

**Tehnici de programare**

**Documentatie Tema 4**

**FOOD DELIVERY MANAGEMENT SYSTEM**

Muresan Ioana Danina

Grupa 30226

Cuprins:

1.Obiectivul temei

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

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

4.Implementare

5.Rezultate

6.Concluzii

7.Bibliografie

**1.Obiectivul temei**

Cerinta proiectului a fost: Proiectarea și implementarea unui sistem de management al livrărilor de alimente 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.

Aplicatia trebuie implementata in conformitate cu modelul arhitecturii stratificate si ar trebui sa utilizeze (minim) urmatoarele clase:

-Clasele din pachetul Presentation Layer

-Clasele din pachetul Business Layer

-Clasele din pachetul Data Layer

**Obiectivul principal** : Proiectarea si implementarea unei aplicatii de gestionare a comenzilor de alimente pentru o companie de catering

**Subobiective** :

• Analizați problema și identificați cerințele

• Proiectați aplicația de gestionare a comenzilor

• Implementarea aplicatiei de gestionare a comenzilor

• Testați aplicația de gestionare a comenzilor

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

Caz de utilizare: adăugați produsul în meniu

Actor principal: administrator

Principalul scenariu de succes:

1. Administratorul selectează opțiunea de a adăuga un produs nou

2. Aplicatia va afisa un formular in care detaliile produsului

se inserează

3. Administratorul introduce numele produsului, numărul de

calorii, proteine, grăsimi, sodiu și prețul său

4. Administratorul face clic pe butonul "Adăugați"

5. Aplicația salvează datele produsului și afișează o

mesaj de confirmare

Secvență alternativă: valori nevalide pentru datele produsului

- Administratorul introduce o valoare negativă pentru numărul de calorii/proteine/grăsimi/sodiu/prețul produsului

- Aplicatia afiseaza un mesaj de eroare si solicita angajatului pentru a insera o valoare validă pentru numărul de calorii/proteine/grasimi/sodiu/pret

- Scenariul revine la pasul 3

Diagram

Description automatically generated

Primul pas pe care trebuie sa il facem pentru a putea testa cu succes programul este sa alegem daca vrem sa ne logam ca si client,angajat sau administrator,introducand un username si o parola.

Apoi,in functie de ce am ales,putem face operatiile necesare.

1. **Composite Design Pattern** este un “ design pattern “ de partitionare si descrie un grup de obiecare ce e tratat ca o singura instanta a aceluias tip de obiect . Intentia sa este sa “ compuna “ obiectele in 3 structuri , ce reprezinta ierarhii partiale . Acest lucru ti permite sa ai o structura de tip “ arbore “ , si sa faci fiecare nod din structura arborelui sa efectuezre un anumit task . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Composite Design Patern are 4 participanti :

1. Componenta : Componenta declara interfata pentru obiectele din compozitie si pentru accesul si controlul componentelor-copii . Implementeaza , de asemenea , in mod implicit , si un comportament pentru interfata comuna tuturor claselor , dupa caz .
2. Frunzaele : Frunzele definesc comportamentul pentru obiectele primitive din compozitite .
3. Compozitia : Compozitia stocheaza componentele-copii si implementeaza operatii legate de copii , in interfata de componente .
4. Clientul : Clientul manipuleaza compozitia obiectului prin interfata componentelor .
5. **Observer Pattern** este utilizat cand avem relatii de tip “ one – to – many “ intre obiecte , astfel incat daca un obiect este modificat , obiectele sale dependente sunt informate automat . “ Observer Pattern “ se incardreaza in categoria modelului comportamental .

“Observer Pattern “ utilizeaza 3 clase :” Subject “ , ” Observer “ si “ Client “ . “ Subject “ este un obiect cu metode ce ataseaza si detaseaza observatorii de obiectele “ client “

Pentru a ne administra datele in mod eficient , utilizam : . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1. **HashSet :** “ HashSet “ implementeaza intergata “ Set “ , sprijinita de un “ hash table “ , care este , de fapt , o instant “ HashMap “ . Nu ofera nicio garantie cu privire la ordinea de interatie a setului , ceea ce inseamna ca , clasa nu garanteaza ordinea constanta a elementelor de-a lungul timpului . Aceasta clasa permite elemenul “ null “ . Clasa poate oferi , de asemenea , performanta timpului constant , pentru operatii de baza , ca “ add ” , “ remove “ , “ contains “ and “ size “ , presupunand ca functia “ hash “ disperseaza elementele corespunzator .

2. **HashMap** : “ HashMap “ asigura implementarile de baza ale intefetei “ Map “ . Stocheaza informatiile in perechi ( Key , Value ) . Pentru a accesa o valoare , trebuie sa ii cunoastem cheia . “ HashMap “ este cunoscuta ca “ HashMap “ , deoarece utilizeaza o tehnica numita “ Hashing “ . “ Hashing “ este o tehnica de convertire a unui String mare , intr-un string mic , ce reprezinta de fapt acelasi String . O valoare mai mica ajuta in indexarea si cautarea mai rapida . “ HashSet “ utilizeaza intern “ HashMap “ . Intern , foloseste un “ link list “ pentru a stoca perechi “ key - value

**· Cazuri de utilizare**

Atunci cand se apasa butonu ‘Run’, se deschide interfata grafica , care este usor de utilizat. Apar 4 butoane: Administrator , Client , Employee si Register si doua text- fielduri pentru a introduce un username si o parola. Utilizatorul va apasa butonul aferent operatiei pe care doreste sa o realizeze si va deschide unul dintre cele 3 frame – uri,dup ace introduce o combinatie buna de id si parola.

In cadrul frame – ului ‘Admin‘ utilizatorul poate importa produsele,poate adauga un produs nou,crea un produs compus , sterge un produs , edita detaliile unui produs sau genera rapoarte despre comenziile realizate de clienti.

In cadrul frame – ului ‘Clients ‘ utilizatorul poate adauga o comanda noua , cauta niste produse dupa anumite criterii sau poate vedea lista cu toate produsele.

In cadrul frame – ului ‘Employee‘ utilizatorul este notificat de fiecare data cand se realizeaza o comanda noua.

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

Pentru a respecta paradigmele POO, programul a fost impartit in mai multe pachete cu nume sugestive care contin diferite clase:

-Pachetul GUI : Contine clasele : ResultView ,ClientView, Start , OrdersView, MainFrame ,CreateComposedProductView ,AdminView . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

-Pachetul DataAccesLayer : Contine clasele: Serializator, FileWriter

-Pachetul BusinessLogicLayer : Contine clasele BaseProduct ,CompositeProduct, DeliveryService, MenuItem ,Order si interfata IDeliveryServiceProcessing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**Diagrama UML a proiectului:**

Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora ( numite și obiecte ). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte, pentru business Process Design etc.

Graphical user interface

Description automatically generated

**4.Implementare**

**-Pachetul GUI :**

**Contine clasele :**

**ResultView :** Aici voi scrie rezultatele unor interogari. De exemplu,voi afisa o lista de produse,selectate dupa anumite criterii.

**ClientView:** Contine 3 butoane: Search product, care deschide o noua fereastra unde vor fi afisate produsele selectate dupa anumite criterii, cu ajutorul unui JCombox. Al doilea buton este pentru crearea unei comenzi iar ultimul pentru a vedea toate comenzile. De asemenea,mai contine si un tabel,unde importez toate produsele din product.csv( le importeaza in prealabil admin-ul). Pe langa butoane ,jcombox si tabel, clasa mai contine si 7 textFielduri, unde pot introduce date despre un nou produs( title, rating, calories, protein, fat, sodium si price). Daca se introduce o valoare negative,interfata va afisa un mesaj de eroare.

**Start :** De aici pornesc mainFramul, contine metoda de main

**OrdersView :** Se deschide o interfata noua,unde voi afisa toate comenziile inregistrate.

**MainFrame :** contine 4 butoane: admin🡪 duce spre AdminView() , client🡪 duce spre ClientView(), employee🡪 duce spre EmployeeView() ,register 🡪 duce spre RegisterView() .

**CreateComposedProductView :** contine un tabel cu lista de produse compuse din produse simple( se pot alege variante 3 variante de produse, pe care le alegem cu ajutorul a 3 jComox , si introducem intr-un jTextField un nume ( de exemplu “Meniu1”). Clasa mai contine un buton de create,si de back . Butonul de create deschide tabelul cu lista de produse compuse, numele,numele produselor, si pretul total ( construit ca suma celor 3 produse de baza).

**AdminView** : Contine 11 butoane: “Import products”, “Add new product” ,” Delete product”, “Edit product” ,”Generate raport 1”, ”Generate raport 2”, ”Generate raport 3”, ”Generate raport 4” ,”Create new products “, “View composed products” si butonul de “Back” . Clasa mai contine si 9 JTextField-uri : , unde se pot introduce date despre un nou produs( title, rating, calories, protein, fat, sodium si price ) dar si date1, date2 unde se pot introduce doua date pentru cand s -au facut comenzile .Acestea vor fi introduce cu formatul : “yyyy-MM-dd hh:mm” .De exemplu, 2022-05-09 11:28 este o data valida. De asemenea, clasa mai contine si un tabel unde vor fi importate datele din fisierul products.csv .

**-Pachetul DataAccesLayer :**

**Contine clasele:**

**Serializator:** are 2 metode : Serialization ( ) : ia datele din parametrul “restaurant “ si le scrie in fileStreamObject si le pune intr-un fisier

Deserialization ( ) : creeaza o noua instanta a restaurantului folosind informatiile salvate in fisier in urma serializarii

**FileWriter :** contine metoda de facture care va genera intrun fisier bill.txt o facture( date pentru o comanda plasata)

**-Pachetul BusinessLogicLayer :**

**Contine clasele**

**BaseProduct :** are doar metodele getters si setters si metoda pentru calcularea pretului definite in clasa abstracta MenuItem

**CompositeProduct :** are doar metodele getters si setters si metoda pentru calcularea pretului definite in clasa abstracta MenuItem

**DeliveryService :**

**MenuItem**

**Order :** contine getters si setters pentru fiecare atribut , si , in plus , avem si clasa hashCode ,care calculeaza codul functiei Hash al fiecarei comenzi .

**interfata IDeliveryServiceProcessing**

**5.Rezultate**

*Se vor prezenta scenariile pentru testare cu Junit sau alt framework de testare.*

*Chart

Description automatically generatedCalendar

Description automatically generatedTable

Description automatically generated*

**6.Concluzii**

In concluzie , sunt de parere ca acest proiect mi-a aprofundat cunostiintele in tot ce inseamna limbajul Java , implementarea paradigmelor POO , creearea unui program cu o interfata grafica . De asemenea , m-a ajutat sa imi reamintesc tehnicile de programare invatate in semestrul trecut . În același timp, a avut un rol important și în aprofundarea cunoștințelor despre HashTabel si HashMap. Totodata, felul in care am structurat proiectul si am creat clasele si metodele m-a ajutat sa eficientizez codul din punct de vedere al lungimii si al usururintei intelegerii acestuia, lucruri vitale in cazul programelor complexe care necesita mai multe clase sau care presupun un numar mai mare de programatori.

La capitolul imbunatatiri si dezvoltari ulterioare intra foarte multe elemente : id-ul clientilor si parola sa fie salvate prin serializare, sa fie mai multi angajati si clienti, sa fie niste interfete mai dragute. Ca si eficienta, as putea imbunatati partea de procesare a comezilor deoarece pun random un id pentru client si pentru comanda, si posibil sa se nimereasca doua persoane cu aceasi id sau doua comenzi cu acelasi id si sa fie tratate separat.

**7.Biibliografie**

1. <http://www.tutorialspoint.com/java/java_serialization.htm>
2. <https://www.baeldung.com/java-serialization>
3. <https://www.geeksforgeeks.org/serialization-in-java/>
4. <https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html> <http://javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.html>
5. <http://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html>
6. <http://javarevisited.blogspot.ro/2012/01/what-is-assertion-in-java-java.html>
7. <http://stackoverflow.com/questions/11415160/how-to-enable-the-java-keywordassert-in-eclipse-program-wise>
8. <https://intellij-support.jetbrains.com/hc/en-us/community/posts/207014815-How-toenable-assert>