

**Technical University of Cluj-Napoca**

**Faculty of Automation and Computer Science**

Programming Techniques

Assignment 4

FOOD DELIVERY MANAGEMENT SYSTEM

Teacher: Prof. Ioan Salomie Student: Alina Mihuț

Teaching assistant: Ing. Ciprian Adrian Stan Group 30422

Table of contents

1. Assignment objective
2. Problem analysis, modeling, scenarios, use cases
3. Design (design decisions, UML diagrams, class design, relationships, packages, algorithms, user interfaces)
4. Implementation
5. Results
6. Conclusions
7. Bibliography
8. Assignment objective

Design and implement a food delivery management system for a catering company. The client can order products from the company’s menu. The system should have three types of users that log in using a username and a password: **administrator**, **regular employee**, and **client**.

The administrator can:

* • Import the initial set of products which will populate the menu from a .csv file.
* • Manage the products from the menu: add/delete/modify products and create new products composed of several products (an example of composed product could be named “daily menu 1” composed of a soup, a steak, a garnish, and a dessert).
  1. • Generate reports about the performed orders considering the following criteria: o *time interval of the orders* – a report should be generated with the orders performed between a given start hour and a given end hour regardless the date.
  2. o *the products ordered more than a specified number of times so far*.
  3. o *the clients that have ordered more than a specified number of times and the value of the order was higher than a specified amount*.
  4. o *the products ordered within a specified day with the number of times they have been ordered*.

The **client** can:

* • Register and use the registered username and password to log in within the system.
* • View the list of products from the menu.
* • Search for products based on one or multiple criteria such as keyword (e.g. “soup”), rating, number of calories/proteins/fats/sodium/price.
* • Create an order consisting of several products – for each order the date and time will be persisted and a bill will be generated that will list the ordered products and the total price of the order.

The **employee** is notified each time a new order is performed by a client so that it can prepare the delivery of the ordered dishes.

1. Problem analysis, modeling, scenarios, use cases
2. Analysis

The application should resemble a delivery management system application with three types of users: admin, client and employee, each of them performing different tasks. The admin is able to view and import the initial set of products, add, modify or delete base products, add composed products and generate reports based on different types of criteria. The client is able to view the whole menu, search for menu items based on one or more filters and make orders. When a new order is made by a client, the employees will be notified. The employee can view the list of pending orders and process them.

B) Modeling, scenarios and use cases

The main scenario is the following one:

The user is able to sign in as one of the three types: admin, client or employee. Then, he logs in and he is able to perform different operations, depending on the type of user he logs in as, meaning he is sent to one of the three windows, with the operations specific to each user. The admin performs the following tasks: importing the initial menu from a CSV file, add a new product, modify a product or delete one, create composed menu items, from more base products and generate reports, related to the orders, products and clients. The client logs in to its window, where he can view both the based and composed products in the menu, search for menu items based on their name, price, rating, calories, proteins, fats or sodium and make an order with one or more menu items (based or composed). After a new order is made, a bill is generated and the employees are notified. The employee logs in and he can view the list of pending orders from the clients.

1. Design

A Layered Architecture is the organization of the project structure into four main categories: presentation, application, domain, and infrastructure. Each of the layers contains objects related to the particular concern it represents.

The presentation layer contains all of the classes responsible for presenting the UI to the end-user or sending the response back to the application.

The application layer contains all the logic that is required by the application to meet its functional requirements and, at the same time, is not a part of the domain rules.

The domain layer ( business layer ) represents the underlying domain, mostly consisting of domain entities and, in some cases, services. It contains the classes that encapsulate the application logic.

The Data Layer contains all the classes responsible for saving and processing the sets of objects.through serialization and deserialization and the classes which handle the file writing.

I chose to design my application using a Layered Architecture. My application has four packages corresponding to the three design layers: data layer, business layer, presentation layer and application.

The **business** layer package contains the classes which handle the application logic, creating the lists of objects, menu items, orders and handling the operations on these: creating menu items, deleting menu items, modifying menu items, creating orders, creating the list of users, etc.

The **presentation** package the package with the controller classes ( 4 in total – one controlling each scene), as its name suggests, which controls both the GUI classes and the model classes and the data flow between the two. It receives keyboard inputs from the GUI as well as button presses and translates the events into requests, which are sent to the model and the user interface class, with the methods concerning the running of the Graphical User Interface and its’ proper working.

Last but not least, the **application** package contains the main class, initializes the primary stage and displays it on the running of the application.

1. Implementation

Each of the three packages in the Layered Architecture pattern is divided into several classes, containing specific methods.

I designed my application according to the system design presented below.

The interface IDeliveryServiceProcessing containing the main operations that can be executed by the administrator and client, as follows: administrator: - import products, manage the products from the menu, generate reports; client - create new order which implies computing the price for an order and generating a bill in .txt format, searching for products based on several criteria.

The business layer package is divided in multiple classes, performing important tasks. First, we have the Order, Menu Items and two classes which extend the Menu Items class – Base Product and Composite Product, each with their specific attributes. These classes contain constructors, getters and setters for creating and accessing the objects. I used the Composite Design Pattern for defining the classes MenuItem, BaseProduct and CompositeProduct. The Delivery Service class, which implements the IDeliveryServiceProcessing interface, hold all the main methods performed by the users, the lists of objects, obtained by means of serialization and deserialization,

a structure of type Map<Order, Arraylist <MenuItem>>, used for storing the order related information in the following way: the key of the Map is formed of objects of type Order, and the values are the collection of menu items in each order. The map is accessed and processed using streams and lambda expressions. The Date class has attributes of time int – day, month, year, hour, minute, second, which are given as parameter for each order and are used to get the exact current time.

The data layer package is divided into 2 classes, File Writer and Serializator, used to access the serialized objects and put them in collections, serialize new objects and write to file the generated bill after a new order was made by the client and the reports generated by the administrator.

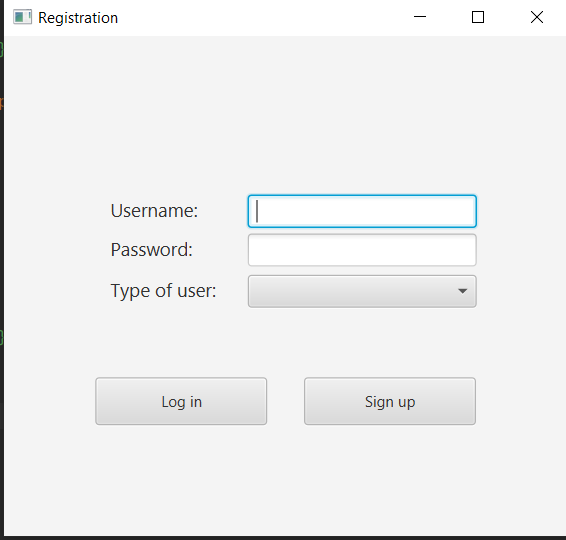
The presentation package, has the user interface classes and the controller classes, with methods which perform actions on buttons or display alerts in the graphical interface for data validation and confirmations. Each of the six controller classes corresponds to one of the scenes in the graphical user interface. They have methods which get the inputs from the user and send them in order to be validated and inserted in the tables. They also have methods actions on buttons, allowing the user to select the operation he wishes to perform.The buttons also allow the user to easily navigate between the scenes. Displaying the tables on screen is handled by the Table View objectswhich takes as parameters a list of objects and generates the header of the table then populates the table with the values of the elements from the list. This method is called for all three types of objects and the contents the tables are displayed in a table view in each scene and can also be processed on the action of the mouse.

The Main class in the application package initializes the primary stage and displays it on the execution of the application.

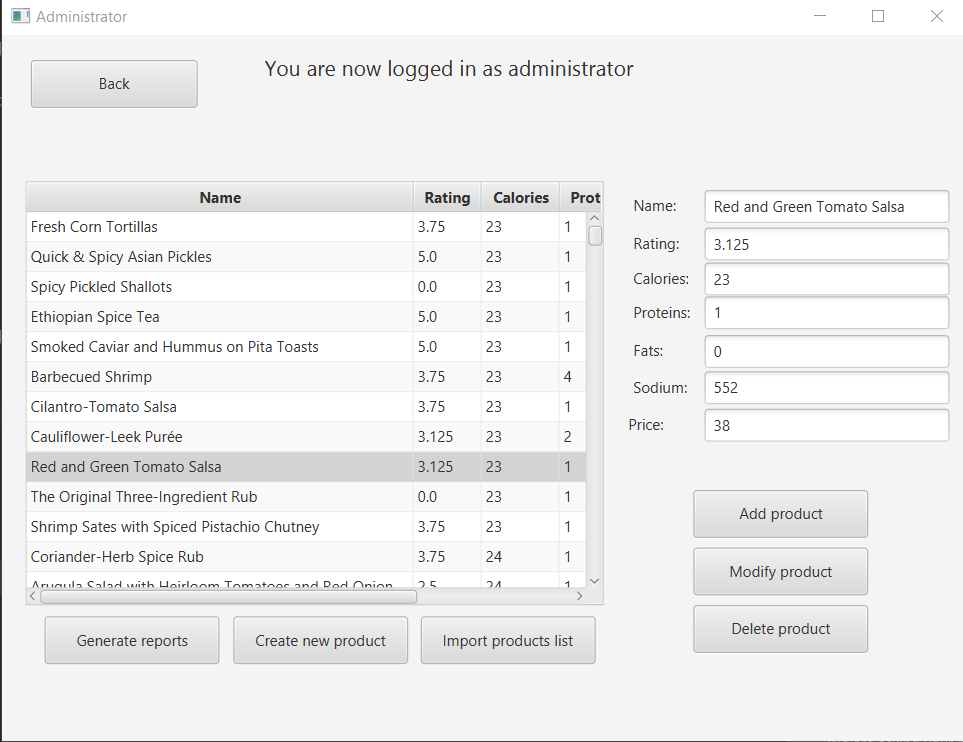
In the implementation of the graphical user interface, I used Scene Builder. The GUI contains several windows, a main one (for log in and sign up) and three for logging in for each type of user. The user has to enter in each of the text fields or choice boxes the parameters of the entry to be inserted/updated/deleted or searched for and press a button for choosing the operations he wants to perform. The data he enters is verified and in case it’s not correct, he receives an alert box on the screen. The user can see the data in each table, updated after each operation. He can also navigate through the windows with the actions set on buttons.

I designed the user interface in order to be easy and inductive for the user. The design of the user interface window can be seen in the following pictures.

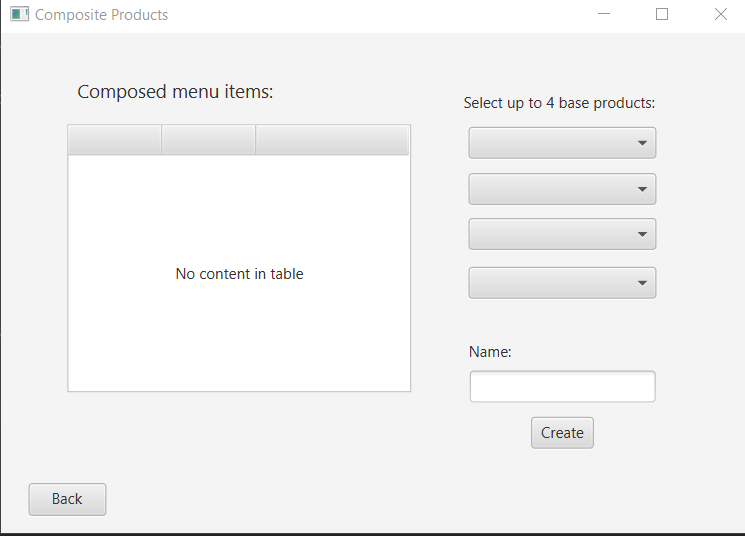
The main window, where the user is able to sign in as one on the three types or login if he already has an account :



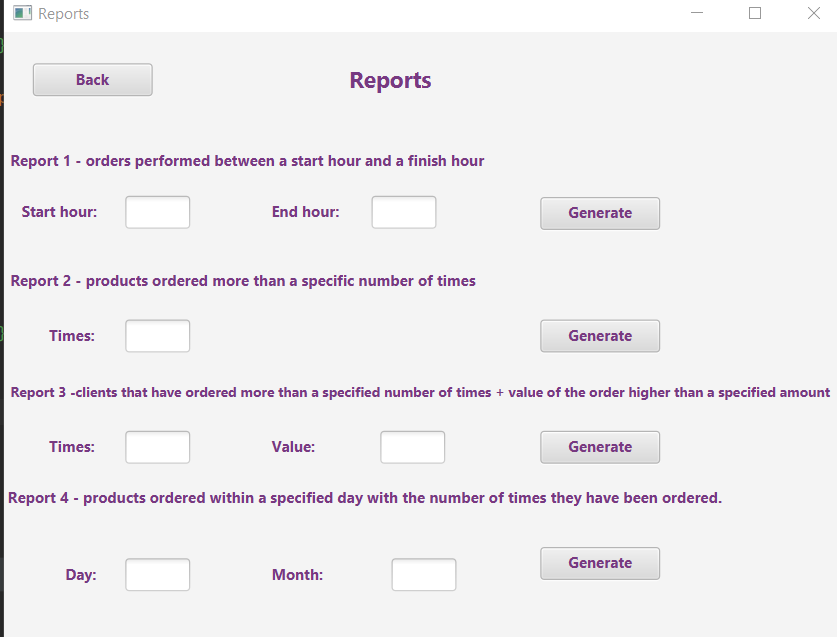
The administrator main window:



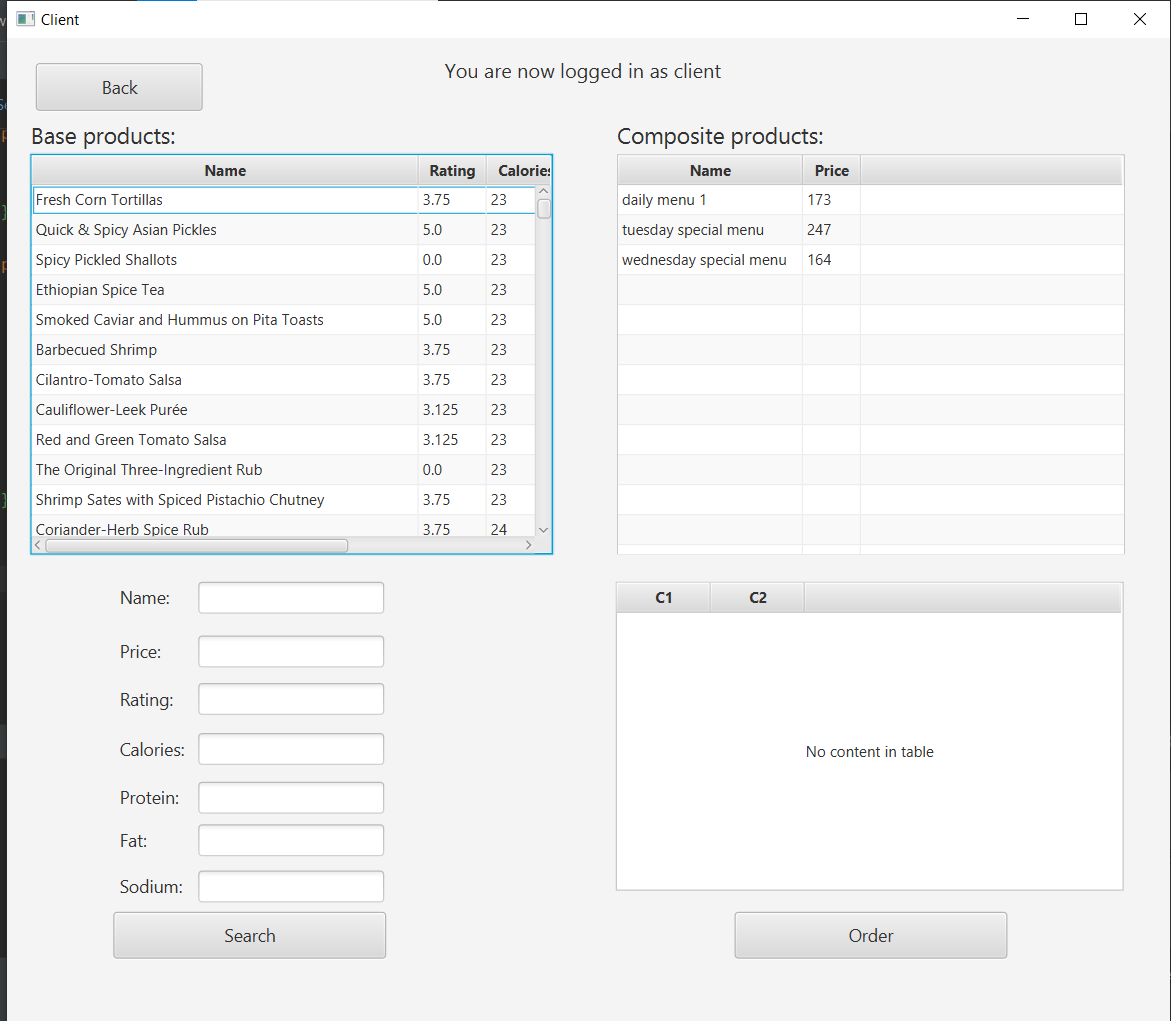
The second administrator window, where composed menu items are created:



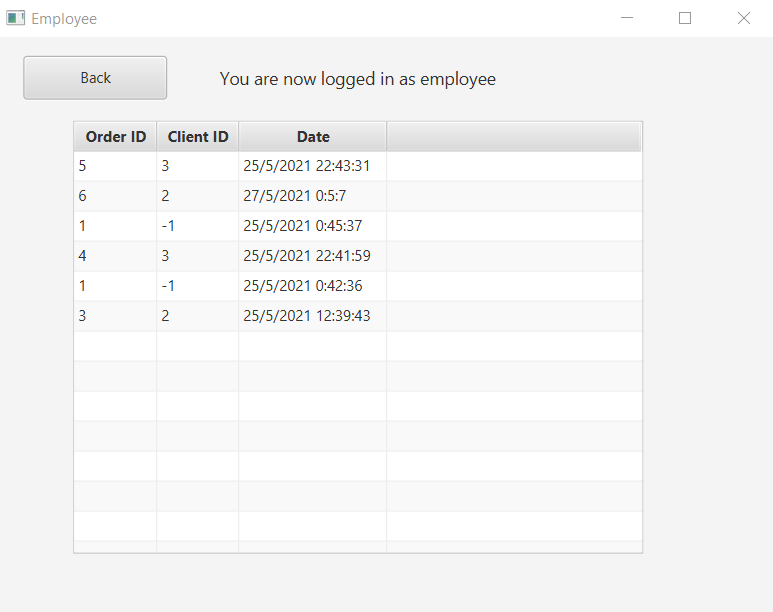
The third administrator window, where reports are generated:



The client window:



And the employee window, where the pending orders are displayed:



1. Results

The results of the performed operations on the menu items can be seen by displaying the contents of the tables in the graphical user interface. After each order is made, a bill is generated in a text file and the employee user is notified. The admin can also generate reports, based on different criteria. An example of report is the following: products orders on a specific date, with the number of times they were ordered.

REPORT 4 :   
----------------------------------------  
Products ordered on 25/05/2021   
Scallion Cilantro Pancakes ordered 1 times  
tuesday special menu ordered 1 times  
Barbecued Shrimp ordered 1 times  
Ginger Tea ordered 1 times  
Spicy Pickled Shallots ordered 2 times  
daily menu 1 ordered 1 times  
Cauliflower-Leek Purée ordered 2 times  
Tomato Salsa ordered 1 times  
Red Pepper Sauce ordered 1 times  
The Original Three-Ingredient Rub ordered 1 times  
Ethiopian Spice Tea ordered 1 times  
Red and Green Tomato Salsa ordered 2 times

1. Conclusions

Working on this project was a good opportunity for me to gain a deeper understanding of the OOP concepts learned in the first semester as well as working on an application with a graphical user interface. It was a bit challenging at first, until I understood how to work with serialization and deserialization, as well as with maps of objects and hash codes and how to structure and organize the project in order to respect the layered architecture pattern and the given class design. I realized that a good modelling of the problem from the beginning really helps a lot in the future development.

All in all, I liked working on this project and I feel like I gained a lot of knowledge from it.

As further improvements, a lot could be made. For example, to make a more complex GUI and add new features.

1. Bibliography

- ***Assignment\_4\_Support Presentation*** PowerPoint presentation