Documentatie

Restaurant management system



Tehnici de programare

Tema4

Realizat de Balc Horia-Ovidiu

Grupa 30227

Cuprins

1 . Cerinte Functionale

2 . Obiectivul temei

3 . Analiza problemei

4 . Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

5 . Implementare

6 . Concluzie

7 . Bibliografie

1 . Cerinte Functionale

Proiectati si implementati un model de aplicatie a unui restaurant.

Programul:

-va folosi clasa restaurant care contine un administrator ( clasa implementata cu ajutorul unei interfete AdministratorInterface) si un waiter ( clasa implementata cu ajutorul unei interfete WaiterInterface)

-folosirea operatiilor cu baze de date ( inserarea, stergerea si modificarea datelor din tabele)

- utilizarea unui jar file

-implementarea clasei Order

-implementarea celor trei interfete grafice ( pentru administrator, waiter si chef)

-Administratorul poate sa adauge, sa stearge, sa editeze un item din meniu si sa vada toate itemele din meniu intr-un tabel JTable

-Waiterul poate sa creeze o noua comada (Order), sa vada toate comenzile existente sis a calculeze nota de plata

-Chef-ul este instiintat (cu ajutorul interfetei Observable si a clasei predefinite Observer) de catre Waiter cand trebuie sa gateasca (doar in cazul in care acesta din urma a primit in comanda cel putin un composite product)

-utilizarea Composite Design Pattern pentru clasele MenuItem, BaseProduct, CompositeProduct (MenuItem este super clasa, iar celelalte 2 subclase, in CompositeProduct existand o lista de MenuItem)

-Afisarea notii de plata intr-un fisier de format txt

-creearea java doc-ului pentru clasa restaurant si folosirea de asserturi

-folosirea serializarii si deserializarii pentru a se salva datele si dupa inchiderea programului

-impartirea proiectului pe pachete

2 . Obiectivul temei

2.1 Obiectivul principal

Obiectivul temei este crearea, in limbajul JAVA, a unui model de aplicatie pentru un restaurant in care datele se salveaza cu ajutorul serializarii si deserializarii. Aceasta tema are practicabilitate in viata reala si are multe obiective importante care stau la baza programarii in limbajul JAVA.

2.2 Obiectivul secundare

Obiectivele secundare sunt: - folosirea programarii obiectuale prin utilizarea incapsularii

- scrierea variabilelor in stilul Camel Case

- utilizarea unui jar file

-utilizarea serializarii si deserializarii

-utilizarea clasei predefinite Observer si a intrefetei Observable

-scrierea in fisier a datelor

-impartirea proiectului pe pachete

-crearea unui java-doc si folosirea aserturilor

3 . Analiza problemei

In aceasta sectiune vom trata anumite scenarii. Eu am proiectat programul in felul urmator: in clasa restaurant retin hash-map-ul, un obiect de tipul RestaurantSerializator un administrator in care se retine meniul si unde sunt definite metodele din interfata si un waiter in care se retin orderurile si unde sunt definite metodele din interfata.

4 . Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

4.1 Decizii de proiectare si Structuri de date

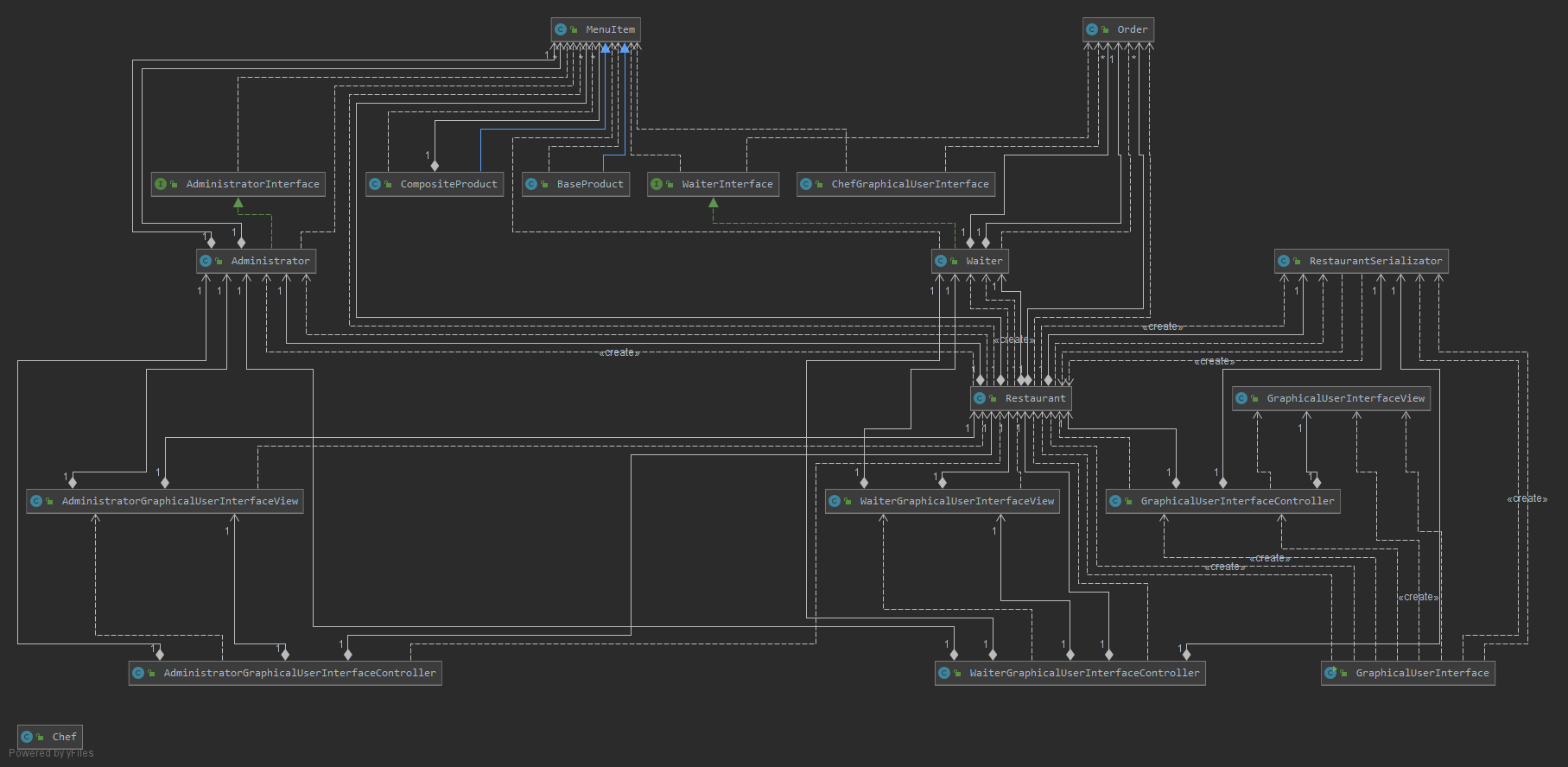
Am decis sa modelez acest proiect cu ajutorul a 3 pachete:

-businessLayer unde au loc creeare majoritatea claselor si metodelor importante din program

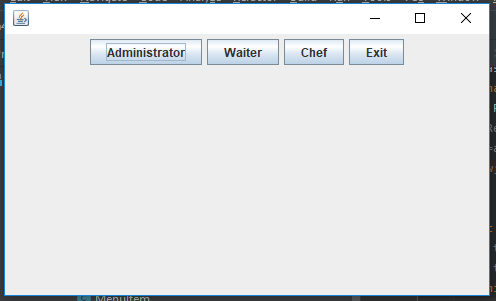
-presentationLayer unde sunt facute interfetele (meniu, Administrator,Waiter si Chef) proiectate cu ajutorul modelului MVC

-dataLayer unde este facuta serializarea si deserializarea

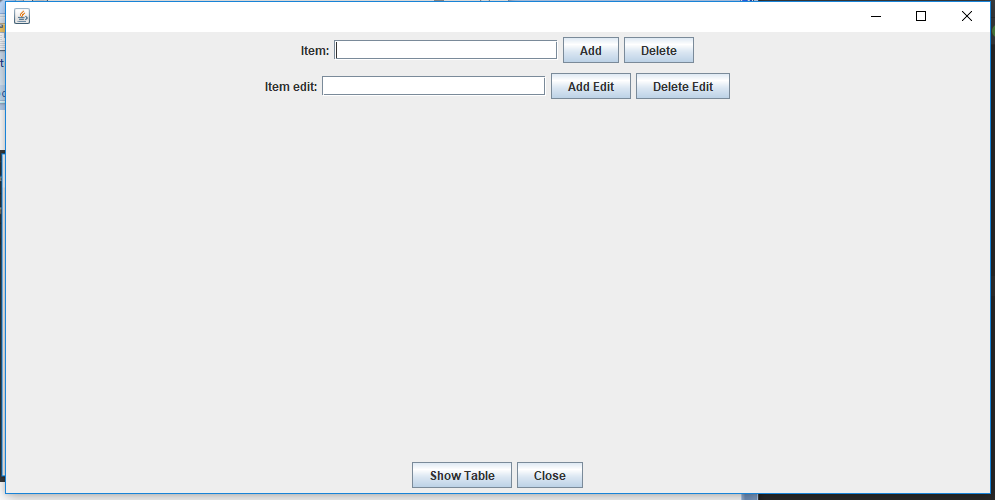
4.2 Diagrame UML



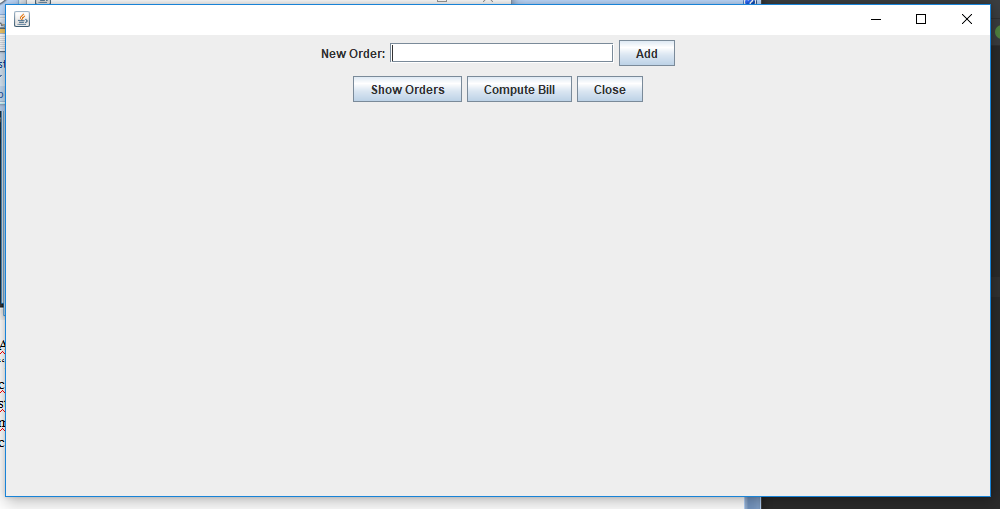
4.3 Utilizarea interfetei



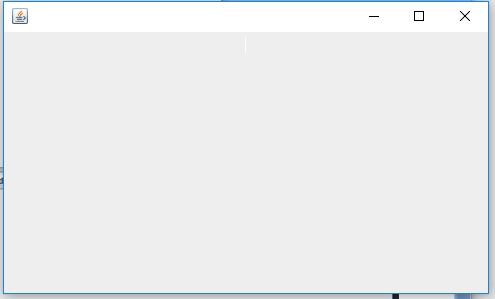
Acesta este meniul de unde se decide printr-o apasare de buton ce functie are persoana care utilizeaza programul (administratot, waiter sau chef) si respective posibilitatea de a-l inchide.



Aceasta este interfata pentru Administrator. Pentru butonul Add trebuie scris fie in primul textbox itemul in formatul “nume,pret” in cazul in care este BaseProduct, fie in primul textbox “nume” si in cel de-al doilea textBox numele componentelor produsului. Pentru butonul Delete trebuie scris in primul textbox numele itemului pe care dorim sa il stergem. Pentru butoanele DeleteEdit si AddEdit trebuie scris in primul textbox numele itemului pe care dorim sa il modificam, iar in al doilea sa scriem numele elementului pe care dorim sa il stergem respectiv sa il adugam in composite-ul product dat in primul textbox . ShowTable este un buton la apasarea caruia se face un tabel intr-o alta fereastra(JFrame) in care sunt toate elementele din meniu. Close este un buton ce inchide fereastra, neoprind executia programului.



Aceasta este interfata pentru waiter. In textbox se asteapta urmatorul format “ id,data,masa,elemente comandate ” pentru butonul de Add. ShowOrders este butonul care afiseaza un tabel in alta fereastra (JFrame) in care se afla toate comenzile.



Aceasta este interfata pentru chef. Aici se afiseaza in TextArea daca chef-ul trebuie sa gateasca (adica daca exista in comanda un preparat compus ).

5 . Implementare

Pachetul businessLayer

-Clasa Administrator implementeaza interfata AdministratorInterface si interfata default Serializable. Aceasta Clasa are metodele:

-delete care are rolul de a sterge din lista de meniuri incorporata in aceasta clasa un MenuItem primit ca paramentru in aceasta metoda.

-addToMenu care are rolul de a adauga in lista de meniuri un obiect de tipul MenuItem primit ca parametru in aceasta metoda.

-isWellFormed are rolul de a verifica metodele enuntate mai sus, cu ajutorul lui assert

-Interfata AdministratorInterface are urmatoarele metode declarate: delete (care primeste ca parametru un obiect de tipul String s), addToMenu(care primeste ca parametru in obiect de tip MenuItem), isWellFormed fara parametrii.

-Interfata WaiterInterface are urmatoarele metode declarate: computePrice (care primeste ca parametru un obiect de tipul Obiect si unul de tip Hash-Map), generateBill(care primeste ca parametru un obiect de tipul Obiect si unul de tip Hash-Map), isWellFormed fara parametrii.

Clasa Waiter implementeaza interfata WaiterInterface si interfata default Serializable. Aceasta Clasa are metodele:

-createOrder care are rolul de a adauga in HashMap un nou element (format dintr-o lista de obiecte de tip-ul MenuItem si de un obiect de tipul order) primit ca parametru. Metoda returneaza noul HashMap.

-computePrice este metoda ce are rolul de a calcula nota de plata prin parcurgerea HashMap-ului primit ca parametru si verifica daca Obiectul de tip order (si el trimis ca parametru) are acelasi id cu order-urile existente in HashMap. Daca da, atunci se adauga in fisier-ul txt si se adauga pretul comenzii la pretul curent (calculate pana in momentul acela). Metoda returneaza un double ce reprezinta pretul comenzii.

-generateBill este metoda ce apeleaza metoda computePrice

-isWellFormed are rolul de a verifica metodele enuntate mai sus, cu ajutorul lui assert

Clasa MenuItem implementeaza interfata default Serializable. Aceasta Clasa are metodele:

-getteri si setteri pentru prêt si nume

-computePrice este folosit in aceasta clasa ca un geter de prêt

-toString returneaza un String care scrie Numele si Pretul din aceasta clasa

Clasa BaseProduct extinde clasa MenuItem implementeaza interfata default Serializable. Aceasta Clasa are metoda:

-computePrice apeleaza functia cu acelasi nume din super

Clasa CompositeProduct extinde clasa MenuItem implementeaza interfata default Serializable. Aceasta Clasa are metodele:

-competePrice care are rolul de a calcula pretul total al itemu-lui prin parcurgerea listei de MenuItem si se adauga pretul fiecarui produs la pretTotal care se reurneaza

-adauga care are rolul de a adauga MenuItemul primit ca parametru in ArrayList-ul de MenuItem si se seteaza pretul apeland metoda explicate mai sus

-sterge are rolul de a sterge MenuItem primit ca parametru din lista de MenuItem a CompositeProduct. Dupa care se face computePrice din aceeasi clasa

-find are rolul de a verifica daca Stringul primit ca parametru se afla in componenta a obiectului de tip CompositeProduct. In cazul in care se gaseste se returneaza true, in caz contrar false.

- getter si setter pentru ArrayListul de MenuItem ce reprezinta componentele CompositeProductului

Clasa Order implementeaza interfata default Serializable. Aceasta Clasa are metodele:

-hashCode care returneaza hash code in functie de OrderId si de Table

-getteri pentru fiecare variabila existent intr-un obiect de tip Order ( OrderId, Data, Table )

Clasa Restaurant implementeaza interfata default Serializable. Restaurant are un obiect de tip RestaurantSerializator. Aceasta Clasa are metodele:

-met are rolul de a face serializarea, daca int primit ca parametru este 0 atunci se face serializarea pentru administrator si waiter

-getteri getAd, getLista si getW pentru W ( Waiter ), Lista( HashMap de Order si ArrayList de MenuItem)

si pentru Ad ( Administrator )

-setteri setW, setAd si setLista pentru W ( Waiter ), Ad ( Administrator ) si pentru Lista (HashMap de Order si ArrayList de MenuItem )

Pachetul dataLayer

Clasa Restaurant implementeaza interfata default Serializable. Aceasta Clasa are metodele:

-serializarea are rolul de a serializa obiectul de tip restaurant primit ca parametru, intr-un fileOut FileOutputStream “ restaurant.ser ” si se creeaza un obiect de tip ObjectOutputStream cu fileOut ca parametru

-deserializarea are rolul de a deserializare care returneaza un obiect de tip Restaurant. Initializam un fisier “restaurant.ser” si FileInputStream fileIn(cu “restaurant.ser” ca parametru ) si ObjectInputStream in (cu fileIn ca parametru ) dupa care restaurantul care trebuie sa fie returnat ia valoarea la in.readObject (metoda predefinita) avand cast la obiectul de tip Restaurant.

` Pachetul presentationLayer

Clasa GraphicalUserInterface este clasa care contine metoda main. Se initializeaza un obiect nou de tip Restaurant numit r,un obiect nou de tip GraphicalUserInterfaceView numit view, un obiect nou de tip GraphicalUserInterfaceController numit controller avand ca parametru pe view si pe r. Dupa care se seteaza view ca fiind vizibil cu ajutorul functiei predefinite setVisible( true).

Clasa GraphicalUserInterfaceView este clasa ce contine 4 butoane ,pe care cu ajutorul functiilor predefinite in JFrame pe care o extinde aceasta clasa. Metodele din aceasta clasa sunt listenerele de la cele 4 butoane aflate pe prima interfata prezentata in sectiunea 4.3.

Clasa GraphicalUserInterfaceController initializeaza in constructor cele 4 listeneruri mentionate in clasa explicate mai sus. Aceasta clasa contine clase care implementeaza ActionListener si care implementeaza listenerurile pentru fiecare buton din cele 4 de pe interfata principal. Listenerul pentru administrator initializeaza un administrator cu restaurant.getAd si un AdministratorGraphicalUserInterfaceView a care primeste ca parametru restaurant si un AdministratorGraphicalUserInterfaceController c care primeste ca parametru variabilele restaurant si a. . Listenerul pentru Waiter initializeaza un waiter cu restaurant.getW si un WaiterGraphicalUserInterfaceView a care primeste ca parametru restaurant si un WaiterGraphicalUserInterfaceController c care primeste ca parametru variabilele restaurant si w. Listenerul pentru Chef initializeaza un obiect de tip ChefGraphicalUserInterface chef si se foloseste metoda addObserver pentru a-l pune pe chef sa astepte sa preia o comanda.

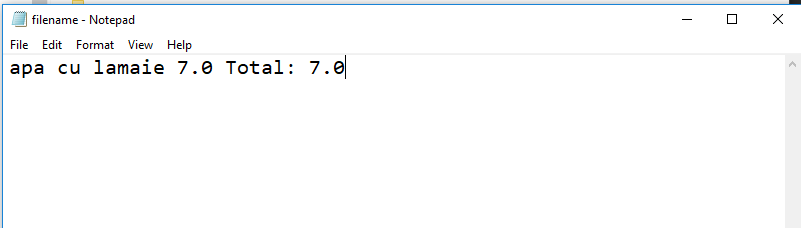
Clasa AdministratorGraphicalUserInterfaceView creeaza interfata cum este aratata in sectiunea 4.3 si folosit metodele Listener pentru cele 6 butoane existente pe interfata.

Clasa AdministratorGraphicalUserInterfaceController initializeaza un administrator,un restaurant, un view si listenerurile pentru toate butoanele de pe aceasta fereastra ( JFrame ). Listenerul pt butonul Add foloseste functii regex pentru ca sa retina datele date in textbox de la tastatura, dupa care se apeleaza metoda din Administrator de adugare de MenuItem. Listenerul pt butonul Delete foloseste functii regex pentru ca sa retina datele date in textbox de la tastatura, dupa care se apeleaza metoda din Administrator de stergere de MenuItem si toate MenuItem-urile care contin elemental sters cu ajutorul functiei de find Administrator. Listenerul pt butonul AddEdit foloseste functii regex pentru ca sa retina datele date in textbox de la tastatura, dupa care se adauga itemul din al doilea textbox in itemul de pe primul textbox. Listenerul pt butonul DeleteEdit foloseste functii regex pentru ca sa retina datele date in textbox de la tastatura, dupa care se sterge itemul din al doilea textbox in itemul de pe primul textbox. Listenerul pt butonul ShowMenuItems unde creeaza un tabel in care se pun toate MenuItem-ele din lista.

Clasa WaiterGraphicalUserInterfaceView creeaza interfata cum este aratata in sectiunea 4.3 si folosit metodele Listener pentru cele 4 butoane existente pe interfata. De asemenea sunt implementate metodele pentru toate cele 4 butoane.

Clasa WaiterGraphicalUserInterfaceController initializeaza un administrator,un restaurant, un view si listenerurile pentru toate butoanele de pe aceasta fereastra ( JFrame ). Listenerul pt butonul Add foloseste functii regex pentru ca sa retina datele date in textbox de la tastatura, dupa care se apeleaza metoda din Waiter de creare de order. Listenerul pt butonul Bill foloseste functii regex pentru ca sa retina datele date in textbox de la tastatura, dupa care se apeleaza metoda din Waiter de getBill prin care afiseaza in fisier nota de plata. Listenerul pt butonul ShowOrder unde creeaza un tabel in care se pun toate Orderurile din hash-map.

Clasa ChefGraphicalUserInterface arata ca in iterfata din sectiunea 4.3 care contine doar un JTextArea , si se utilizeaza metoda update care face update la fiecare order adaugat.



Asa trebuie sa arate bill-ul daca se comanda apa cu lamaie care este deja serializat in Administrator.

6 . Imbunatatiri, dezvoltari ulterioare si concluzii

Serializarea merge doar la Administrator, unde se salveaza serializarea, dar la Waiter nu salveaza.

Ca imbunatatiri ar putea fi o fereastra dupa apasarea a unuia dintre butoanele Administrator, Waiter, Chef unde sa se introduca o parola si de asemenea rezolvarea anumitor bug-uri existente in proiect.

7 . Bibliografie

M-am inspirat la creearea de tabel si de interfete si hash-map:

<https://stackoverflow.com/>

<https://www.geeksforgeeks.org/>

<https://docs.oracle.com/en/>