软件工程第十一组 Lab2 文档

一、各个功能模块的实现思想（包括如何体现可扩展性和易维护性）

1. AccoutService 模块：

1） signup 函数实现思想：

● signup 函数中，写了 2 个子程序 checkUsername(String userName),checkPassword(String pass)分别用于判断注册用户名和密码是否符合规范，如果不符合规范，则抛出对应的错误并在 signup 函数中 catch 后将错误信息记入日志，并继续抛出当前错误到调用函数中。

当对用户名和密码的检测全部通过之后，调用 UserRepositoryImpl 中的 createUser 函数创建用户,创建成功后将登陆成功的记录写入 logger 中。

●易维护性：

①对用户名和密码的判断都分别写在子程序中，如果出现错误可以立刻检查对应子程序的代码实现。

②运用 try-catch，便于知道报错的内容，由此找到运行错误的原因。

●易扩展性：

①判断用户名和密码都写成了子程序，便于之后添加对注册所需的新的要求，比如用户名必须全是英文，密码需英文数字结合等未来可能出现的要求。

②整个程序写了 try-catch，非常方便在未来添加不同的注册可能出现的错误类型与提示。

2） login 函数实现思路：

●login 函数中，实现了子程序 checkUserName(String userName)的复用，用于判断用户输入的用户名是否规范，比如是否为空、是否存在等。同时写了 checkUserInfoMatch(User user)子程序，用来做用户信息的匹配，现在只是检查密码。这些子程序检测到用户名不规范或用户登陆失败时会抛出运行异常，被 login 函数 catch 以后使用 logger 记入日志，然后继续向上抛出到调用 login 的函数之中。

接着，调用 UserRepositoryImpl 中的 getUser 函数，得到 csv 文件中记录的该用户名对应的用户信息，该过程若捕获到调用函数抛出的异常，就同样使用 logger 记入日志，然后继续向上抛出到调用 login 的函数之中。若无异常，就比较两者密码的差别，将 hasLogin 这个全局变量赋值为比较结果，若相同就成功登陆，logger 记入成功信息，正确返回；不相同就登陆失败，logger 记录失败信息，并抛出运行异常。

●易维护性：

①将函数的逻辑分离，子程序职责单一，若以后出了问题，很容易找到问题所在。

②运用 try-catch，便于知道报错的内容，由此找到运行错误的原因。

●易扩展性：

①用了两个子程序，实现了逻辑的一定程度的分离：对于检查用户名规范的子程序的复用，降低代码重复率和耦合度；对于检查用户匹配的子程序，之后可以任意扩展除密码外其他信息是否匹配。

②整个程序写了 try-catch，非常方便在未来添加不同的注册可能出现的错误类型与提示。

3) getStatus 函数实现思想：

●建立一个全局变量，每次登录成功之后，都会对该全局变量进行改变，获取当前的登录状态，只需要返回该全局变量的值，并且在返回之前将当前状态记入日志

2. PriceService 模块实现思想：

●根据提供的价格计算公式，对提供的 order 遍历，取值处理，由于直接调用后端接口，所以在编写函数时采用防御式编程，对杯数为负的情况进行了判断抛出 RuntimeException 并写入日志记录。

在重载 coffee 子类 Espresso 和 Cappuccino 的 cost()函数时根据公示写出函数，由于杯型价格不同，专门用函数实现，为了防止后端调用时杯型可能非法取值，所以进行了检测，检测错误采用默认杯型并记入日志。

●易维护性：

①对程序进行抽象拆分，把函数又划分为几个有意义的小的功能模块。对函数抽象，使得单个函数职责单一。PriceService 的 cost()函数划分一个 checkInteger(Integer integer)来检测杯数是否为负数。Espresso 和 Cappuccino 的 cost()函数划分 priceOfSize()和 checkSize()函数分别用来返回杯型对应价格和检测杯型取值是否正确。这两个函数在两个实例中作用一样，为了减少代码重复率将其作为 coffee 类的函数，

②对函数都进行了有意义的注释标识各个函数的功能。

●易扩展性：

①对函数抽象，使得单个函数职责单一，使得职责扩展时，只需要修改单个函数逻辑。比如 Espresso 和 Cappuccino 的 cost()函数划分 priceOfSize()用来返回杯型对应价格。当杯型发生变化，只需要修改该函数，而不需要对 cost 函数进行修改。包括所有起 check 作用的函数，当所需要 check 的内容变复杂也只需要修改该函数。

3. Main 方法的实现

●根据一般流程，提示用户输入命令（注册-登录-点单），点单时，一旦检测到非法命令，立即中止程序

●易维护性：

①将提示用户输入和获得用户输入，都写成具体的方法，主要业务逻辑写在 main 方法，便于对各个模块具体功能的维护

●易扩展性：

①将每个提示用户输入和获得用户输入的地方都写成具体的方法，方便以后加入界面时，在界面上面获得这些输入，而不必在 main 主程序中作大的改动

②每次登录之后，都会对一个循环外部的变量 status 做出相应的改变，便于以后加入 session 机制保持当前的登录状态

③全局变量中，创建一个咖啡种类的全局变量，如果在以后具体实现中推出其他咖啡种类，直接在该全局变量中作修改，以便之后用户提示中出现该选择

二、遵循的代码规范：

1. 缩进：tab 键缩进。

2. 分行: 遵循 if(){
Content;
{
形式且满足 if 条件的语句进行换行缩进。每条语句都进行分行。
3. 括号：复杂条件判断中使用括号区分优先级。
4. 命名：变量名遵循驼峰命名法并以用意命名；子程序名遵循驼峰命名法且使用动宾结构表明用意；
5. 注释：放在判断/代码头部表明意思；使用英文
6. 函数：函数内部验证参数正确性；每个函数只干一件事；异常处理并记录；不必要的变量设置为私有成员 private；logger 记录运行日志并使用常量字符串来标识程序运行状态；

三、实现中遇到的问题与解决方案

1. 在和同学们用 git 进行团队编程时，出现过代码上传覆盖 merge 出错等沟通不利的情况，也有过对某个变量的约定不明而出现错误的情况，这些都通过积极的沟通以及代码回滚（版本管理）解决了。
2. 在写 main 函数中，对于输入非法性检查应该写在 main 中还是相应函数里产生问题，通过沟通与软件工程课程上的指导解决了。
3. 刚写代码时没有搞清楚具体应该对抛出的异常进行什么操作，只是简单的抛出异常使程序停止运行，后来通过讨论学习又将运行时产生的异常都 catch 住，再统一向 main 函数中抛异常，更加深入了解了异常抛出的用处。
4. 在团队合作过程，我们发现交流和注意细节是非常重要的。因为胡宵宵和石睿欣的电脑是 mac 系统，而张涵涵和宋怡景的电脑是 windows 系统。在 mac 系统在 push 的时候，可能会将系统隐藏文件传上去，如.DS_store，而这种情况可能会对 windows 系统的 pull 或运行产生一定的影响。
5. 对于一次错误信息可能被 logger 多次的情况，我们采取的措施就是在子程序中只抛异常，不进行 logger 的记录。而在功能实现函数中捕获异常的时候，先用 logger 记录再向上抛出即可。
6. 对 coffee 的 csv 文件理解不够清晰，导致以为该文件会在创建 coffee 实例的时候，对该 csv 文件作修改，因此在 main 中直接通过具体的咖啡类创建实例，后来通过研读文档理解，在 main 中做出相应的修改
7. 第一次提交的代码在华为云的代码检查中提示致命错误，原因为无限循环，没有终止条件，在对代码的人为理解中，该循环是可以退出的，并且在真实运行中也可以正确退出，但是静态扫描中，会认为 while（true）是一个无限循环，没有结束条件，该问题通过改变循环条件与结构解决

四、小组分工

姓名-commit 用户名-分工

石睿欣 -- achillessanger -- sign up 函数实现

胡宵宵 -- 胡宵宵 -- login 函数实现

宋怡景 -- songyijing -- check status 函数实现&main 函数测试

张逸涵 -- zyhlx -- price service 实现