UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

SISTEM DE MANAGEMENT

RESTAURANT

-TEMA 4-

Nume: Bolduț Denisa-Teodora

Grupa: 30225

Cuprins:

Tema 4:

Consider implementing a restaurant management system . The system should have three types of users: administrator, waiter and chef. The administrator can add, delete and modify existing products from the menu. The waiter can create a new order for a table, add elements from the menu, and compute the bill for an order . The chef is notified each time it must cook food ordered through a waiter .

1. Obiectivul temei

Obiectivul temei urmareste gestionarea operatiilor intampinate intr-un sistem precum cel al unui restaurant. Putem intalni aici mai multe functii, fiecare cu indatoriile sale. Dupa cum spune si cerinta, avem un administrator, un ospatar si un sef bucatar. Sarcinile administratorului se leaga de gestiunea produselor existenta, acesta poate sa le vizualizeze intr-un tabel, poate sa editeze un produs, poate sa stearga un produs existent si in cele din urma poate sa adauge la lista deja existenta un produs nou introducand numele acestuia. Problema des intalnita in aplicatiile de zi cu zi, acest proiect prezinta o posbila rezolvare a acestor indatoriri. Cu privire la ospatar, acesta are putin mai mult de lucru. El trebuia sa introduca o camanda noua, acest lucru presupunand ca trebuie sa introduca un id pentru comanda, data in care s-a inregistrat comanda, numarul mesei si sa poata vizualiza produsele existente (asa zisul meniu cu produse al restaurantului) si sa selecteze ce doreste clientul. Aceasta comanda trecandu-se intr-un tabel nou specific comenzilor. Pe langa acestea el trebuie sa calculeze pretul final al comenzii in functie de produsele selectate (vom intra in amanunt mai tarziu) si in cele din urma sa

genereze un bon fiscal care sa cuprinda detaliile privind comanda respectiva. Seful bucatar va fi notificat ori de cate ori s-a inregistrat o comanda care necesita combinarea a cel putin doua produse de baza. Acesta va avea disponibila o fereasta pentru a-l instiinta ce produse trebuie sa adauge pentru a gati si a combina ingredientele astfel incat clientul sa fie servit cu ceea ce doreste. Acestea sunt obiectivele pe care le urmarim, mai departe vor fi descrise fiecare atribuire si modul de implementare a functiilor stabilite pentru fiecare rol.

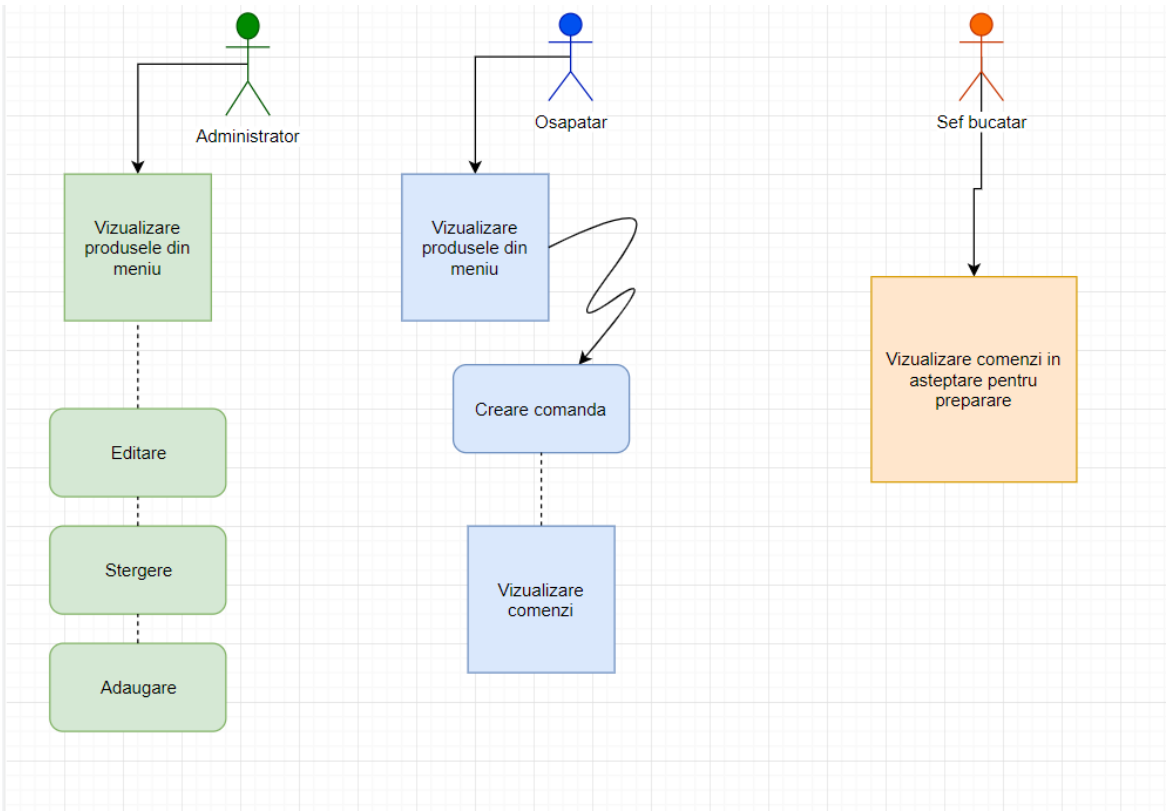
2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Din ceea ce deja am observat ca avem nevoie sunt acele produse pe care le vrem in meniu. Pentru a ne pastra si actualiza aceasta lista se foloseste conceptul de serealizare si deserealizare, acestea fiind stocate sau salvate dintr-un fisier. In momentul in care se incarca rularea aplicatiei, are rol serealizarea produselor in fisier si deseralizarea acestor pentru a putea fi vizualizate in tabel. Acestea vor putea fi disponibile in tabelul administratorului. Acestea poate sa selecteze un anumit produs si sa il editeze, apasand dupa pe butonul de salvare editare. Pentru a sterge un produs va fi nevoit sa il selecteze inainte si mai apoi sa apese pe butonul de stergere selectat. La dorinta acestui adminstrator de introduce un nou produs, el va fi nevoit sa introduce in chenarul destinat denumirea produsului si sa apese pe butonul de adugare. Acest nou produs va putea fi deja vizualizat in tabelul de produse fara alt efort.

Ospatarului i se vor afisa produsele existente pentru a putea crea o comanda. Acesta va fi nevoit sa introduce id – ul comenzii, data comenzii si numarul mesei si apoi sa apese pe butonul de incarcare comanda. Aceste

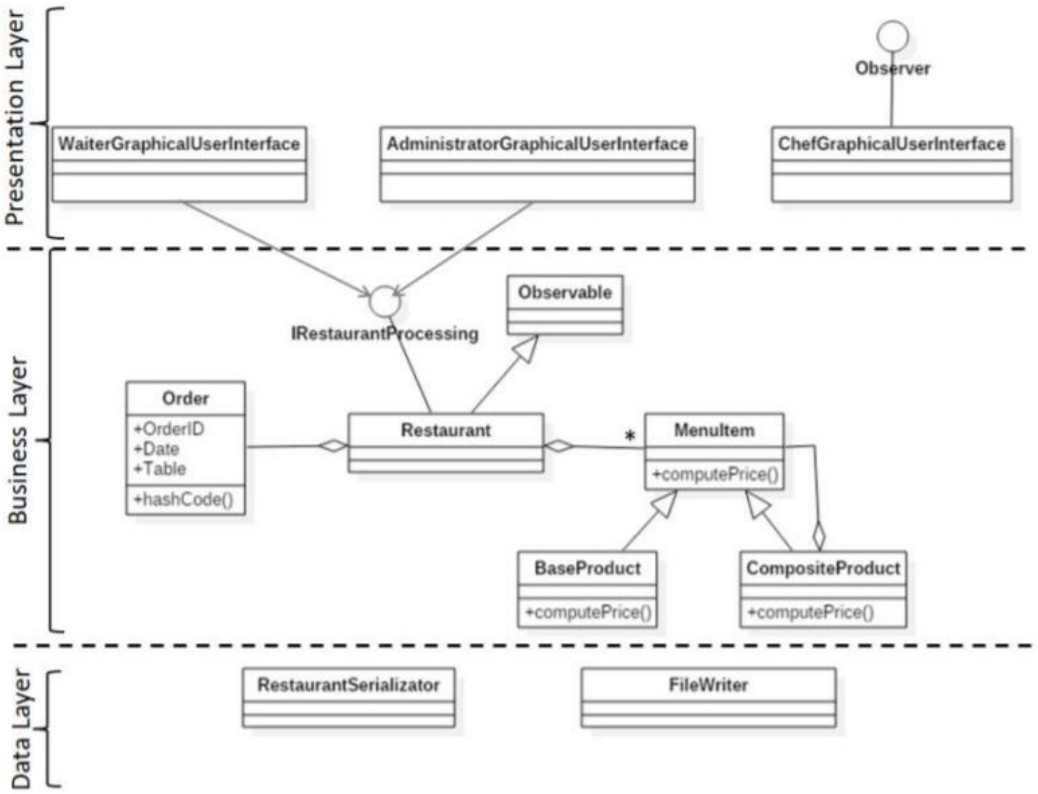
detalii introduse deja vor putea fi vizualizate in tabelul destinat comenzilor. Pentru a finaliza comanda, ospatarul va alege din tabelul produselor cele dorite. Pentru generare bon fiscal, este nevoie ca sa existe o comanda si cand apasa pe butonul specific se va genera un fisier txt.

Seful bucatar tot ce are de facut este sa observe. Acesta va fi instiintat cu ajutorul Observer Design Pattern cand este inregistrat o noua coamanda care contine un produs compus din mai multe ingrediente pentru a putea sti sa il gateasca. Diferenta dintre un produs de baza si unul compus si calcularea pretului final se va face cu ajutorul Composite Design Pattern .

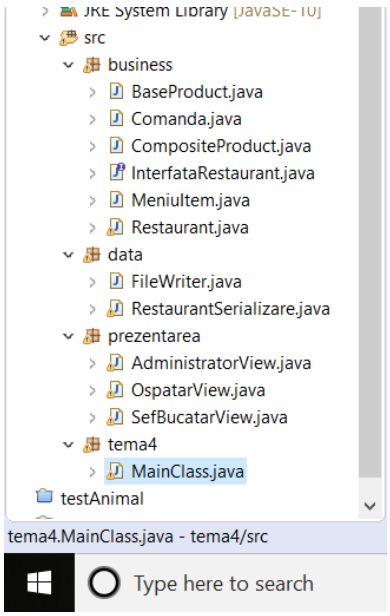


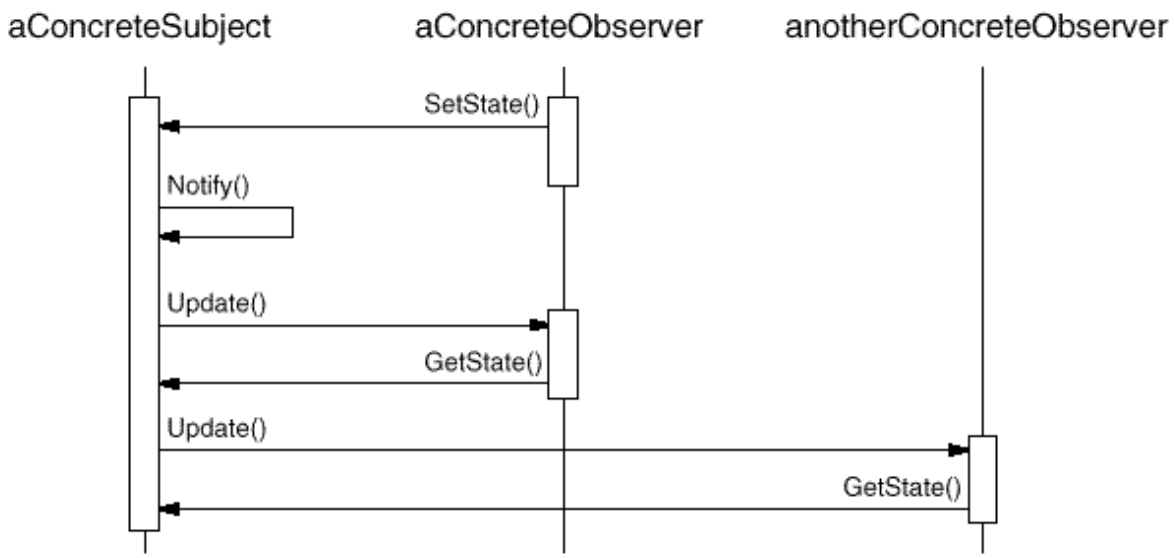
3. Proiectare

DIAGRAMA UML



Structuri de date & PROIECTARE CLASE





Un observator cere schimbarea starii subiectului. Acesta anunta toti observatorii ca trebuie sa-si actualizeze starea in concordanta cu noua sa stare. Pentru actualizarea starii, fiecare observator cere noua stare a subiectului.

> Notificarea nu este intotdeauna facuta (apelata) de subiect. Poate fi apelata de un observator sau de un alt tip de obiect. Diferitele variatii sunt discutate in sectiunea Implementare.
>
> Acesta este un sablon care ne ajuta sa implementam ceea ce tine de sef bucatar.

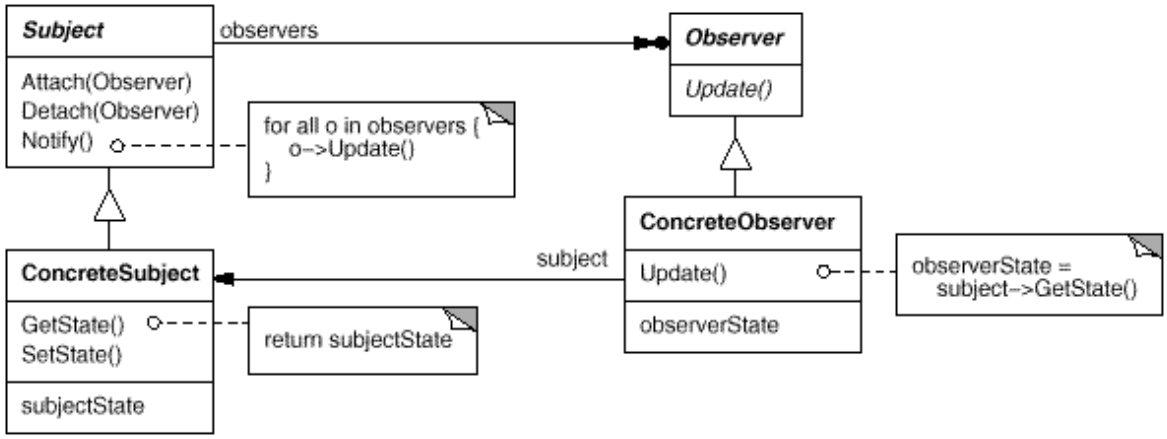
Ce este un sablon de proiectare?

Un sablon de proiectare descrie o problema care se intalneste in mod repetat in proiectarea programelor si solutia generala pentru problema respectiva, astfel incat sa poata fi utilizata oricand dar nu in acelasi mod de fiecare data. Solutia este exprimata folosind clase si obiecte. Atat descrierea problemei cat si a solutiei sunt abstracte astfel incat sa poata fi folosite in multe situatii diferite.

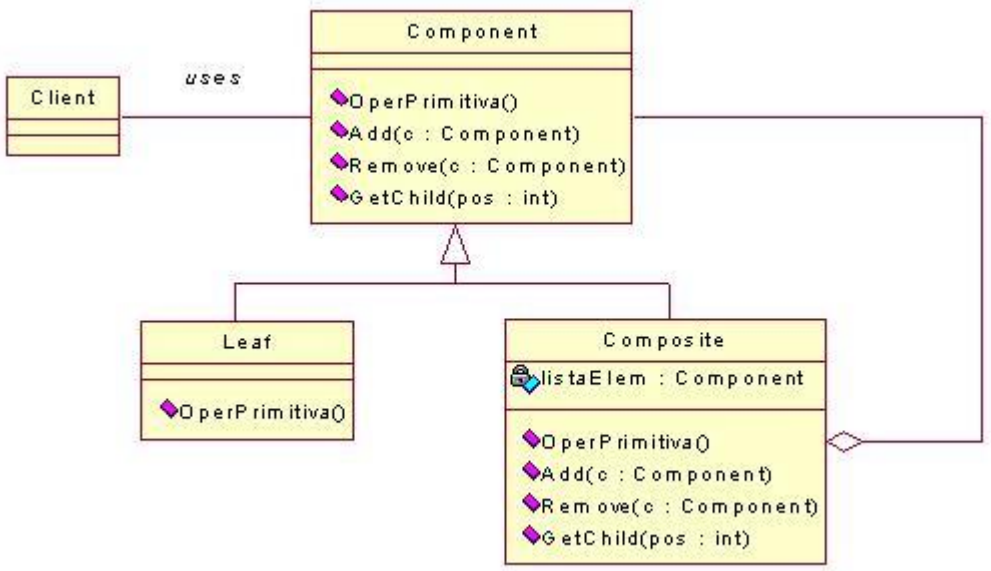
In general, un sablon are 4 elemente esentiale:

1. Un nume prin care poate fi referit. Prin atribuirea de nume sabloanelor de proiectare se creaza un vocabular de proiectare, un limbaj comun de comunicare si documentare. Alegerea numelui este importanta deoarece el devine parte din vocabularul de proiectare.
2. Descrierea problemei. Se explica problema si contextul in care apare, cand trebuie aplicat sablonul.
3. Descrierea solutiei. Sunt descrise elementele care compun proiectul, relatiile dintre ele, responsabilitatile si colaborarile.
4. Consecintele aplicarii sablonului. Descrierea consecintelor este critica pentru evaluarea alternativelor de proiectare, intelegerea costurilor si a beneficiilor aplicarii unui sablon.

La sablonul Observem avem urmatoarea diagrama:



Iar pentru stocarea menultem-urilor se foloseste Patternul Composite



Unul din scopurile sablonului Composite este, asa cum s-a aratat mai sus, acela de a permite clientilor sa lucreze cu obiectele Leaf si Composite in mod uniform, clasa concreta de care apartin obiectele respective fiind transparenta. Pentru aceasta, clasa Component trebuie sa defineasca cat mai multe dintre operatiile ce pot sa apara inLeaf si Composite. De obicei, Component ofera implementari implicite la aceste operatii, urmand ca Leaf si Composite sa le redefineasca.

Pe de alta parte, aici apare un conflict cu unul dintre principiile proiectarii ierarhiilor de clase, care spune ca intr-o superclasa trebuie definite doar acele operatii care au sens pentru subclasele ei. Tinand cont de aceasta, se observa ca in Component avem anumite operatii care nu par sa aiba sens pentru Leaf (este vorba de operatiile de gestionare a elementelor unui container). Se pune problema daca se poate defini o implementare implicita pentru asemenea operatii. De exemplu, in cazul operatiei GetChild, daca vom considera ca un obiect Leaf este un caz particular de container, adica unul care nu are niciodata nici o componenta, putem sa impunem ca operatiaComponent::GetChild sa nu returneze nici o

componenta, urmand ca Leaf sa utilizeze aceasta implementare,
iar Composite sa o redefineasca corespunzator.

Principii OOP

Serializare

Utilizand fluxurile putem scrie aplicatii care salveaza si incarca datele in fisiere. Java permite si un mecanism mai avansat si anume serializarea obiectelor. In forma cea mai simpla serializarea obiectelor inseamna salvarea si restaurarea starii obiectelor. Obiectele oricarei clase care implementeaza interfata Serializable, pot fi salvate intr-un stream(fluxde date) si restaurate din acesta. Pachetul java.io continedoua clase specialeObjectInputStream respectiv ObjectOutputStream pentru serializarea tipurilor primitive. Subclasele claselor serializabile vor fi si ele serializabile. Mecanismul de serializare salveaza variabilele membri nestatice si netransiente. Daca un obiect este serializat, atunciorice alt obiect care contine o referinta la obiectul serializat va sisi el insusi serializat. Se poate serializa orice multime de obiecte interconectateintr-o structura de graf.

public interface Serializable

Aceasta interfata nu declara nici metode, nici atribute, serveste doar pentru identificarea obiectelor serializabile. Clasele care nu implementeaza aceasta interfata nu vor putea fi serializate sau deserializate. Operatia de deserializare presupune ca clasa obievtului poseda si un constructor cu vizibilitate publica si fara argumente. Atributele obiectului vor fi initializate dintr-un flux de date. In cazul serializarii unui obiect care apartine unui graf

de obiecte daca se ajunge la un obiect care nu implementeaza interfata Serializable, atunci se genereaza exceptia NoSerializableException .

In programarea procedurala, datele sunt descrise separat de prelucrari, astfel ca legatura dintre ele se realizeaza numai la nivel conceptual. Pentru tipurile de date primitive s-a adoptat conceptul, conform caruia prin tip de date se intelege o multime de valori, careia i se asociaza o multime de operatii care se pot efectua asupra valorilor respective. Constatam deci ca, inca de la inceputurile programarii in limbaje de nivel superior, s-a facut o legatura stransa intre date si operatii, legatura incorporata in fiecare limbaj de programare.

Un obiect care apartine unei clase se numeste si instanta a clasei (este o instantiere, o realizare particulara a clasei respective). De exemplu, clasa Barbat si clasa Femeie sunt subclase ale clasei Om. In schimb, Ion_Popescu este o instanta a clasei Barbat (un obiect care apartine acestei clase), iar Maria_Preda este o instanta a clasei Femeie.

Polimorfismul (engleza: polymorphism) permite ca aceeasi operatie sa se realizeze in mod diferit in clase diferite. Sa consideram, de exemplu, ca in clasa Figura_geometrica exista metoda arie(), care calculeaza aria figurii respective. Clasele Cerc, Triunghi, Patrat sunt subclase ale clasei Figuri_geometrice si vor mosteni, deci, de la aceasta metoda arie(). Este insa evident ca aria cercului se calculeaza in alt mod decat aria patratului sau cea a triunghiului. Pentru fiecare din instantele acestor clase, la calcularea ariei se va aplica metoda specifica clasei sale.

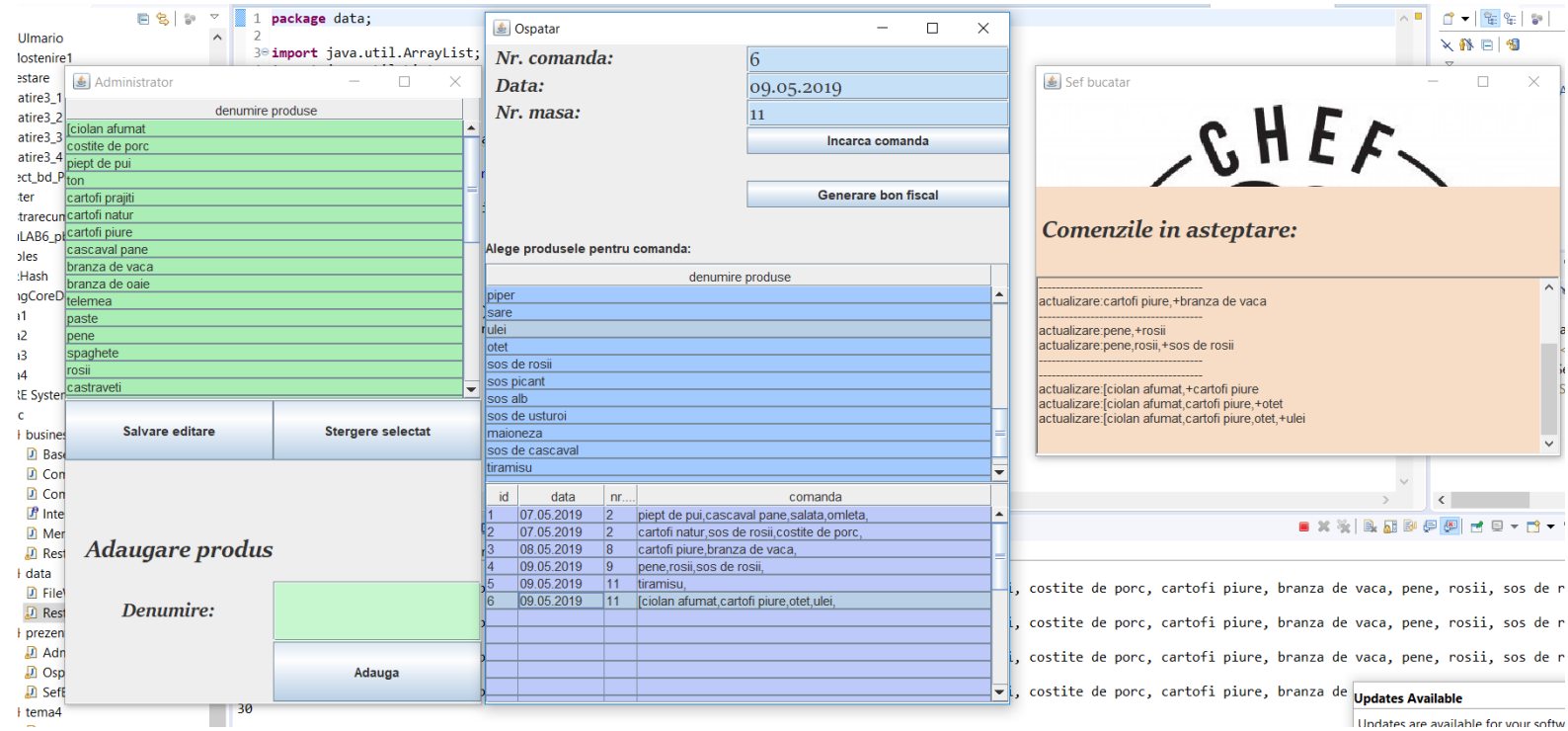
Incapsularea (engleza: encapsulation) este proprietatea obiectelor de a-si ascunde o parte din date si metode. Din exteriorul obiectului sunt accesibile ("vizibile") numai datele si metodele publice. Putem deci sa ne imaginam obiectul ca fiind format din doua straturi.

Agregarea (engleza: aggregation) este proprietatea obiectelor de a putea incorpora alte obiecte. Asa dar, "datele" continute intr-un obiect pot fi nu numai date primitive, ci si obiecte. Se pot astfel crea obiecte cu structuri din ce in ce mai complexe.

Mostenirea (engleza: inheritance) este proprietatea unei clase de a contine toate atributele (variabilele) si metodele superclasei sale. In consecinta, trecerea de la clasa la subclasa se face prin adaugarea de atribute si/sau de metode. De exemplu, clasa Barbat si clasa Femeie au ambele toate atributele clasei Om, dar fiecare din acestea are si atribute specifice.

4. Implementare

Interfata cu utilizatorul



Se pot observa indatoririle fiecarui utilizator al aplicatiei. Sarcinile administratorului se leaga de gestiunea produselor existenta, acesta poate sa le vizualizeze intr-un tabel, poate sa editeze un produs, poate sa stearga un produs existent si in cele din urma poate sa adauge la lista deja existenta un produs nou introducand numele acestuia. Problema des intalnita in aplicatiile de zi cu zi, acest proiect prezinta o posbila rezolvare a acestor indatoriri. Cu privire la ospatar, acesta are putin mai mult de lucru. El trebuia sa introduca o camanda noua, acest lucru presupunand ca trebuie sa introduca un id pentru comanda, data in care s-a inregistrat comanda, numarul mesei si sa poata vizualiza produsele existente (asa zisul meniu cu produse al restaurantului) si sa selecteze ce doreste clientul. Aceasta comanda trecandu-se intr-un tabel nou specific comenzilor. Pe langa acestea el trebuie sa calculeze pretul final al comenzii

in functie de produsele selectate (vom intra in amanunt mai tarziu) si in cele din urma sa genereze un bon fiscal care sa cuprinda detaliile privind comanda respectiva. Seful bucatar va fi notificat ori de cate ori s-a inregistrat o comanda care necesita combinarea a cel putin doua produse de baza. Acesta va avea disponibila o fereasta pentru a-l instiinta ce produse trebuie sa adauge pentru a gati si a combina ingredientele astfel incat clientul sa fie servit cu ceea ce doreste.

5. Rezultate

In urma generarii unui bon fiscal se formeaza un fisier text incluzand detalii privind comanda. Data, nr . mesei, ce a comandat si totalul de plata ca in exemplele urmatore:

----BON FISCAL----

Data: 20.10.2019

La masa: 2

S-a inregistrat urmatoarea consumatie:

[ton, cartofi natur, branza de vaca]

In valoare de: 99 lei.

Va mai asteptam!

----BON FISCAL----

Data: 20.10.2019

La masa: 2

S-a inregistrat urmatoarea consumatie:

[ton, cartofi natur, branza de vaca, cartofi piure, cartofi natur, branza de vaca, usturoi]

In valoare de: 264 lei.

Va mai asteptam!

6. Concluzii, ce s-a invatat din tema, dezvoltari ulterioare

In urma implementarii acestui proiect am invatat sa lucrez serealizand obiectele dintr-un fisier, sa gestionez si sa prelucrez informatiile din tabele pentru a le putea afisa si pentru a putea interactiona utilizatorul cu ele.

Am invatat mai bine sa folosesc JTable, operatiile de selectare a produsului putand fi astfel usurate. Jtable iti permite sa selectezi un rand si poate stoca informatiile din acesta, usurand astfel lucru cu datele existente.

Dezvoltari ulterioare a acestei pot fi mai multe tabele, mai multe produse in meniu si chiar organizate in functie de categoria din care face parte fiecare preparat, apropiindu-ne astfel cat mai mult de viata reala.

7 . Bibliografie

http://www.lec-academy.ro/utilizarea-jtable-in-swing-partea-i/

https://ms.sapientia.ro/~manyi/teaching/oop/oop_romanian/curs16/curs16.html

https://howtodoinjava.com/oops/object-oriented-principles/

http://labs.cs.upt.ro/doc/DP/DP08.html

https://www.tutorialspoint.com/design_pattern/observer_pattern.htm