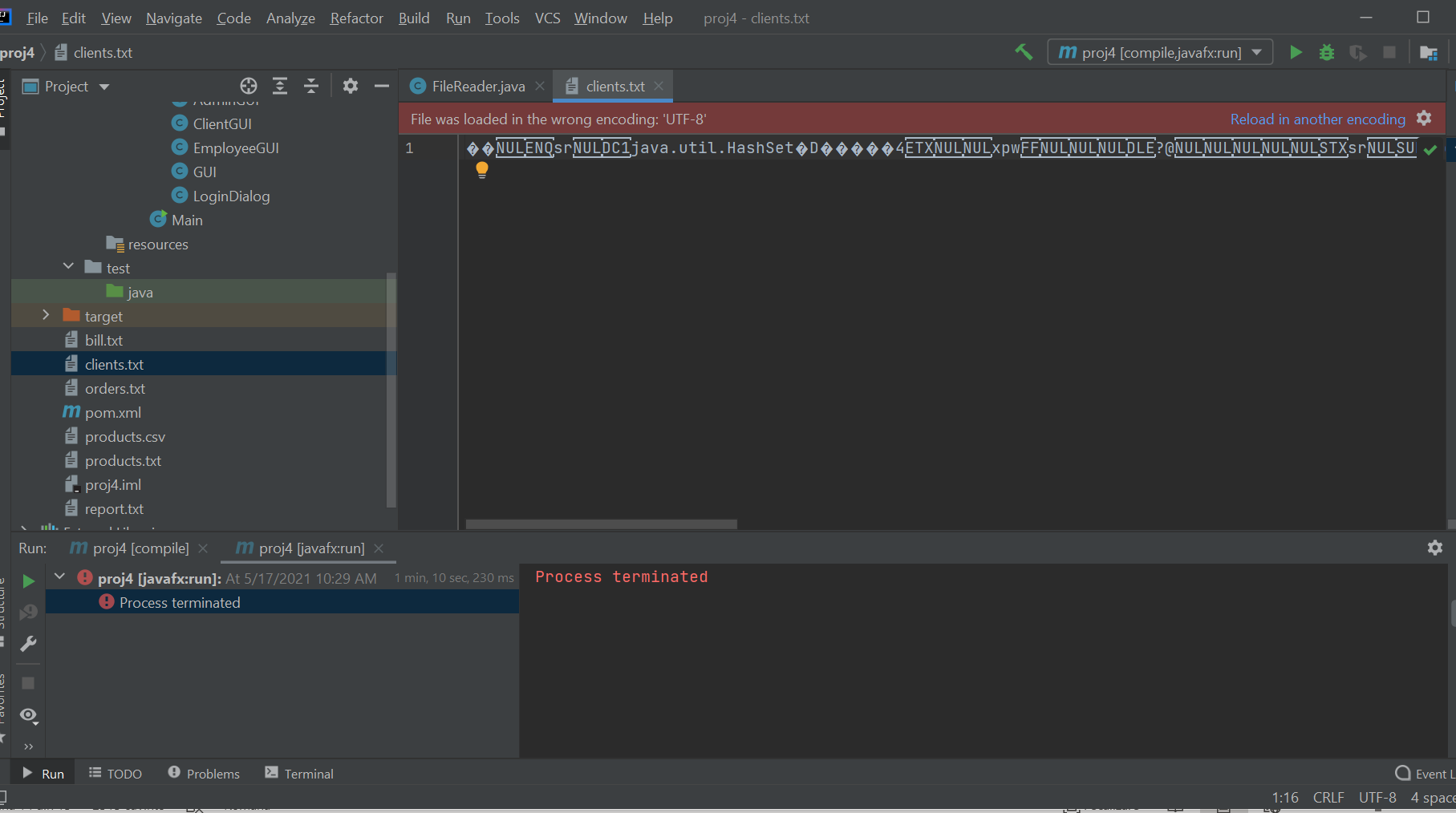
Pachetul DataAccess:

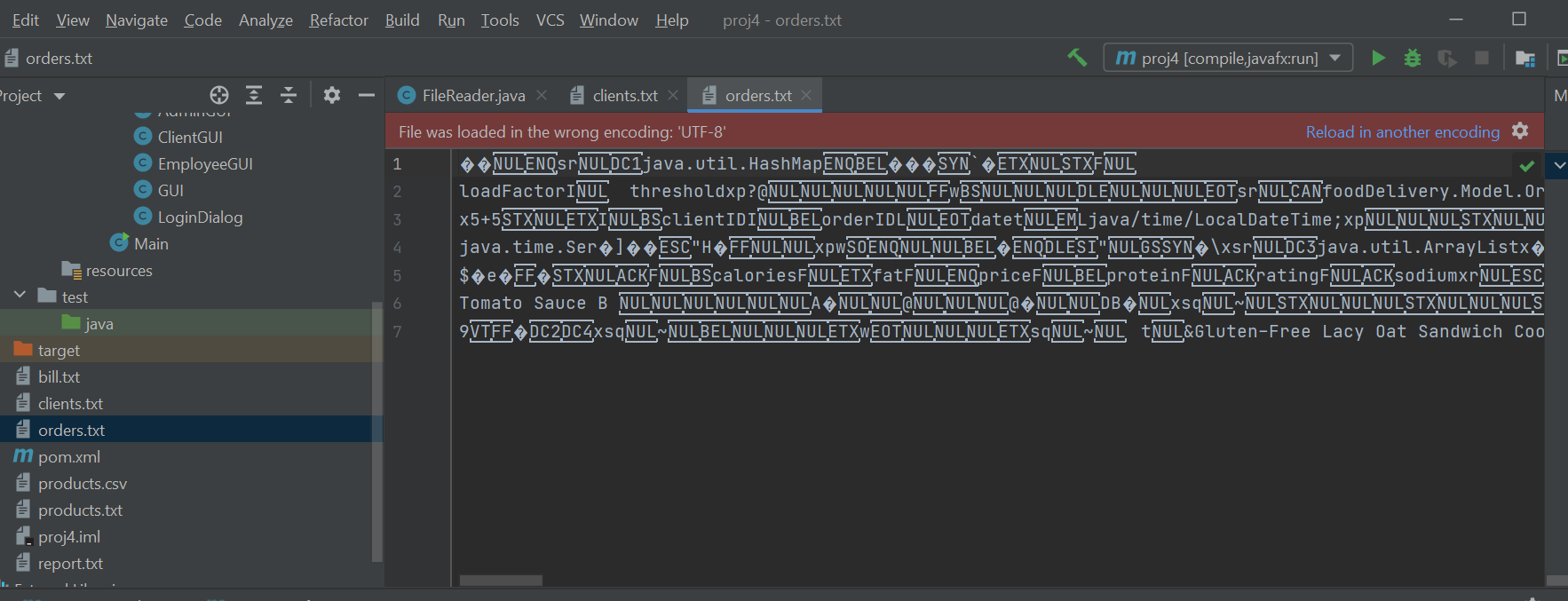
In clasa Serializator avem metoda serialize care are definite fout, file, si deasemenea mai avem deserializare care defineste file si fin.

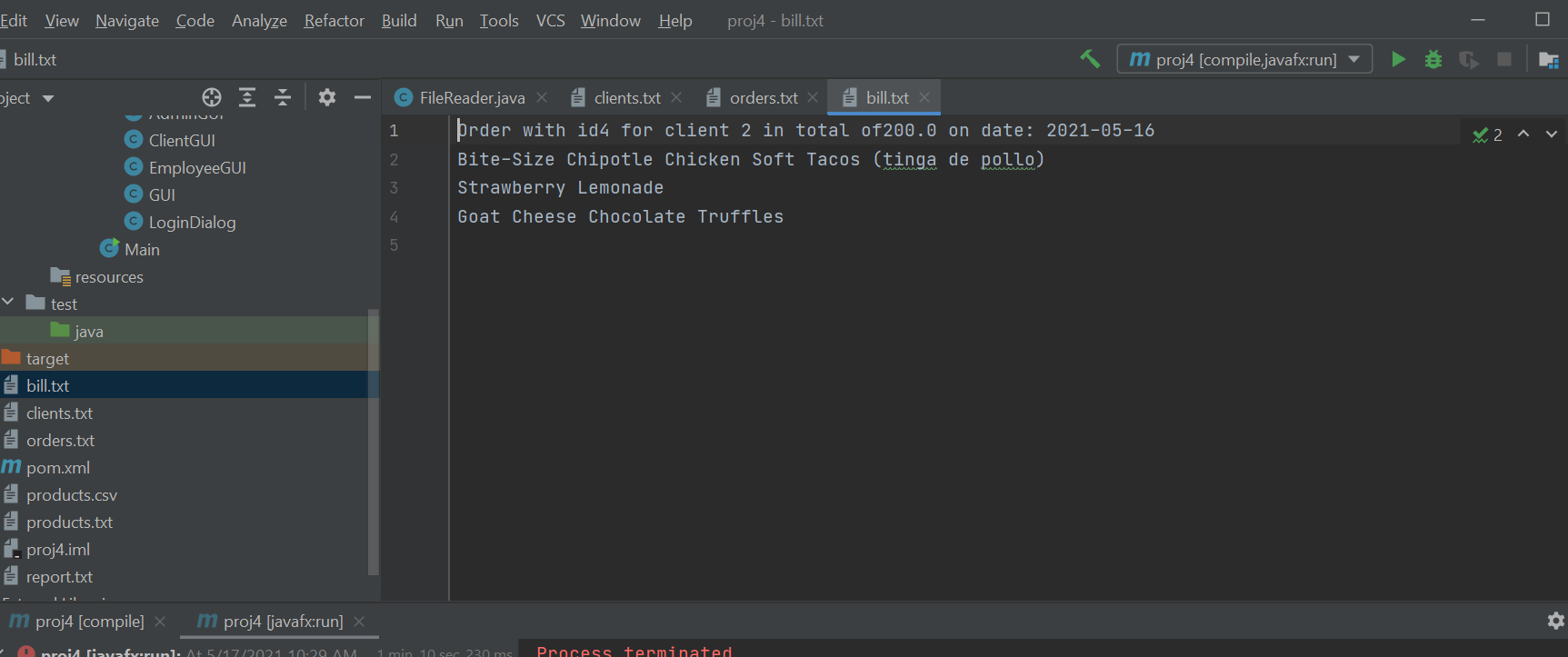
In clasa FileReader avem un arraylist base product, si am facut intr-un try cach o initializare pentru sc si pattern si dupa tot in try cach am facut parsari pentru specificatii.

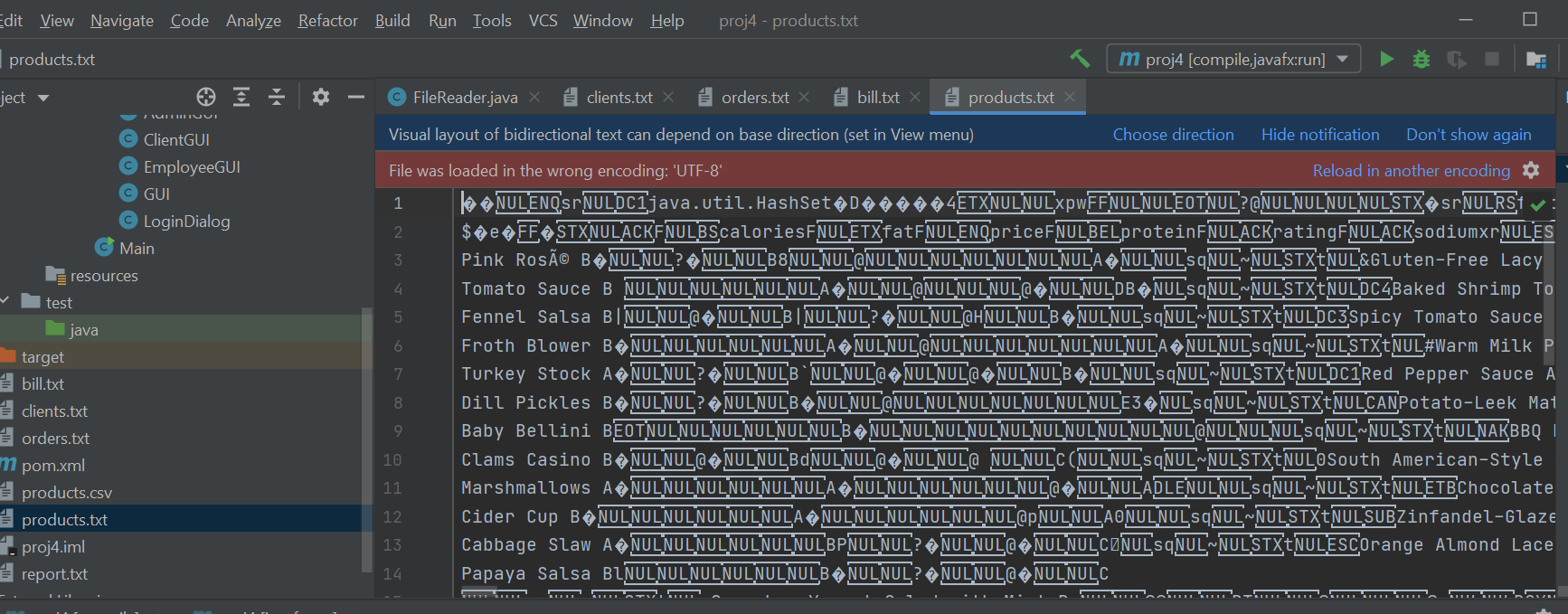
**5.Rezultatele:**

Pentru a verifica daca codul nostru merge ne folosim de mvnjavafx:run pentru a verifica daca ruleaza condului functioneaza, iar dupa ce aceasta s-a realizat cu suucces folosim mvncompile care ne compileaza codul si ne deschide interfata grafica afisand toate datele necesare.









In orders.txt si client.txt avem raportul specific administrator si pentru client avem functionalitatile de cautare disponibile pentru client. Mai am avut de generat si bill intr-un format de tip txt. Si a mai fost necesar sa salvam informatiile din clasa DeliveryService intr-un fisier de tip txt folosind serializarea.

**7.Concluzii:**

In cadrul celui de al patrulea proiect, o concluzie importanta ar fi ca: am realizat inca o data rolul si importanta informaticii in orice domeniu.

De asemenea, acest proiect m-a ajutat sa imi dezvolt mult mai bine cunostiintele in tot ceea ce inseamna limbajul Java, crearea unui program cu ajutorul unei interfete grafice.

Acest proiect a necesitat o documentare foarte amanuntita si lunga deoarece a fost mai complex ca celelalte teme si a fost ceva cu totul nou pentru mine si a trebuit sa caut si pe surse externe de informatie despre acest concept. Intr-un final am reusit sa inteleg oarecum aceasta idee si am reusit o implementez in proiect meu. Ca si dezvoltari ulterioare, consider ca s-ar putea implementa acest algoritm in viata reala, pentru gestionarea comenzilor la un supermarket, la nivelul unui aeroport , fiind implementat o aplicatie automatizata care sa ajute clientii sa economiseasca timp prin comenzi.

**8.Bibliografie:**

Am folosit wikipedia pentru diferite definitii si informatii pentru interfata acestui proiect.

Deasemenea am mai folosit diverse site-uri de programare (geeksforgeeks sau W3schools) pentru a incerca sa implementez cat mai bine programul meu.

Am folosit javaFX din inteliji, pentru aceasta m-am documentat de pe diferite site-uri, cum ar fi: [https://cs108.epfl.ch/archive/17/c/JAFX/JAFX-notes.html,https://www.jetbrains.com/help/idea/javafx.html, unde](https://cs108.epfl.ch/archive/17/c/JAFX/JAFX-notes.html,https://www.jetbrains.com/help/idea/javafx.html,%20unde) sunt explicate lucruri de baza utile in crearea proiectului si a interfetei. Si desemenea m-am folosit de cateva tutoriale vizualizate pe youtube unde explica modul in care putem crea o interfata mai complexa.