Java实验报告

徐云凯 1713667

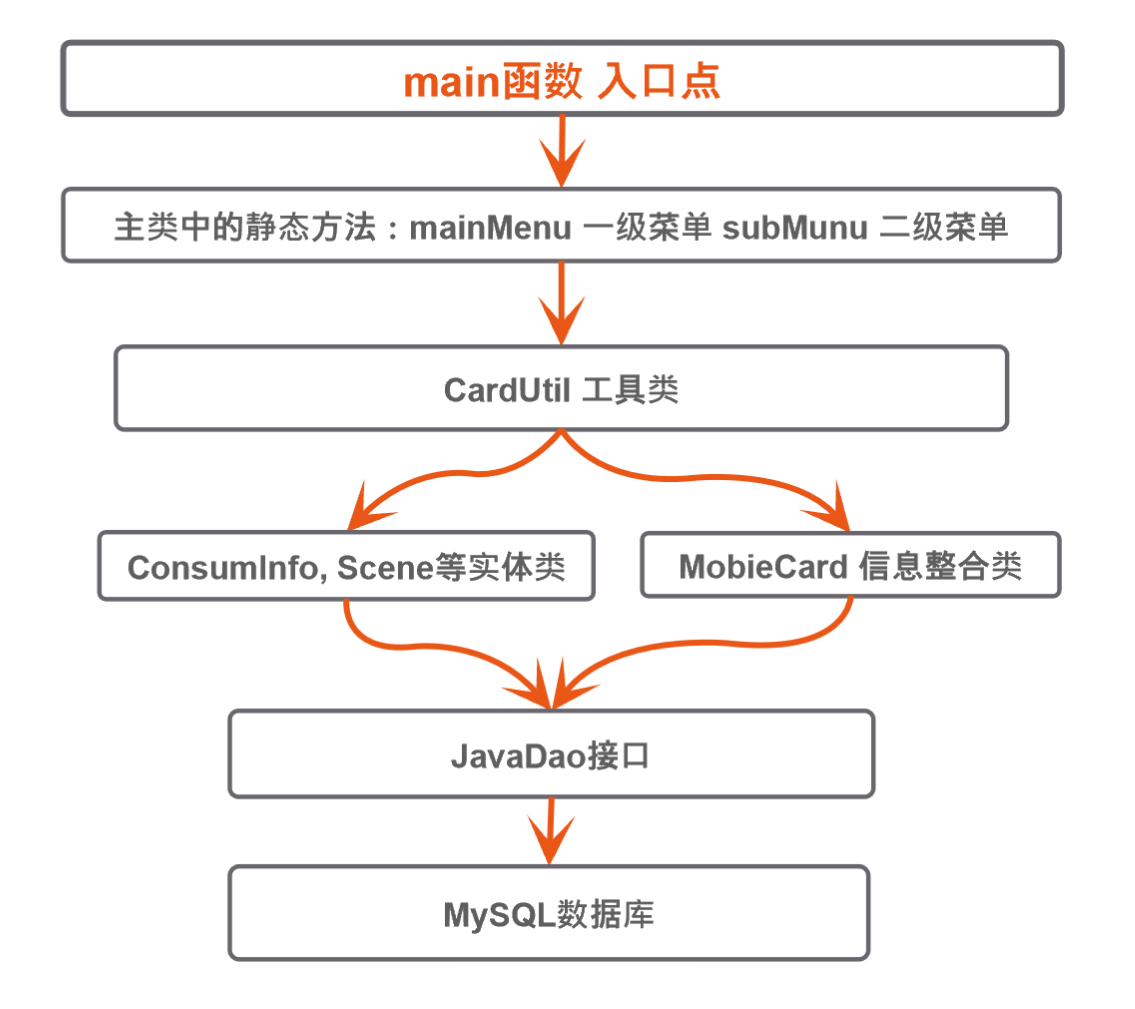
[已编译的JavaDoc](doc/index.html)

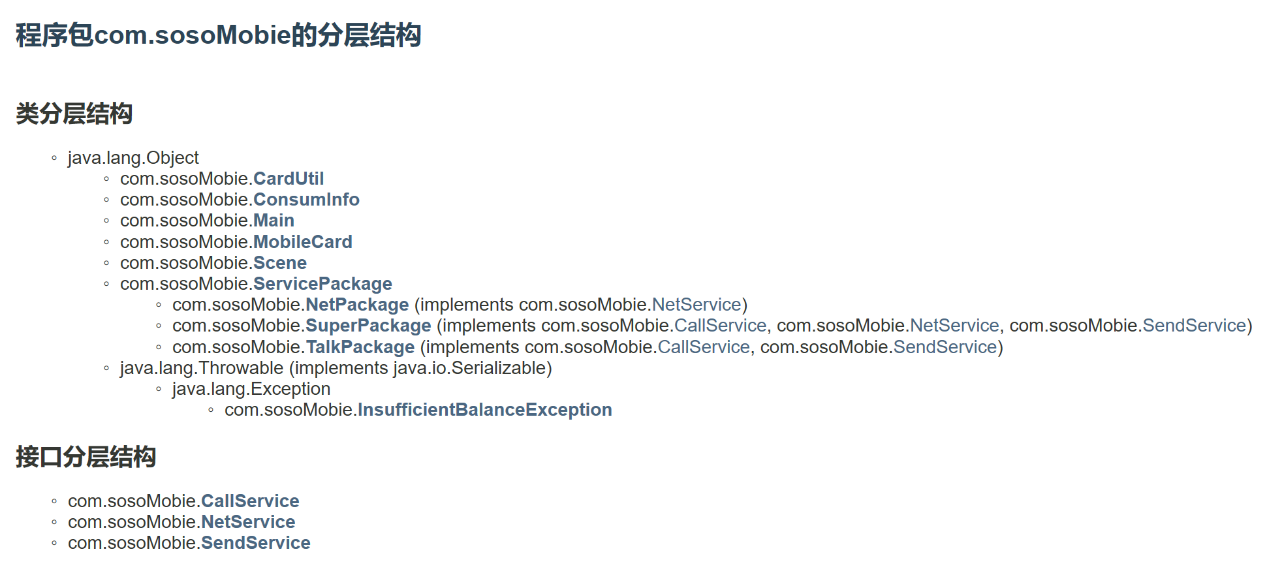
### 问题分析

#### 整体结构

项目需要能够针对不同的套餐使用不同的具体处理方式，但是需要共用同一套方法调用，故应当为每一个套餐设置一个类并在其中实现所有套餐间有差异的方法。所有套餐类共同继承自同一父类ServicePackage，从而利用多态实现灵活调用。而不同的套餐具有重复内容，可能使用到同一方法，故使用接口实现不同的类调用类似方法，提高代码重用性。此外，项目要求了大量的菜单页与具体功能性应用，不适合分散到各个类，故使用一个单独的工具类CardUtil，整合大多数工具方法。

另外，对于项目要求的各个一级二级菜单，在主类中建立单独的静态方法放置一级二级菜单代码，菜单只进行对工具类的调用而不进行具体实现，从而实现了main函数-菜单方法-工具类-底层类具体功能实现 的四级层次分离。最大程度上提高了代码的可复用与修改的灵活性。



故依据以上要求，项目各个类与接口结构如下：

其中，除了在项目中要求的几个类之外，自定义了InsufficientBalanceException异常类，用于在处理消费场景时可能遇到的余额不足异常。

#### 二. 各功能实现

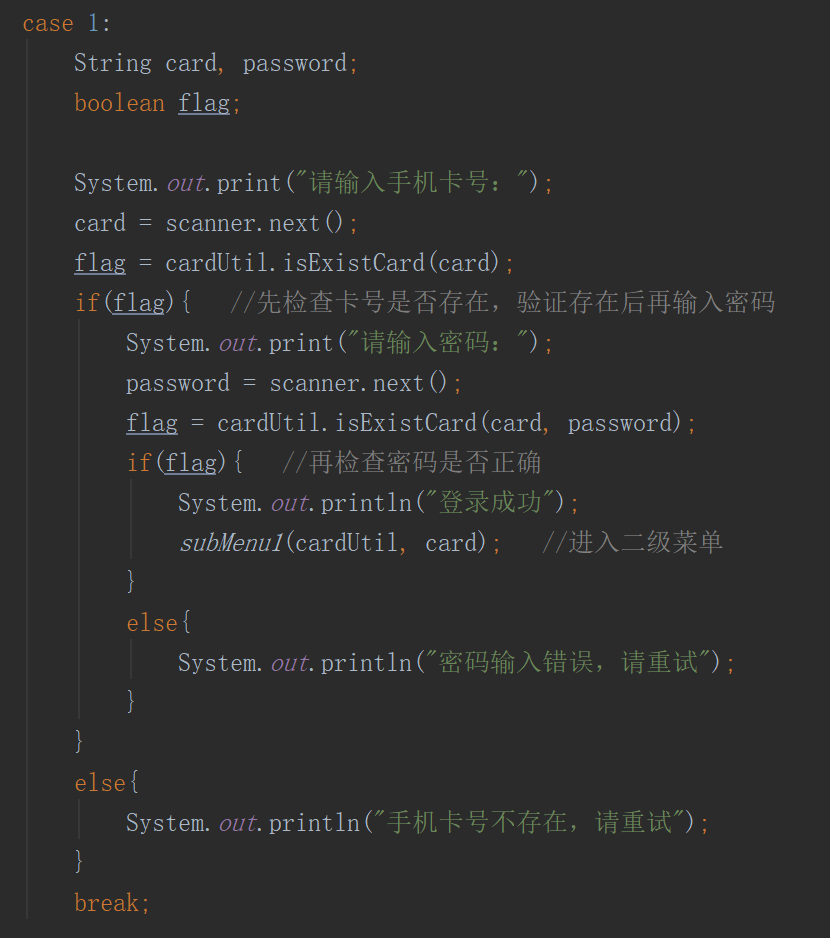
（按项目中菜单项次序排列）

##### 一级二级菜单

* 1. 使用do...while循环反复执行对菜单项目的选择

内部使用switch-case语句选择选项执行，对于非法输入使用try-catch(InputMismatchException e)处理，并新建scanner对象以刷新输入缓冲区

* 1. 一级菜单 第一选项

 逐层验证卡号密码，一旦不正确就返回重新输入，都验证成功后调用二级菜单方法。

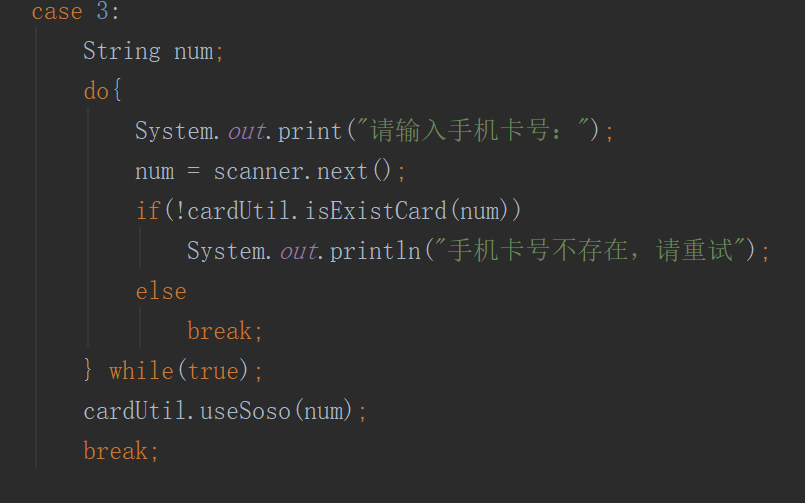
* 1. 一级菜单 第二选项

调用CardUtil类对应方法生成新卡号，并格式化输出，输入时检查输入合法性。

随后输入姓名密码预存话费等信息，输入时除string类型外均检查合法性。

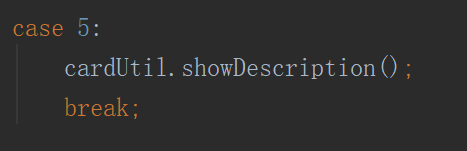
最后直接扣除首月月费，存入后台。

* 1. 一级菜单 第三选项

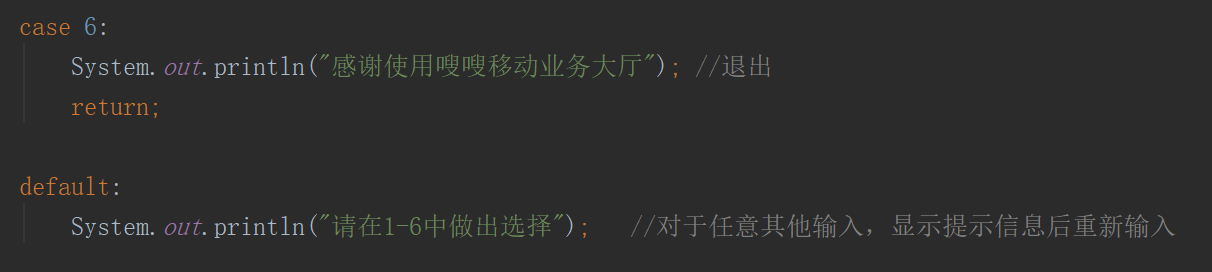
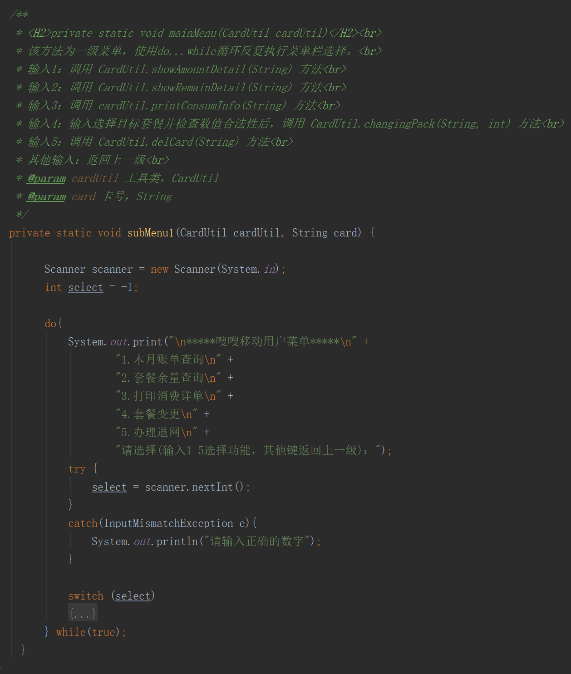
直接调用useSoso方法

* 1. 一级菜单 第四选项

输入并检查合法性后，调用工具类的chargeMoney方法

* 1. 一级菜单 第五选项

调用工具类方法

* 1. 一级菜单 第六选项及其他输入
  2. 二级菜单

类似一级菜单循环方式

* 1. 二级菜单功能

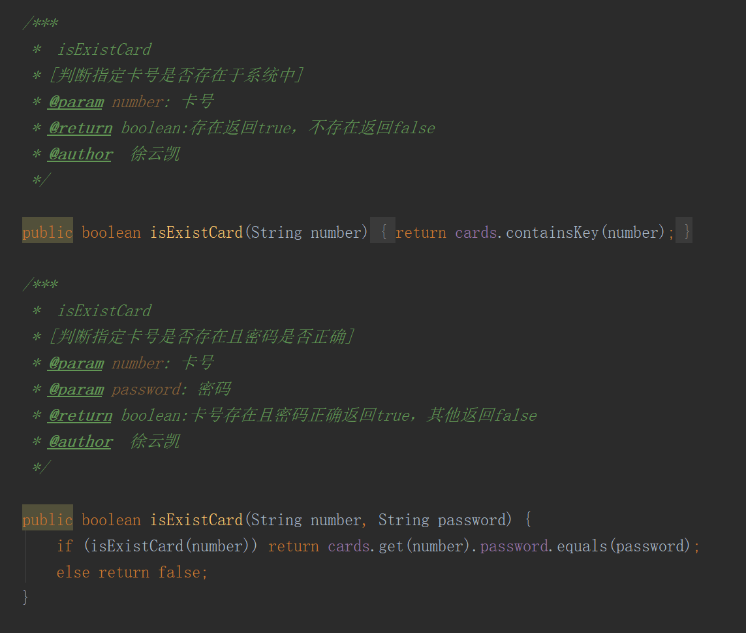
分别调用工具类对应方法

##### 工具类

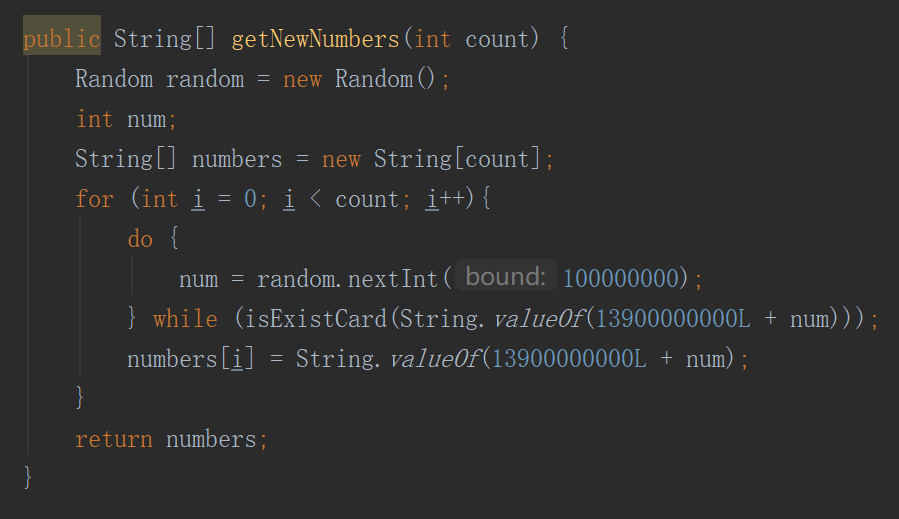
* 1. 构造函数与初始化

构造函数初始化所有列表并载入预置的账号。构造函数的最后调用initScenes()函数初始化List<Scene> scenes完成对场景的初始化

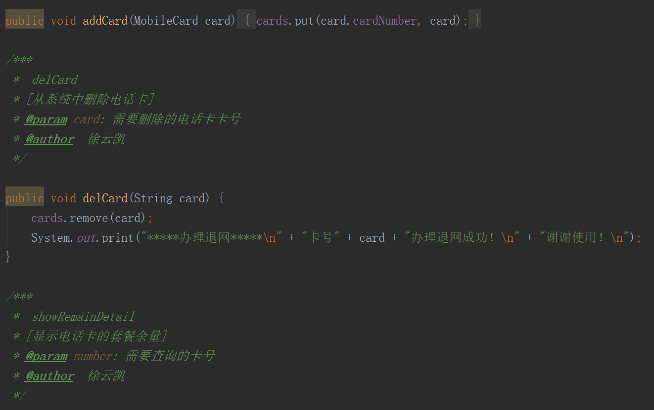
* 1. 判断卡号是否存在及卡号密码是否正确

两个函数功能接近，使用函数重载。前者直接使用Array的containsKey方法查询卡号，后者使用get检查正确密码是否符合传入的密码参数。

* 1. 获取新卡号

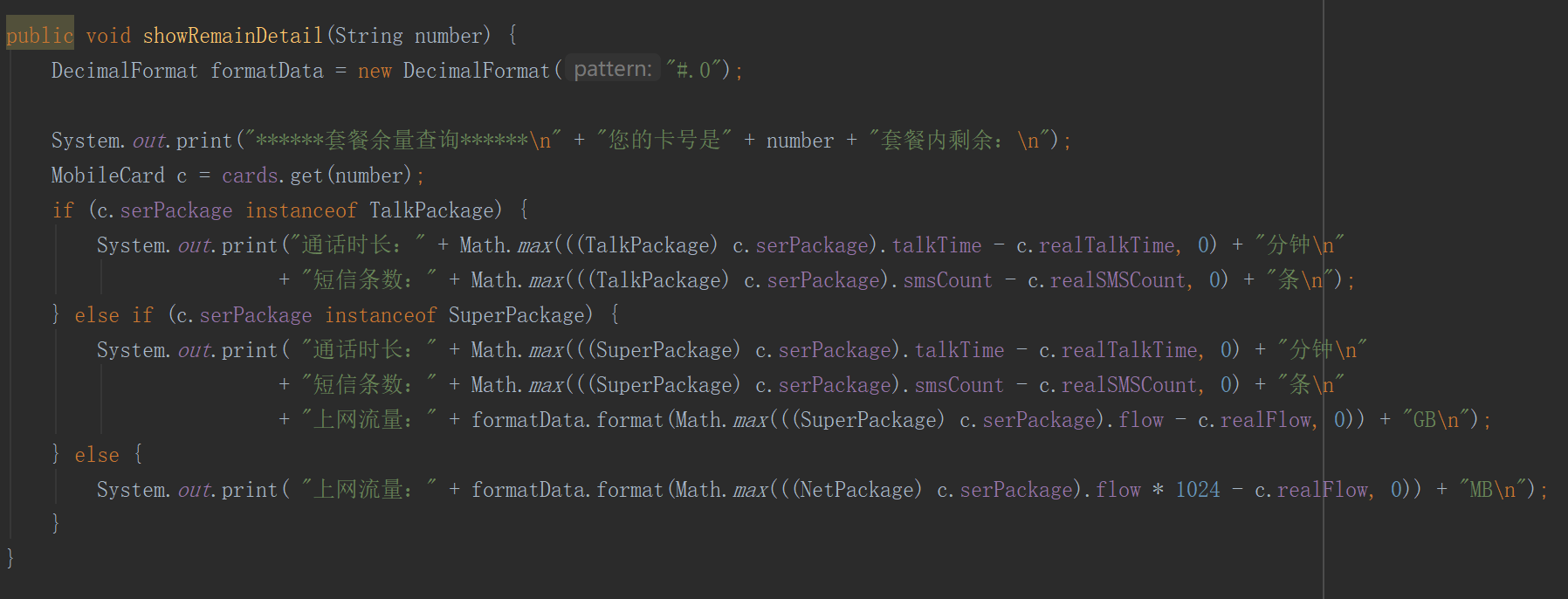
使用random.nextInt(int)获取不重复的新号码，检查不重复之后加入输出数组中。

* 1. 添加与删除

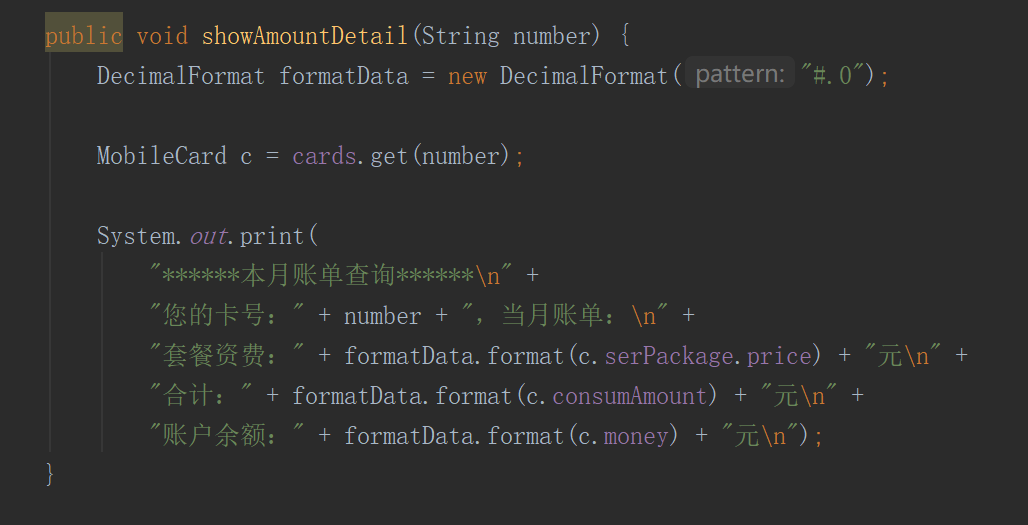
调用array的put,remove方法实现增加删除

* 1. 显示套餐余量

使用instanceof判断套餐类型之后调用对应数据

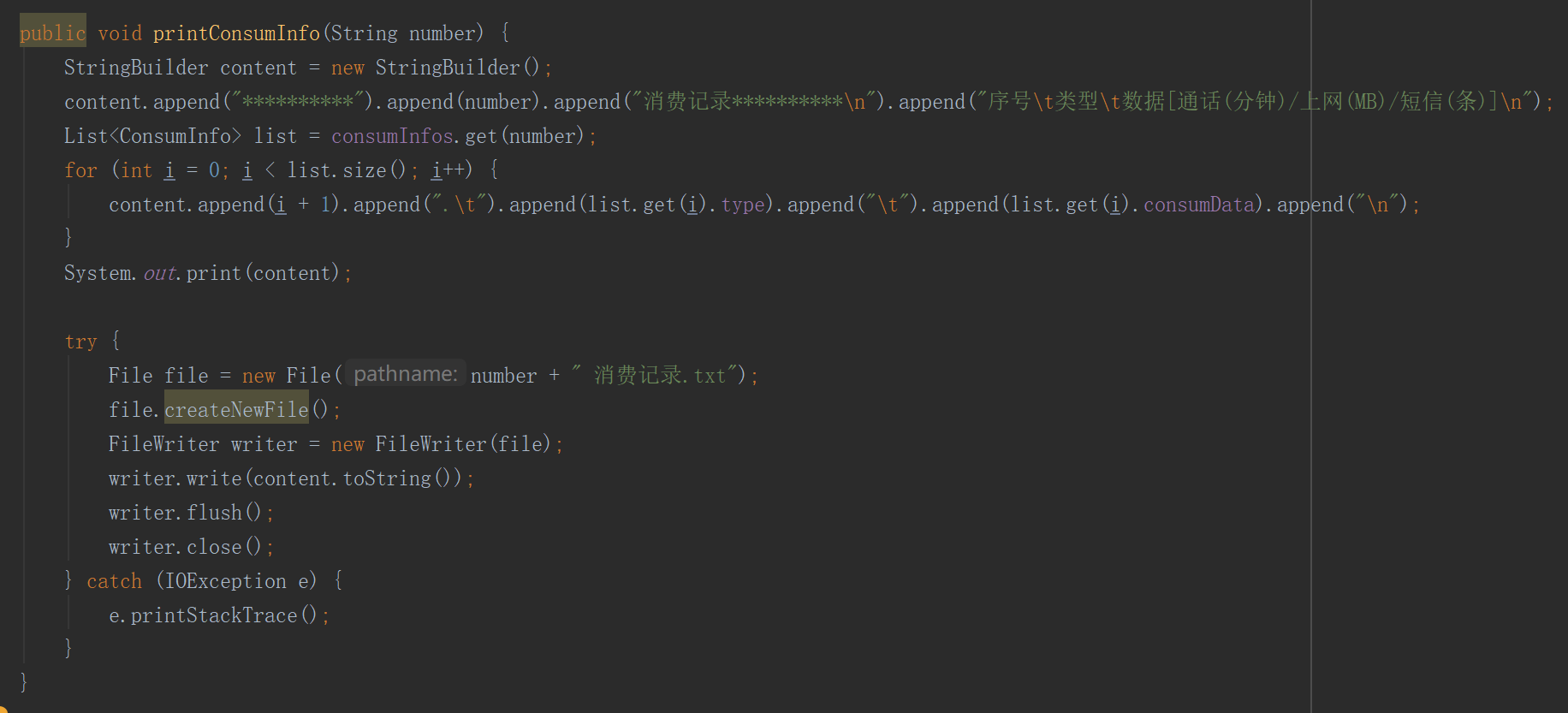
格式化输出使用Java的DecimalFormat类

* 1. 本月账单查询

格式化输出使用Java的DecimalFormat 类

* 1. 打印消费记录

为避免频繁调用print函数，将所有数据使用append添加到缓冲字符串，最后一次性输出。

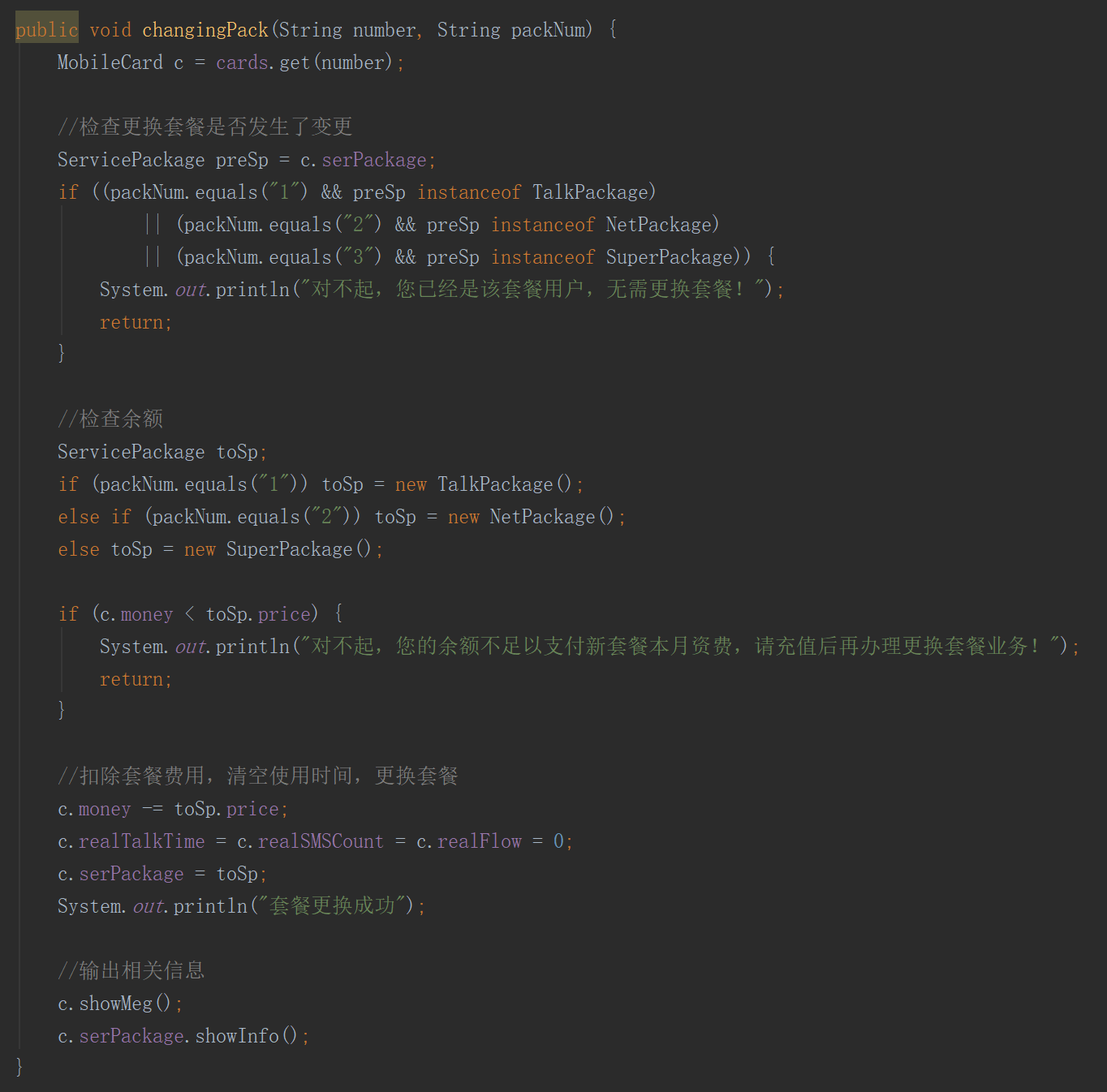
向屏幕输出完成后，调用File类将缓冲字符串写入文件

* 1. 显示资费说明

资费说明.txt被预先制作好放置于项目根目录下，程序会尝试读取并输出

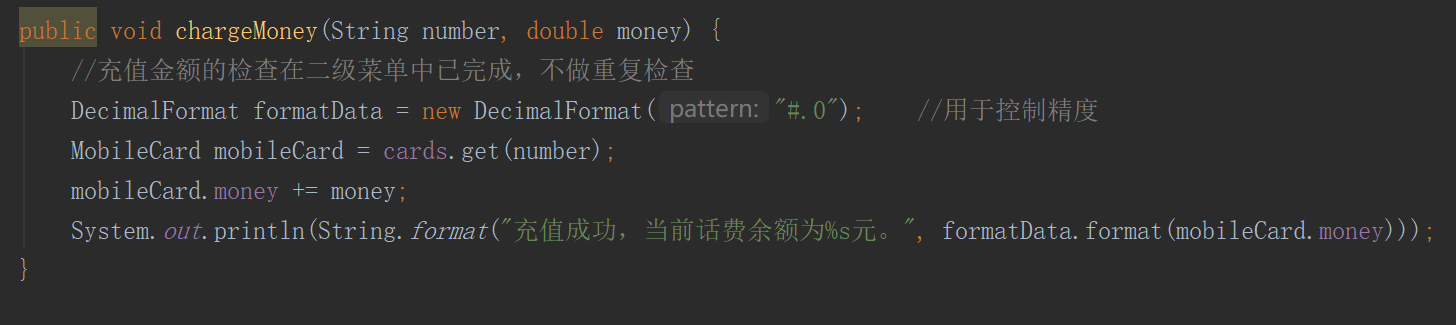
* 1. 更换套餐

先使用instanceof检查套餐是否即将更换

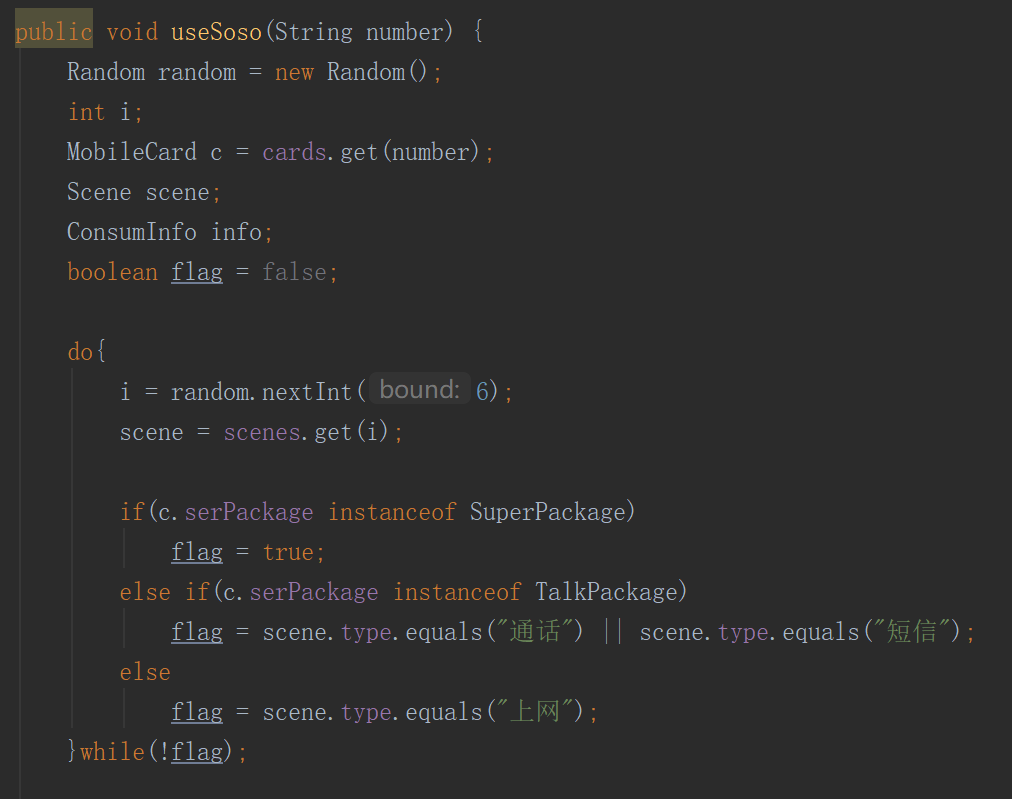
再检查余额是否满足新套餐月资费，最后更换套餐，扣除余额与各项使用时长并显示新套餐与账户信息。

* 1. 充值

最低充值限额检查被分离到菜单方法中进行，这里不做重复检查。

充值完成后按要求的精度输出。

* 1. 使用嗖嗖

首先使用random类生成随机数，调用对应的场景，检查该卡套餐确实可以进行该场景后继续，否则重复生成随机场景。

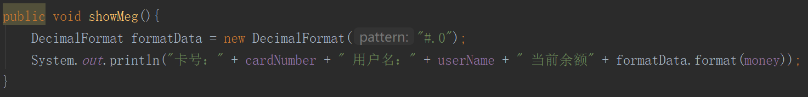
随后分场景类型调用对应套餐的功能性方法，并通过异常捕获，检查是否出现了余额不足。并进行相应的异常处理。具体扣款与检查逻辑分离至套餐内部实现。一切完成后添加消费记录。

##### 电话卡实体类

* 1. 构造函数

为方便工具类中的方法，这里保留无参构造函数，同时也编写了有参构造函数。

* 1. 打印电话卡信息

使用DecimalFormat控制精度

##### 用户信息类

* 1. 构造函数

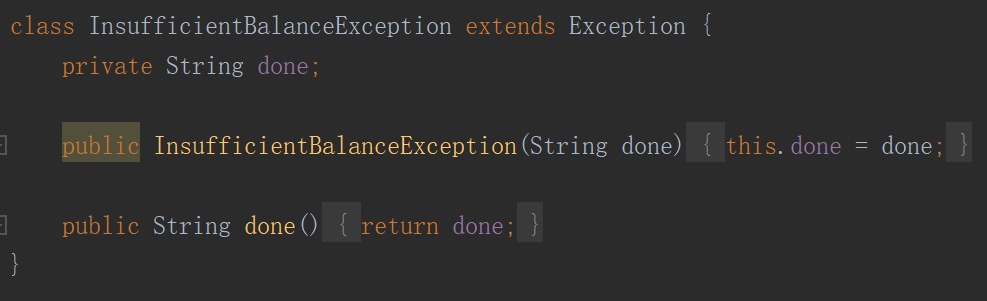
初始化用户通话记录中各项变量

##### 场景类

* 1. 构造函数

初始化场景各项成员变量，包括消费类型，数额与详细信息

##### 异常类InsufficientBalanceException

* 1. 异常类包括一个成员变量完成度done，用于记录在发生余额不足异常时，已经完成的消费数量，构造函数为其setter，成员方法print为其getter。

##### 套餐类的公共抽象父类ServicePackage

* 1. 成员变量price表示套餐月租价格，方便多态中的调用。成员方法public void showInfo()用于打印套餐详情。

##### 三个套餐类

* 1. 各个套餐类中自带有各自有的套餐余量成员变量（talkTime、smsCount、flow）与相应的功能函数（call、send、netPlay），构造函数将会把套餐余量预置为套餐每月额度。
  2. 每个类的三种功能函数逻辑大体一致，这里以SurperPackage的call成员方法为例展示。
     1. Call方法

该方法分多钟情况进行套餐余量检查：

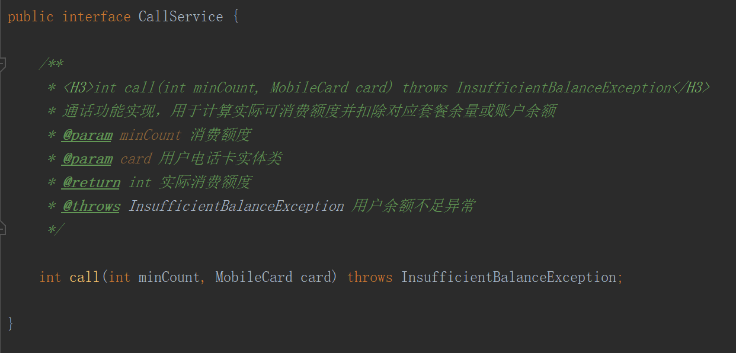
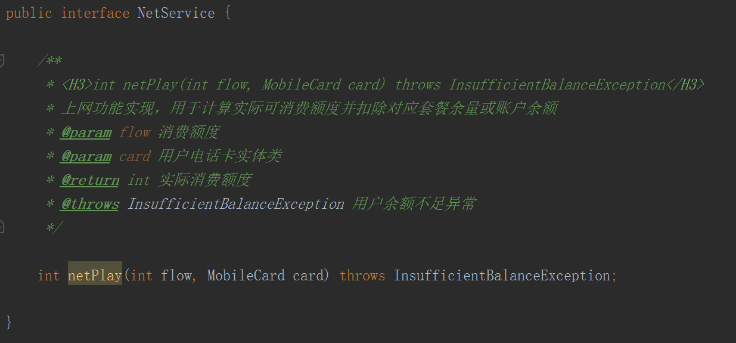
1. 套餐未用完但套餐余量不足以完成本次消费，将会扣除套餐余量后合并入情况2中继续执行
2. 套餐已经用完

检查余额判断是否能够完成消费需求，抛出异常或扣费返回

1. 套餐未用完且套餐余量充足

直接扣除对应套餐余量后返回

##### 三个接口

* 1. 每个接口中只有对应的功能函数的声明。

# 反思

##### 类的结构

在按照作业要求实现三个接口的时候发现接口的实现部分代码冗余度较高，而三个类在继承其父类的时候反而不需要使用继承机制，可以通过接口来实现。所以如果想要进一步精简代码的话，可以尝试把父类ServicePackage替换为接口或并入父类，将三个接口合并为一个不抽象的父类并直接实现相关方法，子类中将父类方法封装并选择性调用。结构如下：

同时为了保证各个套餐不会调用自己不存在的消费方法，在子套餐类中对消费方法进行封装，工具类只能调用封装好的消费方法。