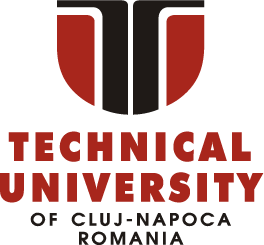
Technical University of Cluj-Napoca

Programming Techniques

Laboratory Assignment Three Documentation

Restaurant Management

**

Teacher: prof. Dr. Ioan Salomie

Laboratory teacher: Dr. Cristina Bianca Pop

Student: Nemeș Mihnea-Andrei

Group: E\_30424

Contents

[1.Assignment Objective 3](#_Toc73597685)

[2.Problem Analysis, modelling, scenarios and use cases 3](#_Toc73597686)

[3.Design 5](#_Toc73597687)

[4.Implementation 8](#_Toc73597688)

[5.Conclusions 10](#_Toc73597689)

[6.Bibliography 11](#_Toc73597690)

# 1.Assignment Objective

Design and implement a food delivery management system for a catering company. The client can order products from the menu, he can filter the products by a field, such as name, rating or calorie count. The system should have three types of users that can log in using a username and a password. The three types of users are administrator, client and employee.

The administrator can do the following actions:

* Import the initial set of menu items which will populate a table from a .csv file
* He can manage the products from the menu, meaning he can add, delete, or edit products. He can also create new products, either as base products, or as products comprised of several other products, a composed product. An example of a composed product would be a “menuOne” composed of a steak, fries, soup and dessert.
* The administrator can also generate reports about the placed orders following different criteria: a report generated with the orders that were placed between a time interval, a given start and end hour; a report displaying the products that were ordered more than a specified number of times; a report displaying the users that placed more orders than a specified number of times, and whose orders add up to be greater than a specified amount; a report displaying the products ordered in a specified day, with the number of times that they were ordered.

The client can do the following actions:

* Register using a username and a password and log into the system.
* View the list of products available on the menu.
* Search for products based on a filter, such as name, calorie count or rating.
* 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 contain the list of the ordered products and the total price of the products comprising the order.

The employee plays the role of an observer, and they will be notified whenever a client places an order so they can prepare for delivery. This notification will contain the date that the order was placed on, the items that were ordered, and the total price of the order.

# 2.Problem Analysis, modelling, scenarios and use cases

1.Problem Analysis

The application should resemble a food delivery management system, where there are three types of users: admins, clients and employees. They should all be able to perform specific operations according to their user type, just like in real life for such an activity.

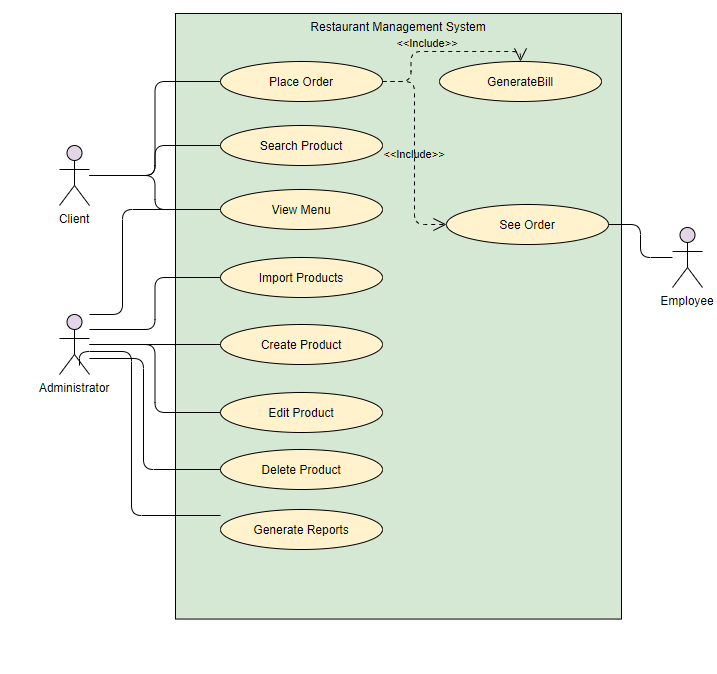
2. Modelling

The goal of the project is for the user to be able to manage a food delivery application with a login and register system. There are three types of users: Admin, Client and Employee. The admin can import menu items from a .csv file, create new items, be it base products or composite products, edit items, delete items and generate reports. The client can order items, search for specific items based on a filter and can generate a bill for their order. The employee will receive a notification whenever an order is placed in the system.

3. Scenarios and use cases

A use case is a list of actions or event steps typically defining the interactions between a role(known in UML as an actor) and a system to achieve a goal. The actor can either be an external system or a human. The use cases are in strong correlation with the steps that the user has to follow, this is the reason why my interface was designed in a friendly way, in order to be easy to use.

1.Use case diagram



2.Use cases

1.Use case: View menu

Actors: admin, client

Resume: The admin or the client can see the items currently in the menu table in their respective interfaces, if they want to update the table they can press the refresh button from the interface.

2.Use case: Search item

Actors: client

Resume: The client can search for items based on a filter such as name, rating or fat content of the product. The client chooses the filter from the drop down menu, fills the field representing the value of the filter and then presses the button. The result of the search will be displayed in the already visible table in the interface

3.Use case: Make order

Actors: client

Resume: The user will select the items they want by selecting an item in the menu table and clicking the add button. They can see the items selected in a drop down menu. If they change their mind about an item they selected, the user can select it from the drop down menu and click the remove button. When the user has all the items they want into the “basket” they will click the order button in order to finalize the order. The bill of the order will be automatically generated into a file.

4.Use case: Create menu item

Actors: admin

Resume: The user can choose to create either a base product or a composite product. If they want to create a base product, they will fill all the necessary fields and click the create base product button. If on the other hand they wish to create a composite product, they can select the items from the menu table, and using the add button, add them to the composite product. After the user has filled the name of the new composite product they will click the create composite product button, and the new button will be visible in the menu table.

5.Use case: Delete item

Actors: admin

Resume: The user can delete an item by selecting it from the menu table and clicking the delete button.

6.Use case: Generate reports

Actors: admin

Resume: The user can choose to generate a report from a possible four different reports, each with their different criteria. The result of the report will be displayed in a new window into a text area.

7. Use case: Edit item

Actors: admin

Resume: The user selects the item to be edited from the menu table, and depending on the type of product, either base product of composite product, a window will open where the user can change either the fields of the product in the case of the base product, or the composition in the case of the composite product.

8.Use case: Be notified and See order

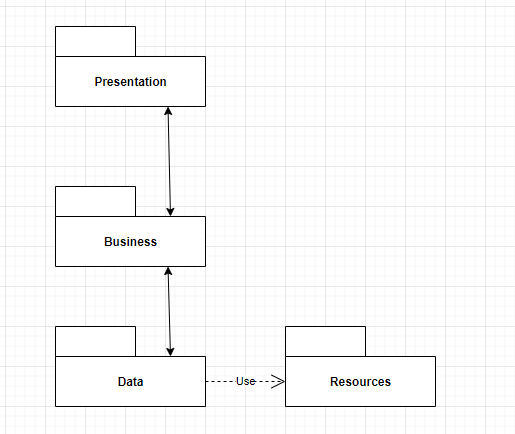
Actors: employee

Resume: Every time an order is created, the employee will be notified and the order will appear in a text area in their respective employee interface.

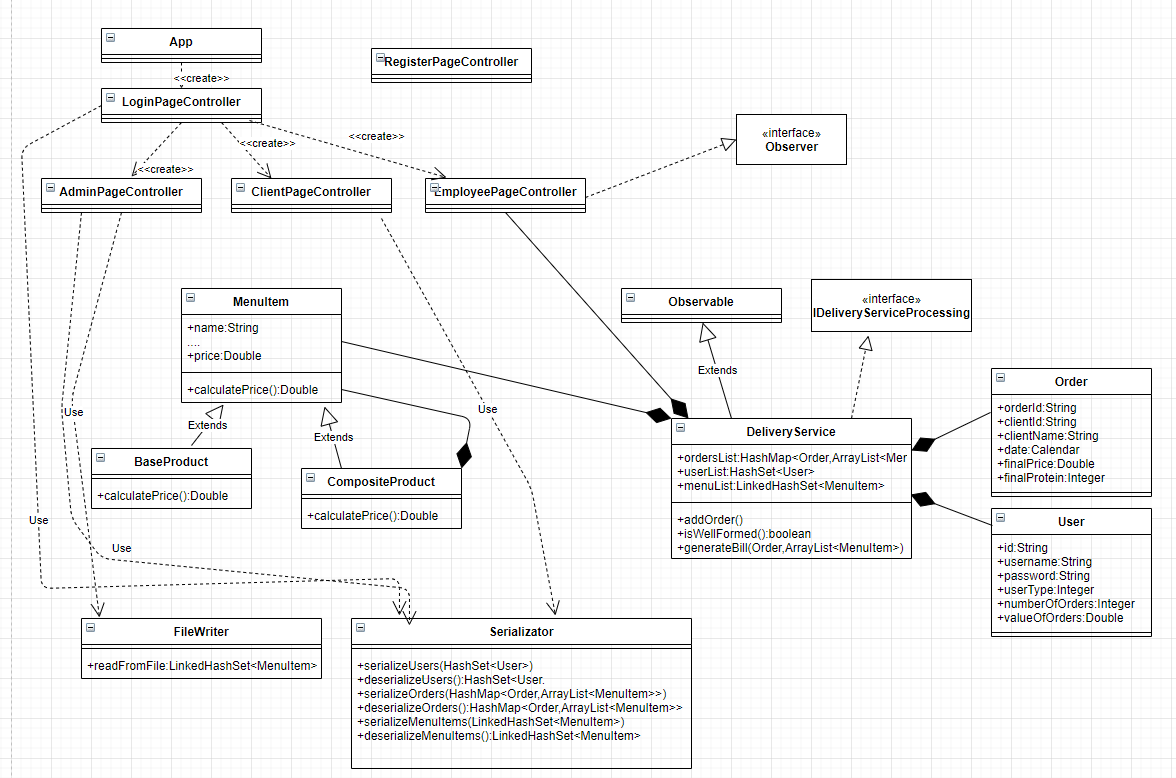
# 3.Design

A.UML Diagrams

1.Package Diagram



2.Class Diagram



B.Data Structures

The data structures that I used for my project are mostly primitive data types such as integers and doubles, but I have also some complex ones like ArrayList, LinkedHashSet, HashSet, HashMap and the objects that I created myself, such as MenuItem, BaseProduct, CompositeProduct, DeliveryService, Order and user. I chose to use LinkedHashSet for working with the menu items because it maintains the order that the items were introduces in while also not allowing duplicates.

C.Packages

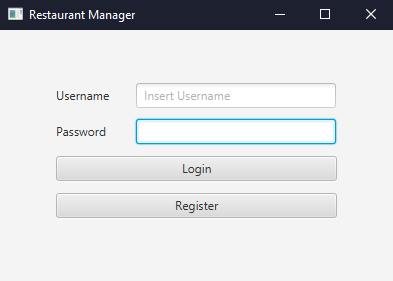
Java packages are used in order to organize modules and group classes and interfaces that are related in Object Oriented Programming(OOP) development. For this project I have used a Layered Pattern Architecture:

* Presentation: it contains all the fxml files responsible with the interfaces needed for the application and the controller classes responsible with the GUI functionality. The main classes are AdminPageController, ClientPageController, EmployeePageController, LoginPageController, RegisterPageController. They control their respective GUI windows and implement their functionality. Aside from those, I have a few more classes that are used to implement additional functionality for the client and administrator.
* Business: IDeliveryServiceProcessing: interface where all the methods for the DeliveryService class are described; DeliveryService: contains the users, orders and the item menu of the whole system, implements the operations required for the administrator and user; Order: details of an order, orderId, clientId, client username, date, total price; MenuItem: abstract class, representation of a menu item from the food system, has fields name, rating, calories, protein, fat, sodium and price; BaseProduct: simple product, extends MenuItem; CompositeProduct: represents a composed product, extends MenuItem; User: details of an user, userId, username, password, type, number of orders and value of orders.
* Data: FileWriter: reads the first menu items from the .csv file; Serializator: class that serializes and deserializes the menuList, userList and ordersList fields from the DeliverySystem Class; User: details of an user, userId, username, password, type, number of orders and value of orders; HelperStuff: helper class used for inter controller communication.

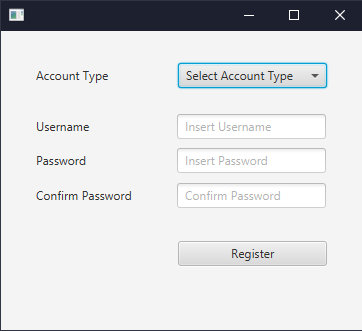
D.User Interface

The interface represents the link between the software application and the user, and it must be friendly and easy to use, so that the user has a nice experience.

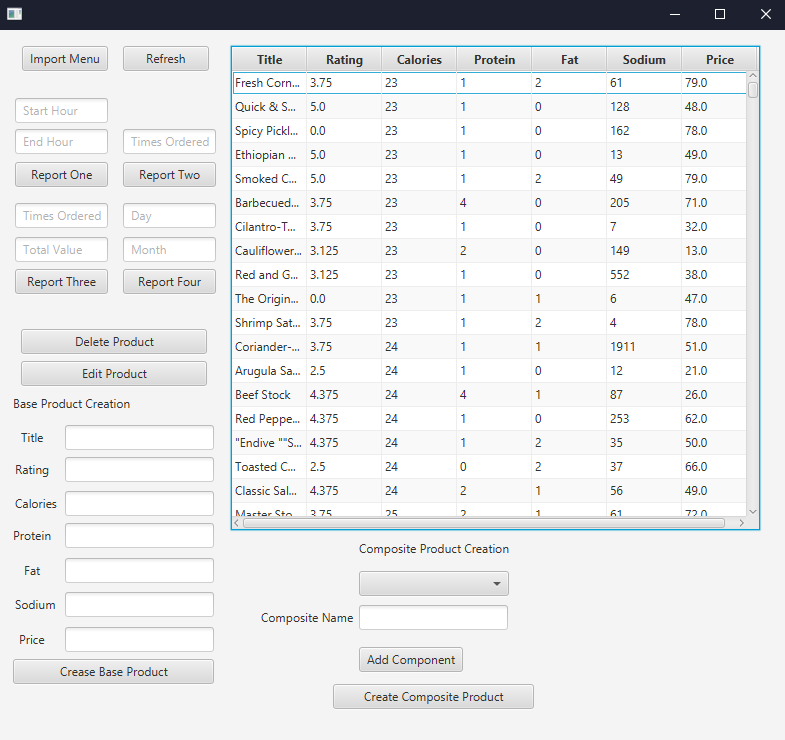
This is an illustration of the first window that is opened when running the application:



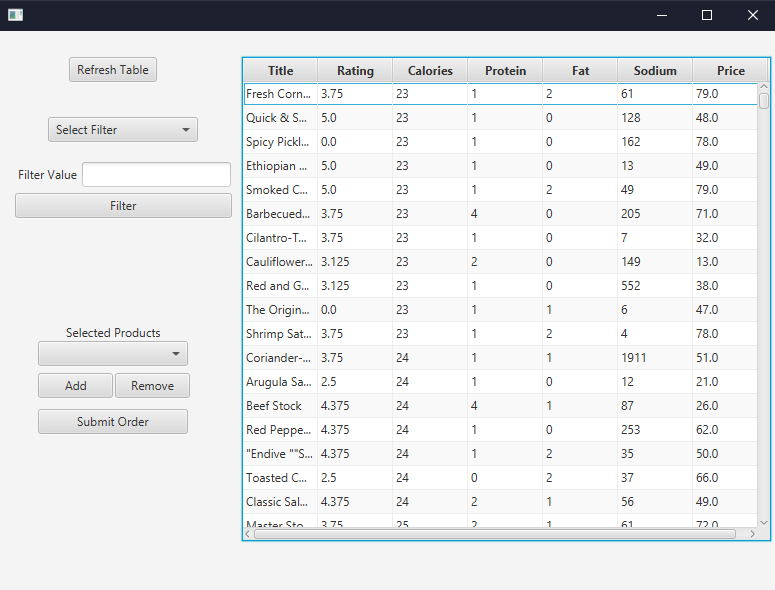
The register window:



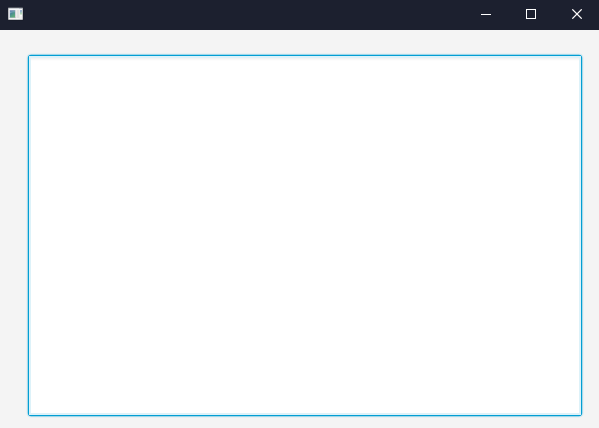
The administrator window:



The user window:

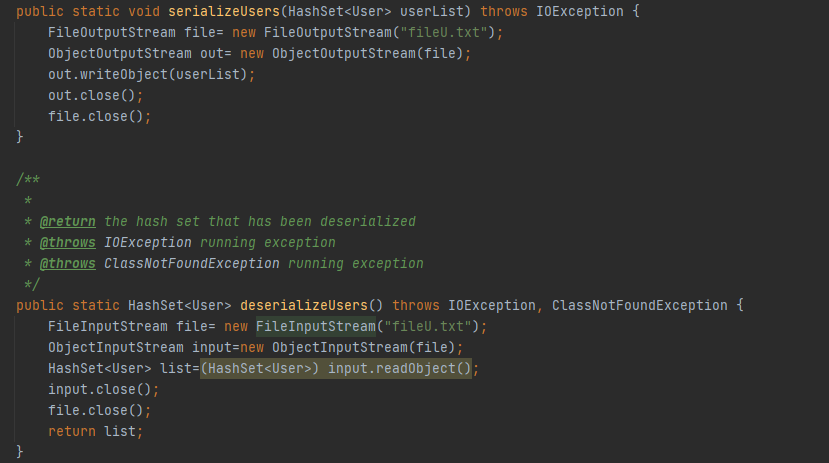


The employee window:

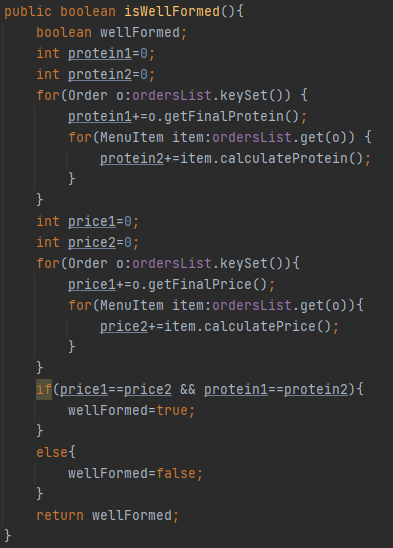


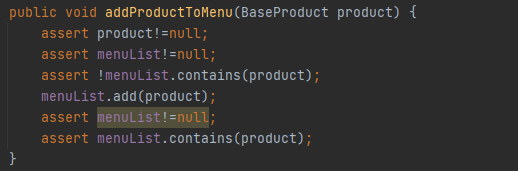
# 4.Implementation & Results

The classes and what they do was described previously, so I would just like to present a few methods that were the most useful and most important in this assignment.

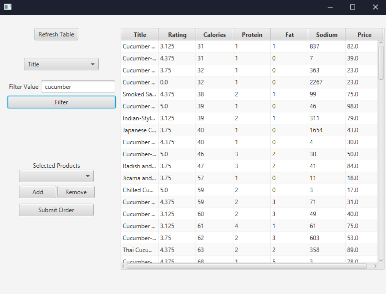


This method is used to retain the modifications that are done inside to application with the use of an external file, that can always be opened and read from, thus ensuring that the application can run completely local, without the need of a database for storing for example the users. The picture above shows the methods for serializing and deserializing the users.

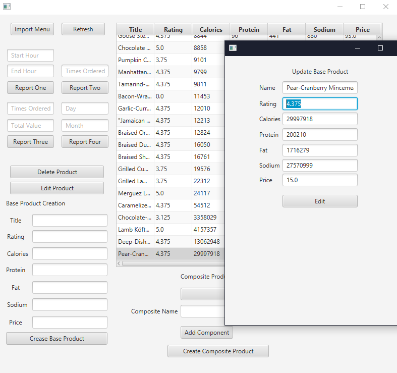
This is the invariant for the DeliveryService class, which makes sure that all orders in the class abide by a predefined rule: if we were to calculate the individual price of each component of an order, we must at the end get a result equal to the field finalPrice from the order. The same goes for the protein. These invariants are important because they make sure that the classes in our application are correctly defined.



I have also employed the use of assertions in order to more easily determine whether the data that I’ve been given in the method has the correct format, or to make sure that the fields in the DeliveryService class that I need to use are not null, thus avoiding a lot of errors.



This screenshot shows the evolution of the GUI when the user tries to filter the products by the field title, searching for products that contain the keyword “cucumber”.



This screenshot shows the functionality when the administrator wishes to edit a Base Product from the menu table.

# 5.Conclusions

This assignment offered me a good opportunity to remember and apply the Object Oriented programming paradigms and concepts. It also offered me an insight into the process of designing and developing a software application. Also, it was very interesting and eye opening to learn how to serialize and deserialize objects, as before now that information was hear say for me. Even more, it was interesting to learn how to create and use custom Javadoc tags, how to employ the use of lambda expressions and stream in order to more easily deal with large arrays without the need for loops, and how to apply the Composite Design Pattern and the Observer Design Pattern. All these new things I’ve learned will come in handy in the future, be it for future projects or even at a job.

What I take from every assignment like this, where I have to research how to use different elements of Java classes, is not to panic and try to stay calm and divide the problem into smaller ones that are easier to deal with, because otherwise the project can become quite overwhelming. Another very important thing is that testing frequently helps you avoid big errors in the long run. It’s always better to test a small functionality of the application and confirm that it works at intended, than to do everything at once and then be overwhelmed by the number of errors that arise. I can say that I am proud that I managed to finish this project and that I found solutions for all the problems I have encountered during the development process.

The finished application is by no mean perfect, a lot of improvements can still be made. For example, more functionalities could be added, such as adding a request function, where the user can request a type of product to be added to the menu, or delivery request, where the user can select the address where the products should be delivered. The GUI could also be improved in order to increase the clarity of the application and to make it nicer overall. The methods could also be improved as to increase the efficiency of the running of the application, as well as the usage of the classes.

# 

# 6.Bibliography

1.Programming Techniques-Lecture Slides by Prof. Ioan Salomie

2. Assignment 4 Support Presentation-Programming Techniques Teams

3. <https://openjfx.io/openjfx-docs/> - configuring javafx

4. <https://www.baeldung.com/java-8-streams-introduction> - introduction to java streams

5. <https://www.jetbrains.com/help/idea/opening-fxml-files-in-javafx-scene-builder.html> - configuring Scene Builder

6. <https://app.diagrams.net> – creating the class & package diagrams

7. <https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html> - javadoc tags