**Documentatie tema4**

**Sistem de administrare a restaurantelor**

**Șimon Bianca Andreea**

**Grupa 30222**

**Cuprins:**

1.Cerinte functionale…………………………………………………………….3

2.Obiectivul temei…………………………………………………………….…3

3.Analiza problemei………………………………….………………………….4

4.Proiectare………………………………………………………………………5

5.Implementare………………………………………………..…………………8

6.Concluzii…………………………………………………………...…………12

7.Bibliografie……………………………………………………………...……12

**1.Cerinte functionale**

Proiectati si implementati un system de administrare a unui restaurant, avand 3 tipuri de utilizatori: administrator, chelner si bucatar. Fiecare dintre utilizatori poate sa efectueze diferite comenzi: administratorul poate sa adauge noi produse de baza si produse mixte, sa stearga si sa modifice produsele deja existente in meniu. Chelnerul poate sa creeze o noua comanda pentru o anumita masa, sa calculeze nota finala unui anumite comenzi, sa genereze bon sis a vizualizeze comenzile.

De asemenea se poate vizualiza continutul meniului.

**2.Obiectivul temei**

2.1.Obiectivul principal

Obiectivul principal al acestei teme este crearea unei aplicatii care are scopul de a administra un restaurant, iar in functie de utilizatorul care foloseste aplicatia si anume administrator sau chelner, se pot efectua diferite comenzi.

2.2.Obiective secundare

|  |  |  |
| --- | --- | --- |
| Obiectiv secundar | Descriere | Capitol |
| Dezvoltarea de use case-uri si scenarii | “Use case”-urile reprezinta un set de scenarii legate de cum sistemul este folosit si interactiunea acestuia cu un rol(in cazul nostrum, un utilizator al aplicatiei). | 3 |
| Structuri de date | Alegerea structurii de date pentru a duce la capat obiectivul principal | 4 |
| Impartire pe clase si pachete | Folosirea arhitecturii 3-tier, combinate cu un MVC(Model-View-Controller). | 4 |
| Diagrame UML | Se va prezenta schema UML de clase | 4 |
| Dezvoltarea algoritmilor | Se vor prezenta algoritmii folositi pentru realizarea operatiilor. | 4 |
| Implementarea solutiei | Se va descrie clasele cu campurile si metodele importante. Se va descrie implementarea interfetelor administrator si chelner. | 5 |
| Testare | Se va prezenta cateva scenarii de testare a unor comenzi. | 6 |

**3.Analiza problemei**

3.1.Use case-uri

Aceasta aplicatie consta in administrarea unui restaurant de catre doi utilizatori diferiti, administratorul si chelnerul. In functie utilizatorul logat, se pot efectua diferite comenzi.

Administratorul poate adauga noi produse de baza si produse mixte, poaste sterge produse deja existente, poate sa modifice pretul unui compus de baza, iar in acest caz se modifica si pretul unui produs compus care contine produsul de baza respective. De asemenea poate adauga sau sterge produse de baza din/in produse compuse.

Chelnerul poate adauga noi comenzi pentru o anumita masa la care nu s-a luat comanda inainte, poate sa calculeze nota finala pentru o anumita masa la care s-a efectuat o comanda, iar dupa aceea poate sa genereze nota de plata (se poate si inainte de a se calcula nota finala, dar atunci pe bon va aparea ca nota de plata se 0). In plus poate sa vada si toate comenzile efectuate.

In momentul in care se deschide aplicatia, va mai aparea o comanda pe care poate sa o faca si administratorul si si chelnerul, si anume vizualizarea meniului.

Este posibil ca cei doi utilizatori sa introduca date invalide. Daca administratorul doreste sa adauge un nou produs deja existent, va aparea un panou specific unei erori, care va atentiona utilizatorul ca acel produs exista. Pentru fiecare comanda pe care fie adminsitratorul fie chelnerul vrea sa o efectueze, se verifica unele date, pentru ca operatia sa se efectueze cu success.

3.2.Scenarii

Principalele scenarii de success:

1. Chelnerul introduce numele unei mese la care a luat comanda.
2. Acesta introduce produsele compuse pe care masa respective le-a comandat.
3. Apasa butonul de “Add order”.
4. Comanda a fost efectuata cu succes.
5. Chelnerul poate sa intre in tabelul cu comenzi, unde poate vedea comanda introdusa de el.
6. Dupa ce clientii cer nota de plata, acesta apasa butonul de “Total order”, iar nota de plata va fi calculata.
7. Chelnerul poate sa intre din nou in tabelul cu comenzi pentru a vedea ca apare in tabel si pretul final.
8. Apoi acesta genereaza bonul pe care le-o poate inmana clientilor.

Secvente alternative:

1. Chelnerul introduce numele unei mese la care s-a luat deja comanda.
2. Ii va aparea un panou in care il atentioneaza ca fie la masa respective s-a luat deja comanda, fie produsele compuse pe care le-a introdus nu exista in meniu.
3. Chelnerul va introduce din nou numele mesei si produsele comandate, pana cand ambele vor fi valide.

**4.Proiectare**

4.1.Structuri de date

Ca si structure de date am folosit o lista de <MenuItem> in care am salvat produsele compuse din meniu, o lista de <BaseProduct> in care am salvat produsele de baza si un hashmap pentru a retine comenzile. In structura hashmap, cheia este formata din obiectele de tip Order, pentru care metoda hashCode() a fost suprascrisa pentru a calcula valoarea hash din atributele clasei Order. Valoarea din aceasta structura este reprezentata de lista de <MenuItem> care reprezinta produsele compuse care sunt comandate.

4.2.Impartirea pe clase si pachete

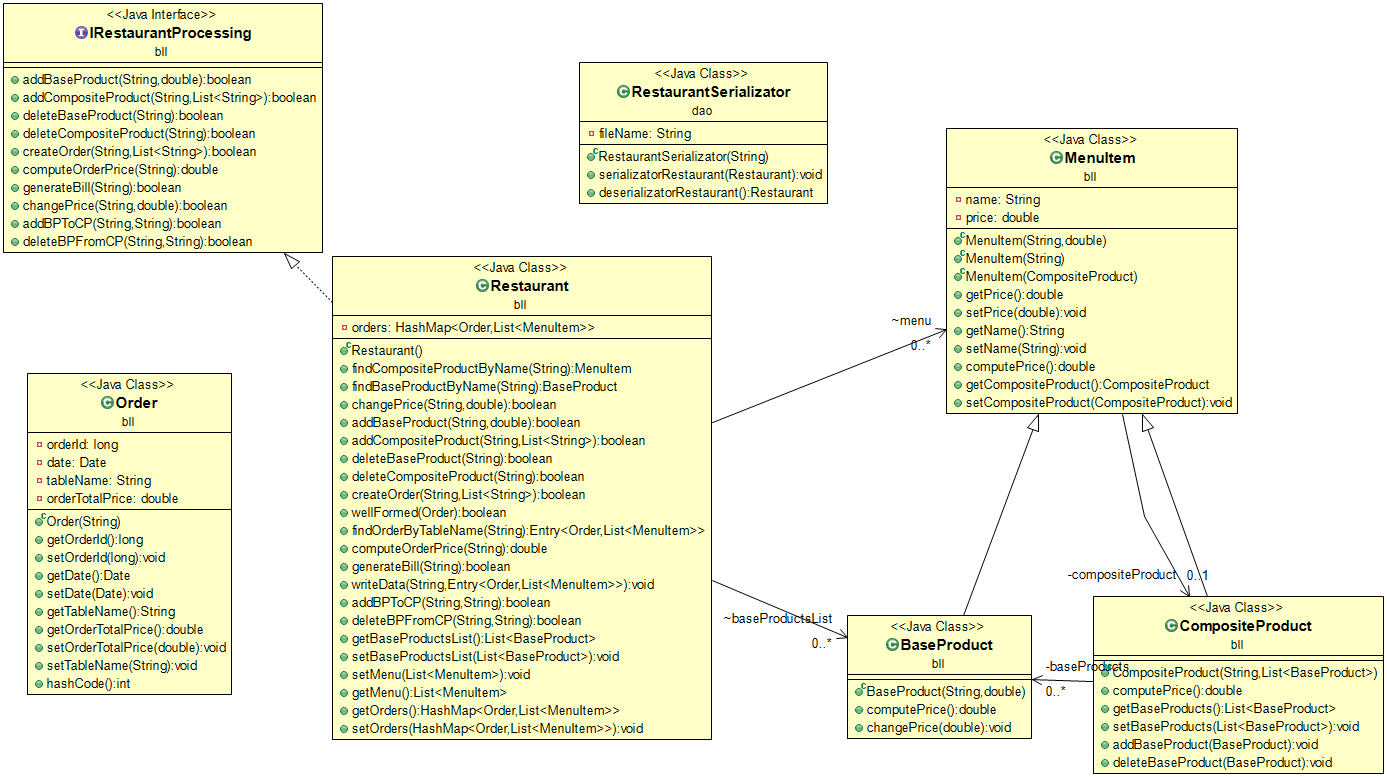
Am folosit o structura 3-tier combinat cu o structura Model-View-Controller.

In pachetul “bll” am creat clasele “BaseProduct”(reprezinta produsul de baza), “CompositeProduct”(reprezinta produsul compus din produse de baza), “MenuItem”(reprezinta produsul compus, cel pe care clientii il pot comanda), “Order”(reprezinta comanda clientilor), “Restaurant”(reprezinta restaurant propriu-zis) care implementeaza interfata “IRestaurantProcessing” creat tot in acest pachet.

In pachetul “dao” am creat clasa “RestaurantSerializator” prin intermediul careia datele restaurantului sunt extrase sau actualizat atunci cand se fac modificari(se introduce produse noi, se adauga comenzi noi etc.).

In pachetul “presentation” am creat mai multe clase care corespunde reprezentarii grafice , adica ce interafctioneaza cu utilizatorii. Pentru fiecare utilizator am creat o interfata diferita. Toate comenzile pe care un utilizator le poate face in cadrul acestei aplicatii, au fost implementate in clase diferite. De exemplu cee ace se vede prin adaugarea de produse de baza de catre un administrator este implementata in clasa “AddBaseProductPanel”, iar modul in care acesta operatie este controlata este implementata in clasele “GraphicAdministratorController” si “AdministratorController”.

4.3.Diagrama de clase



4.4.Dezvoltarea algoritmilor

Pentru fiecare comanda pe care o pot face utilizatorii am creat o metoda separata. De fiecare data s-a verificat daca datele introduse de catre utilizatori sunt valide.

De exemplu atunci cand administratorul vrea sa introduca un nou compus de baza, prima data se verifica daca acel produs exita deja, in caz contrar, produsul este adaugat cu success.

Pentru validarea datelor am verificat daca acestea se potrivesc unui anumit model. De exemplu atunci cand se doreste sa se adauge un nou preparat format din produse de baza, trebuie sa se introduca un string cu numele produselor de baza, cu virgula intre ele. In cazul in care nu este respectat acest model, va aparea un panou de atentionare, in care i se spune utilizatorului sa introduca din nou.

4.5.Interfate utilizate

La acest proiect am creat interfata “IRestaurantProcessing” care contine operatiile de baza pe care le pot executa utilizatorii. In plus am folosit interfata “Serializable” unde obiectul “Restaurant” poate fi reprezentat ca si o secventa de biti care include atat datele cat si si informatii despre tipul obiectului, si tipurile datelor stocate in obiect. Dupa ce un obiect serializat a fost scris intr-un fisier, prin metoda de deserializare, obiectul poate fi recreat in memorie.

4.6.Interfata utilizator

Atunci cand aplicatia este rulata, se deschide o fereastra, in care utilizatorul are posibilitatea fie sa vizualizeze meniul, fie sa se log-eze.

Administratorul poate sa efectueze urmatoarele comenzi:

* Adaugarea produselor de baza
* Adaugarea produselor compuse
* Stergerea produselor de baza
* Stergerea produselor compuse
* Schimbarea pretului unui produs de baza
* Adaugarea unui produs de baza intr-un preparat existent (extra)
* Stergerea unui produs de baza dintr-un preparat existent (“pretentii” ale clientilor)

Chelnerul poate sa efectueze urmatoarele comenzi:

* Vizualizarea tuturor comenzilor efcetuate
* Adaugarea unei noi comenzi
* Calcularea totalului de plata pentru o anumita comanda
* Generarea bonului pentru o anumita comanda

**5.Implementare**

5.1.Clasa MenuItem

Clasa MenuItem are ca si atribute numele articolului din meniu, pretul acestuia si un obiect de tip „CompositeProduct” care reprezinta practic preparatul din meniu, adaugat cu scopul de a retine produsele de baza din care este compus.

In aceasta clasa, am generat Getters si Setters pentru fiecare atribut.

Clasa Monomial implementeaza interfata „Serializable”, deoarece acest obiect este un atribut din clasa „Restaurant”, acesta din urma fiind supus operatiilor de serializare si deserializare.

5.2.Clasa Order

Clasa Order are ca si atribute un id de tip long(reprezinta ultimele 5 cifre din hashCode-ul acestei clase), data la care s-a realizat comanda, numele mesei unde s-a luat comanda si pretul total al acelei comenzi.

Aceasta clasa implementeaza interfata “Serializable” facand parte din clasa “Restaurant”.

Am suprascris metoda “hashCode()” prin intermediul careia se calculeaza valoarea hash cu ajutorul atributelor acestei clase.

5.3.Clasa Restaurant

Aceasta clasa are ca si atribute o lista de <MenuItem> in care sunt salvate produsele compuse din meniu, o lista de <BaseProduct> in care sunt pastrate produsele de baza care intra in alcatuirea compuselor de baza, si o structura de tip HashMap, a carei cheie o reprezinta obicetul de tip Order, iar valoarea o reprezinta o lista de <MenuItem>, adica produsele care au fost comandate.

In aceasta clasa au fost implementate toate operatiile care ulterior vor fi folosite de clasa “RestaurantController”(aceasta clasa face parte din pachetul “presentation” si care are scopul de a extrase si decodificate datele, iar apoi sunt transmise claselor care se ocupa cu afisarea rezultatelor.

Aceasta clasa, impereuna cu clasa “RestaurantController” implementeaza interfata “IRestaurantProcessing”, ambele implementand metodele specificate in interfata.

**public** **boolean** addBaseProduct(String name, **double** price);

**public** **boolean** addCompositeProduct(String name, List<String> baseProducts);

**public** **boolean** deleteBaseProduct(String productName);

**public** **boolean** deleteCompositeProduct(String productName);

**public** **boolean** createOrder(String tableName,List<String> orderedProducts);

**public** **double** computeOrderPrice(String tableName);

**public** **boolean** generateBill(String tableName);

**public** **boolean** changePrice(String name, **double** price);

**public** **boolean** addBPToCP(String name,String toAdd);

**public** **boolean** deleteBPFromCP(String name, String toDelete);

5.4.Clasa RestaurantSerializator

Aceasta clasa are ca si atribut numele fisierului unde si de unde datele clasei “Restaurant” sunt adaugate/extrase. Aici am implementat doua metode care realizeaza serializarea si deserializarea restaurantului.

5.5.Clasele specifice vizualizarii

Clasele care se ocupa de ceea ce utilizatorii vad sunt:

* “AddBaseProductPanel” – contine doua textField-uri unde trebuie introduse numele produsului de baza si pretul acestuia
* “AddBPToCPPanel”-contine doua textField-uri unde sunt introduse numele produsului de baza care trebuie adaugat intr-un produs compus deja existent, si numele acestui preparat
* “AddCompositeProductPanel”-trebuie introduse numele noului preparat si produsele de baza din care acesta este compus
* “AddMenuItemPanel”-contine trei butoane: unul prin care poti merge la fereastra precedenta, iar celelalte doua prin care alegi ce fel de produs vrei sa introduce, compus sau de baza, iar in functie de alegerea facuta, se va deschide o noua fereastra(am mentionat mai sus ce afiseaza ferestrele respective)
* “AddOrderPanel”-se introduce numele mesei de la care s-a luat comanda si lista de produse comandate
* “AdministratorMainPanel”-prezinta 4 butoane: “Add Menu Item” (se deschide fereastra implementata de clasa “AddMenuItem”), “Delete Menu Item” (se va prezenta mai jos), “Edit Menu Item” (se va prezenta ma jos), si “Cancel” (se merge la fereastra anterioasa, corespunzatoare clasei “LogInPanel” care va fi prezentata mai jos)
* “ChangePricePanel”-se introduce numele produsului a carui pret se doreste sa se modifice si noul pret
* “DeleteBaseProductPanel”-se introduce numele produsului care trebie sters
* “DeleteBPFromCPPanel”-se introducele numele produsului de baza care se sterge dintr-un produs compus, si numele acestuia din urma
* “DeleteCompositeProductPanel”-se introduce numele preparatului care se doreste sa se stearga din lista de meniu
* “DeleteMenuItemPanel”-este fereastra prin care alege ce fel de produs se doreste sa se stearga: de baza sau compus
* “EditMenuItemPanel”-fereastra in care se alege una dintre urmatoarele butoane “Change price”, “Add base product to CP”, “Delete base product from CP” sau “Cancel”
* “GenerateBillPanel”-se introduce numele mesei unde se doreste sa se genereze nota de plata, iar un fisier in format .txt este generat
* “LogInPanel”-fereastra de unde utilizatorul alege fie sa intre la el in cont, fie sa vizualizeze lista de meniu
* “MenuMainPanel”-de aici se alege daca se doreste sa se vizualizeze produsele de baza din cadrul restaurantului, sau produsele compuse care pot fi comandate
* “TotalOrderPanel”-se introduce numele mesei unde se doreste sa se calculeze nota de plata
* “WaiterMainPanel”-aceastra fereastra este folosita de catre chelner si prezinta 5 butoane “Orders”(prin care se poate vedea toate comenzile effectuate), “Add order”, “Total order”, “Generate bill”, “Cancel”

5.6.Clasele specifice felului in care aplicatia este controlata

Prin intermediul acestor clase, sunt extrase datele de la utilizatori, sunt validate si decodificate, iar apoi rezultatele sunt transmise utilizatorilor. Acestea sunt:

* “AdministratorController”-prin acesta clasa, se controleaza ceea ce administratorul doreste sa vada sau sa efectueze, sunt validate datele si trimite rezultatele
* “ApplicationController”-prin acesta clasa se alege cine foloseste aplicatia(administrator sau chlener) sau se poate afisa meniul
* “GraphicAdministratorController”-prin intermediul acestei clase se realizeaza conexiunea dintre control si afisare(aici pentru utilizatorul administrator)
* “GraphicController”-aceasta clasa face dintre clasa “ApplicationController” si clasa “LogInPanel”
* “GraphicMenuController”-conexiunea dintre “MenuMainPanel” si “MenuController”
* “GraphicWaiterController”-conexiunea dintre contul chelnerului si ceea ce vede acesta
* “MenuController”-se controleaza ceea ce se doreste sa se afiseze in acesta parte
* “RestaurantLogicController”-prin acesta clasa se face legatura dintre restaurantul propriu-zis si fisierul de unde datele sunt deserializate, si tot aici datele sunt decodificate si rezultatele sunt effectuate
* “WaiterController”-prin aceasta clasa, se controleaza cee ace chelnerul doreste sa vada sau sa efectueze, datele sunt validate si apoi rezultatele sunt transmise

**6.Concluzii**

In urma acestei teme, am invatat cum pot pastra date si apoi sa le folosesc ori de cate ori am nevoie, am invatat mai bine despre cum se implementeaza o interfata grafica folosind mai multe frame-uri.

Se pot adauga/modifica multe lucruri pentru ca acesta aplicatie sa fie mai eficienta, ba chiar sa poate fi folosita de catre vreun restaurant. Asadar, cateva dintre dezvoltarile pe care le-as putea face sunt: clientii sa poate comanda si produse de baza(de exemplu: suc, apa, paine) pentru a nu mai fi nevoie sa le introducem in produse compuse, sa adaugam produse de baza extra pentru un anumit client, neadaugandu-le in produsul compus, atunci cand se doreste sa se calculeze nota de plata sa se genereze automat si bonul.

**7.Bibliografie**

1. <https://www.tutorialspoint.com/java/java_serialization.htm>
2. <http://zetcode.com/tutorials/javaswingtutorial/>
3. <https://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html>
4. <https://www.vogella.com/tutorials/JavaRegularExpressions/article.html>