Logo, company name

Description automatically generated

Tema 4

Food Delivery Management System

*Tehnici de programare*

*Facultatea de Automatică și Calculatoare*

*Calculatoare și Tehnologia Informației*

Profesor coordonator Student

Dorin Vasile Moldovan Lupulescu Teodor-Damian

Grupa 30226

Cuprins:

1. Obiectivul temei: pagina 3.

2. Analiza Problemei: pagina 3 – pagina 4.

3. Proiectare: pagina 4 – pagina 5.

4. Implementare: pagina 5 – pagina 14.

5. Rezultate: pagina 14 – pagina 16.

6. Concluzii: pagina 16 – pagina 17.

7. Bibliografie: pagina 17.

1. Obiectivul temei:

Proiectarea unui program care se ocupa de gestionarea unui presupus restaurant care ofera mai multe sortimente de produse. Exista 3 tipuri de utilizatori. Administratorii, care pot importa produsele dintr-un fisier, pot adauga manual produse, pot modifica sau sterge produse, si pot crea produse noi prin combinarea produselor deja existente si pot genera rapoarte. Clienti, care pot sa caute produse dupa toate criteriile posibile si pot creea o comanda cu toate produsele pe care le selecteaza. Angajati normali(Employees) care sunt notificati mereu cand un client plaseaza o comanda si afiseaza comenzile plasate.

Obiectivul este de a manipula corect si eficient lista de produse si de a plasa comenzi din postura de client

Ca obiective secundare pentru a îndeplini obiectivul principal avem:

* Transformarea datelor de intrare introduse de către utilizator în date de program care pot fi folosite în efectuarea operațiilor
* Validarea datelor pentru fiecare utilizator si deschiderea ferestrei corespunzatoare
  + Filtrarea produselor din meniu astfel incat sa generam rapoarte utile si sa putem cauta produse dupa multe criterii
  + Salvarea starii curente a programului astfel incat sa nu fie nevoie importarea de produse de fiecare data cand este pornit
* Crearea unei chitante pentru fiecare comanda noua.

1. Analiza problemei:

Scenariul este următorul:

Se deschide o fereastra de Login. Aici se pot conecta 3 cele 3 tipuri de utilizatori Administrator, Client si Employee. Dupa ce se introduce datele si sunt verificate sa corespunda cu utilizatorii programului, se deschide fereastra corespunzatoare tipului de utilizator care s-a inregistrat unde fiecare poate sa foloseasca programul in modul in care are nevoie.

Diagrama Use-Case

|  |  |
| --- | --- |
| Caz de utilizare | Funcționare |
| Cazul normal | 1. Utilizatorul introduce datele. 2. Utilizatorul apasa butonul de Login si se autentifica in program. 3. Se introduc date necesare efectuarii operatiilor specifice fiecarui tip de utilizator 4. Se vor efectua operatiile cerute si se vor afisa informatiile utile pentru folosirea corecta a programului. |
| Cazul excepțional | Daca se incearca autentificarea unui utilizator care nu este inregistrat in programul, programul va afisa un mesaj de eroare.  Acelasi lucru se intampla si daca se cere programului sa execute o operatiune fara a completa toate field-urile necesare, exceptie fac field-urile pentru filtrarea produselor care se pot lasa goale in cazul in care nu se doresc acele criterii de filtrare. |

1. Proiectare:
2. Prezentarea scurta a claselor:

Pentru a implementa un astfel de program am folosit 3 clase pentru interfata grafica a programului, o clasa pentru salvarea starii programului, o interfata si o clasa care o implementeaza care acopera majoritatea logicii de functionare a programului. O interfata si o clasa Observable pentru functionalitatea de observer. Trei clase model pentru gestionarea tipurilor de utilizatori si o clasa albstracta de utilizator care este mostenita de celelalte 3. Doua clase care genereaza un id unic pentru comenzi si pentru clienti. O interfata si doua clase care o implementeaza pentru modelarea produselor presupusului restaurant. Si in final o clasa care implementeaza ce se petrece atunci cand se inchide fereastra de admin pentru a salva starea curenta a programului.

1. Diagrama UML de clase:

Graphical user interface, diagram

Description automatically generated

1. Structuri de date și algoritmi:

Pentru implementarea acestui program am folosit foarte des colectii cum ar fi liste, Set-uri sau Map-uri pentru a retine produsele si comenzile inregistrare de program. Pentru filtrarea datelor in functie de datele introduse de utilizator s-au folosit in loc de parcurgeri normale ale colectiilor, stream-uri. Pentru modelarea claselor BaseProduct am folosit Composite Design Pattern, iar pentru notificarea unui employee atunci cand o comanda este plasata am folosit Observer Design Pattern.

1. Implementare:
2. Clase si Interfete:
3. Clasa Observable:

Clasa Observable este responsabila de a anunta/notifica toti observatorii in privinta faptului ca un eveniment a avut loc si este nevoie sa se execute un anumit lucru. Aceasta retine un HashSet de obsevatori si are doar metodele de a adauga sau sterge observatori si metoda de notificare a tuturor observatorilor.

1. Interfata Observer:

Interfata Observer are rolul de a fi implementata de toate clasele care trebuie sa efectueze un anumit lucru in momentul in care o clasa Observable trimite notificarea. Interfata aceasta are doar metoda de update pentru executarea operatiilor necesare.

1. Interfata MenuItem:

Interfata MenuItem reprezinta un produs din meniu. Acesta poate fi un produs normal standard sau poate fi un produs compus din mai multe produse standard. Aceasta interfata contine doar metode de tip getters si setters si o suprascrierea a metodei toString si este implementata de clasele BaseProduct si CompositeProduct.

1. Clasa BaseProduct:

Clasa BaseProduct este clasa care reprezinta un produs standard, simplu, din meniul restaurantului. Clasa implementeaza interfata MenuItem. Aici regasim metode de tip getters si setters, metoda toString suprascrisa si metoda de hashCode pentru realizarea hashMap ului la stocarea produselor. Clasa BaseProduct are ca si campuri informatiile esentiale pentru descrierea unui produs.

1. Clasa CompositeProduct:

Clasa CompositeProduct este clasa care reprezinta un produs compus din mai multe produse standard simple. La fel ca si clasa de mai devreme implementeaza interfata MenuItem si are doar metode de tip getters, setters, metoda toString suprascrisa si metoda de hashCode pentru realizarea hashMapului in stocarea produselor. Clasa CompositeProduct are ca si campuri informatiile esentiale pentru descrierea unui produs.

1. Clasa User:

Clasa User este clasa care reprezinta un user care incearca sa se autentifice in program. Acesta poate fi de tipul administrator, employee sau client. De aceea este o clasa abstracta care este mostenita de clasele Administrator, Employee si Client. Clasa User are ca si campuri un username, o parola, si un tip pentru a identifica ce utilizator incearca sa se autentifice. Metodele implementate sunt getters setters si suprascrierea metodei toString.

1. Clasa Employee:

Clasa Employee este clasa care reprezinta un user de tip employee si mosteneste clasa User. Aceasta clasa contine doar un id, un rol si un nume, si metodele de tip getters si setters.

1. Clasa Client:

Clasa Client este clasa care reprezinta un user de tip client si mosteneste clasa User. Aceasta clasa contine doar un id, un nume, si metodele de tip getters si setters.

1. Clasa Administrator:

Clasa Administrator este clasa care reprezinta un user de tip administrator si mosteneste clasa User. Aceasta clasa contine doar un id, un nume, si metodele de tip getters si setters.

1. ClientIdGenerator:

Clasa ClientIdGenerator este o clasa care genereaza un id unic pentru fiecare client inregistrat. Aceasta contine doar un camp id pe care il incrementeaza de fiecare data cand trebuie sa returneze un id pentru un client.

1. Interfata IDeliveryService:

Interfata IdeliveryService este interfata implementata de clasa DeliveryService. Interfata contine declararea metodelor folosite pentru a implementa functionalitatea programului. Metodele declarate sunt metode care modeleaza produsele cum ar fi cele de add, delete, edit sau import products, metode de generare a rapoartelor pentru administrator, o metoda pentru crearea de chitante pentru comenzi si o metoda pentru cautarea anumitor produse in functie de mai multe criterii.

1. Clasa DeliveryService:

Clasa DeliveryService implementeaza interfata IdeliveryService si este clasa care implementeaza puternic logica de functionare a acestui program. Campurile retinute de aceasta clasa sunt un HashSet pentru a retine un meniu adica o colectie de produse, un HashMap pentru a retine o lista de produse specifica unei anumite comenzi si un generator pentru id-ul comenzilor. Aici este implementata metoda getProducts care se ocupa de importarea produselor din fisierul .csv in lista de produse a programului. Mai departe se regasesc metodele pentru gestionarea meniului si anume metodele prin care se adauga, se modifica sau se sterge un produs. Aici regasim si metoda care creaza un obiect de tip order cu ajutorul datelor primite de la interfata grafica a clientului. Urmeaza implementarea metodei de cautare filtrata a produselor care cauta in meniu si filtreaza folosind stream-uri elementele care indeplinesc anumite conditii. Tot folosind stream-uri se face si implementarea metodelor pentru generarea rapoartelor pentru administrator, iar pe langa acestea au mai fost implementate metode ajutatoare printre care si metoda de removeNulls care are rolul de a indeparta elementele nule din lista de produse in cazul in care acestea sunt adaugate din greseala.

1. Clasa LoginService:

Clasa LoginService este clasa care se ocupa de verificarea datelor de autentificare primite de la interfata grafica. Citind din fisier folosind metoda readFromFile aceasta clasa memoreaza un HashSet cu toti utilizatorii iar la apelul metodei checkLogin aceasta clasa verifica ce tip de utilizator a incercat sa se autentifice si daca datele introduse sunt valide returneaza un obiect de tipul utilizatorului introdus.

1. Clasa Order:

Clasa Order este clasa care reprezeinta o comanda trimisa de catre un client. Campurile retinute sunt id-ul comenzii, id-ul clientului, produsele comandate, data si pretul total al comenzii. Clasa aceasta suprascrie metodele toString si hashCode si implementeaza metode de tip setters si getters.

1. Clasa OrderIdGenerator:

Clasa OrderIdGenerator este clasa responsabila cu generarea de id-uri unice pentru fiecare comanda si este foarte similara cu clasa ClientIdGenerator care face aproape acelasi lucru.

1. Clasa SerializeIt:

Clasa SerializeIt este clasa responsabila de salvarea unei copii a obiectului DeliveryService la momentul inchiderii programului pentru a o putea incarca la urmatoarea pornire a programului si a avea datele precedente deja incarcate. Contine doar doua metode, prima fiind pentru incarcarea obiectului copiat iar cea de-a doua pentru salvarea obiectului.

1. Clasa AdminView:

Clasa AdminView este de fapt interfata grafica pentru administrator. Aici administratorul poate adauga, modifica, sterge sau compune un produs. Tot aici se pot genera si cele 4 rapoarte specifice administratorului. Clasa contine declararea multor componente ale interfetei grafice si implementarea metodelor pentru validarea datelor de intrare si prelucrarea datelor primite de la DeliveryService pentru a putea fi afisate inapoi in interfata.

1. Clasa EmployeeView:

Clasa EmployeeView este interfata grafica pentru un employee. Aici sunt afisate comenzile trimise de catre client.

1. Clasa ClientView:

Clasa ClientView este interfata grafica pentru un client. Cu aceasta interfata grafica un client poate cauta produse care sa indeplineasca anumite conditii si poate adauga produsele acelea la o comanda pe care o poate plasa ulterior.

1. Clasa LoginView:

Clasa LoginView este interfata grafica in care se face autentificarea unui utilizator. Aici dupa introducerea datelor si apasarea butonuli de Login, interfata va folosi clasa LoginService si va verifica daca s-au introdus datele unui administrator, client sau employee valid iar apoi deschide fereastra corespunzatoare utilizatorului autentificat.

1. Clasa Main:

Clasa Main este clasa principala care pune in functiune programul, aici se creeaza un obiect de tipul deliveryService care primeste obiectul vechi salvat la ultima rulare a programului. Iar apoi se instantiaza si fereastra de Login.

1. Clasa WindowClosing:

Clasa WindowClosing joaca rolul de ActionListener pentru inchiderea ferestrei administratorului. Astfel la inchiderea acestei ferestre se salveaza starea curenta a obiectului deliveryService pentru a putea fi incarcat la urmatoarea rulare a programului.

1. Interfața Grafică:

Graphical user interface, application

Description automatically generated

In aceasta fereastra utilizatorul poate sa introduca datele unui User nou si sa se autentifice.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Aceasta este fereastra administratorului de unde poate sa introduca, sa stearga sa modifice sau sa creeze produse noi, si mai poate de asemenea sa genereze o serie de rapoarte.

Graphical user interface, text, application, email

Description automatically generated

Aceasta este fereastra clientului unde se pot cauta produse dupa anumite criterii iar apoi adauga la o comanda.

In cazul in care programul este folosit necorespunzator se intampla urmatoarele lucruri:

Graphical user interface, application

Description automatically generated

Aici s-a incercat introducerea unor date care nu apartineau niciunui user si s-a afisat un mesaj de eroare.

Graphical user interface

Description automatically generated

Aici s-a incercat adaugarea unui nou produs fara a introduce ceva in campurile destinate caracteristicilor produsului si s-a afisat un mesaj de eroare.

Graphical user interface

Description automatically generated

Aici s-a incercat generarea unui raport fara a introduce date si s-a afisat un mesaj de eroare.

1. Rezultate:

Aici vor fi prezentate câteva capturi de ecran care ilustrează rezultatele programului:

Text

Description automatically generated

Aici avem o chitanta produsa la crearea unei comenzi unde se poate vedea data si ora la care s-a efectuat comanda, id-ul clientului si pretul total

Graphical user interface, text, application

Description automatically generated

Aici avem zona in care se poate creea un nou produs cu produsele deja existente.

1. Concluzii:

Dezvoltarea acestei teme m-a ajutat sa invat sa implementez interfete grafice putin mai bine organizate si sa invat sa folosesc stream-uri in java.

De asemenea a ajutat și la dezvoltarea personala în domeniul programării orientate pe obiecte și a fost un exercitiu foarte bun pentru a mentine in forma abilitatile de a scrie o documentatie.

Idei de imbunatatire a programului sunt numeroase deci voi enumera doar cateva dintre ele:

* Posibilitatea de a inregistra noi utilizatori folosind interfata grafica
* Posibilitatea de a scoate pe rand produse din comanda atunci cand un client alcatuieste o comanda
* O interfata grafica mai stilizata si mai ordonata.

1. Bibliografie:

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

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

<http://javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.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://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/>

https://www.baeldung.com/java-serialization