



Tema 4

Sistem de gestiune a unui restaurant

Student: Spinean Sebastian

Grupa: 302210

Cuprins:

1. Obiectivul temei
2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie

1.Obiectivul temei

Obiectivul principal:

Obiectivul principal al acestei teme este de a implementa o aplicatie care sa asigure gestionarea unui restaurant. Aplicatia trebuie sa defineasca trei utilizatori: administrator, waiter si chef. Fiecare user are anumite operatii specifice pe care le poate realiza. De asemenea, aplicatia trebuie sa serializeze datele intr-un fisier.

Obiective secundare:

Obiective secundare:	Descriere	Capitol
Determinarea cazurilor de utilizare	Se identifica cazurile de utilizare si se descriu prin intermediul diagramelor use case	2
Alegerea structurilor de date	Se aleg structurile de date potrivite	3
Impartirea pe clase si pachete	Impartirea codului in pachete si clase	3
Dezvoltarea algoritmilor	Implementarea algoritmilor	4
Serializarea datelor intr-un fisier	Se realizeaza operatia de serializare a datelor din restaurant	4

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

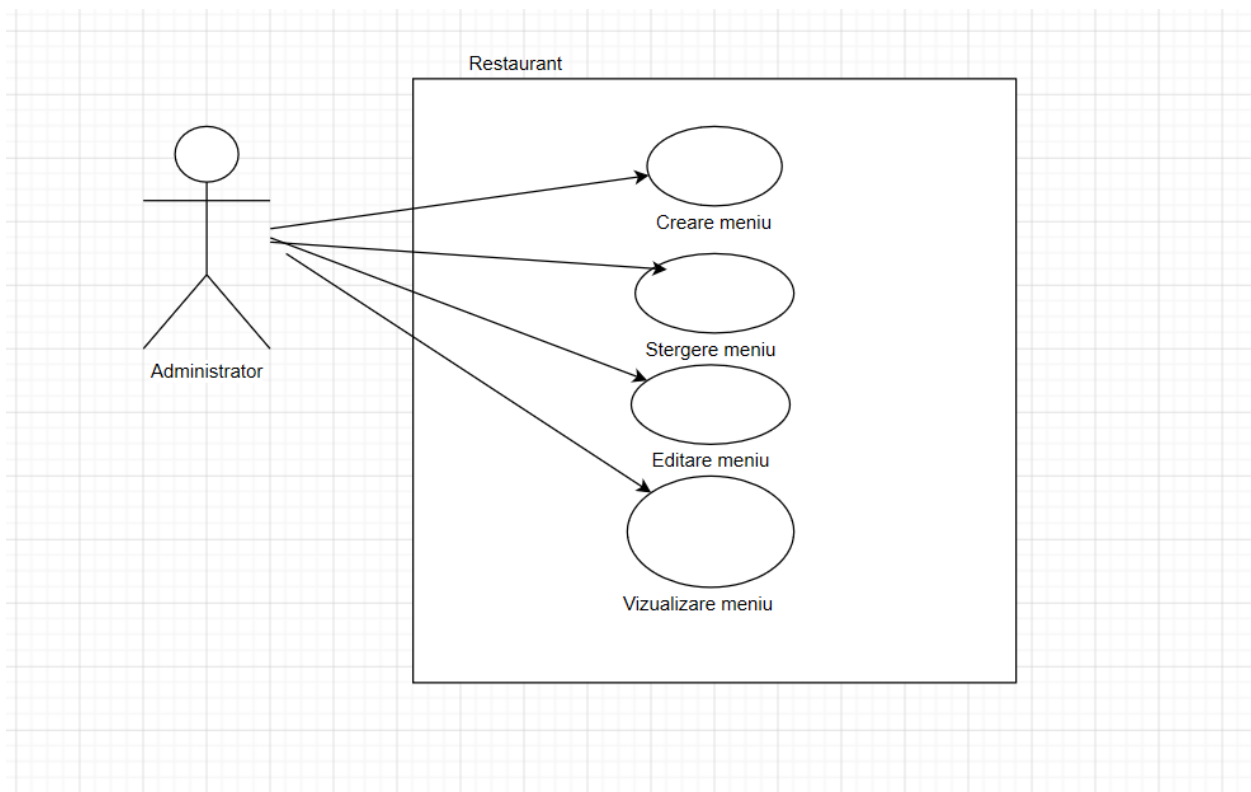
Cerintele functionale:

In cadrul proiectului se disting urmatoarele cerinte functionale:

- Crearea a trei interfete grafice corespunzatoare celor trei useri
- Implementarea unei clase care sa faca serializarea si deserializarea datelor
- Generarea facturilor pentru comenzi

Diagrame use case:

Pentru analiza cazurilor de functionare se folosesc diagramele use case. Astfel, avem doi actori: administratorul si waiter-ul. Cei doi actori au operatii specifice pe care le pot executa.



Cazul de utilizare: Creare meniu

Actorul principal: Administratorul

Scenariul cu success:

1. Administratorul deschide din fereastra sa principala optiunea de Create
2. Se introduce numele noului element din meniu
3. Se introduce pretul daca este cazul (la crearea unui base product)
4. Se apasa pe butonul de create
5. Se afiseaza un mesaj conform caruia operatia s-a executat cu success

Scenariu alternativ:

- a. Se doreste crearea unui produs deja existent
 1. Se afiseaza mesajul ca acest produs exista deja
 2. Nu se executa operatia de adaugare a unui nou element
 3. Administratorul revine la pasul 2
- b. Se doreste crearea unui produs compus dintr-un produs care nu exista
 1. Se afiseaza un mesaj ca nu exista un anumit produs
 2. Nu se executa operatia de adaugare a unui nou element
 3. Administratorul revine la pasul 2
- c. Nu se completeaza un anumit field
 1. Se afiseaza un mesaj de eroare care spune ca un field este gol
 2. Nu se executa operatia de adaugare a unui nou element
 3. Administratorul revine la pasul 2

Cazul de utilizare: Stergere meniu

Actorul principal: Administratorul

Scenariul cu success:

1. Administratorul deschide din fereastra sa principala optiunea de Delete
2. Se introduce numele elementului care urmeaza a fi sters
3. Se apasa butonul de delete
4. Se executa stergerea

Scenariu alternativ:

- a. Se introduce numele unui produs care nu exista
 1. Nu se executa nicio stergere
 2. Administratorul revine la pasul 2
- b. Nu se completeaza fieldul
 1. Se afiseaza un mesaj de eroare
 2. Nu se executa operatia

3. Administratorul revine la pasul 2

Cazul de utilizare: Editare meniu

Actorul principal: Administratorul

Scenariul cu success:

1. Administratorul deschide din fereastra sa principala optiunea de Edit
2. Se introduce numele produsului de baza care se doreste a fi editat
3. Se introduce noul pret
4. Se apasa butonul de edit
5. Se executa operatia

Scenariu alternativ:

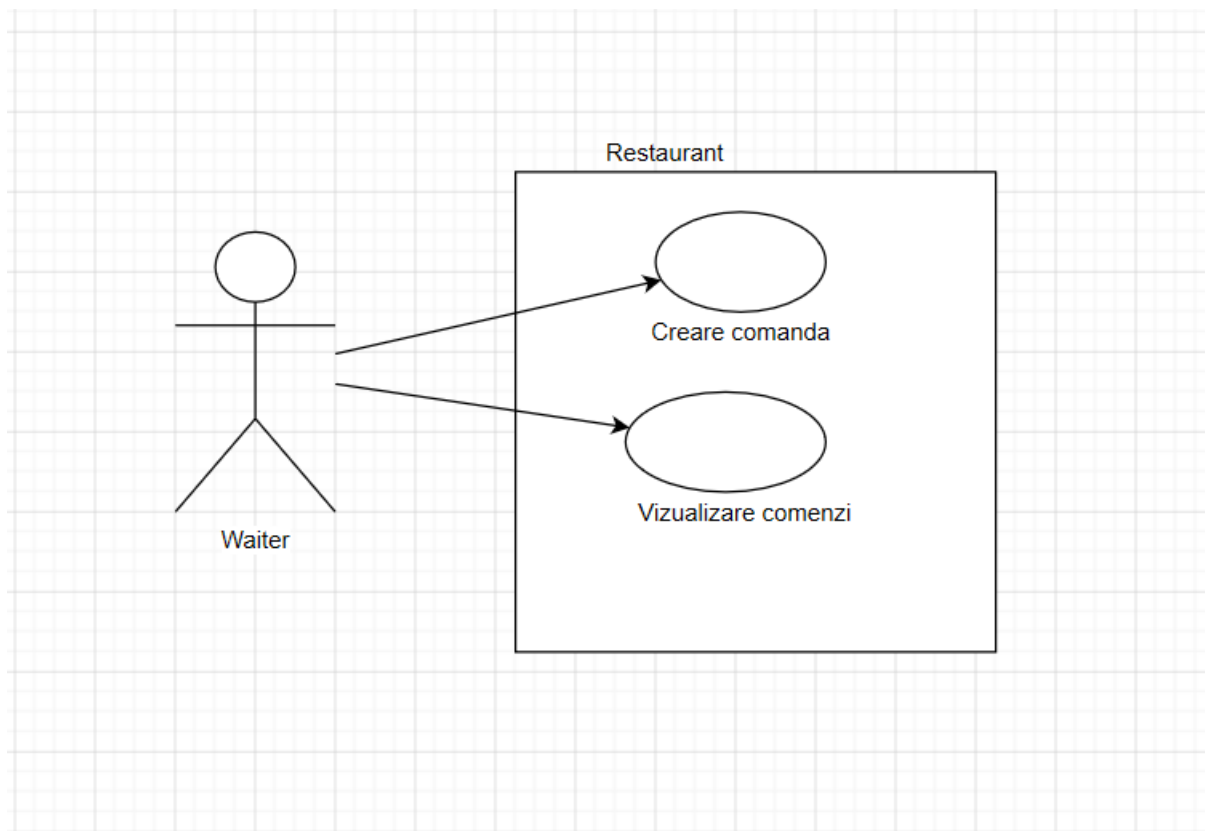
- a. Se introduce numele unui produs inexistent
 1. Nu se executa operatia
 2. Administratorul revine la pasul 2
- b. Nu se completeaza un anumit field
 1. Se afiseaza un mesaj de eroare
 2. Nu se executa operatia
 3. Administratorul revine la pasul 2.
- c. Pretul nu este un numar real
 1. Se afiseaza un mesaj de eroare
 2. Nu se executa operatia
 3. Administratorul revine la pasul 2.

Cazul de utilizare: Vizualizare meniu

Actorul principal: Administratorul

Scenariul cu success:

1. Administratorul deschide din fereastra sa principala optiunea de View
2. Se deschide o noua fereastra care contine un tabel in care sunt produsele si preturile lor



Cazul de utilizare: Creare comanda

Actorul principal: Waiter

Scenariul cu success:

1. Waiterul deschide din fereastra sa principala optiunea de Create order
2. Se deschide fereastra chef-ului
3. Se introduce numarul mesei de la care se face comanda
4. Se adauga produsele care se comanda
5. Se apasa pe butonul de finalizare comanda
6. Se realizeaza comanda
7. Se genereaza o facture pentru comanda in format txt
8. Cheful primeste notificarea si afiseaza ca prelucreaza comanda

Scenariu alternativ:

- a. Numarul mesei nu este un numar intreg
 1. Se afiseaza un mesaj de eroare
 2. Nu se executa comanda
 3. Waiter-ul revine la pasul 3
- b. Nu se comanda niciun produs

1. Nu se executa comanda
2. Waiter-ul revine la pasul 4

Cazul de utilizare: Vizualizare

Actorul principal: Waiter

Scenariul cu success:

1. Waiterul deschide din fereastra sa principala optiunea de View
2. Se deschide o noua fereastra care contine un tabel in care sunt inregistrate comenzile

3 Proiectare

Codul este impartit in patru pachete:

- PresentationLayer
- BusinessLayer
- DataLayer

Astfel, codul aplicatiei este structurat pe modelul precizat in cerinta temei. Pachetul PresentationLayer contine clasele care se ocupa de interfata grafica a programului. In pachetul BusinessLayer este continuta logica aplicatiei. Pachetul DataLayer contine clasele care se ocupa de serializarea datelor si de generarea facturilor in format txt.

Diagrama de pachete:

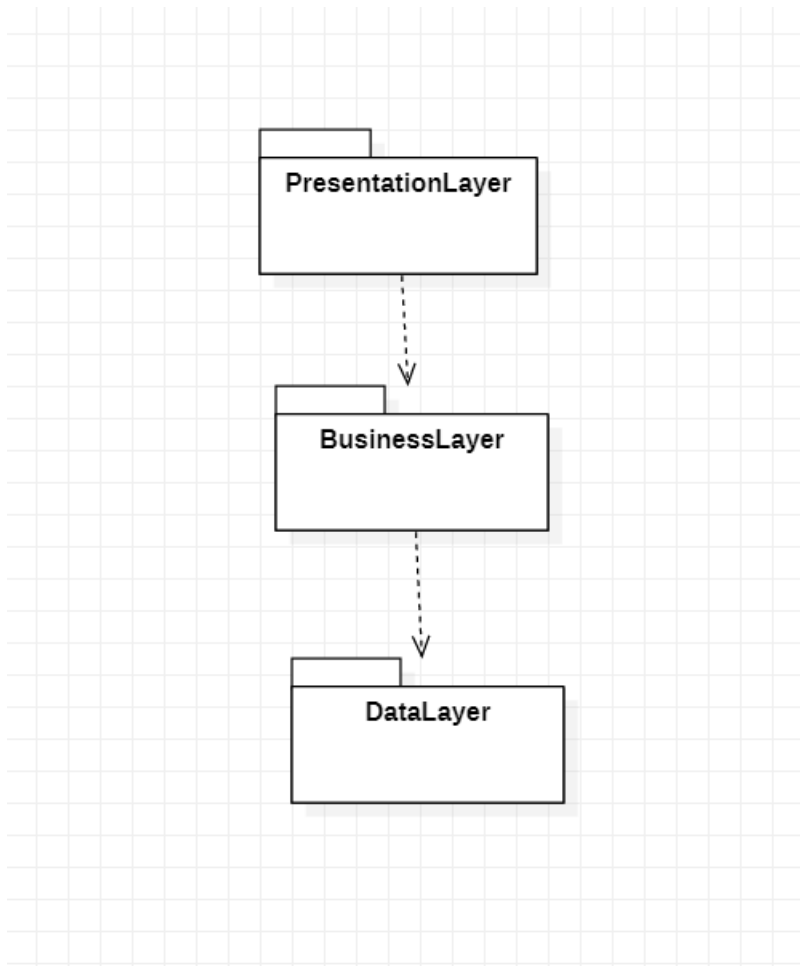
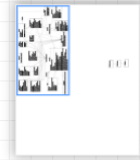


Diagrama de clase:



Dupa cum se poate vedea din diagrama de clase UML, codul aplicatiei este impartit in 19 clase.

Pachetul PresentationLayer:

Acest pachet contine clasele: MainClass, AdministratorGui, AdministratorCreateMenuItem, , AdministratorDeleteMenuItem, , AdministratorEditMenuItem, , AdministratorViewMenu, WaiterGui, WaiterCreateOrder, WaiterViewOrder si ChefGui. Aceste clase se ocupa de interfata grafica a aplicatiei de unde sunt introduse datele de catre utilizatori.

Pachetul BusinessLayer:

Acest pachet este format din clasele: MenuItem, BaseProduct, CompositeProduct, Order, Restaurant si Validator. De asemenea, pachetul mai contine interfata IRestaurantProcessing in care se gasesc metodele corespunzatoare operatiilor pentru administrator si waiter.

Pachetul DataLayer:

Acest pachet contine doar doua clase: Serializator si FileWriterBill. Aceste clase realizeaza serializarea si deserializarea precum si generarea facturilor.

Structurile de date:

Pentru produsele din meniul restaurantului s-au folosit trei clase: clasa abstracta MenuItem si doua clase care o mostenesc pe aceasta: BaseProduct si CompositeProduct. Pentru construirea acestor clase s-a folosit Composite Design Pattern. Pretul unui meniu compus se calculeaza ca fiind 90% din suma preturilor elementelor care il compun.

Pentru a memora meniul in restaurant s-a folosit o structura de Set<MenuItem>. Pentru a memora comenzile s-a folosit o structura de forma Map<Order,ArrayList<MenuItem>.

4 Implementare

Pachetul PresentationLayer:

Clasa MainClass:

Aceasta clasa contine metoda main in care se seteaza numele fisierului din care se face deserializarea respective serealizarea. De asemenea, se instantiaza doua obiecte de tipul AdministratorGui si WaiterGui.

Clasa AdministratorGui:

Aceasta clasa reprezinta fereastra principala a administratorului.

Variabile instantia:

Are ca variabila instantia panelul principal, un label pentru titlu, patru butoane pentru operatiile administratorului si un obiect al clasei RestaurantSerializator.

Constructor:

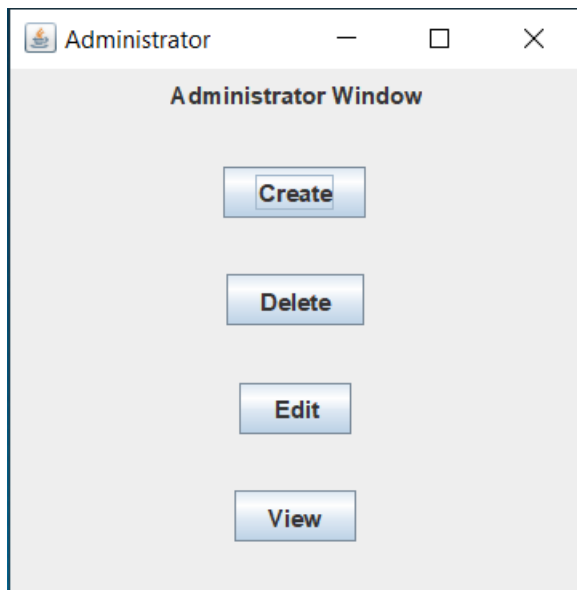
Clasa are un constructor in care se construiesc interfata grafica.

Metode:

Metodele adauga ascultatori pentru butoane.

Clasa are patru metode prin care se deschid ferestrele specific fiecarei operatii.

In metoda view se face o deserealizare a restaurantului dupa care se extrage meniul. Se formeaza continutul celulelor care urmeaza a fi afisate in tabel si se deschide tabelul.



Clasa AdministratorCreateMenuItem:

Clasa corespunde ferestrei pentru crearea unui nou element de meniu.

Variabile instantia:

Are ca variabila instantia panelul principal, labeluri pentru titluri, butoane pentru a crea produse, zone de text pentru a scrie numele produsului si un obiect al clasei RestaurantSerializer.

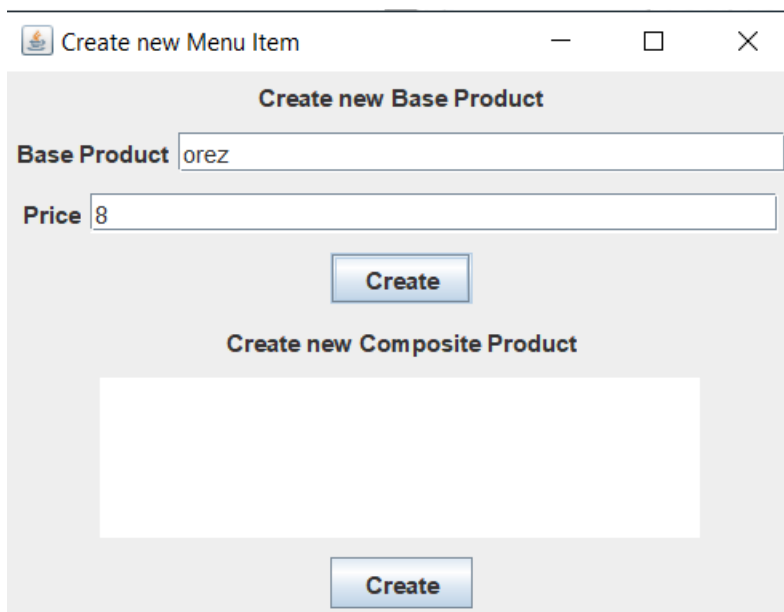
Constructor:

In constructor se realizeaza implementarea interfetei grafice.

Metode:

Metodele adauga ascultatori pentru butoane.

Metoda createBase realizeaza crearea unui nou base product. Aceasta metoda incepe prin deserealizarea restaurantului. Citeste numele produsului si pretul acestuia. Daca campurile citite sunt valide si nu sunt goale se apeleaza metoda de creare de base product nou.



Metoda createComposite functioneaza asemanator cu cea de createBase.

Pentru crearea unui composite product format doar din base product-uri se scrie pe o singura linie acele produse care vor forma noul produs:

Create new Menu Item

Create new Base Product

Base Product

Price

Create

Create new Composite Product

orez gratar

Create

Pentru un composite product care este format si dintr-un alt composite product se scriu pe mai multe linii produsele care formeaza noul produs:

Create new Menu Item

Create new Base Product

Base Product

Price

Create

Create new Composite Product

orez gratar
salata

Create

Clasa AdministratorDeleteMenuItem:

Aceasta clasa reprezinta fereastra pentru stergere a unui produs.

Variabile instantia:

Are o variabila pentru panelul principal, un label pentru titlu, un buton pentru efectuarea operatiei si o zona de text pentru introducerea produsului si un restaurant.

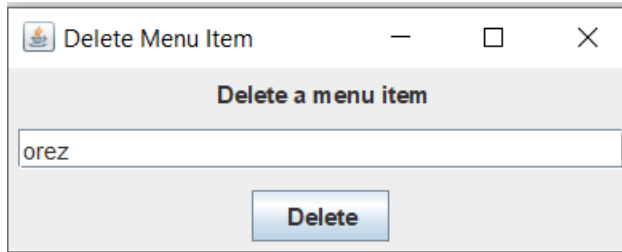
Constructor:

Aici se construiește interfata.

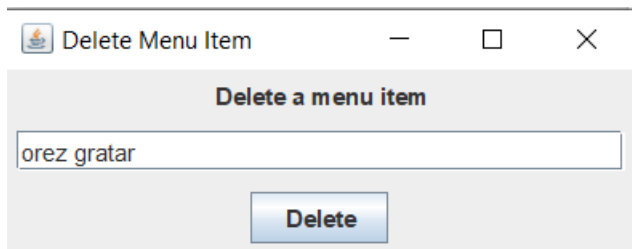
Metode:

Are doar o metoda delete care adauga un ascultator pe butonul de delete. Se deserializeaza restaurantul, se citeste ce a fost scris in zona de text si se apeleaza stergerea elementului.

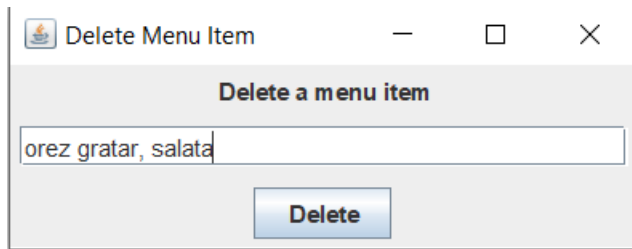
Stergerea unui base product



Stergerea unui composite product format doar din base product-uri



Stergerea unui composite product



Clasa AdministratorEditMenuItem:

Aceasta clasa reprezinta fereastra pentru editare a unui produs.

Variabile instantă:

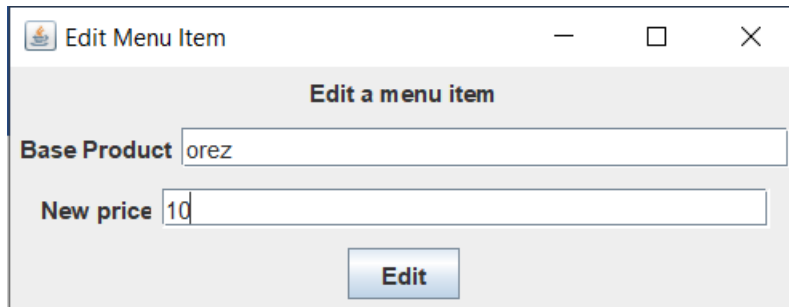
Exista variabile instantă pentru paneluri, labeluri, un buton, un obiect al clasei RestaurantSerializator si un restaurant.

Constructor:

Aici se construiește interfata.

Metode:

Este metoda edit care adaugă un ascultător pentru butonul de edit. Se deserealizează restaurantul, se citesc valorile corespunzătoare produsului și noului preț și dacă sunt valide se apelează metoda de editare.



Clasa AdministratorViewMenu:

Clasa reprezintă fereastra în care se afișează meniul. Pentru reactualizarea tabelului acesta trebuie să fie închis și redeschis.

Variabile instanță:

O variabilă de tip `JTable` și una de tip `JScrollPane`.

Constructor:

Constructorul primește ca parametru o matrice de tipul `Object` care reprezintă liniile tabelului și un vector de tip `String` care reprezintă numele coloanelor.

Clasa WaiterGui:

Reprezintă fereastra principală a waiterului.

Variabile instanță:

Panelul principal, un label pentru titlu și două butoane pentru operațiile waiterului.

Constructor:

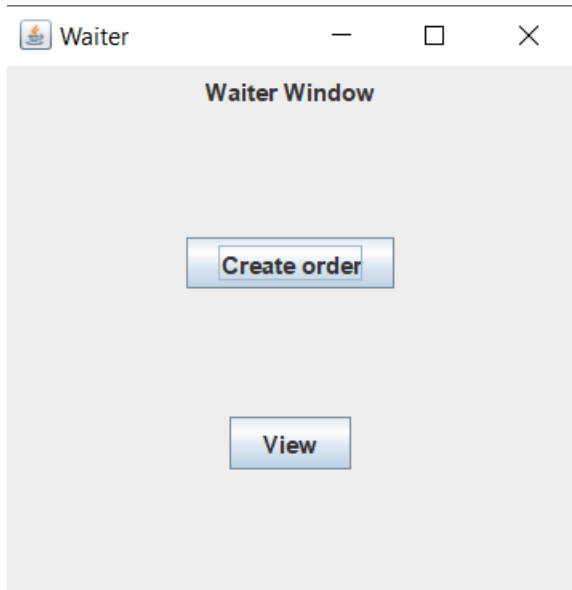
Aici se construiește interfata.

Metode:

Adaugă ascultători pentru cele două butoane.

Metoda create deschide fereastra pentru operația de create order.

Metoda view construiește conținutul tabelului care afișează comenzile. Se deserealizează restaurantul, se returnează hashmap-ul care stochează comenzile și se construiește conținutul tabelului.



Clasa WaiterCreateOrder:

Reprezintă fereastra pentru operația de creare de comandă. La deschiderea acestei ferestre se deschide și o fereastră pentru chef

Variabile instanță:

Are panouri, etichete, zone de text, butoane, un combobox pentru meniu, are două obiecte de tip ChefGui, o listă statică de MenuItem-uri și un obiect al clasei RestaurantSerializator.

Constructor:

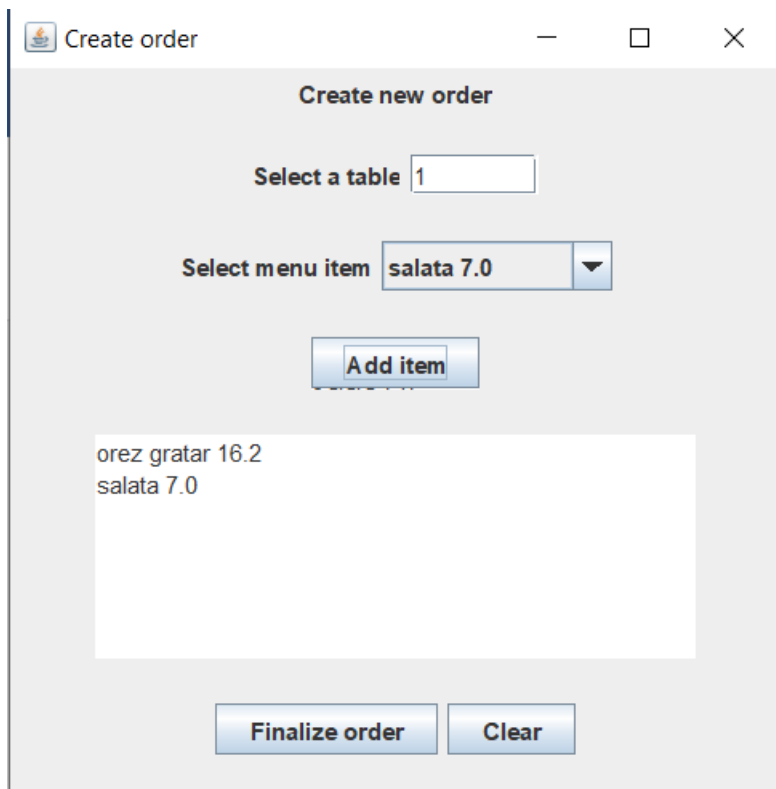
Aici se construiește interfața.

Metode:

Metoda addItem adaugă un nou element în comandă. Se deserealizează restaurantul după care se adaugă în lista statică elementul selectat.

Metoda finalizeOrder finalizează comanda. Se deserealizează restaurantul, se închide vechea fereastră pentru chef și se deschide una nouă în care se afișează mesajul că cheful pregătește comanda. Se citește numărul mesei pentru care se face comanda. Comanda se creează pe baza elementelor din lista statică după care aceasta este golită.

Metoda clear șterge elementele adăugate în lista statică.



Clasa WaiterViewOrder:

Clasa reprezinta fereastra in care se afiseaza comenzile. Pentru reactualizarea tabelului acesta trebuie sa fie inchis si redeschis.

Variabile instantia:

O variabila de tip `JTable` si una de tip `JScrollPane`.

Constructor:

Constructorul primeste ca parametru o matrice de tipul `Object` care reprezinta linile tabelului si un vector de tip `String` care reprezinta numele coloanelor.

Clasa ChefGui:

Aceasta metoda implementeaza interfata `Observer`

Variabile instantia:

Panelul principal, un label, o zona de text si un restaurant.

Constructor:

Aici se construiesc interfata.

Metode:

Suprascrie metoda update si afiseaza un anumit mesaj.

Pachetul BusinessLayer:

Clasa MenuItem:

Aceasta este o clasa abstracta care contine metoda abstracta computePrice ce returneaza un float. Implementeaza interfata Serializable.

Clasa BaseProduct:

Aceasta clasa mosteneste clasa MenuItem.

Variabile instantata:

Name de tip String defineste numele produsului.

Price de tip float reprezinta pretul produsului.

Constructor:

Clasa are doi constructori: unul fara parametri si unul cu parametri care initializeaza toate variabilele instantata.

Metode:

Metoda computePrice returneaza pretul produsului.

Clasa mai are metode de getters si setters si o metoda de toString

Clasa CompositeProduct:

Aceasta clasa mosteneste clasa MenuItem.

Variabile instantata:

Name de tip String defineste numele produsului.

Price de tip float reprezinta pretul produsului.

Menu o multime de MenuItem – contine elementele din care este format acest composite product

Constructor:

Un constructor fara parametri care initializeaza multimea.

Metode:

Metoda computePrice returneaza suma preturilor elementelor din care este format acest composite product.

Metoda addItem adauga un nou MenuItem in multimea sa.

Mai exista metode de getters si setters si o metoda de toString.

Clasa Order:

Aceasta clasa reprezinta datele unei comenzi. O sa constituie cheia la care v-or fi mapate produsele comandate in tabela de dispersie. Implementeaza interfata Serializable.

Variabile instantia:

OrderId de tipul int – id-ul comenzii

Date de tipul Date – data la care s-a facut comanda

Table de tipul int – numarul mesei de la care s-a facut comanda.

Constructori:

Clasa are doi constructori: unul fara parametri si unul cu parametri care initializeaza toate variabilele instantia.

Metode:

Clasa suprascrie metoda equals si metoda hashCode.

Mai exista metode de getters si setters.

Clasa Validator:

Aceasta clasa este utilizata pentru a valida anumite date de intrare.

Variabile instantia:

Aceasta clasa are o singura variabila instantia de tip String numita variable.

Constructori:

Clasa Validator are doi constructori: unul cu un parametru de tip String si una fara niciun parametru.

Metode:

Clasa dispune de doua metode: isValid si isValidFloat care nu primesc niciun parametru. Ambele metode returneaza true daca variabila este valida si false in caz contrar. Metoda isValid verifica daca variabila care este de tip String este are forma unui numar intreg pozitiv in timp ce metoda isValidFloat verifica daca variabila instantia are forma unui numar real pozitiv.

Interfata IRestaurantProcessing:

Aceasta interfata defineste metodele specifice operatiilor administratorului si waiterului: createNewBaseProduct, createNewCompositeProduct, deleteMenuItem, editMenuItem, createOrder, computePriceForOrder, generateBill.

Clasa Restaurant:

Aceasta clasa implementeaza interfata IRestaurantProcessing si Serializable. De asemenea, mosteneste clasa Observable.

Variabile instantia:

idOrder de tip int – id-ul comenzii.

menuItems de tip menuItems – meniul restaurantului

orderItems de tip Map<Order,ArrayList<MenuItem>> - comenzile.

Constructor:

Un constructor fara parametrii.

Metode:

Metoda containsNull este un invariant al clasei. Aceasta returneaza true daca toate elementele din menuItems sunt diferite de null si false in caz contrar.

Metoda eliminareSpatii primeste ca argument un String si returneaza tot un String. Elimina spatiile multiple dintr-un String precum si spatiile de la inceput sau final daca exista.

Metoda findBaseProduct returneaza un base product a carui nume este egal cu Stringul primit ca parametru.

Metoda findCompositeProduct returneaza un composite product a carui nume este format din Stringurile din vectorul de Stringuri primit ca argument.

Metoda findSameComposite returneaza true daca mai exista in meniu un alt composite care este format din aceleasi elemente cu composite-ul primit ca argument si false in caz contrar.

Metoda createNewBaseProduct primeste ca argument un String pentru nume si un String pentru pret. Verifica folosind assert ca cele doua argumente sa fie valide si returneaza 0 daca exista deja un base product cu acest nume sau 1 in caz ca nu exista si creaza acest nou base product. La final verifica invariantul.

Metoda createNewCompositeProduct primeste ca parametru un String. Verifica ca parametrul sa fie diferit de null. Initializeaza un composite nou. Desparte stringul line dupa caracterul de “\n”. Daca avem o singura linie returneaza valoarea functiei createSimpleCompositeProduct. Daca avem mai multe linii parcurgem linie cu linie si pe fiecare o despartim in cuvinte dupa “ ”. Daca avem doar un cuvint avem de a face cu un base product. In acest caz cautam sa vedem daca exista un base product cu acest nume iar daca exist ail adaugam in lista lui composite. Daca avem mai multe cuvinte pe o linie cautam sa vedem daca exista un composite product cu acel nume, caz in care il adaugam in lista lui composite. La final setam numele noului composite si pretul. Daca nu mai exista un composite cu aceste elemente cream unul nou. Metoda returneaza 0 daca un element din composzita produsului pe care vrem sa-l cream nu exista, 1 in caz ca operatia s-a executat cu succes si 2 daca mai exista deja acest produs. La final se serializeaza restaurantul si se verifica invariantul.

Metoda createSimpleComposite creaza un composite format doar din base product.

Metoda deleteMenuItem sterge un element. Aceasta primeste ca argument un String. La inceput verifica sa fie valid argumentul. Desparte parametrul dupa virgula. Daca avem doar un cuvant despartim dupa spatii. Daca din nou avem doar un cuvant inseamna ca dorim sa stergem un base product. Se cauta base product-ul cu acel nume si se sterge din lista. De asemenea, se sterg toate composite producturile care contin acest base. Daca avem mai multe cuvinte cautam un composite product cu acel nume si il stergem din lista. De asemenea, se sterg toate composite producturile care contin acest element. Daca avem cel putin o virgula apelam metoda deleteCompositeProduct. La final se serializeaza restaurantul si se verifica invariantul.

Metoda deleteCompositeProduct primeste ca argument un vector de Stringuri. Face stergerea unui composite product care este format si dintr-un alt composite product.

Metoda editMenuItem primeste ca argument un String care reprezinta numele produsului si un pret de tip float. Se verifica daca exista acest base product si daca exista atunci se modifica pretul acestuia. De asemenea, se modifica pretul tuturor elementelor care contin acel base. La final se serializeaza restaurantul si se verifica invariantul.

Metoda createOrder primeste ca parametru un ArrayList<MenuItem> si un int table. Se incrementeaza idOrder. Se creaza un order avand idul idOrder, data curenta si numarul mesei table. Se adauga noua comanda, se serealizeaza restaurantul, se genereaza o factura si se trimite o notificare la chef.

Metoda computePriceForOrder primeste ca argument un ArrayList<MenuItem> si calculeaza suma preturilor elementelor din lista primita.

Metoda genereateBill genereaza o factura folosind un obiect al clasei FileWriterBill.

Pachetul DataLayer:

Clasa RestaurantSerializator:

Variabile instantia:

O variabila statica de tip String ce reprezinta numele fisierului in care se face serializarea.

Constructor:

Doi constructori: unul cu parametru si unul fara.

Metode

Metoda statica serializare primeste ca argument un Restaurant pe care il serializeaza.

Metoda statica deserializare care returneaza un restaurant pe care l-a deserializat.

Clasa FileWriterBill:

Constructor:

Are un singur constructor care creaza un fisier txt in care scrie date despre comanda.

5 Rezultate

S-a testat programul utilizand fisierul pentru serializare restaurant.ser.

6 Concluzii

In concluzie aceasta aplicatie implementeaza un system de gestiune a unui restaurant.

Posibilitati de dezvoltare ulterioara: introducere de mai multi utilizatori.

7 Bibliografie

<https://www.baeldung.com/java-serialization>

