****

**TEHNICI DE PROGRAMARE**

**ASSIGNMENT 4**

**FOOD DELIVERY MANAGEMENT SYSTEM**

**Chelemen Stefana-Bianca**

**GRUPA 30221**

**1.Obiectivul temei**

Principalul obiectiv al acestei teme este implementarea unei aplicatii care simuleaza un sistem de livrari de mancare . Aceasta aplicatie are o parte care administreaza sistemul ( adauga produse in meniu , le editeaza sau le sterge ) , o parte care se ocupa cu comenzile ( angajatul ) si partea de client care pot face comenzi .

Ca obiective secundare avem :

* Utilizarea unui “ design patterns “ din Java ( “ Composite “ si “ Observer “ ) ;
* Utilizarea unor colectii predefinite in implementarea sistemului de livrari ( “ HashMap “ ) ;
* Serializarea si deserializarea datelor .

**2.Analiza problemei,modelare,scenarii,cazuri de utilizare**

***2.1.Analiza problemei***

Principalul scop este de a dezvolta un program ce implementeaza toate cerintele cerute , si , in acelasi timp , foloseste paradigmenele Programarii Orientate pe Obiect . Pentru a face asta , avem nevoie sa gasim un mod sa ne stocam informatiile , si pentru asta , alegem “ HashSet “ pentru a retine meniului sistemului , o lista cu toate comenzile effectuate , si un “ HashMap “ , care are ca si cheie comanda , si ca valoare , o lista de “ MenuItems “ . Oricum , adevarata provocare este sa implementam un anumit “ desing pattern “ , si sa serializem sis a deserializam obiectele noastre , pentru a nu pierde datele cand aplicatia este inchisa .

Pentru ca programul sa ruleze corect , atunci cand un utilizator ( fie el administrator , client sau angajat ) incearca sa se conecteze la sistemul de livrari , acesta trebuie sa introduca un username valid si o parola valida . In cazul in care un client nu are cont , acesta se poate folosi de butonul ‘ REGISTER ‘ care il va duce sper un nou frame unde acesta isi va putea crea un cont nou.

* Cazuri de utilizare

Atunci cand se apasa butonu ‘Run’, se deschide interfata grafica , care este usor de utilizat. Apar 4 butoane: Administrator , Client , Employee . Utilizatorul va apasa butonul aferent operatiei pe care doreste sa o realizeze si va deschide unul dintre cele 4 frame – uri.

In cadrul frame – ului ‘ Administrator ‘ utilizatorul poate adauga un produs nou , sterge un produs existent , edita detaliile unui produs , poate vedea lista cu toate produsele si poate genera diferite tipuri de rapoarte .

In cadrul frame – ului ‘ Clients ‘ utilizatorul poate vizualiza tabelul de produse , poate cauta produsele dupa anumite criterii sau poate creea o comanda .

In cadrul frame – ului ‘ Employee ‘ utilizatorul va fi informat de fiecare data cand un client plaseaza o comanda pentru a fi preparata si livrata .

Frame – ului ‘ Register ‘ va fi folosit doar de catre clientii care inca nu au un cont in cadrul sistemului si doresc sa isi faca unul .

***2.2.Modelare***

1. **Composite Design Pattern** este un “ design pattern “ de partitionare si descrie un grup de obiecare ce e tratat ca o singura instanta a aceluias tip de obiect . Intentia sa este sa “ compuna “ obiectele in 3 structuri , ce reprezinta ierarhii partiale . Acest lucru ti permite sa ai o structura de tip “ arbore “ , si sa faci fiecare nod din structura arborelui sa efectuezre un anumit task .

Composite Design Patern are 4 participanti :

1. Componenta : Componenta declara interfata pentru obiectele din compozitie si pentru accesul si controlul componentelor-copii . Implementeaza , de asemenea , in mod implicit , si un comportament pentru interfata comuna tuturor claselor , dupa caz .
2. Frunzaele : Frunzele definesc comportamentul pentru obiectele primitive din compozitite .
3. Compozitia : Compozitia stocheaza componentele-copii si implementeaza operatii legate de copii , in interfata de componente .
4. Clientul : Clientul manipuleaza compozitia obiectului prin interfata componentelor .
5. **Observer Pattern** este utilizat cand avem relatii de tip “ one – to – many “ intre obiecte , astfel incat daca un obiect este modificat , obiectele sale dependente sunt informate automat . “ Observer Pattern “ se incardreaza in categoria modelului comportamental .

“Observer Pattern “ utilizeaza 3 clase :” Subject “ , ” Observer “ si “ Client “ . “ Subject “ este un obiect cu metode ce ataseaza si detaseaza observatorii de obiectele “ client “ .

Pentru a ne administra datele in mod eficient , utilizam :

1. **HashSet :** “ HashSet “ implementeaza intergata “ Set “ , sprijinita de un “ hash table “ , care este , de fapt , o instant “ HashMap “ . Nu ofera nicio garantie cu privire la ordinea de interatie a setului , ceea ce inseamna ca , clasa nu garanteaza ordinea constanta a elementelor de-a lungul timpului . Aceasta clasa permite elemenul “ null “ . Clasa poate oferi , de asemenea , performanta timpului constant , pentru operatii de baza , ca “ add ” , “ remove “ , “ contains “ and “ size “ , presupunand ca functia “ hash “ disperseaza elementele corespunzator .

2. **HashMap** : “ HashMap “ asigura implementarile de baza ale intefetei “ Map “ . Stocheaza informatiile in perechi ( Key , Value ) . Pentru a accesa o valoare , trebuie sa ii cunoastem cheia . “ HashMap “ este cunoscuta ca “ HashMap “ , deoarece utilizeaza o tehnica numita “ Hashing “ . “ Hashing “ este o tehnica de convertire a unui String mare , intr-un string mic , ce reprezinta de fapt acelasi String . O valoare mai mica ajuta in indexarea si cautarea mai rapida . “ HashSet “ utilizeaza intern “ HashMap “ . Intern , foloseste un “ link list “ pentru a stoca perechi “ key - value “ .

Pentru a ne pastra datele dupa ce aplicatia se inchide , folosim “ serializarea “ :

Java ofera un mecanism , numit “ object serialization “ , unde un obiect poate fi reprezentat ca o secventa de biti , ce include informatiile obiectului , precum si informatii despre tipul obiectului si tipurile de date stocate in obiect .

Dupa ce un obiect serializat a fost scris intr-un fisier , el poate fi citit din fisier si deserializat , ceea ce reprezinta , de fapt , tipul informatiei , si bitii ce reprezinta obiectul si informatiile lui pot fi utilizate pentru a crea obiectul in memorie .

***2.3.Scenariul***

Pentru ca programul sa ruleze corect , atunci cand un utilizator ( fie el administrator , client sau angajat ) incearca sa se conecteze la sistemul de livrari , acesta trebuie sa introduca un username valid si o parola valida . In cazul in care un client nu are cont , acesta se poate folosi de butonul ‘ REGISTER ‘ care il va duce sper un nou frame unde acesta isi va putea crea un cont nou.

***2.4.Cazuri de utilizare***

Atunci cand se apasa butonu ‘Run’, se deschide interfata grafica , care este usor de utilizat. Apar 4 butoane: Administrator , Client , Employee . Utilizatorul va apasa butonul aferent operatiei pe care doreste sa o realizeze si va deschide unul dintre cele 4 frame – uri.

In cadrul frame – ului ‘ Administrator ‘ utilizatorul poate adauga un produs nou , sterge un produs existent , edita detaliile unui produs , poate vedea lista cu toate produsele si poate genera diferite tipuri de rapoarte .

In cadrul frame – ului ‘ Clients ‘ utilizatorul poate vizualiza tabelul de produse , poate cauta produsele dupa anumite criterii sau poate creea o comanda .

In cadrul frame – ului ‘ Employee ‘ utilizatorul va fi informat de fiecare data cand un client plaseaza o comanda pentru a fi preparata si livrata .

Frame – ului ‘ Register ‘ va fi folosit doar de catre clientii care inca nu au un cont in cadrul sistemului si doresc sa isi faca unul .

**3.Proiectare**

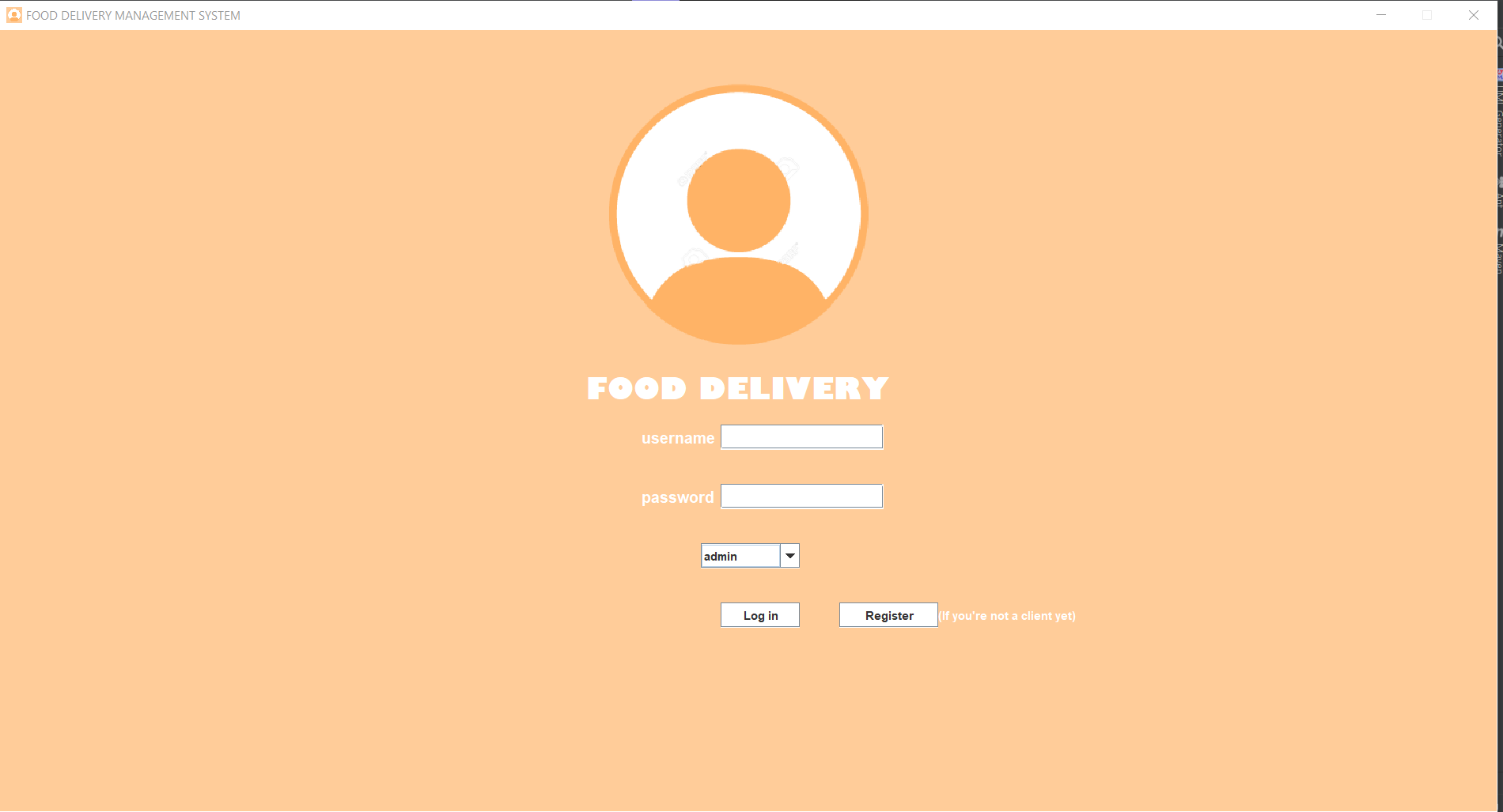
Pentru a respecta paradigmele POO , programul a fost impartit in mai multe pachete cu nume sugestive care contin diferite clase :

* Pachetul ‘ businessLayer ’ care contine clasele : BaseProduct , CompositeProduct , DeliveryService, MenuItem, Order si interfata IDeliveryService ;
* Pachetul ‘ dataLayer ‘ care contine clasele : DeliverySerializator , FileWriter ;
* Pachetul ‘ presentation ‘ care contine clasele : AdministratorView , ClientView , EmployeeView , RegisterFrame , View ;
* Pachetul ‘ start ‘ care contine clasa : Start .

Interfata grafica cuprinde urmatoarele elemente:

* Frame *=* “rama” in care se adauga toate elementele de care avem nevoie pentru buna functionare a programului . E practice o fereastra care , dupa utilizare , apasand butonul ‘X’ din dreapta sus se poate inchide ( Exit\_on\_close ) ;
* Butoane – avem 3 butoane in frame – ul principal fiecare cu alta functionalitate :
* CLIENTS – deschide frame - ul pentru client care contine la randul lui alte butoane cu operatiile care se pot realiza de catre clientii sistemului de livrari ;
* ADMINISTRATOR – deschide frame – ul pentru administrator care contine la randul lui alte butoane cu operatiile care se pot realiza de catre administratorul sistemului de livrari ;
* EMPLOYEE – deschide frame – ul pentru angajati care contine la randul lui alte butoane cu operatii care se pot realiza de catre angajatii sistemului de livrari ;
* TextField = spatii dreptunghiulare in care se pot introduce date de la tastatura , dar pe langa asta pot fi folosite si pentru a afisa rezultatul fara a se putea introduce date de la tastatura .
* Label = o etichetă , care poate fi titlul , informații , indicații etc . , care pot ajuta utilizatorul să folosească progamul. In cazul nostru e folosit doar pentru afisarea informatiilor aflate deasupra casetelor de text .

**5.Rezultate**

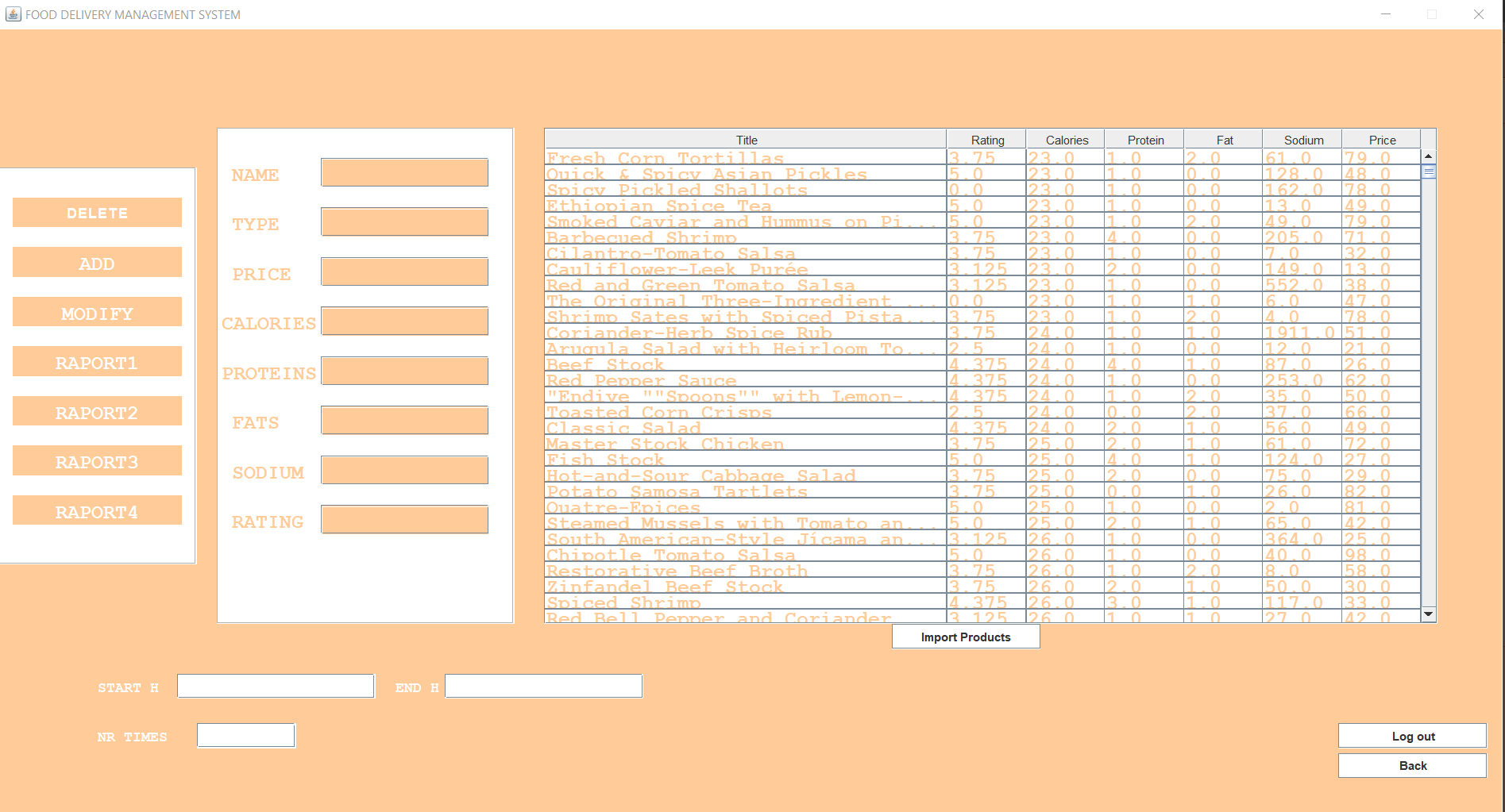
****

Aceasta este fereastra de login. Utilizatorul poate alege sa introduca usernamul si aprola, si dintr-un ComboBox poate alege ce tip de utilizator este: admin/ client/employee. Daca nu are un cont, utilizatorul de tip Clinet isi poate creea unul, apasand butonul Register.

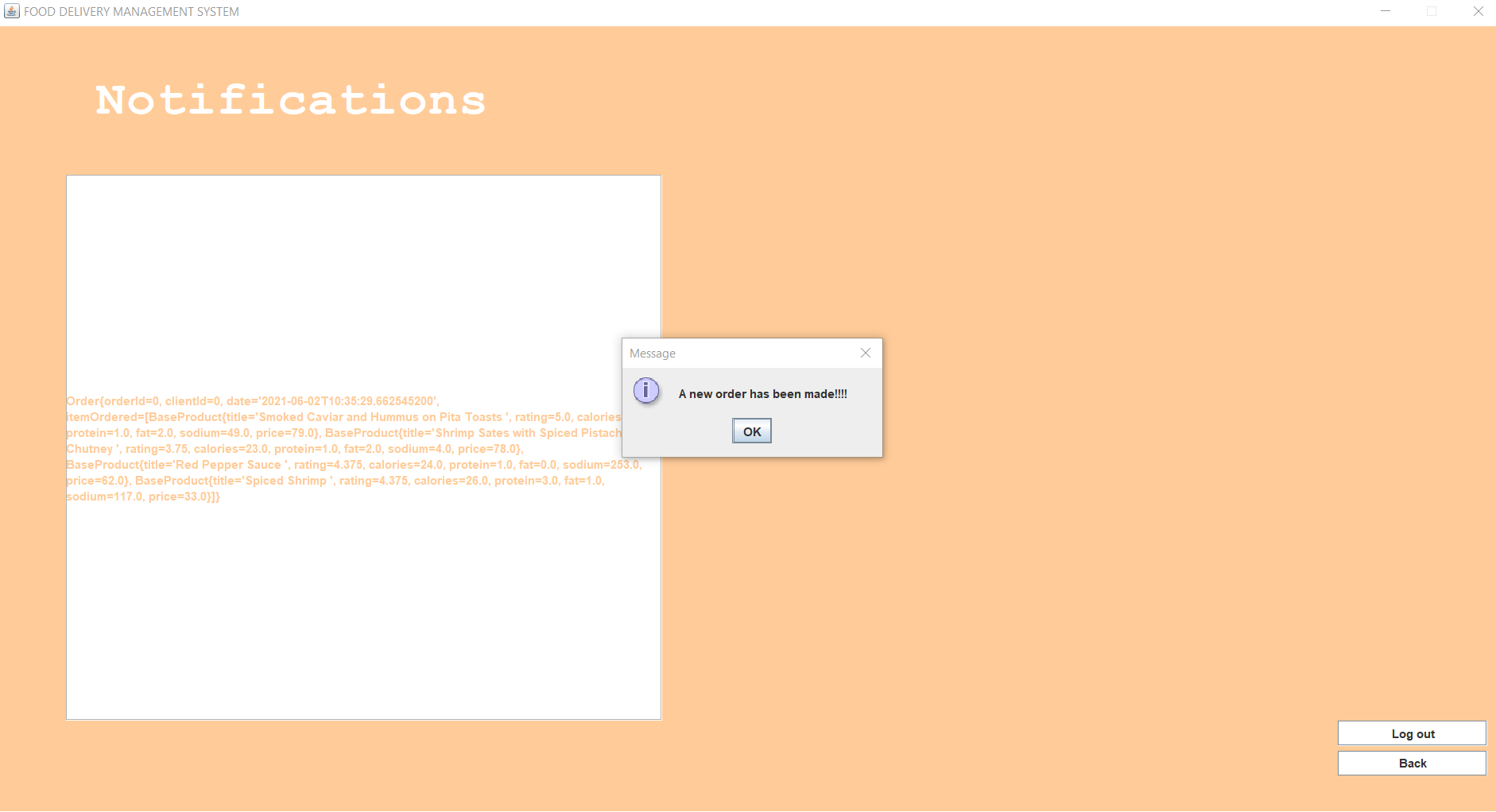
Daca utilizatorul apasa butonul “Register”, va aparea urmatoarea interfata grafica:



In aceasta fereastra. Utilizatorul trebuie sa completeze campurile: “First Name”, “Last Name” , ”Adress” , “Tell” , “Email” , “Username” , si “Password” , iar apoi poate sa apese butonul “Create Account” pentru a se putea loga. Daca a introdus toate datele corecte, atunci utilizatorul va fi trimis la pagina de login si va putea sa intre in fereastra de Client.

****

Aceasta este fereastra de admin. Din acesta fereastra se poate adauga un produs nou, se poate modifica un produs, se poate sterge un produs si se pot genera anumite rapoarte legate de comenziile facute pana in momentul respectiv. Apasand butonul “Import Products”, in table vor aparea produsele citite din sisierul products.csv.

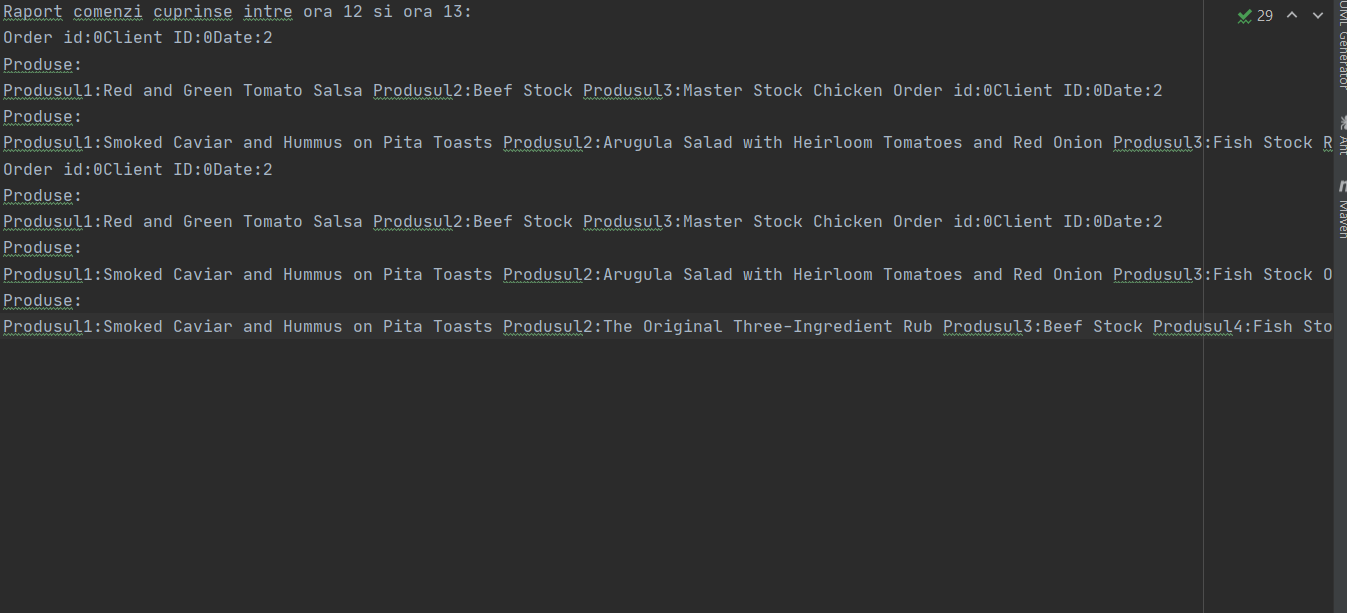
****

Aceatsa este fereastra de Employee, fereastra care are un panel in care apar notificari in momnetul in care se afce o comanda noua. Atunci cand un client introduce o noua comnada, se va genera un Bill cu detaliile comenzii si pretul total, intr-un format txt, si va aparea un JOptionPane cu un mesaj sugestiv, in care suntem informati ca s-a facut o comanda noua.

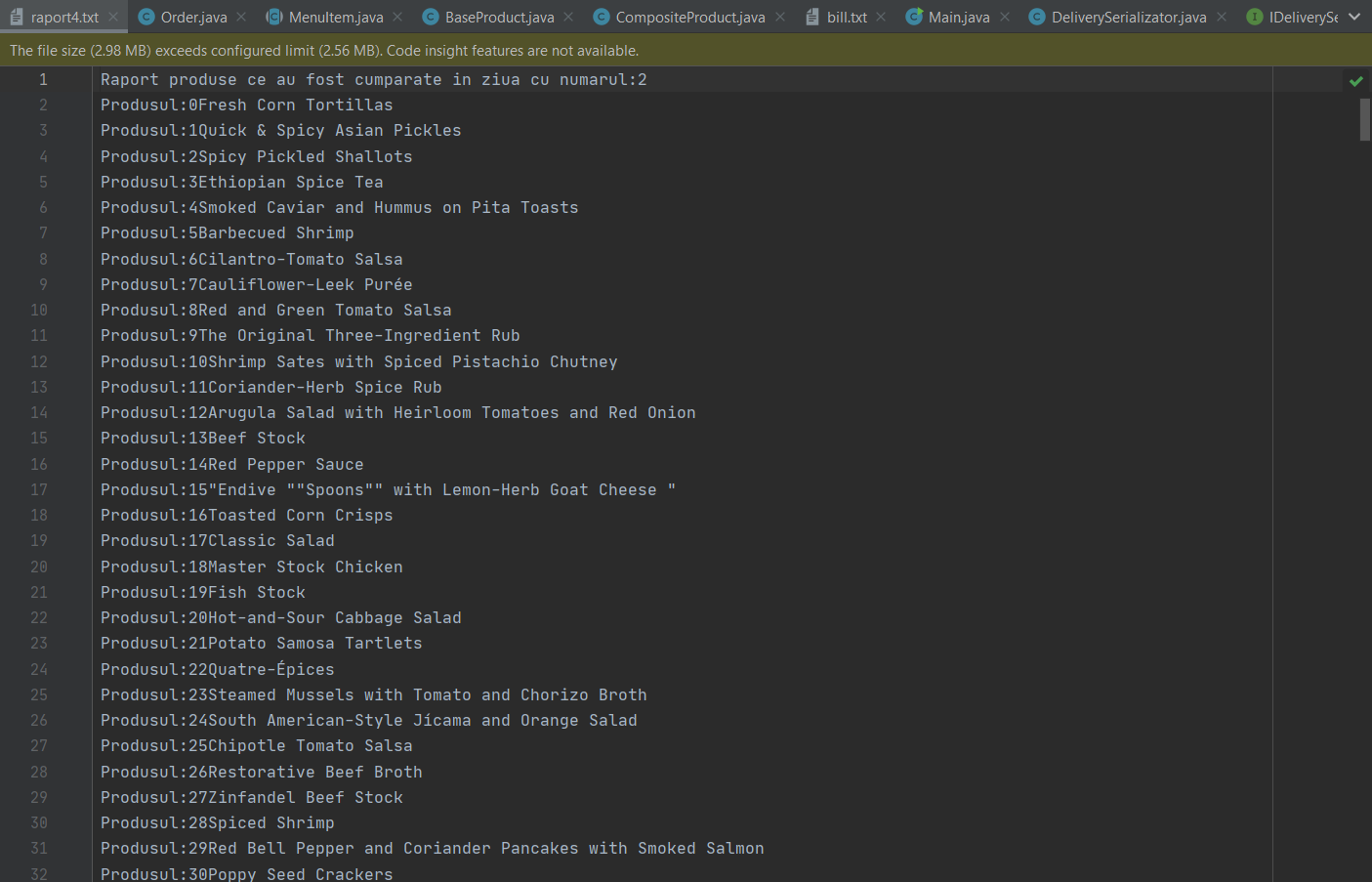
In fereastra de Admin, adminul poate genera rapoarte cu privire la comenzile facute pana in momentul respectiv.

Exista 4 butoane in functie de ce raport dorim sa generam.

De exemplu, daca dorim sa afisam ce comenzi au fost facute in data de “2021-06-02 12:00” si “2021-06-02 13:00”, in fisierul raport1.txt se va genera urmatorul raport:



Sau, daca dorim sa vedem ce comenzi au fost facute intr-o anumita zi a lunii, cum ar fi in ziua 02 din luna a 6-a a anului 2021 vom complete in casuta “Time” din AdminFrame “2021-06-02” si vom apasa butonul “Raport4”. Se va genera fisierul “raort4.txt”:



**6.Bibliografie** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1. <http://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html>
2. <http://javarevisited.blogspot.ro/2012/01/what-is-assertion-in-java-java.html>
3. <http://stackoverflow.com/questions/11415160/how-to-enable-the-java-keywordassert-in-eclipse-program-wise>
4. <https://intellij-support.jetbrains.com/hc/en-us/community/posts/207014815-How-toenable-assert>
5. <https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html#tag>