DOCUMENTATIE

TEMA *4*

NUME STUDENT: Zaharia Robert Jan
GRUPA: 30225

CUPRINS

1. Obiectivul temei

Obiectivul principal al temei este implementarea unei aplicatii pentru gestionarea comenzilor de alimente pentru o firma de catering. Aplicatia trebuie sa asigure functionarea pentru 3 tipuri de utilizatori: Administrator , client , Angajat . De asemenea datele trebuie sa persiste de la o pornire la alta.

Administratorul poate:

• Importați setul inițial de produse care vor popula meniul dintr-un fișier .csv.

• Gestionați produsele din meniu: adăugați/ștergeți/modificați produse și creați produse noi compus din mai multe produse din meniu (un exemplu de produs compus ar putea fi numit „meniu zilnic 1” compus dintr-o supă, o friptură, o garnitură și un desert).

• Generați rapoarte despre comenzile efectuate luând în considerare următoarele criterii:

o intervalul de timp al comenzilor – trebuie generat un raport cu comenzile efectuate între o anumită oră de început și o dată de sfârșit, indiferent de dată.

o produsele comandate de mai mult de un anumit număr de ori până acum.

o clienții care au comandat mai mult de un anumit număr de ori până acum și valoarea comenzii a fost mai mare decât o sumă specificată.

o produsele comandate într-o zi specificată cu numărul de ori pe care le au fost comandat.

Clientul poate:

• Înregistrați-vă și utilizați numele de utilizator și parola înregistrate pentru a vă conecta în sistem.

• Vizualizați lista de produse din meniu.

• Căutați produse pe baza unuia sau mai multor criterii, cum ar fi cuvântul cheie (de exemplu, „supă”), evaluarea, numarul de calorii/proteine/grasimi/sodiu/pret.

• Creați o comandă formată din mai multe produse – pentru fiecare comandă va fi data și ora

a persistat și va fi generată o factură care va enumera produsele comandate și prețul total a ordinului.

Angajatul este anunțat de fiecare dată când o nouă comandă este efectuată de către un client pentru a putea pregăti livrarea preparatelor comandate.

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

Caz de utilizare: adăugați produs 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 sunt detaliile produsului ar trebui introdus

3 . Administratorul introduce numele produsului, numărul de calorii, proteine, grăsimi, sodiu și prețul acestuia

4 . Administratorul face clic pe butonul „Adăugați”.

5 . Aplicația salvează datele produsului și afișează un confirmarea mesajului

Caz de utilizare : adaugam o comanda

Actor principal : client

Pricipalul scenariu de succes:

1. Clientul selecteaza optiunea de a adauga o comanda noua
2. Aplicatia va afisa un tabel cu toate produsele disponibile pentru o comanda
3. Clientul va adauga in cos produsele dorite
4. Clientul va apasa pe butonul de achitare a comenzii
5. Aplicatia va genera un bon pentru comanda achitata

Caz de utilizare: Creeam un raport al comenzilor

Actor principal : Administrator

Principalul scenariu de succes:

1. Administratorul selecteaza optiunea de a genera rapoartele
2. Va introduce datele necesare unuia dintre cele 4 tipuri de rapoarte
3. Se va generea raportul intr-unul dintre fisierele : Raport1 .txt , Raport2 .txt , Raport3 .txt , Raport4 .txt .

Tabel Hash

Structura de date de rezervă pentru HashSet, LinkedHashSet, HashMap, HashTable, LinkedHashMap etc.

Folosit pentru implementarea unui tablou asociativ (prin maparea cheilor la valori) cu timp de acces constant la acesta elemente

• Timp de acces constant => fără structuri repetitive => acces direct la memorie • Cheile vor fi folosite ca indici într-un tablou: stocați perechea (cheie, valoare) ca găleată[cheie]=valoare • Elementele matricei se numesc găleți

• Problema acestei abordări este memoria mare alocată și neutilizată dacă setul de chei este rar => Soluție: definiți o funcție hash pentru a reduce setul de chei la un set mai mic de dimensiune N hash : chei -> {1..N}

• Perechea (cheie, valoare) va fi stocată ca: găleată[hash(cheie)] = valoare

Java oferă un mecanism, numit serializare obiect, în care un obiect poate fi reprezentat ca o secvență de octeți care include datele obiectului, precum și informații despre tipul obiectului și tipurile de date stocate în obiect.

După ce un obiect serializat a fost scris într-un fișier, acesta poate fi citit din fișier și deserializat, adică informațiile de tip și octeții care reprezintă obiectul și datele acestuia pot fi utilizați pentru a recrea obiectul în memorie.

Cel mai impresionant este faptul că întregul proces este independent de JVM, ceea ce înseamnă că un obiect poate fi serializat pe o platformă și deserializat pe o platformă complet diferită.

3. Proiectare

Aplicatia este realizata din 18 clase impartita in 3 pachete : View , Data , Business . Pachetul View contine clasele : Admin_view care realizeaza meniul principal al utilizatorului administrator , Admin1 realizeaza interfata grafica pentru adaugarea produselor simple in meniu , Admin2 care realizeaza interfata grafica pentru adaugarea produselor compuse in meniu , ClientView care realizeaza meniul principal al utilizatorului client , DeliveryService_view , Employee view realizeaza interfata grafica pentru angajat , Login_view care realizeaza interfata grafica pentru inregistrarea unor noi clienti si intrarea in aplicatie , My order care realizeaza interfata grafica pentru afisarea comenzilor clientului curent , Order all care realizeaza interfata grafica pentru afisarea tuturor comenzilor si generarea rapoartelor pentru comenzi . Acest pachet reprezinta prim planul aplicatiei fiind format din interfatele grafice pentru cei 3 utilizatori. Pachetul Data este format din clasele : Order , BaseProduct , DeliveryService , Employee , MenuItem , CompositeProduct , Login si pachetul Business contine clasa IdeliveryServiceProcessing .

Metoda cea mai importanta este cea de Deserializare / Serializare prin intermediul careia se pastreaza datele de la iteratiile anterioare ale aplicatiei fiind o alternativa eficienta pentru o baza de date :

```
public Map < Order , ArrayList < MenuItem > > Deserializare()
{
    Map < Order , ArrayList < MenuItem > > h = new HashMap< > () ;
    try {
        FileInputStream fileIn = new
FileInputStream("C:\\Users\\HP\\Desktop\\TP teme\\Tema4_tp\\Delivery1.ser");
        ObjectInputStream in = new ObjectInputStream( fileIn ) ;

        h= ( Map < Order , ArrayList < MenuItem > > ) in.readObject();
        in.close();
        fileIn.close();

    } catch ( ClassNotFoundException e) {
        e.printStackTrace();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return h;
}
```

4. Implementare

Aplicatia este realizata din 18 clase impartita in 3 pachete : View , Data , Business . Pachetul View contine clasele : Admin_view care realizeaza meniul principal al utilizatorului administrator , Admin1 realizeaza interfata grafica pentru adaugarea produselor simple in meniu , Admin2 care realizeaza interfata grafica pentru adaugarea produselor compuse in meniu , ClientView care realizeaza meniul principal al utilizatorului client , DeliveryService_view , Employee view realizeaza interfata grafica pentru angajat , Login_view care realizeaza interfata grafica pentru inregistrarea unor noi clienti si intrarea in aplicatie , My order care realizeaza interfata grafica pentru afisarea comenzilor clientului curent , Order all care realizeaza interfata grafica pentru afisarea tuturor comenzilor si generarea rapoartelor pentru comenzi .

Aplicatia incepe cu clasa Login_view care Initializeaza DeliveryService-ul prin recuperarea datelor din fisierul Delivery.ser si lista de clienti din fisierul Clients.ser prin procesul de serializare . Apoi in functie de tipul de utilizator se vor deschide meniurile speciale fiecarui tip . In cazut administratorului vom deschide Clasa admin_view .

Pachetul Data contine clasa Order pentru Comenzi care are ca parametri : private int Orderid ; private int Clientid , private String OrderDate , private int hour si care contine metodele Hashcode si equals pentru Hashmap-ul din delivery Service. Clasa Client contine parametrii clientilor : private String username , private String password , private String telefon , private int id . Clasa MenuItem contine parametrii pentru produsele din meniu : private String title , private double Rating , private int Calories , private int protein , private int fat , private int sodium , private int Price . Clasele BaseProduct si Composite Product sunt realizate prin procesul de mostenire avand drept clasa parinte clasa MenuItem. Clasa Composite Product fiind formata dintr-un ArrayList de MenuItems . Clasa Login contine informatiile despre conturile de utilizator : private String username , private String parola , private int id . Clasa DeliveryService este clasa de baza a programului ea fiind principala unitate de stocare a informatiilor din aplicatie . Aceasta are ca si parametrii public Map< Order , ArrayList < MenuItem > > table = new HashMap () , public ArrayList < MenuItem > meniu = new ArrayList<>() , public ArrayList < Client > clients = new ArrayList <> () , public List < Observer > obs = new ArrayList <> () . Aceasta clasa are multiple metode importante AddOrder(Order o , ArrayList < MenuItem > menu) care adauga o noua comanda la hashmapul de comenzi . Metoda Search (String nume , double rating , int calories , int protein , int fat , int sodium , int price) realizeaza sortarea hashmapului in functie de unul sau mai multi parametrii prin intermediului Streamurilor si expresiilor Lambda . Metoda verifica daca functia sa foloseasca parametrii prin compararea parametrilor intregi cu - 2 si titlul cu null .

5. Rezultate

In cadrul aplicatiei multe afisari sunt realizate in fisiere exterioare . Fisierul pentru bonul fiscal

Order { Orderid = 438 , Clientid = 1 , Order Date = ' 19 / 03 / 2001 ' , hour = 18 }
BaseProduct @ 4a39a5c9 BaseProduct @ 4297d5 e BaseProduct @ 5dcc829b BaseProduct @ 1819a5c6
Price : 212
Exemplu pentru primul raport :
Order { Orderid = 56 , Clientid = 1 , OrderDate = ' 20 / 02 / 2001 ' , hour = 11 }
Order { Orderi d= 173 , Clientid = 2 , OrderDate = '19/04/2019 ' , hour = 9 }
Order { Orderid = 117 , Clientid = 4 , OrderDate = '19/03/2001 ' , hour = 7 }
Exemplu pentru al doilea raport :
Smoked Caviar and Hummus on Pita Toasts
Barbecued Shrimp
The Original Three-Ingredient Rub
Shrimp Sates with Spiced Pistachio Chutney
"Endive ""Spoons"" with Lemon-Herb Goat Cheese "
Toasted Corn Crisps
Fish Stock
Exemplu pentru al treilea raport :
Clientul 2
Clientul 4
Exemplu pentru al patrulea raport :
Barbecued Shrimp de 1
Red and Green Tomato Salsa de 1
"Endive ""Spoons"" with Lemon-Herb Goat Cheese " de 1
Toasted Corn Crisps de 1
Fish Stock de 1
Quatre-Épices de 1

6. Concluzii

Rezultatul codului este o aplicatie pentru gestionarea comenzilor de alimente pentru o firma de catering . Aceasta tema implica folosirea unui nou tip de colectie in care se pot pastra datele „ HashMap „ cat si stocarea un nou mod de a filtra o colectie de date prin intermediului Streamurilor si expresiilor lambda . De asemenea aplicatia ofera sansa folosirii Procesului de serializare pentru a pastra datele aplicatiei de la o iteratie la alta

7. Bibliografie

https://www.geeksforgeeks.org/serialization-in-java/

https://www.tutorialspoint.com/java/java_serialization.htm