

Facultatea de Automatică și Calculatoare

Tehnici de programare fundamentale

Laborator – Tema 4

FOOD DELIVERY MANAGEMENT SYSTEM

Profesor: Dr. Cristina Pop

Student: Talpoș Andreea Georgiana

Grupa: 30228

1. Obiectivul temei

Obiectivul temei este proiectarea și implementarea unui sistem de gestionare al livrării unor produse, în special mâncare, pentru o firmă de catering. Clientul care accesează aplicația va putea comanda produse din meniul companiei. Sistemul proiectat trebuie să aibă trei tipuri de utilizatori care se pot conecta utilizând un username și o parolă: administratorul, un angajat obișnuit și clientul.

**Administratorul** companiei poate executa următoarele operații:

* Importă setul inițial de produse care vor popula meniul dintr-un fișier de tip .csv
* Gestionează produsele din meniu: adaugă/ șterge/ modifică produse și creează noi produse compuse din mai multe produse
* Generează rapoarte despre comenzile depuse, după următoarele criterii:
  + Intervalul de timp în care au fost depuse comenzile: acest raport presupune afișarea comenzilor care au fost depuse într-un interval de timp dat de administrator
  + Produsele care au fost comandate mai mult decât un număr de ori dat
  + Lista clienților care au comandat de mai multe ori decât un număr dat și valoarea produselor comandate să fie mai mare decât o sumă furnizată de către administrator
  + Produsele care au fost comandate într-o anumită zi, însoțite de numărul care reprezintă de câte ori au fost comandate

**Clientul** care accesează aplicația poate executa operațiile:

* Să se înregistreze în sistem și să își utilizeze numele de utilizator și parola pentru a se conecta
* Poate vizualiza lista de produse din meniu
* Poate căuta produsele din meniu care îndeplinesc unele criterii precum: numele produsui conține un anumit cuvânt, după rating, după numărul de calorii, proteine, grăsimi, sau au un anumit preț
* Poate, de asemenea, crea o comandă care conține mai multe produse: pentru fiecare comandă se va genera o factură cu lista produselor, prețul acestora și prețul total al produselor

**Angajatul** companiei va fi notificat de fiecare dată când o nouă comandă a fost depusă de către un client, deci el va putea începe prepararea produselor comandate.

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

Această aplicație trebuie să îndeplinească toate cerințele necesare pentru ca mai întâi utilizatorii să se conecteze în aplicație și apoi, în funcție de identitatea utilizatorului ( administrator, client sau angajat), să poată realiza operațiile cerute.

Pentru îndeplinirea cerințelor trebuie utilizate design pattern-urile Observer Design Pattern( pentru a notifica angajații de fiecare dată când o nouă comandă a fost plasată) și Composite Design Pattern ( pentru definirea claselor BaseProduct, CompositeProduct care vor extinde MenuItem).

Design Pattern-ul *Observer* definește o relație de dependență 1..\* (one to many) între obiecte astfel încât când un obiect își schimbă starea, toți dependenții lui sunt notificați și actualizați automat. Folosirea acestui pattern implică existența unui obiect cu rolul de subiect, care re asociată o listă de obiecte dependente, cu rolul de observatori, pe care le apelează automat de fiecare dată când se întâmplă o acțiune.

Acest pattern este de tip Behavorial ( comportamental), deoarece facilitează o organizare mai bună a comunicației dintre clase în funție de rolurile acestora.

Design Pattern-ul *Composite* este utilizat când avem nevoie să tratăm un grup de obiecte în mod similar ca și cum ar fi un singur obiect. În cadrul acestei aplicații, este utilizat în cazul obținerii produselor compuse din mai multe produse de bază, aceste produse compuse fiind tratate ca un singur produs de tip MenuItem.

1. **Modelarea problemei**

Aplicația va dispune de o interfață cu utilizatorul care îi va oferi utilizatorului posibilitatea de a alege ce acțiuni dorește să execute. De fapt, va exista o interfață principală care va face legătura dintre utilizator și interfața corespunzătoare în funcție de rolul acestuia. În mod mai clar, vor exista interfețe pentru client, administrator și angajat. Fiecare dintre acestea vor avea butoane și câmpuri de text, pentru ca aplicația să poată recepționa cerințele utilizatorului, pe care pe urmă le va rezolva și va returna rezultatele.

1. **Scenarii și cazuri de utilizare**

Scenariile și cazurile de utilizare sunt metode folosite in analiza sistemelor pentru a identifica, clarifica și a organiza cerințele sistemului. Un caz de utilizare reprezintă un set de posibile interacțiuni dintre utilizator și aplicație. Deoarece cazurile de utilizare reprezintă, de fapt, pașii pe care utilizatorul îi are de parcurs, am încercat să proiectez o interfață cât mai ușor de utilizat (“User Friendly”).

**Diagrama UML – Use-case**

Diagram

Description automatically generated

**Titlu:** Food Management System

**Rezumat**: Primul lucru pe care trebuie să îl facă un utilizator este să se înregistreze în aplicație cu un nume de utilizator și o parolă și să își aleagă rolul: client, angajat sau administrator. În funcție de rolul utilizatorului, acesta va putea executa mai multe operații: administratorul – poate importa lista inițială de produse, poate crea produse compuse, adăuga produse noi,edita unele prduse existente și genera rapoarte în funcție de diferite criterii; clientul – vizualizează lista de produse din meniu, caută produse după anumite criterii, creează comenzi și după plasarea comenzii primește factura; angajatul doar este notificat în momentul în care o nouă comandă a fost plasată.

**Actorii**: administratorul, clientul și angajatul

**Scenarii de utilizare** – după cum se poate observa, există mai multe scenarii de utilizare

După cum am precizat anterior, utilizatorul deschide aplicația, își alege rolul și, dacă nu e înregistrat în aplicație, își creează un cont nou, adică introduce un nume și o parolă, altfel își introduce detaliile contului deja creat. Apoi, în funcție de tipul utilizatorulu:

**Administratorul** dispune de opțiunea de a importa setul de produse inițial, care este salvat în fișierul products.csv, poate adăuga un nou produs, furnizându-i detaliile necesare în câmpurile de text puse la dispoziție, produsul nu trebuie să existe deja în meniu, poate modifica detaliile unui produs sau poate șterge un produs. În plus, poate crea un nou produs compus dintr-o listă de produse simple și să îl adauge în meniu. Administratorul mai are facilitatea de a crea rapoarte despre comenzi și produsele comandate după anumite criterii: comenzi plasate într-un anumit interval de timp, cele mai comandate produse, clienții care au comandat de cel puțin un anumit număr de ori și o valoare mai mare decât cea introdusă, despre produsele care au fost comandate într-o anumită zi a săptămânii.

**Clientul** poate alege să vadă produsele conținute în meniu, să caute produse după anumite criterii: o parte de denumire ( key word) , rating, număr de proteine, calorii, preț etc. În plus, cea mai importantă facilitate este plasarea comenzilor, acesta trebuie să aleagă o listă de produse, după terminarea adăugării produselor și plasarea comenzii se va genera o factură și un angajat va fi notificat ca să pregătească comanda.

**Angajatul** se poate înregistra în aplicație, rolul său este de a realiza comenzile, va fi notificat de fiecare dată când o nouă comandă este plasată.

1. Proiectare

Proiectarea aplicației a fost realizată folosind un model arhitectural stratificat (Layered Architecture). Fiecare pachet are propriul scop și apelează funcții ale pachetelor de sub acesta.

**Presentation** – conține clasele care implementează interfața grafică a aplicației și interacțiunea dintre exterior( ceea ce vede utilizatorul) și logica interioară. Acest pachet are patru clase care implementează reprezentarea grafică, au rolul de a prelua datele introduse de către utilizatori și de a returna rezultatul cerut. Există o interfață principală în care se realizează înregistrarea utilizatorilor în aplicație. Fiecare utilizator își va alege rolul și va fi trimis către interfața corespunzătoare, deci există câte o fereastră pentru clienți, administrator și angajați, fiecare având câmpurile necesare pentru alegerea operației de executat, introducerea datelor de intrare și returnarea rezultatelor.

**BLL(business logic layer)** – conține clasele care încapsulează logica aplicației.

**DataLayer** – conține clasa *Serializator* pentru a reține informațiile clasei *DeliveryService* utilizând metoda serializării, în acest mod, modificările aduse asupra datelor vor fi stocate, clasa FileWriter pentru generarea facturii pentru client și clasele Admin, Client și Employee care reprezintă clasele utilizatorilor.

**Diagrama UML de pachete**

Diagram

Description automatically generated

**Pachetul PresentationLayer** conține 4 clase: MainGUI, AdministratorGUI, ClientGUI și EmployeeGUI în care sunt implementate ferestrele interfeței corespunzătoare fiecărui tip de utilizator, și fereastra principală ( MainGUI ) care oferă posibilitatea de autentificare sau înregistrare în aplicație a utilizatorilor

**Pachetul. DataLayer** – conține clasa *Serializator* pentru a reține informațiile clasei *DeliveryService* utilizând metoda serializării, în acest mod, modificările aduse asupra datelor vor fi stocate, clasa FileWriter pentru generarea facturii pentru client și clasele Admin, Client și Employee care reprezintă clasele utilizatorilor.

**Pachetul BLL** – conține clasele care împreună reprezintă logica aplicației. Clasa principală din acest pachet este DeliveryService care implementează interfața IdeliveryServiceProcessing, având definite metodele care realizează toate cerințele necesare îndeplinirii necesităților clienților, administratorului și angajaților.

**Diagrama UML de clase**

A picture containing text, sky, map

Description automatically generated

**Structuri de date**

Ca structură de date nouă am utilizat tipul Map<..<ArrayList>> pentru a stoca datele despre comenzi în clasa DeliveryService. Cheia colecției va fi reprezentată de obiectele de tipul Order, în care metoda hashCode() a fost suprascrisă pentru a calcula o valoare în funcție de atributele clasei Order( orderID, date..). Avantajul este că vor fi eliminate duplicatele, pe care oricum nu vrem să le avem.

**Interfața cu utilizatorul**

Pentru această aplicație inetrfața cu utilizatorul este compusă din patru ferestre între care se poate comuta cu ajutorul butoanelor( back). Fereastra principală este MainGUI care realizează autentificarea sau înregistrarea utilizatorului. Odată ales tipul de utilizator, prin apăsarea butonului corespunzător fiecărui tip, va apărea fereastra destinată operațiilor clientului sau cea destinată administratorului. În momentul în care clientul plsează o comandă interfața angajatului ( EmployeeGUI ) va apărea și va afișa detaliile comenzii.

1. Implementare

**Pachete** – am implementat pachetele descrise anterior :

* + - BLL
    - DataLayer
    - PresentationLayer

Implementarea claselor:

1. Pachetul BLL

În pachetul BLL se află clasele: MenuItem, BaseProduct, CompositeProduct, DeliveryService, interfața IdeliveryServiceProcessing, Csvreader și Order.

**Clasa MenuItem** modelează un obiect de tip BaseProduct sau CompositeProduct, având atributele corespunzătoare detaliilor despre produsele ce vor popula meniul restaurantului.

Clasa CsvReader execută citirea și procesarea produselor din fișierul products.csv, returnând o listă cu toate produsele distincte din lista inițială de produse care va fi importată de către administrator.

Clasa DeliveryService implementează interfața IdeliveryServiceProcessing în care sunt definite toate metodele care trebuie implementate pentru a se putea realiza cerințele clientului și ale administratorului. Metodele realizează operații asupra setului de produse și de asemenea, returnează informații despre produse și comenzi.

1. Pachetul DataLayer

În pachetul DataLayer se află clasele: Serializator, FileWriter, Admin, Client, Employee. Clasa Serializator se ocupă de serializarea și deserializarea datelor din DeliveryService, practic toate datele importante din aplicație, pentru a nu se pierde. Clasa FileWriter are rolul de a scrie factura clientului după ce acesta depune o comandă, factură ce va conține datele relevante despre comandă precum produsele comandate, prețul total al acestora și data în care a fost depusă comanda. Clasele Admin, Client și Employee se ocupă cu detaliile utilizatorilor, și anume numele de utilizator și parola utilizate pentru folosirea aplicației.

1. Pachetul PresentationLayer

Clasele acestui pachet sunt ferestrele care corespund cu fiecare utilizator + fereastra în care se realizează procesul de înregistrare în aplicație. Deci, în acest pachet este implementată interfața grafică cu utilizatorul.

1. Interfața grafică cu utilizatorul

Graphical user interface, PowerPoint

Description automatically generated

Aceasta este fereastra implementată în MainGUI cu ajutorul căreia utilizatorul se va conecta în aplicație, introducând datele cerute și alegându-și identitatea : client sau angajat.

A picture containing chart

Description automatically generated

Aceasta este fereastra implementată în clasa ClientGUI, corespunzătoare utilizatorului client, primele butoane din partea dreaptă îi permit cătarea produselor după criteriile dorite și vizualizarea produselor din meniu. În partea de jos, acesta trebuie să introducă denumiri de produse pe rând, apăsând după fiecare butonul *Add,* când termină trebuie să apese butonul OK iar comanda va fi plasată, și angajatul notificat.

Graphical user interface

Description automatically generated

Fereastra implementată în clasa AdministratorGUI corespunzătoare utilizatorului administrator care poate gestiona colecția de produse din meniu și poate genera rapoarte legate de comenzi.

PowerPoint

Description automatically generated with low confidence

În fereastra implementată în clasa EmployeeGUI se vor afișa detaliile legate de ultima comandă plasată. Această fereastră apare automat când o comandă nouă este generată.

1. Rezultate

În fișierul bill.txt va fi afișat conținutul unei comenzi plasate de către un client. Tot pentru client, în câmpul pentru afișarea rezultatelor corespunzătoare cerințelor de căutare, vor fi afișate produse din meniu. În câmpul de text din interfața administratorului se vor afișa rezultatele rapoartelor. În interfața pentru angajat la fiecare comandă depusă se vor afișa detaliile comenzii respective.

1. Concluzii

Proiectul a fost un bun exercițiu pentru a-mi reaminti conceptele POO învățate în trecut, și chiar aprofundarea acestora. Sunt câteva lucruri pe care le-am învățat din experiența acestui proiect pe care le voi prezenta în continuare. În primul rând, probabil cel mai important lucru este gestionarea corectă a timpului, pentru că o organizare bună, începerea din timp la rezolvarea unei sarcini asigură găsirea mai multor modalități de rezolvare. În al doilea rând, modelarea corectă a problemei de la început ajută la rezolvarea mai rapidă a acsteia și ne scapă de implementarea unor funcții care nu sunt utile.

Am ajuns la concluzia că întâmpinarea problemelor de implementare, la codul scris de mine, m-au determinat să cercetez și în acest mod am găsit mai multe modalități de rezolvare, deci am învățat multe concepte noi și le-am aprofundat pe cele cunoscute deja. Deci, prin prisma acestui proiect am reușit să aflu pașii elementari de urmat pentru proiectarea și implementarea unui sistem de gestionare a comenzilor a unui restaurant.

1. Bibliografie

<https://app.diagrams.net/> - pentru realizarea diagramelor

<https://creately.com>

[www.stackoverflow.com](http://www.stackoverflow.com)

<https://www.baeldung.com/java-maps-streams>

[www.wikipedia.org](http://www.wikipedia.org)

<https://www.youtube.com>

Pe lângă aceste resurse, le-am folosit și pe cele date în prezentarea temei.