DOCUMENTATIE

TEMA *4*

NUME STUDENT: Bodogae George Stefan
GRUPA: 30424

CUPRINS

1. Assignment’s objective

This assignment’s purpose was creating a food delivery application that is usable and simulates the management system of food delivery company. Things like orders, clients, the date and time of orders and reports regarding the orders can be viewed within the app. I designed this application with ease of use in mind, each type of user: admin, employee, client having a separate window to work with. The main purpose of the application is to optimize the process of ordering and delivering food, making it easier for both the clients and the employees. The administrator has access to important data about the clients and orders which keeps him/her updated on the most ordered items and the clients that order the most from the application.

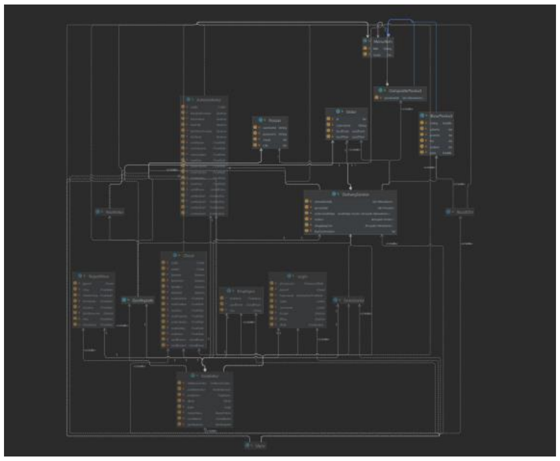
2. Analisys, use cases

The food delivery application has 3 types of users: administrator ,client and employee. The admin can add, delete and edit new products and make composite products of the existing products. This user can also generate reports based on criteria selected in a separate window regarding the orders that have been made. The Employee can view the onging orders in real time in a window as well as who, how many products and the exact date and time the each order has been made. The client can view and add to a shopping cart the desired products and filter them based on title, price, fats, sodium, calories, proteins and rating. The cient can make orders with multiple products. The Employee window starts at the same time with the client window so that when an order is being placed, the employee is notifyed imidiately. In this way, the employee can start and process the order right away instead of wainting for the order to be received.

3. Design

I used a layered architecture for organizing the classes in packages. The application has three main layers: Bussiness layer, DataLayer and PresentationLayer. The Bussiness layer contains the classes: BaseProduct, CompositeProduct, DeliveryService, GenReports, IDeliveryServiceProcessing, MenuItem, Order and Person. The DataLayer contains the classes: FileWriter, ReadCSV and Serializator. The PresenationLayer contains the classes: Administrator, Client, Controller, Employee, Login and ReportView. The layered architecture is beneficial for the programming process because it simplifyes the work flow by having the classes sorted at all times and being able to draw a diagram that represents the starting point of the application.

The image below represents the UML class diagram o the application. It is mandatory to design a class diagram for better understanging the design process and be able to design in the most simplistic and usable way the application.



4. Implementare

The application has three main layers: Bussiness layer, DataLayer and PresentationLayer. The Bussiness layer contains the classes: BaseProduct, CompositeProduct, DeliveryService, GenReports, IDeliveryServiceProcessing, MenuItem, Order and Person. The DataLayer contains the classes: FileWriter, ReadCSV and Serializator. The PresenationLayer contains the classes: Administrator, Client, Controller, Employee, Login and ReportView.

1.BussinessLayer

- The MenuItem class defines a product. It has fields title, count (which calculates the no. of times a product has been ordered) and is extended by BaseProduct and CompositeProduct classes. This class contains overrriden methods from BaseProduct and composite product which are mainly getters and setters.
- The BaseProduct class extends MenuItem and defines a simple product with the fields: title, rating, calories, protein, fat, sodium and price. It contains basic getters and setters which return or set and object of type BaseProduct's fields.
- The CompositeProduct class extends the MenuItem class and contains modified getters and setters such that in case of a composite product that is formed by multiple base products, the fields are calculated acordingly. For example, for calculating the rating we apply the average formula:

```
@Override
public double calculateRating() {
    double avgRating=0;
    for(MenuItem menuItem : productList){
        avgRating += menuItem.calculateRating();
    }
    avgRating/=productList.size();
    return avgRating;
}
```

  For calculating the price we just add all the base products' price to the total price:

```
@Override
public double calculatePrice() {
    double totalPrice=0;
    for(MenuItem menuItem : productList){
        totalPrice += menuItem.calculatePrice();
    }
    return totalPrice;
}
```

- The Order class extends Observable and implements Serializable contains all the fields that define an order such as: id, cUsername, localDate, localTime. The class contains basic getters and setters for all the fields and also custom getters for the localDate and localTime which return and int value that represents the date as an integer and the hour as an integer:

```
public int getHour(){return localTime.getHour();}
public int getDay(){return localDate.getDayOfMonth();}
```

- The Person class defines a person with all the required fields: username, password, count and role. The class defines getter and setter type methods for all the fields and is used to define a Person type object which can take 3 roles: Admin, Client Or Employee based on what the uses chooses in the Login window.
- The GenReports class is used to generate reports containing data about the orders that have been made. The reports generated by this class are the following:
    - time interval of the orders – the orders performed between a given start hour and a given end hour regardless the date.
    - the products ordered more than a specified number of times so far.
    - the clients that have ordered more than a specified number of times so far and the value of the order was higher than a specified amount.
    - the products ordered within a specified day with the number of times they have been ordered.

  The methods from this class use lamda expressions for filtering the lists of orders, clients and items like so:

```
public void report1(int H1, int H2,DeliveryService deliveryService)
{
    ArrayList<Order> orders;
    orders= (ArrayList<Order>)
deliveryService.getOrders().stream().filter(h1 -> {return
h1.getHour()>= H1;}).filter(h2 -> {return h2.getHour() <=
H2;}).collect(toList());
    FileWriter.raport1(orders);
}
```

- The DeliveryService class is used to basically define the bussiness model. This class extends Observable and Serializable. This class contains the following fields: menuItemList, personList, orderHashMap, orders, shoppingCart, lastOrderIndex. The lastOrderIndex is used to number the orders such that when a client makes an order, the order number is incremented. This class contains getters and setter as well as multiple custom methods. The getPersonByUsername method is used for searching for a person in the personList. The deleteMenuItem Method is used for deleting an item from the MenuItemList. The addProduct method is used for adding a new product by the administrator. The addCompositeProduct method is used to add a new composite product that contains multiple base products by the admin. The modifyProduct method is used for editing a product by the administrator and changing all its fields. The addOrder method is used for adding a new product to the shopping cart by the client. The generateBill method is used for creating a text file that contains a bill when a new order is placed. The createOrder method is used for creating a new order by a client containing all the items form the shopping cart.

2.DataLayer

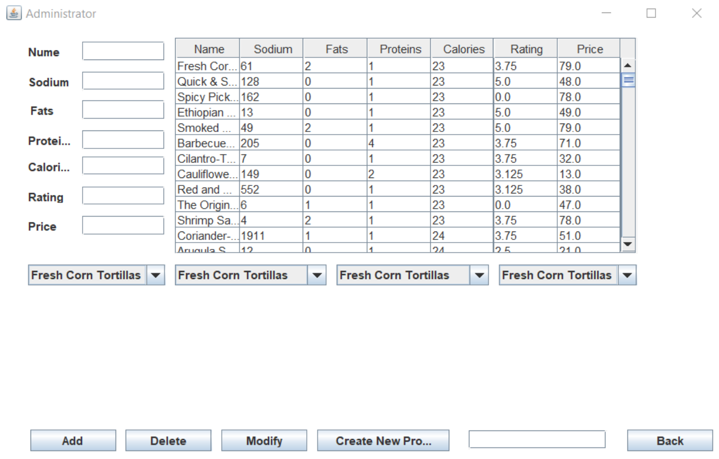
- The FileWriter class is used for creating text files that contain the reports generated by the administrator.

```
ex: public static void raport1(List<Order> orderList)
{
    try(BufferedWriter scrieFisier = new BufferedWriter(new
java.io.FileWriter("report1.txt"))){
        for(Order order:orderList)
        {
            scrieFisier.write("Order no: "+order.getId() +
"\nClient: " + order.getcUsername() + "\nData: " +
order.getLocalDate() +"\nTime: "+order.getLocalTime()+ "\n");
        }
        System.out.println("Report1 done");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

- The ReadCSV method is used for reading a CSV file. It contains the method read that reads all the items from the products.csv, fills a list of MenuItem with them and returns a List<MenuItem>
- The Serializator class is used for serializing and deserializing the DeliveryService type object defined in the Controller class. The serialize method is used to serialize a DeliveryService type object and store it in the file.ser file. The deserialize method deserializes the DeliveryService type object from the file.ser file.

3.PresentationLayer

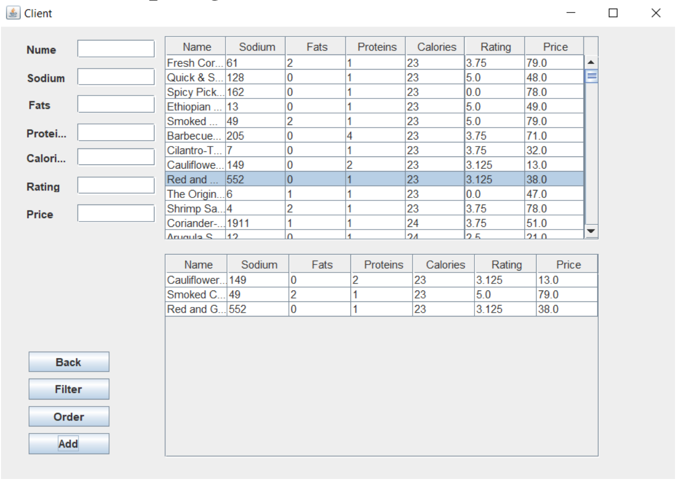
- The Administrator class is used to create the window for the administrator. It contains all the definitions for the buttons, labels and fields. The Administrator can select an item from the table and delete it, fill the fields required for a product and edit an existing one, adding a new product and creating a composite product made of 4 base products and setting a name for it.



Name	Sodium	Fats	Proteins	Calories	Rating	Price
Fresh Cor...	61	2	1	23	3.75	79.0
Quick & S...	128	0	1	23	5.0	48.0
Spicy Pick...	162	0	1	23	0.0	78.0
Ethiopian ...	13	0	1	23	5.0	49.0
Smoked ...	49	2	1	23	5.0	79.0
Barbecue...	205	0	4	23	3.75	71.0
Cilantro-T...	7	0	1	23	3.75	32.0
Cauliflowe...	149	0	2	23	3.125	13.0
Red and ...	552	0	1	23	3.125	38.0
The Origin...	6	1	1	23	0.0	47.0
Shrimp Sa...	4	2	1	23	3.75	78.0
Coriander-...	1911	1	1	24	3.75	51.0
Arugula S...	12	0	1	24	2.5	21.0

The Administrator window

- The Client class is used to create the window for the client. It contains all the definitions for the buttons, labels and fields. The client can select an item from the table and add it to the shopping cart represented by a table below the product table. The client can make an order with 1 or more items in the cart. The client can also filter the products in the product table based on title, sodium, fats, protein, calories, rating and price.

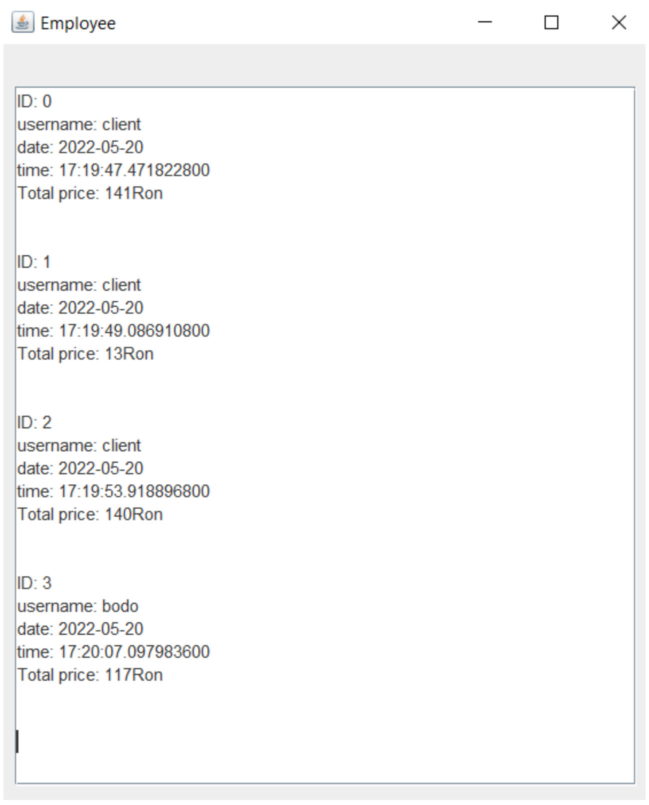


Name	Sodium	Fats	Proteins	Calories	Rating	Price
Fresh Cor...	61	2	1	23	3.75	79.0
Quick & S...	128	0	1	23	5.0	48.0
Spicy Pick...	162	0	1	23	0.0	78.0
Ethiopian ...	13	0	1	23	5.0	49.0
Smoked ...	49	2	1	23	5.0	79.0
Barbecue...	205	0	4	23	3.75	71.0
Cilantro-T...	7	0	1	23	3.75	32.0
Cauliflowe...	149	0	2	23	3.125	13.0
Red and ...	552	0	1	23	3.125	38.0
The Origin...	6	1	1	23	0.0	47.0
Shrimp Sa...	4	2	1	23	3.75	78.0
Coriander-...	1911	1	1	24	3.75	51.0
Arugula S...	12	0	1	24	2.5	21.0

Name	Sodium	Fats	Proteins	Calories	Rating	Price
Cauliflower...	149	0	2	23	3.125	13.0
Smoked C...	49	2	1	23	5.0	79.0
Red and G...	552	0	1	23	3.125	38.0

Administrator Window

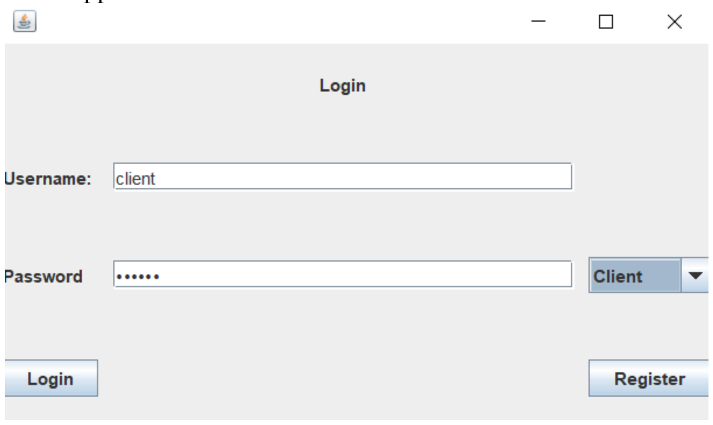
- The Employee class implements Observer and is used to create the window for the employee. The Employee window contains a text field which is used for viewing all the orders that are being processed in real time



```
ID: 0
username: client
date: 2022-05-20
time: 17:19:47.471822800
Total price: 141Ron


ID: 1
username: client
date: 2022-05-20
time: 17:19:49.086910800
Total price: 13Ron


ID: 2
username: client
date: 2022-05-20
time: 17:19:53.918896800
Total price: 140Ron


ID: 3
username: bodo
date: 2022-05-20
time: 17:20:07.097983600
Total price: 117Ron
```

The employee window

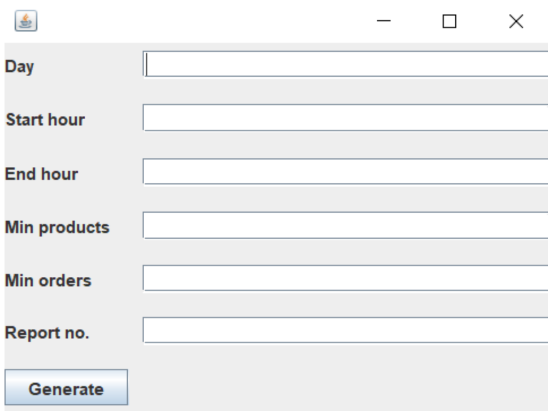
- The Login Class is used for creating the login window. It contains 2 formattedTextField fields in which the user can input an username and a password and also select the type of user(Administrator, Client or Employee) from the comboBod and click Register for creating a new account or login to log in the application.



Username: client
Password: ••••••
Client

Login window

- The ReportView class is used for creating the window in which the admin can generate reports. It contains 6 fields in which the admin can enter a Day, a start hour, an end hour, a no. of products, a no. of orders and a no. for the report and then click Generate for creating the text files that contain the reports.



Generate reports window

- The Controller class is used for controlling the whole application. The controller class has the following fields: DeliveryService deliveryService, Administrator administrator, Employee employee, Client client, Login login, ReportView reportView, Serializator serializator and GenReports genReports. This class contains the listeners for all the buttons in the application and implement their functionality. This class contains methods for: Adding products, deleting products, adding composite products, modifying products, implementing the login system, registering a new user, the back button for all the windows, adding a new item in the shopping cart, creating a new order, filtering products and generating the reports by the administrator.

5. Concluzii

The final product is an usable application which can be implemented by a delivery bussiness. Almost all the functionality from the requirements has been implemented, although there are a few bugs. This assignment was really beneficial for learning about new concepts like serialization, lamda expressions and working with different kinds of files. I can see this application being developed into a real life usable app which should be used by a delivery complany, it will make their work easier and optimize the delivery service by automatically processing data about the orders and clients. This app is also useful for making charts and sections with most ordered items from the menu and also giving coupons to the clients that order the most. Further developments can be done to this application by making a version for the mobile that the clients download through an application store on their smartphones. Another update could be integrating the application with waiting times for every item on the menu such that the client will know when to expect the delivery and also a gps support and a window for the clients to track their delivery in real time.

6. Bibliografie

Programming Techniques (dsrl.eu) – learning about the concepts used in this project

Filtering - Lambda Expressions In Java 8: Tutorial 3 - YouTube – filtering with lamda expressions

java - read csv file and filter specific columns in Lambda (Java8) - Stack Overflow – reading from a csv file

Java - Serialization (tutorialspoint.com) – Learning about serialization and how to implement it

Layered.pdf (uwaterloo.ca) – about layered architecture

Java Create and Write To Files (w3schools.com) – learning about file creating and writing in java

The Observer Pattern in Java | Baeldung – used for learning about the observer in java