DOCUMENTAȚIE

TEMA *4*

NUME STUDENT: Georgian Andreea

GRUPA: 30221

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 3](#_Toc95297887)

[4. Implementare 3](#_Toc95297888)

[5. Rezultate 3](#_Toc95297889)

[6. Concluzii 3](#_Toc95297890)

[7. Bibliografie 3](#_Toc95297891)

1. **OBIECTIVUL TEMEI**

Obiectivul principal al acestei teme este proiectarea și implementarea unei aplicații, care are ca scop gestionarea comenzilor de mancare a unei firme de catering. Aplicatia pune la dispozitia utilizatorilor o gama larga de functii, cum ar fi: adaugarea unor produse in meniu, editarea produselor existente, crearea comenzilor de mancare, stergerea produselor si multe altele. De asemena, aplicatia emite o facture la plasarea fiecarei comenzi.

Obiectivele secundare ale temei sunt:

* Analizarea problemei și identificarea cerințelor
* Proiectarea aplicației de plasare si gestionare a comenzilor
* Implementarea aplicației de plasare si gestionare a comenzilor
* Testarea aplicației de plasare si gestionare a comenzilor

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

Comenzile online devin tot mai populare in ultimul timp, deoarece ele sunt mai eficiente. Gestionarea comenzilor unei firme de catering utilizand foi scrise de mana este dificila si necesita destul de mult timp, fapt care duce la intarzieri in coordonarea angajatilor si in livrarea comenzilor. Solutia ar fi o aplicatie de gestionare a comenzilor cu support pentru notificarea angajatilor.

Caz de utilizare : adaugarea unui produs in meniu

User principal : administratorul

Scenariul de succes:

1. Administratorul selecteaza optiunea de a adauga un produs nou.
2. Aplicatia va afisa un formular in care detaliile produsului ar trebui inserate.
3. Administratorul insereaza numele produsului, numarul de calorii, protein, grasimi, sodiu si pretul lui.
4. Administratorul apasa butonul pentru adaugare.
5. Aplicatia salveaza datele produsului nou si afiseaza un mesaj de confirmare.

Scenariu alternativ: Set de valori incorect

* Administratorul introduce o valoare negative pentru numarul de calorii / protein / grasimi / sodiu / pretul produsului
* Aplicația afișează un mesaj de eroare și îi solicit administratorului să introducă valori valide
* Scenariul revine la pasul 3

# Proiectare

Pentru stocarea produselor din meniu am folosit metoda “Composite Design Pattern” : o clasa abstracta “MenuItem”, in care sunt stocate dateliile produselor, constructor, setter-ele si getter-ele aferente, plus metodele de calculare a caracteristicilor, pentru produsele compuse din mai multe produse de baza. Aceasta clasa este mostenita de alte doua clase, “BaseProduct” si “CompositeProduct”.

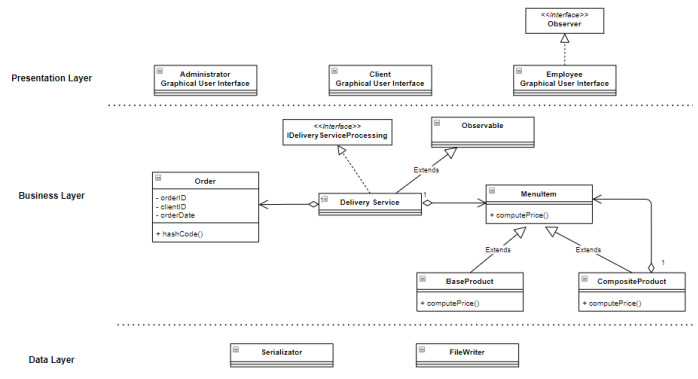
Stocarea utilizatorilor presupune patru clase : o clasa abstracta care retine un model al utilizatorilor, si alte trei clase care mostenesc aceasta clasa si retin detaliile fiecarui tip de utilizator (administrator, client si angajat).

De asemenea, functiile pe care aplicatia le are sunt definite in interfata “IDeliveryServiceProcessing”, urmand sa fie scrise in clasa “DeliveryProcessing”, care implementeaza interfata.

Pe langa acestea, mai exista clasele”Serializator” si “DataFileWriter”, care ajuta la exportarea si importarea datelor comenzilor.

Implementarea interfeței grafice este făcută după modelul MVC (model-view-controller). Acesta este un model pentru proiectarea software-ului utilizat în mod obișnuit pentru a implementa interfețe cu utilizatorul, date și logica de control. Presupune o separare între logica software-ului și afișaj. Această “separare a preocupărilor” asigură o mai bună diviziune a lucrurilor și o întreținere în timp îmbunătățită. Multe interfețe se bazează pe acest șablon. Ideea este separarea programelor în Model, View (vedere, vizualizare) (creează afișajul, interacționând cu Modelul după nevoi), și Controller (răspunde la cererile utilizatorului, interacționând atât cu Vizualizarea cât și cu Modelul) după nevoi.

Unified Modeling Language (UML) joacă un rol important în dezvoltarea de software, dar și în pentru sistemele non-software din numeroase domenii, deoarece este o modalitate prin care puteți vizualiza comportamentul și structura unui sistem sau proces. UML ajută la prezentarea posibilelor erori din structurile aplicațiilor, comportamentele sistemelor și alte procese de afaceri.

****UML a apărut pentru prima dată în anii 90, datorită lui Grady Booch, Ivar Jacobson și James Rumbaugh, trei ingineri de software care au dorit să dezvolte o modalitate mai puțin haotică de a reprezenta dezvoltarea de software din ce în ce mai complexă, separând, în același timp, metodologia de proces. În prezent, UML este în continuare notația preferată atât pentru dezvoltatori, cât și pentru manageri de proiect, deținători de companii, antreprenori din domeniul tehnologiei și specialiști din diferite domenii.

# Implementare

**CLASA ‘MENUITEM’**

Aceasta este o clasa abstracta care ajuta la gestionarea produselor din aplicatie. Exista doua tiouri de produse: produse de baza si produse compuse, care au sunt formate dintr-o lista de mai multe produse de tip “MenuItem”. Acest model de gestionare a datelor se numeste “Composite Design Pattern”.

Clasa “MenuItem” contine detaliile unui produs (nume, rating, calorii, protein, grasimi, sodiu, pret), un constructor are sa initializeze toate aceste variabile, plus setter-ele si getter-ele acestor variabile. In plus, mai exista metodele abstracte care calculeaza variabilele mentionate anterior pentru un produs compus.

**CLASA ‘COMPOSITEPRODUCT’**

Aceasta clasa mosteneste clasa mentionata anterior. Astfel, aceasta contine implementarea metodelor abstracte din clasa mama. Pentru a obtine valorile corecte ale variabilelor trebuie sa adunam toate acele variabile de la produsele care se afla in lista care construiesc produsul compus.

O imagine care conține text

Descriere generată automat

**CLASA ‘SERIALIZATOR’**

O imagine care conține text

Descriere generată automatAceasta clasa preia informatiile din fisierul „product.csv” prin constructorul ei. De asemenea, mai exista o metoda prin care se adauga produsele in meniu, in care se verifica daca numele produsului care trebuie adaugat se regaseste deja in meniu, pentru ca acesta sa nu contina produse duplicate.

**CLASA ‘USER’**

Această clasa este o clasa abstracta, care retine informatii despre utilizator. Pe aceasta aplicatie se pot loga trei tipuri de utilizatori (Administrator, client si angajat), fiecare dintre ei avand diferite functionalitati care ii sunt puse la dispozitie. Fiecare tip de utilizator este retinut printr-o clasa separata, care mostenesc clasa “User”.

**CLASA ‘DELIVERYSERVICE’**

Aceasta clasa implementeaza interfata “IDeliveryServiceProcessing”. Ca si variabile, contine un HashMap care sa retina lista de comenzi care au fost facute pana acum, avand ca si cheie o variabila de tip “Order” si ca valoare o lista de variabile de tip MenuItem, adica produsele care au fost cumparate in comanda asociata lor. De asemenea, aceasta clasa mai are o lista separata de produse de tip MenuItem, pentru a retine toate produsele existente, care formeaza meniul firmei de catering. Pe langa acestea, se mai declara o variabila de tip Observer, deoarece angajatii sunt anuntati atunci cand o comanda este create prin metoda “Observer Design Pattern”.

La partea de metode, aceasta clasa contine toate oparetiile pe care clientul sau administratorul le poate face, dupa cum urmeza: administratorul poate sa importe produse, sa gestioneze produsele existente in meniu si sa genereze rapoarte, iar clientul poate sa creeze o comanda, fapt care implica calcularea pretului total pentru acea comanda si generarea unei facturi in format .txt, si sa caute produse bazandu-se pe diferite criterii.

De asemenea, pentru implementarea acestei clase s-a folosit metoda “Design by contract” (preconditii si postconditii pentru variabile).



O imagine care conține masă

Descriere generată automatMetoda pentru modificarea unui produs :

Crearea unei comenzi se face pe baza unui vector de indecsi, reprezentand pozitia pe care se afla elementele pe care utilizatorul le-a selectat pentru a face comanda.

O imagine care conține text

Descriere generată automat

O imagine care conține text

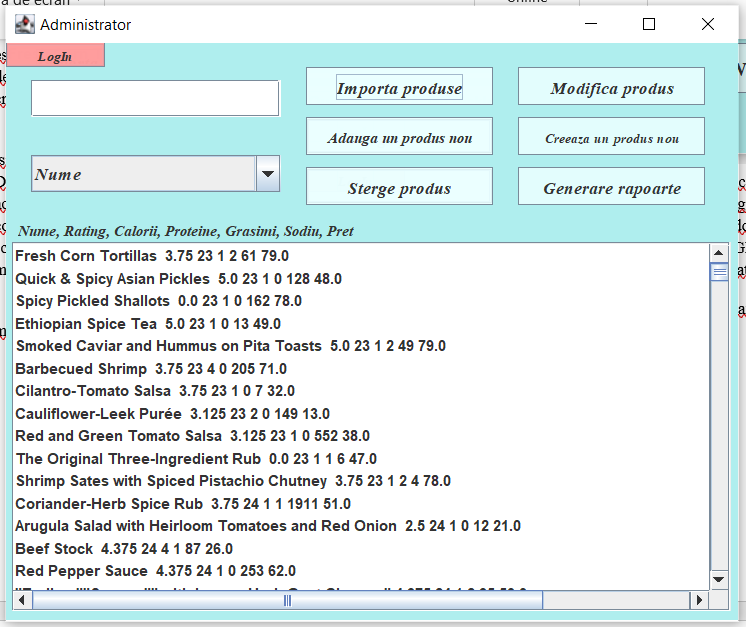
Descriere generată automatDe asemenea, utilizatorul poate sa genereze patru tipri de rapoarte, in format .txt: un raport cu acele comenzi efectuate intre o anumita ora de inceput si o anumita ora de final, introduse in interfata grafica de administrator, sau un raport cu produsele comandate de mai mult de un anumit numar de ori pana acum, sau un raport care sa contina clientii care au comandat de mai mult de un anumit numar de ori pana cum si valoarea comenzii a fost mai mare decat o suma specificata. De asemenea, mai poate fi generat un raport cu produsele copmandate intr-o zi specificata, impreuna cu numarul de ori in care au fost comandate acele produse.

O imagine care conține text

Descriere generată automat

**CLASA ‘VIEW’**

Această clasă face parte din modelul MVC pentru interfața grafică. Practic, această clasă redă conținutul unui model. Specifică exact cum se prezintă utilizatorului datele din model. Daca modelul se schimbă, vederea trebuie să își actualizeze prezentarea după nevoi. Vizualizarea și modelul sunt obligatoriu independente, dar comunică între ele.

****În cazul aplicației de față, exista mai multe interfete care compun aplicatia. Clasa care va fi prezentata in cele ce urmeaza este clasa “LogIn”. Aceasta interfata este prima care apare atunci cand aplicatia este rulata, deoarece aici utilizatorul trebuie sa isi introduca credentialele pentru a intra propriu-zis in aplicatie.

Interfata contine doua campuri, unde utilizatorul isi va scrie username-ul si parola pentru a accesa contul lor. Dupa ce aceste campuri sunt completate, exista doua scenarii: in cazul in care clientul nu are un cont pentru aplicatie, acesta va apasa butonul “Creeaza un cont nou”, iar datele introduse in prealabil vor fi folosite pentru a se adauga un cont in lista, iar clientul va putea folosi acele credentiale pe viitor cand va dori sa intre in aplicatie. Cel de al doilea scenario este in cazul in care utilizatorul are deja un cont existent, iar atunci acesta va apasa pe butonul “LOGIN”, moment in care datele introduse vor fi verificate si se va identifica tipul de utilizator care vrea sa acceseze aplicatia.

In momentul in care utilizatorul se logheaza la contul lui, se va deschide o alta fereastra in care este afisat meniul si acesta poate alege din butoanele existente urmatoarele lui actiuni.

**CLASA ‘MESSAGE’**

O imagine care conține text

Descriere generată automatAceastă clasă a fost creată pentru a afișa un mesaj de eroare în cazul în care utilizatorul introduce un set de date invalid, sau pentru a afisa un mesaj in cazul in care se creeaza o comanda noua sau se adauga un produs in meniu, pentru ca utilizatorul sa stie ca acea operatie s-a efectuat cu succes.

**CLASA ‘MAIN’**

Toate programele Java trebuie să aibă un punct de intrare, care este întotdeauna metoda main (). Ori de câte ori programul este apelat, acesta execută automat mai întâi metoda main ().

O imagine care conține text

Descriere generată automatPrincipala [metoda](https://www.greelane.com/link?to=method-signature-2034235&lang=ro&alt=https://www.thoughtco.com/method-signature-2034235&source=main-class-2034233) poate apărea în orice clasă care face parte dintr - o aplicație, dar în cazul în care cererea este un complex care conține mai multe fișiere, este comun pentru a crea o clasă separată doar pentru main (). Clasa principală poate avea orice nume, deși în mod obișnuit va fi numită doar „Main”, la fel cum s-a procedat și în proiectul de față.

# Rezultate

# 

Atunci cand un client face o comanda, se va genera automat o factura cu detaliile comenzii (numarul de comanda, id-ul clientului, numele si pretul produselor comandate si totalul pe care clientul trebuie sa il plateasca). De exemplu, data utilizatorul cu id-ul 2 comanda Chicken Satay Bites, in valoare de 35.0 de lei, Chocolate Walnut Biscotti, in valoare de 25.0 de lei, Smoked Salmon and Cucumber Squares in, valoare de 75.0 de lei, Basic Veal Stock, in valoare de 98.0 de lei si Cheddar Dill Puffs, in valoare de 74.0 de lei, atunci se va genera factura cu un total de 307.0 lei.

O imagine care conține text

Descriere generată automat

# Concluzii

Acest aplicatie online pentru comandarea mancarii si gestionarea produselor din mediul unei firme de catering este foarte utila în ceea ce privește scopul în care a fost creat, oferind o rezolvare pentru clasicele carnetele pe care se scrie de mana, care necesita timp in plus si duc la intarzieri in coordonarea angajatilor si in livrarea comenzilor. Aceasta are o interfață simplă, prietenoasă, ușor de înțeles și de folosit de către utlizator.

În plus, această aplicație respect conceptele fundamentale ale programării orientate pe obiect. S-au folosit metode de încapsulare, clase și obiecte. Toate obiectele care prezintă caracteristici și comportamente similar au fost grupate intr-o singura clasă.

De asemenea, modul în care a fost creata aceasta aplicatie pentru comandare si gestionare online o face ușor de îmbunătățit pe viitor, fiind unele funcții care nu sunt oferite în prezent și care pot fi implementate ulterior. De exemplu, comenzile, produsele adaugate si utilizatorii noi pot fi retinuti chiar daca aplicatia este inchisa, iar in acest fel nu se mai pierd informatii. In plus, clientul ar putea sa selecteze produsele dorite si sa le bage provizoriu intr-un cos, si doar mai apo isa plaseze comanda. In acest fel, utilizatorul poate selecta si cantitatea dorita dintr-un produs, in acest moment putandu-se comanda doar cate o bucata din fiecare produs,

O alta imbunatatire care poate sa fie adusa aplicatiei ar fi stocarea cantitatii disponibile pentru fiecare produs, astfel utilizatorul poate sa fie notificat in momentul in care comanda nu poate sa fie procesata din cauza ca nu exista destule produse in stoc pentru a-i fi livrate.

# Bibliografie

<https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>

<https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html>

[https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8- streams.html](https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-%20streams.html)

<https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/>

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

<http://www.tutorialspoint.com/java/java_serialization.htm>

<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>

<http://javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.html>

<http://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html>

<http://javarevisited.blogspot.ro/2012/01/what-is-assertion-in-java-java.html>

<http://stackoverflow.com/questions/11415160/how-to-enable-the-java-keywordassert-in-eclipse-program-wise>

<https://intellij-support.jetbrains.com/hc/en-us/community/posts/207014815>

<https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html>