DOCUMENTATION

Assignment nr 4

Dora Cristian Adrian

30422

Contents

1. Objectives

1.1. Main Objectives

The main objective for this project is to design and implement an application for managing the food orders for a catering company. It should be able to receive orders from clients. They will also be able to filter the items from the menu. For all of that, login and register pages will be created. If admin logs in or registers, a separate window will be displayed with operations available only for the administrator like adding a product or a composite product (the difference will be explained later in this documentation). Also, the admin can edit or delete a product from the table he will have. A composite product with a custom name and automatic rating, price, fats, proteins, sodium, calories which are calculated in the background as an average (rating) or sum (rest of attributes).

1.2. Secondary Objectives

- Analyse the problem and identify requirements
- Design the orders management application
- Implement the orders management application
- Use a layered architecture
- Test the orders management application

2. Functional and non-functional requirements

2.1. Functional requirements

2.1.1. MoSCoW classification

- Must have: A graphical user interface that allows the user to interact with the application.
- Should have: Generic classes and methods that allow the use of the same methods on different types of objects, without duplicate code.
- Could have: Eye catching graphical user interface with nice colours and good-looking aspect regarding

- The administrator:
  - From a.csv file, import the first batch of products that will populate the menu
  - Manage menu items: add/delete/modify products and construct new products made of several menu goods (an example of a composed product would be "daily menu 1," which includes a soup, a steak, a garnish, and a dessert).
  - Produce reports on completed orders based on the following criteria: time interval of the orders, the items requested more than a indicated number of times so far, the clients that have requested in excess of a predetermined number of times up to this point and the value of the request was higher than a predetermined sum, the items requested within a indicated day with the number of times they have been ordered
- The client:
  - Enlist and utilize the enlisted username and password to log in inside the system
  - View the list of products from the menu
  - Look for items based on one or different criteria such as keyword
  - Make a request comprising of a few items and a bill will be generated
- The employee:
  - notified each time a new order is performed by a client

2.1.2. Use case scenarios

- Log in as administrator, employee or client

- Depending on their status they will be able to do different procedures: as a client to order, filter products from the menu; as an admin: add, edit/modify, delete products, import products and generate reports based on different criteria; as an employee: see the current orders and remove them from the window (but not from storage)

2.2. Non-functional requirements

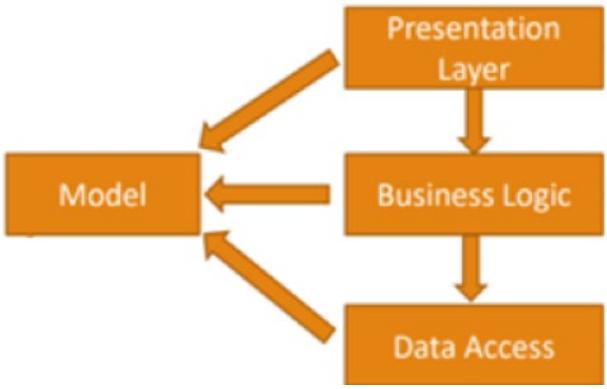
- Ease of use: the user interface should be simple and intuitive, with labels that indicate what the application is expecting from the user
- Use of authentication to restrict the access of clients and regular employees to the administrator's functionalities
- Compact design: the application's graphical interface should not have a greater size than necessary but should be big enough to display all the information that has to be shown
- The response time for each transaction should not exceed 2 seconds and should perform instantly for most of the transactions
- Reliability of database: the system should use a reliable database management system

3. Conceptual architecture

The code is written in OOP style, following the 4 pillars of Object-Oriented Programming paradigm: Encapsulation, Abstraction, Polymorphism, Inheritance.

4. Design and Implementation

4.1. Package diagram



Layered architecture is considered the most common and widely used architectural framework in software development. It is an n-tiered pattern where the components are organized in horizontal layers. They work together as a single unit of software, but do not depend on each-other.

There are four layers in this architecture:

- The presentation layer – is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. In the current application, it receives which is the next step the user wants to take and the data input (if it is the case). It also displays new windows bringing the information the user wanted.

- The business logic - handles the business rules, calculations, and logic within an application which dictate how it behaves. It performs data validations and decides which will be the next step taking into account the input from the user.
- Data Access – layer of a computer program which provides simplified access to data stored in persistent storage of some kind, such as an entity-relational database.
- Model – this is where all the data structures are implemented. There is a class for administrator, client, order and menu.

4.2. Classes implementation

4.2.1. Model

- Administrator – this class implements Serializable, which is a marker interface. It extends Person and uses its constructor.
- BaseProduct – extends MenuItem and is part of the Composition design pattern. The initial products can all be seen as base products. But multiple base products or menu items can represent a composite product.
- Client – like Administrator, it is a child of the class Person and uses its constructor
- CompositeProduct – has an ArrayList of MenuItems because it can be composed of both base products and other composite products. Since all of them are MenuItems, composite product uses an array list of menu items to store its components; the Composite Design Pattern is used
- MenuItem – is the parent class for both base product and composite product. It has the following fields: title, rating, calories, protein, fat, sodium and price; uses the Composite Design Pattern, which allows treating individual objects and compositions of objects in the same way
- Order – it has an id, generated randomly, the client that performed the order, an arraylist of menu items ordered by the client and a date that is generated at runtime and retrieves the local date
- Person – the parent class for both administrator and client. It has the following attributes: first name, last name, age, username and password. Password is validated at registration and must be safe.

4.2.2. Data access

- Deserializer – has methods for reading serialized data from files for orders, clients and admins
- FileWriter – has the reverse methods, for serialization. Data about the same classes order, client and admin are stored in a serialized manner. For this all the classes that make part of those classes must be serializable. If a class is not serializable by definition, it must implement Serializable interface.
- MyReader – it has methods for reading and writing in a csv file. It is used for loading and saving any changes on the initial menu items. I chose to use this way of storing data just to exercise a little bit with different file types as I already made 3 methods that use serialization and deserialization

4.2.3 BLL

- RegisterBLL – has methods for validating the data entered by a client while trying to make a new account. It checks for the password to be safe and for the username to be unique. The new user is then added in the required hash set and serialized
- LoginBLL – checks if the username and the password correspond to an already registered user, searching through the hash sets. It returns a result that is used later on for determining which graphical user interface to open.
- DeliveyService – by far the most complex class, it has lots of important methods that will be discussed. Firstly, it should be noted that this class implements the interface IDeliveryService which has the signatures for the required methods. It uses the Design by Contract technique having pre and post conditions, invariants and assertions. Also a javadoc is made. More complicated methods are those that involve filtering. I used stream processing to achieve the desired results. The method which filters the items for the clients and the one that generate the reports for the administrator are those two methods. Other methods involve the necessary operations like add, edit and delete for admin.

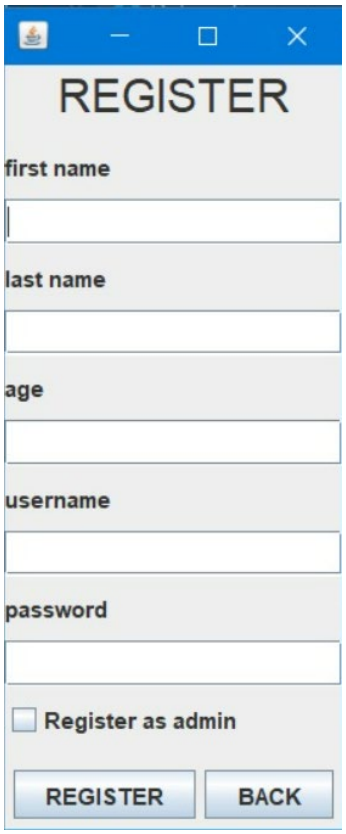
4.2.4. Presentation/View

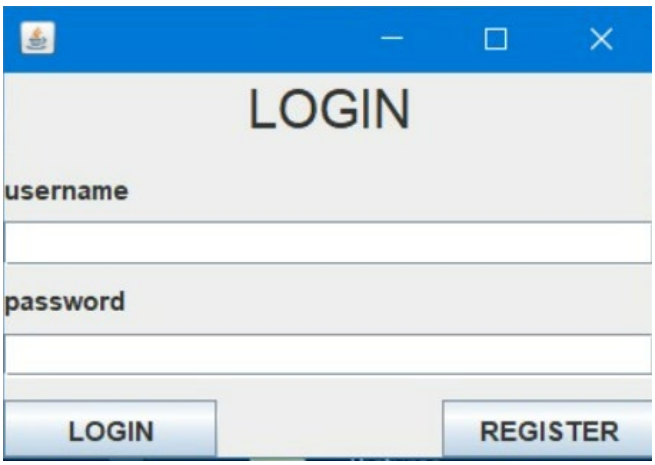
- In the AdminGUI class are used JButoons that help the user perform the operations that he wants. JTextField are utilized to display informations such as title, calories, other health related piece of information and price. The cart is presented as a table using JTable and DefaultTableModel.

- ClientGUI – is extremely similar to the previous class, the only difference is made by the operations that can be made. The client is able to order the food he chooses, which is added to the Hashset of orders from class DeliveryService and saved in csv file. Moreover, the employee gets a notification that an order has been made
- EmployeeGUI – uses the Observer Design Pattern for notifying the employee when a new order is made. In this way, he can prepare the dishes in time and be ready for the delivery
- LoginView – is a basic login GUI with username and password. Bellow a picture with the page is showed.
- LoginController – implements the addActionListener. Using the switch case, a message dialog appears with specific messages "Admin good" and a new window is opened or, in case of an error, just an error message
- RegisterView – the register class contains details that are highly useful for making an account and deliver an order; contains the username and password fields which are essential for the login and, also, specific informations about a user; extends the JFrame class
- RegisterController – has the addActionListener and getters that are needed; similar with the LogginController class

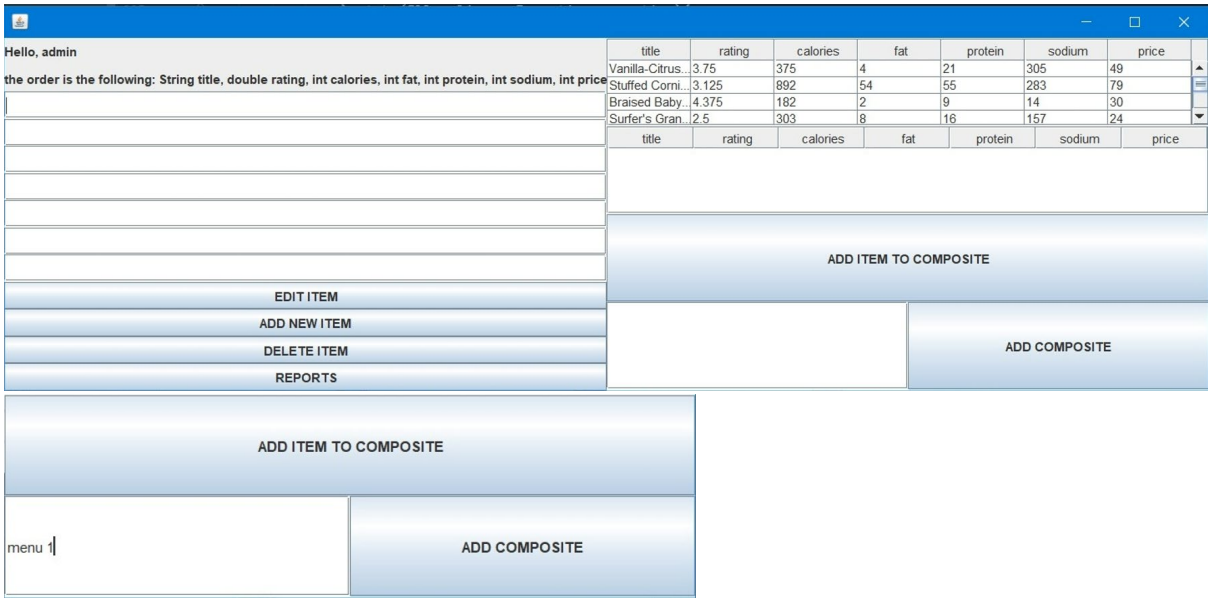
GUI presentation

- Firstly, it is presented the login and register windows that make the separation between clients, admins and employees.

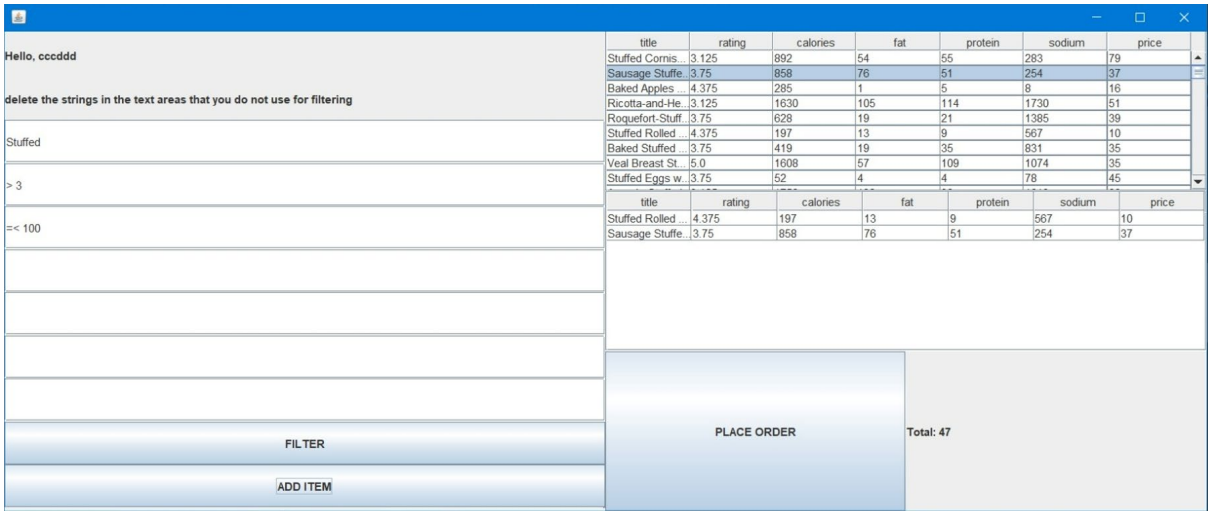




- Here is the GUI for the admin; for every action that they can do, there is a specific button such as: add / edit / delete, do reports or add a composite or an item to composite. It is the most compact window



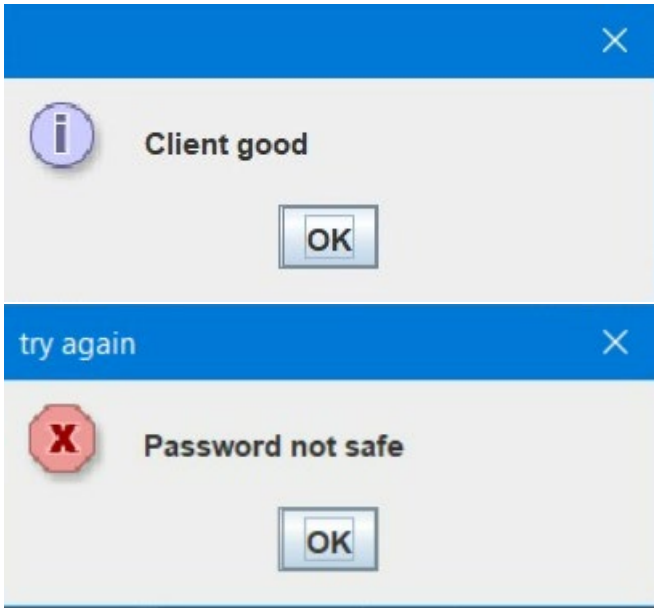
- The window for the client is the simplest and straightforward one. There was no ought to include more buttons. In the right, is the food list and another one with the order and the total. Also, the user can filter his options, for an easier search



5.1. Successful operations

5.2 Validation

Here are some examples of the validation tests. If a field that should contain only numerical values contains some non-numeric value, a message will be displayed with a suggestive warning. Also, if the password doesn't accomplish the basic requirements, such as: eight characters, an uppercase letter, a number or special character (and others, using java.util.regex, the Matcher class, which is predefined). In case all data is correctly introduced, a confirmation message will be displayed.



5. Conclusion

This project helped me discover new functionalities in Java programming language, for instance, Java Collections Framework, which provides useful data structures and algorithms that reduces programming effort and increases the program's performance. Moreover, I learned about Hashtable and HashMap, the difference between the two classes and how to use them or about lambda expressions and stream processing and Java serialization. All the main and secondary requirements have been accomplished. At the end of the assignment, I consider that my Java programming skills evolved since the last project.

6. Bibliography

- the lectures from the course
- support presentation
- Wikipedia
- Geeks for Geeks
- W3schools
- Stack Overflow
- Quora
- freeCodeCamp
- javaTpoint
- Tutorials point
- IBM
- Baeldung
- Rose India
- Academia
- Oracle Java Documentation