

Lab Report for Software Engineering course  
Lab 4: Starbubucks coffee online retailing system  
v3.0

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

School of Software  
Fudan University

April 19, 2019

# Contents

<b>1 Overview of this lab</b>	<b>2</b>
1.1 The Objectives of the Project . . . . .	2
1.2 Specifications of the Lab . . . . .	2
1.2.1 Add ingredients . . . . .	2
1.2.2 Adding drinks . . . . .	2
1.2.3 Price calculation . . . . .	3
1.2.4 Promotional programs . . . . .	3
1.2.5 Basic price list . . . . .	3
1.3 The division of work in the team . . . . .	3
1.3.1 Division of work: Wang, Chen . . . . .	3
1.3.2 Division: Huang, Jiani . . . . .	4
1.3.3 Division: Tang, Xinyue . . . . .	4
1.3.4 Division: Liu, Jiaxing . . . . .	4
1.4 Division of work for documentation . . . . .	4
1.4.1 Parts required in the documentation . . . . .	4
1.4.2 Principal for each part of the documentation . . . . .	4
<b>2 Improved skills on team collaboration</b>	<b>5</b>
2.1 Consistent git commit message styles . . . . .	5
2.1.1 Commit message requirements . . . . .	5
2.1.2 Types allowed in the subject of the commit message . . . . .	6
<b>3 Designed Ideas of work planning</b>	<b>7</b>
<b>4 Design ideas of implementation</b>	<b>8</b>
4.1 Entities implementation . . . . .	8
4.1.1 drinkEntity . . . . .	8
4.1.2 ingredientEntity . . . . .	8
4.1.3 OrderItem . . . . .	8
4.2 Order services implementation . . . . .	9
<b>5 Understanding of demand changes</b>	<b>10</b>
<b>6 Testing of the features</b>	<b>11</b>
6.1 Testing methods adopted . . . . .	11
6.2 Testing results . . . . .	11
<b>7 Problems encountered and solutions</b>	<b>12</b>

# Chapter 1

## Overview of this lab

### 1.1 The Objectives of the Project

Through this experiment, we will experience the impact of demand changes on development work, experience the software project management functions provided by Huawei's development cloud platform DevCloud, and experience the rapid deployment of application services with SpringBoot.

### 1.2 Specifications of the Lab

The company now hopes that the development team will develop an **Online Beverage Sales System** based on the existing system. Based on the existing system, in addition to the existing **basic functions**, the **new requirements** are as follows:

#### 1.2.1 Add ingredients

- To meet the needs of our customers, we offer a wide range of ingredients including: milk, chocolate, cream and sugar.
- The price of adding a unit of milk and chocolate is \$1.2 for the price of the beverage; the price of the unit of cream and sugar is \$1 for the price of the beverage; and a portion of the beverage can be added with a plurality of ingredients.

#### 1.2.2 Adding drinks

- In order to meet the needs of customers, we offer a variety of drinks, and now we have added drinks, so that the system can provide coffee and tea sales, including cup type matching, ingredient addition, price calculation and so on.
- Coffee types now include: Espresso, Cappuccino; Tea varieties now include: Green Tea (GreenTea), Black Tea (RedTea).

### 1.2.3 Price calculation

Due to the adjustment of ingredients and drinks, the price calculation of drinks is adjusted accordingly.

- The salesperson can choose the type of drink, the type of drink, or a variety of ingredients. The system will calculate the final price of the cup drink for the salesperson. The final price of the drink is calculated as follows: final price = drink price + drink cup type price + multiple ingredient price (unit: \$);
- The salesperson can select the number of cups under the premise of selecting a certain type of drink type. The system will calculate the final price of the multi-cup drink for the salesperson. The price is calculated as follows: multi-cup final price = single cup final price \* cup number (unit: \$);
- Coffee and tea cups are currently divided into three types: large cup (3), medium cup (2), and small cup (1). Different cup types have different prices:

Large Cup	Medium Cup	Small Cup
Coffee Price + \$6	Coffee Price + \$4	Coffee Price + \$2
Tea Price + \$5	Tea Price + \$4	Tea Price + \$2

### 1.2.4 Promotional programs

In order to attract more customers at a more favorable price, the company now offers the following promotional strategies for different beverages selected by customers:

**The first category: combination offer**

**The second category: full reduction offer**

### 1.2.5 Basic price list

## 1.3 The division of work in the team

### 1.3.1 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.

### 1.3.2 Division: Huang, Jiani

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

She creates the concrete classes for the diverse drinks and ingredients.

### 1.3.3 Division: Tang, Xinyue

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

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

### 1.3.4 Division: Liu, Jiaxing

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

He tests the implementation in this project via the interface given by the teaching assistants.

## 1.4 Division of work for documentation

### 1.4.1 Parts required in the documentation

In the requirement documentation of the lab, we are required to accomplish the following parts in this documentation:

1. Explain the design ideas of work item planning in this experiment(PLAN);
2. Explain the design ideas of code implementation in this experiment(IMPLEMENT);
3. Explain the understanding of demand changes and project management (project planning, defect management, etc.) in this experiment(UNDERSTAND);
4. Explain the problems and solutions (if any) encountered in the implementation(PROBLEM).

For the convenience of being noted, each requirement is labeled with a tag, which is used in the next part for mentioning.

### 1.4.2 Principal for each part of the documentation

Tag	Writer
PLAN	Wang, Chen
IMPLEMENT	Huang, Jiani&Tang, Xinyue
UNDERSTAND	Wang, Chen
PROBLEM	Liu, Jiaxing

## Chapter 2

# Improved skills on team collaboration

### 2.1 Consistent git commit message styles

#### 2.1.1 Commit message requirements

The following are the requirements for the commit message in our team, this version of specifications are revised according to the commit message style recommendation from website *Commit message guidelines · GitHub*<sup>1</sup>, *How to Write a Git Commit Message*<sup>2</sup>, *How To Write a Good Commit Message*<sup>3</sup> and the git commit message recommendation from one of the most authoritative open source project **GNOME** *Guidelines for Commit Messages*<sup>4</sup>.

1. Only ASCII characters are allowed in the entire commit message
2. All commit messages must start with one of the types identified in the following table, all words are lowercase
3. It is best to have an associated work item, associated with the work item, followed by type, space # number space followed by content, such as fix #123 content
4. The total number of characters recommended in the subject (*note that it is the number of chars instead of the number of words*) is less than 50, and the maximum number is not more than 74 characters (including the previous type and item number, etc.)
5. There is no need to add a period at the end of the head
6. After the type in the subject line, the first letter of the first word after the task number (if any) is capitalized and that indicates the beginning of a sentence

---

<sup>1</sup><https://gist.github.com/robertpainsi/b632364184e70900af4ab688decf6f53>

<sup>2</sup><https://chris.beams.io/posts/git-commit/>

<sup>3</sup>[https://api.coala.io/en/latest/Developers/Writing\\_Good\\_Commits.html](https://api.coala.io/en/latest/Developers/Writing_Good_Commits.html)

<sup>4</sup><https://wiki.gnome.org/Git/CommitMessages>

7. Use the imperative tone in the subject sentence (although it is you who have actually done the work)
8. The tense of the subject is the general present tense
9. It is recommended that for commits involving complex modifications, body should be added in addition to the subject for further explanation. The method is as follows: break a new line and then write is the body, [do not follow the head without line break]
10. For the body part of the commit message, there is no requirement other than writing Chinese, and it is relatively free. It is also recommended to write more than one line instead of one line in order to facilitate reading.

### 2.1.2 Types allowed in the subject of the commit message

Table 2.1: Commit message types

Type	Description
feat	A new feature
fix	A bug fix
wip	While working on a fix/feature
docs	Documentation only changes
style	Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)
refactor	A code change that neither fixes a bug or adds a feature
test	Adding missing tests
chore	Changes to the build process or auxiliary tools and libraries such as documentation generation

## Chapter 3

# Designed Ideas of work planning



## Chapter 4

# Design ideas of implementation

### 4.1 Entities implementation

The implementation of entity package can be divided into two packages: the package of drinkEntity and the one of ingredients.

To our attention, I apply this.getClass.getName() to simplify the constructor by avoiding inputting string every time.

#### 4.1.1 drinkEntity

At present we have four different kinds of drinks: Cappuccino, Espresso, GreenTea and RedTea. But considering the possibilities of adding other kinds of coffee and tea and the rationality of logic, the abstract classes of Coffee and Tea are applied to construct the whole inheritance relationship. That is, Both Coffee and Tea will inherit the OrderItem class, and all specific kinds of coffee and tea will inherit the abstract classes of Coffee and Tea.

We should pay attention to the constructor method of the different concrete classes. Since the initialization only need the different basic price of drinks, we only need to call the setPrice() method in constructor method. And all the generic method of these classes will be implemented in the parent class, the OrderItem class, such as cost() and size2price() methods. Thanks to the inheritance tree, in the size2price() method, we can use the instanceof keyword to simplify the judge of cup-size.

#### 4.1.2 ingredientEntity

It is evident that the four concrete ingredients inherit the Ingredient class in dto package: chocolate, cream, milk and sugar. The constructor like the above drinkEntity's only need to set the price.

#### 4.1.3 OrderItem

I add two methods: size2price and cost in this class in order to make all its subclasses use this method.

## **4.2 Order services implementation**

## Chapter 5

# Understanding of demand changes

## Chapter 6

# Testing of the features

### 6.1 Testing methods adopted

### 6.2 Testing results

## Chapter 7

# Problems encountered and solutions

# 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>