

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

SISTEM DE GESTIONARE A UNUI RESTAURANT

Documentatie Tema 4

Elev: Ţoc Roxana-Ştefania

Grupa: 30224

Profesor indrumător: Antal Marcel

Cuprins:

1.Obiectivul temei

Obiectivul principal

2.Analiza problemei, modelarea si cazuri de utilizare

3.Proiectare

4.Implementare

5.Rezultate

6.Concluzii, dezvoltari ulterioare

7. Bibliografie

1.Obiectivul temei

Obiectivul principal

Proiectul are ca scop implementarea unei aplicatii de management a unui restaurant. Sistemul trebuie să aibă trei tipuri de utilizatori: administrator, chelner si bucatar. Administratorul poate adăuga, șterge și modifica datele existente a produselor din meniu. Chelnerul poate crea o nouă comandă pentru o masă, poate adăuga elemente din meniu și calculează factura pentru o comandă. Bucătarul este notificat de fiecare dată când trebuie să gătească alimente care sunt comandate printr-un chelner

2.Analiza problemei, modelarea si cazuri de utilizare

2.1 Analiza problemei

Pentru a simplifica aplicația, puteți presupune că sistemul este utilizat de un singur administrator, un chelner și un singur bucătar și că nu este nevoie de un proces de conectare.

Aplicatia trebuie sa indeplineasca urmatoarele cerinte:

1. Definiți interfața IRestaurantaProcesing care conține principalele operații care pot fi executate de chelner/administrator, după cum urmează:
   * Administrator: Create New Menu element, Delete Menu element, Edit Menu element
   * Chelner: Creați o nouă comandă; prețul de calcul al unui ordin; generați factura în formatul .txt.
2. Definiți și implementați clasele din diagrama clasei indicată mai sus:
   * Utilizați modelul de design compus pentru definirea claselor MenuItem, BaseProduct și CompositeProduct
     + Utilizați modelul de proiectare al observatorului pentru a notifica bucătarul de fiecare dată când se adaugă o comandă nouă care conține un produs compus.
   1. Implementați clasa Restaurant utilizând o colecție JCF predefinită care se bazeaza pe o structura de date hashtablr. Cheia hashtable va fi generată pe baza ordinii de clasă, care poate avea asociate mai multe MenuItems. Utilizați JTable pentru a afișa informații despre restaurant.
      * Definiți o structură de tip Map<Order, Collection<MenuItem> pentru stocarea informațiilor referitoare la comandă în clasa Restaurant. Cheia Map-ului va fi formată din obiecte de tip comandă, pentru care metoda hashCode() va fi suprascrisă pentru a calcula valoarea hash în cadrul Map-ului din atributele comenzii (OrderID, data etc.).
      * Definiți o colecție corespunzătoare formată din obiecte MenuItem pentru a stoca meniul restaurantului.
      * Definiți o metodă de tip “bine format” pentru clasa Restaurant.
      * Implementati clasa Restaurant utilizant metoda Design by Contract (care implică condiții prealabile, post, invariabil și afirmații).
   2. Articolele de meniu pentru popularea obiectului Restaurant vor fi încărcate/salvate dintr-un/într-un fișier utilizând serializările.

2.2 Modelarea si cazuri de utilizare

In primul rand, se alege utilizatorul: administrator, chelner sau bucatar.

1. Administratorul - are optiunile : Adaugare BaseProduct, Adaugare CompositeProduct, Stergere produs si Editare produs.
2. Chelnerul – are optiunile : Creaza comanda, Calculeaza pret si Creaza bon.
3. Bucatarul – va fi anuntat de fiecare data cand se adauga o comanda si va primi un tabel cu produsele comandate.

Principii urmarite pe parcursul implementarii:

1. Abstractizarea: este procesul de grupare a datelor şi metodelor de prelucrare specifice rezolvării unei probleme. Totodata, exprimă toate caracteristicile esenţiale ale unui obiect care fac ca acesta să se distingă de alte obiecte si oferă o definire precisă a graniţelor conceptuale ale obiectelor din perspectiva unui privitor extern.
2. Incapsularea: ascunde detaliile implementării unui obiect. Gruparea datelor şi metodelor aplicabile acestora într-o singură structură de date, definind totodată modul în care obiectul şi restul programului pot referi datele din obiect. Concept care defineşte apartenenţa unor proprietăţi şi metode faţă de un obiect. Constă în separarea aspectelor externe ale unui obiect care sunt accesibile altor obiecte de aspectele interne ale obiectului care sunt ascunse celorlalte obiecte
3. Polimorfismul: Este proprietatea de a procesa obiectele în mod diferit, în funcție de tipul sau de clasa lor. Mai exact, ofera sansa de a redefini metode pentru clasele derivate.
4. Mostenirea: Mecanismul prin care o clasă preia structura (datele membre) şi comportamentul (metodele) unei alte clase la care adaugă elemente specifice. Clasă de baza este clasa de la care se preia structura şi comportamentul si clasa derivată este clasa care preia structura şi comportamentul.

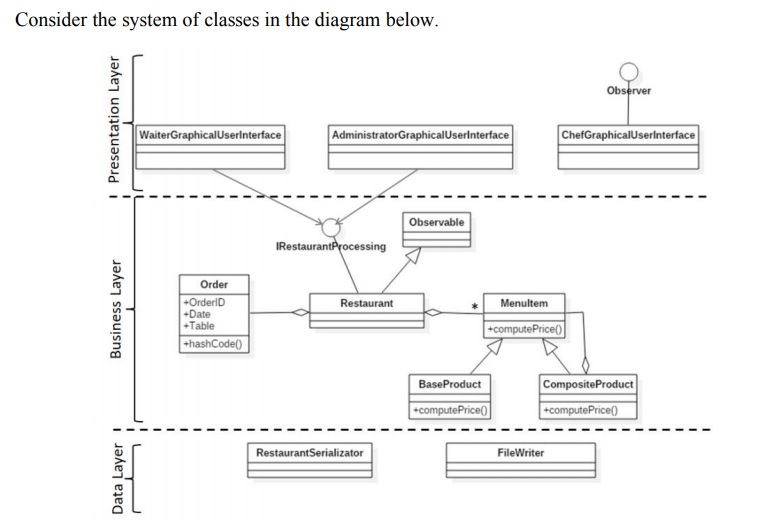
3.Proiectare

3.1 Structuri de date

Structurile de date care s-au folosit in principal sunt ArrayList si HashMap. Pentru a forma un CompositeProduct, dar totodata pentru a putea retine produsele existente in restaurant am utilizat ArrayList. Pentru stocarea comenzilor am utilizat HashMap pentru care se va utiliza HashCodul() implementat.

3.2 Diagrama de clase

Unified Modeling Language (UML) este un limbaj standard pentru descrierea de modele si specificatii pentru software. Este folosita pentru reprezentarea vizuala a claselor si a interdependentelor, taxonomiei si a relatiilor de multiplicitate dintre ele. Diagramele de clasa sunt folosite si pentru reprezentarea concreta a unor instante de clasa, asadar obiecte, si a legaturilor dintre acestea.



4.Implementare

Proiectul este structurat dupa arhitectura Layered Architectured care cuprinde pachetele: BusinessLayer, DataLayer, PresentationLayer, tema4.

Nivelul superior al proiectului se afla in pachetul PresentationLayer in care sunt descrise frame-urile pentru cei 3 utilizatori, care este format din clasa View si administrata de clasa Controller. Nivelul intermediar este pachetul BusinessLayer , unde se gasesc clasele model si in care sunt prelucrate informatiile primate de la utilizatori utilizand interata grafica. Nivelul inferior este reprezentat de pachetul DataLayer in care se realizeaza stocarea datelor din restaurant.

**Pachetul BusinessLayer**

Clasa MenuItem

Este o clasa abstracta care are ca argumente numele si pretul produselor. In continuare am realizat metodele getter si setter si in ultimul rand, metoda hashCode() pentru a calcula valorea hash din cadrul Map-ului. Aceasta clasa este mostenita de clasele BaseProduct si CompositeProduct, iar aceste clase preiau structura si comportamentul clasei MenuItem si sunt adaugate elemente specifice.

Clasa Order

Are ca argumente: idComanda, masa pentru care s-a facut comanda, data, pretul si o lista de MenuItem. Aceasta lista contine produsele selectate pentru comanda care pot fi BaseProduct sau CompositeProduct. Am creat un constructor pentru a initializa argumentele si am creat metodele corespunzatoare pentru gettere si settere cu ajutorul carora aceste attribute pot fi verificate si modificate in timpul procesarii informatiilor in celelalte clase.

Interfata IRestaurantProcessing

Cuprinde metodele pe care trebuie sa le indeplineasca utilizatorul si sunt implementate in clasa Restaurant.

Clasa Restaurant

Are ca argumente un obiect de tip RestaurantSerializator pentru a putea salva informatiile restaurantului prin serializare, un HashMap<> pentru a memora comenzile, un ArrayList<> pentru a stoca comenzile. Mai jos am implementat metodele de getter si setter pentru aceste argumente. Metodele care sunt declarate in interfata grafica sunt urmatoarele:

Metoda isWellFormed() verifica daca elementele pe care lucram sunt instante ale clasei Restaurant. Metoda adaugareProdus(MenuItem) se utilizeaza atunci cand administratorul doreste sa introduca in lista de produse un MenuItem care poate fi BaseProduct sau CompositeProduct. Totodata, am salvat informatiile restaurantului intr-un fisier restaurant.ser folosind serializarea. Metoda stergereProdus(String) realizeaza stergerea unui produs din lista de produse, acesta fiind identificat dupa nume. Si in acest caz, folosim serializarea pentru a putea salva modificarile facute asupra restaurantului. Metoda editareProdus(String, double) identifica produsul dupa nume si realizeaza editarea pretului pentru acel produs. La fel, am utilizat serializarea. Metoda creareComanda(Order, ArrayList) realizeaza introducerea in lista a unei comenzi. Metoda computePrice(int) calculeaza pretul unei comenzi, aceasta fiind identificata dupa id-ul sau. Ultima metoda, factura(int) identifica o comanda dupa id si i se genereaza un bon.txt care cuprinde fiecare produs din comanda cu pretul corespunzator, iar in final este suma totala a comenzii.

In final, am creat doua JTable-uri, unul pentru produse si unul pentru comenzi. Tabelul pentru produse memoreaza numele si pretul acestora. Tabelul pentru comenzi memoreaza id-ul comenzii, data in care aceasta a fost facuta, masa pentru care s-a facut si dupa ce se seteaza butonul de calculare a comenzii, va aparea si pretul acesteia.

**Pachetul DataLayer**

Are clasa FileWrite care are un index de tip static si il folosim pentru a contoriza bonurile emise pentru comenzi.

Clasa RestaurantSerializator are doua metode: serializarea si deserializarea produselor pentru a putea fi salvate. Am implementat functiile de read si write in fisierul restaurant.ser dat ca parametru la rularea programului.

**Pachetul PresentationLayer**

Clasa View

In primul rand, am creat un frame pentru Restaurant. In panel-ul panou se afla un buton pentru administrator, unul pentru chelner si unul pentru bucatar.

Frame-ul pentru Administrator contine un panel in care se vor pune operatiile pe care le face acest utilizator: buton pentru Adaugare BaseProduct, Adaugare CompositeProduct, Stergere produs, Editare produs. Mai contine un panel care contine datele produselor inserate: Nume Produs, Lista Componente atunci cand vrem sa adaugam un CompositeProduct vom insera produsele BaseProduct separate prin ‘,’ .si pretul produsului. Pentru aceste elemente am creat cate un JLabel si un JTextField.

Mai am un panel pentru tabelul in care se vor inregistra produsele adaugate.

Frame-ul pentru Chelner contine un panel in care se vor pune operatiile pe care le face acest utilizator: buton pentru Creaza comanda, Calculeaza pret, Creaza bon. Mai contine un panel care contine datele comenzii inserate: Componente comanda alese din tabelul de produse si se vor scrie separate prin ‘,’ si masa pentru care se face aceasta comanda si totodata un TextField in care introduce id comenzii pentru care vreau sa calculez pretul sau sa creez o factura. Ultimul panel este utilizat pentru a include tabelul de comenzi.

Frame-ul pentru Bucatar contine un panel care va contine un tabel cu produsele comandate pentru fiecare comanda.

Am creat un constructor in care pentru fiecare buton am adaugat ActionListener.

In continuare, am creat metodele de getter pentru toate atributele, pentru a avea acces la ele din clasa Controller. La final am implementat metoda creareMeniu() care reprezinta pagina initiala a programului de unde se alege utilizatorul pe care il dorim.

Clasa Controller

In primul rand, am creat un obiect view de tip View pe care l-am initializat in constructor. Am facut override la metoda actionPerformed si am implementat functionalitatea fiecarui buton.

Din meniul principal, daca se apasa butonul “Administrator” se deschide un frame nou care cuprinde detaliile prezentate mai sus cand am vorbit de clasa View cu observatia ca datorita serializarii, chiar daca inchidem fereastra principala, cand aceasta este deschisa din nou, datele introduce initial se pastreaza.

Cand administratorul doreste sa adauge un BaseProduct, trebuie introdus numele produsului in TextField-ul “Nume produs” si pretul acestuia in TextField-ul “Pret”, iar apoi se apasa butonul “Adaugare BaseProduct”, iar produsul va aparea in tabel. In cazul in care doreste sa adauge un CompositeProduct introduce in TextField-ul “Nume produs” cum doreste sa se numeasca produsul compus, iar in TextField-ul “Lista Componente” trebuie sa introduca cu ‘,’ o lista de BaseProduct din tabelul de produse si se apasa butonul “Adaugare CompositeProduct”, iar produsul va aparea in tabel. Daca introduce in aceasta lista un CompositeProduct, operatia va esua sau daca introduce un produs care nu este in tabel va aparea in consola mesajul “Produs indisponibil”. Daca doreste sa stearga un produs din tabel, trebuie introdus in TextField-ul “Nume produs” numele produsului pe care doreste sa il stearga si se apasa butonul “Stergere produs”, iar tabelul de produse se va actualiza de fiecare data. Cand se doreste sa se faca o editare asupra unui produs, se va scrie numele produsului din tabel pentru care vrem sa facem aceasta editare si noul pret pe care dorim sa il aiba acest produs si se apasa butonul “Editare produs”, iar modificarea va aparea in tabel.

Din meniul principal, daca se apasa butonul “Chelner” se deschide un frame nou care cuprinde detaliile prezentate mai sus cand am vorbit de clasa View. In plus, avem tabelul cu produsele existente si din care se creaza o comanda.

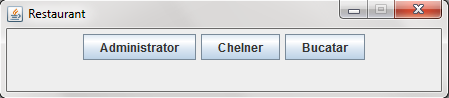
Cand chelnerul doreste sa creeze o comanda, insereaza in “Componente comanda” produsele, din tabelul de produse existent mai jos, care doreste sa le comande si se scrie “Masa” pentru care se creaza aceasta comanda, iar apoi se apasa butonul “Creaza comanda”. Daca acesta doreste sa calculeze pretul unei comenzi, insearaza in “Id comanda” id-ul comenzii din tabel pentru care doreste sa se calculeze pretul, iar acesta va aparea in tabel. Iar in ultimul rand, daca doreste sa creeze bonul pentru o comanda, se scrie in “Id comanda” id-ul comenzii din tabel.

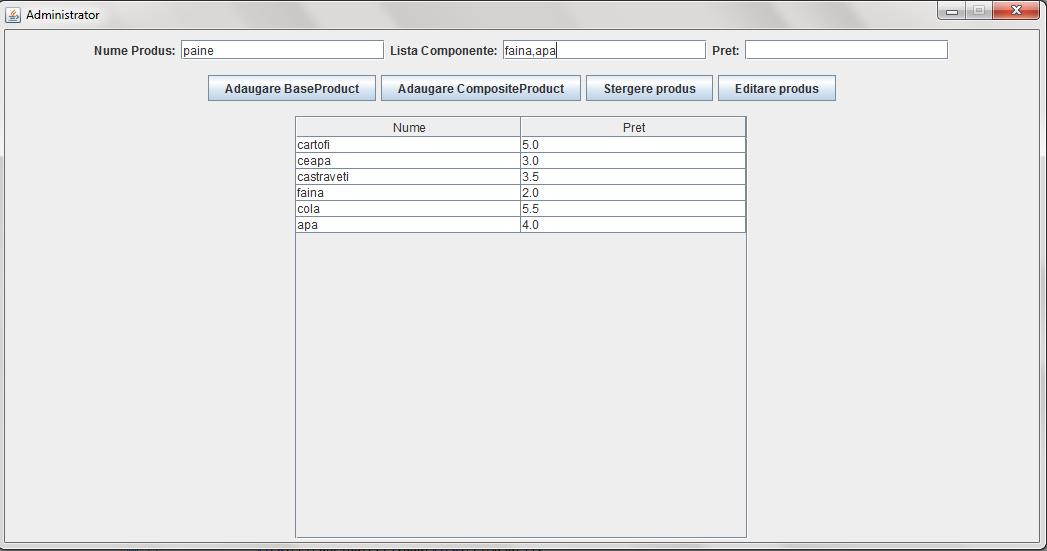
Din meniul principal, daca se apasa butonul “Bucatar” se deschide un frame nou care cuprinde detaliile prezentate mai sus cand am vorbit de clasa View si apar intr-un tabel produsele necesare pentru fiecare comanda.

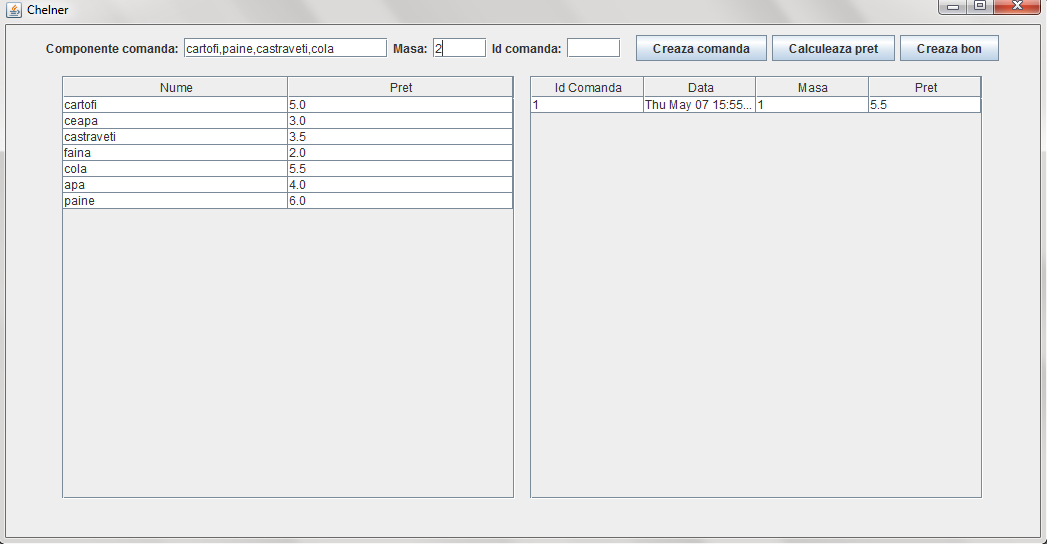
**Pachetul tema4**

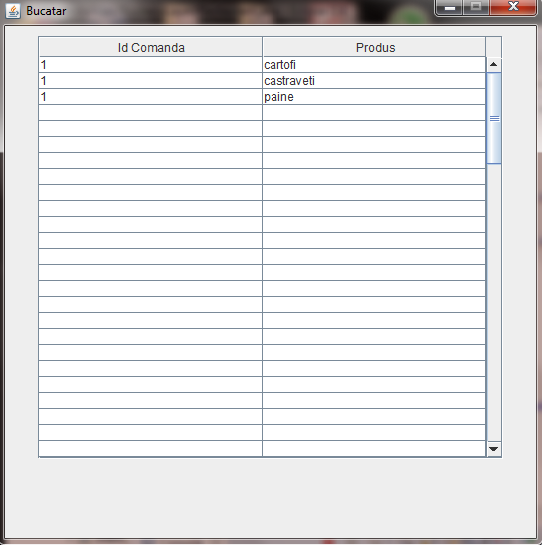
Cuprinde clasa Main cu metoda main(String[]) care seteaza ca argument fisierul creat in urma serializarii si creaza o noua instanta a clasei View().

5.Rezultate









6. Concluzii

In concluzie, prin intermediul acestei teme am reusit sa imi consolidez cunostintele despre arhitectura Layered Architecture, despre GUI, utilitatea claselor abstracte si a variabilelor statice. De asemenea am invatat serializarea datelor si lucrul cu tabele in interfata grafica.

1. Bibliografie

<http://coned.utcluj.ro/~salomie/PT_Lic/4_Lab/Assignment_4/Assignment_4.pdf>