RESTAURANT MANAGEMENT SYSTEM

Stefan Tudor

Grupa 30229

Profesor laborator: Mitrea Dan

Cuprins:

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

2.Obiective………………………………………………………...

3.Proiectare………………………………………………………..

4.Implementare……………………………………………………

5.Rezultate………………………………………………………...

6.Concluzii………………………………………………………..

7.Bibliografie……………………………………………………..

**1.CERINTE FUNCTIONALE**

In aceasta tema se cere implementarea unei aplicatii responsabile pentru managementul unui restaurant.Acest lucru presupune operarea sub trei moduri: administrator, chelner sau bucatar.Administratorul este responsabil pentru crearea meniurilor, stergerea produselor din meniuri si editarea acestora.Chelnerul este responsabil de crearea unei chitante care contine numele produselor comandate de client si preturile acestora cat si de crearea unei comenzi in care se vor afla produsele dorite.Bucatarul va fi responsabil de gatirea produselor.Acesta va primi o notificare cu produsele pe care acesta trebuie sa le gateasca.

Aplicatia se va deschide si se vor afisa cele doua moduri sub care utilizatorul doreste sa opereze(Administrator sau chelner).

Pentru a incarca datele restaurantului din fisierul “restaurant.ser” se va apasa pe butonul “LOAD RESTAURANT DATA”.

Dupa utilizarea aplicatiei se va apasa “SAVE RESTAURANT DATA” pentru a salva datele restaurantului si a le incarca in fisierul “restaurant.ser”.

**Pentru modul administrator** se poate alege una dintre operatiile pe care utilizatorul doreste sa le faca: “Create”, “Edit”, “Delete” ,”See products”.

**Daca se opteaza pentru “Create”** atunci utilizatorul va fi intampinat de o noua interfata grafica.Pentru a crea un obiect de tipul “BaseProduct” se va introduce in JTextField-uri numele si pretul produsului.

Pentru a se crea un obiect de tipul “CompositeProduct” se va introduce numele acestuia si se va apasa butonul “Create composite product”.Mai apoi se va introduce numele unui “BaseProduct” deja creat si se va apasa butonul “Add base product to composite product”.

Pentru a se crea un obiect “CompositeProduct” care poate sa contina mai multe obiecte de tipul “CompositeProduct”( un obiect ce poate sa simuleze un meniu format din mai multe produse de tipul “CompositProduct”) se va proceda astfel: se va introduce in JTextField numele unui “meniu” si se va apasa butonul “Create menu item”.Apoi se va introduce in campul responsabil de numele unui “CompositeProduct” deja existent , numele unui “CompositeProduct” deja existent si se va apasa butonul “add composite product”.

**Daca se opteaza pentru optiunea “Edit” ,** utilizatorul va fi intampinat cu o interfata noua in care acesta poate sa editeze anumite produse deja existente.

Pentru a verifica daca un obiect de tipul “BaseProduct” exista se va introduce numele acestuia in JTextField si se va apasa butonul “Check for base product”.Pentru a edita numele si pretul acestuia, numele din JTextField-ul anterior nu se va sterge, se va lasa in field.Pentru a introduce numele nou al produsului se va introduce in JTextField-ul cu JLabel “Edit base product name” numele nou al produsului.Pentru a edita pretul se va introduce un pret nou in JTextField.Dupa ce au fost introduse datele necesare, se va apasa butonul “Edit base product”.

Pentru a verifica daca un obiect de tipul “CompositeProduct” exista se va introduce numele acestuia in JTextField si se va apasa “Check for composite product”.In caz afirmativ acesta va putea fi editat.Dupa ce a fost confirmat ca obiectul exista, se va lasa numele acestuia in JTextField, nu va fi sters.Pentru a adauga un obiect de tipul “BaseProduct” se va introduce numele unui produs de baza deja existent si se va adauga prin intermediul butonului “Add base product”.Pentru celelalte optiuni se va opera in acelasi mod(Numele produsului de tip “CompositeProduct” nu va fi sters dupa verificarea existentei).

**Daca se opteaza pentru optiunea “Delete” ,** utilizatorul va fi intampinat de o interfata in care va aparea un JTextField in care se va introduce numele unui produs deja existent pentru a fi sters.Daca se sterge un produs de tipul “BaseProduct”, se vor sterge toate obiectele de tipul “CompositeProduct” care il contin.Dupa ce se introduce numele, se va apasa butonul “DELETE PRODUCT”.

**Daca se opteaza pentru optiunea “SEE PRODUCTS”,** se vor afisa toate produsele din restaurant intr-un JTabel.

**Pentru modul “Chelner”,** utilizatorul va fi intampinat de interfata responsabila de operarea sub modul “chelner”.

**Pentru a crea o comanda noua** se va introduce ID-ul comenzii, data comenzii sub forma de String, numarul mesei si un produs pe care clientul doreste sa il comande.Daca clientul doreste sa comande mai multe produse, nu se vor sterge datele comenzii (ID,data,numarul mesei) din JTextField-uri.Se va scrie in JTextField-ul responsabil pentru adaugarea unui nou produs, numele unui produs deja existent si se va apasa butonul “ADD PRODUCT TO ORDER”.

**Pentru a calcula pretul total al comenzii** nu se vor sterge din JTextField-uri datele comenzii(ID,data,numarul mesei) si se va apasa butonul “COMPUTE TOTAL PRICE”.

**Pentru a genera o chitanta** nu se vor sterge din JTextField-uri datele comenzii(ID,data,numarul mesei) si se va apasa butonul “GENERATE BILL”.

**Pentru a se afisa toate comenzile** se va apasa butonul “SHOW ORDERS”.

**Pentru a se trimite comanda bucatarului** nu se vor sterge din JTextField-uri datele comenzii(ID,data,numarul mesei) si se va apasa butonul “FINISH ORDER AND SEND TO CHEF”. Dupa ce a fost finalizata comanda, utilizatorul va fi intampinat de interfata grafica a bucatarului, interfata ce simuleaza bonul cu produsele ce trebuie sa fie gatite de catre bucatar.

**2.OBIECTIVE**

Obiectivul acestei teme este de a implementa un sistem ce poate sa permita lucrul a mai multor tipuri de utilizatori(Administrator,Chelner,Bucatar).

***Obiectivele secundare*** si atingerea lor realizeaza indeplinirea obiectivului principal.

Aceste obiective secundare sunt:

* Alegerea structurilor de date necesare
* Impartirea claselor
* Dezvoltarea unor anumiti algoritmi ce pot ajuta la o buna functionare a programului
* Implementarea unei solutii
* Testarea programului

**3.PROIECTARE**

Ca si prim pas de proiectare al acestei aplicatii am ales sa imi impart interfata grafica in mai multe interfete grafice pentru fiecare tip de utilizator si apoi sa le implementez.Am creat clasele “BaseProduct” si “CompositeProduct” care extend clasa abstracta “MenuItem”.Apoi am implementat clasa “Restaurant”, ce are in componenta sa o structura de tipul ArrayList<MenuItem> in care vor fi tinute produsele restaurantului.Tot in aceasta clasa se afla inca o structura de date, anume HashMap<Order,ArrayList<MenuItem>>.Aceasta structura este responsabila de memorarea comenzilor clientilor.Obiectul de tipul order din interiorul acestei structuri va fi folosit ca si cheie pentru a accesa obiectul de tipul ArrayList<MenuItem>(produsele comenzii respective).La final am decis ca pentru serializarea produselor restaurantului, se va folosi clasa “RestaurantSerializator”.

**Pachetele** folosite in acest proiect:

**BusinessLayer** care contine clasele: BaseProduct, CompositeProduct, interfata IRestaurantProcessing, MenuItem, Order, Restaurant.

**DataLayer** care contine clasele: FileWriterClass si RestaurantSerializator.

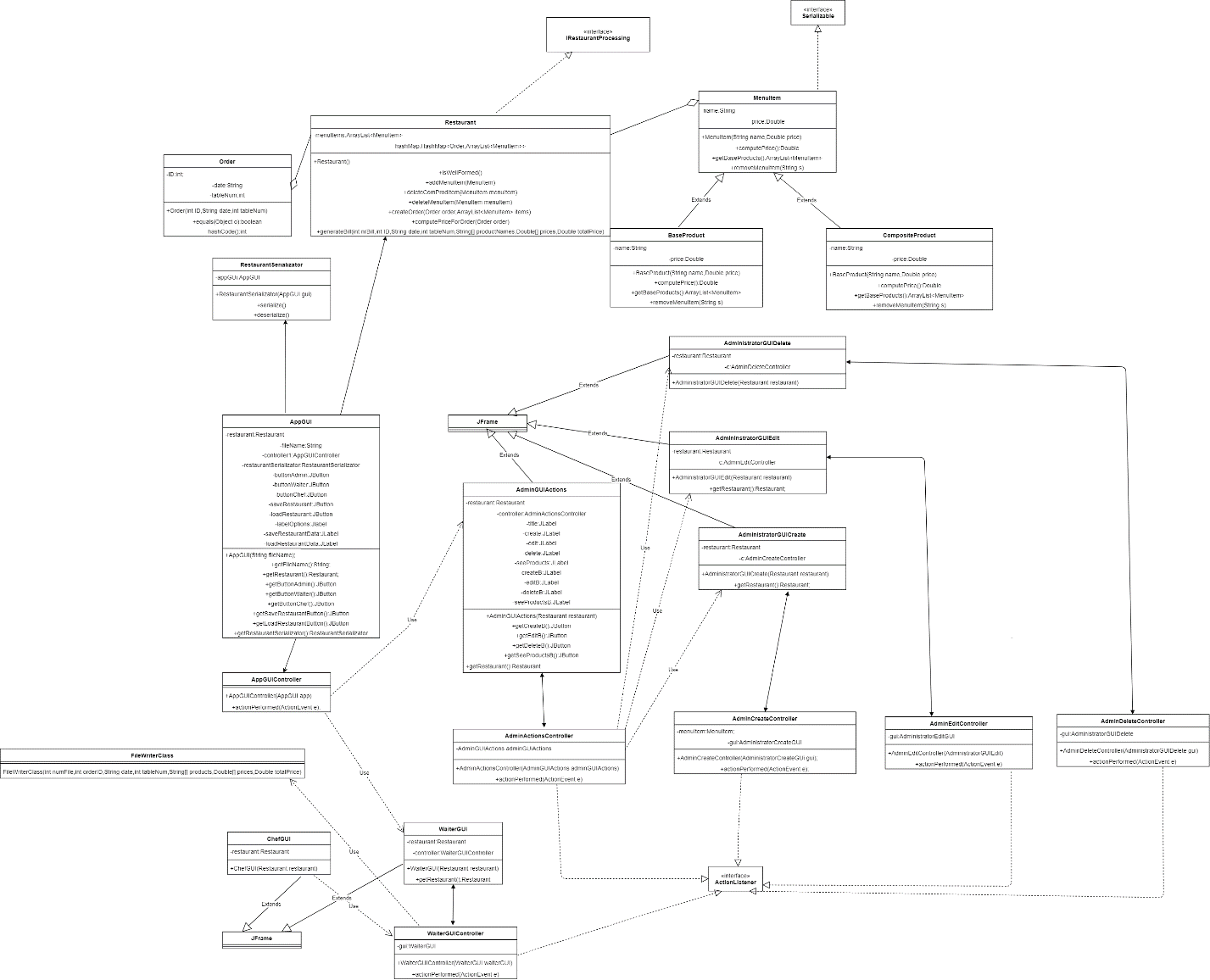
**PresentationLayer** care contine pachetele Administrator(contine view-uri si controllere pentru utilizatorul Administrator), Chef(Contine view-ul pentru utilizatorul Chef), Waiter( contine view-uri si controllere pentru utilizatorul Waiter).

**Interfetele** folosite in acest proiect:

* IRestaurantProcessing
* ActionListener
* Serializable
* Observer

**Diagrama UML:**

In continuare va fi prezentata diagrama UML in care sunt descrise relatiile intre clase, intre clase si interfete.



**4.IMPLEMENTARE**

Ca si prim pas de implementare au fost construite clasele “BaseProduct”, “CompositeProduct” si “MenuItem” , in care sunt continute numele produselor si preturile acestora.

**Metode importante** din aceste clase:

* computePrice() , metoda ce calculeaza pretul unui produs.
* getBaseProducts() metoda ce returneaza un obiect de tipul ArrayList<MenuItem>, care contine produsele unui obiect de tipul “CompositeProduct”
* removeMenuItem(String s), metoda ce primeste ca si parametru un String ce reprezinta numele unui produs ce va fi scos din lista de produse a unui “CompositeProduct”.

**In clasa “AppGUI”** a fost construita interfata de start a aplicatiei.Aceasta contine un obiect de tipul “Restaurant”, unul de tipul “RestaurantSerializator” , un String cu numele fisierului folosit pentru serializare/deserializare si un controller pentru interfata propriu-zisa.Controller-ul pentru aceasta interfata(clasa AppGUIController), este implementata in interiorul clasei “AppGUI”.

In pachetul “Administrator” din pachetul PresentationLayer se afla clasele : **“AdminGUIActions”, “AdministratorGUICreate”, “AdministratorGUIEdit”,”AdministratorGUIDelete”.**Aceste clase sunt responsabile pentru interfetele grafice ale operatiile de creare,stergere sau editare.Tot in acest pachet se afla si controllerele acestor interfete.

In clasa **“AdminActionsController**” :

In metoda actionPerformed(ActionEvent e): in cazul in care sursa evenimentului este butonul de creare, atunci se va crea o noua interfata responsabila pentru crearea de obiecte.

In cazul in care se apasa butonul de “edit”, se va crea o interfata responsabila pentru editare.

In cazul in care se apasa butonul de “delete”, se va crea o interfata responsabila pentru stergerea produselor.

In cazul in care se apasa butonul de “SEE PRODUCTS” se va crea o interfata ce va contine un JTabel cu numele si preturile produselor.Acest lucru se realizeaza prin intermediul metodelor “getProducts()” si “getPrices()” din clasa “Restaurant”, metode ce vor returna toate numele produselor cat si preturile acestora.Aceste valori vor fi stocate in vectori ce vor fi folositi pentru crearea tabelului.

In clasa **“AdminCreateController”**: metodele cele mai importante care vor fi apelate se afla in clasa “Restaurant”. Aceasta clasa foloseste datele introduse in JTextField-urile din interfata “AdministratorGUICreate”.

Metodele mentionate sunt :

* addBaseProduct(MenuItem p) care adauga un produs de tipul “BaseProduct” la lista de produse din clasa “Restaurant”.
* checkForBaseProd(String s) care verifica daca un produs de baza se afla in lista de produse.In aceasta metoda se va parcurge lista de produse si in cazul in care se afla produsul cu numele dorit se va returna valoarea “true”.
* getCompositeProductWithName(String s) va cauta in lista de produse obiectul de tipul “CompositeProduct” cu numele dorit si il va returna.Aceasta metoda este folosita pentru returna obiectul de tipul “CompositeProduct” cu un anumit nume la care dorim sa adaugam un obiect de tipul “BaseProduct”.
* getBaseProductWithName(String s) va cauta in lista de produse obiectul de tipul “BaseProduct” cu numele dorit si il va returna.
* addMenuItem(MenuItem menuItem) adauga la lista de produse un obiect de tipul “MenuItem”.
* checkForMenuItem(String s) verifica daca in lista de produse exista un obiect de tipul “CompositeProduct” cu un anumit nume si care ar in componenta sa unul sau mai multe obiecte de tipul “CompositeProduct”.

In clasa **“AdminEditController”** este folosit acelasi principiu de cautare dupa nume a obiectelor de tipul “BaseProduct” si “CompositeProduct”, care a fost folosit si in clasa ”AdminCreateController”.Dupa ce sunt returnate obiectele cu numele dorite, se poate realiza editare proprietatilor acestora prin intermediul metodelor de tip setter. Dupa ce sunt efectuate operatiile de editare se va apela metoda computePrice() pentru a calcula pretul produsului.

In clasa “**AdminDeleteController**” este folosit acelasi principiu de cautare dupa nume a unui obiect de tipul “MenuItem”.In cazul in care este gasit, atunci se va apela metoda deleteMenuItem din clasa “Restaurant”, care va parcurge lista de produse si va sterge recursive produsele care contin produsul care se doreste a fi sters, cat si produsul propriu-zis.

In pachetul **“Waiter”** se afla clasa “WaiterGUI” care este responsabila pentru crearea interfetei pentru utilizatorul de tipul chelner.Tot in acest pachet se afla si clasa “WaiterGUIController”.

In clasa **“WaiterGUIController”** : se vor extrage din JTextField-uri datele unei comenzi si se va crea un obiect nou de tipul Order cu datele respective.Sunt folosite metode de cautare bazate pe principiul descris anterior, cat si metode pentru crearea unei comenzi(metoda se gaseste in clasa “Restaurant”).

Metoda **createOrder(Order order, ArrayList<MenuItem> items)**  creeaza un element nou in interiorul structurii HashMap a restaurantului cu cheia “order” si elemente comenzii “items”.

In cazul in care se doreste sa se adauge un produs la comanda respectiva cu datele (ID,data,tableNum) se va folosi un ArrayList<MenuItem> si se va scoate din lista de produse a restaurantului, produsul pe care dorim sa il adaugam.Odata ce acesta a fost scos, in lista creata se va pune lista de produse a comenzii respective.Dupa ce a fost obtinuta lista respectiva, putem adauga produsul in lista.

Metoda **computePriceForOrder(Order order)** va calcula pretul total al unei comenzi.Se trimite ca parametru un obiect de tipul “Order” si se va accesa lista de produse a comenzii respective.Se va parcurge aceasta lista si pentru fiecare element se va apela metoda computePrice().La final se va returna rezultatul adunarii preturilor produselor.

Metoda **generateBill()** ia datele unei comenzi, numele produselor, preturile produselor si pretul total al comenzii.In interiorul acestei metode se va afla si o variabila de tipul **FileWriteClass** care va genera un fisier.txt cu toate datele comenzii.

In cazul in care dorim sa terminam o comanda si sa o trimitem bucatarului, atunci se va crea un obiect nou de tipul “ChefGUI” care va fi adaugat la obiectul de tip “Restaurant” sub forma de “Observer”.Pentru a trimite comanda bucatarului se vor folosi metodele **sendOrderToChef(Order order)** si **displayOrder(Order order).**Prin intermediul acestor metode este pus in valoare “Observer Design Pattern”.De fiecare data cand se modifica o comanda acest lucru este trimis bucatarului prin intermediul metodelor **“setChanged()”** si **“notifyObervers(Object o)”.**

In clasa “Order” se gasesc ca si variabile instanta ID-ul comenzii, data comenzii si numarul mesei.Ca si metode ale acestei clase se regasesc: equals(Object o) , hashCode() si metode de tip getter si setter.

In interfata **IRestaurantProcessing** se afla metodele:

* addMenuItem(MenuItem menuItem)
* deleteCompProdItem(MenuItem menuItem)
* deleteMenuItem(MenuItem menuItem)
* createOrder(Order order,ArrayList<MenuItem> items)
* computePriceForOrder(Order order)
* generateBill(int nrBill,int ID,String date,int tableNum,String[] productNames,Double[] prices,Double totalPrice)

Aceste metode au fost implementate in clasa “Restaurant” si au fost explicate pe parcursul capitolului 4.

In clasa **“Restaurant”** se gaseste structura ArrayList<MenuItem> si structura HashMap<Order,ArrayList<MenuItem>>, structure responsabile de comenzi si de lista de produse. Metodele din aceasta clasa au fost explicate pe parcursul capitolului 4.

In pachetul **“DataLayer”** se afla clasele:

* FlieWriteClass: clasa care este responsabila de scrierea chitantelor pentru fiecare comanda in parte.
* RestaurantSerializator care prezinta metodele: serialize(), metoda responsabila pentru serializarea obiectelor in fisierul “.ser”.La final se afla metoda deserialize(), care este responsabila pentru deserializarea obiectelor din fisierul “.ser” si punerea lor spre folosire.

**5.TESTARE**

Ca si mod de testare se va folosi fisierul “restaurant.ser” si se vor efectua operatiile descrise in documentatie asupra obiectelor din interiorul acestuia.Se vor sterge obiecte, se vor crea obiecte si se vor edita obiecte.Se vor crea comenzi in interfata chelner.

**6.CONCLUZII SI DEZVOLTARI ULTERIOARE**

Ca si dezvoltari ulterioare ale acestui proiect as avea in vedere implementarea unei interfete grafice mult mai estetice.

Mai mult, se poate implementa un sistem de login pentru un numar mult mai mare de angajati. Se poate implementa si o optiune de chat prin care angajatii sa comunice intre ei.

**7.BIBLIOGRAFIE**

<https://stackoverflow.com/>

<https://objectcomputing.com/resources/publications/sett/september-2011-design-by-contract-in-java-with-google>

<https://www.w3schools.com/>

<https://www.tutorialspoint.com/java/index.htm>