



**UNIVERSITATEA TEHNICĂ**  
DIN CLUJ-NAPOCA

PROGRAMMING TECHNIQUES  
ASSIGNMENT 4

# FOOD DELIVERY MANAGEMENT SYSTEM

**STUDENT:** GYARMATHY CSABA

**UNIVERSITY:** TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

**FACULTY:** COMPUTER SCIENCE

**TEACHER:** CRISTINA BIANCA POP

**TEACHER ASSISTANT:** ANDREEA VALERIA VESA

**YEAR:** 2

**ACADEMIC YEAR:** 2021/2022

**GROUP:** 30424

## Contents

Assignment Objective .....	3
Project analysis, scenario, approach and use cases .....	3
Analysis .....	3
Scenario and modelling.....	4
Use case scenarios.....	7
Implementation .....	7
Diagrams .....	7
Class diagram.....	7
Use case diagram .....	8
Packages.....	9
Business Logic Layer (BLL).....	9
Data Access Object (DAO).....	9
Presentation.....	10
Further development .....	10
Conclusion .....	10
Bibliography .....	11

## 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 from the menu (an example of composed product could be named "daily menu 1" composed of a soup, a steak, a garnish, and a dessert).
- Generate reports about the performed orders considering the following criteria:
  - 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.
  - the products ordered more than a specified number of times so far.
  - the clients that have ordered more than a specified number of times so far and the value of the order was higher than a specified amount.
  - 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/prices.
- 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.

## Project analysis, scenario, approach and use cases

### Analysis

This project consists of creating a very application that is self-explanatory and easy to understand. The data of the application is stored in serializable files, that upon each application run are deserialized accordingly, and similarly are serialized on application exit.

The application should be able to fulfil all the requirements in order to display, modify, and keep track (CRUD operations) of orders and products. The initial set of products is stored in the products.csv file, from which the data is imported into a serializable file.

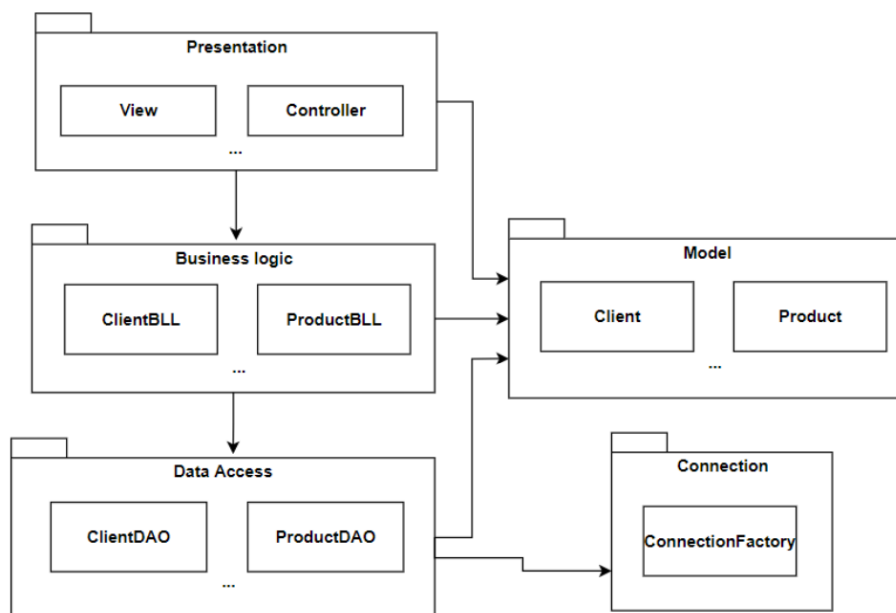
What is CRUD? CRUD is an acronym that comes from the world of computer programming and refers to the four functions that are considered necessary to implement a persistent storage application: create, read, update and delete. Persistent storage refers to any data storage device that retains power after the device is powered off, such as a hard disk or a solid-state drive. In contrast, random access memory and internal caching are two examples of volatile memory - they contain data that will be erased when they lose power.

Java provides a mechanism, called object serialization where an object can be represented as a sequence of bytes that includes the object's data as well as information about the object's type and the types of data stored in the object. After a serialized object has been written into a file, it can be read from the file and deserialized that is, the type of information and bytes that represent the object and its data can be used to recreate the object in memory. Most impressive is that the entire process is JVM independent, meaning an object can be serialized on one platform and deserialized on an entirely different platform.

## Scenario and modelling

The use is first presented with a login screen where different types of users can input their credentials. Employees, such as admin or regular employee, have their own credentials, while a client can register and create their own account, which is then saved in a serializable file.

Unlike previous projects, this project is not based on a Model-View-Controller (MVC) architecture and was instead built around a Layered Architecture.



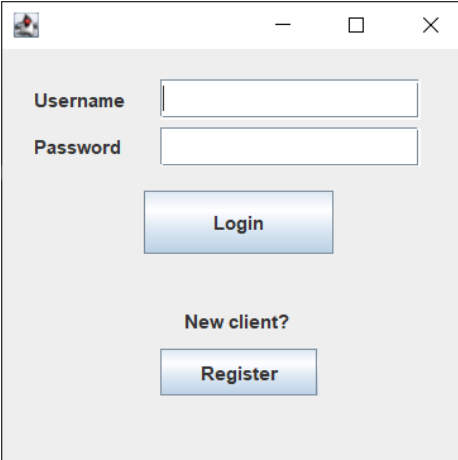
The layered architecture style is one of the most common architectural styles. The idea behind Layered Architecture is that modules or components with similar functionalities are organized into horizontal layers. As a result, each layer performs a specific role within the application.

The layered architecture style does not have a restriction on the number of layers that the application can have, as the purpose is to have layers that promote the concept of separation of concerns. The layered architecture style abstracts the view of the system, while providing enough detail to understand the roles and responsibilities of individual layers and the relationship between them.

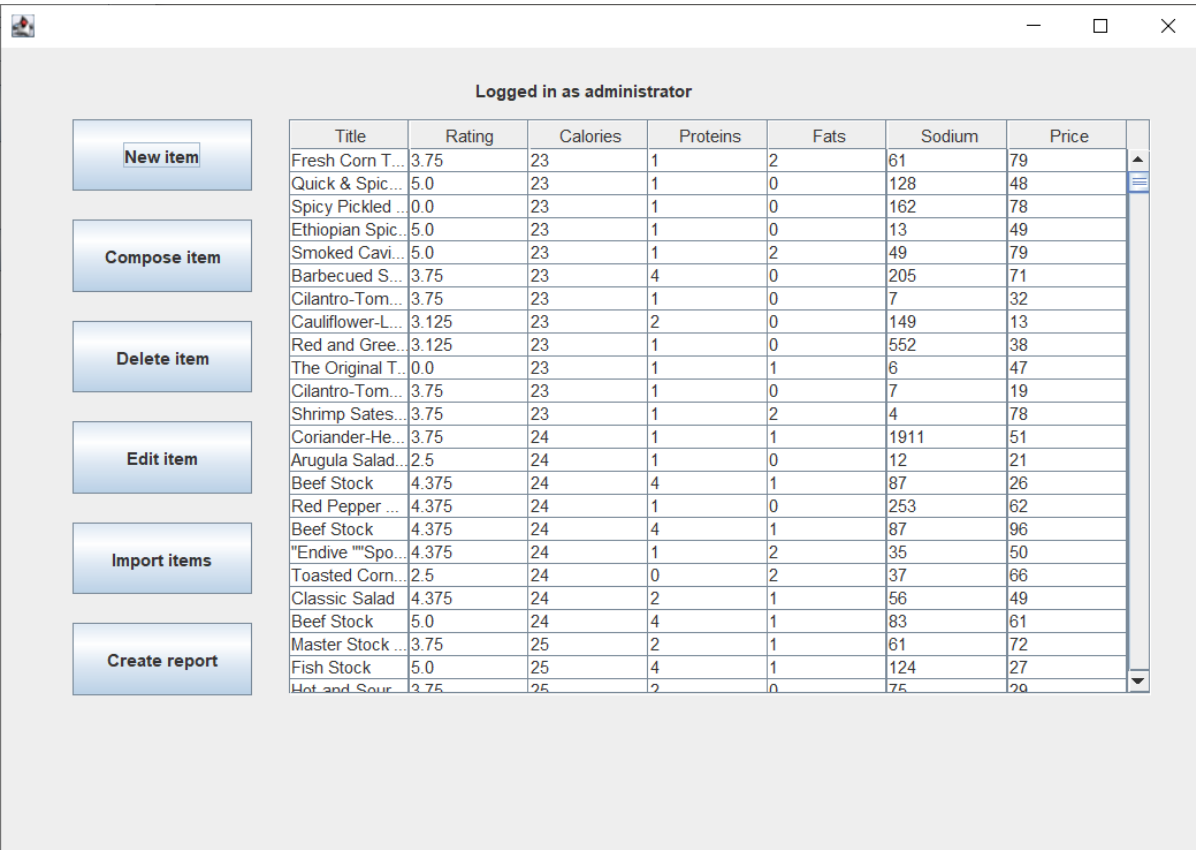
The application also contains an observer pattern. Each time an order is created, an employee is notified, making the order Observable and the employee the observer. Since this pattern is not recommended to use anymore, and it's even deprecated in the current version of Java, I created my own interface that I named Observer in order to implement this pattern, which is a requirement to the project.

Each button will open a new window, therefore in order to go back to the main menu, the user must close all open windows except this one.

The Graphical User Interface (GUI) is self-explanatory and very easy to use by anyone, because I made it so and lost a good part of my time doing this project with the GUI.



A login and registration window with a light gray background. It features a title bar with a small icon, a minus button, a maximize button, and a close button. The main area contains two text input fields labeled 'Username' and 'Password'. Below these is a blue 'Login' button. Further down is the text 'New client?' followed by a blue 'Register' button.



A window titled 'Logged in as administrator' showing a list of food items. On the left is a sidebar with buttons: 'New item', 'Compose item', 'Delete item', 'Edit item', 'Import items', and 'Create report'. The main area contains a table with 7 columns: Title, Rating, Calories, Proteins, Fats, Sodium, and Price. The table lists 25 items with their respective values.

Title	Rating	Calories	Proteins	Fats	Sodium	Price
Fresh Corn T...	3.75	23	1	2	61	79
Quick & Spic...	5.0	23	1	0	128	48
Spicy Pickled...	0.0	23	1	0	162	78
Ethiopian Spic...	5.0	23	1	0	13	49
Smoked Cavi...	5.0	23	1	2	49	79
Barbecued S...	3.75	23	4	0	205	71
Cilantro-Tom...	3.75	23	1	0	7	32
Cauliflower-L...	3.125	23	2	0	149	13
Red and Gree...	3.125	23	1	0	552	38
The Original T...	0.0	23	1	1	6	47
Cilantro-Tom...	3.75	23	1	0	7	19
Shrimp Sates...	3.75	23	1	2	4	78
Coriander-He...	3.75	24	1	1	1911	51
Arugula Salad...	2.5	24	1	0	12	21
Beef Stock	4.375	24	4	1	87	26
Red Pepper ...	4.375	24	1	0	253	62
Beef Stock	4.375	24	4	1	87	96
"Endive ""Spo...	4.375	24	1	2	35	50
Toasted Corn...	2.5	24	0	2	37	66
Classic Salad	4.375	24	2	1	56	49
Beef Stock	5.0	24	4	1	83	61
Master Stock ...	3.75	25	2	1	61	72
Fish Stock	5.0	25	4	1	124	27
Hot and Sour	3.75	25	2	0	75	29

Create order

Reset table

Search items

Name

name filter

min

max

Logged in as client

Title	Rating	Calories	Proteins	Fats	Sodium	Price
Fresh Corn T...	3.75	23	1	2	61	79
Quick & Spic...	5.0	23	1	0	128	48
Spicy Pickled...	0.0	23	1	0	162	78
Ethiopian Spic...	5.0	23	1	0	13	49
Smoked Cavi...	5.0	23	1	2	49	79
Barbecued S...	3.75	23	4	0	205	71
Cilantro-Tom...	3.75	23	1	0	7	32
Cauliflower-L...	3.125	23	2	0	149	13
Red and Gree...	3.125	23	1	0	552	38
The Original T...	0.0	23	1	1	6	47
Cilantro-Tom...	3.75	23	1	0	7	19
Shrimp Sates...	3.75	23	1	2	4	78
Coriander-He...	3.75	24	1	1	1911	51
Arugula Salad...	2.5	24	1	0	12	21
Beef Stock	4.375	24	4	1	87	26
Red Pepper ...	4.375	24	1	0	253	62
Beef Stock	4.375	24	4	1	87	96
"Endive ""Spo...	4.375	24	1	2	35	50
Toasted Corn...	2.5	24	0	2	37	66
Classic Salad	4.375	24	2	1	56	49
Beef Stock	5.0	24	4	1	83	61
Master Stock ...	3.75	25	2	1	61	72
Fish Stock	5.0	25	4	1	124	27
Hot and Sour	3.75	25	2	0	75	20

OrderID	ClientID	Date	Value
1	1	2022/05/25 22:36:00	183
2	1	2022/05/25 22:36:02	98
3	1	2022/05/25 22:36:08	788
2	1	2022/05/25 22:56:06	817
3	1	2022/05/25 22:56:08	341
4	1	2022/05/25 22:56:16	589
6	5	2022/05/25 22:57:47	278
1	1	2022/05/26 00:07:55	107
1	1	2022/05/26 00:09:54	145
1	1	2022/05/26 00:10:27	39
1	1	2022/05/26 00:12:38	129
1	1	2022/05/26 00:12:57	171
1	1	2022/05/26 00:13:15	112
1	1	2022/05/26 00:13:28	161
1	1	2022/05/26 00:19:04	113
2	1	2022/05/26 00:19:13	105
3	1	2022/05/26 00:19:17	258
1	1	2022/05/26 00:19:39	168
1	1	2022/05/26 00:30:36	167
2	1	2022/05/26 00:30:40	230
3	1	2022/05/26 00:30:43	128
4	1	2022/05/26 00:30:46	180
5	1	2022/05/26 00:30:48	162

## Use case scenarios

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The use cases are strongly connected with the user steps.

As mentioned before, the user must either input their login credentials, or register in order to create a new client account. The administrator can perform all CRUD operations on the list of menu items, and generate order specific reports. A client can search through the list of products by filtering items by any type of filter criteria, such as name, rating, etc. and create an order consisting of several products. Each time an order is created, a bill is also generated as a text file, and an employee is notified of the order through an observer pattern.

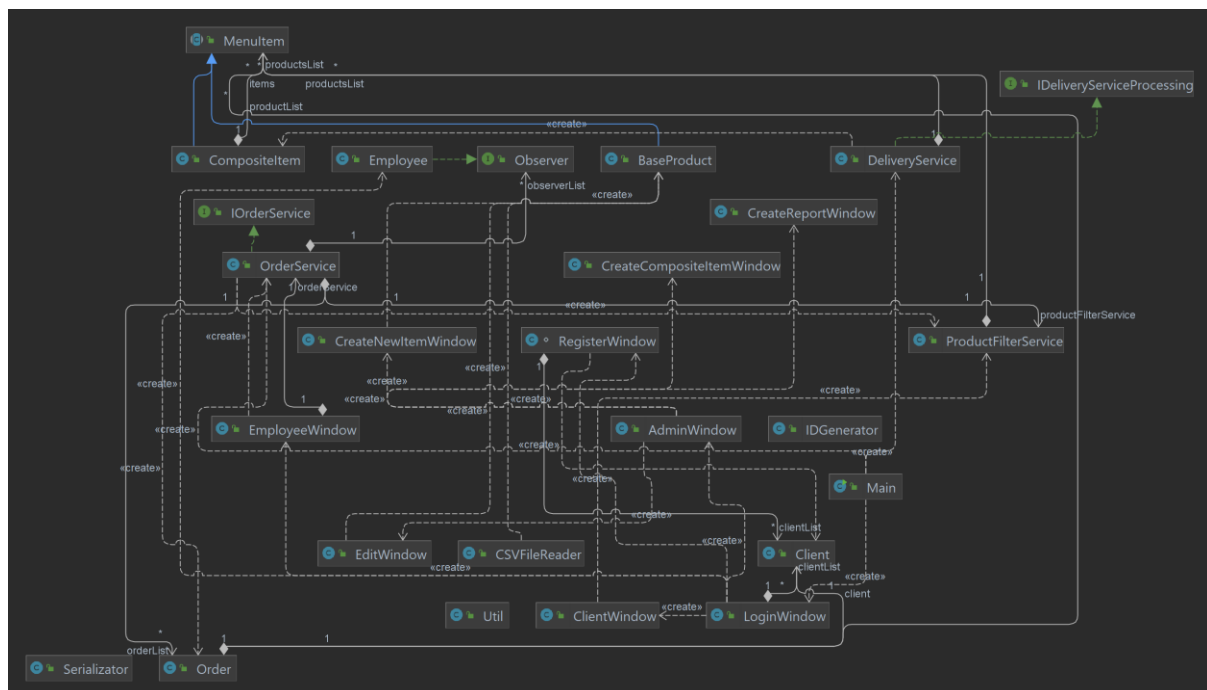
## Implementation

## Diagrams

## Class diagram

In the course of creating the project, several diagrams were made to help with the development.

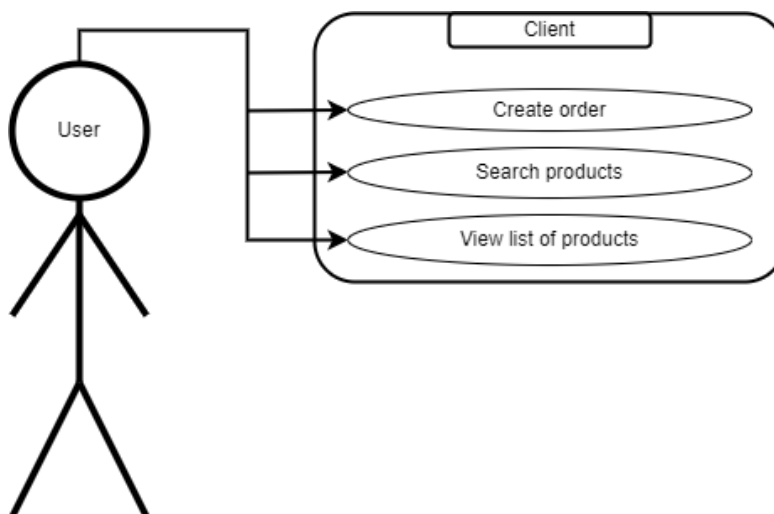
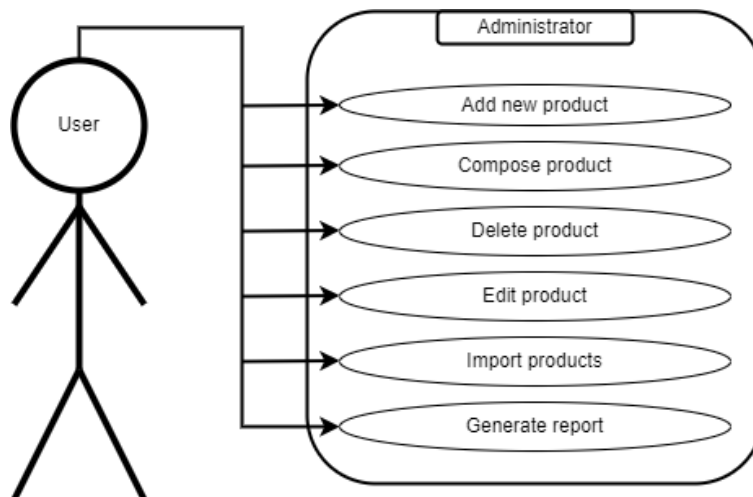
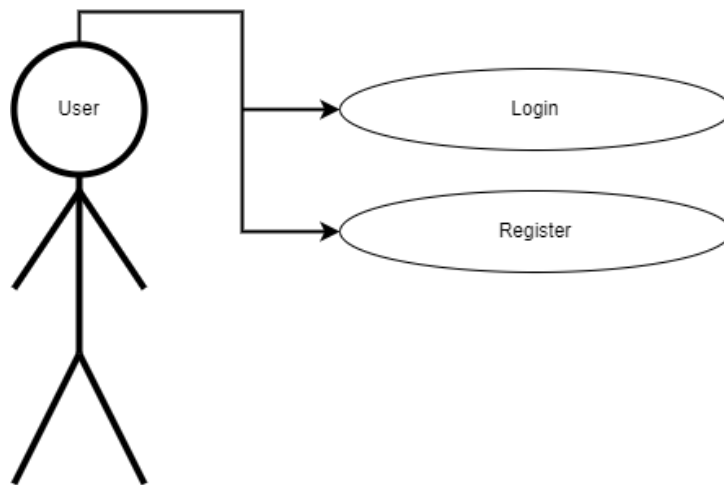
Below you will see a UML (Unified Modelling Language) diagram, used to visualize, specify, and document the software system. This diagram contains all the Java classes used in the implementation of this project, as well as their dependencies and relations with each other. This diagram is known as the class diagram.



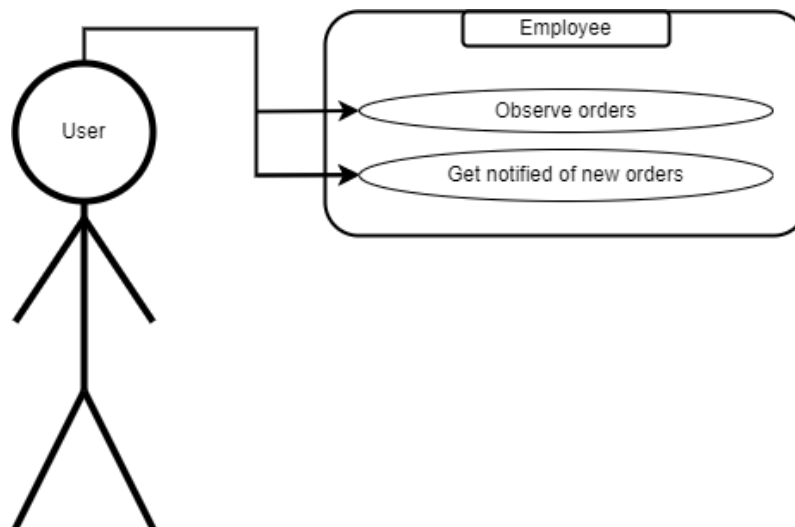
Since several fields and components were made in each class, to see a more detailed diagram, please refer to the generated one in the Java IntelliJ IDE.

## Use case diagram

The following diagrams are the use case diagrams, depicting the different cases a user finds itself in.







## Packages

### Business Logic Layer (BLL)

This package contains the main logic of the application and connects the Data Access Objects (DAOs) with the GUI. Each model class has its own BLL and performs the same operations as in its DAO but with access to the user interface through creating an instance of the DAO in the BLL and calling its CRUD methods.

The business logic layer is the business components that provide Open Applications Group Integration Specification (OAGIS) services to return data or start business processes. The presentation layer uses these OAGIS services to display data, or to invoke a business process. The business logic provides data required by the presentation layer. The business logic layer exists because more than just fetching and updating data is required by an application; there is also additional business logic independent of the presentation layer.

In this package we will find several classes that control the logic of the application, the most important being the `DeliveryService` and `OrderService` classes, which perform the operations the admin respectively client can perform. These both implement their own interface `IDeliveryServiceProcessing` and `IOrderProcessing`. The idea is this structure is to help identify the operations more easily by whomever is reviewing the code.

This package also contains a sub-package called `Util`, which contains classes for creating `JTables` and an `IDGenerator` to help generate unique IDs throughout the application for clients and orders (the `IDGenerator` needs further work that I have no more will to do).

### Data Access Object (DAO)

In software, a data access object (DAO) is a pattern that provides an abstract interface to some type of database or other persistence mechanism. By mapping application calls to the persistence layer, the DAO provides some specific data operations without exposing details of the serializable files.

In this package we will find that we know from the MVC architecture as Model. This package contains the most important class of the project, an abstract class called `MenuItem`. Basically, each product from the products list found in the `products.csv` file is first and foremost a `MenuItem`. This abstract class is implemented by two more classes, namely `BaseProduct` and `CompositeItem`.

The BaseProduct class implements the MenuItem's constructor, so whenever a new product is added to the list of products, it's added as BaseProduct. The CompositeItem has a list of MenuItems, a name and a rating which are input by the user. Since this class also implements MenuItem abstract class, it will be added to the products list, after having calculated it's attributes by going through the items list it has as a parameter.

Another very important class in this package is the Serializable class, which implements all the methods used to serialize and deserialize the three files that are used in the background of the application to store the data about the products, orders and respectively clients.

This isolation supports the single responsibility principle. It separates what data access the application needs, in terms of domain-specific objects and data types (the public interface of the DAO), from how these needs can be satisfied with a specific DBMS, database schema, etc. (the implementation of the DAO).

## Presentation

Finally, we have the presentation package. This consist of 9 classes, each representing a different window of the application. The package contains only graphical user interface elements implemented in Java Swing.

Swing is a set of program components for Java programmes that provide the ability to create graphical user interface (GUI) components, such as buttons and scroll bars, that are independent of the windowing system for specific operating system. Swing components are used with the Java Foundation Classes (JFC).

On each window there may be one ore more button, which all implement an ActionListener. Java ActionListener is notified every time you click the button. It is notified by ActionEvent. The ActionListener interface exists in java.awt.event package. It has only one method actionPerformed(). This method is invoked automatically every time you click the button.

## Further development

The code is a big sausage with no real way of debunking what was done if the creator of the code is not present to explain it. It lacks a real structure outside the layered architecture. Therefore, as a possible further development, it would be nice if it were implemented with an MVC pattern too, and most importantly a Controller class is needed to separate the GUI code from the logic.

## Conclusion

This project was a good exercise in remembering the OOP concepts learned in the first semester, but also learning new ones, which I found very useful and challenging at first. It was on ok exercise that didn't need to be as complicated as it was required to be. Some requirements made no sense at all and were just absolutely ridiculous.

I arrived at the conclusion that facing problems with your code and trying to make it work by yourself, through the mean of research, has the benefit of learning new concepts and a better use of the known ones

## Bibliography

<https://dsrl.eu/courses/pt/>  
<https://stackoverflow.com/>  
<https://www.youtube.com/>  
<https://www.geeksforgeeks.org/>

[Data access object - Wikipedia](#)  
[Reflective programming - Wikipedia](#)  
[Business logic layer \(hcltechsw.com\)](#)  
[asd.drawio - diagrams.net](#)  
[Java Swing | JTable - GeeksforGeeks](#)  
[Layered.pdf \(uwaterloo.ca\)](#)