SISTEM DE PROCESARE A COMENZILOR

TEHNICI DE PROGRAMARE

UNIVERSITATEA TEHNICA CLUJ-NAPOCA

FACULTATEA DE AUTOMATICA SI CALCULATOARE

SPECIALIZAREA CALCULATOARE SI TEHNOLOGIA INFORMATIEI

Student Urcan Denisa-Teodora

Grupa 30226 CTI-Romana seria B

Cuprins

1. Obiective

1.1. Obiectiv principal

1.2. Obiectiv secundar

2.Analiza problemei

2.1. Analiza problemei

2.2. Modelarea

2.3. Scenarii

2.4. Cazuri de utilizare

3. Proiectare

3.1. Decizii de proiectare

3.2. Diagrame UML

3.3. Structuri de date

3.4. Proiectare clase

3.5. Algoritmi

3.6. Interfata utilizator

4. Implementare

5. Rezultate

6. Concluzii

7. Bibliografie

1. Obiective
   1. Obiectivul principal

Subiectul temei consta in propunerea, proiectarea si implementarea unui sistem de procesare a comenzilor in vederea determinarii produselor alese de un client. Cerintele aferente acestei teme sunt respectarea paradigmelor programarii orientate pe obiecte, structurate in pachete, clase cu numar maxim 300 linii de cod si metode cu numar maxim 30 de linii. Implementarea temei respecta conventiile de numire Java, si implicit stilul de scriere camel case.

De asemenea, se respecta proiectarea sugerata din punct de vedere al arhitecturii, si anume Layered Architecture, aplicatia continand mai multe pachete, printre care cele mai relevante fiind dataAccess, businessLayer, model, presentation.

Arhitercura Layered presupune proiectarea aplicației folosind cel puțin patru pachete de clase, și anume: **businessLayer** este pachetul care conține clasele în care se regăsește implementarea metodelor necesare pentru efectuarea operațiilor, iar pachetul **presentationLayer** conține clasele care definesc interfața grafică, dataLayer care contine procesul de salvare a datelor atat in fisier pentru crearea unei facturi, cat si serializarea ca procedeu de accesare a datelor dupa inchiderea aplicatiei.

Acest proiect simulează operațiile de adăugare, editare, ștergere și afișare pentru produsele din meniu de catre manager, vizualizarea meniului de catre client si comandarea unui produs intr-o anumita cantitate, precum si crearea comenzii si notificarea bucatarului de catre waiter. Aplicația este destinată unui lant de restaurante, cu scopul de a ține o evidență a produselor și clienților și, de asemenea, a comenzilor efectuate.

1.2 Obiective secundare

|  |  |  |
| --- | --- | --- |
| Obiectiv secundar | Descriere | Capitol |
| Folosirea structurilor de date convenabile | Structurile de date sunt alese astfel incat sa faciliteze adaugarea si extragerea clientilor. | 3 |
| Impartirea pe pachete | Fiecare pachet este format din mai multe clase care indeplinesc sarcini inrudite | 3 |
| Impartirea pe clase | Fiecare clasa se ocupa de manipularea unui singur obiect, respectiv a unei singure operatii. | 3 |
| Algoritmi utilizati | Algoritmii utilizati pentru a realiza diferitele operatii sunt cei matematici. | 3 |

1. Analiza problemei
   1. Analiza problemei

Analiza problemei presupune identificarea claselor proiectului, functionalitatile pe care acesta le implementeaza, legaturile dintre ele si modelul impus din punct de vedere al structurii. Programarea orientata pe obiecte ofera strategia de dezvoltare „Top-Down”, prin care utilizatorului ii sunt dezvaluite doar informatiile de suprafata. Aplicatia, care are ca si scop principal efectuarea operatiilor asupra clientilor si produselor este structurata in clase, fiecare detinand o functionalitate, implementate ulterior.

Datorită strategiei oferite de programarea orientate pe obiecte, se pot identifica clasele, denumite prin substantive, și funcționalitățile, denumite cu ajutorul verbelor.

Utilizarea serializarii ne permite stocarea informațiilor intr-un fisier, in vederea completarii meniului.

Operatia de adaugare a unui produs presupune introducerea datelor despre obiectele respective, respectiv nume, pret, greutate si compozitia, in cazul unui CompositeProduct (acest camp ramane necompletat pentru BaseProduct). Operatia de adaugare a unui produs in meniu se bazeaza pe completarea campurilor mentionate doar de catre manager, asemenea operatiei de editare a unui produs din meniu. Operatia de stergere se realizeaza pe baza id-ului, de asemenea realizata de manager. Clientul poate alege un produs din meniu si o anumita cantitate, urmand ca chelnerul sa realizeze comanda si sa notifice bucatarul despre comanda. Chelnerul poate de asemenea sa realizeze o factura si sa calculeze pretul total pentru comanda clientului.

2.2 Modelarea

Din punct de vedere al modelarii aplicatiei, s-au implementat diverse clase, fiecare operand asupra unui singur obiect.

Pachetul businessLayer contine clasele care tin de modelul aplicatiei. Clasa abstracta MenuItem contine informatii relevante atat pentru un CompositeProduct, cat si pentru un BaseProduct, precum nume, greutate si pret. Atat clasa CompositeProduct cat si BaseProduct extind clasa abstracta si concomitent atributele acesteia. Pentru a le diferentia, clasa CompositeProduct contine de asemena compozitia unui menuItem, camp care ramane necompletat in cazul unui baseProduct, reprezentat printr-o lista de menuItems. Clasa Order are id-ul comenzii, data realizarii comenzii, masa la care s-a realizat comanda si cantitatea ceruta de client. Acest pachet contine interfata RestaurantProcessing care defineste operatiile implementate ulterior in Restaurant, precum adaugare, editare si stergere a unui MenuItem, precum si crearea unei Comenzi. S-au folosit pre/post conditii precum si invariant al clasei si invariant pentru o metoda well-formed.

2.3 Scenarii

Utilizarea corecta a aplicatiei presupune asteptarea ca clientul sa ajunga la masa. Acesta poate vizualiza meniul. Waiterul poate realiza o comanda, calcula pretul comenzii, realiza o factura in format txt pentru comanda curenta, iar managerul poate adauga, sterge si edita un anumit item din meniu. Pentru a accesa meniul se face click pe masa de langa client, pentru a accesa managerul se face click pe imaginile deasupra bucatarului, iar pentru a accesa waiterul se face click pe accesta.

Cazul ideal de utilizare presupune procesarea corecta a comenzii de catre un client, caz care duce la notificarea bucatarului.

* Utilizatorul intra in vederea managerului, pentru a adauga, edita sau sterge produse din meniu
* Utilizatorul alege vederea meniului
* Utilizatorul intra in modul waiter, care poate adauga o comanda, crea o factura si calcula un pret al comenzii
* Programul va simula introducerea, editarea, stergerea produselor din meniu/ crearea unei comenzi, calcularea pretului, facturii, notificarea bucatarului, actualizarea meniului
* La finalul simularii vor fi generate date statistice despre comenzi: factura( prin apasarea itemului care a intrat in magazin)
* Daca se apasa butonul de back, aplicatia va fi resetata si pot fi introduse si simulate date noi de simulare.

view

Manager

view

waiter

Menu

2.4 Cazuri de utilizare

1. Utilizatorul apasa butonul de start inaintea introducerii anumitor date obligatorii pentru adaugare/editare/stergere. Astfel, va fi afisat un mesaj de eroare.

view

ERROR! BAD INPUT

1. Utilizatorul selecteaza un client si unul/ mai multe produse pentru a realiza o comanda, sau nu introduce o cantitate, acest lucru fiind de asemenea semnalat.

view

ERROR! BAD INPUT

1. Proiectarea

3.1 Decizii de proiectare

Pentru proiectarea aplicației, am folosit patru pachete principale, după structura **Layered Architecture.** Această structură presupune împărțirea claselor pe proiecte în funcție de scopul acestora și o succesiune logică între ele, astfel: Implemetarea operațiilor la nivel de cod Java, din bussinesLogic se face prin apel de metode direct, acestea reprezentand si modelul aplicatiei. Implemetările din bussinesLogic se vor apela mai apoi în pachetul destinat implementării interfeței grafice cu utilizatorul.

Pachetul BusinessLayer contine astfel 4 clase si o interfata, astfel: clasa abstracta MenuItem continand datele comune despre base si composite product, precum name, price si weight, clasa BaseProduct extinzand clasa MenuItem si avand astfel aceleasi atribute, clasa CompositeProduct extinzand de asemenea clasa abstracta, avand date suplimentare legate de compozitia unui produs din meniu, precum un ArrayList de menuItems si un sir de caractere definind compozita, interfata RestaurantProcessing care defineste operatiile implementate ulterior in Restaurant, si clasa Restaurant care reprezinta modelul propriu zis al aplicatiei. Clasa Restaurant defineste atributele unei liste de menuItems precum si un hashmap pentru a retine comenzile.

Pachetul dataLayer contine doua clase abstracte cu metode statice, respectiv write pentru scrierea facturii in fisier in clasa FileWriter si metodele serialization si deserialization pentru a serializa/deserializa in fisier.

Pachetul Presentation contine numeroase clase ale interfetei grafice precum si controllerele corespunzatoare.

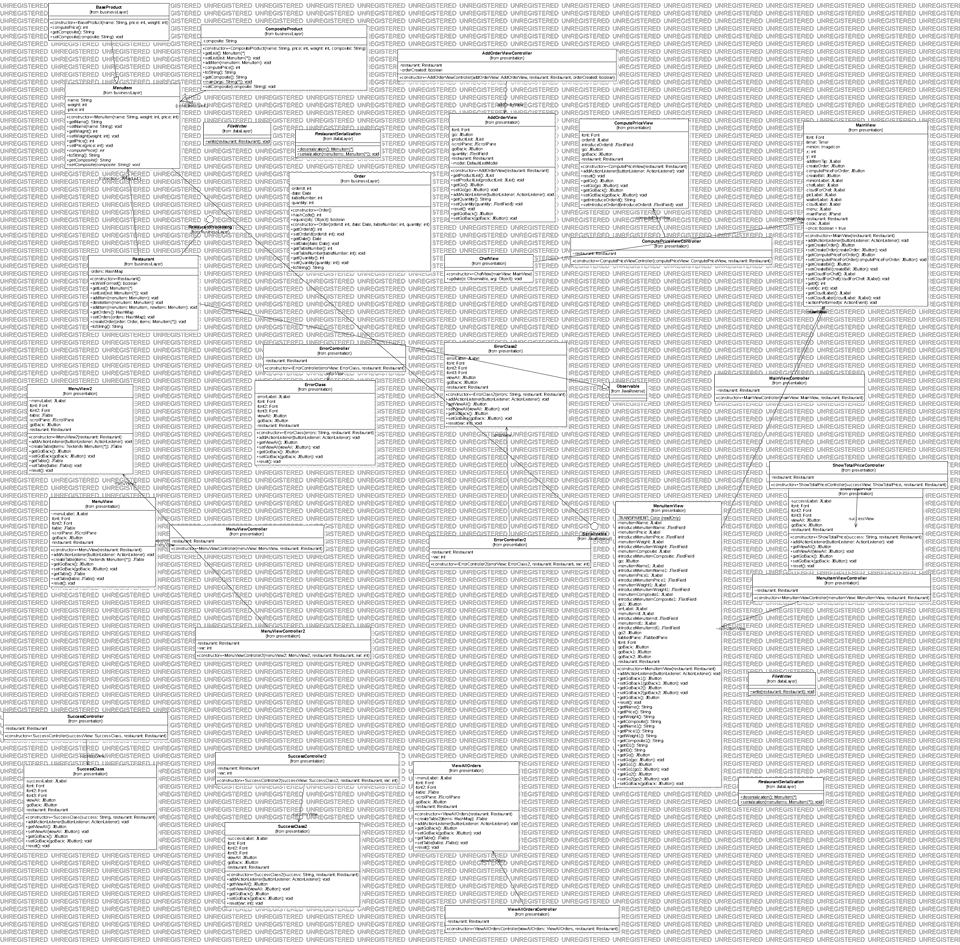
Toate pachetele si clase continute in descrierea de mai sus se bazeaza pe doua metode design pattern, respectiv cea a observerului prin care bucatarul reprezentat de ChefView este notificat de fiecare data cand apare o comanda noua, precum si cea composite prin realizarea mostenerii in clase menuItem, composite si baseproduct.

3.2. Diagrame UML

3.2.1. Diagrama de pachete



3.2.2. Diagrama de clase



3.3. Structuri de date

Pentru stocarea datelor despre produse s-a folosit o structura de tip ArrayList, datele fiind ulterior serializate. Aceste tipuri de date au lungime variabila, care se modifica odata cu operatia realizata. Totodata, datorita faptului ca adaugarea unui client se face la sfarsitul cozii, parcurgerea listei presupune un timp necesar O(n). De asemenea, comenziile sunt retinute intr-o structura de date hashmap, al carui hashcode se calculeaza, structura care detine ca si cheie o anumita comanda, iar datele propriu zise sub forma unui ArrayList de menuItems.

3.4. Proiectare clase

Proiectarea claselor imiplica folosirea unor interfete predefinite in cadrul libariilor Java, implementarea interfetei ActionListener de catre clasa Controller permite „ascultarea” si interpretarea comenzilor date de utilizator.

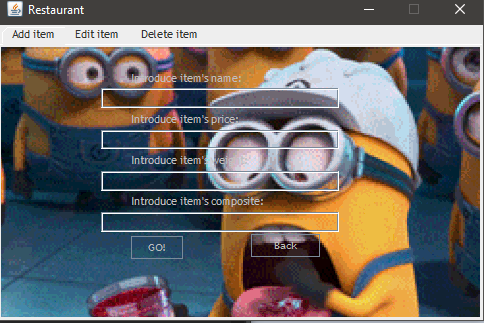
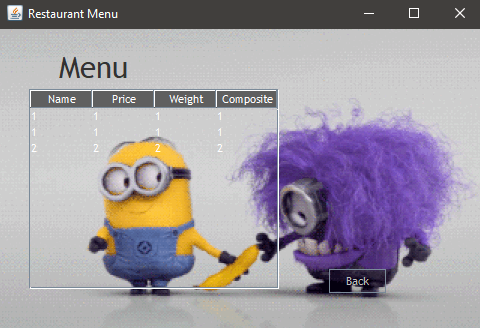
3.5. Algoritmi

Algoritmul care se remarca face referire la modul de implementare a hashcode-ului.

3.6. Interfata utilizator

Interfata simplista permite folosirea aplicatiei care simuleaza procesarea unei comenzi de catre orice utilizator, cu sau fara cunostinte in unul dintre domeniile informaticii.





1. Implementare
   1. Pachetul BusinessLayer
      1. Clasa MenuItem

Clasa abstracta menuItem contine atribute definitorii atat pentru un compositeproduct cat si pentru baseproduct, precum nume, greutate si pret, si un constructor aferent. Clasa respectiva contine metodele accesoare si mutatoare datorita privatizarii datelor din motive de confidentialitate si siguranta, afisarea prietenoasa a unui item din meniu precum si unele metode abstracte, precum computePrice, getComposite si setComposite. Prima din metodele abstracte defineste calcularea pretului pentru un produs, fie composite/base. Ultimele doua se implementeaza in clasa CompositeProduct care extinde aceasta clasa, pentru a diferentia un baseproduct de compositeproduct. Aceasta clasa abstracta implementeaza interfata Serializable, care ajuta la retinerea unei liste de menuItems intr-un fisier de unde vor fi citite ulterior si introduse in meniu, chiar si dupa inchidere aplicatiei.

* + 1. Clasa CompositeProduct

Clasa CompositeProduct extinde clasa abstracta definita mai sus, avand ca si tipuri de date suplimentare o lista de menuItems si un sir de caractere reprezentand compozitia produsului. Clasa implementeaza un constructor specific, care primeste ca argument atributele definitorii descrise mai sus si concomitent metodele abstracte. Variabilele instanta alea clasei sunt private din motive de protejare si ascundere a informatiilor legate de implementare fata de utilizator, motiv pentru care clasa contine metode accesoare si mutatoare. Aceasta clasa se ocupa si de afisarea „prietenoasa” a unui produs, pentru a transpune imaginea acestuia cat mai aproape de realitate. Toate tipurile folosite sunt tipuri de data.

* + 1. Clasa BaseProduct

Pentru o mai mare modularite, clasa BaseProduct contine doar datele definite in clasa abstracta menuItems, pentru care realizeaza metode accesoare si mutatoare. Implementeaza de asemenea si metodele abstracte, pe care insa le seteaza cu ceva nul, din considerenta ca un baseProduct nu are mai multe componente.

* + 1. Clasa Order

Clasa Order contine un id al comenzii, data realizarii comenzii, numarul mesei la care s-a realizat comanda precum si cantitatea comandata. Aceasta clasa calculeaza hashcode-ul pentru structura de date folosita ulterior, suprascriind totodata si metoda equals. Variabilele instanta alea clasei sunt private din motive de protejare si ascundere a informatiilor legate de implementare fata de utilizator, motiv pentru care clasa contine metode accesoare si mutatoare. Aceasta clasa se ocupa si de afisarea „prietenoasa” a unei comenzi, pentru a transpune imaginea acestuia cat mai aproape de realitate. Toate tipurile folosite sunt tipuri de data

* 1. Pachetul dataLayer
     1. Clasa FileWriter

Clasa abstracta FileWriter contine o singura metoda statica, care se ocupa de calcularea unui pret total si scrierea in fisier a unei facturi, reprezentata de comanda curenta.

* + 1. Clasa RestaurantSerialization

Clasa abstracta RestaurantSerialization continde doua metode statice, care se ocupa de serializarea si respectiv deserializarea datelor, in vederea alcatuirii unui meniu dupa inchiderea aplicatiei. Acest procedeu se refera la scrierea datelor comandate intr-un fisier si deserializarea in momentul deschiderii aplicatiei pentru a forma un meniu cu datele introduse anterior. Metoda de deserializare realizeaza citirea obiectelor din fisier, returnand un ArrayList de menuItems, in timp ce metoda de serializare scrie in fisier un ArrayList primit ca argument.

1. Rezultate

Verificarea flow-ului clientilor este realizata in cadrul interfetei grafice, intr-un scrollpanel care permite vizualizarea unui numar mare de clienti/produse/comenzi.

1. Concluzii

Acest proiect m-a ajutat la intelegerea diferitelor concepte ale programarii orientate pe obiecare, care au fost aprofundate pe parcursul dezvoltarii aplicatiei. Structurarea in pachete si intelegerea nevoi de a structura astfel, crearea unei clase care se ocupa de manipularea unui singur obiect si restrangerea metodelor de rezolvare in mai putin de 30 de linii de cod, respectiv evitarea structurilor imbricate la sugestia laborantului au dus in final la descoperirea unor metode de rezolvare mai simpliste, care pot fi urmarite mai usor.

1. Bibliografie

* <https://stackoverflow.com/>
* <http://users.utcluj.ro/~igiosan/>
* <https://www.geeksforgeeks.org/composite-design-pattern/?fbclid=IwAR0aX8id6Baam9EwqlUez9BEiwo6BjB4HCWim_Q5NkxVIuZ-qJuf3_4wOPY>
* l<https://dzone.com/articles/composite-design-pattern-in-java-1?fbclid=IwAR2zCtr2FIZILcaVexJJfFxZJqRBfVuBdMMSrAYmRC7odJfjokFFKzcYCCk>
* <https://dzone.com/articles/composite-design-pattern-in-java-1?fbclid=IwAR2zCtr2FIZILcaVexJJfFxZJqRBfVuBdMMSrAYmRC7odJfjokFFKzcYCCk>
* <https://dzone.com/articles/design-patterns-uncovered?fbclid=IwAR0RdVr-Y58YEaLWehA9Wr3INskyOyFsnDGoBqCOAXvIToMuizZuCY565IQ>
* <https://medium.com/@priya104/the-composite-pattern-2edec432dd58?fbclid=IwAR3si9sLmqR2gsVrgkXumoORW39AKrUm7i0o7ULpS8roSpZ5mRtxbQgP_48>
* <https://refactoring.guru/design-patterns/observer/java/example?fbclid=IwAR3cU4mgRwMqvaO6bGMmSeyPi5xE3IGlRMVmZKaq5NGdWlgsWl8qjOdtYdk>

