

Documentatie – Tema 4

Restaurant Management System

Nume: Dascal Raluca Georgiana

Grupa: 30226

Profesor coordonator: Dorin Moldovan

Cuprins:

1. Cerinte functionale	3
2. Obiective.....	3
2.1 Obiectiv Principal.....	3
2.2 Obiective Secundare.....	3
3. Descrierea Interfetei Grafice	4
4. Analiza Problemei	5
5. Proiectare.....	6
5.1 Diagrame de clase	6
5.2 Descrierea Algoritmilor Implementati.....	6
6. Implementare	7
7. Rezultate.....	8
8. Concluzii si Posibilitati de Dezvoltare Ulterioare	12
9. Bibliografie	12

1. Cerinte Functionale

Sa se dezvolte un sistem care sa permita interactionarea clientului si a detinatorului unui restaurant folosind o interfata grafica prietenoasa.

2. Obiective

2.1 Obiectiv Principal

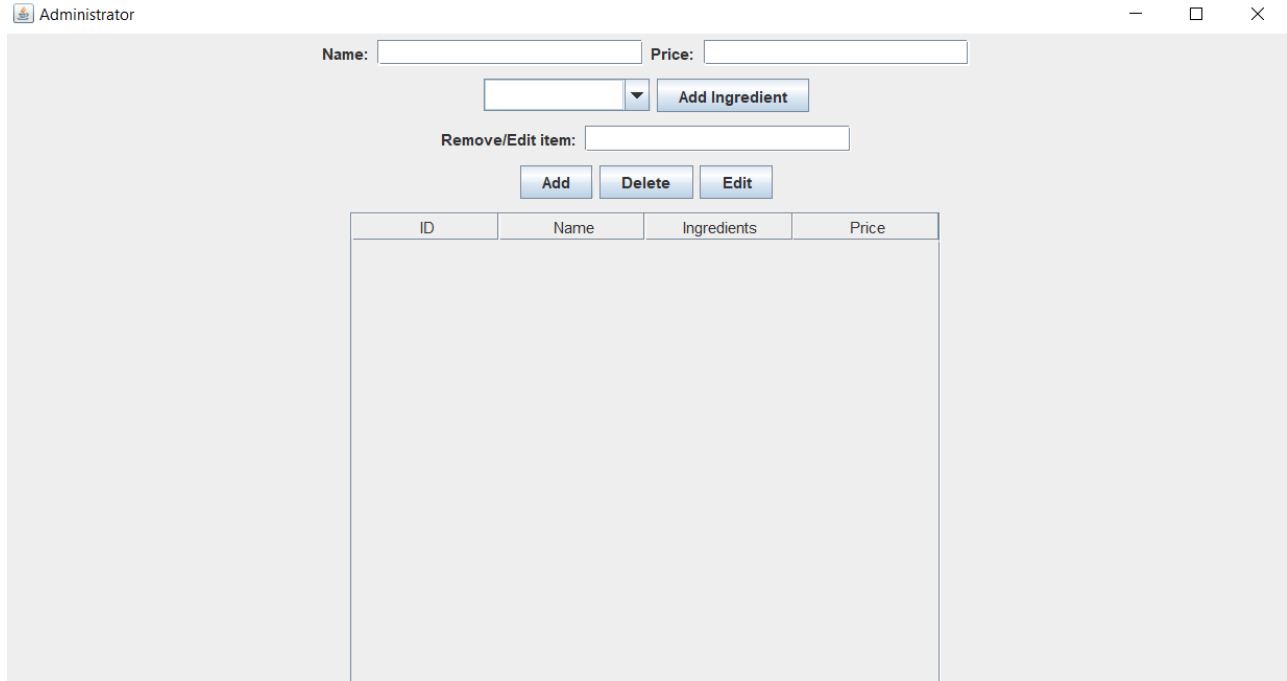
Dezvoltarea a doua interfete grafice care sa permita urmatoarele operatii: prima interfata, cea a detinatorului restaurantului sa ii permita acestuia sa modifice meniul, adica sa adauge, sa steraga, sau sa modifice pretul sau numele unui produs, iar cea de a doua sa permita clientului sa adauge o comanda si sa fie capabil sa vada pretul total al comenzii.

Se cere si respectarea unor restrictii la nivel de cod (crearea unor clase cu maxim 300 de linii, metode cu un numar maxim de 30 de linii de cod, respectarea functionalitatilor pentru cele 2 interfete: pentru administrator crearea,stergerea si modificare unui element din meniu, precum si vizualizarea acestora intr-un tabel, pentru Waiter adaugarea unei comenzi, vizualizarea tuturor comenzilor intr-un tabel, precum si generarea unei note de plata pentru fiecare comanda, si generarea unui fisier jar.

2.2 Obiective Secundare

Obiective secundare	Capitol
Descrierea interfetei grafice	3
Descrierea algoritmilor utilizati	5.2
Prezentarea claselor	6
Rezultatele obtinute	7

3. Descrierea Interfetei Grafice

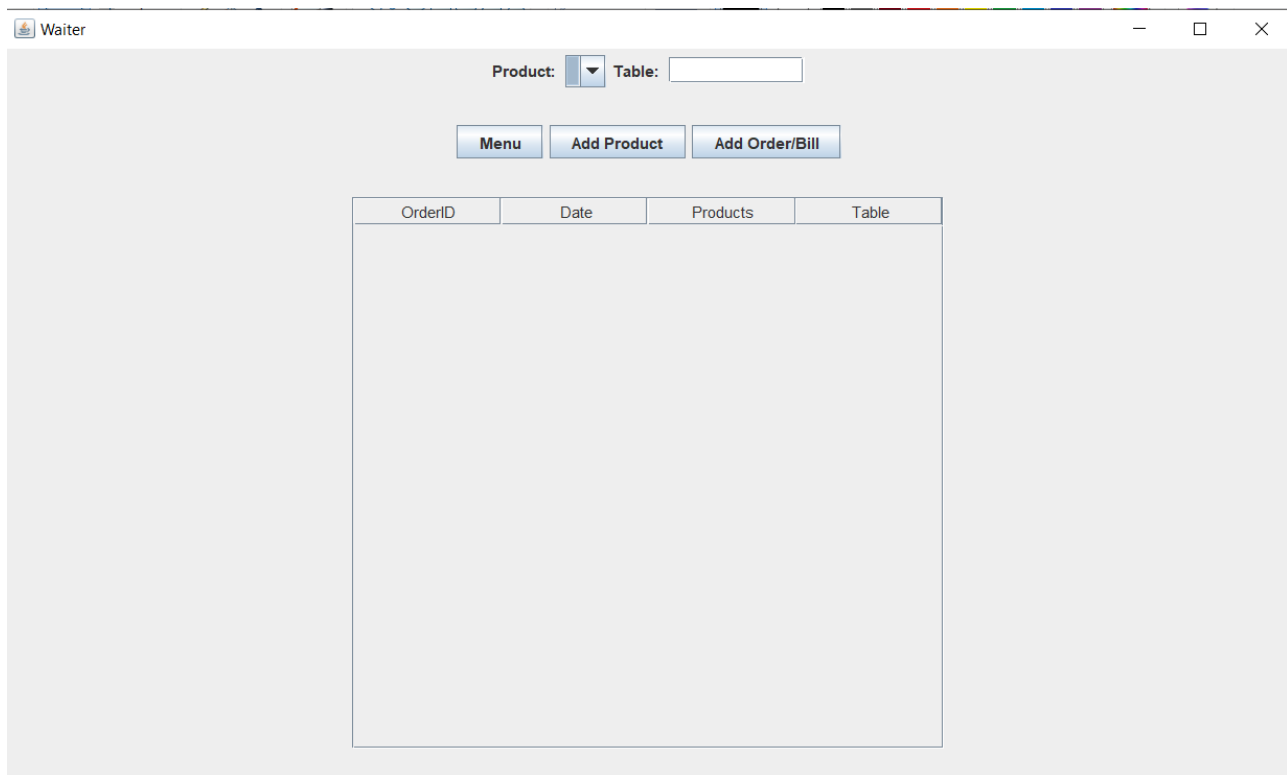


The Administrator interface window has a title bar with a small icon and the text "Administrator". The window contains the following elements:

- Name:** A text input field.
- Price:** A text input field.
- Ingredients:** A dropdown menu.
- Add Ingredient:** A button.
- Remove/Edit item:** A text input field.
- Add, Delete, Edit:** Three buttons.
- Table:** A table with the following structure:

ID	Name	Ingredients	Price
----	------	-------------	-------

Fig 3.1



The Waiter interface window has a title bar with a small icon and the text "Waiter". The window contains the following elements:

- Product:** A dropdown menu.
- Table:** A text input field.
- Menu, Add Product, Add Order/Bill:** Three buttons.
- Table:** A table with the following structure:

OrderID	Date	Products	Table
---------	------	----------	-------

Fig 3.2

In figura 3.1 este prezentata interfata pentru administrator. In campul nume si in campul price vor fi introduse detaliile despre produs. La apasarea butonului Add se va introduce produsul atat in campul ComoboBox cat si in tabelul din fereastra prezentata. Butonul Add ingredient se va folosi pentru a genera un produs compus selectand produse din ComboBox. Atunci cand s-au ales ingredientele, se va introduce numele produsului (campul pentru pret ramanand necompletat) se va apasa butonul Add care va efectua aceeaasi procedura precum in cazul precedent inclusive se va calcula pretul total ca fiind suma preturilor produselor. In campul Remove/ Edit name se va introduce numele unui produs care se doreste a fi edit sau sters. In cazul in care ne dorim stergerea se introduce doar numele explicat mai sus si se apasa butonul Delete. In cazul editarii se va introduce numele produsului care se doreste a fi modificat iar in campurile nume si pret se vor introduce noile date- Obligatoriu se completeaza amabele campuri si se va apasa butonul Edit care va modifica atat produsul de baza cat si toate celalalte produce care il contin.

In figura 3.2 este prezentata interfata pentru Waiter. In campul product se vor adauga dupa apasarea butonului Menu toate produsele disponibile in meniu. Butonul Add Product se va folosi petnru a adauga produse in comanda selectand campul present al ComboBox-ului Product. Dupa selectarea tuturor produselor dorite, se va introduce si numarul mesei dupa care se va apasa butonul Order/Bill. Acest buton va realiza atat introducerea informatiilor despre comanda in tabelul dn interfata cat si generarea unui fisier text care va contine numarul comenzii, masa, produsele solicitate si pretul total.

4. Analiza problemei

In figura de mai jos (4.1) este descrisa intr-o forma grafica relatia dintre utilizator,,interfata si metodele implementate cu ajutorul unor simboluri sugestive pentru a fi mai usor de inteles pentru un utilizator oarecare .

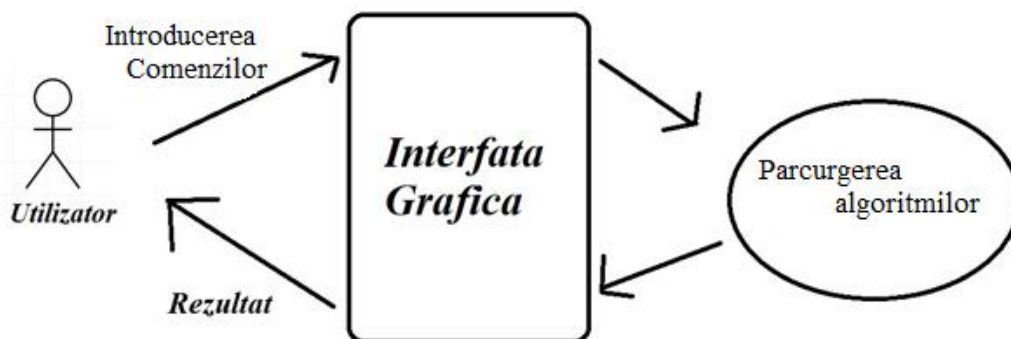


Fig 4.1

5. Proiectare

5.1 Diagrama de clase

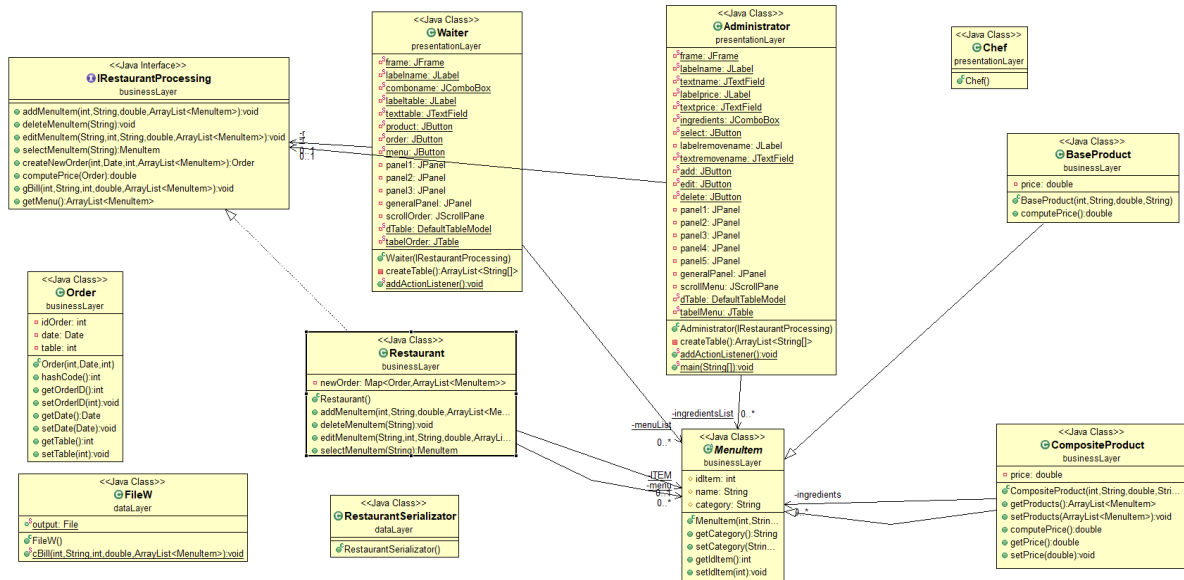


Fig 5.1.1

În figura 5.1.1 sunt prezentate diagramele UML a claselor proiectului realizat. Relația dintre acestea a fost sintetizată de către programul în care a fost scris și codul, Eclipse Java 2019. În această imagine se evidențiază atât fiecare clasă în parte (cu variabilele declarate și metodele ce definesc fie constructorii, gettere, settere fie algoritmi utili în realizarea funcționalităților aparatului de calcul) cât și legăturile dintre acestea. Aceste diagrame UML ne ajută la formarea unei imagini în ansamblu a proiectului ajutând totodată și la înțelegerea unor concepte și paradigme a programării orientate pe obiect.

5.2 Descrierea algoritmilor implementați

Fiecare metodă definită și descrisă în acest proiect, cu excepția constructorilor și a celor care setează și returnează valorile variabilelor declarate în clase, descriu câte o sarcină pe care proiectul trebuie să o îndeplinească folosind un algoritm mai mult sau mai puțin riguros. Fiecare clasă, variabilă și metodă vor fi prezentate în următorul capitol.

6. Implementare

În acest capitol se va prezenta fiecare clasă a proiectului și se va explica fiecare variabilă declarată și metoda implementată în acestea având la bază diagrama UML sintetizată de către programul în care a fost scris codul. S-au creat 3 pachete, fiecare îndeplinindu-și “misiunea”.

6.1. Clasa MenuItem

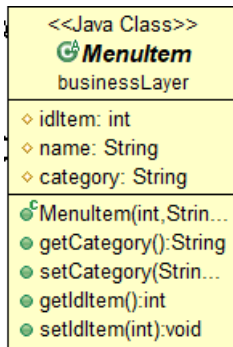


Fig 6.1.

În figura alăturată (6.1) este prezentată clasa MenuItem care reprezintă clasa de temelie a proiectului. Aceasta reprezintă definirea caracteristicilor produselor din meniu. Variabila idItem reprezintă identificatorul unic al fiecărui produs. Name reprezintă denumirea fiecărui element din meniu. Category reprezintă tipul produsului, baseProduct pentru produsele de bază, compositeCategory pentru elementele compuse din produse de tip baseProduct. Clasa conține un constructor, gettere și settere precum și o metodă abstractă pentru calcularea pretului.

6.2. Clasa BaseProduct

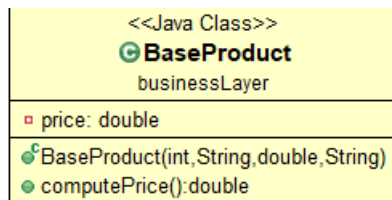
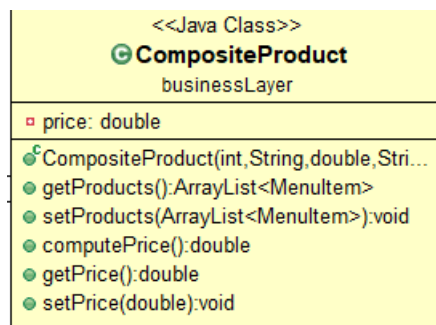


Fig 6.2

În figura alăturată este definită clasa BaseProduct. Aceasta clasa moștenește clasa prezentată mai sus și mai adaugă și variabila price care reprezintă prețul. Clasa conține un constructor, și prima definiție a metodei abstracte pentru calcularea prețului.

6.3. Clasa CompositeProduct



În figura 6.3 este prezentată clasa CompositeProduct care moștenește clasa MenuItem. Pe lângă atributele moștenite de la clasa “parinte” aceasta mai conține atât variabila price, cât și un ArrayList de MenuItem care reprezintă produse de tip baseProduct care se află în componența acestui produs. Aceasta clasa conține un constructor, gettere și settere precum și o nouă definiție a metodei abstracte computePrice.

6.4 Clasa Order

<<Java Class>>	
Order	
businessLayer	
idOrder: int	
date: Date	
table: int	
Order(int,Date,int)	
hashCode():int	
getOrderID():int	
setOrderID(int):void	
getDate():Date	
setDate(Date):void	
getTable():int	
setTable(int):void	

Clasa Order definește toate caracteristicile unei comenzi. Variabila idOrder reprezintă identificatorul unic al fiecărei comenzi, date reprezintă data și ora la care o comandă a fost plasată, iar variabila table se referă la numărul mesei de la care s-a solicitat aceasta. Această clasă conține un constructor, gettere și settere, precum și metoda hashCode care permite generarea unui număr pentru a putea fi realizată stocarea tuturor comenzilor într-un HashMap.

Fig 6.4

6.5 Clasa Restaurant

<<Java Class>>	
Restaurant	
businessLayer	
newOrder: Map<Order,ArrayList<MenuItem>>	
Restaurant()	
addMenuItem(int,String,double,ArrayList<Me...	
deleteMenuItem(String):void	
editMenuItem(String,int,String,double,ArrayLi...	
selectMenuItem(String):MenuItem	

Clasa Restaurant reprezintă clasa care cuprinde toate definițiile și descrierile funcționalităților aplicației. Această clasă are ca variabile un HashMap care reține și permite stocarea tuturor comenzilor, precum și o listă de MenuItem care reprezintă întregul meniu. Clasa conține un constructor, metoda addMenuItem care reprezintă adăugarea unui element în meniul restaurantului, metoda deleteMenuItem permite ștergerea unui element a cărui nume va fi primit ca parametru. Metoda selectItemMenu returnează obiectul MenuItem a cărui nume e primit ca parametru. Metoda

Fig 6.5

createNewOrder realizează crearea unei noi comenzi care apelează metode definite în clasa Order. Metoda gBill apelează o metodă dintr-o clasă ce va fi descrisă mai jos și are ca scop crearea unui fișier text care va conține detalii despre o comandă precum- identificatorul unic al acesteia, data la care a fost plasată, masa de la care a fost solicitată, o listă cu produsele alese și prețul total al comenzii. Clasa mai conține și un getter și un setter pentru lista de MenuItem care reprezintă meniul restaurantului.

6.6 Clasa IRestaurantProcessing

<<Java Interface>>	
IRestaurantProcessing	
businessLayer	
addMenuItem(int,String,double,ArrayList<MenuItem>):void	
deleteMenuItem(String):void	
editMenuItem(String,int,String,double,ArrayList<MenuItem>):void	
selectMenuItem(String):MenuItem	
createNewOrder(int,Date,int,ArrayList<MenuItem>):Order	
computePrice(Order):double	
gBill(int,String,int,double,ArrayList<MenuItem>):void	
getMenu():ArrayList<MenuItem>	

În figura alăturată este prezentată o interfață care permite folosirea tuturor metodelor prezentate în clasa Restaurant.

Fig 6.6

6.7 Clasa Administrator



Fig 6.7

În figura 6.7 este prezentată definirea interfeței Administrator prezentă și în capitolele anterioare. Clasa conține variabilele ce ajută la crearea interfeței precum ComboBox-uri, câmpuri de text, butoane și etichete, un tabel, precum și un constructor, o metodă care definește crearea tabelului în fereastra de interfață, o metodă care conține ascultători pentru fiecare buton. Tot aici se găsește și metoda `main` a aplicației.

6.8 Clasa Waiter

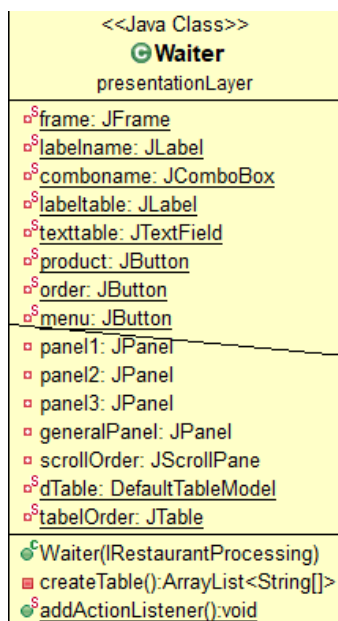
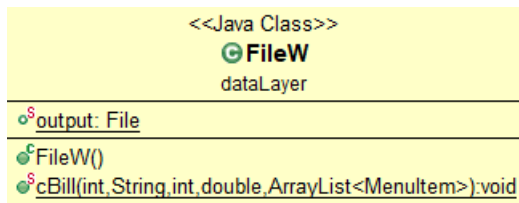


Fig 6.8

În figura 6.8 este prezentată definirea interfeței Waiter prezentă și în capitolele anterioare. Clasa conține variabilele ce ajută la crearea interfeței precum ComboBox-uri, câmpuri de text, butoane și etichete, un tabel, precum și un constructor, o metodă care definește crearea tabelului în fereastra de interfață, o metodă care conține ascultători pentru fiecare buton.

6.9 Clasa FileW



In figura alaturata este prezenta clasa care face posibila scrierea intru-un fisier text. Variabila output reprezinta fisieul in care se vor scrie rezultatele. Aceasta clasa contine un constructor, precum si metoda cBill, metoda pe care o vom apela in clasa Resutaurant si care are ca scop generarea unei note de plata si scrierea acesteia intru-un fisier text. Detaliile pe care le va contine aceasta nota de plata sunt: - identificatorul unic al acesteia, data la care a fost plasata, masa de la care a fost solicitata, o lista cu produsele alese si pretul total al comenzii.

Fig 6.9

6.10 Clasele Restaurant Serializator si Chef

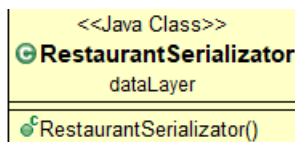


Fig 6.10

Aceste clase sunt niste clase goale, pe care le-am creat doar cu scopul de a pastra structura ceruta in enuntul temei.

7. Rezultate

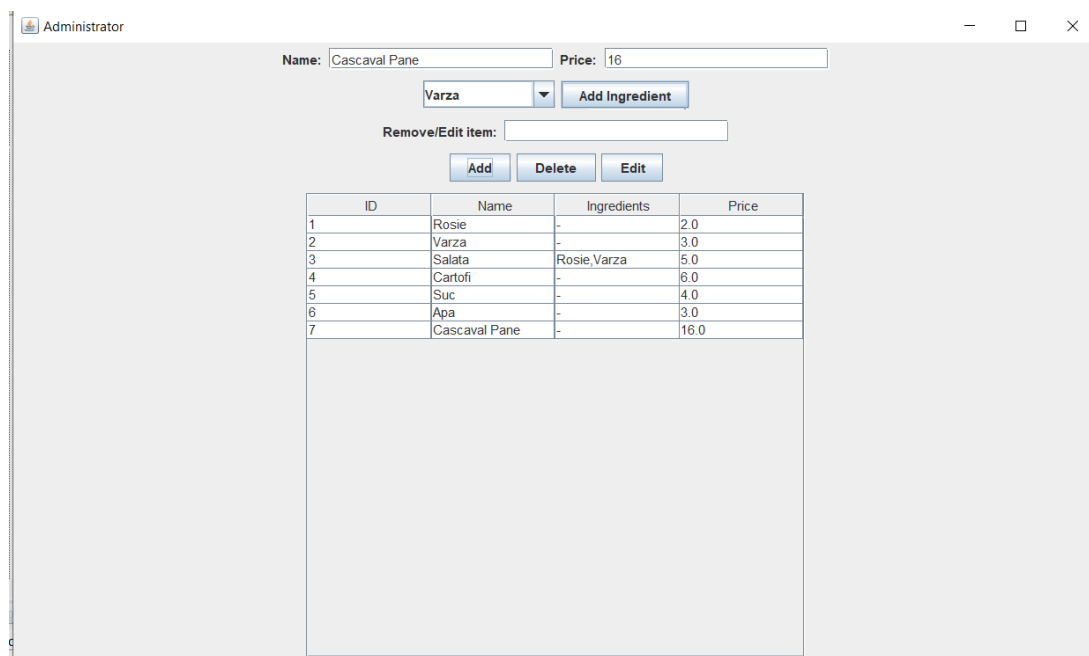


Fig 7.1

In figura de mai sus este prezentata fereastra Administrator. Se poate verifica atat corectitudinea datelor precum si functionalitatea aplicatiei. Aceasta fereasra permite administratorului o mai usoara gestionarea a produselor dintr-un meniu. In exemplul de mai sus s-au aplicate cateva comenzi si au fost urmarite, care au fost identice cu cele asteptate. Prezentarea interfetei a fost realizata mai sus, aici urmand sa fie descries doar tabelul. Prima coloana reprezinta identificatorul unic pentru fiecare produs, cea de a doua coloana ontine denumirea fiecarui produs. A 3-a coloana contine ori caracterul “-“ pentru produsele baseProduct (adica nu contine alte ingrediente), sau o lista care reprezinta totalitatea ingredientelor din produsul respective (in exemplul de mai sus Salata), iar ultima coloana contine pretul- pentru baseProduct este specifecat in momentul in care a fost adaugat, in timp ce pentru un produs compus aplicatie face singura calculul. Rezultatul obtinut este suma tuturor preturilor produselor de baza pe care produsul le contine.

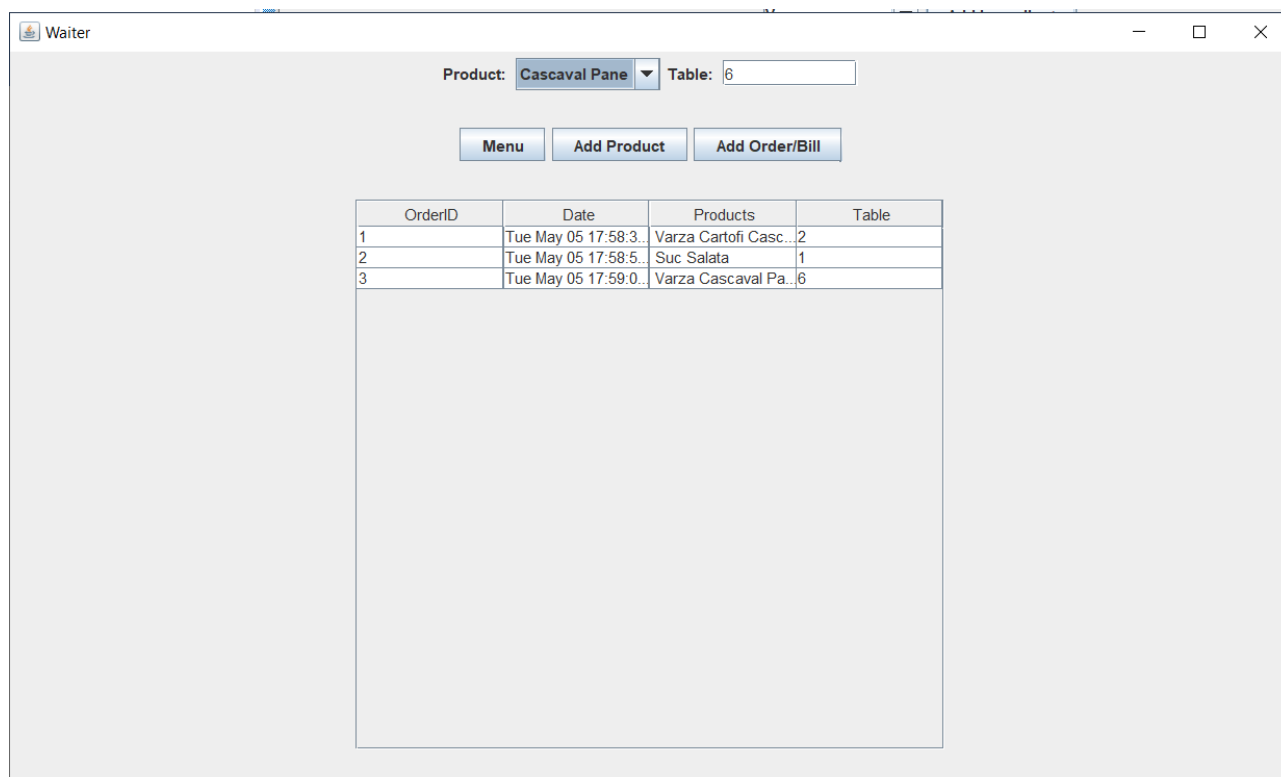


Fig 7.2

In figura de mai sus este prezentata fereastra Waiter. Se poate verifica atat corectitudinea datelor precum si functionalitatea aplicatiei. Aceasta fereasra permite atat clientului cat si administratorului o mai usoara comunicare. In exemplul de mai sus s-au aplicate cateva comenzi si au fost urmarite, care au fost identice cu cele asteptate. Prezentarea interfetei a fost realizata mai sus, aici urmand sa fie descries doar tabelul. Prima coloana reprezinta identificatorul unic pentru fiecare comanda, cea de a doua coloana contine data si ora la care a fost plasata fiecare comanda. A 3-a coloana contine o lista cu produsele care au fost selectate pentru comanda respective, iar ultima coloana contine masa de la care a fost realizata comanda.

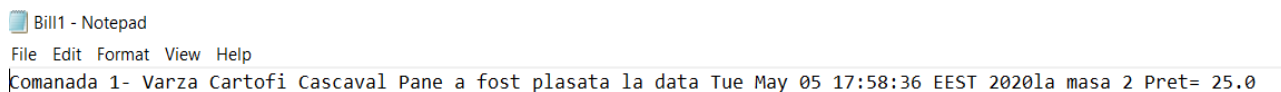


Fig 7.3

In figura 7.3 este prezenta un exemplu de nota de plata, care contine toate datele din tabelul prezentat si in fereastra Waiter, iar in plus este afisat si pretul pentru intreaga comanda.

8. Concluzii si posibilitati de dezvoltare ulterioare

In timpul dezvoltarii acestei teme, adica in timpul realizarii unei interfete grafice care sa vina in atat in ajutorului detinatorilor de restaurant cat si a clientilor si a angajatilor acetuia consider ca am reusit sa aprofundez atat unele paradigme ale programarii orientate pe obiect cat si abilitatile mele de a scrie cod si/ sau a corecta eventualele erori atat de sintaxa cat si la nivel de gandire aparute pe parcursul descrierii algoritmilor necesare functionalitatilor interfete. Pe langa acestea am realizat cat de importanta este citirea cu atentie si intelegerea unei cerinte si capabilitatea de a structura/ imparti problema initiala in mai multe probleme mai mici si inceperea rezolvarii acestora ca un prim pas in indeplinirea sarcinilor unei teme,ca apoi sa asociezi toate rezultatele obtinute din surse pentru a ajunge la un rezultat corect in ansamblu.

O posibilitate de dezvoltare a proiect realizat si descris in acesta documentatie ar putea fi:

- adaugarea unor noi functii care se permita si alte comenzi.
- mai mult de atat s-ar putea face o revizuire asupra tuturor algoritmilor folositi si realizarea implementarii acestora intr-o maniera mai eficienta din punct de vedere a timpului de executie sau a memoriei folosite sau mai usor de descris (pentru programator) folosind functii deja definite ale pachetelor Java, astfel reusind o reducere a timpului petrecut pentru asigurarea functionarii unui astfel de aplicatii.
- o alta posibilitate de dezvoltare a proiectului ar putea fi imbunatatirea interfetei grafice apeland la partea artistica a fiecarui programator

9. Bibliografie

<https://www.geeksforgeeks.org/java-swing-jtable/>

<https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/>

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