# IS1220HB
# SOFTWARE ENGINEERING
## Final Report: Part II

Modified implementations, tests and
Command Line User Interface of an
Enjoy-Your-Meal System

Paris, France
2016

Written by
**LOH CHENG WAI MATHIAS**
**XIONG TIANKAI**

In association with

CentraleSupélec

# Contents

# 1 Foreword

This report provides an extension update from phase I implementations of the **Enjoy-Your-Meal-System** (EYMS). The principle objective of phase II is to build on the LMS core developed in phase I and develop a **Command Line User Interface** (CLUI).

In order to do this, a lot of previously implemented features were tweaked and edited to better suit client specifications and demands. In this report, we aim to make known the changes made from the previous iterations. Following which, we will discuss more on the implementation of the CLUI and its features.

Do note that as opposed to our submitted version in phase I, a lot of code has been modified, edited, or deleted in view of these changes. We seek your understanding and implore you (the reader) to refer to our latest implementation and UML to evaluate our final results.

# 2 Key changes from phase 1

One of the key criteria for creating the CLUI was to ensure that it met client specifications of the command inputs required. This required some changes as previously, our methods were concatenated (for example, the method of ordering a meal was previously a combination of `selectMeal <>`, `personalizeMeal <>` and `saveMeal <>`). Over this section, we will discuss some of the notable changes we have made.

## 2.1 `Meal` package

Previously, we made 3 abstract subclasses to better accommodate different kinds of `Meal`, such as `Appetizers`, `MainCourse`, `Dessert`, etc. However, this was unnecessary as it was not required by the client. Nonetheless, we have kept these implementations as we believe it could be useful for the restaurant in the long run (where they might take interest in separating meal objects into different categories).

In order to do this, we have changed the `Meal` class to be a subclass of an abstract `AbstractMeal` class instead, following the implementations that we have previously made with minimal changes. (Please refer to UML/source code/JDocs directly) This way, we would be able to create new instances of `Meal` objects directly (previously, it was an Abstract class, hence we were unable to instantiate it). This also allows clients to call `Appertizer`, `Dessert`, `MainCourse` meal objects if they require so in the future. This was done in a fairly easy manner, and goes to demonstrate our compliance to the `Open-Close` principle.

## 2.2 `Orders` package

Similarly, we needed to make changes to our `Order` package because it previously contained methods that nest many implementations that are required by our Client. In this overhaul, we have added smaller methods that handles specific tasks that our clients required. (Please refer to UML/source code/JDocs directly)

### 2.2.1 Personalizing meals

Previously, we have noted that our `Meal` package contained different Behaviors to handle personalization of different meals. Behaviors do work as intended, however, as we have feared, changing the quantity of ingredients, adding of ingredients or removing ingredients have caused a permanent change in either of our `Meal` object or the `Ingredient` object.

Again, we were faced with the choice of either serializing both Meals and Ingredients (hence the ability to achieve `deep copy`), or to manually create new instances of Meal and Ingredient objects whenever we want to personalize a meal. We have chosen the latter as it is the more elegant solution and saves us the trouble of selecting `transient` attributes. To achieve this, we have created the method `createnewinstance()` in both our `Meal` and `Ingredient` classes, which returns a new instance of the object in question when called. We would then change the quantity of this new instance to preserve the integrity and originality of the instance it was created upon.

## 2.3 `Update` package

Similar to the previous 2 packages, we have separated our `Notify()` method to `NotifyAd()` and `NotifyBirthday()` to better client specifications.

# 3 Command Line User Interface (CLUI)

We have created an entire new package `commandinterface` to handle our CLUI as opposed to `LoadSystem` and `RunSystem` classes previously designed and chosen in our `coresystem` package, to better reflect the independence of CLUI from the other packages of our core system.

As specified by the client, our CLUI has the following functions enabled:

- `login <username, password>` : to perform the login
- `createMeal <mealName, price>` : to create a meal with a given name
- `addIngredient <ingredientName, quantity>` : to add an ingredient to the current meal
- `currentMeal <>` : to show the name of the construction meal

- `saveMeal <>` : to save the current meal
- `selectMeal <mealName, quantity>` : to add quantity of meals of a given name to an order
- `personalizeMeal <mealName, ingredient, quantity>` : to add/remove an ingredient from a meal
- `putInSpecialOffer <mealName, price>` : to add a meal in a special offer
- `removeFromSpecialOffer <mealName>` : to add a meal in a special offer
- `listIngredients <mealName>` : show all ingredients of a meal
- `saveOrder <>` : to save the current order
- `insertOffer <mealName, newPrice>` : to insert a meal in a special offer policy
- `registerClient <firstName, lastName, username, password>` : to add a user to the system
- `addContactInfo <contactInfo>` : to add a contact info to the current user
- `associateCard <userName, cardType>` : to associate a fidelity card to a user
- `associateAgreement <username, agreement>` : to associate an agreement to a user
- `insertChef <firstName, lastName, username, password>` : to add a chef to the system
- `notifyAd <message, mealName, specialPrice>` : send a message to alert the users about an offer
- `notifyBirthday <>` : sends a special offer to the users that celebrate their birthday
- `showMeal <orderingCriteria>` : to see the list of meals ordered according to a certain criteria. orderingCriteria could either be `AsItIs`, `AsMostModified` and `JustOnSale`

We have also added additional commands to aid client navigation

- `listMeals <>` : lists all available meals of the restaurant
- `-help` : lists helpful methods that users can use
- `exit` : quits the program

**Do note that syntax is really important while using the CLUI. In particular, a space should be included between the method call and the left bracket "<".**

### 3.1 CLUI Test codes

Different forms of testings were prepared for the CLUI. As specified by the client, a "eval_N.txt" (N = 1,2,3) text file has been provided for evaluation, in which it contains CLUI inputs for a general test `test1.java` found in our `test` package (see Section 3.2). It houses a sequence of CLUI commands that allows the testing of main functionality of the EYMS-app following the test cases specified by the client. The purpose of these various tests are self explanatory in the names of the given test files. Users can easily easily navigate the test by pressing the enter key after each test step.

### 3.2 Other test codes

We have also isolated and compiled different test files from our various packages in a unified `test` package. In it, you can find various JUnit Tests which evaluates the packages that we have created, and its dependency with each other (if any).

A summary `test1` scenario has also been provided that handles a series of steps which requires the use of all packages combined.

## 4 Graphical User Interface

The GUI can be located in the `GUI` package, and can be launched from `HomeGUI.java` class.

### 4.1 Outline of implementation

As of our latest implementation, our GUI package is **not** complete. What we do have now, however, only contains very basic graphical layout of how we would imagine our eventual software to look like. We have a `HomeGUI` class that handles the "homepage" of our clients, where they would require a username and password to login.

A working register button is administered for users to create new users if they have not already. This will redirect users back to the homepage where they would require to log in.

After a user logs in, it would redirect users to our eventual platform to make orders, etc. However, **these features are not implemented yet!**

Due to the tight dateline, it is truly unfortunate that we required more time to tweak our previous code to Client specifications as well as to change the test cases involved. This GUI serves more as a platform for us to understand the capabilities of Swing and how it interacts with our command line UI. Although it remains incomplete, we are encouraged that our current implementation works and given extra time during the vacations, we would definitely finish it up.

# 5 Division of labor

Work was evenly distributed between both developers. To ensure that our code could be worked on at all times, we used a GitHub repository to enable simultaneous implementations. To ensure continuity and understanding of particular sections of our code, areas that were implemented by one would require the other to check the code and ensure that it runs well (testing). Subsection 5.1 will provide a concise summary of the distribution of tasks between both developers.

## 5.1 Distribution of tasks

| Task | LOH Mathias | XIONG Tiankai |
|---|---|---|
| **Conceptualization and Planning** | | |
| UML Design | X | X |
| Design patterns choices | X | X |
| Data structures | X | X |
| **Phase I** | | |
| Card Fidelity package implementation | X | |
| Core System package implementation | X | X |
| Ingredients package implementation | X | |
| Meal System package implementation | X | |
| Orders package implementation | X | X |
| Update package implementation | | X |
| Users package implementation | X | |
| Test codes | X | X |
| Creation of final UMLs | | X |
| Creation of Design Pattern UMLs | | X |
| Report (Part I) | X | X |
| **Phase II** | | |
| CLUI Implementation | X | X |
| CLUI Tests | X | X |
| GUI Implementation | | X |
| Code cleanup | X | X |
| Optimization of algorithms | | X |
| Revamped test package | X | X |
| Report (Part II) | X | |

# 6 Conclusion

We thank you for taking the time to read this report and for using the CLUI of our Enjoy-Your-Meal-System. We are sincerely apologetic that a GUI implementation was not completed as desired. Nonetheless, we would be appreciative of any feedback and you may contact us at **mathias.loh-cheng-wai@student.ecp.fr** and **tiankai.xiong@student.ecp.fr**.

This implementation of EYMS would not be possible without the patient aid and guidance of Ms. Francesca Bugiotti, Mr. Paolo Ballarini and other coaching members of IS1220HB, Software Engineering, CentraleSupélec (2015-2016).