Technical University Of Cluj-Napoca

Programming Techniques

Assignment 4

Delivery Service



Student: Bogdan Tudor-Alexandru

Group: 30424

Teacher: Ioan Salomie

Teacher Assistant: Pop Cristina

**Main Objective**

Design and implement an application aimed to manage administrators, employees and clients in a restaurant (delivery service). The main goal is to design the solution and to implement a delivery service application.

**Sub-Objectives:**

* Analyze the problem and identify the requirements
* Design the application
* Implement the application
* Provide the user with an easy way to login as an admin and to perform the operations associated with it: display the menu, create a new base or composite products, provide a way to add/edit/delete an existing menu product, generate reports regarding the existent orders
* Provide the user with an easy way to login as a client and to perform the operations associated with it: display the menu, create a new order and filter the menu objects according to some filters introduced by the user
* Provite the user with an easy way to login as an employee and see when there are new orders placed by customers.
* Generate .txt bills containing information about the newly generated order

**Project Specifications**

1. **Problem Analysis**

The delivery service holds clients, orders and menu items and receives new orders and commands from new clients. The application provides client operations, admin operations and employee operations (which are notifications to be more precise).

1. **Functional Requirements**

The client is able to register if he does not already have an account, or log in if he has one with the existing username and password. Afterwards, it can view the menu items and filter them by the fields of choice by completing the text box associated with a product’ fields (title, rating, calories, protein, fat, sodium, price) and pressing the button. Moreover, the client can choose to place a new order by selecting the wanted products, and when finished, signaling it by pressing the button to finish the order. When a new order is created, a bill containing the order’ information is created and stored in a .txt file. At the same time, the employee is notified by using the Observer – Observable classes from Java (the notifying methods) about the newly created order, and in its interface it displays information describing it.

The employee can log in within the system using an existing username and password. As mentioned above, the employee is notified every time a new order is placed by a client and it displays information regarding the order and the total price of the order.

The admin can log in within the system using an existing username and password. The admin can choose to show the menu items, or do other operations. One can add, modify or delete and existing product from the menu, and is notified if the operation failed, or can create a new composite item containing more base items (for example, create a menu containing soup, steak and desert). Moreover, the admin can generate reports regarding the existent orders in the delivery service, which are four in number. The first report is about orders which took place at a time between a specified first hour and last hour, the products ordered more than a specified number of times so far, the clients that have ordered more than a specified number of times and the value of the order was higher than a specified amount, and the last one about the products ordered within a specified day with the number of times they have been ordered.

1. **Uses Cases**

In the application there are 3 types of actors: Administrator, Client and Employee.

**The Administrator:**

* Is able to import/add/delete/edit products(MenuItems)
* Is able to generate reports based on given data, with regards to Clients, Orders and Products.

**The Client:**

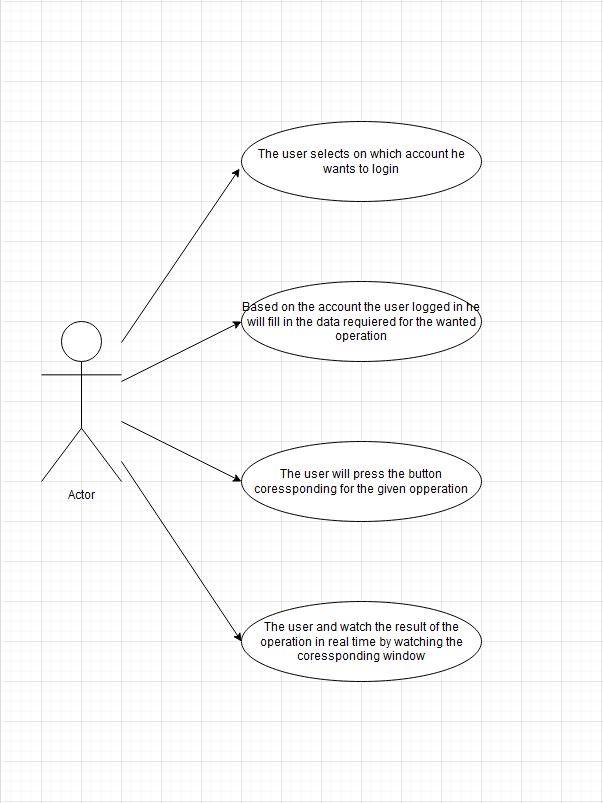
* Can search for products freely or based on filters.
* Can selecte products and place an order based on the selected products and the current local date and time.

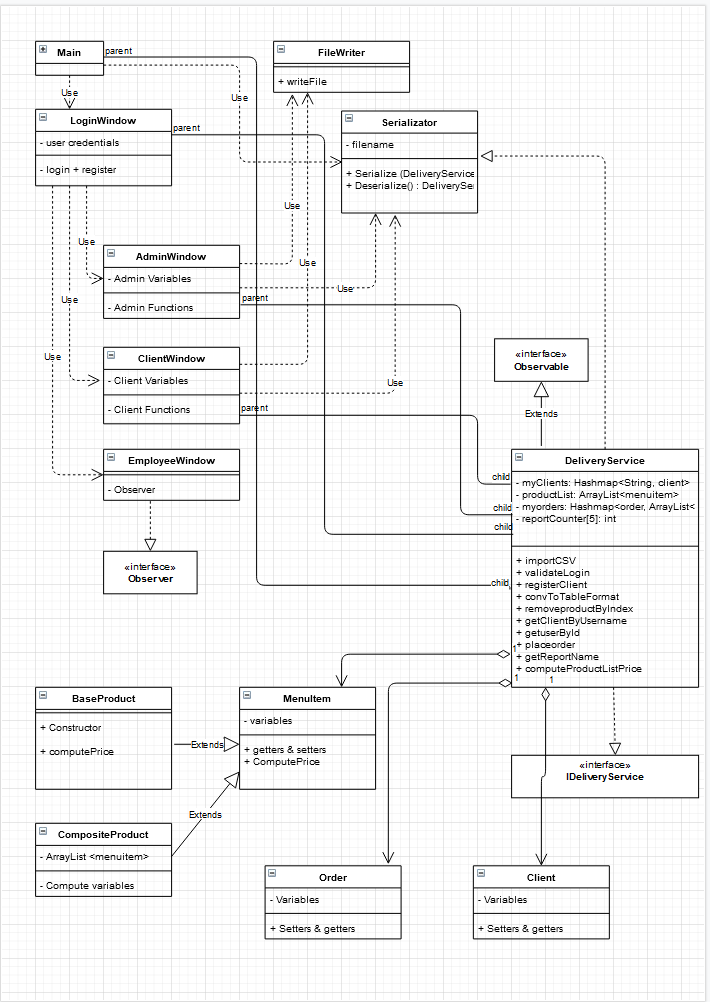
**The Employee**

* Is notified when an order is placed

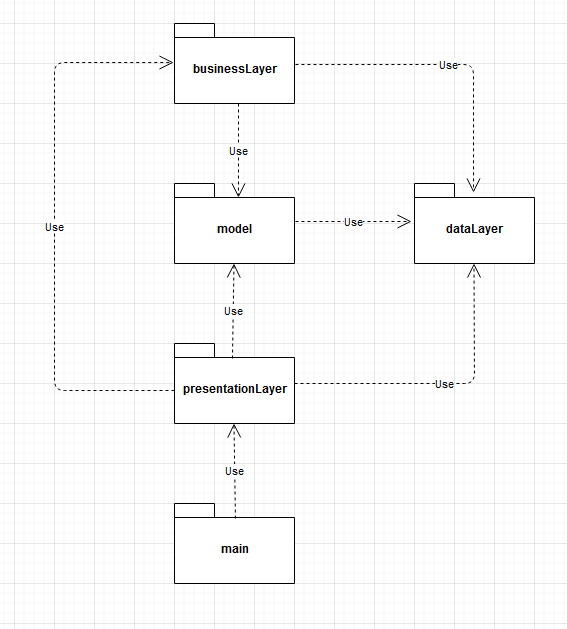
1. **Design**

**Use Case Diagram:**



**Class Diagram:** ****

**Package Diagram:**

****

**Packages:**

* Main class: the one that starts the application
* Presentation layer: the classes that compose the graphical user interface (GUI): LoginWindow, AdminWindow, ClientWindow, EmployeeWindow
* Model layer: it contains the classes that represent the main objectives of our program, the basic data: Client, Order, MenuItem which extends into BasicProduct and CompositeProduct
* businessLayer: it contains that classes that implements the logic of the application, the DeliveryService class with it’s interface.
* dataLayer: it contains the classes that represents the saved data: the serialized information that will be deserealized at the beginning of the app to load it, the Serializator and the FileWriter which saved the bills and the reports.

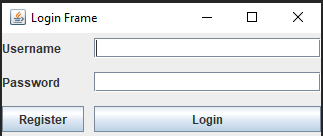
**Data Structures:**

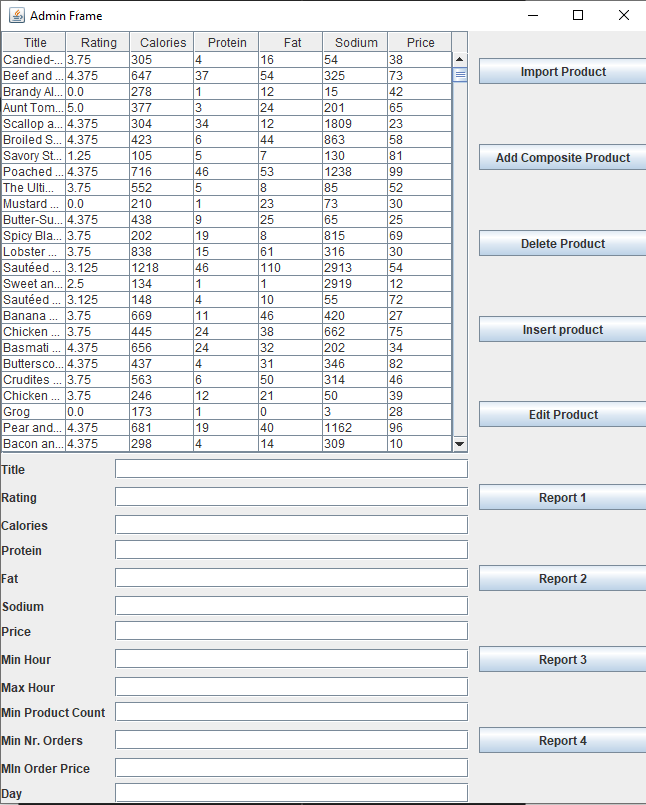
The data structures used in the application are mostly primitive ones which compose the basic data of the app(Client, Order, MenuItem, BasicProduct, CompositeProduct), but there are also some “more special ones” like the HashMap. The HashMap it’s a data structures that is used in optimizing applications by keeping for a key a certain set of values. What makes this type of data structure special is the fact that it has a very small search time for a given key, based on the hashing function (an encoding function to simplify it) that it’s using to encode the keys and to find them fast when requested so.

**User Interface:**

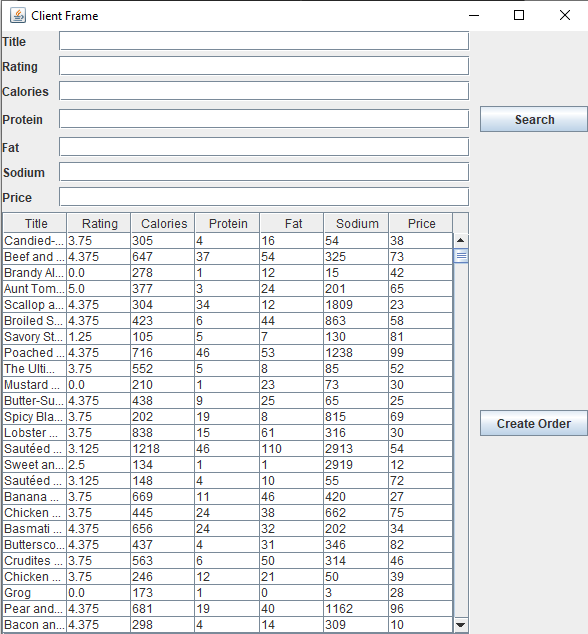
The GUI (graphical user interface) is what the user of the application sees for the first time when running the application. To make the application easier to use and reduce the mistakes made by the user, the user interface must be user-friendly. The graphical user interface of my application has been done using JavaSwing with GUI forms. It contains a main window (frame) which, by logging in using the right credidentials will open a new window (for admin ,employee or client).

**Login Window:**

 The login window is the first wondow opened by the application and it is used by all the users to log in in their respective account. For the administrator and employee there already exists accounts, while for the Client, if he already has an account he can directly log in, else he will have to register his account first by pressing the register button. In case that the login credentials are wrong a pop-up dialog will show to inform the user.

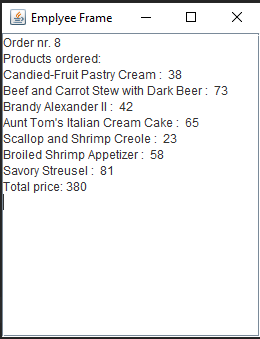
**Admin Window:**

The admin is the main controller for the application. The admin can log in into their account using the “**administrator**” username and password. When loggin in he will be able to perform different operations in the application: import the products from csv file, insert/delete/edit products, create new composite products based on existing menu items, or generate reports based on given criterions using the 4 reports buttons and the respective fields. The reports will be generated in text files.



**Client Window:**

The client will have to register first and then use the registered username and password to login into the application. The client will be able to search for products based on several filters which can be inserted into the empty fields, or leave them empty if the user doesn’t want to use the respective field. To refresh the data in the table just do an empty filter search. To create an order the current date and time are used and the selected products by the user. To select more products the user has to use CTRL + RIGHT CLICK. Each order will generate a bill that is saved by Order\_id.

**Employee Window:**

The employee will have a window with a text area where the new orders will pop-up with the products ordered and the total price. The employee cand login into the corresponding window by using the “**employee**” username and password.

**Implementation:**

The implementation is done using and object oriented approach, with object oriented concepts, the most remarkable ones being serialization, the composite design pattern and the observable-observer technique. Each class is enrolled in a coressponding package to fulfill the Layered Architecture.

1. Model classes: Client, Order, MenuItem, BaseProduct, CompositeProduct

They are composed from setters, getters and constructors and helps quering data from the database through the reflection technique. The MenuItem class is extended into the base product and the composite product classes.

1. Presentation Layer: is composed from the 4 classes, which corresponds to the 4 typed of wyndows that the program has: LoginWindow, AdminWindow, EmployeeWindow, ClientWindow. Except the constructors and the getter for the Panels corresponding to each window, there is a load method used in the ClientWindow to load the data from a given list in the panel, this methos being used on the filtered list to be presented to the client.
2. Data Layer: it contains the classes used to save/write the data and the information into files. The classes are the Serializator and the FileWriter. The Serializator has 2 methods:  
    - Serialize: which loads the data from the program into a txt file such that it can be loaded later, when the application starts.  
    - Deserialize: which convers the data from the txt file into the corresponding object: the DeliveryService at the start of the application.  
   The FileWriter has only a method that receives as a parameter 2 Strings: the file name and the content that must be put into the file and it write the given data.
3. The BusinessLayer: it contains the logic of the application. I would say that it is the most important layer in the entire application. It is composed only from the DeliveryService class with it’s interface. The main methods are:
   * Public void importCSV(String inputFilePath) – used in importing the products from the CSV file.
   * public String validateLogin(String user, String pass) – used to check if the login credentials are valid
   * public String registerClient(String user, String pass) – used to register a new Client
   * public String[] convItemToTableFormat(MenuItem x) – converts a MenuItem to String Array to be inserted into the table to be shown to the user
   * public ArrayList<String[]> convToTableFormat() – converts the entire product list kept inside the DeliveryService Class to the respective format to be show to the users in the table.
   * public ArrayList<MenuItem> getProductList() – retrives product list
   * public HashMap<Order, ArrayList<MenuItem>> getMyOrders() – retrives order HashMap
   * public void removeProductByIndex(int index) – removes product given by the transmitted index
   * public MenuItem getProductByIndex(int index) - retrieves product given by the transmitted index
   * public Client getClientByUsername(String user) – retrives Client through the given user
   * public void listProd() – list products
   * public void listUsers() – list Users
   * public void listOrders() – list Orders
   * public void placeOrder(Order myOrder, ArrayList<MenuItem> orderedProducts) – Place an order into the hashMap
   * public void notifyEmployee(String bill) – sends notification to employee that an order has been placed
   * public int computeProductListPrice(ArrayList<MenuItem> myProductList) – computes the prive of the given list
   * public String getUserById(int id) – retrives user by the given id.
   * public String getReportName(int type) – created the report name for the file that the report will be written into based on the type the requested report is.

**Conclusion:**

I can certainly say that this was the most complex project I have ever worked on, and even though it was not easy I learn lots of things that I think that will help me in the future. For me this project was an opportunity to realize what lays at the foundation of the big applications and startups that rules the food delivery chains in the big cities, and what apps like “FoodPanda”, “BoltFood”, “Clever” started from. I think that this kind of application has lots of potential to be developed furthermore, the products being extended from food to different categories that people use in their daily lives. Even thought for the last applications I used JavaFX to code the graphical user interface, this time I used JavaSwing, just like in the first project, and I managed to use something new, such as the JavaSwing GUI form, which made my job much easier. Like this I was able to work on the front end and back end at the same time, taking the project from nothing to the end result, but always being able to test each step I took when writing the application.

**Bibliography:**

<https://www.baeldung.com/java-composite-pattern>

<https://www.geeksforgeeks.org/composite-design-pattern/>

<https://www.tutorialspoint.com/design_pattern/composite_pattern.htm>

[Lambda Expressions (The Java™ Tutorials > Learning the Java Language > Classes and Objects) (oracle.com)](https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html)

[Method References (The Java™ Tutorials > Learning the Java Language > Classes and Objects) (oracle.com)](https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html)

[Processing Data with Java SE 8 Streams, Part 1 (oracle.com)](https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html)

[Java 8 Stream Tutorial (winterbe.com)](https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/)

[Java - Serialization - Tutorialspoint](https://www.tutorialspoint.com/java/java_serialization.htm)

[Streams in Java: An In-Depth Tutorial with Examples (simplilearn.com)](https://www.simplilearn.com/tutorials/java-tutorial/streams-in-java)