Universitatea Tehnică din Cluj-Napoca

Facultatea de Automatică și Calculatoare

Sistem de administrare al unui restaurant

Matei Anechitei

Grupa 30221

1.Obiectivul temei

Implementati un sistem de administrare al unui restaurant.

Detalii:

* Sistemul trebuie sa contina 3 tipuri de utilizatori: administrator, ospatar, bucatar.
* Administratorul poate sa adauge, sa elimine si sa modifice produse deja existente din meniu.
* Ospatarul poate crea o noua comanda pentru o masa, sa adauge elemente din meniu si sa faca nota de plata.
* Bucatarul trebuie sa fie notificat de fiecare data cand trebuie sa gateasca ceva ce e comandat printr-un ospatar.

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

In rezolvarea problemei am folosit conceptul de serializare si deserializare pentru a importa si exporta in cadrul aplicatiei datele pe care le avem deja in restaurant; am folosit HashMap pentru a stoca comenzile, java.swing pentru realizarea celor 3 interfete grafice, Composite Design Pattern si Observer Design Pattern. Am presupus ca exista un singur administrator, un singur ospatar si un singur bucatar, astfel toate cele 3 ferestre au fost deschise simultan si nu exista fereastra de login.

3.Proiectare

In proiectarea aplicatiei am ales sa folosesc colectii predefinite in Java, precum HashMap si ArrayList, dar de remarcat ca majoritatea elementelor din diferitele meniuri accesibile utilizatorilor au fost implementate cu ajutorul DefaultListModel si JList-urilor. Interfetele sunt implementate pe baza MVC, dar clasa model lipseste, fiind inlocuita implicit de Restaurant.

Aplicatia contine 4 pachete: start, data, business, presentation.

La rularea aplicatiei, restaurantul este populat cu datele dintr-un fisier primit ca argument, iar la inchiderea aplicatiei se genereaza automat un nou fisier numit “restaurant.ser” in care sunt salvate toate datele restaurantului.

Rularea aplicatiei se poate face din IDE, setand args[0] la numele fisierului de intrare corespunzator, dar exista un fisier .jar care poate fi rulat din linia de comanda conform sintaxei:



4.Implementare

In cele ce urmeaza voi prezenta pachetele, clasele si metodele importante din acestea.

Start.start

Clasa Start.start are rolul de main, folosing argumentul primit pentru a crea un obiect de tip Restaurant, si a-l trimite catre toate interfetele si controller-ele.

Data.RestaurantSerializator

Clasa Data.RestaurantSerializator se ocupa cu trecerea din fisier in obiect Restaurant si invers prin intermediul serializarii si deserializarii.

Business.MenuItem

Aceasta este doar o interfata cu 2 metode, implementata de 2 clase diferite, BaseProduct si CompositeProduct.

Business.BaseProduct

BaseProduct sunt obiectele specifice elementelor de baza din meniu. Acestea sunt caracterizate doar de nume si pret.

Business.CompositeProduct

CompositeProduct sunt obiectele specifice elementelor compuse dintr-un meniu. Acestea sunt create din mai multe MenuItems.

Business.Order

Order sunt obiectele specifice datelor comenzii precum masa, data, pretul.

Business.IRestaurantProcessing

Aceasta este o interfata pe care o implementeaza Restaurant, cu metodele necesare pentru utilizatori.

Business.Restaurant

Clasa Restaurant contine toate datele necesare restaurantului, avand un HashMap ce contine fiecare comanda impreuna cu tot ce s-a comandat, o lista cu toate elementele din meniu, mesele ocupate/libere in momentul actual.

Presentation.AdministratorGraphicalUserInterface

Aceasta clasa contine interfata grafica pentru administrator, avand 5 actiuni diferite pe care le poate executa pe baza TextField-urilor si List-urilor date: adaugarea unui produs de baza, adaugarea unui produs compus, editarea unui produs de baza, editarea unui produs compus, stergerea unui produs din meniu. Orice actiune facuta de administrator modifica in timp real oricare alta lista, atat din interfata curenta, cat si din cele alea ospatarului si ale bucatarului.

Elementele grafice specifice fiecarei actiuni posibile, exceptand butoanele, sunt:

* Add Base Product: 2 JTextField pentru numele si pretul noului produs ce urmeaza sa fie adaugat;
* Add Composite Product: un JList constand in toate produsele disponibile ce pot fi selectate si un JTextField pentru numele noului produs;
* Edit Base Product: un JList cu toate BaseProduct disponibile, si 2 campuri pentru noile atribute;
* Edit Composite Product: un JList cu toate CompositeProduct disponibile, un JList cu toate produsele disponibile, si un JTextField pentru numele nou al produsului;
* Delete Product: un JList cu toate produsele, alegand care din ele sa fie eliminat.

Presentation.AGUIController

Aceasta este controller-ul pentru interfata administratorului si contine si clase ce implementeaza ActionListener pentru ca butoanele sa fie functionale si sa execute operatia corespunzatoare.

Presentation.WaiterGraphicalUserInterface

Aceasta clasa contine interfata grafica pentru ospatar, avand 3 actiuni diferite: crearea unei comenzi, calcularea pretului si generarea notei de plata. De remarcat ca pentru o masa la care s-a facut o comanda pentru care nu s-a creat o nota de plata, daca masa respectiva mai face o comanda, la final se va genera o singura nota de plata. Totodata, ospatarul vede pentru cele 2 actiuni din urma numai comenzile si mesele asupra carora inca actiunea nnu s-a executat, apoi dispar.

Elementele grafice specifice fiecarei actiuni posibile sunt:

* CreateOrder: un JList cu toate mesele, un JList cu toate produsele, un JList cu produsele ce urmeaza sa fie adaugate la comanda;
* ComputePrice: un JList cu toate comenzile a caror pret nu a fost procesat;
* GenerateBill: un JList cu toate mesele asupra carora putem crea o nota de plata.

Presentation.WGUIController

La fel ca si la interata precedenta, exista un controller ce face posibila actiunile butoanelor.

Presentation.ChefGraphicalUserInterface/Observer/

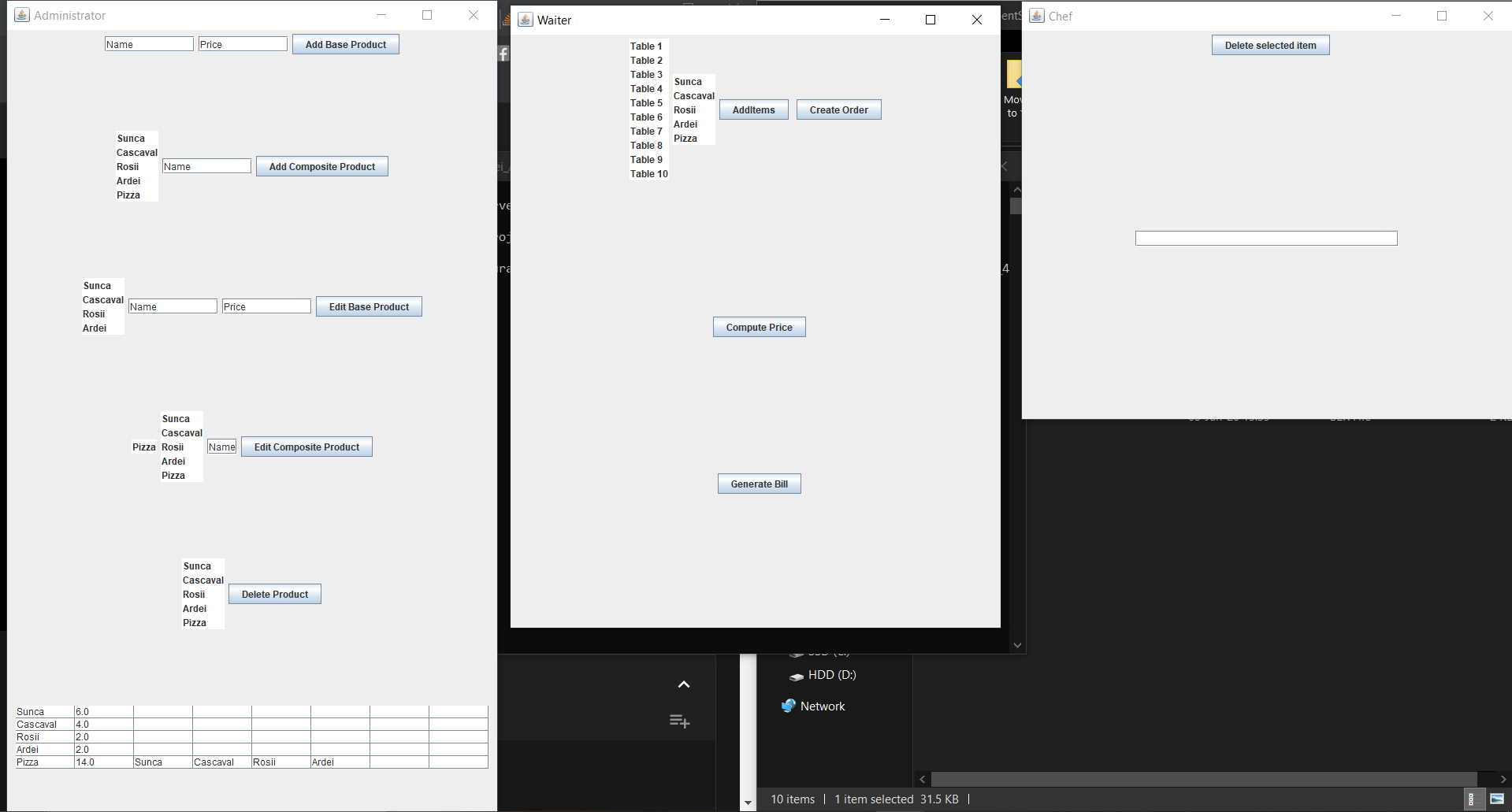
ChefObserver

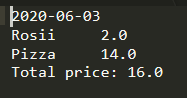
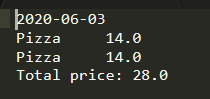
Aceste clase respecta Observer Design Pattern, astfel ca interfata specifica bucatarului se actualizeaza in timp real, iar pentru orice actualizare, un mesaj apaae intr-un TextField, notificand bucatarul asupra oricarei noi comenzi executate de ospatar. Totodata bucatarul poate sa elimine un produs din lista cu toate produsele ce trebuie preparate.

Elementele grafice specifice sunt un JList cu toate produsele ce trebuie sa fie preparate de catre bucatar, si un JTextField in care apare un mesaj de fiecare data cand exista un nou produs ce trebuie preparat.

5.Rezultate

Pentru a testa aplicatia, am introdus cateva produse in meniu, apoi am creat 2 noti de plata in format .txt.





6.Concluzii

In urma implementarii acestui program am invatat mai multe Pattern-uri, conceptele de serializare si deserializare si mi-am imbogatit cunostintele asupra interfetelor grafice si obiectelor java asociate.

Imbunatatiri posibile:

* Fereastra de login, si mai multi utilizatori de fiecare tip.
* Notificarea prin sunet a bucatarului in momentul unei actualizari.
* Mentinerea stocului fiecarui produs.
* Verificarea valorilor introduse si tratarea erorilor in mod corespunzator.

7.Bibliografie

* <https://docs.oracle.com/javase/tutorial/uiswing/components/list.html>
* <https://www.tutorialspoint.com/java/java_serialization.htm>
* <https://www.tutorialspoint.com/design_pattern/observer_pattern.htm>
* <https://dzone.com/articles/working-with-hashcode-and-equals-in-java>
* <http://coned.utcluj.ro/~salomie/PT_Lic/4_Lab/Assignment_4/Assignment_4_Indications.pdf>
* <https://www.geeksforgeeks.org/composite-design-pattern/>