TEMA 4

Food Delivery Management System

Vasilachi Eugen

Grupa 30227

Cuprins

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

1. Obiective

Proiectati si implementati o aplicatie ce are ca scop gestionarea unui sistem de livrare a comenzilor pentru o firma de catering, avand ca date clientii si produsele ce se gasesc in depozit, administratorul si un angajat.

Obiectivul principal: Realizarea unei aplicatii care proceseaza comenzi pentru un sistem de livrarea a mancarii la o companie de catering utilizand un fisier „products.csv” pentru a memora produsele si un fisier „file.ser” pentru a pastra informatiile si a le incarca din nou in aplicatie in momentul rularii ei. Aplicatia va functiona pe baza unei interfete care va executa un comportament particular specific actiunii alese. O alta functie importanta a aplicatiei este de a genera documente txt in care sunt salvate diferite informatii in functie de comenzile executate.

Obiective secundare:

1. Analiza problemei si identificarea cerintelor

* Alegerea structurilor de date si proiectarea claselor dorite

1. Proiectarea aplicatiei de gestionarea a comenzilor

* Realizarea diagramei UML
* Descrierea algoritmilor folositi in realizarea simularii

1. Implementarea intregei aplicatii

* folosirea Business– Presentation – Data Layes
* descrierea fiecarei clase impreuna cu metodele si campurile importante
* descrierea interfetei utilizatorului

1. Testarea simulatorului

* Prezentarea rezultatelor ce se obtin in urma rularii cu anumite date de intrare

1. Concluzii

* Prezentarea posibilitatilor pentru a dezvolta aplicatia ulterior

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

Scenariu principal de utilizare:

1. Utilizatorul alege modului unde doreste sa realizeze operatii ( Clienti, Produse sau Comenzi)
2. Se apasa butonul corespunzator operatiei dorite
3. Afisarea tabelelor modificate / crearea bonului si actualizarea istoricului de comenzi

Detalierea scenariului de utilizare:

Pentru a putea rula si obtine rezultatele in urma rularii aplicatiei trebuie mai intai sa ne alegem modulul in care dorim sa efectuam operatii: Client – adaugarea, editarea, stergerea unui client sau afisarea tabelului cu clienti; Product - adaugarea, editarea, stergerea unui produs sau afisarea tabelului cu produse; Order – adaugarea unui produs in comanda si realizarea comenzii finale. Dupa ce trimitem o comanda, se va crea un fisier text cu bonul respectiv si de asemenea, comanda va fi adaugata in fisierul text cu istoricul comenzilor.

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

Pentru a putea implementa aplicatia noastra, mai intai trebuie sa intelegem conceptul de serializare si lambda expression.

Serializare reprezinta transformarea unui obiect intr-o secventa de octeti din care sa poata fi refacuta ulterior obiectul original. Procesul invers, de citire a unui obiect serializat pentru a-i reface starea originala, se numeste deserializare.

Lambda expresiile ne permit să creăm instanţe ale claselor cu o singură metodă/ ale interfetelor functionale într-un mod mult mai compact. O  lambda expresie constă:

* dintr-o listă de parametri formali, separaţi prin virgulă şi cuprinşi eventual între paranteze rotunde,
* săgeata direcţională ->,
* un body ce constă dintr-o expresie sau un bloc de instrucţiuni.

Scenariu principal de utilizare:

1. Utilizatorul se logheaza cu credentiale sale. Daca nu are cont, se poate inregistra doar pe post de client.
2. Se deschide fereastra specifica utilizatorului (daca e client se deschide fereastra corespunzatoare lui, daca e administrator se va deschide fereastra in care poate modifica produsele si genera rapoarte)
3. Crearea bonului/raportului/ modificarea fisierului cu produsele initiale

Detalierea scenariului de utilizare:

Pentru a putea rula si obtine rezultate in urma rularii aplicatiei trebuie mai intai sa ne logam pentru a putea accesa operatiile specifice noua: Client –cautarea unui produs in functie de o anumita categorie, adaugarea unui produs intr-o comanda si plasarea comenzii; Administrator – adaugarea, editarea, stergerea de produse de baza, crearea unui meniu nou compus din mai multe produse de baza si generarea de rapoarte. Dupa ce trimitem o comanda, se va genera un fisier text cu bonul respectiv. De asemenea, cand generam un raport, acesta va fi scris intr-un fisier text, in functie de raportul ales.

3. Proiectare

In implementarea acestui proiect, am ales sa folosesc un model ce cuprinde urmatoarele pachete cu clasele aferente: business (User, Order, MenuItem, BaseProduct, CompositeProduct, DeliveryService, interfata IDeliveryService si ConnectionFactory), dataLayer (FileWriterClass si Serializator), presentation (ClientView, AdministratorView, LogInView, ReportsView si EmployeeView).

* Pachetul business:
* Clasa User: lucreaza cu numele, parola si tipul specifice fiecarui user (0-client, 1- administrator)
* Clasa Order: lucreaza cu id-ul comenzii, numele clientului, data in care a fost data si suma specifice fiecarei comenzi
* Clasa MenuItem : lucreaza cu numele, rating-ul, caloriile, proteinele, sodiul, grasimile si pretul fiecarui produs
* Clasa BaseProduct: extinde MenuItem si lucreaza cu cu aceleasi atribute precum MenuItem
* Clasa CompositeProduct: extinde MenuItem si lucreaza cu cu aceleasi atribute precum MenuItem, in plus are un HashSet ce contine produsele care alcatuiesc noul produs si compositePrice() care aplica o reducere a pretului final
* Clasa ConnectionFactory: implementeaza metodele prin care se face legatura cu baza de date pentru stocarea utilizatorilor
* Clasa DeliveryService: implementeaza toate metodele din interfata IDeliveryService, adica toate functionalitatile aplicatiei
* Interfata IDeliveryServiceProcessing: antelele metodelor functionalitatilor aplicatiei
* Pachetul dataLayer:
* Clasa FileWriterClass: implementeaza metodele de afisare a bonului si istoricului comenzilor intr-un fisier text specific
* Clasa Serializator: implementeaza metodele de serializare si deserializare a datelro
* Pachetul presentation:
* Clasa LogInView: imaginea filei de logare, unde se poate si inregistra un nou client
* Clasa ClientView: imaginea filei Client unde utilizatorul foloseste aplicatia pentru adauga produse intr-o comanda, a plasa comanda si a cauta produse in functie de o anumita categorie
* Clasa AdministratorView: imaginea filei Administrator unde utilizatorul foloseste aplicatia pentru a edita, adauga, sterge produse sau pentru a crea un CompositeProduct(meniu) pe care ulterior sa il adauge in lista de produse initiala
* Clasa ReportsView: imaginea filei Reports unde utilizatorul foloseste aplicatia pentru a genera rapoarte legate de clienti, produse, comenzi in functie de ore, zile, totalul comenzii, etc.
* Clasa EmployeeView: fereastra in care un angajat va fi anuntat la plasarea unei comenzi

Diagram

Description automatically generated

4.Implementare

In continuare se vor descrie amanuntit clasele insotite de cele mai importante metode.

* Clasa User

Aceasta clasa contine datele care mapeaza tabelul „user”. Contine ca field-uri toate coloanele din tabel, numele acestora fiind identice cu cu numele coloanelor din tabel pentru a se putea folosi thenica reflectiei in crearea obiectelor de tipul respectiv in urma unei interogari. Ca si metode existente in aceasta clasa sunt doar metoda de tipul setter si getter pentru a accesa variabilele de clasa. De asemenea, contine si metoda toString pentru a putea afisa informatiile despre un client.

* Clasa Order

Aceasta clasa contine datele ce caracterizeaza o comanda is anume: id, clientName, date si totalul comenzii. Ca si metode existente in aceasta clasa sunt doar metoda de tipul setter si getter pentru a accesa variabilele de clasa. De asemenea, contine si metoda toString pentru a putea afisa informatiile despre o comanda.

* Clasa MenuItem

Aceasta clasa contine datele despre un produs si anume: product, rating, calories, protein, sodiu, fat si price. Ca si metode existente in aceasta clasa sunt doar metoda de tipul setter si getter pentru a accesa variabilele de clasa. De asemenea, contine si metoda toString pentru a putea afisa informatiile despre un client. Folosind Composite Design Pattern, aceasta clasa va fi mostenita de catre BaseProduct si CompositeProduct.

* Clasa BaseProduct

Mostenind clasa MenuItem, este caracterizata de aceleasti atribute.

* Clasa CompositeProduct

Mostenind clasa MenuItem, este caracterizata de aceleasti atribute, in plus avand un HashSet in care pastram produsele ce compun un astfel de produs.Ca si metode, le vom rescrie pe cele din clasa MenuItem, particularizandu-le ( exememplu: la calorii, proteine, grasimi, sodiu – le vom adauga intr-o variabila valorile specifice fiecarui produs, la rating vom realiza media aritmetica, iar la pret, pentru fiecare produs adaugat in comanda, din suma produselor vom aplica o reducere de 10%)

* Clasa Connection

Aceasta clasa are scopul de a returna o conexiune la baza de date si de a inchide unele componente folosite pentru a realiza o interogare. Clasa ConnectionFactory contine urmatoarele metode: createConnection (aceasta metoda este un privata, creaza o conexiune care mai apoi este returnata), getConnection (returneaza o conexiune la baza de date obtinuta prin apelarea metodei prezentate precedent). Aceasta clasa contine si un constructor care verifica existenta driver-ului in conectorul mysql. De asemenea, contine si metodele: close(Connection), close(Statement) si close(ResultSet), 3 metode supraincarcate care au rolul de a inchide obiectul primit ca argument.

* Interfata IDeliveryService

Aceasta interfata contine toate antetele metodelor ce vor fi implemetate ulterior in clasa DeliveryService.

* Clasa DeliveryService

Aceasta clasa implementeaza interfata IDeliveryService, deci vom implementa toate metodele definite aici. Variabilele de clasa folosite sunt: meniu (lista produselor), produseComandate (HashMap in care cheia este comanda, iar valoarea e un HashSet de MenuItem) si comenzi ( lista comenzilor).

Metode implementate:

* menu – punem in lista de produse inregistrarile din fisierul „products.csv”
* refreshCSV – rescriem in CSV noile informatii. Aceasta metoda ne ajuta sa pastram in fisier datele mereu actualizate, astfel incat tabelul dupa o anumita operatie se va actualiza imediat
* deleteProduct – metoda utilizata de catre administrator cand doreste sa stearga un produs
* addProduct - metoda utilizata de catre administrator cand doreste sa adauge un produs
* editProduct - metoda utilizata de catre administrator cand doreste sa editeze un produs
* addMenu – metoda utilizata de catre administrator cand creaza un nou CompositeProduct si il adauga in lista de produse
* order – metoda utilizata de client in momentul in care transmite o comnda. In acel moment, se va adauga o comanda la lista de compenzi si produsele din comanda la lista produselor comandate
* findProducts- foloseste lambda expression si stream-uri. Are drept parametrii categoria (in functie de ce parametru cautam) si detaliu (poate fi string sau int, in functie daca cautam dupa nume sa dupa restul atributelor). De exemplu, pentru nume, lambda expression, folosind stream.filter vor gasi elementele ce contin in numele lor cuvantul introdus de noi. La final, vom pune „.collect(Collectors.toList())” pentru a le introduce intr-o lista pe care ulterior o vom afisa in JTable
* generateInterval – are ca parametrii ora de inceput si cea de sfarsit. Se foloseste in aceeasi maniera ca la metoda gaseste produs lambda expression, utilizand un filtru pentru a vedea daca ora comenzii este in intervalul dat de noi.
* productTimes – parcurgem o data comenzile pentru a introduce produsele intr-un ArrayList pe care ulterior, utilizand lambda expression vom numara numarul de aparitii (functioneaza ca un vector de frecventa). Daca numarul de aparitii este mai mare decat atributul dat de noi in antet, vom printa acel produs.
* clientsReport - filtram o data comenzile pentru a le lua doar pe cele cu pretul de peste suma introdusa de noi, urmand sa numaram comenzile date de fiecare client, iar in cazul de e mai mare decat numarul dat de noi, vom afisa in raport numele clientului cu numarul de comenzi
* productOrderedWithinADayReport - filtram comenzile penrtu a le obtine pe cele comandate intr-o anumita zi, urmand sa tinem intr-un arrayList produsele pentru a le putea numara ulterior utilizand lambda expression, urmand sa le afisam.
* Clasa FileWriterClass

Aceasta clasa ne ajuta sa printam rezultatele obtinute in fisierele text dorite, dat ca parametru in functia writeToFile(String fisier). Totodata, se va crea si fisierul „Bon comanda (idComenzii) (numele Clientului).txt” in care va fi printat bonul comenzii date de catre un anumit client.

* Clasa Serializator

Aceasta metoda contine metodele serialize() si deserialize() cu ajutorul carora luam informatiile stocate anterior in fisierul „file.ser” pentru a nu pierde informatiile in momentul in care inchidem aplicatia. Serializarea o realizam dupa plasarea unei comenzi, iar deserializarea o facem in momentul in care deschidem aplicatia.

* Clasa LogInView

Aceasta extinde JFrame. Aici sunt definite ca atribute private fiecare buton de care avem nevoie si fereastra propriu-zisa. Cu ajutorul conexiunii cu baza de date, verificam daca este client sau administrator si in momentul in care apasam LogIn ne va duce pe fereastra destinata noua. Daca nu suntem client inca, vom apasa pe Sign Up, urmand sa fim introdusi in sistem drept clienti, urmand apoi sa intram pe fila noastra.

Graphical user interface

Description automatically generated

* Clasa AdministratorView

Aceasta extinde JFrame. Aici sunt definite ca atribute private fiecare buton de care avem nevoie, tabelul in care vor fi afisate datele, filed-urile in care scriem informatii despre un anumit produs, fereastra propriu-zisa, variabila in care calculam totalul unui nou meniu. In functie de butonul accesat, tabelul se va actualiza in acelasi timp. Fiecare buton este utilizabil cu ajutorul apelarii metodei „addActionListener”, in care vom implemnta metoda propriu-zisa pentru butonul respectiv.

Table

Description automatically generated

* Clasa ReportsView

Aceasta extinde JFrame. Aici sunt definite ca atribute private fiecare buton de care avem nevoie, tabelul in care vor fi afisate datele, filed-urile in care scriem informatii despre un anumit raport, fereastra propriu-zisa. In functie de butonul accesat, se va genera raportul respectiv. Fiecare buton este utilizabil cu ajutorul apelarii metodei „addActionListener”, in care vom implemnta metoda propriu-zisa pentru butonul respectiv.

Graphical user interface

Description automatically generated

* Clasa ClientView

Aceasta extinde JFrame. Aici sunt definite ca atribute private fiecare buton de care avem nevoie, tabelul in care vor fi afisate datele, filed-urile in care scriem informatii despre un anumit produs ce vrem sa il cautam, field-ul in care se screi informatia despre produsul selectat din tabel, fereastra propriu-zisa si comboBox-ul ce ne ajuta in cautarea unui produs in functei de ceva anume. In functie de butonul accesat, se va genera raportul respectiv. Fiecare buton este utilizabil cu ajutorul apelarii metodei „addActionListener”, in care vom implemnta metoda propriu-zisa pentru butonul respectiv.

Graphical user interface, table

Description automatically generated

* Clasa EmployeeView

Aceasta extinde JFrame, insa implementeaza Observable prin care la plasarea unei comenzi de catre un client se afiseaza comanda impreuna cu produsele care vor trebui pregatite. Pentru acest lucru, in metoda comanda, din DeliveryService folosim metoda setChange(0 pentru a introduce schimbarile, urmand sa notificam observer-ul prin metoda notifyObserver().

Graphical user interface, application

Description automatically generated

5.Rezultate

In urma rularii aplicatiei, se obtin fisierele corespunzatoare comenzilor date de fiecare client, cele 4 fisiere corespunzatoare cerintei administratorului si de asemenea cel in care pastram toate informatiile din sistem pentru a nu le pierde la inchiderea aplicatiei. Astfel, datele pe care le introducem, modificam, stergem in fila administratorului, vor fi actualizate in fisierul „products.csv”, iar cele care caracterizeaza clasa „DeliveryService”, vor fi serializate in fisierul „file.txt” (nu putem intelege acest fisier deoarece datele sunt scris codat).

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

7.Bibliografie

* <https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>
* <https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/>
* <https://www.youtube.com/watch?v=7lHv_Dh2Hz4>