# Lab1: 版本控制，代码质量和单元测试

1. 概述

目标：

体验版本控制和单元测试工具，体验代码质量和需求变更对开发工作的影响。

场景：

骨哥公司（BoneBrother Corp.）面临破产危机，大量老员工离职，留下一个未完成的星爸爸饮料价格计算工具。一组新入公司的职员想到了一条妙计——完成这个星爸爸饮料价格计算工具，只要将这个工具按客户的要求完成并提交，就可以让公司获得苟延残喘的资金，也能够开发新的项目。就这样，一组个性鲜明的员工决定一起努力完成这个工具。

任务：

对版本库中已有的代码进行改进，针对新需求修改代码添加功能

根据示例为要求的方法编写单元测试的测试用例

提交一份报告，谈三点感想，如实现过程中发现的代码的问题、位置与解决方法，或谈谈哪些实践经验对你们有所帮助。长度请控制在一千字以内，字数不作为评分的标准，简要，清楚即可。

1. 版本控制

目标：熟悉 git 客户端的基本操作，对其概念有初步理解。

1. 各组将项目：<https://github.com/SELabTest/lab1>，fork到自己的小组内部，小组内部人员合作完成lab任务，并且将项目地址发给各个小组负责的TA
2. 小组成员每完成一次大功能进行一次提交，中间可以进行多次提交，在提交的Log记录中记录好本次记录完成的任务
3. 将项目文档放入项目仓库的wiki中，或者建立Github Page，描述项目，介绍心得体会。样式不作要求，整洁就好！！可以使用各种博客模板。
4. 软件设计与构造

背景：

老员工留下的星爸爸饮料价格计算工具只实现了最基本的功能，根据输入的饮料种类，size，和配料输出对应的饮料价格。现在我们需要对该工具进行升级和维护。

目标：

体验需求变更对软件开发的影响，体会中途接收一个软件项目进行开发和维护的难点，并对代码中的坏味道（bad smell）、设计模式、软件重构等概念进行初步理解。

3.1增加新的饮料。

说明：

1. 星爸爸咖啡店新进了一种咖啡叫Decaf Mocha: decaf coffee + 1 serving of Chocolate。
2. 用户能够选择这种新饮料并指定尺寸和配料，工具能够计算出其价格。

3.2为饮料增加新的size。

说明：

1. 原来咖啡店里的饮料只有large, middle, small三个尺寸，现在星爸爸咖啡店推出了超大杯的饮料，尺寸为grant。
2. 用户能够选择尺寸为grant的饮料，工具根据以下规则计算出尺寸为grant的饮料的价格。(该规则是对specification文件中第五条规则c的修改)

5.c The cost of sizes:

* + 1. Coffee beverages:
       1. Grant:$1.3
       2. Large: $1
       3. Medium: $0.7
       4. Small: $0.4
    2. Tea beverages:
       1. Grant:$0.9
       2. Large: $0.7
       3. Medium: $0.5
       4. Small: $0.2

1. 为饮料计算增加新的尺寸时，程序员应该考虑到程序的可扩展性，未来是否能够方便地添加其他尺寸。因此在进行软件设计时，需要认真考虑。

3.3 这个题目只有卓班的同学来做即可。修改工具的输入方式以支持多杯饮料同时计算价格。

说明：

1. 老工具只支持输入一种饮料配方然后立即计算出饮料的价格，现在需要支持在开始指定输入饮料的数量，然后再输入对应数量的饮料配方，最后统一计算出价格。
2. 以下为老工具的输入格式：

*<beverage name> <size> [<ingredient 1, ingredient 2, ingredient 3>]*

*e.g. a small mocha with two serves of milk*

*Mocha small milk milk*

*or:*

*houseblend large whipcream*

现在新工具也要支持以下的输入格式：

*<beverage number> <beverage name> <size> [<ingredient 1, ingredient 2, ingredient 3>] ; <beverage name> <size> [<ingredient 1, ingredient 2, ingredient 3>] ...*

*e.g. 2 cups of coffee which are* ***a small mocha with two serves of milk*** *and* ***a large White Tea with Ginger***

*2 Mocha small milk milk ; White Tea large Ginger*

3.新工具不仅要支持新的输入方式，之前的输入方式应该也能够不受影响，同时在设计时也要考虑到程序的可扩展性，未来是否能够方便地其他输入方式。

3.4 软件重构，改善游戏的设计、代码的可理解性和可维护性

说明：

1.各组拿到的代码都有一些不足之处，一些是代码中的坏味道，另一些则是设计

上的问题，请大家找到并改正之。

2. 在进行这部分工作时，请务必在 git的提交信息中简明地说明发现的问题，也

请在最后的报告中谈谈自己的做法和感想。

3. 对代码的改进可以从以下几个方面着手：面向对象设计、代码效率、代码风格

等，代码中的任何问题都可以考虑。

4. 代码重构、代码坏味道和软件设计模式方面的相关知识请自主进行初步的学习。

5. 部分参考资料：

代码重构：http://en.wikipedia.org/wiki/Code\_refactoring

代码坏味道：http://en.wikipedia.org/wiki/Code\_smell

软件设计模式：http://en.wikipedia.org/wiki/Software\_design\_pattern

1. 单元测试

背景：

我们每编写完一个函数之后，都应该对这个函数的进行测试，这样的测试我们称之为

单元测试。传统的编程方式，进行单元测试是一件很麻烦的事情，你要重新写另外一个程序，

在该程序中调用你需要测试的方法，看看是否有错。于是这里推荐使用 JUNIT 来进行单元测试。

目标：

体会单元测试，并学会使用 JUNIT 来进行单元测试。

任务：

参考给出的 CoffeeBasedTest.java文件，对项目中的 其他几个类进行单元测试的测试用例的撰写以及运行。体会单元测试在编程开发中的作用。

说明：

1.在 eclipse 中配置 JUNIT。

右键项目打开项目 coolJUnit 的属性页 -> 选择“Java Build Path”子选项 -> 点选“Add

Library”按钮 -> 在弹出的“Add Library”对话框中选择 JUnit。

点击项目选择新建“JUnit Test Case”，选择要进行测试的类（Class under test），点击下一步，选择要进行测试的方法。

2.请尽量详细的对每个方法的各种可能的输入进行单元测试。