***Aplicatie de management a unui restaurant***

***-Assigment 4-***

Student : Brehuescu Andrei

Grupa: 30224

Facultatea CTI RO,an II

Contents

[1. Obiectivul temei 4](#_Toc39566768)

[1.1 Obiectivul principal 4](#_Toc39566769)

[1.2 Obiective secundare 4](#_Toc39566770)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 5](#_Toc39566771)

[2.1 Cerinte functionale 5](#_Toc39566772)

[2.2 Use-case 6](#_Toc39566773)

[2.3 Use-case Diagram 6](#_Toc39566774)

[3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii packaged, algoritmi, interfata utilizator). 7](#_Toc39566775)

[3.1 Deczii de proiectare 7](#_Toc39566776)

[3.2 Utilizare si proiectare 7](#_Toc39566777)

[3.2 Diagrama UML 8](#_Toc39566778)

[3.2 Proiectarea claselor, pachete, interfete 8](#_Toc39566779)

[3.3 Interfata utilizator 8](#_Toc39566780)

[4.Implementare 9](#_Toc39566781)

[4.1 Pachetul Run 9](#_Toc39566782)

[4.2 Pachetul DAL 9](#_Toc39566783)

[4.3 Pachetul BLL 9](#_Toc39566784)

[4.4 Pachetele Presentation.Views si Presentation.Controllers 10](#_Toc39566785)

[5. Rezultate si utilizare 11](#_Toc39566786)

[5.1 Rezultate 11](#_Toc39566787)

[6. Concluzii si devoltari ulterioare 11](#_Toc39566788)

[7. Bibliografie 11](#_Toc39566789)

# Obiectivul temei

## Obiectivul principal

Principalul scop al acestei teme este proiectarea si implementarea unei aplicatii de gestiune a activitatilor si a resurselor dintr-un restaurant. Aplicatia stocheaza informatii despre produsele disponibile in meniu cat si toate modificarile ce apar intr-un fisier cu extensia .ser .

Principalele functii ale aplicatiei au fost impartite pe roluri, Administrator, Ospatar, Chef Bucatar. Astfel in cadrul aplicatiei Administratorul are acces la stergere / adaugarea de produse noi, cat si de modificarea produselor existente. Rolul ospatarului permite crearea comenzilor,calculul pretului total al comenzii si generarea unei chitante cu informatii despre comanda.

## 1.2 Obiective secundare

Pentru proiectarea aplicatiei, problema este impartita in mai multe obiective secundare, acestea fiind implementate separat. Obiectivele secundare sunt:

* Crearea unei interfete grafice intuitive si cat mai usor de utilizat
* Struncturarea interfetei grafice folosind petternul MVC
* Structurarea claselor pe Layere, Bussines Layer, Data Access Layer, Presentation Layer
* Definirea claselor pentru modelul de date corespunzator unei transpunere a obiectelor din lumea reala ( produse, produse compuse, inregistrare din meniu )
* Stabilirea atributelor fiecarei clase
* Definirea unei clase abstracte pentru operatiile din cadrul restaurantului
* Definirea unei clase generale , Restaurant, responsabila de gestiunea tuturor activitatilor care implementeaza intefata ce contine operatiile necesare in cadrul restaurantului
* Crearea unei clase pentru serializarea si deserealizarea produselor disponibile
* Definirea unei clase pentru generarea de fisiere text pentru factura
* Procesarea comenzii create si generarea unei chitante intr-un fisier extern de tip text, aceasta continand informatii despre produsele comandate, totalul de plata, data calendaristica cand a fost emisa factura si numarul mesei.
* Notificarea Chef-ului cand o comanda a fost creata

# Analiza problemei, modelare, scenarii, cazuri de utilizare

## Cerinte functionale

Cerinte:

* Serializarea datelor pentru a salva datele din meniu la inchiderea aplicatiei
* Deserializarea datelor pentru a genera meniul conform cu datele ce au existat intr-o rulare anterioara a aplicatiei
* Posibilitatea de a insera atat produse simple cat si produse compuse ( ce pot fi formate din mai multe produse atat simple cat si compuse )
* Stergerea produselor compuse care contin un produs simplu atunci cand acesta a fost sters
* Updatarea tabelelor dupa fiecare modificare pentru a oferi vizibilitate pentru toate rolurile din cadrul restaurantului
* Generarea fisierelor de tip text cu nume diferit pentru evitarea suprascrierei
* Definirea fiecarei interfete grafice pentru rolurile din cadrul restaurantului

Pentru a facilita si usura utilizarea acestei aplicatii, tabelele pot fi manipulate prin selectarea unei linii si actionarea butonului cu operatie dorita. De exemplu pentru stergerea unui produs, va fi selectat randul pentru produsul corespunzator de catre administrator si actionat butonul de stergere a produsului.

Pentru serializare si deserializare au fost folosite obiectele : ObjectInputSteam , ObjectOutputStream pentru a se face scrierea si citirea unei colectii de obiecte intr-un fisier de tipul SER File. Acest fisier este deschis pentru citere intr-un obiect de tipul FileInputStream si pentru scriere in FileOutputStream.

Pentru efectuarea corecta a operatiilor se vor tine cont de cateva aspect:

* Pretul produselor atunci cand acestea sunt inserate trebuie sa fie un numar real
* Numarul mesei la crearea comenzii sa fie un numar intreg
* In cazul in care unele produse au fost editate dupa ce o comanda a fost creata dar aceasta nu a fost finalizata prin crearea bonului, calcularea costului tota, ulterior editarii produselor nu genereaza un cost nou de plata

Am ales impartirea fiecarui rol cu actiunile pe care le poate face in meniuri diferite, aplicatia avand o singura fereatra cu un toolbar din care poate fi apasat butonul corespunzator rolului dorit. La actionarea butonului vor aparea butone pentru fiecare actiune ce poate fi executata corespunzator rolului selectat.

## Use-case

Datele de intrare:

Aplicatia va prelua date de intrare despre comanda si creareare / editarea produselor prin intermediu interfetei grafice.

Date de iesire:

Cand se cere generarea unei facturi corespunzatoarea unei comenzi va fi generat un fisier text cu numele : Bill\_<ID\_Order> in care vor fi informatii despre comanda efectuata, detalii despre produse si cantitatea comandata, suma de plata, data la care a fost efectuata comanda, numarul mesei si ID-ul comenzii.

## Use-case Diagram

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

## Deczii de proiectare

Programarea orientată pe obiecte reprezintă o paradigmă de transpunere a obiectelor din lumea reală în clase și pachete. In aceasta tema obiectele sunt impartite in layere, fiecare layer (Bussines , Data acces si Presentation ) este reprezentat de un pachet. Obiectele transpuse din lumea reala sunt: produsele (simple , compuse), restaurant( ce contine atat o lista a produselor(meniu) cat si comenzi si acctiuni de gestiune a restaurantului, MenuItem ce reprezinta un produs din meniu ce poate fi compus sau simplu. Transpunerea celor 3 roluri, Chef, Administrator, Waiter sunt transpuse prin 3 interfete grafice, cate una pentru fiecare rol, continand actiunile potrivite acelui rol.

Conform indicatiilor din laborator am ales ca stergerea unui produs sa duc si la stergerea produselor ce contineau produsul sters initial.

In fereastra Chef-ului sunt vizinile produsele comandate. In cerinta temei este specificat atat notificarea Chef-ului cand este comandat un produs composite, cat si notificare cand este creata o noua comanda. Am ales notificare Chef-ului cand este creata o noua comanda, pe interfata destinata acestuia vor fi vizibile toate produsele comandate si tipul acestora ( Base sau Composite).

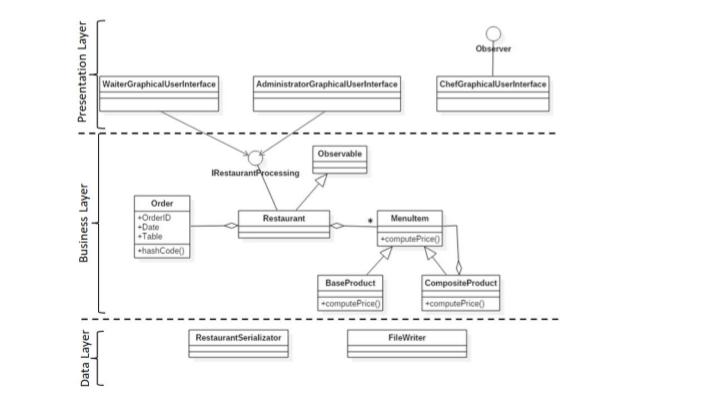
### 3.2 Utilizare si proiectare

Dupa ce aplicatia este lansata va fi deschisa o fereastra cu meniul principal. Din acel meniu se poate selecta rolul dorit dintre : Administrator ( gestioneaza produsele din restaurant ), Waiter (gestioneaza comenzile si genereaza chitantele corespunzatoare comenzilor ), Chef ( este notificat la crearea unei comenzi ).

In fereastra destinata Administratorului pot fi create produse noi prin apasarea butonului „Add New Item” care va afisa o fereastra noua pe ecran, in care se poate introduce numele produsului si selectra daca acesta este un produs compus sau simplu. Daca produsul este compus i se vor putea adauga si componente iar daca este simplu pretul. Odata setate detaliile despre produs se actioneaza butonul de confirmare pentru a se crea produsul. Pentru editarea produselor curente se selecteaza linia din tabel ce contine produsul si se actioneaza butonul de „Edit existing Item”. In functie de ce tip de produs a fost selectat va fi deschisa o fereastra corespunzatoare editarii : pentru produse simple ( se poate schimba numele sau pretul ) sau pentru produse compuse ( se poate modifica numele sau componentele, adaugarea de componente noi sau stergerea celor existente ). In cazul in care un produs compus ramane fara componente am considerat ca acesta sa fie sters. Ultimul buton din fereastra administratorului este butonul de stergere a unui produs. Acesta este intai selectat in tabel si apoi apasat pe buton. In cazul stergerii unui produs care se afla in componenta altor produse am considerat stergerea si acestor produse, ele fiind dependente de fiecare componenta.

In fereastra destinata Ospatarului, acesta are 3 operatii posibile reprezentate de cate in buton. Butonul pentru crearea comenzii creaza o fereastra in care poate fi introdus numarul mesei si selectate produsele ce depinde de acea comanda. Odata oferite aceste detalii se actioneaza butonul „Confirm” pentru crearea comenzii. Odata creata comanda pretul acesteia poate fi generat prin selectarea comenzii dorite in tabel si apoi apasarea butonului de Compute Price. Comenzile a caror pret a fost procesat pot fi finalizate prin actionarea butonui „Generate Bill” care creaza un fisier text cu detalii privind comanda ( data si ora la care a fost creata comanda, produsele si cantitatea acestora, pretul total , numarul mesei si ID-ul comenzii).

## 3.2 Diagrama UML



## Proiectarea claselor, pachete, interfete

Pachetele aplicatiei sunt: BLL, DAL, Run, Presentation.Controllers si Presentation.Views. Main-ul aplicatiei se afla in pachetul Run si are ca scop lansarea spre executie a programului prin afisarea pe ecran a interfetei grafice. Pachetul BLL contine datele transpuse in diferite obiecte (produse simplu, produs compuse, restaurant, etc ) si interfata ce contine operatiile corespunzatoare restaurantului. Pachetul DAL contine 2 clase ce se ocupa de manipularea datelor, respectiv serializare/deserializare si generarea fisierelor text ce contin date despre factura. Pachetul Presentation.Views contine definirea tuturor interfetelor grafice, fiecare clasa corespunde unui Frame, caruia ii sunt atribuite diferite butoane, campuri , tabele , etc. De controlul acestor interfete grafice se ocupa clasele din pachetul Presentation.Controllers unde sunt definite metodele de control al butoanelor, campurilor text, butoane radio etc. Astfel interfata grafica foloseste pattern-ul MVC.

## Interfata utilizator

Interfata utilizator are o singura fereastra la lansarea aplicatiei, in care sunt disponibile 4 butoane intr-un toolbar. Aceste butoane sunt: Administrator, Waiter, Chef si HomPage. Primele 3 butoane, la apasarea lor, genereaza in aceeasi fereastra un meniu corespunzator rolului selectat. Astfel pentru butonul de Administrator vor aparea 3 butoane pentru gestiune produselor in cadrul restaurantului si un panou de vizualizare a acestora. Pentru butonul Waiter, vor aparea in fereastra 3 butoane pentru creare unei comenzi noi, calcularea totalului pentru comanda selectata din tabel, generarea notei intr-un fisier text si in centrul ferestrei un tabel ce contine toate comenzile existente in rularea curenta a aplciatiei. Butonul Chef contine notificari cu privire la comanda efectuata , iar butonul HomePage trimite din nou la fereastra principala .

# 4.Implementare

## 4.1 Pachetul Run

**Clasa MainClass**

Clasa MainClass contine doar metoda main() in care este definita o instanta la clasa de control a interfetei grafice a restaurantului. Astfel aplicatia este lansata in executie prin afisarea ferestrei generale a restaurantului.

## 4.2 Pachetul DAL

**Clasa FileWriter**

Clasa FileWriter contine 2 atribute: order, cu informatii privind comanda, si items , ce reprezinta o list cu produsele comandate in cadrul comenzii. Clasa contine un constructor care salveaza comanda si obiectele comandate in obiect si o metode **write()** care realizeaza creaza si scrierea intr-un fisier text a unui raport cu detalii despre comanda ( ce produse au fost comandate, totalul de plata si data la care a fost creata comanda) .

**Clasa RestaurantSerialization**

Clasa RestaurantSerialization contine 2 metode, **write(items)** si **read()**. Metoda **read()** returneaza un HashSet<MenuItem>, aceasta metoda realizeaza deserializarea meniului iar lista returnata va fi utilizata de Controllere pentru afisarea meniului. Metoda **write(items)** primeste ca parametru un HashSet<MenuItem> si realizeaza salvarea acestor produse pentru o executie viitoare a aplicatie, este realizata serializarea produselor, instante ale clasei MenuItem. Aceste metode, atat pentru serializare cat si pentru deserializare, iau fiecare obiect si il convertesc intr-o secventa de biti pentru a fi scrisi intr-un fisier , sau sunt cititi si convertiti inapoi in obiecte.

## 4.3 Pachetul BLL

**Clasa MenuItem**

Clasa MenuItem este o clasa abstracta si transpune inregistrarile dintr-un meniu de restaurant. Aceste inregistrari ( produse) pot fi simple sau compuse ( contin la randul lor alte produse simple sau compuse ). Clasa abstracta contine ca atribute un nume si un pret. Clasa ofera implementare metodei **hashCode(),** aceasta fiind folosita la serializarea si deserializarea obiectului.

**Clasa BaseProduct si CompositeProduct**

Clasele extind clasa MenuItem, clasa BaseProduct contine implementarea metodei **computePrice**(), returnand pretul produsului. Clasa CompositeProduct are ca atribut suplimentar un HashSet<MenuItem>, ce reprezinta o colectie de produse din care este compus respectivul CompositeProduct. Implementarea metodei **computePrice**() fiind diferita, aceasta calculeaza suma preturilor tuturor componentelor si returneaza costul total.

**Interfata Irestaurant si Clasa Restaurant**

**Interfata Irestaurant** contine antetul metodelor de manipulare a meniului restaurantului respectiv: adaugare unui produs nou, actualizarea unui produs existent, stergerea unui produs, crearea unei comenzi , calculul totalului unei comenzi si generarea raportului despre comanda. Acesta interfata este implementata de clasa Restaurant care ofera implementare pentru fiecare metoda.

**Clasa Restaurant** are ca atribute un HashSet<MenuItem> care reprezinta produsele existente in meniu si un HashMap<Order, HashSet<MenuItem>> ce reprezinta o colectie ce contine comenzi, iar fiecarei comenzi ii este atribuita o colectie de produse comandate. Constructorul clasei realizeaza deserializarea pentru a initializa meniul cu produse create la rulari anterioare a aplicatie.

Metoda createMenuItem(item) primeste ca parametru un produs ce se doreste a fi inserat in meniu. Metoda verifica daca nu exista deja un produs cu acelasi nume pentru a realiza inserarea.

Metoda deleteMenuItem(item) primeste ca parametru un produs, acesta urmand a fi eliminat definitiv din lista meniului.

Metoda editMenuItem(item1, item2) primeste ca parametru 2 produse din meniu, aceasta metoda va inlocui primul produs cu al 2lea oferit ca paramentru. Aceasta functie este o functie de inlocuire dar in cadrul aplicatie este folosita ca o metoda de actualizare a datelor.

Metoda createOrder(tableNumber, orderedItems) creaza o noua comanda pentru masa introdusa ca parametru cu lista de produse comandate. Crearea unei noi comenzi are ca efect notificarea Chef-ului indicant toate produsele comandate pentru comanda curenta.

Metoda generateBill(order) apeleaza metoda de generare a facturii din clasa FileWriter, prin intermediul acestei metode este creat fisierul Text ce contine informatii despre comanda efectuata.

Metoda computePrice(order) este folosita pentru calculul total al pretului unei comenzi , pret total necesar pentru crearea bonului fiscal. Astfel o generare cu succes a unui bon depinde de metodele createOrder(), computePrice() si in final generateBill().

# 4.4 Pachetele Presentation.Views si Presentation.Controllers

Clasa RestaurantFrame este fereastra principala a acestei aplicatii, ea contie cele 3 butoane pentru selectia rolurilor si respectiv pentru fiecare rol un set de butoane cu functiile corespunzatoare. Aceasta clasa defineste doar afisajul grafic, actiunile butoanelor sunt definite in clasa RestaurantFrameController care adauga un controller pentru fiecare buton din aceasta fereastra principala.

In clasa RestaurantController sunt definite clase ce implementeaza ActionListener, cate una pentru fiecare buton. Pentru butoanele de selectie a rolului, clasele au ca functionalitate aducerea in panoul principal a butoanelor cu operatii pentru rolul corespunzator. Odata vizivbile butoanele pentru fiecare rol, acestea vor putea fi utilizitate, unele vor crea frame-uri noi ce corespund cate unui controller de exemplu pentru adaugarea unui nou produs sau editarea unui produs existent, in cadrul administratorului, dar si in cadrul ospatarului pentru creare unei comenzi noi.

In clasa RestaurantController sunt si patru metode pentru afisarea tabelelor dupa fiecare modificare asupra comenzilor sau a produselor. Aceste tabele vor aparea , in cadrul rolului de administrator un tabel cu toate produsele, iar in cadrul ospatarului un tabel al comenzilor existente.

Pentru usurarea utilizarii interfetei grafice am introdus diferite mesaje de avertizare in cazul in care se incearca stergerea unui produs dar acesta nu a fost selectead sau actionarea butoanelor ce depind de selectii din tabel fara ca liniile sa fie selectate, de asemenea si mai multe verificari pentru evitarea erorilor ( nu se poate adauga la un produs compus produsul insusi).

# 5. Rezultate si utilizare

## 5.1 Rezultate

Bill for order : 1

Table : 1

Products:

Salata Varza 4.0 x1 : 4.0 RON

Friputra 20.0 x2 : 40.0 RON

Cartofi prajiti 5.0 x1 : 5.0 RON

Ursus 6.0 x2 : 12.0 RON

Coca-cola 5.0 x4 : 20.0 RON

Total : 81.0 RON

Data : Tue May 05 11:28:06 EEST 2020

# 6. Concluzii si devoltari ulterioare

Din aceasta tema am invatat sa creez o aplicatie pornind de la o diagrama de clase, adaptand codul la cerintele furnizate prin intermediul diagramei. De asemenea am reusit sa combin pattern-ul MVC cu structurarea datelor pe layere.

Ca dezvoltari ulterioare as adauga un buton pe selectia mesei pentru ospatar si legarea la aplicatiei cu o noua componenta, vizibila de clienti la fiecare masa care prin apasarea unui buton sa poata notifica un ospatar sa le preia comanda.

# 7. Bibliografie

* <https://www.geeksforgeeks.org/serialization-in-java/>
* <https://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html>
* <https://javarevisited.blogspot.com/2012/01/what-is-assertion-in-java-java.html>
* <https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html#tag>
* <https://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html>