Lab Report for Software Engineering course
Lab 2: Starbubucks coffee online retailing system v1.0

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

School of Software
Fudan University

March 20, 2019

Contents

Chapter 1

Overview of the lab

1.1 the Objectives of the Project

1.1.1 Basic Functions

In this lab, we are going to complete an online coffee retailing system called "Starbubucks coffee online retailing system", to experience the software development process and to feel the importance of code quality control and software design. In general, this lab is based on the Git and DevCloud cloud platform for the convenience of team collaboration.

1.1.2 The life cycle of software development

Through this lab, we also have the chance to experience the real process of software development, and understand the importance of coding quality and software design. We are going to experience the most of a life cycle in the software development. A systems development life cycle is composed of a number of clearly defined and distinct work phases which are used by systems engineers and systems developers to plan for, design, build, test, and deliver information systems. [1]

In this lab, we can see the full period of the life cycle in a project in software engineering. Specifically, the steps are displayed in the following ways:

1. **Planning.** The platform to place the codes and accomplish the management work, the framework to accomplish the back end and the front end of the server, the server system environment, the displaying user interface and the way to hand in the works are already worked out by the teaching assistants.

2. **Analysis.** The teaching assistants should consider how long the time limit should be given to the students, how detailed the documentation should be written and whether the difficulty is suitable for the students.

[1] Wikipedia contributors. (2019, March 10). Systems development life cycle. In *Wikipedia, The Free Encyclopedia.* Retrieved 06:13, March 10, 2019, from `https://en.wikipedia.org/w/index.php?title=Systems_development_life_cycle&oldid=887015682`

3. **Design.** After analysis, the teaching assistants should make appropriate frameworks, make sure that they can be started successfully, the interfaces are clearly specified and finally, place the codes on the Huawei Cloud platform.

4. **Implementation.** The students should follow the instructions of the teaching assistants and implement the concrete methods. Here specifically, we need only to modify a String and a port number.

5. **Maintenance.** Here this version of the Lab requires no maintenance after submission.

1.1.3 Teamwork Based on Git and DevCloud

First of all, the start-up codes that the teaching assistants have prepared are put on the Code Management section on the cloud platform. We ought to clone the codes from the repository to our local desktop to make modifications. After proper modification, we need to commit our changes and push them to the remote repository on the Huawei Cloud. We will develop the project as a team and utilize the advantages of the Git service offered by Huawei cloud.

1.2 The division of work in the team

Based on the actual work load of the whole project, the condition of each member in the team and according to the principle of equal responsibility, we have divided the entire workload into the following sections:

1.2.1 Login and Signup

Before entering the coffee system, the user should signup/login first. This part is finished by Jiani Huang.

1.2.2 Check Status of Login and Price

The system should always record the status of user and the order. This part is finished by Chen Wang.

1.2.3 Calculate the Coffee Price

The system will calculate the total order price according to the standard input made by the user. This part is finished by Jiaxing Liu.

1.2.4 Code Checking

Finally, to ensure the correctness and quality of the whole system, we will perform several code checks on the DecCloud platform. This part is finished by Xinyue Tang.

1.3 Framework: Spring Boot

1.3.1 The feature of Spring Boot in comparison to other Java EE frameworks

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that we can "just run".

The designers of Spring Boot take an opinionated view of the Spring platform and third-party libraries so we can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

Specifically, Spring Boot has the following features:

1. Create stand-alone Spring applications;
2. Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files);
3. Provide opinionated "starter" dependencies to simplify your build configuration;
4. Automatically configure Spring and 3rd party libraries whenever possible;
5. Provide production-ready features such as metrics, health checks and externalized configuration;
6. Absolutely no code generation and no requirement for XML configuration.

Chapter 2

Steps of accomplishing this Lab

2.1 the Design of Login and Signup

2.1.1 Idea and Method

Note: we only consider the situations of login/signup failure mentioned in the requirement document, and other operation failures such as database/network errors are not included in the error handling.

A. Login

As the document says, login can be divided into two situations: login successfully (both the name and password are matched) and login failed (throw the runtime exception).

Therefore, in the whole login method, we use "loginStatus", the private variable to record the status, and apply the if-else branches to these two situations: if the return value of the "getUser" method in "UserRepository" class is not null and the return value of "getPassword" method and the parameter user's password are the same, it goes to login successfully, and the logger record information.

Otherwise, it goes to login failed. After logger, it will directly throw a runtime exception.

B. Signup

Also, the signup part can be divided into two cases: signup successfully (the name can be used) and signup failed (the name already exists in the file).

Therefore, in the whole signup method: if the name doesn't exist in the file, it means the name is not repetitive. So it goes to the catch branch so as to create the new user, and write the logger.

Otherwise, it throws the runtime exception and write the logger.

C.Cost

The price of a cup of coffee consists of two parts: coffee cost and cup-type cost. Therefore, the *cost* method in class *Coffee* only needs to return the sum of the two costs. In order to get the cost of different cup types, I design a method called *size2price*. And it works out the price with switch-case statement. In terms of the *cost* method in class *PriceService*, I only need to calculate the sum of all orders and output each order's information to the console.

What's more, I create a class called *Size*–which includes only three meaningful constant values–to express the cup types clearly. I think it maybe a good idea and it is very useful when to read codes.

2.1.2 Encoding specification

Detailed Descriptions of Thrown Exceptions

```
throw new RuntimeException(InfoConstant.                    1
    USERNAME_OR_PASS_ERROR);
throw new RuntimeException(userAlreadyExistStr);            2
```

Catch the Particular Exceptions

```
{                                                           1
...                                                         2
} catch (RuntimeException e){                               3
        .....                                               4
}                                                           5
```

Readability of Variables

e.g.userSignupOkStr/userLoginStr

Proper comments and consistent comment style

```
//user exist and the password correct                       1
```

2.2 Problems and Methods

2.2.1 the error caused by nextLine( )

solution: add another line of "in.nextLine( )" to absorb the redundant line feed.

2.2.2 Code Checking: Password or Certificate

solution: rename the password-related constants (since this lab does not relate to encryption)

2.3 testing and optimizing

2.3.1 testing

result:find two problems:

1. the PriceService.cost module output:

“ name: null, size: 1, number: 2, price:4$
20$ ”

solution:use the new CappuccinoRepositoryImpl() and new EspressoRepositoryImpl()

```
Coffee coffee = coffeeType == 1 ? new                    1
    CappuccinoRepositoryImpl()
                .getCappuccino("cappuccino") : new       2
                    EspressoRepositoryImpl()
                .getEspresso("espresso");                3
```

2. the logic of login after signing up

our program let the user to login automatically after signing up before. When the TA said that user should login in by themself after signing up, I changed the logic of Lab2Application.main function.

2.3.2 optimizing

the name and password's validity verification:

at first we write two while loop for the null value and mismatching value condition, we merged them to one while loop.

```
while ((nameStr.equals("")) || (!nameStr.matches(        1
    InfoConstant.USER_REGEX))) {
            ...                                          2
        }                                                3
```

Chapter 3

Interface Documentation

Chapter 4

Structure of the project

Bibliography

[1] Wikipedia contributors. (2018, December 24). Version control. In *Wikipedia, The Free Encyclopedia*. Retrieved 06:12, March 10, 2019, from `https://en.wikipedia.org/w/index.php?title=Version_control&oldid=875227317`

[2] Wikipedia contributors. (2019, March 10). Systems development life cycle. In *Wikipedia, The Free Encyclopedia*. Retrieved 06:13, March 10, 2019, from `https://en.wikipedia.org/w/index.php?title=Systems_development_life_cycle&oldid=887015682`

[3] Stolen, L. H. (1999). Distributed control system. *international telecommunications energy conference.*

[4] Murayama, T. (1991). Distributed Control System. *international conference on advanced robotics robots in unstructured environments.*

[5] Wikipedia contributors. (2019, March 6). Distributed control system. In *Wikipedia, The Free Encyclopedia*. Retrieved 06:18, March 10, 2019, from `https://en.wikipedia.org/w/index.php?title=Distributed_control_system&oldid=886468871`