Facultatea de Automatică și Calculatoare

Departamentul Calculatoare

TEMA 4

la disciplina

TEHNICI DE PROGRAMARE

- MANGEMENTUL UNUI SISTEM DE LIVRARI MANCARE-



Proiect realizat de : Moloce Sabina-Maria

An: 2

Grupa: 30225

ANUL ȘCOLAR 2020-2021

Cuprins

1. Obiectivul temei..................................................................................................................................pag 3
2. Analiza problemei, modelare, scenarii, cazuri de...............................................................................pag 4
3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator) ............................................................................................pag 6
4. Implementare .....................................................................................................................................pag 9
5. Rezultate ............................................................................................................................................pag 15
6. Concluzii ............................................................................................................................................pag 15
7. Bibliografie.........................................................................................................................................pag 15

# 1. Obiectivul temei

1.1 Obiectivul principal

Obiectivul acestei teme de laborator este să proiectăm și să implementăm un sistem de management al livrării alimentelor pentru o companie de catering. Clientul poate comanda produse din meniul companiei. Sistemul ar trebui să aibă trei tipuri de utilizatori care se conectează folosind un nume de utilizator și o parolă: administrator, angajat obișnuit și client.

1.2 Obiectivele secundare

Dezvoltarea de use case-uri – capitolul doi

Use case-urile reprezintă un set de scenarii legate de modul în care este utilizat sistemul. Daca le utilizam putem sa descoperim:

* entități de sistem
* actori de sistem (roluri)
* comportament
* atribute
* cum interacționează actorii cu resursele sistemului

* Dezvoltarea de diagrame UML pentru pachete și clase – capitol trei

Vom utiliza diagrama de pachete pentru a asigura un „good practice”. Cele 3 parți ale acestei arhitecturi fiind separate în câte un pachet: data, business si presentation. Vom utiliza diagramele UML de obiecte pentru a obține clasele necesare pentru rezolvarea obiectivului principal.

* Dezvoltarea algoritmilor – capitol trei

Pentru a ne atinge obiectivul principal, avem nevoie să dezvoltam algoritmi sau să căutam algoritmi ce realizeaza funcționalitatea dorită. Specific pentru acest proiect cel mai aproape de o descriere algoritimica sunt metodele care folosesc procesarea pe stream-uri si expresiile lambda.

* Implementarea soluției – capitol patru

Vom descrie clasele si interfata, si motivele pentru alegerea acestei implementari.

* Testarea – capitol cinci

Testarea consta in testarea prin intermediul interfetei grafice, dar si prin intermediul preconditiilor si postconditiilor, implementate cu ajutorul assert-urilor.

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

# 

Se dorește o soluție capabilă să rezolve problema generala: clientul comanda cu succes produse din meniul companiei. Sistemul trebuie să sustina si operatiile facute de ceilalti utilizatori (angajat si adminstartor). Toti utilizatorii se conectează folosind un nume de utilizator și o parolă, insa clientul are posibilitatea de a se intregistra, pe cand adminstartorii si angajatii sunt introdusi direct in cod.

Interfata trebuie sa fie intuitiva si usor de folosit, pentru ca o mare parte din aplicatie tine de introducere corecta a datelor de catre utilizator. Programul poate fi predispus erorilor, datorate introducerii gresite a dateleor de catre utilizator.

Modelarea problemei este extrem de importanat pentru a ajunge la soluția dorita. Structura aleasa fiind detaliata in capitolul trei si patru.

Caz de utilizare: procesarea comenzilor, gestionarea produselor

Actor principal: administrator

Scenariul principal de succes:

1. Administratorul introduce corect datele de logare (nume de utilizator si parola).
2. Se deshide interfata corespunzatoare adminstratorului.
3. Se introduc datele corecte si se selecteaza randurile corespunzatoare din tabelul ce contine meniul.
4. Se apasa butonul corespunzator pentru operatia dorita.
5. Se afiseaza rezultatele dorite, care sunt in concordanta cu datele din introduse pana in acel moment in sistem.
6. Se apasa butonul de back/sau se inchide fereastra si se revine la interfata principala, de log in.

Secvențe alternative:

Parola si numele de utilizator gresite:

- se afiseaza un mesaj corespunzator si se cere utilizatorului sa introduca datele corecte

Datele introduse in casutele text sunt gresite:

- se afiseaza un mesaj corespunzator si se cere utilizatorului sa introduca date corecte

Nu a fost selectat niciun rand din tabel:

- se afiseaza un mesaj corespunzator si se cere utilizatorului sa selecteze un rand din tabel

Caz de utilizare: plasare comenzilor

Actor principal: client

Scenariul principal de succes:

1. Clientul introduce datele de inregistrare (nume de utilizator si parola).

2. Clientul se logeaza cu datele introduse anterior.

3. Se deschide interfata pentru clienti.

4. Clientul filtreaza produsele dupa cele sapte campuri si mai apoi selecteaza produsele dorite si

le adauga la comanda.

5. Se introduce data actuala a comenzii sub formatul cerut.

6. Se apasa butonul de plasare a comenzii

7. Se apasa butonul de back/sau se inchide fereastra si se revine la interfata principala, de log in.

Secvențe alternative:

Parola si numele de utilizator gresite:

- se afiseaza un mesaj corespunzator si se cere utilizatorului sa introduca datele corecte

Datele introduse in casutele text sunt gresite:

- se afiseaza un mesaj corespunzator si se cere utilizatorului sa introduca date corecte

Nu a fost selectat niciun rand din tabele:

- se afiseaza un mesaj corespunzator si se cere utilizatorului sa selecteze un rand din tabel

Caz de utilizare: notificarea angajatului

Actor principal: angajat obisnuit

Scenariul principal de succes:

1. Angajatul introduce corect datele de logare (nume de utilizator si parola).

2. Se deshide interfata corespunzatoare angajatului.

3. Daca vreun client plaseaza vreo comanda angajatul este notificat si primeste detaliile comenzii.

4. Se inchide fereastra si se revine la interfata principala de log in.

Secvențe alternative:

Parola si numele de utilizator gresite:

- se afiseaza un mesaj corespunzator si se cere utilizatorului sa introduca datele corecte

3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfețe, relații, packages, algoritmi, interfață utilizator)

Se cere implementarea unei scheme specifice cu o arhitectura stratificata, in care se remarca ca decizii de proiectare folosirea design pattern-urilor, precum Composite Design Pattern pentru definirea claselor MenuItem, BaseProduct si CompositeProduct si Observer Design Pattern, folosit pentru a notifica angajatii de fiecare data cand se un client plaseaza o comanda noua.

Stratul de prezentare(presenatation) - conține toate clasele responsabile de interfata prezentata utilziatorului final si de trimiterea a rezultatului inapoi la sistem.

Stratul business - conține logica pe care se bazeaza aplicație pentru a functiona corect, conform cerintelor cerute.

Stratul de data access – contine clasele care se ocupa serializare si de scrierea in fisier .txt

Diagram

Description automatically generated

3.1 Diagrame UML

* Diagrama de pachete

Diagramele de pachete ne ajută să decompunem sistemul în subsisteme, astfel

descompunem sistemul nostru in trei pachete presentation, data si business.

Diagram

Description automatically generated

* Diagrama de clase

Diagrama de clase descrie clasele din sistem și relațiile dintre ele, ce pot fi asocieri, generalizari, de realizare, de dependenta. Clasele vor fi doar enumerate în aceast capitol, detaliile despre câmpuri și metode se vor dezvolta în capitolul cinci.

Diagram

Description automatically generatedClasele din pachetul business:

Clasele din pachetul data: Serializator, FileWrite

A picture containing text, screenshot, clipart, picture frame

Description automatically generated

Diagram

Description automatically generatedClasa din pachetul presemtation:

Clasa Main:

Text

Description automatically generated

* 1. Structuri de date folosite

Structurile de date folosite preponderent sunt listele (List<>). Se remarca folosirea interfetei Map, care se implementeaza cu structura concreta HashMap. Pentru a putea retine comenzile si produsele din fiecare comanada ne definim o structură de tip Map <Order, Collection <MenuItem>>. Cheia Map-ului va fi

formata din obiecte de tip Order, pentru care vom suprascrie metoda hashCode() (si metoda equals())

pentru a calcula valoarea hash-ului din map pe baza atributeleor comenzii (orderID, clientID, data).

Structura HashMap foloseste tabele de dispersie pentru a-si realiza functionalitate.

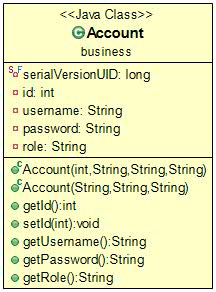
* 1. Detalii legate de implementare

Este important sa mentionm si conceputul de serializare, care consta in conversia unei stari a unui obiect

intr-un flux de octeti si cel de deserializare, care consta in recreerea unui obiect real Java in memorie ditr-un flux de biti. Ne folosim de serializare pentru a persista datele pe care le introducem in aplicatie si pentru a le deserializa la urmatoarea folosire.

# Implementare

4.1 Clasele din pachetul business

* Clasa Account

Clasa Account modeleaza userii pe care ii suporta aplicatia. Clasa are patru atribute: id, username, password si role. Pe baza username-ului si a password-ului se face logarea in aplicatie, iar role descrie tipul de user: administrator, client sau angajat.

Clasa are doi constructori unul cu id si unul fara pentru ca avem nevoie sa creeam account-uri fara id, acesta fiind stabilit de DeliveryService in functie de ce id-uri au si ceilalti utilizatori din aplictaie.

Clasa implementeza interfata Serializable, pentru ca va trebui sa serializam DeliveryService.

De asemenea, avem gettersi si settersi pentru fiecare camp din clasa.

Table

Description automatically generated with medium confidence

* Clasa MenuItem

Clasa MenuItem este folosita ca si componenta de uz

General in cadrul Composite Design Patter. Clasele

BaseProduct si CompositeProduct mostenesc din

acesta clasa metodele, dar ele vor fi suprascrise de

fiecare in parte. MenuItem este un element generic din

ierarhia de produse.

De asemenea, clasa implementeaza interfata Serializable

pentru ca DeliveryService va trebui serializat.

Text

Description automatically generated with medium confidence

* Clasa BaseProduct

.

Clasa BaseProduct este clasa responsabila pentru stocare

produselor de baza ce sunt importate din .cvs . Clasa are

sapte campuri care descriu un produs de baza: title, rating,

calories, proteins, fats, sodium si price.

Clasa extinde MenuItem si face parte din Composite Design Pattern reprentand o frunza in ierarhie.

Clasa va suprascrie toate metodele mostenite de la MenuItem, desi sunt numite compute, metodele sunt de

fapt gettersi pentru ca toate valorile au fost „calculate” deja

in fisierul .cvs.

* Clasa CompositeProduct

Table

Description automatically generated

Clasa CompositeProduct este responsabila de produsele

Compuse, de exemplu un meniu. Campurile clasei sunt

numele noului produs creat din agregarea a mai multor

produse, care pot fi simple sau la randul lor compuse. (de

aceea avem o lista de MenuItem, pentru a pastra

caracterul general.

Clasa extinde MenuItem si face parte din Composite Design Pattern reprentand o produs compus in ierarhie

(un subarbore).

Clasa va suprascrie toate metodele mostenite de la MenuItem, si va calcula valorile asociate pentru toate

produsele din lista.

* Text

  Description automatically generatedClasa Order

Clasa Order modeleaza o comanda data de un client.

Campurile clasei sunt: orderID, clientID, orderDate si

totalAmount.

Pentru ca vom folosi clasa Order pe post de key in Map,

vom suprascrie hashCode(), pentru a fi calculat in functie de

campurile clasei. Daca modificam hashCode-ul este

recomandat sa suprascriem si metoda equals().

De asemenea, Clasa implementeaza interfata Serializable.

Text

Description automatically generated

* Clasa DeliveryService

Clasa DeliveryService este clasa pricipala, ea modeleaza firma de catering. Campurile clasei sunt menu, care contine toate produsele pe care le

pune la dispozitie firma de catering, users, care contine datele pentru toti utilizatorii din aplicatie si orders, care este o asociere intre o anumita comanda si lista produselor comandate. Int-ul

lastOrderId ne ajuta la numerotare auto-increment a id-urilor.

Clasa implementeaza interfata IdeliveryService

care ne obliga sa scriem metode concrete care ofera functionalitate pentru adminstrator (operatii

basic de stergere/adaugare/modificare, dar si cea de importare a produselor din fisierul .cvs, plus

generarea rapoartelor) si pentru client (searchProdcts, care ne lasa sa filtram produsele dupa anumite criterii). Metodele se folosesc de expresii lambda si de procesarea pe stream-uri pentru a face o filtrare cat mai usor de implementat si inteles.

Clasa implemnteaza, de asemnea, si interfata Serializable(vrem sa salvam datele si sa le incarcam din la pornirea app) si Observble (care notifica observatorii-angajatii de comenzile pe care le plaseaza clientii).

* 1. Clasa din pachetul data

Text

Description automatically generated

* Clasa Serializator

Clasa este responsabila pentru serializarea si deserializarea deliveryservice, pentru a face posibila salavarea datelor apliactiei si reincarcarea acestora la pornirea aplicatiei.

.

* Text

  Description automatically generatedClasa FileWrite

Clasa se ocupa de redactarea facturii pentru o anumita

comanda, acest lucru se va face atunci cand clientul plaseaza

o coamanda.

* 1. Clasele din pachetul presenation

Sunt clasele care se ocupa de interfata grafica a aplicatiei. Se remarca doar clasa EmployeeGUI care implementeaza intefata Observer (si prin urmare trebuie sa implementeze concret update), pentru a fi notificata atunci cand un client plaseaza o noua comanada.

Prezentarea interfetei aplicatiei:

A screenshot of a map

Description automatically generated with medium confidence

Pagina de log in, doar clienti pot sa-si faca cont cu Register, administratorii si angajatii trebuie sa fie

introdusi in cod. In functie de username si parola se va deschide interfata corespunzatoare.

Graphical user interface, application

Description automatically generated

Interfata pentru adminstrator in care se pot face operatii basic add/modify/delete, dar si importarea

produselor din .cvs, crearea de noi produse compuse si genrarea de rapoarte.

Graphical user interface, website

Description automatically generated

Putem genera patru tipuri de rapoarte, a caror parametrii trebui

dati in causutele text.

Graphical user interface, application, table

Description automatically generated

Interfata clientului care poate sa selecteze produse direct din meniu, dar poate sa filtreze si dupa anumite

Criterii produsele care vor aparea in cel de-al doilea tabel. Dupa ce toate produsele dorite au fost adaugate

si data comenzii completate, atunci putem plasa comanada.

A picture containing graphical user interface

Description automatically generated

Interfata pentru angajat este simpla, permite notificare cum ca o comanada cu anumite produse a fost plasata.

# 5. Rezultate

Pentru a ne asigura că programul functioneaza cum ne dorim vom efectua mai multe teste pentru diferitele operatii puse la dispozitie prin interfata utilizatorului. Cu ajutorul preconditiilor si postconditilor implementate cu assert-uri putem verifica mereu daca totul se desfasoara corect. Folosim si metoda wellFormed() pentru a verifica starea generala a aplicariei.

6. Concluzii

Prin implementarea acestei teme mi-am aprofundat cunostințele de programare in Java, cunostințele despre postconditii si preconditii, structui de date bazate pe HashTables, expresii lamda si procesares stream-urilor. De asemenea, am exersat și construirea unei interfețe, si am intrat in detalii privind lucrul cu tabelele JTabel.

Ca și posibilitați de dezvoltarea ulterioară, se poate lucra la interfața, o realizare mai „user friendly”, care pune la dispozitie utiliatorului mai multe functionalitati, iar modul de operarea asupra datelor s-ar putea face strict pe JTable . De asemenea, mesajele de eroare la introducerea datele ar putea fi mult mai explicite. La nivel de proiectare se pot aduce înbunătațiri, iar codul poate fi optimizat.

# 7.Bibliografie

[1] Cursurile de tehnici de progarmare

[2] Cursurile de POO

[3] Wikipedia

[4] https://stackoverflow.com

[5] <https://www.geeksforgeeks.org/>

[7] Prezentarea suport