

基于关联规则的商品推荐算法

姓名：赵林华

学号：2016207275

专 业：软件工程（服务外包）

院 系：数据科学与软件工程学院

班级：java中文04班

# 摘要

随着中国互联网的飞速发展，出现了很多连接客户与商家（B2C）的互联网服务平台，例如天猫，淘宝，京东，美团等。就平台而言，如何更好的服务客户，获取更多的注册量和使用量，以此来掌握市场占有率便成为了一个很重要的问题。就消费者而言，互联网平台的出现打破了地理上的界限，将众多的商家一并展示到了客户的面前，如何从琳琅满目的商品中挑选自己喜欢的，便成为了消费者面临的一个问题。就商家而言，在众多的竞争者中，如何将自己商品向更合适的人展示，也成了商家必须要解决的问题。

为了解决三方所面临的问题，一种更完善的进行商家与消费者之间资源调度的算法——推荐算法——日渐成熟起来。

而本论文内容包含两大部分，一是介绍基于关联规则的推荐（Association Rule-based Recommendation）算法，二是实现一种简单的，基于关联规则的推荐算法程序。当消费者输入某一商品时，会将此消费者过往消费记录里，所有与此商品一同购买过的商品按照同时购买次数由多到少排列

关键词：推荐算法、关联规则

目录

[基于关联规则的商品推荐算法 1](#_Toc12783230)

[摘要 2](#_Toc12783231)

[第一章 基于关联规则的商品推荐算法 4](#_Toc12783232)

[1.1 算法简介 4](#_Toc12783233)

[1.2 算法举例 4](#_Toc12783234)

[1.2.1数据准备 4](#_Toc12783235)

[1.2.2 特征交叉 5](#_Toc12783236)

[1.2.3 生成关联规则 5](#_Toc12783237)

[1.2.4 置信度(Confidence)和支持度(Support) 5](#_Toc12783238)

[第二章 实践项目概述 6](#_Toc12783239)

[2.1 简介 6](#_Toc12783240)

[2.2 程序设计相关 6](#_Toc12783241)

[2.3 工具配置相关 6](#_Toc12783242)

[第三章 数据持久层设计 6](#_Toc12783243)

[3.1 实体类设计 6](#_Toc12783244)

[3.2 数据库设计 6](#_Toc12783245)

[3.3 DAO层设计 7](#_Toc12783246)

[第四章 服务层设计 10](#_Toc12783247)

[第五章 主函数设计 11](#_Toc12783248)

[第六章 核心实现代码 12](#_Toc12783249)

[第七章 结语 14](#_Toc12783250)

# 基于关联规则的商品推荐算法

## 1.1 算法简介

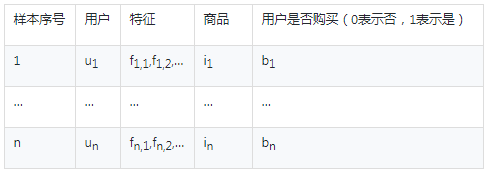
基于关联规则的商品推荐算法可以理解成“根据消费者以往的商品消费关联记录，来为消费者推荐最有可能需要的商品”。也即，此推荐算法是根据客户个人历史购买记录作为处理的数据，所谓“关联规则”指的就是“商品之间购买的关联度”。

假如某位顾客十次购买A商品的同时，有八次购买了B商品，五次购买了C商品，三次购买了D商品，显然，当顾客再次购买A商品时，有80%的概率购买B商品，50%的概率购买C商品，30%的概率购买D商品。也即，当顾客再次购买A商品时，最应该向顾客推荐的是B商品，其次是C商品，再次是D商品，最后是其他的商品。

对于一个规则X->Y（即X商品与Y商品的关联），有两个指标对该规则进行衡量。一个是支持度，表示在所有样本数据中，同时包含X和Y样本的占比。另一个是置信度，表示在所有包含X的样本中，包含Y的样本占比。

## 1.2 算法举例

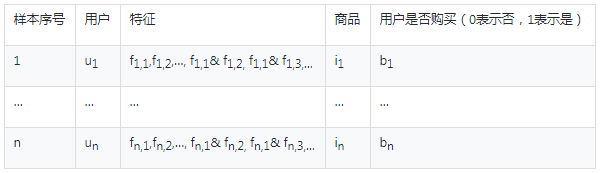
### 1.2.1数据准备



其中用户特征可以是用户历史购买的商品ID，也可以是用户属性特征，例如：年龄、性别、居住地等等。

### 1.2.2 特征交叉

在上表中，对于同一个样本所有特征两两交叉，生成长度为2的特征规则，合并原来的长度为1的特征规则，得到关联规则数据输入表如下：



上述表中只用长度为1（原始特征）和2（原始特征两两交叉）的规则作为后面rule的候选集，不做长度为3的规则主要的考虑点是降低规则空间复杂度。

### 1.2.3 生成关联规则

首先把上表的特征展开，使得一个特征一条记录，如下表：



### 1.2.4 置信度(Confidence)和支持度(Support)

**置信度**，表示在所有包含X的样本中，包含Y的样本占比。**支持度**，表示在所有样本数据中，同时包含X和Y样本的占比。

通俗一点来说，置信度即是，购买了X商品后购买了Y商品的概率；支持度即是，同时购买X，Y商品的概率。

# 实践项目概述

## 2.1 简介

此项目架构分为三层。最底层为数据持久层，包括数据库、实体类、DAO的设计与实现。第二层是服务层，包括Service类，通过调用DAO的各种方法实现最上层的业务需求。最上层是主方法，包括此项目的运行主方法。

## 2.2 程序设计相关

本项目采用JavaSE语言为设计语言，JDK版本为1.8。

数据库类型为MySql数据库，数据库版本为 MySql 5.5。

程序框架采用Hibernate 5.4.2。

数据库连接驱动为 mysql-connector-java-5.1.7-bin.jar 。

## 2.3 工具配置相关

本项目运行操作系统为 Windows 7 旗舰版。

IDE为Eclipse 2019-03版。

数据库可视化工具为 Navicat。

# 第三章 数据持久层设计

## 3.1 实体类设计

本项目中的实体类共有四个，其分别是：顾客类(Consumer)、商品类(Production)、记录类(Record)、记录详情类(Detail)。

顾客类(Consumer)的属性包括唯一标识字段id(String)及顾客姓名name(String)，用于存储顾客信息。由于本项目只是简单实现功能，因此顾客类不作为重点。

商品类(Production)的属性包括唯一标识字段id(String)及商品名name(String)，用于存储一条商品的信息记录，且id字段会与记录详情类(Detail)相关联。

记录类(Record)的属性包括唯一标识字段id与顾客标识字段consumerId(String)，其中consumerId属性与顾客类(Consumer)的唯一标识字段id相关联。

记录详情类(Detail) 的属性包括唯一标识字段id(String)、消费记录标识字段 recordId(String)与商品标识字段productionId(String)。其中消费记录标识字段 recordId与记录类(Record)的属性id相关联，商品标识字段productionId与商品类(Production)的属性id相关联。

## 3.2 数据库设计

数据库名：algorithm；编码格式：UTF-8 Unicode；排序规则：utf8\_general\_ci。

本项目应用了Hibernate框架，将实体类与数据库表做了ORM映射，一张表对应一个实体类。详细对应情况如下：

顾客表（consumer）对应于顾客类(Consumer)，表字段id（varchar，primary key）对应于属性id(String)，表字段name(varchar)对应于属性name(String)。

商品表(production)对应于商品类(Production)，表字段id（varchar，primary key）对应于属性id(String)，表字段name(varchar)对应于属性name(String)。

记录表（record）对应于记录类(Record)，表字段id（varchar，primary key）对应于属性id(String)，表字段consumerId（varchar）对应于属性consumerId（String）。

记录详情表(detail)对应于记录详情类(Detail)，表字段id（varchar，primary key）对应于属性id(String)，表字段recordId（varchar）对应于属性recordId（String），表字段productionId（varchar）对应于属性productionId（String）。

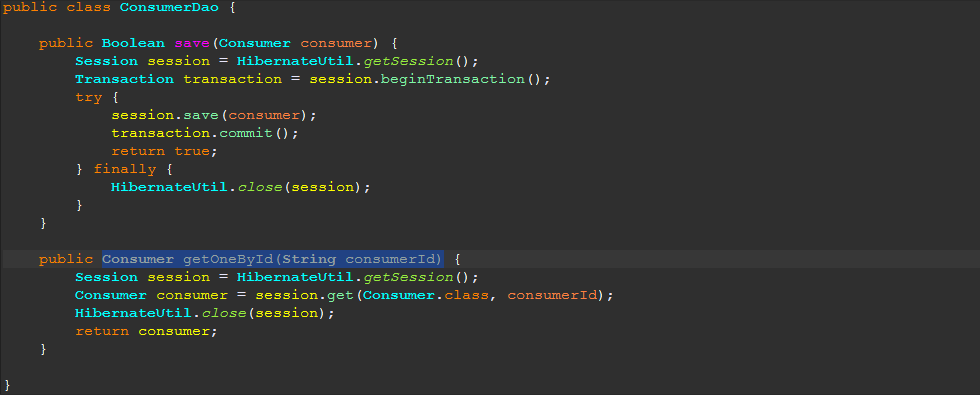
## 3.3 DAO层设计

根据“一个Dao类操作一个数据库的表，一个方法完成一个功能”的原则来设计Dao层。

ConsumerDao类用于实现对顾客表（consumer）的增删改查操作。

ConsumerDao定义了两个方法：Boolean save()和Consumer getOneById()。

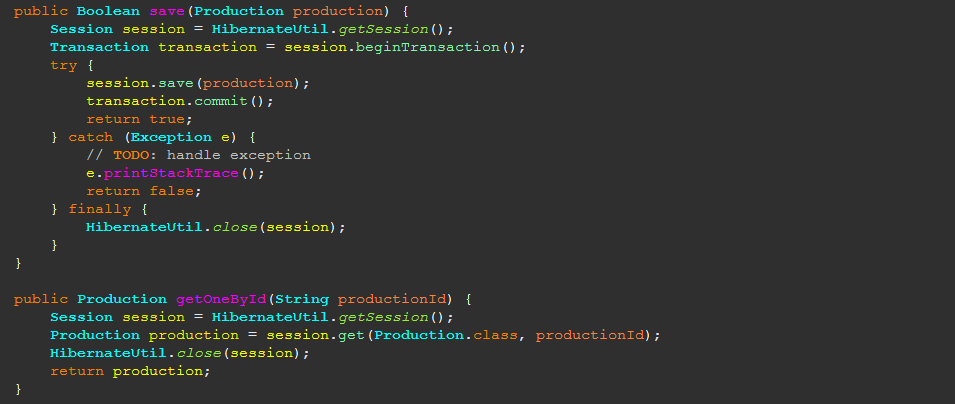
save（）用于将传递过来的Consumer对象保存到数据库中。getOneById方法用于通过id查询一个Consumer对象。具体实现如下：

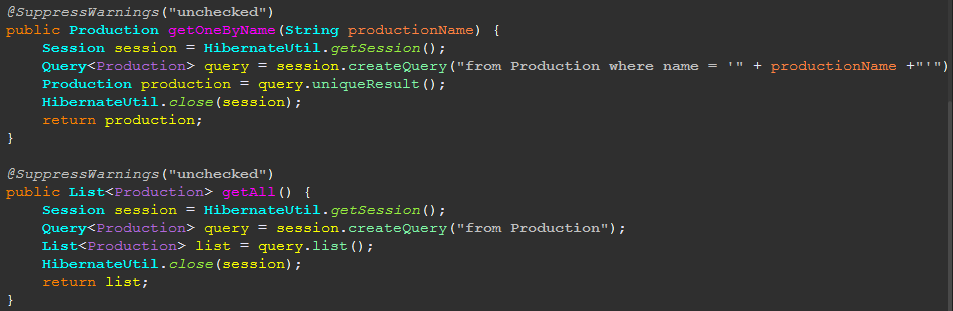


ProductionDao用于实现对商品表（production）的增删改查操作。

ProductionDao定义了四个方法：Boolean save()，Production getOneById()，Production getOneByName（），List<Production> getAll()。

save（）用于将传递过来的Production对象保存到数据库中。getOneById方法用于通过id查询一个Production对象。getOneByName方法用于通过商品名（name）查询一个Production对象。List<Production> getAll()方法用于查询数据库中存放的所有Production对象并打包到一个List集合中返回。具体实现如下：

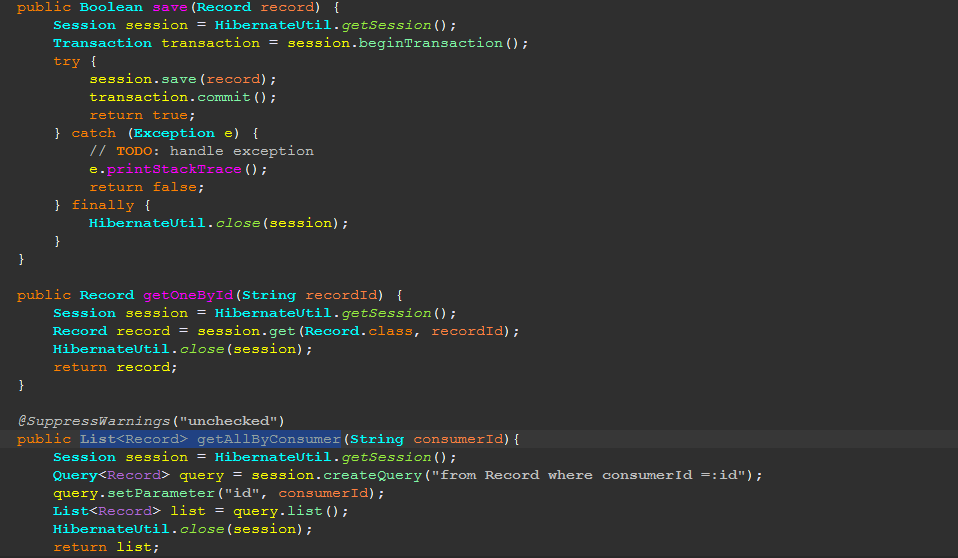




RecordDao用于实现对记录表（record）的增删改查操作。

RecordDao一共实现了三个方法：Boolean save()，Production getOneById()，List<Record> getAllByConsumer（）。

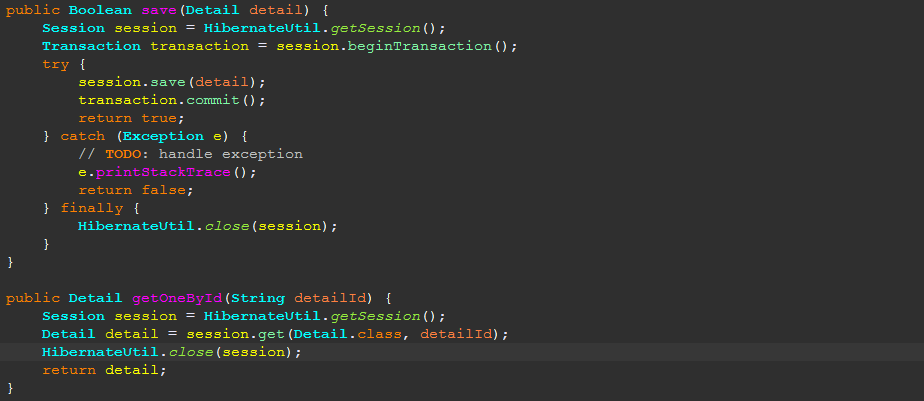
save方法用于将传递过来的Record对象保存到数据库中。getOneById方法用于通过id查询一个Record对象。getAllByConsumer方法用于通过某一顾客id检索此顾客的所有消费记录。具体实现如下：

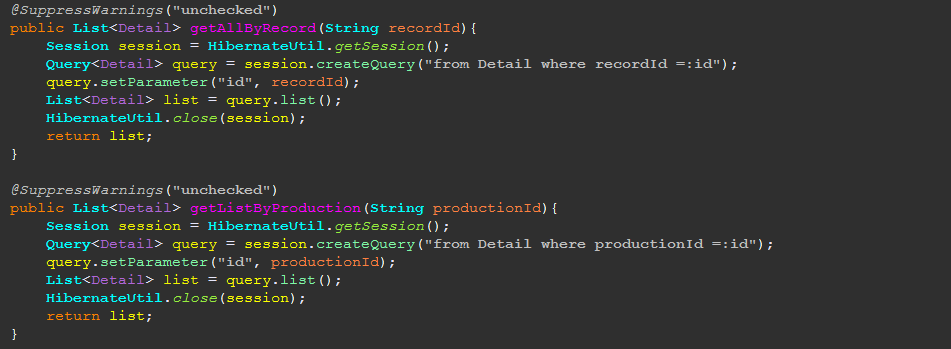


DetailDao用于实现对记录详情表（detail）的增删改查操作。

DetailDao一共实现了四个方法：Boolean save()，Detail getOneById()，List<Detail> getAllByRecord（）, List<Detail> getListByProduction（）。

save方法用于将传递过来的Detail对象保存到数据库中。getOneById方法用于通过id查询一个Detail对象。getAllByRecord方法用于通过一项消费记录查询到此消费记录下的所有记录详情。getListByProduction方法用于查询到某一商品的消费详情记录。具体实现如下：

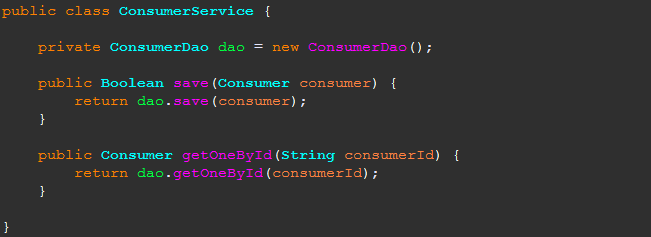




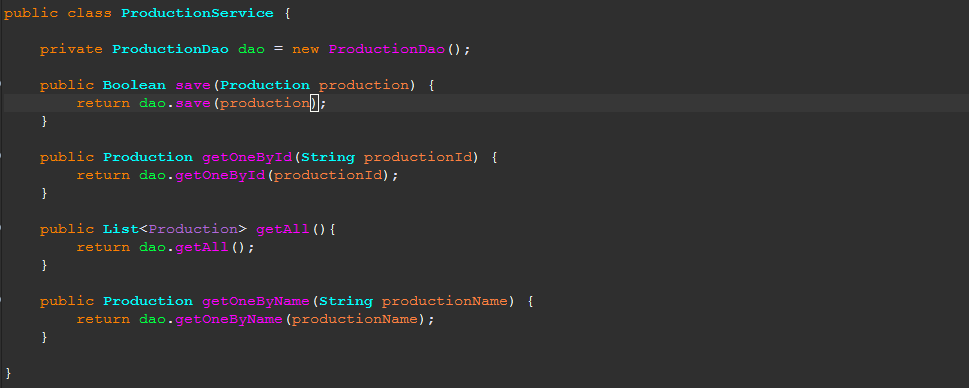
# 第四章 服务层设计

服务层共有五个服务类，分别是ConsumerService, ProductionService, DetailService, RecordService和MainService。其中MailService用于实现算法的核心内容，因此此服务类的内容放到第五章。

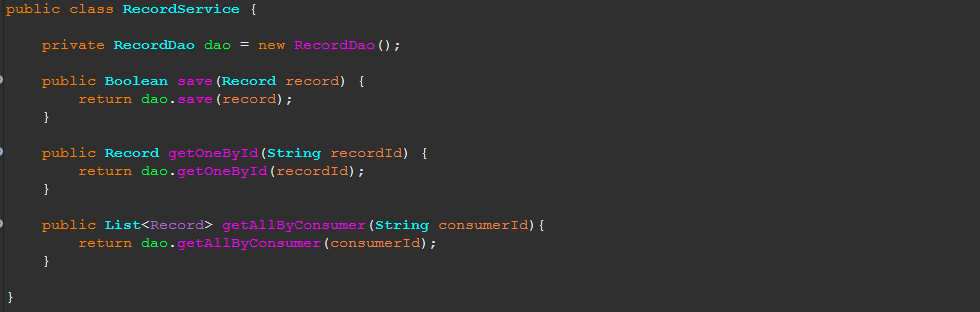
ConsumerService类用于实现顾客相关的服务，主要调用了ConsumerDao的方法。实现如下：



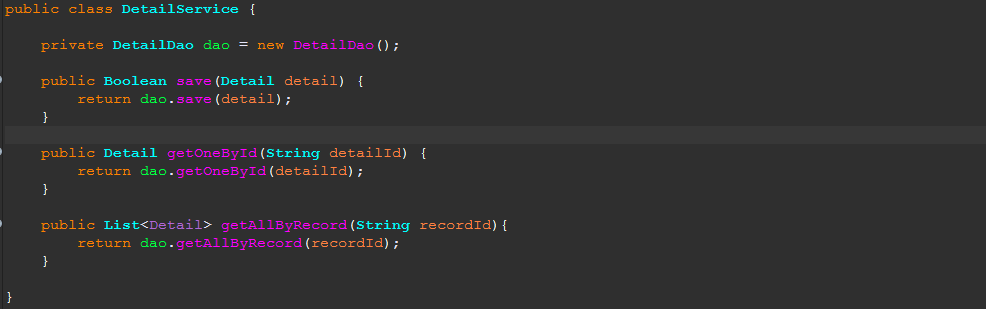
PronductionService类用于实现顾客相关的服务，主要调用了PronductionDao的方法。实现如下：



RecordService类用于实现顾客消费记录相关的服务，主要调用了ServiceDao的方法。实现如下：



DetailService类用于实现顾客消费记录详情相关的服务，主要调用了DetailDao的方法。实现如下：



# 第五章 主函数设计

主函数主要提供了项目启动时服务入口和简单交互界面的功能。用户启动项目时，首先会打印出目前数据库中存在的所有的商品，并以列表的形式展现。



然后用户需要选择一个想要购买的商品。直接输入商品名即可。







用户输入不同的商品名时，会得到不同的推荐商品列表。

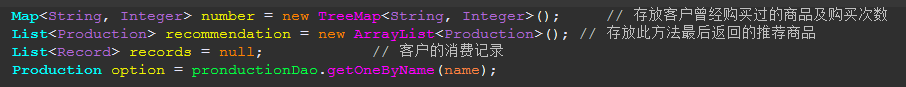
# 第六章 核心实现代码

此项目的核心实现是在服务层的MainServic类中。

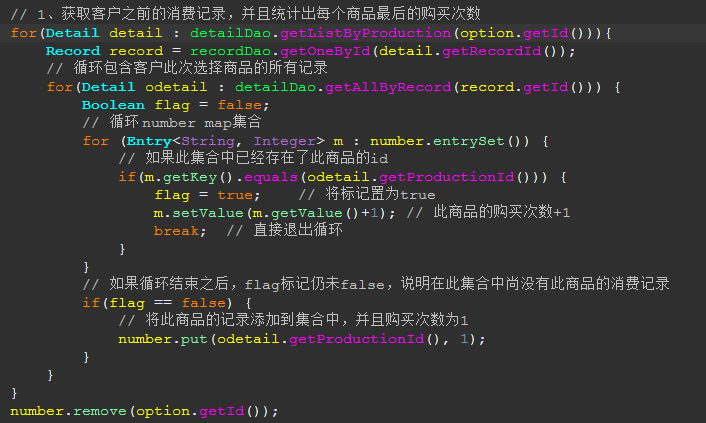
算法的实现逻辑如下：

1. 首先根据用户选择的商品名消费记录详情表（detail），得到所有此商品的消费记录详情。
2. 通过消费记录详情的recordId字段，检索消费记录表（record），得到所有包含此商品的消费记录。
3. 通过得到的所有包含此商品的消费记录，再检索消费记录详情表（detail），得到所有消费记录所包含的所有消费记录详情信息。也即，此时得到的所有消费记录详情是该顾客曾经与所选商品一起购买过的商品。
4. 遍历最后检索的消费记录详情集合，统计出每一个商品购买过几次，并按照次数由高到低打包到集合里。返回该商品列表。

代码实现如下：



上图中的number集合用于存放顾客曾经购买过的商品及购买次数；recommendation集合用于存放此方法最终返回的推荐商品。records则是用于存放客户的消费记录；option对象是用户此次选择的商品，其通过商品名检索得到。



上图中所展示的是算法实现的第一个模块，获取客户之前的消费记录，并且统计出每个商品最后的购买次数。

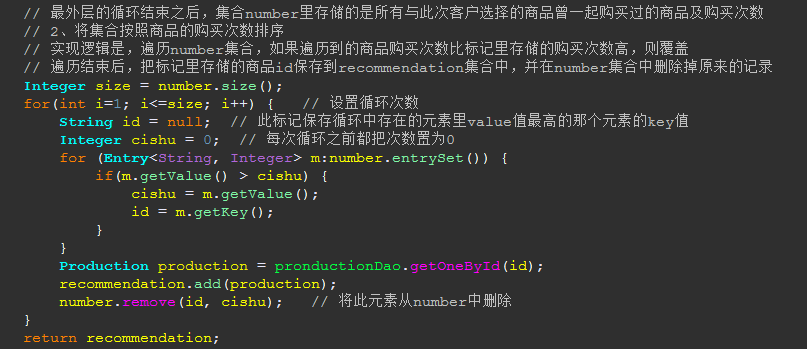
第一层for循环是为了遍历通过用户选择的商品检索出来的所有此商品消费记录详情。record对象是指通过当前循环到的detail记录检索到包含了此详情的消费记录。

第二层for循环是遍历通过record对象检索到的所有包含在此消费记录下的记录详情。

第三层for循环是遍历number对象，如果此对象中已经存在了record所属的某个详情商品，则直接在此基础上设置此元素的value值+1；如果不在集合中，则放入此集合，并设置value值为1。Flag作为是否已经存在在集合中的标记，如果存在，则设置为true，如果不存在，则始终未false。

当所有循环结束之后，number集合中已经保存了所有该用户曾经与所选商品一起购买的商品。并且value值即为此商品的购买次数。

注意此时集合中还包含了顾客选择的商品本身，因此应该调用remove方法删除此元素。



上图中所展示的代码是算法实现的第二个模块，将number集合中的元素按照value值，也就是相应商品的购买次数排序。

排序算法有很多，这里采用了最简单易懂的一种，即循环遍历number集合，找出每次循环中value值最大的元素，将其key值查询到的商品对象保存到List集合中。每次保存结束，都在number集合中删掉已经保存的那个。

当这一部分结束之后，集合recommendation中的元素是按照购买次数排好序的推荐商品，返回即可。

# 第七章 结语

这只是一个很简单的基于关联规则的商品推荐算法，对照的数据主要是顾客早已存在的购买消费记录，也因此这一类的商品推荐算法有一个很大的缺点，即新注册尚没有购买记录的顾客则无法运用此推荐算法，这种情况下可以使用基于协同过滤的推荐算法（Recommendation algorithm based on collaborative filtering），基于用户的协同过滤算法: 基于一个这样的假设“跟你喜好相似的人喜欢的东西你也很有可能喜欢。”所以基于用户的协同过滤主要的任务就是找出用户的最近邻居，从而根据最近邻居的喜好做出未知项的评分预测。