

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

RESTAURANT MANAGEMENT SYSTEM

Documentație

Dorofte Andrei

Grupa 302210

Cuprins

1. Obiectivul temei

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

3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

4. Implementare

5. Rezultate

6. Concluzii

7. Bibliografie

1. **Obiectivul temei**

Proiectati si implementati o aplicatie de simulare care sa analizeze managementul unui restaurant

Un restaurant poate fi observat atat in lumea reala cat si ca model virtual. Obiectivul principal al unui restaurant este de a primi si servi clientii cu ceea ce acestia doresc sa coamnde din meniul restaurantului.

De asemenea, restaurantul are trei tipuri de angajati: administrator, chelner si bucatar. Fiecare dintre acestia pot efectua diferite operatii:

* Administratorul poate adauga elemente in meniu, le poate sterge, dar si edita. De asemenea, acesta poate sa vada tot meniul
* Chelnerul poate doar sa adauge comenzi, sa adauge elemente din meniu la o anumita comanda, sa vada pretul total al unei comenzi si sa genereze nota de plata pentru o anumita comanda.
* Bucatarul este doar notificat atunci cand trebuie sa gateasca ceva care contine mai multe elemente din meniu.

|  |
| --- |
| **Obiective secundare** |
| * Crearea claselor necesare, respectand Composite Design Pattern si Observer Design Pattern |
| * Definirea metodelor de baza pentru fiecare clasa |
| * Crearea interfetelor necesare si legarea de proiect |
| * Implementarea operatiilor fiecarui tip de utilizator |
| * Realizarea algoritmilor care prelucreaza operatiile |
| * Introducerea generatorului de nota de plata in format text |
| * Crearea conexiunii proiectului cu fisierul de tip ser |

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

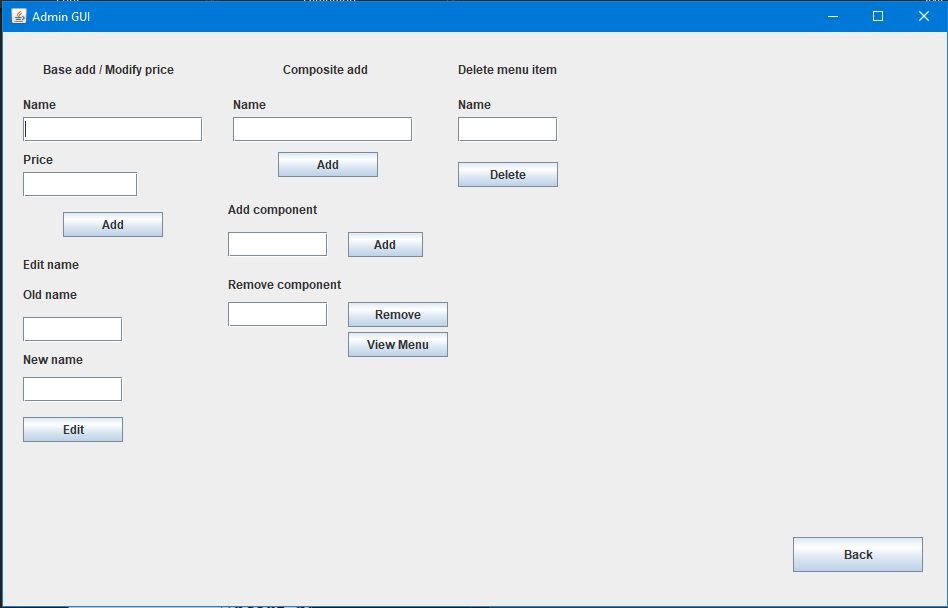
Cu totii mergem uneori la cate un restaurant, dar poate nu ne-am gandit niciodata cum functioneaza sistemul de gestionare al comenzilor si al meniului. O astfel de aplicatie usureaza considerabil munca personalului unui restaurant.

Acest proiect are rolul de a simula un restaurant, al carui meniu este stocat intr-o lista. Asemenea sunt stocate si comenzile, intr-o colectie de tip HashMap, impreuna cu o lista a produselor comandate. Atunci cand chelnerul plaseaza o comanda, trebuie sa se verifice daca produsul se afla in meniu, iar daca da, sa se creeze comanda, alaturi de o noua lista la care se adauga produsul respectiv. Sistemul trebuie sa realizeze operatiile aferente fiecarui utilizator.

* Sistemul trebuie sa realizeze operatiile pe care le poate realiza administratorul.
* Sistemul trebuie sa operatiile pe care le poate realiza chelnerul.
* Sistemul trebuie sa notifice bucatarul la adaugarea unui produs de tip Composite la o comanda

**Datele de intrare:**

Administratorul introduce pe rand elementele din meniu, in functie de tipul lor, BaseProduct, sau CompositeProduct.



**Datele de iesire:**

Chelnerul sau administratorul opereaza cu acestea daca vreun client doreste sa comande ceva si realizeaza operatia dorita, rezultatul fiind afisat ori pe interfata, ori in format txt, in functie de operatie.

Male icon

Restaurant

Male icon

Insert item

Edit item

Admin

Admin

Output

View items

Male iconMale icon

Add Order

Output

View orders

Waiter

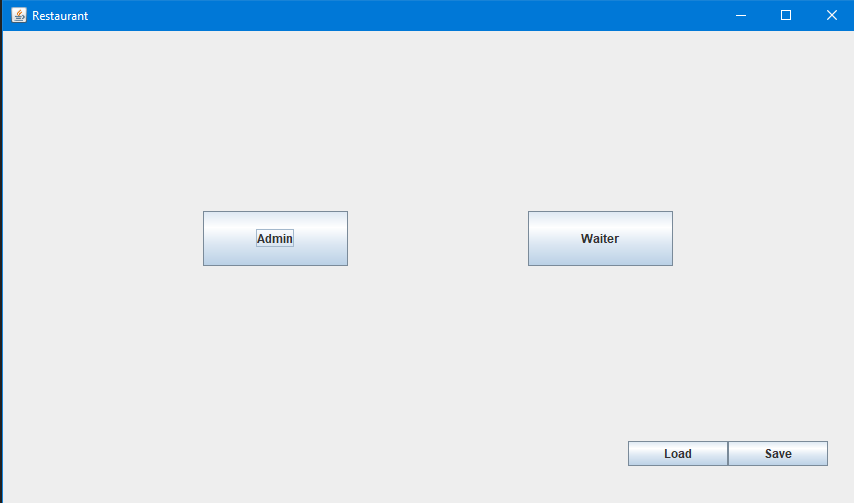
Waiter

File

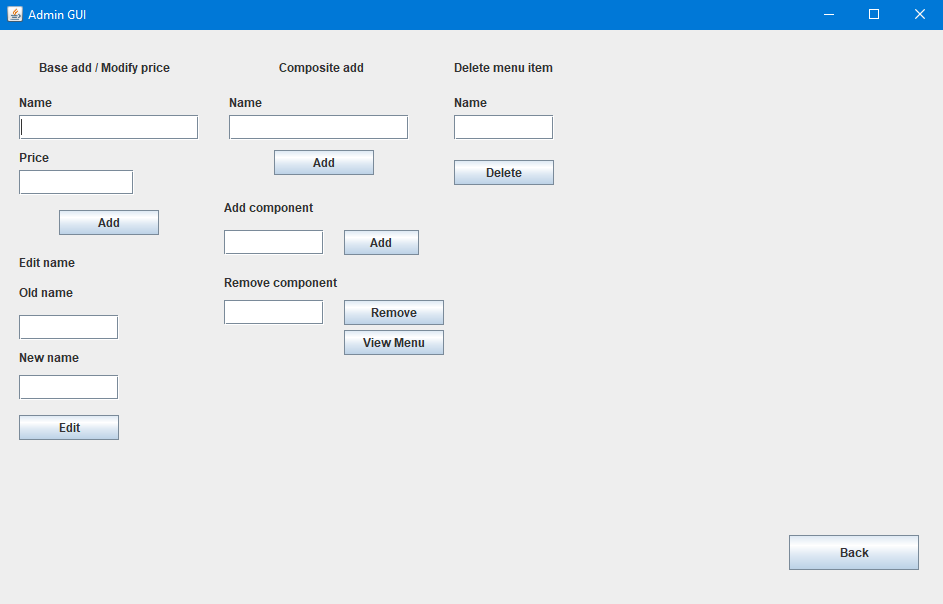
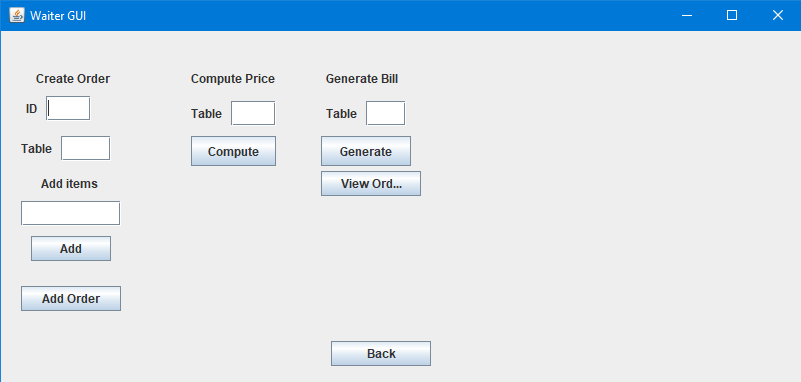
Compute Bill

Caz de utilizare:

1. Utilizatorul ruleaza aplicatia dintr-o fereastra command prompt, deschisa din folder-ul in care se afla aplicatia, utilizand comanda „java -jar App.jar”. Se alege tipul de utilizator aferent operatiei pe care doreste sa o realizeze. Daca se doreste incarcarea unui model de restaurant salvat anterior, se apasa pe load.

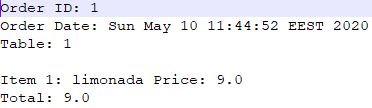


1. Se realizeaza operatiile dorite



Pentru modificarea pretului unui produs de tip Base, se va introduce numele si noul pret in casutele din stanga sus, de unde se di adauga in produs nou de tip Base.

Pentru a adauga produse la o comanda, intai se creeaza comanda introducand id-ul si numarul mesei, dupa care acestea se lasa introduse, si se scrie numele produsului ce urmeaza sa fie adaugat.

 In cazul in care se va genera nota de plata, ea va arata asa:

1. Pentru revenire la meniul anterior se apasa back din meniul curent. Daca se doreste salvarea modelului de restaurant creat, se apasa pe save.
2. **Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfețe, relații, packages, algoritmi, interfață utilizator)**

Unified Modeling Language sau UML, pe scurt, este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora ( numite și obiecte ). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT.

A close up of a map

Description automatically generated

**Diagrama UML:**

Am încercat sa modularizez cerința cat de mult posibil, astfel rezultând clasele din diagrama de mai sus: MenuItem, BaseProduct, CompositeProduct, Order, Restaurant, AdminGUI, WaiterGUI, MainGUI, ChefGUI, RestaurantSerializator, Main si interfetele IrestaurantProcessingAdmin, IrestaurantProcessingWaiter, FileWorker, Observer si Observable. Clasele au fost proiectate intuitiv, pentru a fi ușor de citit si de înțeles.

Pachetele create sunt bll, data, presentation, App, conform arhitecturii structurata pe straturi(layered architecture). Astfel, logica aplicatiei se afla in pachetul bll, iar in pachetul data sunt implemetate operatiile cu fisiere, repectiv generare nota de plata si serializare/deserializare. Pachetul presentation contine interfetele grafice ale utilizatorilor, dar si interfata grafica principala. Pachetul App contine doar main-ul aplicatiei.

1. **Implementare**

**Clasa abstracta MenuItem**

Aceasta clasa face parte din nucleul proiectului, ea stand la baza meniului restaurantului. A fost implementata utilizand Composite Design Pattern, impreuna cu cele 2 clase copil pe care le are, BaseProduct si CompositeProduct. Contine getteri si setteri pentru nume si pret, dar si metode particulare abstracte ce vor fi implementate ulterior de care copiii acesteia.

**Clasa BaseProduct**

Aceasta clasa extinde clasa abstracta MenuItem si a fost creata conform Composite Design Pattern, si implementeaza, pe langa getterii si setterii din superclasa, metoda abstracta computePrice() din clasa parinte MenuItem. Fiind BaseProduct, aceasta metoda returneaza doar pretul produsului.

**Clasa CompositeProduct**

Aceasta clasa extinde clasa abstracta MenuItem si a fost creata conform Composite Design Pattern. Astfel, contine o lista de MenuItems din care este alcatuit produsul Composite. Implementeaza, pe langa getterii si setterii din superclasa, metodele abstracte computePrice(), care calculeaza pretul produsului Composite adunand preturile produselor din lista, getItems(), care returneazaa lista de produse din care este alcatuit, isHere(), care verifica daca un anumit produs se afla in lista de produse pe care o contine, si metodele addItem() si removeItem(), care adauga si scot din lista cate un produs si calculeaza de fiecare data noul pret.

**Clasa Order**

Aceasta clasa implementeaza modelul unei comenzi, prin id, data la care a fost creata comanda si numarul mesei careia ii este atribuita. De asemenea, se implementeaza metodele equals() si hashCode(), necesare pentru colectia de tip HashMap, astfel incat fiecarei perechi de tip comanda-lista sa ii fie asignat un loc diferit. Astfel, se usureaza stocarea comenzilor si a produselor comandate.

**Clasa Restaurant**

Aceasta clasa face parte din nucleul proiectului. Fara aceasta, nu ar functiona nimic, intrucat aici se implementeaza toate operatiile care se pot executa de care administrator sau chelner: addMenuItemBase(), addMenuItemComposite(), add Component(), removeComponent(), editMenuItem(), deleteMenuItem, showTable(), createOrder(), addItemsOrder(), showOrders(), generateBill(). Metodele ce opereaza cu meniul, itereaza prin acesta si realizeaza metodele necesare, pe cand metodele ce opereaza cu HashMap-ul, itereaza cu ajutorul unui entrySet(), ce returneaza perechea de tip comanda-lista. Meniul este stocat intr-o colectie de tip ArrayList, iar comenzile impreuna cu o lista de produse ce au fost comandate, intr-o colectie de tip HashMap. Modificarea sau stergerea unui produs de tip Base, rezulta in recalcularea pretului unui produs de tip Composite ce il avea in alcatuire. Fiecare operatie realizata cu succes se reflecta intr-un mesaj afisat in consola.

**Clasa AdminGUI**

Aceasta clasa defineste interfata grafica pentru utilizatorul de tip administrator, ce poate adauga, sterge sau edita un produs din meniu. Am utilizat programul gratuit GuiGenie pentru crearea interfetei utilizand Absolute Positioning, astfel incat am putut realiza interfata asa cum mi-am imaginat-o. De asemenea, acesta poate vizualiza tot meniul intr-un tabel de tip JTable. Fiecarui buton i-a fost adaugat un actionlistener, astfel incat la fiecare apasare sa se apeleze metoda potrivita, implementata in clasa Restaurant.

**Clasa WaiterGUI**

Aceasta clasa defineste interfata grafica pentru utilizatorul de tip chelner, ce poate adauga o noua comanda si poate adauga produse la aceasta, poate vizualzia pretul unei anumite comenzi dupa numarul mesei careia ii este atribuita, dar poate si genera nota de plata pentru o anumita masa. Am utilizat programul gratuit GuiGenie pentru crearea interfetei utilizand Absolute Positioning, astfel incat am putut realiza interfata asa cum mi-am imaginat-o. De asemenea, acesta poate vizualiza toate comenzile intr-un tabel de tip JTable. Fiecarui buton i-a fost adaugat un actionlistener, astfel incat la fiecare apasare sa se apeleze metoda potrivita, implementata in clasa Restaurant.

**Clasa MainGUI**

Aceasta clasa defineste interfata grafica principala pe care utilizatorul o vede de fiecare data cand ruleaza aplicatia. Am utilizat programul gratuit GuiGenie pentru crearea interfetei utilizand Absolute Positioning, astfel incat am putut realiza interfata asa cum mi-am imaginat-o. Prin intermediul celor 2 butoane principale se alege tipul de utilizator. Fiecarui buton i-a fost adaugat un actionlistener, astfel incat la fiecare apasare sa se apeleze metoda potrivita, implementata in clasa Restaurant.

**Clasa ChefGUI**

Aceasta clasa ar fi trebuit sa implementeze interfata care notifica bucatarul cand un produs de tip Composite este adaugat la o comanda.

**Clasa RestaurantSerializator**

Aceasta clasa implementeaza metodele de serializare si deserializare ale obiectului restaurant creat in clasa main. Metodele sunt apelate prin butoanele load si save din interfata grafica principala din clasa MainGUI.

**Clasa Main**

Aceasta clasa creeaza Jframe-ul interfetei principale si creeaza obiectul de tip restaurant, care este dat mai departe ca parametru Interfetelor grafice.

**Interfata IrestaurantProcessingAdmin**

Aceasta interfata defineste metodele operatiilor pe care le poate realiza utilizatorul de tip administrator, ce au fost ulterior implementate in clasa Restaurant: adaugare, editare si vizualizare meniu.

**Interfata IrestaurantProcessingWaiter**

Aceasta interfata defineste metodele operatiilor pe care le poate realiza utilizatorul de tip chelner, ce au fost ulterior implementate in clasa Restaurant: adaugare comanda, adaugare produse la comanda, afisare pret si vizualizare comenzi.

**Interfata FileWorker**

Aceasta interfata implementeaza metoda statica writeFile, care scrie intr-un fisier detaliile comenzii si fiecare produs comandat, impreuna cu pretul acestuia. De asemenea, la final se scrie si pretul total al comenzii. Fisierul se denumeste conform numarului mesei caruia ii este atribuita comanda.

**Interfata Observable**

Aceasta clasa defineste metodele registerObservers, notifyObservers si remove Observer, ce ar fi trebuit sa fie implementate in clasa Restaurant, conform Observer Design Pattern, pentru a notifica bucatarul de fiecare data cand se adauga un produs de tip Composite la o comanda.

**Interfata Observer**

Aceasta clasa defineste metoda update(), ce ar fi trebuit sa fie implementata in clasa Restaurant, conform Observer Design Pattern, pentru a notifica bucatarul de fiecare data cand se adauga un produs de tip Composite la o comanda.

1. **Rezultate**

Pentru cazurile de testare am testat aplicatia pentru cateva produse de tip Base, si unul de tip Composite, ce este alcatuit din produsele de tip Base. Am editat si sters cu ssucces un produs de tip Base, atat din meniu, cat si din compositita produsului de tip Composite. Ulterior, am adaugat o comanda cu produsul de tip Composite, am afisat pretul, dupa care am generat nota de plata. In final, am salvat modelul obiectului Restaurant, pentru a testa functionalitatea serializarii si deserializarii.

Rezultatele din etapa de testare arata ca aplicatia este modelata corect, iar algoritmii de lucru pentru fiecare operatie implementata, sunt functionali.

1. **Concluzii**

In concluzie, sunt de părere ca acest proiect m-a ajutat sa îmi aprofundez cunoștințele in tot ce înseamnă limbajul Java, implementarea paradigmelor OOP, si dezvoltarea unui algoritm eficient, care sa satisfacă obiectivul unui proiect.

Aceasta tema simuleaza situatii din viata de zi cu zi, intrucat, de fiecare data cand am intrat intr-un restaurant, nu ne-am pus niciodata in postura unui angajat, si nu ne-am imaginat ce presupune lucrul intr-o astfel de locatie.

Chiar daca simularea este corecta, ea nu poate prevedea toate situatiile neasteptate din realitate. In aceasta tema am invatat sa lucrez cu programul GuiGenie si sa generez un document text utilizand limbajul Java. De asemenea, am aprofundat generarea si configurarea unui fisier .jar si rularea din linia de comanda.

In ceea ce privește dezvoltarea ulterioara a programului:

* Se poate imbunatati performanta algoritmilor
* Se pot implementa operatii de notificare a bucatarului atunci cand se adauga la o comanda un produs de tip Composite, conform Observer Design Pattern.
* Se poate imbunatati interfata grafica

1. **Bibliografie**

* <https://stackoverflow.com/>
* <http://docs.oracle.com/>
* <https://www.geeksforgeeks.org/>
* youtube.com - tutoriale