

Lab Report for Software Engineering course
Lab 5: Demand Change and Prototype
Development

Wang, Chen	Liu, Jiaying	Huang, Jiani	Tang, Xinyue
16307110064	17302010049	17302010063	16307110476

School of Software
Fudan University

May 28, 2019

Contents

1	Demands of this lab	3
1.1	Requirements of this lab	3
1.1.1	Documentation	3
1.1.2	Code Style	3
1.1.3	Project Management	3
1.2	Background on the change of demand	3
1.2.1	Drink Customization	4
1.2.2	Currency change	4
1.2.3	Language change(localization)	4
1.2.4	Sales strategies	4
1.3	Specifications of the Lab	4
1.3.1	Drink Customization	4
1.3.2	Currency change	4
1.3.3	Language change(localization)	5
1.3.4	Sales strategies	5
1.4	Additional precautions of this lab	5
1.4.1	Switch interface of currency and language	5
1.4.2	Prototype-based development	5
1.4.3	Methods for switch and pluggable components	5
1.4.4	Freedom of change of the existing frameworks	6
2	Division of work for this lab	7
2.1	Division of work: Wang, Chen	7
2.2	Division: Huang, Jiani	7
2.3	Division: Tang, Xinyue	7
2.4	Division: Liu, Jiaxing	7
3	Analysis of the demands	9
3.1	General analysis of the demands	9
3.2	General workflow	9
3.3	Project Prototype	9
3.4	Currency switch analysis	10
3.5	Language switch	10
3.6	Sales strategies	11
4	General design for the implementation	12

5	Detailed design for the implementation	13
5.1	Test design: for Test-Driven Development	13
5.1.1	Test components of the project	13
5.1.2	Swagger Interface Test	13
5.2	Switch Language Implementation	13
5.2.1	Language Constant Files	16
5.2.2	Language Service Classes	16
5.3	Switch Currency Implementation	16
5.3.1	Currency Property Files	16
5.3.2	Currency Service Classes	16
6	Problems encountered in this project	17
7	Measures against demand change	18
8	Tools and literature involved in this project	19
9	Conclusion for the process of accomplishing this project	20

Chapter 1

Demands of this lab

1.1 Requirements of this lab

According to the documentation of the lab assigner, this lab should satisfy the requirements in the following perspectives:

1.1.1 Documentation

In the documentation, we need to accomplish two parts, that is, we need to: Understand the needs of this experiment, complete the requirements document; Organize overall design documentation and detailed design documentation based on requirements.

1.1.2 Code Style

Before the experiment, our group members should agree on the code specification and the unified code style.

1.1.3 Project Management

The project is generally hosted on the Huawei Devcloud platform, on which platform we are going to finish the following tasks: Project management should be arranged based on the DevCloud platform, including adding work items, assigning work items, associating work items, managing work item status, and other project management functions;

Issues in the development process need to be recorded and managed on DevCloud, including defect reports in integrated development;

Collaborative development based on Git, submit project documentation and code on a team basis.

1.2 Background on the change of demand

Starbucks's coffee shop has received a large amount of financing and decided to open branches in different countries and regions.

In order to adapt to the habits of users in different regions, it is necessary to develop a project prototype with basic functions, and then customize the system for different regions. Customized development needs to consider the following factors:

1.2.1 Drink Customization

At least 2 beverages need to be customized for local features (beverages defined in Lab4 are system default beverages).

1.2.2 Currency change

We need to switch to the corresponding currency according to different countries and regions, for example, a cappuccino in China 20CNY, a cappuccino in the US \$3 (do not consider currency exchange).

1.2.3 Language change(localization)

The system needs to switch the system language to the language of the corresponding country and region according to the language of different countries and regions. The range of language switching is visible to all system users (sales personnel) and customers (beverage buyers). For example, when the system user inputs, the input content should be the language of the corresponding country and region; after the order is successful, the customer sees The transaction content should be switched to the language of the corresponding country and region.

1.2.4 Sales strategies

A good sales strategy can bring huge benefits, and the system's default sales strategy is the sales strategy defined in Lab4. Because sales strategies can change as external markets change, systems are required to add and remove sales strategies in a pluggable manner.

1.3 Specifications of the Lab

The following requirements are to be met for the custom development factors described in the background in the previous section:

1.3.1 Drink Customization

Customize two beverages with reference to the previous lab implementation.

1.3.2 Currency change

There may be multiple currencies in different countries and regions. For example, Hong Kong can circulate Hong Kong dollars and Chinese Yuan, mainland China circulates Chinese Yuan, and the United States circulates US dollars. The prices of different currencies for the same item are different depending on the value of the product itself. The system needs to specify the corresponding

price of the corresponding commodity for different currencies (the pricing of different currencies for each commodity, and the exchange rate pricing can be reasonably justified). Currently, the system only needs to support RMB and Hong Kong Dollar (HK\$).

1.3.3 Language change(localization)

Different countries and regions may have multiple official languages, and the system needs to switch between different languages according to different countries and regions. The system must be able to switch between English and Simplified Chinese.

1.3.4 Sales strategies

The sales strategy is added and deleted in a pluggable manner. The sales strategy in lab4 is the default sales strategy and cannot be deleted. The source of information for all sales strategies is order and system information (date and location). On the basis of lab4, the requirements of (1)(2)(3) and the pluggable components of the sales strategy can be regarded as the prototype of the project. Based on this prototype, add the following sales strategy:

1. When the time zone of the system is November 11th, the total price of all commodities (including all prices) will be 50% off.
2. When the order contains at least one tea item and at least one item of the Coffee category, the total price (including all prices) is 15% off.

1.4 Additional precautions of this lab

1.4.1 Switch interface of currency and language

The switching of currency and language can depend on a “switch”, which is an abstraction rather than a concrete implementation, since the opening and closing of the switch may depend on different external conditions.

1.4.2 Prototype-based development

The addition and deletion of the sales strategy is based on the fact that the prototype of the project has been developed, and the sales strategy is added and deleted through the interface of the sales strategy provided by the prototype. When using the prototype, you can't modify the code of any prototype, and you can only perform secondary development on the basis of the prototype.

1.4.3 Methods for switch and pluggable components

The development of “switches” and pluggable components can refer to the relevant materials.

1.4.4 Freedom of change of the existing frameworks

In order to flexibly realize the four requirements in the previous sections above, it is encouraged that every team should design independently. The fields and method signatures in the framework code provided by the assistants in Lab4 can be modified. It should be emphasized that after the prototype is completed, it cannot be modified, and it can only be developed on the basis of the prototype.

Chapter 2

Division of work for this lab

2.1 Division of work: Wang, Chen

(Git username: *Wang, Chen*; Student ID: *16307110064*)

He constructs the overall structure of the project, divides the entire workload into several parts so that each part can be finish the work separately. In addition, he draws the diagram of the entire project on the Huawei cloud platform that contains the parts like Epic, feature, story and tasks. Furthermore, he scratches the outline of how to implement the methods adopted in this project. At last, he summarized the general parts in the documentation and drafted some regulations for commit messages.

2.2 Division: Huang, Jiani

(Git username: *Currycurrycurry* Student ID: *17302010063*)

She implements the concrete methods in the language and currency switch feature.

2.3 Division: Tang, Xinyue

(Git username: *xinyuetang* Student ID: *16307110476*)

She implements the methods related to order processing, discount processing and total price processing.

2.4 Division: Liu, Jiaying

(Git username: *jiayingliu* Student ID: *17302010049*)

He designed the interfaces for various modules of the project including the *menu* module and the *language* module. In addition, considering the actual circumstances of the product, he adopts the **singleton** design pattern, the **strategy** pattern so as to improve the encapsulation level and the ability for extension of the application.

He also finishes the testing work of the entire project, whose work is an important foundation of the *Test-Driven Development* of the project. In particular, he realizes the *JUnit* test cases within the project and the *Swagger* interface test of the project on the Huawei Cloud Platform.

Chapter 3

Analysis of the demands

3.1 General analysis of the demands

In this version of requirement, a series of new functions are proposed and they are supposed to bring the general construction of the project some prominent change. Specifically, the requirements of the project lie in two perspectives: a prototype that should work for the all future new regions and two detailed and concrete implementations for the region of China and Hong Kong specifically.

Therefore, the main work of this project has two stages:

1. Design and determine our project prototype;
2. Continue finishing the specified work and implementing the corresponding functions based on the prototype.

Therefore, in the following paragraphs, I will try to analyze the demands from the two stages shown above.

3.2 General workflow

According to the general analysis in the previous section, the total workflow of this project can be shown as the following diagram.

3.3 Project Prototype

In the requirement documentation, it is clearly specified that our project should have a clear prototype and continue with the future works based on the existing prototypes. Therefore, the first step for the team members to do is to finish the prototype.

The prototype need to well represent the newly-added features that will be future implemented. Specifically, the prototype should have all the necessary interfaces needed for the customized drink, the currency switch, the language switch and the strategy change.

Should there be future concrete implementation on the futures above, the developers (our team members here) should only need to add new classes implementing the interfaces without modifying the existing interfaces.

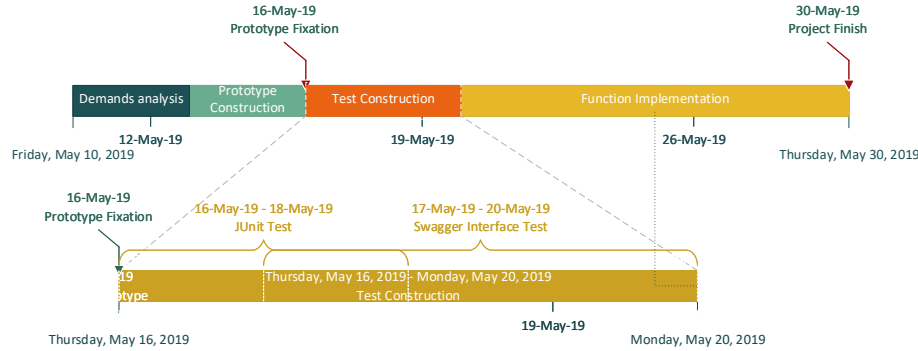


Figure 3.1: Overall workflow

3.4 Currency switch analysis

The requirements of currency switch has a lot to be further analyzed before concrete implementation. As the features of the project display, each drink and its corresponding ingredients have a price. The prices will be used in the order information and later be used to charge from the customers. However, different region has different currencies, while the drink item will also vary from one region to another. Therefore, the price of the order items should not be written directly into the code any more, which can lead more complexity to future feature extension.

Considering we are running a Spring Application, which allows a wide range of property resources lying in the *resources* folder of the application with a file extension *properties*, the price information of the order items are suitable to be stored in such places.

In such way, the price information of the project is depleted from the code section and it is much easier to maintain the data. Furthermore, should new currencies be supported in the future, we also have no need of input the price information in the code, and only need to append the configuration files.

3.5 Language switch

The switch of language is a common requirement in all the international software. To improve maintainability and readability, in addition to reduce future cost of maintenance, all the display language tags are stored in some files as constants. For example, in Android applications, all the *Strings* that need to be displayed are encouraged to be saved into a separate XML file and then read via the constant names. Meanwhile, the the global language switch, either controlled by the Android Operating System or controlled by the application, can determine which *String* to be displayed. That is, the same display info are stored and accessed via the same variable name, while the variables direct the application to different files when different languages are set.

Therefore, in our application, we are encouraged to store the display information via constants and a global language switch can determine which language's constant will be transferred to the application.

3.6 Sales strategies

Sales strategy is an important part of the application. However, in order to better utilize the market, sales strategies need to be easily switched on or off from the pricing services.

Considering such circumstances, each sales strategy need to be a single extract-able method or class so that in the future configuration, we can add on or off such strategies with ease.

Chapter 4

General design for the implementation

Chapter 5

Detailed design for the implementation

5.1 Test design: for Test-Driven Development

5.1.1 Test components of the project

In this project, we have various types of tests so as to ensure that all components are working properly.

Specifically, we have JUnit test to be utilized during the initial stages of development. These unit tests give clear guidance of the project. In addition, we have utilized the Huawei Cloud platform with its strong Interface test function. The interface test function enables the entire project can function well with its public interfaces. What's more, the interface tests are integrated into the project as Swagger Interface test component. In this component, there is a standalone package that has a front-end user interface and will be convenient to present interface test locally.

5.1.2 Swagger Interface Test

The front-end user interface of the local Swagger Interface Test is shown as the following figures(Figure 5.1 and Figure 5.2):

5.2 Switch Language Implementation

The switch of language will be mainly displayed in the user interface, so all the information that need to be multi-translated will be separately placed into different constant files. In this iteration of implementation, we only instantiate the Chinese and English versions. And the correspondent service classes will use a typical mechanism called reflection to implement the switch of different constant language files.

In the following two sections, the detailed design of constant files and language service classes will be respectively clarified.

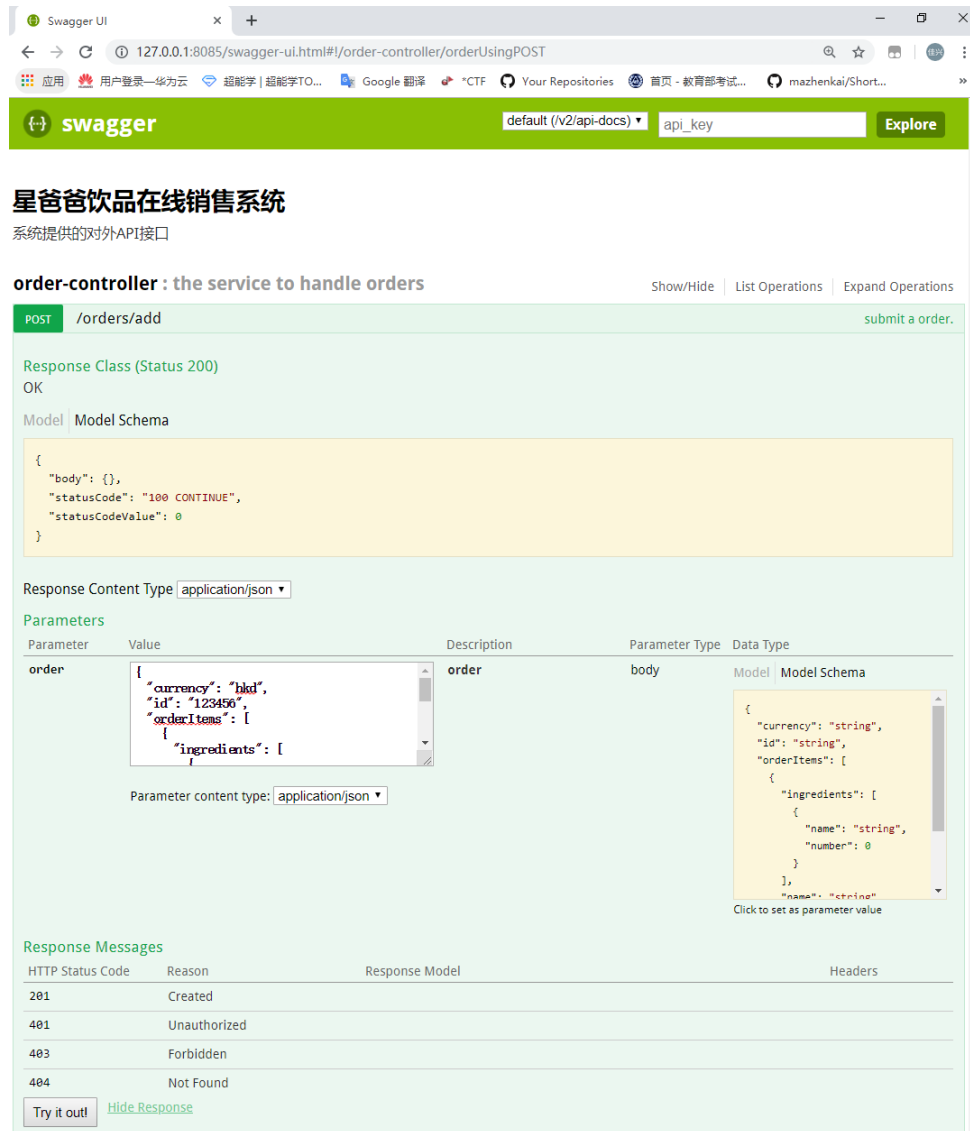


Figure 5.1: Overall workflow

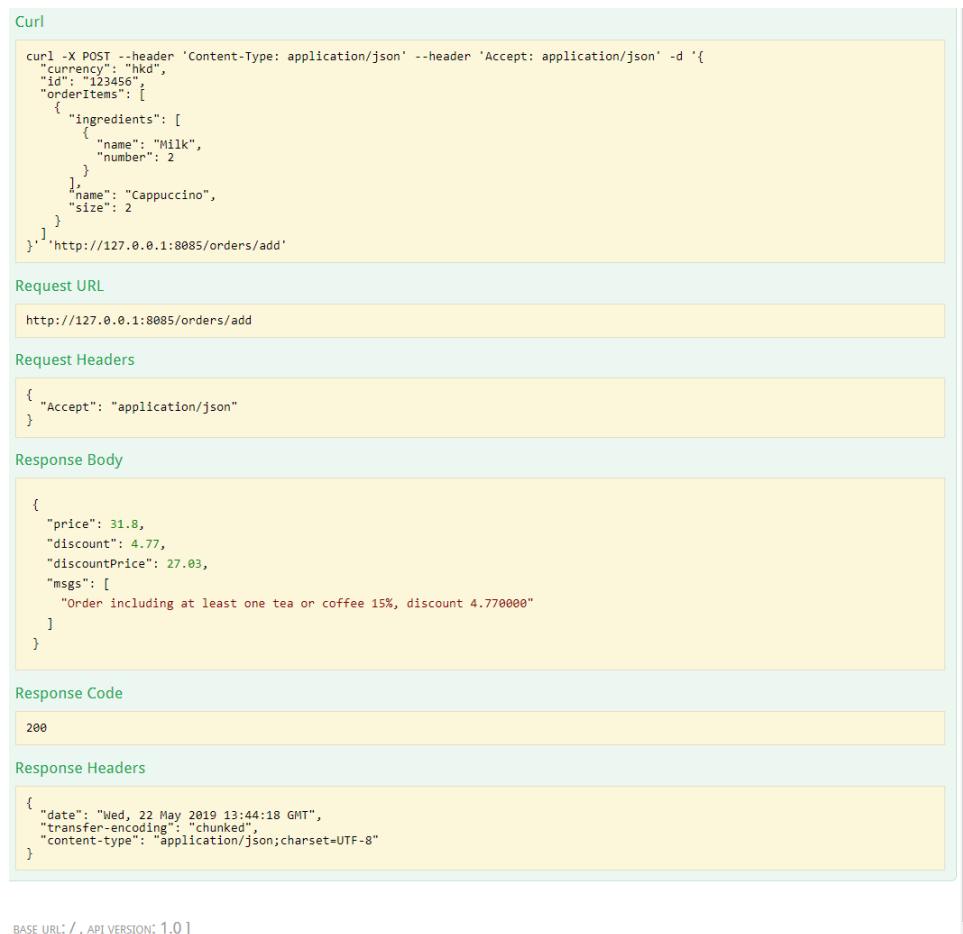


Figure 5.2: Overall workflow

5.2.1 Language Constant Files

The two constant files are positioned in the constant package. In these two files, all the variables (there are no methods) are qualified with public static final String since they are all constant strings.

To our attention, all the necessary variables should have the same names in all the language files to maintain the availability of reflection.

5.2.2 Language Service Classes

To follow the idea of prototype development, all the concrete service classes should implement their corresponding interfaces. The interface make it clear what the service will implement and its parameters. The most notable design pattern in this class is single-instance pattern.

5.3 Switch Currency Implementation

5.3.1 Currency Property Files

5.3.2 Currency Service Classes

Chapter 6

Problems encountered in this project

Chapter 7

Measures against demand change

Chapter 8

Tools and literature involved in this project

Chapter 9

Conclusion for the process of accomplishing this project

Bibliography

- [1] Wikipedia contributors. (2019, March 22). JUnit. In *Wikipedia, The Free Encyclopedia*. Retrieved 14:53, April 1, 2019, from <https://en.wikipedia.org/w/index.php?title=JUnit&oldid=888928403>