

APLICATIE PENTRU MANAGEMENT- UL UNUI RESTAURANT DOCUMENTATIA TEMEI

Student: Bivolaru Alexandru

Grupa: 30225

Profesor laborator: Moldovan Dorin

Cuvant inainte!

- Pentru a putea folosi aplicatia sunt necesare doua conturi, unul pentru admin si unul pentru waiter. Datele de logare pentru cele doua conturi sunt:
Admin) user: **admin**, parola: **admin**
Waiter) user: **waiter**, parola: **waiter**
- Unele probleme care nu afecteaza functionalitatea programului au fost sesizate si trecute in revista la finalul documentatiei in sectiune de “Posibilitati de dezvoltare ulterioara”.
- Pentru a intelege cum functioneaza aplicatia va recomand sa urmariti tutorialul cu imagini atasat in explicatia interfetei grafice (GUI).

CUPRINS

Capitolul 1.....	pg 4
Capitolul 2.....	pg 4
Capitolul 3.....	pg 6
Capitolul 4.....	pg 8
Capitolul 5.....	pg 16

1. Obiectivul temei (Capitolul 1)

Obiectivul temei presupune implementarea unui sistem de administrare al unui restaurant. Aplicatia simuleaza inserarea, modificarea, eliminarea de produse din meniul unui restaurant, inserarea si stergerea (generarea unei chitante) a unor comenzi, si totodata un sistem de notificare a bucatarului de fiecare data cand o comanda formata din cel putin un obiect compus a fost plasata.

2. Analiza problemei (Capitolul 2)

Pasii de dezvoltare a problemei reprezinta lista de etape care trebuie urmata pentru a indeplini obiectivul temei. Pasii aleși pentru dezvoltare sunt, in ordine, urmatoarii

- **Crearea de scenarii din lumea reala.** Acest obiectiv presupune relationarea cu utilizatorii aplicatiei (administrator, chelner, respectiv bucatar). Fiecare trebuie sa isi gestioneze activitatea sa intr-un mod cat mai optim. Administratorul trebuie sa aiba posibilitatea de a adauga produse in meniu, de a le edita sau de a le sterge in functie de cerinte. Chelnerul trebuie sa adauge, respective genereze chitante (stearga) comenzi, iar bucatarul trebuie sa fie notificat de fiecare data cand o comanda voluminoasa (care contine cel putin un produs compus) a fost plasata. Vom realiza aceste scenarii cu ajutorul unor diagrame UML.
- **Alegerea tipurilor / structurilor de date.** In continuare, in partea aceasta a proiectului, trebuie sa ne decidem asupra caror structuri de date vor fi folosite pentru indeplinirea obiectivului principal al temei.
- **Crearea claselor / dezvoltarea algoritmilor.** Cel mai important pas din cadrul acestui proiect este cel in care ne decidem asupra crearii claselor si a dezvoltarii algoritmilor. Acestia asigura buna intelegere a codului si totodata buna functionare a aplicatiei.

Diagrama UML a claselor (cea prevazuta in cerinta si respectata in cadrul proiectului)

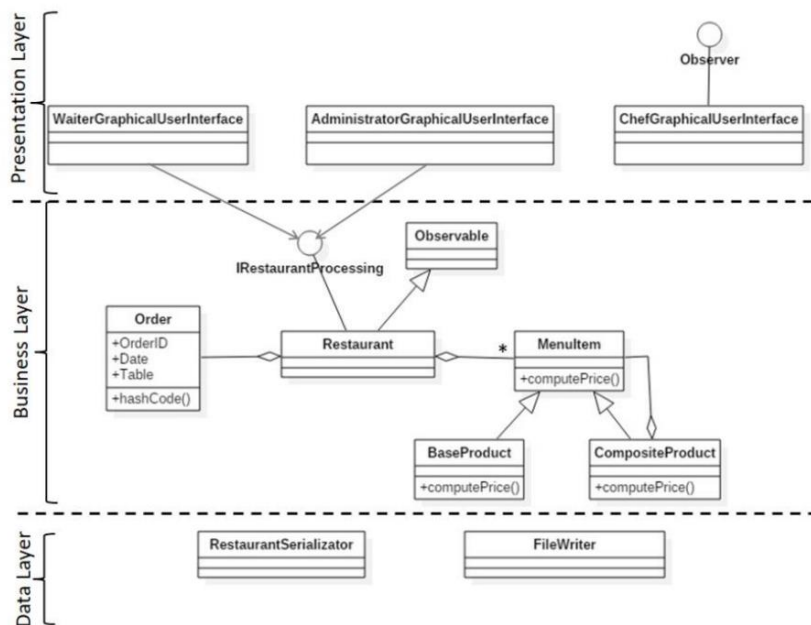
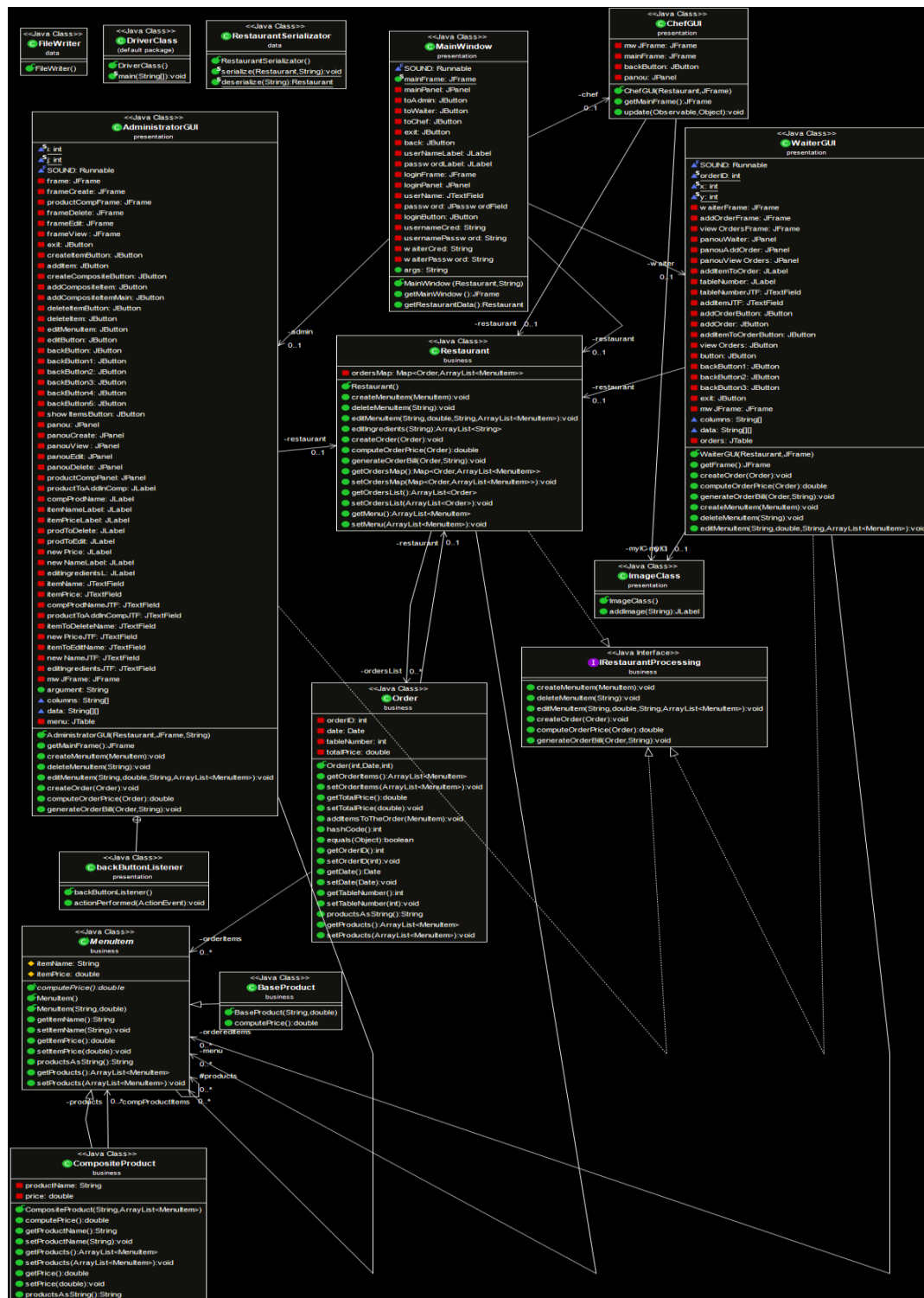


Diagrama UML a claselor proiectului (împreună cu clasele ajutoare)



3. Proiectare (Decizii de proiectare, structure de date, proiectare clase, algoritmi) (Capitolul 3)

3.1 Decizii de proiectare

Utilizatorului ii este cerut sa introduca un cont dupa ce a ales ce tip de utilizator este. Daca este administrator, acesta va avea panoul sau cu functionalitatile specifice rolului sau. Acesta va putea sa introduca produse in meniu,, fie ele produse brute (ex. Paine, lapte, oua etc.), fie ele produse compuse (ex. Oua ochi cu sunca). Totodata acesta poate sa editeze produsele (numele / pretul acestora) sau chiar sa le stearga din meniu. Chelnerul poate sa introduca noi comenzi, iar bucatarul este notificat de fiecare data cand o noua comanda a fost plasata.

- Am creat o interfata **IRestaurantProcessing** care contine operatiile de baza care trebuie executate de Administrator/Chelner, specificate mai sus.
- Am folosit Observer Design Pattern (chiar daca este deprecated, pentru scop didactic am folosit interfata **Observable**).
- Am implementat clasa **Restaurant** in cadrul careia se implementeaza metodele din interfata **IRestaurantProcessing**. Restaurantul este sub forma de **Map<Order, Collection<MenuItem>>**.
- Am serializat clasa **Restaurant** pentru a putea salva datele din Restaurant la fiecare rulare a aplicatiei.
-

3.2 Structuri de date

Pentru a asigura buna functionare a aplicatiei suntem nevoiti sa folosim structuri de date eficiente care sa ne permita controlul fluxului de date care se petrece intre clasele aplicatiei. Structurile de date folosite in realizarea acestui proiect sunt **HashMap** pentru restaurant, unde cheia fiecarei intrari in Map este un **Order**, **ArrayList** pentru stocarea meniului restaurantului, deoarece ne ofera o buna flexibilitate asupra controlului datelor in interiorul listelor. Folosesc de asemenea **FileWriter** pentru scrierea in fisierul de tip .txt si **JTable** pentru stocarea datelor in tabelele din cadrul interfetei utilizatorului. Cam acestea sunt in mare structurile de date preponderente in cadrul proiectului.

3.3 Proiectarea claselor si a pachetelor

Pachetul business

Pachetul business contine clasele:

- **MenuItem** – Clasa abstracta care se ocupa de modelarea “generica” a unui produs, fie el **BaseProduct**, fie **CompositeProduct**. Aceasta clasa contine pe langa clasicele settere si gettere si o metoda **productsAsString()** care ne ajuta sa returnam produsele din cadrul unui composite product sub forma de string, lucru folositor atunci cand afisam lista de produse in meniu.
- **BaseProduct & Composite product** – Clase care extind MenuItem pentru a putea accesa attributele specific unui produs si metoda **computePrice**, pe care fiecare din aceste doua clase o implementeaza in functie de tipul produsului.
- **Order** – Clasa care simuleaza o comanda. Are ca attribute **orderId**, **date**, **tableNumber**, **orderItems**, **restaurant** si **totalPrice**. Functia de **hashCode** implementata in Order este o functie clasica, preluata din documentatia de la **HashMap** (sursa bibliografie)
- Despre clasa **Restaurant** s-a amintit foarte mult in decursul acestei documentatii astfel incat sa ne facem o privire de ansamblu a ceea ce trebuie sa reprezinte aceasta, la fel ca si interfata **IRestaurantProcessing**.

Pachetul data

Pachetul data contine clasele:

- **FileWriter** (clasa goala deoarece am folosit direct un obiect de tip **FileWriter** din pachetul standard).
- **RestaurantSerializer** – clasa in care am implementat metodele de serializare si deserializare folosite pentru salvarea si incarcarea datelor in clasa **Restaurant** la fiecare inchidere si redeschidere a aplicatiei.
-

Pachetul presentation

Pachetul presentation contine clasele folosite pentru interfata Grafica a utilizatorului (GUI). In cadrul claselor din acest pachet au fost implementate diferite functionalitati care actualizeaza tabelele si interfata cu informatia stocata in structurile de date din Restaurant. Singurele clase despre care voi vorbi in cadrul acestui capitol sunt **ImageClass** si **MainWindow**. Clasa ImageClass este responsabila pentru adaugarea a doua imagini in cadrul interfetei grafice pentru Chef si Waiter sub forma de JLabel (sursa in bibliografie). In clasa MainWindow se leaga toate cele 3 panouri pentru fiecare tip de angajat cu ajutorul unui JFrame declarat static pentru a putea fi vizibil in toate clasele care modeleaza interfata grafica.

3.4 Dezvoltarea algoritmilor

Algoritmii cei mai importanti folositi in cadrul dezvoltarii acestui proiect au fost conceputi in felul urmator:

- **Adaugarea in tabelele JTable. (sursa in bibliografie).**

Un tabel este de forma (columns/data) unde data este un array bidimensional cu datele tabelului JTable. Pentru a insera produse in tabelul meni / orders am folosit 2 variabile statice I,j , respective x,y, iar valorile atributelor obiectelor au fost parsate sub forma de string, astfel ca data[i][j] (idem si la x,y) reprezinta numele produsului, data[i][j+1] reprezinta pretul produsului iar data[i][j+2] reprezinta lista cu numele produselor folosite in cadrul produselor composite.

- **Eliminarea din tabelele JTable**

Eliminarea din tabel se face decrementand variabila statica I (variabila care reprezinta linia din table), cu numarul de elemente sterse. **Acest lucru a fost facut dupa o reverificare a functionalitatilor si observarea unor exceptii de tip IndexOutOfBounds aruncate.** O decrementam cu numarul de elemente sterse deoarece daca stergem de exemplu doua elemente la rand si apoi adaugam unul, variabila I s-ar fi decrementat cu 1 in loc de 2 si atunci am fi fost in zona de exceptie.

- **Afisarea in tabelele JTable**

Afisarea in tabele se face parcurgand lista de elemente din meniul restaurantului si setand in fiecare celula elementele conveniente. Acest lucru se face mai “neconventional”, de fiecare data cand se apasa pe buton se creeaza o noua interfata cu un nou tabel care se suprapune peste cel vechi. **(De aceea la stergere se poate doar decrementa i-ul, deoarece dupa ce acesta a scazut, daca dam sa revizualizam meniul se va recrea un tabel cu noile date actualizate care il va suprapune pe cel vechi, totodata aici am realizat ca daca nu fac acest lucru in buton, de fiecare data cand dau login la admin se vor redeschide si mai multe tabele cand apasam sa vedem meniul).**

- **Crearea unui nou Order**

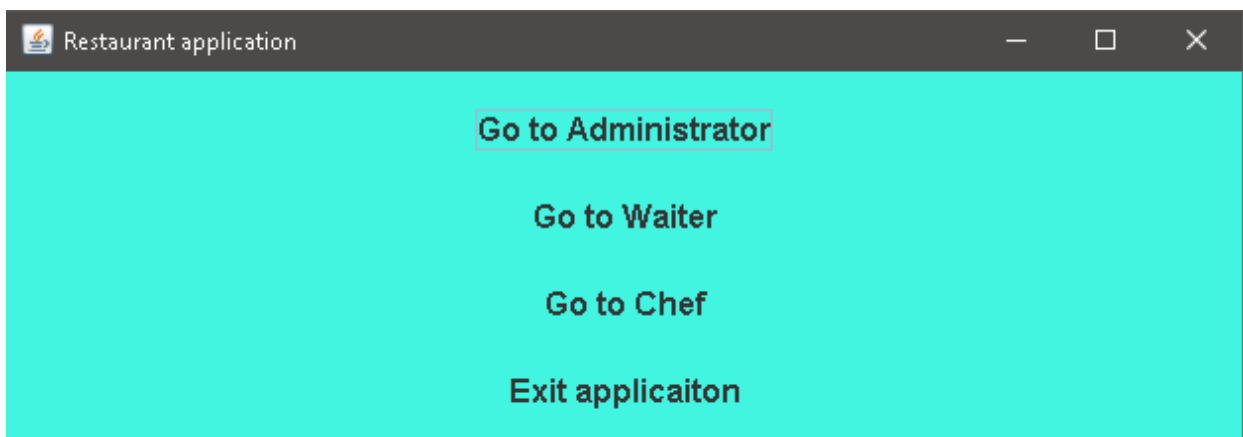
Crearea unui nou order se face incrementand o variabila statica OrderID, astfel incat de fiecare data cand apelam generateOrderBill sa nu existe sansa sa se suprapuna doua chitante cu acelasi nume (exact acelasi principiu cu cel de la Tema 3).

- **Procesarea facturii pentru un Order**

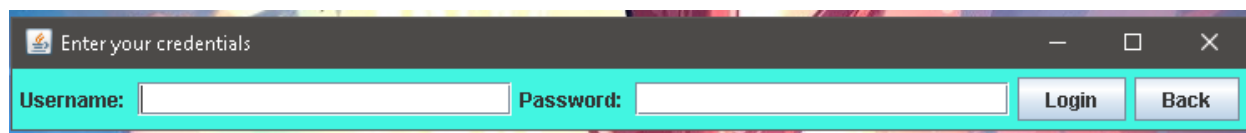
Se folosesc StringBuilder, FileWriter si PrintWriter. StringBuilder pentru a crea mesajul ce urmeaza sa fie printat in fisierul bill.txt, FileWriter si PrintWriter sunt obiectele clasice pentru citire/scriere in fisier. Metoda aleasa pentru a genera bill-ul este urmatoarea: Selectam comanda din tabel (dand click pe randul cu comanda) si apasam butonul GenerateBill. Pentru a realiza selectia am folosit SelectionModel-ul SINGLE_SELECT din javax.swing si o comparare cu -1 a randului selectat (semn ca am selectat un rand si nu apasam in gol). Extragem datele din randul respective si cream un nou order “dummy” pe care il cautam in lista cu comenzi a restaurantului de unde dupa ce am generat facture il stergem.

4. Prezentarea si folosirea Interfetei grafice pentru utilizatori (GUI) (Capitolul 4)

1) Panoul de intampinare



In cadrul acestei ferestre, utilizatorul este rugat sa aleaga unde doreste sa isi continue activitatea. Daca se doreste a continua pe ramura de **Administrator** sau cea de **Waiter**, utilizatorul va fi trimis la o interfata de logare.

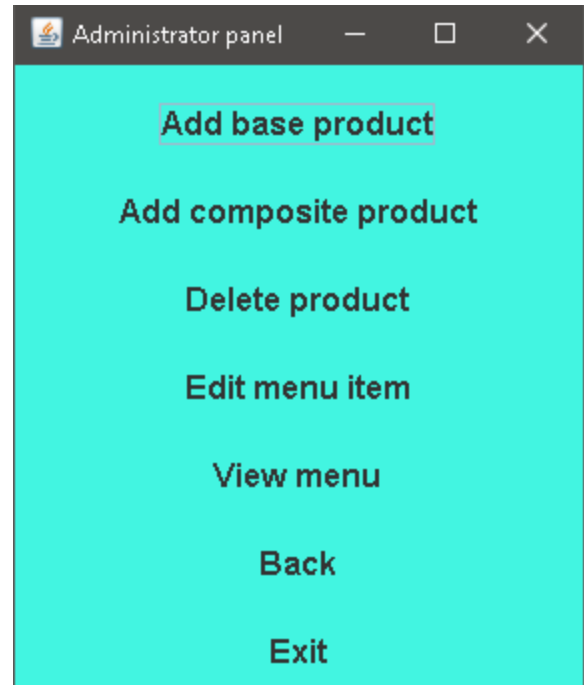


Conturile pentru logare au fost precizate in "Cuvant inainte", dar le voi relua si aici (user: admin, pass: admin sau user: waiter, pass: waiter).

Daca utilizatorul a decis sa foloseasca ramura de Administrator, acesta va fi "teleportat" la panoul specific Administratorului.

2) Panoul pentru administrator

În această fereastră, utilizatorul este rugat să aleagă operația pe care dorește să o efectueze (adaugarea unui produs “brut”, a unui produs compus, să ștergă/editeze un produs existent sau să vizualizeze meniul restaurantului).



În continuare urmează să explic fiecare funcționalitate cum trebuie efectuată pentru a vedea rezultatele corecte.



Primă dată inserăm numele produsului și prețul acestuia, apoi apăsăm pe butonul ADD pentru a-l insera în meniu.

Urmează să explic adăugarea unui produs compus.

Administrator panel

Base product name: Composite product name:

Pasii trebuie urmati in felul urmator.

- Prima data introducem numele unui “Base product” (poate fi chiar si composite) pe care dorim sa il adaugam
- Apasam pe ADD Product pentru a adauga produsul in lista de produse care compun acel Composite product
- Repetam pasii 1 si 2 pentru fiecare item pe care vrem sa-l adaugam.
- Dupa ce am terminat lista de produse, dam un nume produsului compus pe care dorim sa-l adaugam (3).
- La final apasam ADD Comp pentru a insera produsul in meniu.

Pentru a sterge un produs, inseram numele acestuia si apasam Remove.

Administrator panel

Product name: New price: New name: New ingredients:

Pasii pentru a edita un obiect sunt urmatoarii:

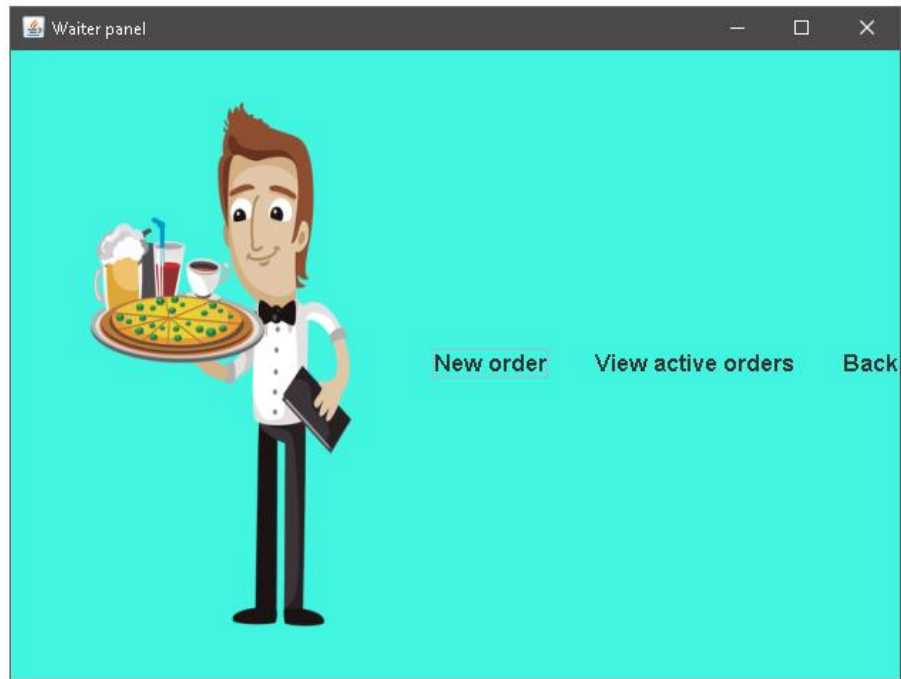
- Inseram numele produsului pe care dorim sa il editam
- Apoi alegem ceea ce dorim sa editam (pret, nume, ingrediente) **SE POT SI TOATE TREI DEODATA**
- **DACA DORIM SA EDITAM LISTA DE INGREDIENTE, ACEASTA TREBUIE SA RESPECTE URMATORUL PATTERN**
INGREDIENT, [SPATIU]INGREDIENT, [SPATIU]INGREDIENT
- Apasam pe butonul Edit Item ca sa finalizam editarea produsului.

Pentru a vizualiza meniul apasam pe View Menu (**ATENTIE! MENIUL DIN EXEMPLU DIFERA DE CEL DIN RESTAURANT.SER DEOARECE AM MAI RETESTAT UNELE FUNCTIONALITATI**).

[illegible]

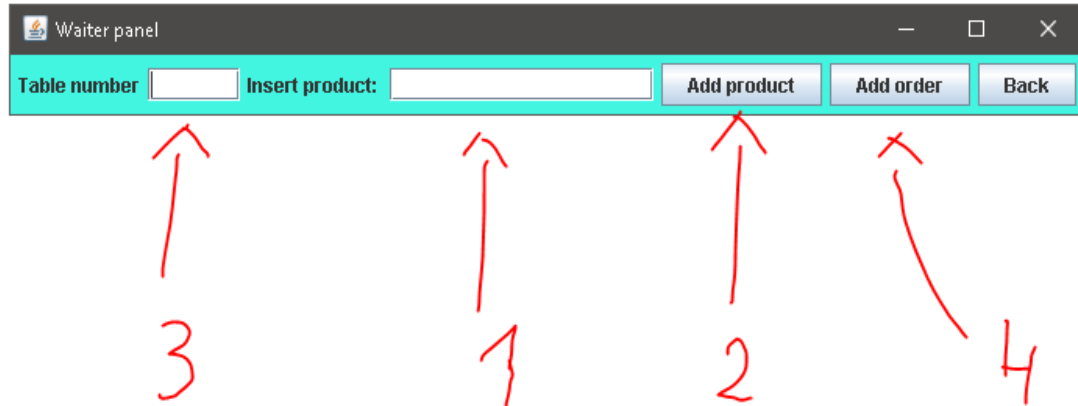
Urmeaza in continuare sa discutam despre panoul de la Chelner si cel de Bucatar.

3) Panoul chelnerului



In cadrul acestui panou treaba este foarte simpla, avem de ales intre a crea o comanda noua sau a le vizualzia pe cele existente (unde putem sa generam chitanta daca se doreste).

Prima data vom discuta despre modul in care cream un nou order:



Pasii pentru inserarea unei noi comenzi sunt, in ordine:

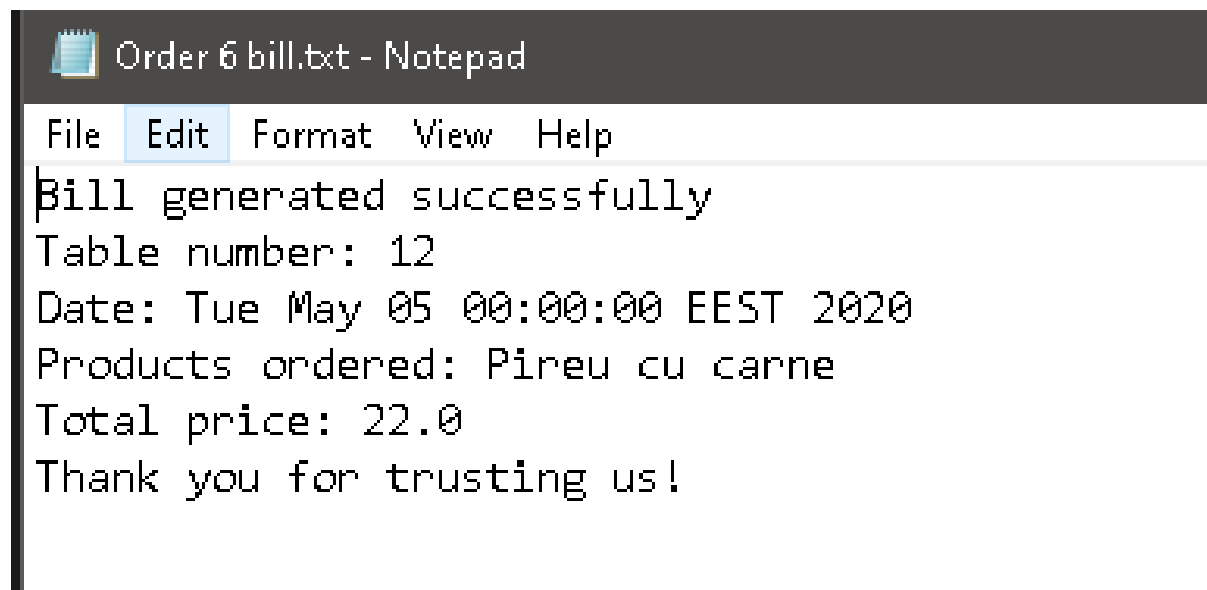
- Introducem (1 + 2) obiectul pe care dorim sa-l atasam comenzii. Acesta va fi introdus in lista cu obiecte specifica fiecarei comenzi in parte.
- Repetam pasii 1 si 2 pana "clientul" nu mai doreste sa comande nimic.
- Dupa ce am adaugat toate obiectele in comanda, selectam numarul mesei (3) si apasam pe Add order(4).

Daca, comanda contine un produs compus, bucatarul va fi imediat anuntat cu ajutorul Observer Design Pattern. (Se putea de asemenea ca bucatarul sa fie anuntat chiar daca, comanda nu contine neaparat un produs compozit, **a fost mai mult o decizie de implementare**).

- Use the Observer Design Pattern to notify the chef each time a new order containing a composite product is added.

Window for Chef user: use Observer Design Pattern to notify each time a new Order is added

Daca se selecteaza randul cu comanda dorite si se apasa butonul Generate Bill, rezultatul va fi generat in fisierul cu proiectul. Un exemplu de factura este urmatorul:



```
Order 6 bill.txt - Notepad
File Edit Format View Help
Bill generated successfully
Table number: 12
Date: Tue May 05 00:00:00 EEST 2020
Products ordered: Pireu cu carne
Total price: 22.0
Thank you for trusting us!
```

Ora nu este in regula deoarece cand am selectat DateFormat am selectat "yyyy-MM-dd".

***ADAUGAT LA REVIZUIREA DOCUMENTATIEI (TABELUL CU TOATE COMENZILE DE DINAINTEA GENERARII FACTURII)**

Concluzii, probleme, posibilitati de dezvoltare ulterioara, bibliografie

(Capitolul 5)

Dintre toate temele rezolvate pana acum, aceasta mi s-a parut cea mai frumoasa.

Am invatat despre serializare, cum sa lucrez mai bine cu JTable, cum sa adaug sunete pentru JOptionPane (**Se gasesc in clasele AdministratorGUI, WaiterGUI si MainWindow**). Am invatat totodata cum sa inserez imagini cu ajutorul JLabel si cum sa le fac rendering astfel incat imaginea sa iasa clar. Am invatat despre Composite Design si despre Observer Design Pattern, care mi se pare cel mai folositor de pana acum si ne ajuta sa scapam de multe probleme. Am invatat sa lucrez cu hashmap-uri (chiar daca nu le-am folosit excesiv in aceasta tema, am incercat sa mai fac printari pentru a observa cum se comporta), dar cel mai important lucru, am invatat cum sa gestionez si mai bine clasele si functionalitatile fata de cum stiam pana in momentul de fata.

Problemele descoperite pe parcursul testelor (+ posibilitati de dezvoltare ulterioara).

- Problemele descoperite la capitolul dezvoltare algoritmi (marcate cu rosu – AU FOST REMEDIATE)
- Faptul ca daca **EXEMPLU**: Avem un composite product format din Lapte si Paine si il numim “Paine cu Lapte”, iar noi mai facem un produs compus care contine Paine cu Lapte, pe care il numim “Lapte cu Paine”, noi daca stergem Laptele sau Painea, nu se va sterge decat Base Product-ul si produsele compuse care contin direct Lapte si Paine, nu si cele care contin produse compuse din Lapte si Paine.
- O idee de dezvoltare ulterioara ar fi atunci cand la o masa se plaseaza mai multe comenzi sa se adune toate pe o singura chitanta (daca se doreste acest lucru) pentru a nu aglomera tabelul generat.

Bibliografie

https://www.tutorialspoint.com/java/java_serialization.htm

<https://javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.html>

<https://www.tutorialspoint.com/how-can-we-remove-a-selected-row-from-a-jtable-in-java>

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

https://www.tutorialspoint.com/java/java_hashtable_class.htm

<https://stackoverflow.com/questions/5920135/printing-hashmap-in-java>

<https://stackoverflow.com/questions/20737064/how-do-i-position-jbuttons-vertically-one-after-another>

<https://stackoverflow.com/questions/33269847/trying-to-get-two-buttons-drawn-on-top-of-each-other>

<https://stackoverflow.com/questions/12128231/using-windows-sounds-when-displaying-joptionpane-windows>

<https://stackoverflow.com/questions/3775373/java-how-to-add-image-to-jlabel>