**DOCUMENTATIE**

**TEMA 4**

**RESTAURANT MANAGEMENT SYSTEM**

TEHNICI DE PROGRAMARE

UNIVERSITATEA TEHNICA DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICA SI CALCULATOARE

SPECIALIZAREA CALCULATOARE SI TEHNOLOGIA INFORMATIEI

*Piștănilă Ana-Maria*

*Grupa 30226*

CUPRINS

1. Obiectivul temei. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   1. Obiectivul principal. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   2. Obiective secundare. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
2. Analiza problemei. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.1 Modelare. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.2 Scenarii. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.3 Cazuri de utilizare. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1. Proiectare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.1 Decizii de proiectare. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.2 Diagrame UML. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.3 Structuri de date. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.4 Proiectare clase. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.5 Algoritmi. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1. Implementare. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
2. Rezultate. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
3. Concluzii. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
4. Bibliografie. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**1.OBIECTIVUL TEMEI**

* 1. *Obiectiv principal*

Obiectivul acestei teme este de a implementa o aplicatie cu interfata grafica pentru a fi folosita ca simulator al unui restaurant, cu trei tipuri de utilizatori: administratorul, chelnerul si bucatarul restaurantului.

* Administratorul poate adauga, sterge si modifica produsele existente din meniu.
* Chelnerul poate crea o noua comanda pentru o masa, adauga elemente din meniu si calculeaza bonul pentru o comanda.
* Bucatarul restaurantului este anuntat (notificat) de fiecare data

* 1. *Obiective secundare*

Obiectivele secundare sunt:

* realizarea operatiilor de adaugare a unui produs/comanda, stergerea unui produs, modificarea unui produs din meniu, calcularea pretului pentru comenzi, respectiv pentru produsele compuse, crearea interfetelor grafice
* folosirea corecta a conceptelor de programare OOP;
* alegerea structurilor de date;
* impartirea pe clase;
* scrirea unui cod care poate fi usor de inteles de catre oricine il citeste;
* dezvoltarea de use caseuri si scenarii;
* testare;

**2.ANALIZA PROBLEMEI**

Realizarea acestei probleme implica pe langa cunoasterea principiilor OOP si cunoasterea realizarii interfetelor grafice.

Interfața grafică  este o [interfață cu utilizatorul](https://ro.wikipedia.org/wiki/Interfa%C8%9B%C4%83_cu_utilizatorul) bazată pe un sistem de afișaj ce utilizează elemente grafice.

Componentele unei interfeţe grafice se numesc elemente de control si pot fi activate de către utilizator cu ajutorul mausului sau al tastaturii. Activarea unui element de control determină un anumit răspuns din partea programului, răspuns concretizat prin execuţia unei anumite operaţii. Cele mai răspândite interfeţe grafice sunt cele bazate pe ferestre, adică fiecărei aplicaţii active la un moment dat i se alocă o zonă dreptunghiulară pe ecran, numită fereastră. În interiorul ferestrei respective, se vor găsi toate elementele de control necesare dialogului utilizator-aplicaţie, precum si informaţiile afişate de aplicaţie.

Pentru realizarea proiectului meu am creat mai multe clase cu scopul de a avea functionalitati specifice clasei respective si de a putea adauga cu usurinta noi functionalitati in urmatoarele versiuni ale aplicatiei, astfel am incercat pe tot posibilul sa generalizez pentru fiecare clasa metodele si atributele acesteia.

Am structurat proiectul pe mai multe clase si mai multe pachete pentru a reda fiecarei clase o singura functionalitate si le-am impartit pe pachete in functie de functionalitatea comuna, astfel pachetele si clasele proiectului meu sunt: DataLayer ( cu clasa RestaurantSerializator), BusinessLayer( cu interfata IRestaurantProcessing si clasele Restaurant, BaseProduct, CompositeProduct, MenuItem, Order), PresentationLayer( cu clasele ChefGUI, AdminGUI, WaiterGUI) si Main( cu clasa Main ) .

Utilizatorului aplicației trebuie să i se ofere o listă de produse disponibile care apartin unui meniu din restaurant și o listă de comenzi efectuate de clienti si preluate de chelnerul restaurantului.. Ambele liste trebuie să conțină toate informațiile legate de comenzi și produse. Utilizatorul se poate afla in ipostaza administratorului și să poata adauga un nou produs in meniu, sterge si edita lista produselor din meniu. Utilizatorul, de asemenea, se poate afla si in ipostaza chelnerului si sa creeze o noua comanda, sa calculeze pretul pentru o comanda data de client si sa genereze factura in format .txt .

**3.PROIECTARE**

Structuri de date folosite:

-ArrayList- deoarece este o colectie simpla care poate stoca orice tip de obiect. Aceasta clasa ne permite sa construim un vector care sa creasca in marime, adaugand mereu elemente noi. In plus, avantajele majore ale acestei clase sunt utilizarea mai buna a memoriei si administrarea mai usoara a elementelor listei.

-Map: -deoarece descrie structuri de date ce asociaza fiecarui element o cheie unica, dupa care poate fi regasit. Obiectele de acest tip nu pot contine chei duplicate si fiecare cheie este asociata la un singur element.

Proiectare clase

Pachetul BusinessLayer:

-Clasa BaseProduct: In aceasta clasa se vor afla produsele de baza dintr-un restaurant. Acestea pot reprezenta atat produse care se folosesc in compozitia unui produs compus, cat si produse separate, cum ar fi o apa minerala, un suc sau ingrediente pe care clientul doreste sa le adauge suplimentar. Aceasta clasa extinde clasa MenuItem. Aici am declarant un String nume care va reprezenta numele produsului si un double pret, pretul acestuia. Metodele prezente in aceasta clasa sunt doar gettere si settere pentru pret si nume.

-Clasa CompositeProduct: In aceasta clasa se vor afla produsele compuse, care vor fi formate din produsele de baza prezentate in clasa anterioara. De exemplu pizza care are ca ingredient sunca, rosii, branza si ciuperci. Aceasta clasa extinde de asemenea clasa abstracta MenuItem. Aici am declarant un ArrayList<MenuItem> care reprezinta lista de ingrediente ce intra in compozitia produsului, dar si un String nume care reprezinta numele produsului. Aici am metode de get si set pentru aceste campuri + metoda getPret care este implementata pentru clasa abstracta MenuItem. Aceasta metoda calculeaza pretul produsului in functie de ingredientele adaugate.

-clasa MenuItem: Aceasta este o clasa abstracta care implementeaza interfata Serializable. Aici am declarant doar un id care reprezinta id-ul fiecarui produs din meniu. Aici am metode de get si set pentru id, cat si cele doua metode getPret si getNume care vor fi implementate de subclase.

-clasa Order: In aceasta clasa se vor afla comenzile pe care clientii doresc sa le faca in functie de meniul pus la dispozitie de administrator. Aici am declarat un id\_comanda care este unic pentru fiecare comanda facuta, o variabila data, care este de tipul Date si care imi va genera data la care s-a efectuat comanda si un numar\_tabel care reprezinta numarul mesei de la care s-a efectuat comanda.Aici am metode de get si set pentru aceste variabile, o metoda equals care imi verifica daca un obiect este de tipul Order si o metoda hashCode care reprezinta functia de hash pentru tabelul meu.

-clasa Restaurant: Aceasta clasa va implementa interfata IProcessingRestaurant. In aceasta clasa am declarant un Map pentru comenzi, unde fiecare comanda va avea o cheie de tipul obiectului order si ca valoare o lista de menuItems, dar am declarat si o lista de produse care reprezinta meniul. Aici am metode pentru: crearea unui BaseProduct si adaugarea lui in lista de produse, crearea unui compositeProduct si adaugarea lui in lista de produse, stergerea unui produs dupa nume, modificarea unui produs din meniu dupa un id dat, crearea unei comenzi, calcularea pretului unei comenzi, cautarea unui anumit id in hashmap si generarea unei chitante pentru fiecare comanda.

-interfata IProcessingRestaurant: Aceasta interfata contine antetele de metode enuntate mai sus in clasa Restaurant.

Pachetul DataLayer:

-clasa RestaurantSerializator:, Aici am implementat cele doua metode specifice interfetei Serializable. Serializarea reprezinta transformarea unui obiect intr-o secventa de octeti, din care sa poata fi refacut ulterior obiectul original. Este un mecanism utilizat pentru salvarea si restaurarea datelor. Metoda writeData()  este responsabila pentru salvarea starii obiectului, iar metoda readData() are responsabilitatea restaurarii starii obiectului.

Pachetul Main:

-clasa Main: Aici am creat instantele pentru clasa Restaurant si clasa RestaurantSerializator, cat si pentru interfete.

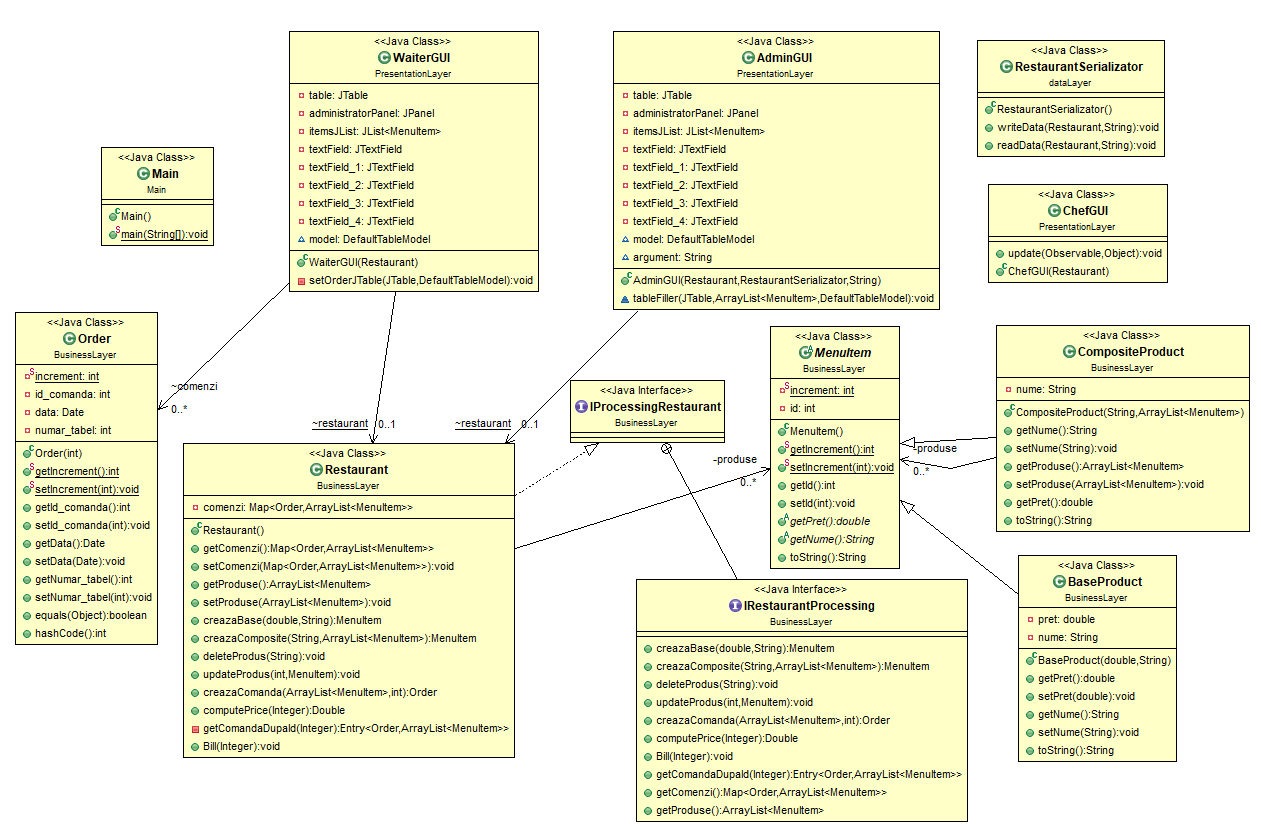
Pachetul PresentationLayer:

-clasa AdminGUI: In aceasta clasa am creat interfata pentru administrator. Aici am 4 butoane, unul pentru adaugarea unui produs, unul pentru stergerea unui produs, unul pentru modificarea unui produs si un buton vizualizare. La apasarea butonului Vizualizare se va afisa continutul tabelului care reprezinta meniul restaurantului. La apasarea butonului adaugaProdus se vor adauga produsele introduse din textField intr-un JList. La apasarea butonului stergeProdus se va sterge un produs din tabel dupa un nume dat in textField, iar la apasarea butonului ModificaProdus se va modifica un anumit produs din tabel dupa un anumit id dat.

-clasa WaiterGUI: In aceasta clasa am creat interfata pentru chelner. Aici am 4 butoane, unul de vizualizare, un buton AdaugaComanda, un buton Chitanta si un buton Meniu. La apasarea butonului chitanta se vor adauga intr-un JList produsele din meniu pentru a fi vizibile clientilor care doresc sa comande. La apasarea butonului AdaugaComanda se vor selecta cu ctrl anumite elemente din JList pe care clientul doreste sa le comande, se va introduce un numar\_masa care reprezinta masa de la care s-a facut comanda si se va adauga acea comanda. La apasarea butonului Chitanta si la introducerea unui anumit id comanda din textField se va genera o chitanta in format .txt cu datele comenzii. Iar cand apasam butonul Vizualizare ni se va afisa continutului tabelului de comenzi efectuate.

DIAGRAME UML

Diagramele sunt grafuri care prezintă simboluri ale elementelor de modelare (model element) aranjate astfel încât să ilustreze o anumită parte sau un anumit aspect al sistemului.Diagramele **UML** fac mult mai evidenta functionalitatea acestei aplicatii. Diagrama de clase foloseste clase si o interfata si mai multe relatii intre acestea. Ea descrie din punct de vedere structural sistemul si ajuta la intelegerea implementarii acestuia.



**4.IMPLEMENTARE**

Aplicatia este facuta astfel incat sa corespunda cu un restaurant din lumea reala. Administratorul va propune un meniu care este accesibil pentru toti clientii care doresc sa vina la acest restaurant, iar chelnerul se va ocupa de comenzi si va genera nota de plata pentru fiecare in parte.

Utilizatorul are acces atat la interfata Administrator cat si la cea Waiter. Astfel el va putea introduce ce elemente doreste la meniu si pretul acestora. De asemenea poate modifica un anumit produs, in cazul in care nu mai e in stoc sau pretul lui s-a schimbat. Si poate sterge un anumit produs daca este necesar. In interfata Waiter, apasand butonul meniu, se va afisa in Jlistul creat toate elementele introduse de Administrator in meniu. Astfel clientii pot selecta de acolo cate elemente doresc folosind ctrl pentru a efectua o comanda. Chelnerul se va ocupa de „notarea” comenzii( numarul mesei de la care s-a efectuat comanda, id-ul de comanda, data la care s-a efectuat si pretul total avand in vedere produsele comandate). Apoi, cand clientii doresc sa paraseasca restaurantul, chelnerul le va genera nota de plata introducand un id de comanda.

**5.REZULTATE**

Rezultatele aplicatiei sunt corecte si simuleaza corect un sistem de gestionare a unui restaurant. Dupa cum se observa din faza de testare, aplicatia poate sa adauge produse, pretul acestora sau sa le stearga din meniu. De asemenea , se realizeaza comenziile si se genereaza factura. Astfel este demonstrata buna functionalitate a aplicatiei.

Aplicatia ne ajuta sa ne dam seama cum functioneaza o baza de date, cum stocheaza anumite informatii, cum le modifica si cum le sterge in functie de comenzile pe care utilizatorul le da.

**6.CONCLUZII**

In concluzie, acest proiect m-a ajutat sa imi aprofundez cunostintele de programare orientata pe obiect si sa invat noi concepte ale limbajului Java.

In cadrul dezvoltarilor ulterioare se pot aminti urmatoarele: implementarea a diverse metode ce ajuta la securizare ( logarea ca chelner , administrator sau bucatar ) . Astfel, aplicatia ar avea un impact mult mai potrivit asupra realitatii , deoarece proiectul actual nu asigura o prea mare siguranta utlizatorilor, rezumandu-se strict la partea teoretica a ideii temei .

**7.BIBLIOGRAFIE**

* <http://stackoverflow.com/>
* <http://users.utcluj.ro/~igiosan/teaching_poo.html>
* <https://profs.info.uaic.ro/~acf/java/slides/extra/serializare_slide.pdf>
* <http://coned.utcluj.ro/~salomie/PT_Lic/4_Lab/Assignment_4/Assignment_4_Indications.pdf>
* <https://docs.oracle.com/javase/8/docs/api/java/util/Date.html>
* <https://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html>