DOCUMENTATIE

TEMA *4*

FOOD DELIVERY MANAGEMENT SYSTEM

NUME STUDENT: CIU ADRIAN-VALENTIN
GRUPA: 30227

CUPRINS

1. Obiectivul temei

Obiectivul principal al temei a fost implementare unei aplicatii ce reuseste sa managerieze o companie de livrat mancare cu clienti, administratori si angajati, rezultate putand fi vazute in timp real in interfata grafica.

Obiectivele secundare sunt reprezentate de:

- Analiza problemei, modelare, scenarii, cazuri de utilizare → Se prezinta cadrul de cerinte functionale formalizat si cazurile, cazurile de utilizare → prezentat in capitolul 2
- Proiectare → Vor fi prezentate modul in care fost proiectata aplicatia in POO (MVC), diagrama UML de clase si pachete, structurile de date folsite, interfetele definite si algoritmii folositi → prezentat in capitolul 3
- Implemantare → Descrierea fiecarei clase cu campuri si metode→ prezentat in capitolul 4
- Rezultate → Prezenatarea scenariilor de testare cu Junit→ prezentat in capitolul 5

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

a. Analiza problemei

Cerinte functionale:
-Aplicatia poate sa inregistreze un nou administrator;
-Aplicatia poate sa inregistreze un nou client;
-Aplicatia poate sa inregistreze un nou angajat;
-Aplicatia lasa sa se logheze un client;
-Aplicatia lasa sa se logheze un administrator;
-Aplicatia lasa sa se logheze un angajat;
-Aplicatia il lasa pe administrator se se ocupe de lista de menuiri ce pot si cumparate de catre client;
-Aplicatia lasa administratorul se genereze rapoarte pentru a vedea cat de bine functioneaza afacerea sa;
-Aplicatia ii arata unui angajat atunci cand o noua comanda a fost plasata;
-Aplicatia il lasa pe un client sa isi faca o lista de meniuri pe care doreste sa le achizitioneze;

b. Modelare

Pentru a stoca toate datele despre utilizatori si meniuri am folosit serializarea datelor. Astfel modificarile se puteau vedea in timp real, iar informatia nu a fost pierduta la inchiderea aplicatiei. De asemenea, pentru a putea notifica in timp real un anagajat ca o comanda noua a fost plasata de un client am folosit Observer Design Pattern. Am folosit in aceasta tema si Design by Contract.

c. Scenarii

In aceasta aplicatie exista cel putin trei scenario de utilizare. Atat un administrator, un client, cat si un angajat se pot afla logati in acelasi timp in aplicatie. In cazul in care sunt conectati doar angajati la plicatie, acestia nu vor primi nici o notificare de comnada deoarece acest lucru este posibil doar atunci cand sunt conectati si clientii ce pot efectua comenzi.

Scenariu de utilizare:

Administratorul selecteaza produselor din care doreste sa creeze un nou produs pentru meniu.

Administratorul apasa pe butonul ce creeaza produsul (dupa ce a selectat mai multe meniuri) si aplicatia salveaza noul produs

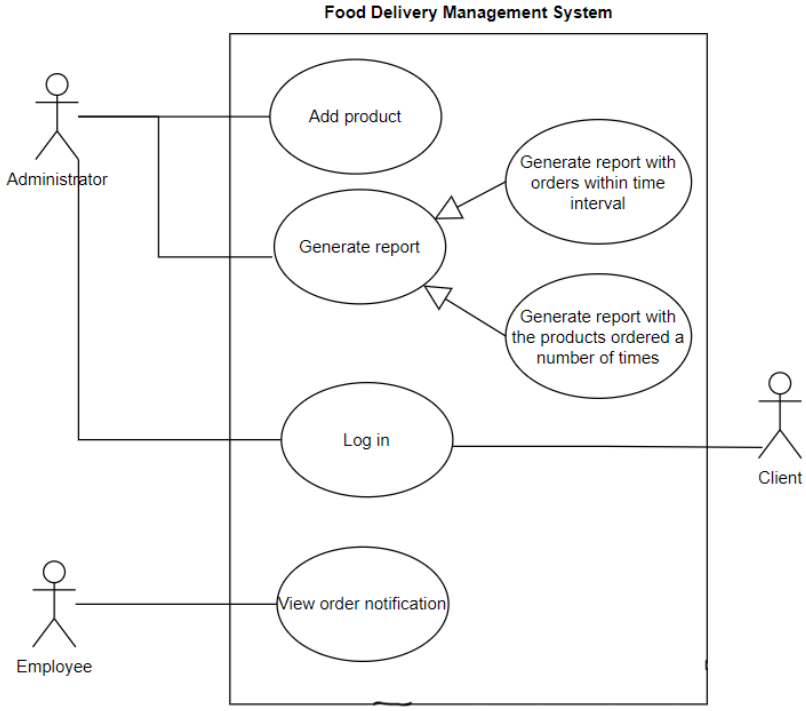
Scenariu alternativ:

Administratorul introduce valori negative.

Aplicatia afiseaza un mesaj de eroare si ii cere administratorul sa intorduca datele corecte.

Aplicatia face acesti pasi pana cand administratorul introduce datele corespunzatoare.

d. Cazuri de utilizare



3. Proiectare

a. Proiectarea aplicatiei

Aplicatia a fost dezvoltata folosind abordarea MVC(Model View Controller). Astfel, partea de model s-a ocupat de partea de manageriere administratorilor, clientilor si angajatilor. Controller-ul s-a ocupat de implementarea functionalitatii tuturor butoanelor. In View a fost implementata interfata grafica in care se poate vedea continutul corespunzator in functie de ce user s-a logat la momentul respectiv. Pentru a impementat functionalitatea calsei DeliveryService, s-a implementata interfata IDeleveryServiceProcessing

b. Diagrama UML → Unified Modeling Language

c. Structuri de date

In cazul acestei aplicatii nuna dintre cele mai folosite strcuri de date a fost reprezentata de cartre List. Astfel, pentru a stoca angajatii, adminitratorii si clientii s-a folosit structura de date LinkedList. Pentru a stoca toate menurile s-a utilizat structura de date ArrayList. Pentru a stoca datele despre o clientul si comanda cu itemele din menu ice au fost puse in "cosul de cumparaturi" s-a autilizat structura de date HashMap, in care in functie de comanda se stocau produsele comandate.

d. Interfete folosite

In aceasta tema, pentru a utiliza in mod corespunzator Observer Design Pattern s-a utilizat interfata Observer pentru a putea updata in timp real daca un angajat trebuie sa pregateasca o noua comanada pentru client. Pe langa decalararea metodelor ce trebuie implementate, s-a utilizat si Design by Contract si s-au declarant in aceasta interfata pre si post conditiile.

e. Mostenirea

Pentru a implementat in mod corespunzator Composite Design Pattern, s-a utilizat mostenirea clasei MenuItem de catre BaseProduct si CompositeProduct. Astfel, a fost posibila sa se actualizeze datele produselor compuse si totodata sa poata fi puse toate itemele din meniu in aceeasi structura de date.
Pentru a implementa in mod corespunzator Observer Design Pattern, clase DeliveryService a mostenit clasa Observable pentru a implemnta metodele ce adauga un nou observer si ce trimit la fiecare observer datele actualizate.

f. Pachete utilizate

Avand in vedere ca aceasta este una dintre cele mai complexe aplicatii implentate pana in momentul de fata, a fost nevoie de o organizare corespunzatoare in mai multe pachete. Astfel, aplicatia a fost organizata in urmatoarele pachete:

- BLL → Acest pachet are rolul de ase acupa de logica din spatele logarii cleintilor, administratorilor, angajatilor, dar si efectuarii unei comenzi noi, stocarea datelor prin serializare pentru a nu fi pierdute odata cu inchiderea aplicateie, dar si anuntarea unui angajat daca acesta primeste o comanada noua.
- Data Access → Aceast pachet are drept scop implementarea claselor ce se ocupa cu citirea meniurilor de produse de baza din fisierul csv (ce este folosit de catre administrator sa initializeze datele pentru meniuri), cu serializarea si deserializarea datelor si a generarii de rapoarte sau de chitante pentru client.
- Presentation → Aceasta clasa este utilizata penteru a implemnata clasele ce se ocupa de interfata grafica si de controlul butoanelor si a evenimentelor la apasarea acestora
-
    - Acest pachet continue si pachetul **JModel** ce se ocupa cu implementarea claselor ce se ocupa cu generarea si popularea cu date JTabel-urile utilizate in interfata grafica.
- **Start** → Acest pachet se ocupa cu pornirea aplicatiei

4. Implementare

In fiecare pachet se afla cel putin o clasa:
In pachetul **BLL** am implemnatat urmatoarele calse:
-**BaseProduct** → Aceasta clasa extinde clase MenuItem si are drept scop creearea unui nou produs de baza ce ce va fi adaugat ulterior in lista cu toate produsele disponibile in meniu
-**CompositeProduct** → Aceasta clasa extinde clase MenuItem si are drept scop creearea unui nou produs compus format din mai multe produse de baza, ce ce va fi adaugat ulterior in lista cu toate produsele disponibile in meniu
-**MenuItem** → Aceasta reprezinta clasa de baza ce continue datele despre un produs din lista de produse disponibile in meniul aplicatiei

-**DeliveryService** → Aceasta clasa are drept scop implementarea tutror metodelor ce se ocupa de managerierea comenzilor pe care le face un administrator, un client sau un angajat al companieie. Astfel, aici sunt implementate metodele ce pot adauga sau loga un client, un anagajat sau un client. De asemenea, au fost implmentate metodele ce se ocupa de adaugarea unui nou produs in meniu sau de plasarea unei noi comenzi ce poate sa contina unul sau mai multe produse, in functie de preferintele clientului. Tot aceasta clasa face parte din Observer Design Pattern, deci comanda pe care trebuie sa on execute un angajat este actualizata din aceasta clasa. In aceasta clasa sunt implmentatae prin streamuri metodelor celor patru rapoarte spceificate in cerintele temei. De asemenea, sunt implementate si metodele de serializare si deserializare folosite strict in fucntie de necesitatile acestei aplicatii. In aceasta clasa este implementata Design by Contract.
-**Observable** → Aceasta clasa are doar scopul de a fi mostenita (de catre DeliveryService) pentru a implemnta metodel ce se ocupa cu adaugarea si notificare observarilor din aplicatie (este folosita pentru a implementa Observer Design Pattern).
-**Order** → Aceasta clasa are drep scop de inregistra o noua comanda in functie de id-ul clientului si id-ul comenzii curente. Aceasta are o importanta sporita in aceasta aplicatie deoarece datele din ea sunt folosite ca key in HashMap-ul ce continue meniurile comandate de catre un client la un moment dat.
-**User** → Aceasta clasa are drept scop de a stoca date despre un user, fie ca este el administrator, client sau angajat. Aceasta clasa continue username-ul, parola si id-ul unic pentru a nu putea avea mai multi utilizatori cu acealsi id. Id-ul este de asemenea necesar pentru a putea sa-l tarnsmite in constructorul calsei Order atunci cand o noua comanda este data de catre un client.

In pachetul **DataAccess** am implemnatat urmatoarele calse:
-**CSVReader** → Aceasta clasa are drept implementarea metodelor ce se ocupa cu extragrea datelor din fisierul csv.
-**FileWriter** → Aceasta clasa are drept scop implementarea metodelor ce se ocupa cu metodelor utilizate pentru a creea un raport sau o chitanta in format PDF.
-**Serializator** → Aceasta clasa are dreprt scop implementarea metodelor ce se ocupa serializarea si deserializarea datelor utilizate in aceasta aplicatie.

In pachetul **Presentation** am implemnatat urmatoarele calse:
-AdministratorGUI → Aceasta clasa are drept scop managerierea produselor de catre un administrator. Astfel, in aceasta clasa un administrator poate sa importe produsele prezente in fisierul csv; poate sa modice in produs in functie de ce considera el; poate sa selecteze mai mule produse de baza si faca un pordus compus ce va fi adugat in Jtabel-ul de contene toate meniurile.
-ClientGUI → Aceasta interfata are drept scop sa-i afiseze unui client intr-un JTable toate produsele disponibile in meniu. De asemenea aceasta poate sa produse oricate produse doreste si sa efectueze comanda pe care o doreste.
-ClientOrderGUI → Aceasta clasa are drept scop sa afiseze unui client produsele din meniu pe care acesta le-a selectat pentru a le comanda. Tot aici este present si butonul pentru a realiza aceasta comanda.
-ComposeProductGUI → Aceasta clasa are drept scop sa afiseze produsele de baza pe care un administrator le-a aselectat pentru a creea un nou produs pentru meniu. Tot aici este present si butonul pentru a realiza aceasta comanda.

- -Controller → Aceasta clasa are drept scop implemantarea functionalitatea tutroor butoanelor din interfata grafica
    - Pachetul JModel are drept scop implementarea urmatoarelor clase:
        - JModelProducts → Aceasta clasa are drept scop crearea prin reflexie al modelului unui JTable prin metoda mode() si populatea cu date (tot prin reflexive) prin metoda modelData().

-EmployeeGUI → Aceasta clasa are drept scop sa-in afiseze unui anagajat ce menuiuri are de pregatit pentru un client
-LogInGUI → Aceasta clasa are drept scop introducerea datelor de logare a unui utilizator sau a datele cu care un utilizator doreste sa se inregistreze in aplicatie.
-Report1GUI → Aceasta clase se ocupa cu crearea interfetei in care trebuie introduce datele in functie de catre raportul 1 trebuie generat.
-Report2GUI → Aceasta clase se ocupa cu crearea interfetei in care trebuie introduce datele in functie de catre raportul 2 trebuie generat.

-Report3GUI → Aceasta clase se ocupa cu crearea interfetei in care trebuie introduce datele in functie de catre raportul 3 trebuie generat.
-Report4GUI → Aceasta clase se ocupa cu crearea interfetei in care trebuie introduce datele in functie de catre raportul 4 trebuie generat.
-SelectLogInGUI → Aceasta clasa are drept scop selectarea de catre un utilizator ca ce fel de tip de utilizator dereste sa se logheze.

5. Rezultate

Rezultataele pot pe care un utilizator (administrator, client sau angajat) pot fi vizualizate in timp real in interfata grafica, dar sunt de asemenea salvate in timp real prin procedeul de serializare. Astfel, se poate observa ca aplicatia functioneaza in mod corespunzator conform cerintelor.

6. Concluzii

Aceasta tema a reprezentat una dintre cele mai complexe si complicate proiecte la care afi um lucrat in aceasta mement deoarece a trebuit sa respect o structura riguroase pentru a implementa design pattren-urile cerute. De asemenea, generarea rapoartelor a fost destul de complexa deoarece au trebiuit sa fie implementate prin streamuri, lucru ce a fost util deoarece am scris main putine linii de cod si am si invatat si acest procedeu de a scrie cod in Java.

Ca dezvoltari ulterioare, aceasta aplicatia ar putea chiar sa fie utilizata intr-un restaurant, acest lucru fiind posbil daca ar fin utilizata o baza de date, iar aplicatia ar fi inacaracate in online pentru a putea sa fie accesbila remote. De asemenea, mai multe functionalitati si moduri de realizare a noi meniuri compuse ar putea sa fie implemnatate.

7. Bibliografie

http://javahungry.blogspot.com/2013/08/hashing-how-hash-map-works-in-java-or.html

https://howtodoinjava.com/java/collections/hashmap/how-hashmap-works-in-java/

http://javahungry.blogspot.com/2014/03/hashmap-vs-hashtable-difference-with-example-java-interview-questions.html

http://javahungry.blogspot.com/2013/08/how-sets-are-implemented-internally-in.html

https://www.baeldung.com/java-serialization

https://www.geeksforgeeks.org/serialization-in-java/

https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html

https://howtodoinjava.com/java8/java-stream-distinct-examples/

dhttps://howtodoinjava.com/java8/java-stream-distinct-examples/

http://stackoverflow.com/questions/11415160/how-to-enable-the-java-keyword-assert-in-eclipse-program-wise