UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

DOCUMENTAȚIE

APLICAȚIE PENTRU GESTIONAREA LIVRĂRILOR UNEI FIRME DE CATERING

BUMBU RĂZVAN-VASILE

GRUPA 30223

PROFESOR COORDONATOR: ALEXANDRU RANCEA

CUPRINS

1. OBIECTIVUL TEMEI

Scopul acestei teme este implementarea unui sistem de management al livrarilor de produse pentru o firma de catering. Sistemul are 3 tipuri de utilizatori care se pot loga utilizand o parola si un username: administrator, angajat si client.

„Food Delivery Management System” este o aplicatie destinata clientilor pentru a le usura munca. Cu totii avem momente cand nu avem chef sau timp sa gatim si recurgem la ideea de a comanda mancare online sau de a merge sa mancam la restaurant. Aplicatia mea isi propune sa vina in sprijinul clientilor, care vor putea sa comande produsele imediat cu doar cateva click-uri. Astfel clientul plaseaza comanda, angajatii firmei de catering vizualizeaza in timp real comenzile si livreaza produsele spre consumatori. Se preteaza pentru un singur restaurant sau mai multe - parteneri ai firmei de catering. Ofera un mod eficient de a comanda mancare care sa fie livrata in cel mai scurt timp. Clientul trebuie sa se autentifice utilizand un username si o parola intr-o interfata de logare pentru a se asigura securitatea informatiilor si poate sa isi selecteze produsele favorite si sa plaseze o comanda. Imediat dupa plasarea comenzii angajatii vor fi notificati despre aceasta, in acest fel realizandu-se procesarea eficienta a comenzilor.

In realizarea aplicatiei am avut in vedere utilizarea tehnicii de programare Design by Contract, folosirea de Design Patterns precum Observer si Composite si implementarea serializarii pentru a stoca datele sistemului. Mai mult corespondenta comanda-meniu este realizata prin intermediul unei structuri de tip HashMap. Obiectivul principal consta in familiarizarea cu aceste concepte si utilizarea lor in construirea logicii aplicatiei.

2. ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE

Analiza problemei presupune identificarea claselor proiectului si a functionalitatilor acestuia, precum si legaturile existente intre acestea. Asadar, folosind un numar minim de informatii, programarea orientata pe obiecte ne ofera perspectiva implementarii unei aplicatii folosindu-ne doar de infomatii de suprafata.

In analiza problemei vom porni de la cele 3 tipuri de utilizatori: administrator, angajat și client.

Administratorul poate efectua următoarele operații:

- Să importe produse dintr-un fișier .csv, pe baza cărora va crea produsele conținute de meniu
- Să importe produse în meniu
- Să șteargă produse din meniu
- Să editeze produsele din meniu

- Să creeze produse compuse (meniuri care conțin o supă, un fel principal, o garnitură și un desert)
- Să genereze rapoarte pe baza intervalului orar al comenzilor (se vor afișa comenzile efctuate în intervalul orar specificat)
- Să genereze rapoarte pe baza numărului minim de comenzi al unui produs (se vor afișa produsele care au fost comandate de mai multe ori decât numărul specificat)
- Să genereze rapoarte pe baza numărului minim de comenzi efectuate de clienți și pe baza valorii minime a comenzii (se vor afișa clienții care au comandat de minim numărul specificat de ori, comenzile lor având minim valoarea specificată)
- Să se genereze rapoarte pe baza datei specificate (se vor afișa produsele care au fost comandate în data specificaă, dar și numărul de comenzi)

Angajatul poate efectua următoarele operații:

- Este notificat de fiecare dată când este depusă o nouă comandă

poate să:

- Își creeze con nou în aplicație
- Să vizualizeze lista de produse din meniu
- Să adauge produsele din meniu în coșul de **Clientul** cumpărături
- Să filtreze produsele din meniu
- Să plaseze comenzi

Alte cerințe specific:

- Definirea clasei IDeliveryServiceProcessing care sa contina operatiile principale care pot fi executate de client si administrator dupa cum urmeaza:
  - ✓ Administrator: importa produse, gestioneaza produsele din meniu, genereaza rapoarte
  - ✓ Client: creaza o comanda noua care presupune calcularea pretului per comanda si generatea unei facturi in format .txt, cautarea produselor dupa mai mutle criterii

- Definirea si implementarea claselor din diagrama UML de mai sus
  - ✓ Folosurea design pattern-ului Composite pentru definirea claselor MenuItem, BaseProduct si CompositeProduct
  - ✓ Folosirea design pattern-ului observer pentru a notifica un angajat de fiecate data cand se plaseaza o comanda noua

- Implementarea clasei DeliveryService folosind o structura predefinita in JCF(Java Collection Framework) care foloseste ca si structura de date un hashtable. Cheia tabelei

va fi generata folosind clasa Order, iar fiecare comanda va avea asociata o colectie de MenuItems

- ✓ Definirea unei structuri Map<Order, Collection <MenuItem>> pentru a stoca informatiile despre o comanda in clasa DeliveryService. Cheia Map-ului va fi formata dintr-un obiect de tip Order, pentru care se va suprascrie metoda hashCode() pentru a calcula valoarea has-ului in cadrul Map-ului din atributele clasei Order (OrderID, date, etc.)
- ✓ Definirea unei structuri de tip Collection<MenuItem> care va stoca meniul (toate produsele) oferite de firma de catering.
- ✓ Definirea unei metode de tip „well formed" pentru clasa DeliveryService.
- ✓ Implementarea clasei folosind metoda Design By Contract (contine pre,post conditii, invarianti si asertiuni)

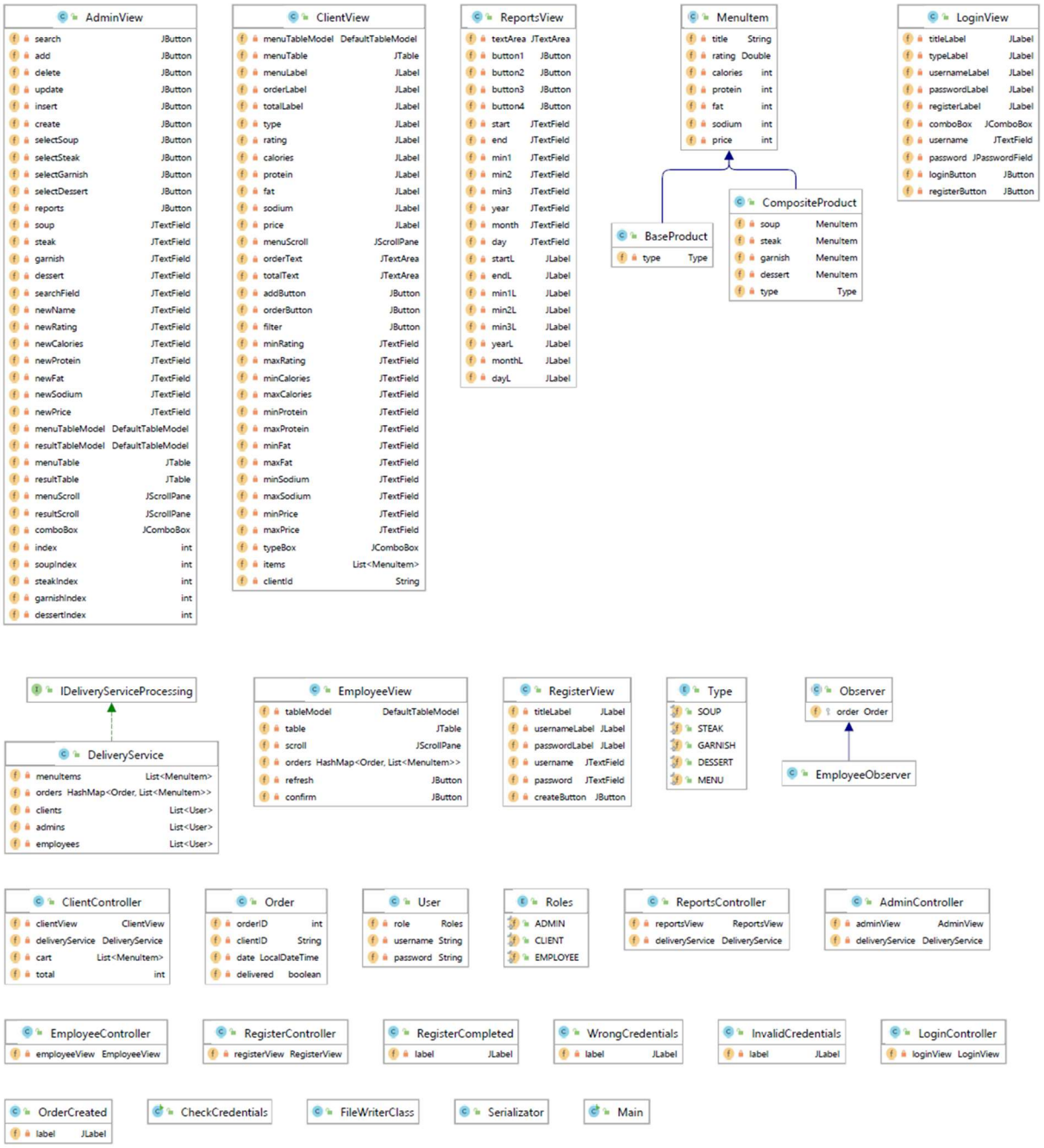
- Produsele din meniu, comenzile si informatiile despre utilizatori vor reprezenta informatie persistenta, si vor si salvate folosind serializarea pentru a fi disponibile pentru accesari ulterioare ale sistemului folosind deserializarea

In timpul utilizarii aplicatiei, la inceput se va deschide o fereastra de logare prin intermediul căreia utilizatorul este lăsat să își aleagă tipul, iar apoi pe baza userului și a parolei să intre in contul său. Apoi, in functie de tipul utilizatorului se va deschide o fereastra care va contine operatiile specifice.

3. PROIECTARE

Unified Modeling Language (UML) este un limbaj standard pentru descrierea de modele și specificații pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea 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.

Diagrama UML a calculatorului este formată din 7 pachete : App, Business Layer, Data Layer, Presentation Layer Controllers, Presentation Layer Message Boxes, Presentation Layer Views și din 30 de clase: Main, Base Product, Composite Product, Delivery Service, IDelivery Service Processing, Menu Item, Order, Type, Employee Observer, File Writer, Observer, Roles, Serializator, User, Admin Controller, Client Controller, Employee Controller, Login Controller, Register Controller, Reports Controller, Invalid Credentials, Order Created, Register Completed, Wrong Credentials, Admin View, Client View, Employee View, Login View, Register View, Reports View.


AdminView
search JButton
add JButton
delete JButton
update JButton
insert JButton
create JButton
selectSoup JButton
selectSteak JButton
selectGarnish JButton
selectDessert JButton
reports JButton
soup JTextField
steak JTextField
garnish JTextField
dessert JTextField
searchField JTextField
newName JTextField
newRating JTextField
newCalories JTextField
newProtein JTextField
newFat JTextField
newSodium JTextField
newPrice JTextField
menuTableModel DefaultTableModel
resultTableModel DefaultTableModel
menuTable JTable
resultTable JTable
menuScroll JScrollPane
resultScroll JScrollPane
comboBox JComboBox
index int
soupIndex int
steakIndex int
garnishIndex int
dessertIndex int
ClientView
menuTableModel DefaultTableModel
menuTable JTable
menuLabel JLabel
orderLabel JLabel
totalLabel JLabel
type JLabel
rating JLabel
calories JLabel
protein JLabel
fat JLabel
sodium JLabel
price JLabel
menuScroll JScrollPane
orderText JTextArea
totalText JTextArea
addButton JButton
orderButton JButton
filter JButton
minRating JTextField
maxRating JTextField
minCalories JTextField
maxCalories JTextField
minProtein JTextField
maxProtein JTextField
minFat JTextField
maxFat JTextField
minSodium JTextField
maxSodium JTextField
minPrice JTextField
maxPrice JTextField
typeBox JComboBox
items List<MenuItem>
clientId String
ReportsView
textArea JTextArea
button1 JButton
button2 JButton
button3 JButton
button4 JButton
start JTextField
end JTextField
min1 JTextField
min2 JTextField
min3 JTextField
year JTextField
month JTextField
day JTextField
startL JLabel
endL JLabel
min1L JLabel
min2L JLabel
min3L JLabel
yearL JLabel
monthL JLabel
dayL JLabel
MenuItem
title String
rating Double
calories int
protein int
fat int
sodium int
price int
BaseProduct
type Type
CompositeProduct
soup MenuItem
steak MenuItem
garnish MenuItem
dessert MenuItem
type Type
LoginView
titleLabel JLabel
typeLabel JLabel
usernameLabel JLabel
passwordLabel JLabel
registerLabel JLabel
comboBox JComboBox
username JTextField
password JPasswordField
loginButton JButton
registerButton JButton
IDeliveryServiceProcessing
DeliveryService
menuItems List<MenuItem>
orders HashMap<Order, List<MenuItem>>
clients List<User>
admins List<User>
employees List<User>
EmployeeView
tableModel DefaultTableModel
table JTable
scroll JScrollPane
orders HashMap<Order, List<MenuItem>>
refresh JButton
confirm JButton
RegisterView
titleLabel JLabel
usernameLabel JLabel
passwordLabel JLabel
username JTextField
password JTextField
createButton JButton
Type
SOUP
STEAK
GARNISH
DESSERT
MENU
Observer
order Order
EmployeeObserver
ClientController
clientView ClientView
deliveryService DeliveryService
cart List<MenuItem>
total int
Order
orderID int
clientID String
date LocalDateTime
delivered boolean
User
role Roles
username String
password String
Roles
ADMIN
CLIENT
EMPLOYEE
ReportsController
reportsView ReportsView
deliveryService DeliveryService
AdminController
adminView AdminView
deliveryService DeliveryService
EmployeeController
employeeView EmployeeView
RegisterController
registerView RegisterView
RegisterCompleted
label JLabel
WrongCredentials
label JLabel
InvalidCredentials
label JLabel
LoginController
loginView LoginView
OrderCreated
label JLabel
CheckCredentials
FileWriterClass
Serializator
Main

4. IMPLEMENTARE

- # Pachetul APP

Conține clasa Main care este responsabilă de crearea aplicației.

- # Pachetul BUSINESS LAYER

Conține clasele BaseProduct, CompositeProduct, DeliveryService, MenuItem, IDeliveryServiceProcessing, Order, Type.

➢ **Clasa BaseProduct**

Se ocupă de crearea și manipularea obiectelor de tip BaseProduct. Produsele de bază pot fi supă, fel principal, garnitură și desert.

➢ **Clasa CompositeProduct**

Se ocupă de crearea și manipularea obiectelor de tip CompositeProduct. Un produs compus este de fapt o uniune de 4 produse de bază, unul de tip supă, unul de tip fel principal, unul de tip garnitură și unul de tip desert.

➢ **Clasa DeliveryService**

Se ocupă de functionalitatea aplicatiei si defineste toate operatiile executate de administrator si de catre client. Aceasta clasa foloseste tehnica Design By Contract. Aceasta clasa isi defineste comportamentul prin implementareainterfetei **IDeliveryServiceProcessing** care defineste comportamentele fiecarui tip de utilizator si ce operatii poate sa realizeze.

Restricțiile acestei clase sunt date sub forma:

- preconditii: reprezinta obligatiile pe care datele de intrare ale unei metode trebuie sa le respecte pentru ca metoda sa functioneze corect
- postconditii: reprezinta garantiile pe care datele de iesire ale unei metode le ofera
- invarianti: reprezinta conditii impuse starilor in care programul se poate afla la un moment dat

- **Clasa MenuItem**

Este o clasa abstracta si are urmatoarele variabile de instanta*: title, rating, fat, calories, sodium, price*, iar fiecare dintre cele doua clase descendente are propria implementare pentru calculul acestor caracteristici ale unui produs. Cu ajutorul acestei tehnici putem sa tratam un grup de obiecte intr-un mod similar cu un singur obiect. Prin folosirea acestei metode declaram o referinta la un MenuItem si instantiem obiectul fie ca un BaseProduct fie ca un CompositeProduct.

- **Clasa IDeliveryServiceProcessing**

Interfață care defineste comportamentele fiecarui tip de utilizator si ce operatii poate sa realizeze.

- **Clasa Order**

Se ocupă de crearea și manipularea obiectelor de tip Order.

- **Clasa Type**

Este o clasă de tip Enum care definește cele 5 tipuri de produse : supă, fel principal, garnitură, desert și meniu.

• Pachetul DATA LAYER

Conține clasele Employee Observer, File Writer, Observer, Roles, Serializator, User.

- **Clasa EmployeeObserver**

Extinde clasa Observer și este responsabilă de crearea observatorilor pentru comenzi noi.

- **Clasa FileWriter**

Este responsabilă pentru scrierea în fișier a datelor.

- **Clasa Observer**

  Este responsabilă de crearea observatorilor pentru comenzi noi.

- **Clasa Roles**

Este o clasă de tip Enum care enumeră cele 3 tipuri de utilizatori : administrator, angajat și client.

- **Clasa Serializator**

Se ocupa cu serializarea datelor aplicatiei, in acest fel mentinandu-se persistenta acestora. Clasele care se serializeaza implementeaza interfata de tip marker Serializable pentru a notifica acest lucru. Informatiile aplicatiei se vor salva in 5 fișiere : unul pentru menținerea datelor referitoare la produsele din meniu, : unul pentru menținerea datelor referitoare la utilizatorii de tip administrator, : unul pentru menținerea datelor referitoare la utilizatorii de tip angajat, : unul pentru menținerea datelor referitoare la utilizatorii de tip client, : unul pentru menținerea datelor referitoare la comenzile efectuate. Serializarea se realizeaza cu ajutorul obiectelor de tipul FileOutputStream/ FileInputStream si ObjectOutputStream/ObjectInputStream.

- **Clasa User**

  Este o clasă care implementează operațiile efectuate asupra utilizatorilor.

• Pachetul PRESENTATION LAYER CONTROLLERS

Conține clasele AdminController, ClientController, EmployeeController, LoginController, RegisterController, ReportsController.

- **Clasa AdminController**

  Este responsabilă de controlul operațiilor din cadrul ferestrei pentru administrator.

- **Clasa ClientController**

  Este responsabilă de controlul operațiilor din cadrul ferestrei pentru client.

- **Clasa EmployeeController**

  Este responsabilă de controlul operațiilor din cadrul ferestrei pentru angajat.

- **Clasa LoginController**

  Este responsabilă de controlul operațiilor din cadrul ferestrei de logare.

- **Clasa RegisterController**

  Este responsabilă de controlul operațiilor din cadrul ferestrei de înregistrare.

- **Clasa ReportsController**

  Este responsabilă de controlul operațiilor din cadrul ferestrei generările de raporturi.

- # Pachetul PRESENTATION LAYER MESSAGE BOXES

  Conține clasele InvalidCredentials, OrderCreated, RegisterCompleted, WrongCredentials.

- **Clasa InvalidCredentials**

  Este responsabilă de crearea ferestrei de eroare pentru acreditări invalide.

- **Clasa OrderCreated**

  Este responsabilă de crearea ferestrei de informare asupra creării comenzii.

- **Clasa RegisterCompleted**

  Este responsabilă de crearea ferestrei pentru înregistrare reușită.

- **Clasa WrongCredentials**

  Este responsabilă de crearea ferestrei pentru logare nereușită.

- **Pachetul PRESENTATION LAYER VIEWS**

Conține clasele AdminView, ClientView, EmployeeView, LoginView, RegisterView, ReportsView.

- **Clasa AdminView**

  Este responsabilă de crearea ferestrei pentru utilizatorul de tip administrator.

- **Clasa ClientView**

  Este responsabilă de crearea ferestrei pentru utilizatorul de tip client.

- **Clasa EmployeeView**

  Este responsabilă de crearea ferestrei pentru utilizatorul de tip angajat.

- **Clasa LoginView**

  Este responsabilă de crearea ferestrei de logare.

- **Clasa RegisterView**

  Este responsabilă de crearea ferestrei de ănregistrare.

5. REZULTATE

Ca și rezultate am obținut o aplicație prin intermediul cărora 3 tipuri de utilizatori (administrator, angajat și client) pot realiza diferite operații în funcție de tipul lor.

Rezultatele se pot observa direct in interfata grafica, iar efectul serializarii/deserializarii este direct vizibil intrucat starea anterioara a sistemului este salvata la o noua logare.

De asemenea prin utilizarea tehnicii Design By Contract pentru clasa DeliveryService este verificata corectitudinea datelor, si nu se permit propagarea unor erori care sa ajunga sa fie serializate. Folosind assert-uri este foarte usor de depistat o problema, si de facut debug.

6. CONCLUZII

În concluzie, această temă m-a ajutat să aprofundez paradigmele Programării Orientate pe Obiect. Astfel, am reușit să îmi dezvolt abilitățile de a lucra cu liste, de a lucra cu interfața grafică De asemenea, consider că am reușit să îmi îmbunătățesc și abilitățile gândirii logice și matematice. Cele mai importante lucruri pe care le-am învățat de la această temă sunt :

- utilizarea colecțiilor de tip MAP
- utilizarea expresiilor de timp lambda
- utilizarea expresiilor de tip stream
- utilizarea serializării obiectelor
- utilizarea observatorilor
- etc

Acest proiect a avut ca și scopuri principale utilizarea colecțiilor de tip map, parcurgerea colecțiilor cu expresii de tip lambda și stream, dar și utilizarea tehnicii serializării obiectelor.

Ca și dezvoltări ulterioare pot fi implementate următoarele :

- adăugarea de imagini cu produsele din meniu
- urmărirea comenzii în timp real pentru utilizatorii de tip client
- discount-uri pentru clienții fideli
- crearea unei aplicații pentru telefonul mobil

7. BIBLIOGRAFIE

- https://www.baeldung.com/
- https://ro.wikipedia.org/wiki/Pagina_principal%C4%83
- https://www.w3schools.com/java/