**DOCUMENTATION**

ASSIGNMENT 4

STUDENT NAME: LACATUS ARPAD ALEX

GROUP: 30423

# CONTENT

[1. Objective of the assignment 3](file:///C:\Users\Administrator\Downloads\Documentation.docx#_Toc95297885)

[2. Problem analysis, modeling, scenarios, use cases 3](file:///C:\Users\Administrator\Downloads\Documentation.docx#_Toc95297886)

[3. Design 4](file:///C:\Users\Administrator\Downloads\Documentation.docx#_Toc95297887)

[4. Implementation 5](file:///C:\Users\Administrator\Downloads\Documentation.docx#_Toc95297888)

[5. Results 6](file:///C:\Users\Administrator\Downloads\Documentation.docx#_Toc95297889)

[6. Conclustions 6](file:///C:\Users\Administrator\Downloads\Documentation.docx#_Toc95297890)

[7. Bibliography 6](file:///C:\Users\Administrator\Downloads\Documentation.docx#_Toc95297891)

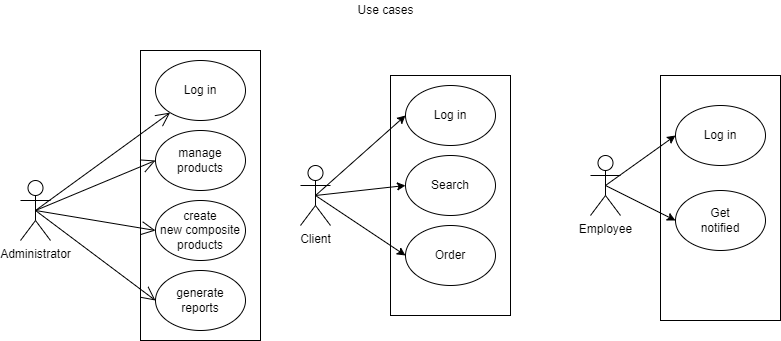
1. The Objective of the assignment - The main objective of this assignment is to design and implement an application for managing a food delivery system by implementing multiple account for administrator, client and employee with different functionalities such as creating new items, ordering items, creating bills and reports. Some secondary objectives are:

-analyzing the problem and identifying the requirements

-designing the log in feature that helps us implement our solution

-implementing a delivery service able to handle all our requests as simple and efficient as possible

-testing the food delivery system implemented in various cases

1. Problem Analysis, modeling, scenarios, use-cases

First of all we need to consider all the possible steps our app could go through and what methods we require in order to implement this app as efficient as we could and as user friendly as possible. We will need a lot of classes and methods to make the implementation easy to understand, work with, and to solve better the bugs that may occur. Another thing to consider is that our implementation of the modeling classes should mirror the tables in our database to create a proper connection.

**Analyzing** the problem, we have the following requirements:

**For inserting:**

-the application should allow administrator users to insert a new menu item with data they wish for (with some restrictions)

-the data structure used for storying the menu items should be updated with each insertion

**For updating:**

-the application should allow administrator users to modify an existing menu item with data they wish for (with some restrictions)

-the data structure used for storying the menu items should be updated with each modification

**For importing:**

-the application should allow administrator users to import an existing set of menu items from a different file with data they wish for (with some restrictions)

-the data structure used for storying the menu items should be updated with each import

**For deleting:**

-the application should allow administrator users to delete an existing menu item they wish to

-the data structure used for storying the menu items should be updated with each deletion

**For creating:**

-the application should allow administrator users to create an composite menu item with other menu items they select from the current menu

-the data structure used for storying the menu items should be updated with each creation

**For searching:**

-the application should allow client users to search for specific menu items by providing filters and keywords

**For ordering:**

-the application should allow client users to create a new order by selecting a few menu items from the current menu item set

-the data structure used for storing the orders should be updated after each order

**For notifying:**

-the application should allow employee users to see each order and get notifyed when there is a new one

For **modeling** our data, we use mainly the Client class, Administrator class and the Client class. Alongside them we have the other classes that work with these two and orchestrate the actual application and provide accessibility for these 3, delivery service class. The methods implemented in those three are mirrored here and given a higher functionality.

**Use cases**: add/delete/update/create menu item

The primary actor is the administrator user, and the **main scenario** would be:

1. The logs in with the correct credentials for an administrator
2. The user chooses the operation
3. The user inserts the values for the fields required (i.e., title, rating, etc.) and makes sure the data is valid
4. The user clicks on the “do” button to go ahead with the operation
5. The application updates the table and the database with the desired input, if we work with orders also creates a new receipt

**Alternative Sequence:** Invalid values for parameters

- The user inserts incorrect parameters in the fields designated

- No change will happen

**Use cases**: create report

The primary actor is the administrator user, and the **main scenario** would be:

1. The logs in with the correct credentials for an administrator
2. The user chooses the kind of the report
3. The user inserts the values for the fields required (i.e., parameter1, date, etc.) and makes sure the data is valid
4. The user clicks on the “generate” button to generate the reports
5. The application creates the report requested for

**Alternative Sequence:** Invalid values for the report parameters

- The user inserts incorrect numbers for the report method.

- An exception will be thrown presenting the problem

**Use cases**: search for menu item

The primary actor is the client user, and the **main scenario** would be:

1. The logs in with the correct credentials for a client
2. The user chooses the filters he prefers by checking some check boxes
3. The user inserts the values for the field keyword and makes sure the data is valid
4. The user clicks on the “search” button to get the requested filtered menu

**Alternative Sequence:** Keyword inexistent entered

- The user inserts an inexistent keyword

- The application will display an empty table

**Use cases**: order menu item

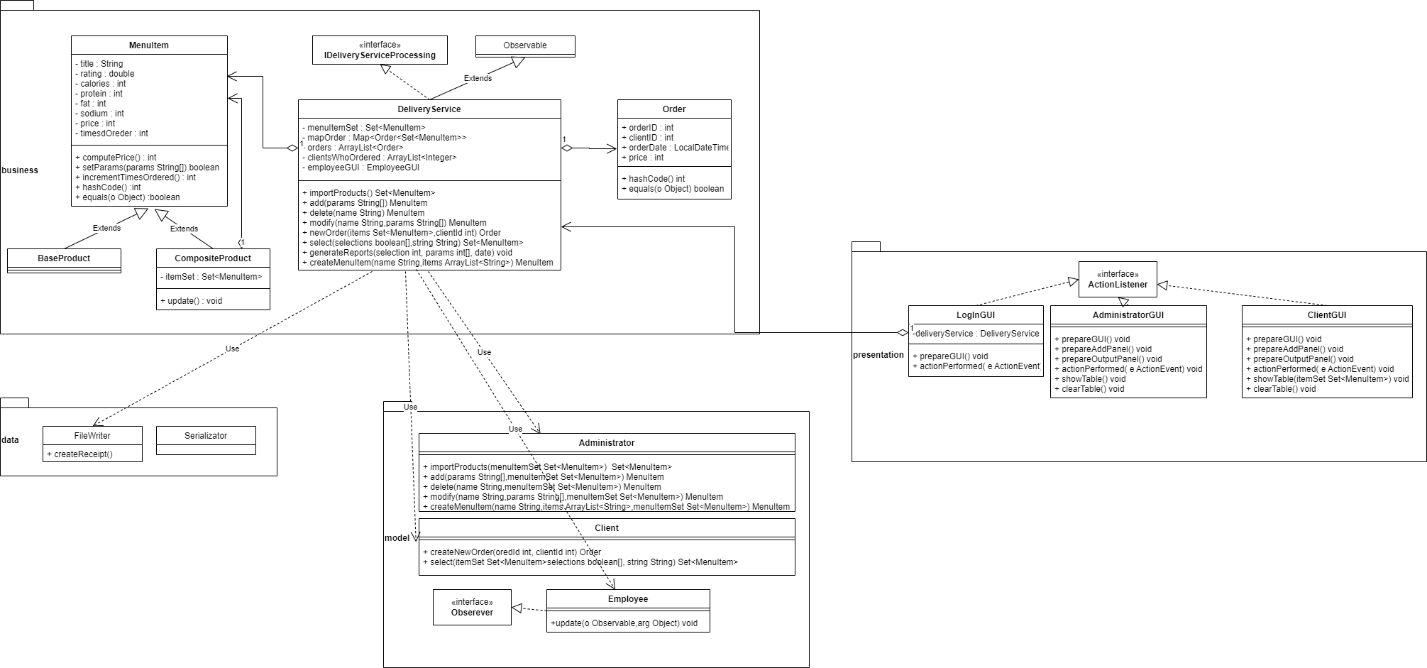
The primary actor is the client user, and the **main scenario** would be:

1. The logs in with the correct credentials for a client
2. The user fills a string field by choosing his desired dishes
3. The user presses on order
4. The application will generate a bill with requested items

**Alternative Sequence:** Dish inexistent entered

- The user inserts an inexistent dish

- The application will not add anything on the bill

1. Design

Leve 1: System design:

At this level we have the application itself that has as the inputs the parameters inserted in the fields that we want to work with and the chosen table by the user and the resulted table as the output in the GUI , to note that for working with orders we also have the output files that are the receipts.

Level 2: Design into sub-systems/packages:

We will need 3 main packages for this assignment:

-Graphical User Interface (presentation) – the package that contains the classes used for implementing the graphical user interface, here we have a class that manages them LogInGUI, and the three that mirror each type of user and give them specific functionalities

-Data Models (models) -the package that contains the classes modeling the application data (Administrator, Client, Employee), here we have the main classes used for implementing the users form the app so they need to be mirroring the GUIs presented before.

-Business Logic(bussines) – the package that contains the classes used for running the application. These classes are used at the higher level and they contain all the required information, and methods for working with the database and validating the inputs thanks to their components. Here we have the classes MenuItem, DeliveryService. We also have a modeling class Order used for implementing the clients order.

-Data Access(data)- the package that contains the classes responsible with implementing the methods for working with the FileWriter (FileWriter,Serializator). This classes ease working with external files

Level 3: Division into classes

The 3 main classes used are Administrator, Client , Order used for modeling data used used with GUI parts and delivery service class. Other classes used are the ones used are there for small functionalities such as the file writer class and employee with its GUI.

Level 4: Division into routines

All the classes implemented are divided in routines in order to facilitate our work and make the code much easier to understand, the methods are implemented in such a way so we don’t have unnecessary long routines with hundreds lines of code in order to not make things too complicated.

Level 5: Internal routine design

In each routine our code is well delimited in pieces of code with different functionalities, which combined give us a final routine easy to follow and comprehend.

1. Implementation

Data Model classes:

**Administratort:** The class used for modeling the administrator user. Here we have implemented the methods add, modify, delete, create new composite menu, import, and generate report. The methods are self-explanatory requiring some parameters to return the items and realise what s asked for. Generate report method uses lambda expressions and stream processing for filtering. also requiring a select to specify the kind of report needed.

**Client:** This class is used for modeling the client user. The methods implemented here are create new order and search. The search methods uses lambda expressions and stream processing to filter the searched items while create new order returns a new order

**Employee:** The class used for modeling the employee user. Implements observer, thus the method update will be called each time the delivery service notifies a change. This method updates the table from the employeeGUI.

Logical classes:

**DeliveryService:** The class that creates the connection between GUI and the modeling part. Here we have mirrored the methods implemented in the modeling classes implemented with the help of the data structures (Map ,Set, List) used for storing orders and menu items.

**Order:** This class models the order created by an client by storing the required information about an order and being hashable in order to be stored in a set.

**MenuItem:** Class used to model the menu items that can be find in our menu. It has 2 extensions base product and composite product. The first one being a basic extension with no notable changes while the second having an array of menu items being composed of several menu items. The composite extension has the attributes define a bit differently. These classes are has able in order to be integrated in sets.

Data Access classes:

**FileWriter:** The class that provides easier methods of working with the printer and files.The method createReceipt creates the file used as a receipt and fills it with required information.

GUI classes:

**LogInGUI:** Class representing both a view and a controller. Is the first interface that pops and lets the use access an administrator, client or employee account. Its only method is action performed that takes the strings provided and prepare the next GUI.

**AdministratorGUI:** Class used for creating the GUI for an administrator user. This class has the action performed method that takes the parameters provided by the user to choose an operation from import/add/delete/modify/create composite and do the task or to chose a type of report and print it.

**ClientGUI:** Class used for creating the GUI for an client user. This class has the action performed method that takes the parameters provided by the user to choose an operation from search and create order. This operations are handled by the delivery service by giving the required parameters.

**EmplpoyeeGUI:** Class used for creating the GUI for an client user. This class has only a table and with the help of the observer design the table is updated in real time.

1. Results

The testing was done manually by introducing some random various inputs in order to test the functionality of the program in all the GUIs created and by checking the resulted files.To note that more than 1 GUI can be opened at any time. It was discovered that the program works as intended and there were no major bugs reported.

.

1. Conclusions

To conclude, I found this homework quite enjoyable and a good practice for working with lambda expression, stream processing. I consider that the final application meets most if not all the requirements presented. I think this assignment really helped me gain some experience working with bot GUI and databases and I am thankful for this opportunity.

1. Bibliography

For this project I mainly use only the resources provided at the class, I found them easy enough to grasp and implement so I did not need the help of other resources whatsoever. The links of the resources are:

<https://dsrl.eu/courses/pt/materials/A4_Support_Presentation.pdf>

<https://dsrl.eu/courses/pt/materials/PT2021-2022_Assignment_4.pdf>

For the documentation the documentation template provided was used