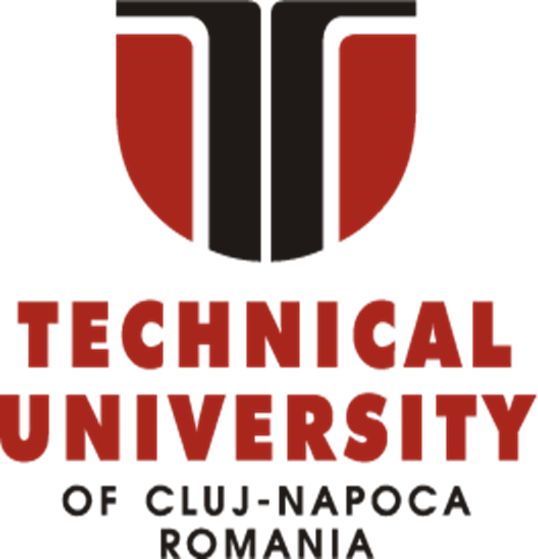
**FUNDAMENTAL PROGRAMMING TECHNIQUES**

**ASSIGNMENT 4**

**FOOD DELIVERY MANAGEMENT** **SYSTEM**

**Mureșan Ioan-Alexandru**

**Group 30421**

**Objective**

Design and implement an application for managing orders, menu items and generating reports for orders from various clients for a catering company, with the use of a serialization for storing the products, client accounts, as well as the orders. The secondary objectives will be:

* Use of JavaDoc comments for documenting classes, and generating the corresponding JavaDoc files, but also implementing the Design by Contract pattern, with preconditions, postconditions and assert statements.
* Design and implementation of a Graphical User Interface for the delivery system – the purpose of this element is to provide the user with the possibility to view the menu items and also search among them based on certain categories, such as price, calories, protein and others, and then place orders after selecting the desired items. Then, the administrator is able to perform CRUD operations on the menu items and generate reports regarding the previous orders created by the clients (presented in the Design section)
* Generating a .txt bill file containing information regarding every order in the order list, when the order was placed (presented in the Design section)
* Use of the Obvervable design pattern in order to implement a notification system for the employee, to receive when a new order has been created by a user (further presented in the Implementation section)
* Use of a Composite Design pattern for the implementation of the MenuItem, BaseProduct and CompositeProduct classes
* Use of streams and Lambda Expressions in order to generate the administrator reports and Menu filtering available to the client
* Use of serialization and deserialization for storing the data of the application, applying it on the class DeliveryService.

**Problem analysis, modeling, scenarios, use cases**

**Problem analysis**

The purpose of the project is to create a realistic delivery service application, capable of storing, gathering and computing information about accounts, menu items, orders.

The application had to provide CRUD capabilities for the menu information, so that the administrator user can easily manage and modify the information within the application and database to their liking. For the client, the application had to provide a view of the available menu items and the ability to create an order, while the employee has to be able to receive a notification when a new order is being placed by a client.

**Problem requirements**

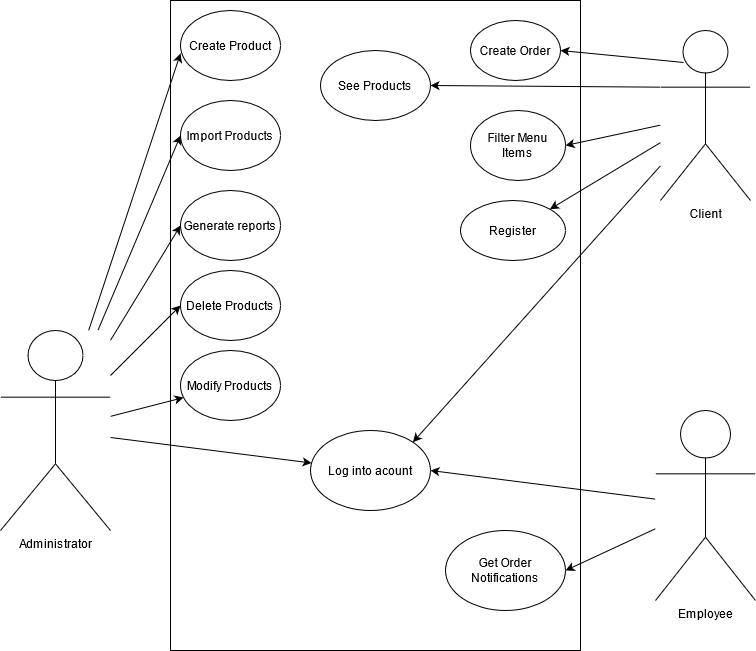
The functional requirements of this project are:

* The application should allow the administrator to insert, update, or delete Menu Items stored by the program, containing details such as: price, calories, fat, sodium, title and rating.
* The application should allow the administrator to create composite products, containing multiple Base Products, or even other, previously created, Composite Products.
* The application should also provide the administrator with a button for firstly importing the Menu Items data from the “products.csv” file.
* The application should allow generating four types of reports regarding order history of the delivery service.
* The application should show the client all Menu Items stored by it, and also allow their selection for ordering.
* The application should allow the client to place an order once the desired Menu Items have been selected.
* The application should notify the employee when a new order has been created by a client.
* The application should allow clients to register, and all users to log into their personal accounts.

The non-functional requirements include:

* The application should be intuitive and easy to use by the client, employee or administrator
* The user interface should have a friendly look, with pleasant color combinations and good placement of elements

**Use cases**



**Use case:** Create a new product

**Primary actor:** the administrator

**Main success scenario:** The administrator correctly inserts a product title, price, rating, calories, fat, protein and sodium and presses upon the Create button within the Administrator window. A new MenuItem will be created and inserted into the program data.

**Alternative sequence:** The information inserted is not correct or missing, and the administrator will receive a warning message when attempting to press the Create button.

**Use case:** Create a composite product

**Primary actor:** the administrator

**Main success scenario:** The administrator correctly inserts a product title, selects the list of products and presses upon the Create Menu button within the Administrator window. A new Composite MenuItem will be created and inserted into the program data.

**Alternative sequence:** The information inserted is not correct or no items were selected, and the administrator will receive a warning message when attempting to press the Create Menu button.

**Use case:** Import products

**Primary actor:** the administrator

**Main success scenario:** The administrator correctly presses upon the Import button in order to get data from the .csv file. Information is succesfuly retrieved and the menu list is populated with data.

**Alternative sequence:** The administrator fails to press the button and the import operation does not happen.

**Use case:** Generate reports

**Primary actor:** the administrator

**Main success scenario:** The administrator correctly inserts either start hour and end hour, number of times, minimal order price, or the day of the month and presses the appropriate button for generating a report fitting the information that has been introduced, and the history of orders present within the system.

**Alternative sequence:** The information inserted is not correct or missing, or the administrator presses the wrong button, and he will receive error messages.

**Use case:** Delete a product

**Primary actor:** the administrator

**Main success scenario:** The administrator selects the product (or products) to be deleted and presses upon the Delete button within the Administrator window. The selected items will be removed from the Menu in the system database and the administrator’s list of products will be also updated.

**Alternative sequence:** The administrator fails to select any products to be deleted or to press the proper delete button.

**Use case:** Update a product

**Primary actor:** the administrator

**Main success scenario:** The correctly inserts a product title, price, rating, calories, fat, protein and/or sodium and presses upon the Update button within the Administrator window, once the product to be updated has been selected.

**Alternative sequence:** The administrator fails to enter correct data or to select the product to be updated.

**Use case:** Create a new order

**Primary actor:** the client

**Main success scenario:** The client manages to select the Menu Items he/she wishes to order and presses on the Make Order button. A new order will be created with the current date and time saved and the employees will be notified when the order is successfully created.

**Alternative sequence:** The client fails to select the list of products to be ordered or press the Create Order button.

**Use case:** Register

**Primary actor:** the client

**Main success scenario:** The client inserts a new username and password in the appropriate fields and presses upon the Register button in the Log In window. A new account will be created and saved, and the user will be able to enter the client window.

**Alternative sequence:** The client fails to press the register button or enter username and/or password.

**Use case:** See Products

**Primary actor:** the client

**Main success scenario:** The client presses on the Products button. He/She will be presented with the entire list of Menu Items available for ordering

**Alternative sequence:** The client fails to press the correct button and nothing happens.

**Use case:** Filter Menu Items

**Primary actor:** the client

**Main success scenario:** The client selects the desired way of filtering the menu items, based on keywords, price, rating, calories, fat, sodium or protein and then enters the proper information to filter by in the text field. Finally, the search button is pressed and the client is presented with the final list of filtered Menu Items.

**Alternative sequence:** The client fails to enter search data or press the search button.

**Use case:** Get order notifications

**Primary actor:** the employee

**Main success scenario:** When a client places a new order, the employee is notified within his/her own user interface about the new order, and is able to take it and generate a new bill for the selected order(s).

**Alternative sequence:** The employee fails to see the notification or press the take order button.

**Use case:** Log into account

**Primary actor:** the user

**Main success scenario:** Any user (employee, admin or client) enters their personal username and password in the proper text fields within the log in window and press upon the log in button. Then the according window will open (for client, employee or user) if the username-password combination is found within the program database and is correct.

**Alternative sequence:** The user fails to enter the correct username – password combination or to press the log in button, so the logging in does not happen.

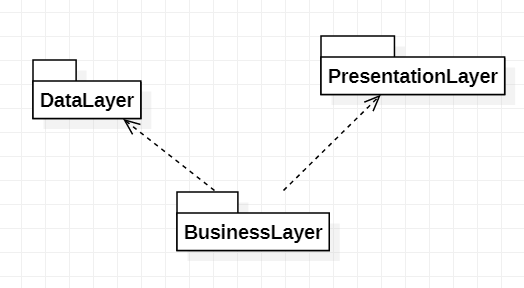
**Design**

By respecting the Object-Oriented design, I have divided the application into multiple classes, each with a cerain role:

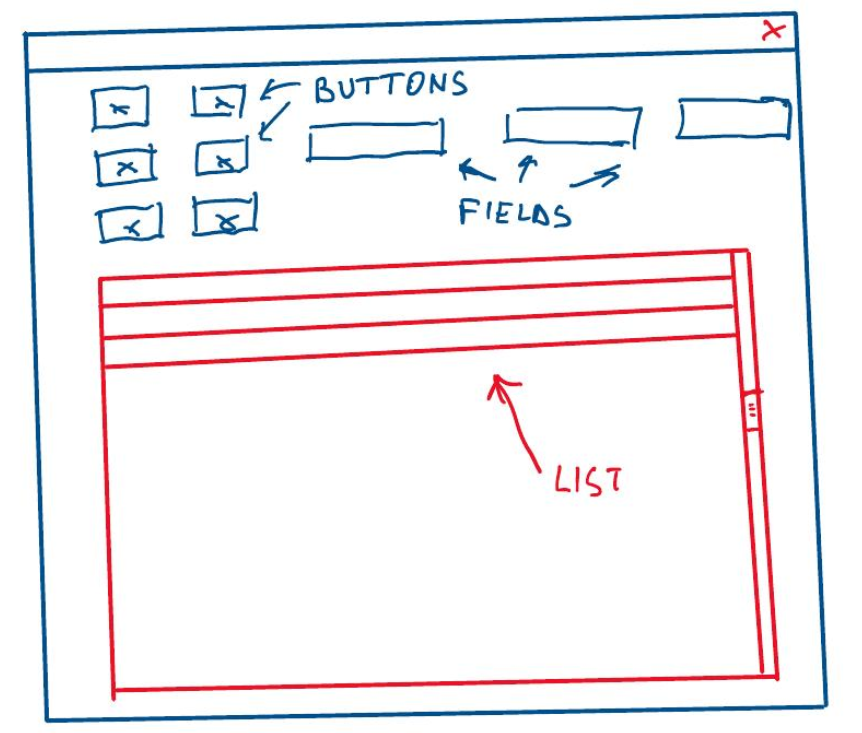
* BusinessLayer – this package is responsible for the main logic and computation behind the system, performing many operations that are essential to the program
* DataLayer – this package has the occupation of storing data from the application, or retrieving it (through deserialization)
* PresentationLayer – this package manages the user interfaces, action listeners and the user – application interaction for all operations and functionalities

**Diagrams**

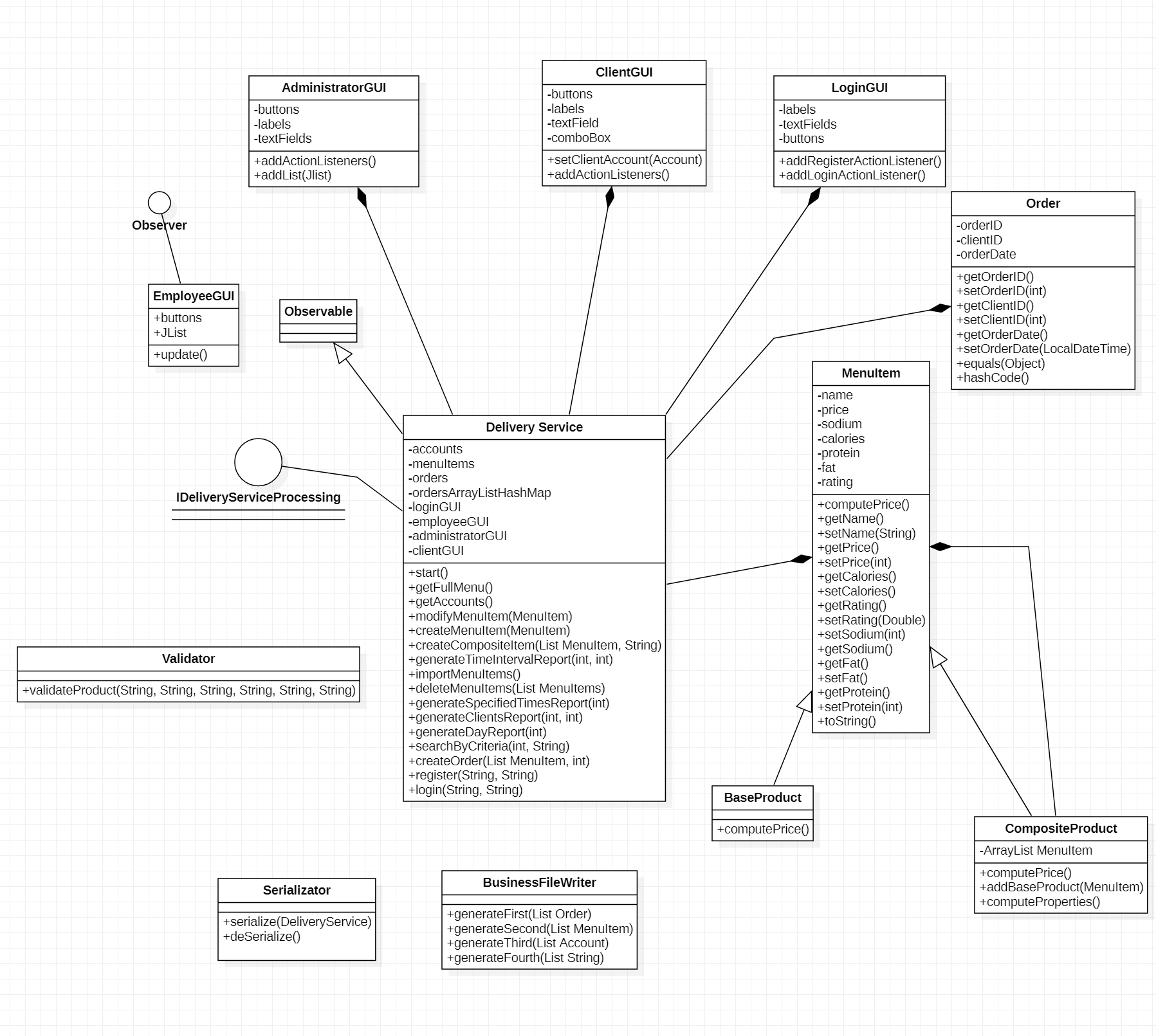
**Package Diagram**



**GUI Design**



**Class Diagram**



**Data Structures**

The most important data structures used were ArrayList for storing the products, accounts and orders, Jlist for presenting the list of Menu Items and HashMap for binding the orders with the list of ordered products.

**Algorithms**

Generating a report of the most loyal clients

Input: The order list, the accounts list and the order and product list HashMap

Output: the list of some of the most loyal clients

Algorithm: Firstly, the list of orders is filtered, leaving only those whose price exceeds the minimum imposed. Then, for the remaining orders, a new Map is created, storing each client ID with the number of orders created. From this map, only the clients exceeding a certain number of orders are added to the final list, which will be sent to the report writer.

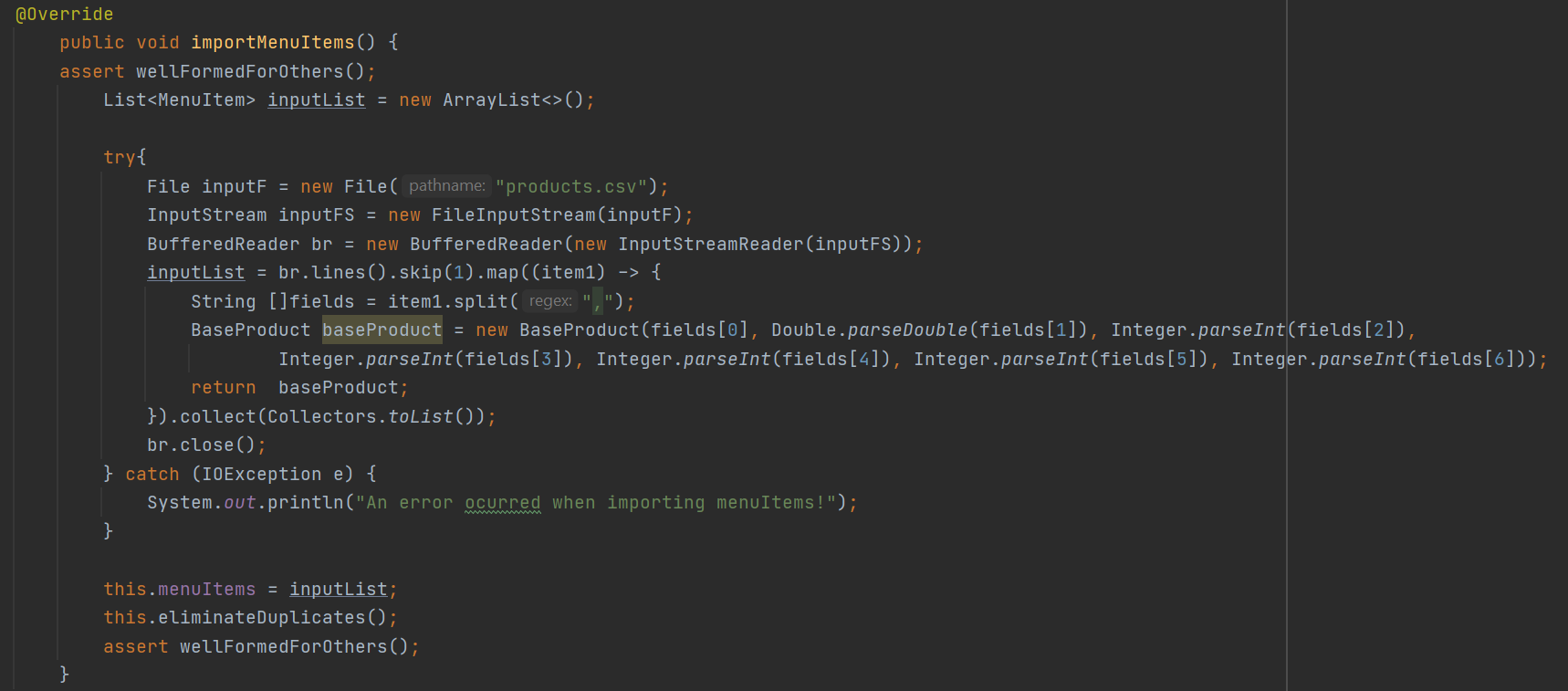
**Implementation**

**The MenuItem class**

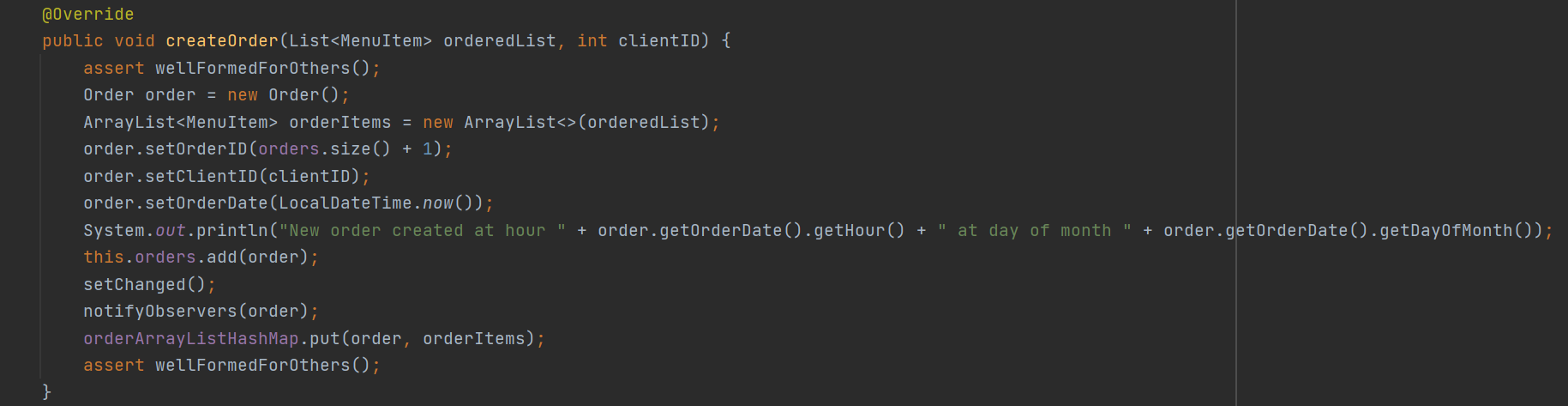
This abstract class has the role of modelling the generalized menu item structure, being a superclass for both the BaseProduct and CompositeProduct classes. This grouping implements the Composite Design Pattern, which allows the combination of multiple Menu Items (either simple or composite) into a single one, proving to be incredibly versatile for the various operations and functionalities of the application. The method computePrice() is overriden by the two subclasses, allowing each of them to operate individually, while also keeping the chemistry of the design pattern.

**The DeliveryService class**

This class has the main role of providing the computations and the logic behind the possible operations available to the administrator and the user, defined within the IDeliveryServiceProcesing interface, whose methods it implements. It receives calls from the user interface, and provides the required operations in response. Besides, the class contains the major data structures used by the application, being the one which uses them the most and therefore also being a serializable class.



The importMenuItems() method has the crucial role of retrieving data from the .csv file, and splitting it into the 7 fields present in the MenuItem data structure : title, price, rating, calories, protein, sodium and fat.



The createOrder(List<MenuItem>, int) method provides the computations behind the functionality provided to the user to make a new order. Once this event has happened and the method is called, it also notifies the Observer, which is the employeeGUI class about the new order, so the employee can instantly see the newly placed order.

Methods in this class:

* deleteMenuItems(List<MenuItem>) : removes the selected items from the database
* createCompositeItem(List <MenuItem>, String) : creates a new CompositeItem, containing the list of Menu Items given as a parameter
* methods for generating the administrator reports

**The Order class**

This class models the data structure used for storing information regarding the clients’ orders. It contains the ID of the client and of the order, along with the date and time when that particular order was placed.

It also overrides the equals() and hashcode() methods in order to be used as a key for the HashMap implemented in the DeliveryService class.

**The User Interface classes**

These classes provide the interface for the administrator, client and employee users, with buttons and text fields for the various functionalities implemented.

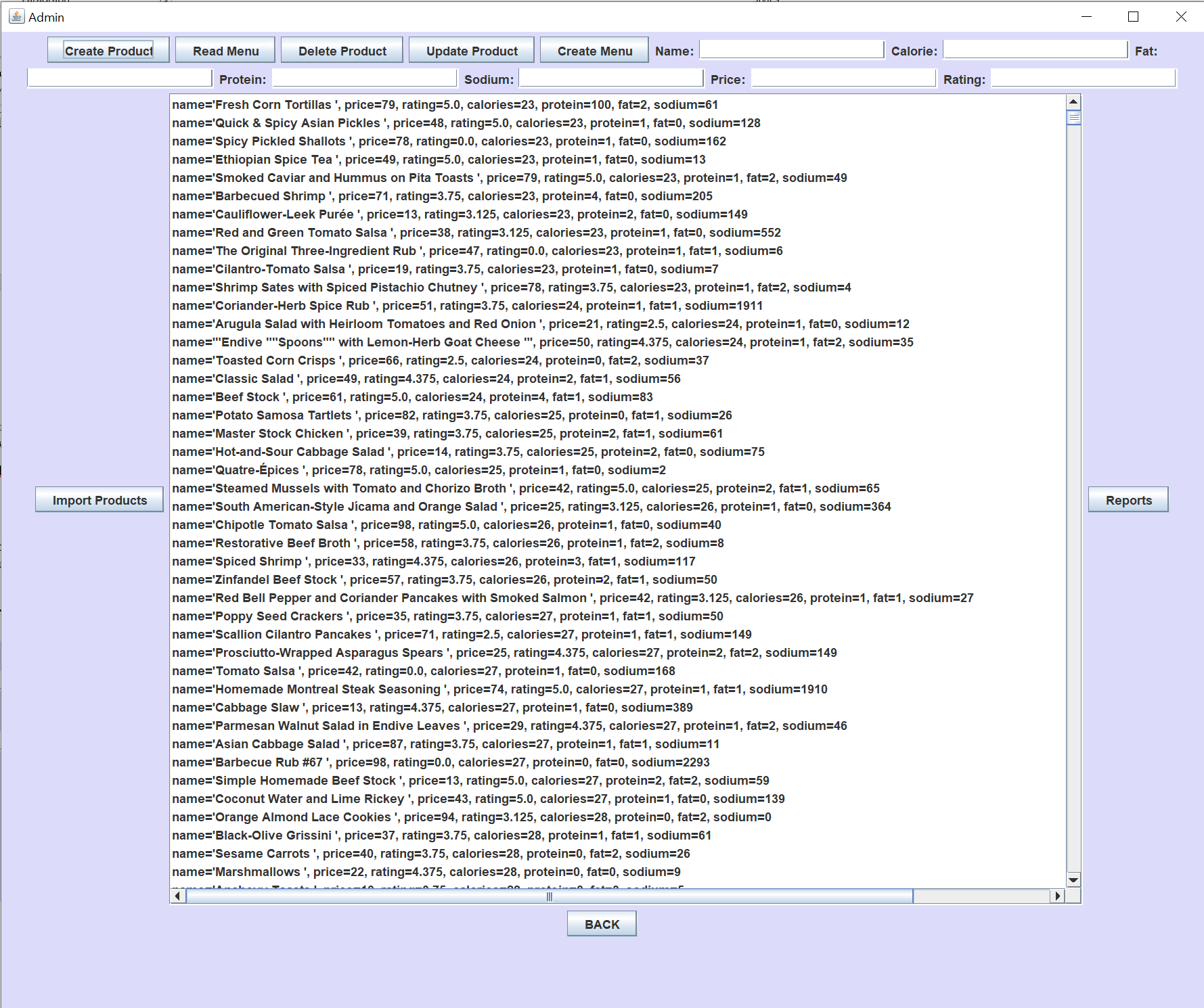
The log in interface offers the user the chance to enter their personal account, or register as a new client.

The Employee interface receives notifications when a new order is being placed and also offers the employees the chance to take the order and a bill will be generated in the text file.

The Client interface offers the option to filter out Menu items based on specific properties, or read the Menu entirely. Besides, it gives the option to the client to place a new order.

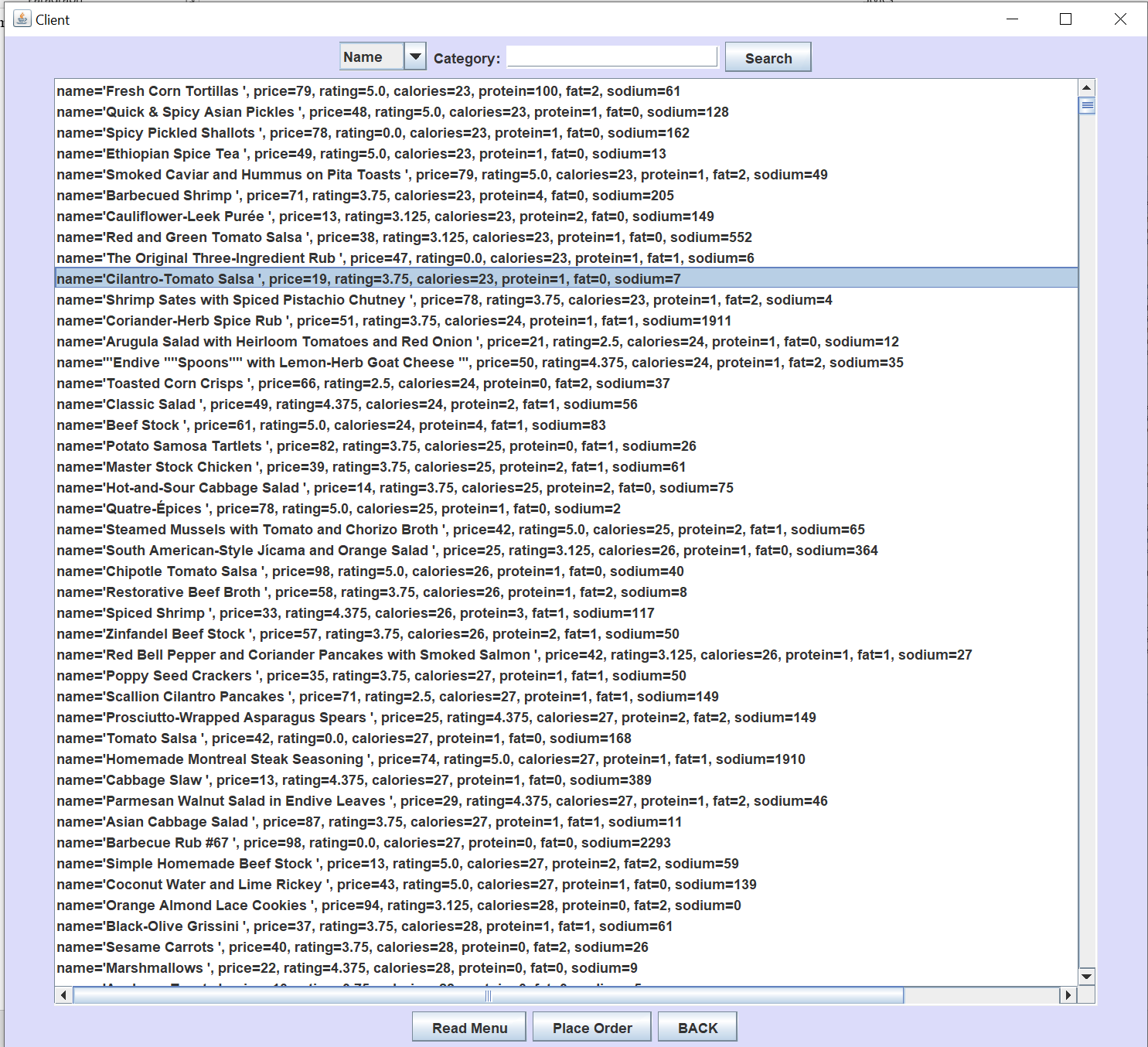
The Administrator interface is considerably more complex, containing multiple buttons and text fields for the operations available to be done on the Menu Items. Also, when the reports button is pressed, the user is greeted by a new window where special data regarding the reports can be entered, and the administrator can decide which reports to be generated by pressing the according button.



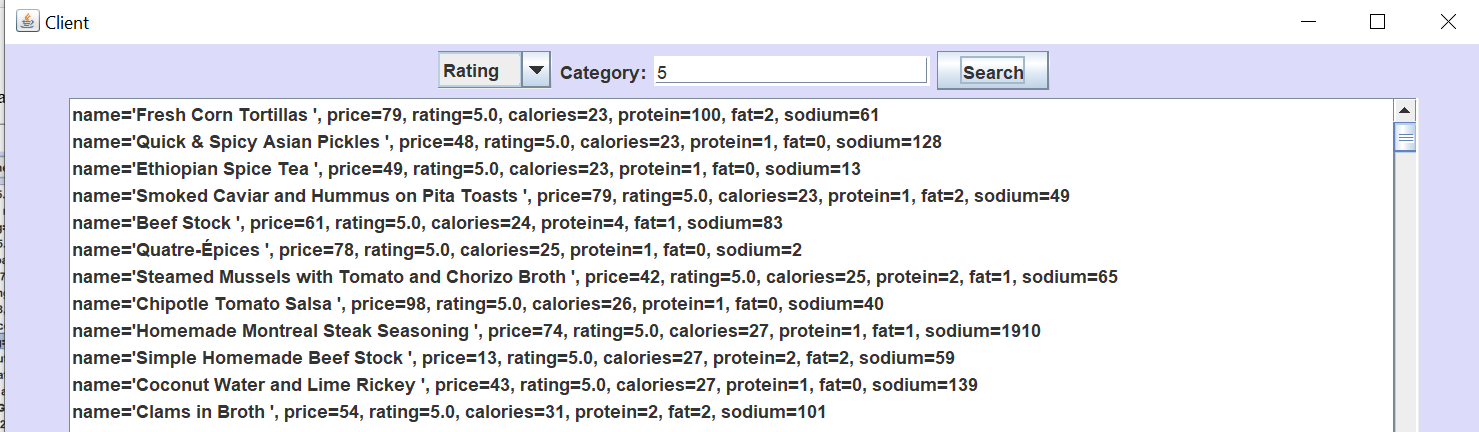
 

**Results**

When pressing the Read button, we are provided with the List containing the database items:



When a filtering category is selected and a filter is applied, the proper results are shown:



When a client makes an order, the employees are notified:



**Conclusions**

This project has also proven to be another great occasion to learn and experience many new things, such as working with serialization, lambda expressions, streams, further improving my knowledge and abilities regarding the construction of a Graphical User Interface and printing data into a .txt log file, but also importing data from a .csv file.

Regarding improvements, there could be added a many more validators in order to provide better safety when the user does not use the application properly. Besides, the Graphical User Interface could definitely use many improvements, in order to be more friendly to the user, more pleasant to use and understandable. The inclusion of a database such as MySQL for storing the accounts, orders, Menu Items and others could also possibly be more practical than the Serialization method.

**Bibliography**

1. [**https://stackoverflow.com/**](https://stackoverflow.com/)
2. [**https://www.tutorialspoint.com/java/index.htm**](https://www.tutorialspoint.com/java/index.htm)
3. [**https://www.draw.io/**](https://www.draw.io/)
4. [**https://www.programcreek.com/2011/03/java-write-to-a-file-code-example/**](https://www.programcreek.com/2011/03/java-write-to-a-file-code-example/)
5. [**https://www3.ntu.edu.sg/home/ehchua/programming/java/J4a\_GUI\_2.html**](https://www3.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI_2.html)
6. [**https://www.w3schools.com/java/java\_files\_create.asp**](https://www.w3schools.com/java/java_files_create.asp)
7. [**https://www.baeldung.com/java-jdbc**](https://www.baeldung.com/java-jdbc)
8. [**https://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html**](https://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html)
9. **javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.html**
10. [**https://www.baeldung.com/javadoc**](https://www.baeldung.com/javadoc)
11. **https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html**
12. **Programming Techniques – Lectures of Dr. Prof. Ioan Salomie**