

**2022年春季学期  
计算学部《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | - |
| 学号 | - |
| 班号 | - |
| 电子邮件 | - |
| 手机号码 | - |

目录

[1 实验目标概述 1](#_Toc102826659)

[2 实验环境配置 1](#_Toc102826660)

[3 实验过程 1](#_Toc102826661)

[3.1 待开发的三个应用场景 1](#_Toc102826662)

[3.2 ADT识别与设计 3](#_Toc102826663)

[3.2.1 任务1：投票类型VoteType 3](#_Toc102826664)

[3.2.2 任务2：投票项VoteItem<C> 6](#_Toc102826665)

[3.2.3 任务3：选票Vote 7](#_Toc102826666)

[3.2.4 任务4：投票活动Poll<C>的测试 9](#_Toc102826667)

[3.2.5 任务5：投票活动Poll<C>的实现类GeneralPollImpl 10](#_Toc102826668)

[3.2.6 任务6：投票活动Poll<C>的子类型 13](#_Toc102826669)

[3.3 ADT行为的设计与实现 17](#_Toc102826670)

[3.3.1 任务7：合法性检查 17](#_Toc102826671)

[3.3.2 任务8：采用Strategy设计模式实现灵活的计票规则 17](#_Toc102826672)

[3.3.3 任务9：采用Strategy设计模式实现灵活的遴选规则 19](#_Toc102826673)

[3.3.4 任务10：处理匿名和实名投票 21](#_Toc102826674)

[3.3.5 任务11：采用Visitor设计模式实现功能扩展 22](#_Toc102826675)

[3.3.6 任务12：基于语法的数据读入 22](#_Toc102826676)

[3.4 任务13：应用设计与开发 23](#_Toc102826677)

[3.4.1 商业表决系统 23](#_Toc102826678)

[3.4.2 代表选举系统 25](#_Toc102826679)

[3.4.3 聚餐点菜系统 26](#_Toc102826680)

[3.5 任务14：应对面临的新变化 27](#_Toc102826681)

[3.5.1 商业表决应用：可以一次表决多个商业提案 27](#_Toc102826682)

[3.5.2 代表选举应用：遴选规则变化 27](#_Toc102826683)

[3.5.3 聚餐点菜应用：取消权重设置、只计算“喜欢”的票数 28](#_Toc102826684)

[3.6 Git仓库结构 29](#_Toc102826685)

[4 实验进度记录 29](#_Toc102826686)

[5 实验过程中遇到的困难与解决途径 29](#_Toc102826687)

[6 实验过程中收获的经验、教训、感想 30](#_Toc102826688)

[6.1 实验过程中收获的经验和教训（必答） 30](#_Toc102826689)

[6.2 针对以下方面的感受（必答） 30](#_Toc102826690)

# 实验目标概述

本次实验覆盖课程第 2、3 章的内容，目标是编写具有可复用性和可维护性

的软件，主要使用以下软件构造技术：

l 子类型、泛型、多态、重写、重载

l 继承、委派、CRP

l 语法驱动的编程、正则表达式

l 设计模式

本次实验给定了多个具体应用，学生不是直接针对每个应用分别编程实现，

而是通过 ADT 和泛型等抽象技术，开发一套可复用的 ADT 及其实现，充分考虑

这些应用之间的相似性和差异性，使 ADT 有更大程度的复用（可复用性）和更

容易面向各种变化（可维护性）。

# 实验环境配置

简要陈述你配置本次实验所需环境的过程，必要时可以给出屏幕截图。

特别是要记录配置过程中遇到的问题和困难，以及如何解决的。

在这里给出你的GitHub Lab3仓库的URL地址（HIT-Lab3-学号）。

https://github.com/ComputerScienceHIT/HIT-Lab3-120L031815

# 实验过程

请仔细对照实验手册，针对每一项任务，在下面各节中记录你的实验过程、阐述你的设计思路和问题求解思路，可辅之以示意图或关键源代码加以说明（但千万不要把你的源代码全部粘贴过来！）。

## 待开发的三个应用场景

**3.1.1 三个应用场景**

①商业表决

面向某个商业公司，其内部成员提出某个商业提案（例如“关于投资 xx 项目的提案”），公司董事会的各位董事对其进行实名表决（支持、反对、弃权），各董事在表决中的权重取决于其所持有的公司股票的比例，根据该持股比例对投票结果进行计算，若“支持”票的比例超过 2/3，则该提案通过，否则该提案不通过。

②代表选举

针对某次活动（例如哈工大学生代表大会），需要从一群候选人中选出若干人，作为代表参加活动。在该选举中，提前确定一部分候选人，投票人从已确定的候选人中选取，不可提名新的候选人。计划选出的代表数量,是提前确定的。投票人针对每个候选人匿名选择“支持、反对、弃权”之一，但选择“支持”的人数不能高于计划选出的代表数量,，否则为非法票。 所有投票人的权重均相等。全体投票人投票之后，首先判定各张选票的合法性，在去除非法选票之后，针对所有候选人，根据其所得到的支持票数量排序，前k个候选人当选；若有多个候选人的支持票数量相等而无法自然排出前k名，则仅 有那些明确可进入前k名的人当选——这意味着最终选出的人数可能小于k。

③聚餐点菜

一群人去餐馆就餐，需要从该餐馆提供的菜单中选择若干道菜，点菜的数量要大于等于就餐总人数，且小于总人数+5。每个人针对菜单上的每一道菜实名表达自己的喜好（喜欢、不喜欢、无所谓），选择这三个选项的数目无限制。不同人的身份不同，其偏好的影响力会有所不同（例如家庭聚餐时，老人的权重更大、子女的权重更小，见下表黄色部分）。所有人表达观点之后，根据影响力加权计票（喜欢、不喜欢、无所谓分别得分 2、0、1），取总得分最高的前k道菜。若因为有多道菜得分相等而无法自然排出前k名，则除了那些明确可进入前k名的菜之外，在其他得分相等的菜中随机选取一部分，凑足k个菜。

**3.2.2 三个应用场景异同分析**

①场景共性分析

共性1：均为投票活动。商业表决场景实质是多名股东对某个商业提案进行投票，决定是否通过执行；代表选举场景实质是选举人对被选举人的投票；聚餐点菜活动实质是多名顾客对菜品的投票活动，决定选择哪些菜品。

共性2：候选对象为具体的实物。商业表决的候选对象为某个商业提案；代表选举的候选对象为某个候选人；聚餐点菜活动的候选对象为菜品。

共性3：投票项为一个投票人对一个候选对象的具体投票信息。商业表决活动中股东对商业提案支持或者反对；代表选举活动选举人对被选举人支持，反对或者弃权；聚餐点菜活动投票人对菜品投喜欢，不喜欢，无所谓。

②场景差异分析

差异1：候选对象数量的差异。商业表决，代表选举，聚餐点菜三个活动具有不同的候选对象数量。

差异2：拟遴选出的候选对象数量差异。三个场景在拟遴选的候选对象上也有差异，商业表决是是否通过某一个提案，代表选举和聚餐点菜则是选出几位代表，选出几道菜品。

差异3：实名与否。三个应用场景在投票实名与否上也具有差异。

差异4：投票人权重。某个投票人的投票结果在最终计票时所占的比重。可以是所有投票人有相同的权重，也可以是不同投票人有不同的权重。

差异5：检查合法性。三个应用场景在投票合法性的标准上也具有差异。

## ADT识别与设计

### 任务1：投票类型VoteType

|  |  |
| --- | --- |
| public VoteType(int type){ } | 根据输入的type，提供已有的三种场景的投票种类。1:商业决策场景 2:代表选举场景3.聚会点餐场景 |
| public VoteType(Map<String,Integer> types) {} | 基于输入的Map表生成投票种类 |
| public VoteType(String regex) {} | 根据满足特定语法规则的字符串，创建一个投票类型对象 |
| Public Boolean checkLegality(String option) {} | 判断一个投票选项是否合法（用于Poll中对选票的合法性检查） |
| public int getScoreByOption(String option) {} | 根据一个投票选项，得到其对应的分数 |
| hashcode/equals | 基础辅助方法 |

表1 VoteType方法列表

该类的RI，AF，Safety from Rep Exposure设计：

RI：options中元素个数大于等于2，即至少应该提供两个投票选项

AF：options中的键代表投票人可选的投票选项，相应的值为该选项所代表的的分数。

安全设计：ADT中的属性都用private修饰，并且在构造方法中采用防御式拷贝以防用户得到属性的引用，在其他成员方法中也没有将属性的引用直接传递给用户

①VoteTypeTest的设计

基于VoteType的设计规约，使用如下的测试策略：

1. 对checkLegality的输入划分：option包含的投票选项，option不包含的投票选项。
2. 对getScoreByOption的输入划分：三种投票选项及不合法的选项。
3. 对equals的输入划分：相等和不相等两种情况
4. 对hashcode的输入划分：两个相同对象



②VoteType的设计

1. 构造函数VoteType的设计

功能：根据输入的类型序号，完成对应类型的投票选项设置

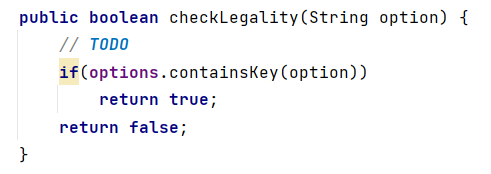
实现：序号0，1，2分别对应商业决策类型，代表选举类型，聚会点餐类型，根据输入的序号，向map表put对应的内容即可。



1. checkLegality设计

功能：检查投票选项是否合法

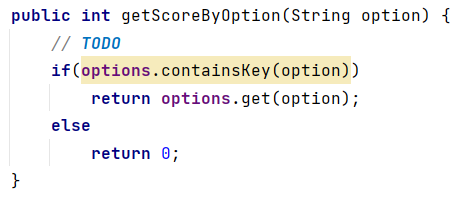
实现：检查options表中是否含有这个选项，若含有，返回ture，否则返回false



1. getScoreByOption设计

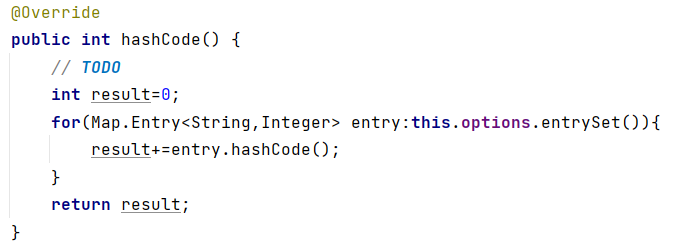
功能：根据投票选项返回其对应的分值

实现：首先判断投票选项是否合法，若合法使用map的get即可



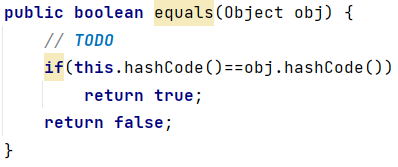
1. hashcode设计

实现：



1. equals设计

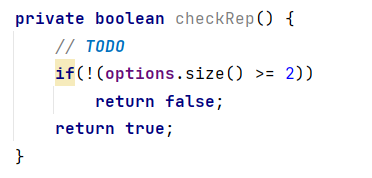
实现：根据hashcode值判断是否相等



1. checkrep设计

功能：基于RI检测rep

实现：RI设计为投票选项至少应该包含两个选项



### 任务2：投票项VoteItem<C>

|  |  |
| --- | --- |
| public VoteItem(C candidate, String value) {} | 根据输入候选人，投票内容创建一个投票项对象 |
| public String getVoteValue() {} | 得到该投票选项的具体投票项 |
| public C getCandidate() {} | 得到该投票选项所对应的候选人 |
| hashcode/equals | 基础辅助方法 |

表2 VoteItem方法列表

该类的RI，AF，Safety from Rep Exposure设计：

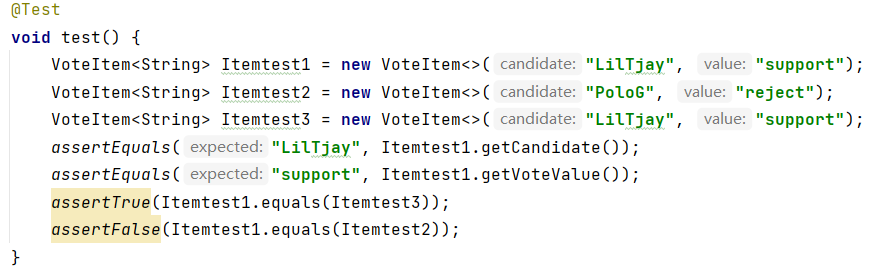
RI：value不能用空格分开

AF：candidate代表对应的候选人，value代表投票人对于该候选人的选票选项

安全设计： ADT的所有属性都是private类型，并且构造方法是没有将可变类型传递给新对象的属性，并且成员方法也没有将可变类型的引用暴露给用户

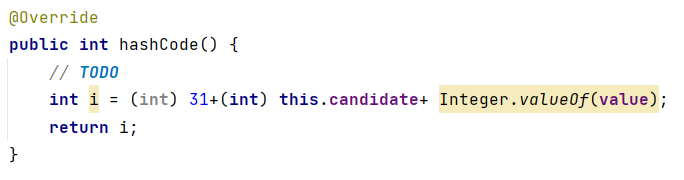
①VoteItemTest的设计

初始化了三个VoteItem对象，测试getCandidate，getVoteValue能否正确返回，测试equals能否正确返回。

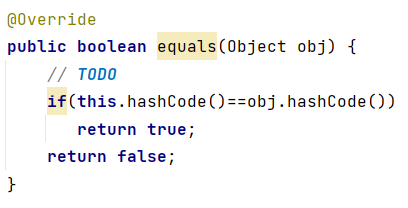


②VoteItem设计

1.hashcode的重写

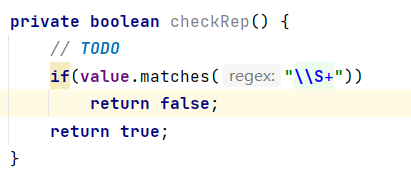


2.equals的重写



3.checkrep设计

RI的设计：value不能包含空格



### 任务3：选票Vote

|  |  |
| --- | --- |
| public Vote(Set<VoteItem<C>> voteItems){} | 根据输入的VoteItem集合创建一个选票对象 |
| public Set<VoteItem<C>> getVoteItems(){} | 查询该选票中包含的所有投票项 |
| public boolean candidateIncluded(C candidate) {} | 判断一个特定候选人是否包含本选票中 |
| hashcode/equals | 基础辅助方法 |

表3 Vote方法列表

该类的RI，AF，Safety from Rep Exposure设计：

RI：voteItems中如果有多个投票项，投票项之间候选人两两不相等，值可以相等。data非空。

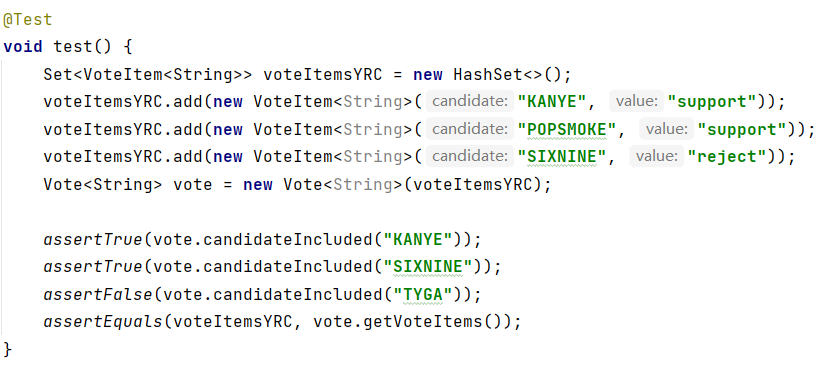
AF：voteItems表示该投票人对他投过票的候选对象的投票项，date表示该选票的日期。

安全设计：ADT中所有属性都用private修饰，在构造函数中，采用防御式拷贝避免将属性的引用暴露给客户，并且成员函数中部分采用防御式拷贝来避免表示泄露。

①VoteTest的设计

测试策略如下：

首先创建一个新的VoteItem集合，向其中添加三张选票，然后用这个集合初始化一个Vote类。测试candidateInclude，向其中输入选票中包含的候选人和不包含的候选人；测试getVoteItems，检测是否能返回正确的集合。

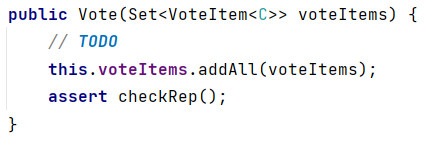


②Vote的设计

1.Vote函数的设计

功能：初始化一个Vote对象

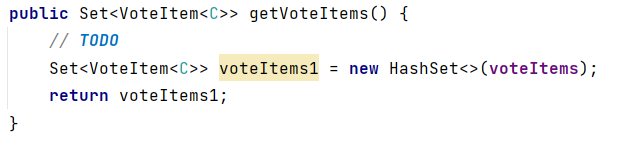
实现：将输入的set参数添加到类内的set当中即可



2. getVoteItems

功能：返回类内的VoteItem集合

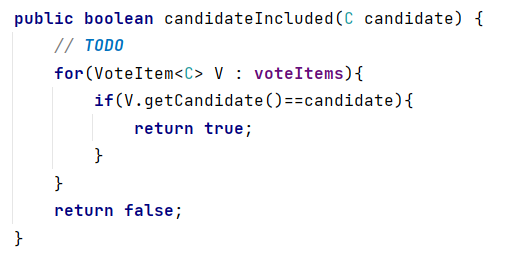
实现：在这里做一下防御性拷贝，创建一个新的对象返回回去



3. candidateIncluded

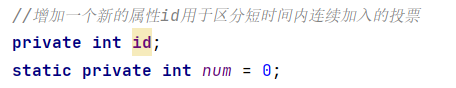
功能：判断选票中是否含有输入的候选人

实现：循环遍历一遍集合即可

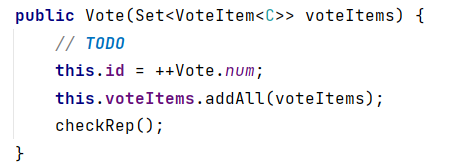


③Vote内新增的属性

由于使用类内原有的两个属性voteItems和date重写hashcode时，短时间内连续投的两个票很容易无法区分，导致缺票，因此需要新增一个属性来区分各个选票。



在初始化时，为id赋值



这样每张票都具有了其独特的id，在分入到集合时不会被看成一张票而导致缺票。

### 任务4：投票活动Poll<C>的测试

根据Poll<C>设计了如下的测试：

创建了两个投票人（权重相同）及一种投票类型，三位候选人，目标遴选出两位，主要测试两位投票人的投票是否进入了票箱即两位投票人的投票是否合法以及最终的投票结果

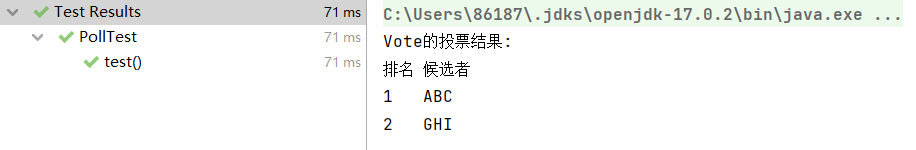
|  |  |  |  |
| --- | --- | --- | --- |
|  | ABC | DEF | GHI |
| v1 | Support | Oppose | Support |
| v2 | Oppose | Waive | Waive |

表4 投票详情



程序的运行结果如下：

可以看到测试通过并且显示了正确的结果。



### 任务5：投票活动Poll<C>的实现类GeneralPollImpl

|  |  |
| --- | --- |
| public Set<Vote<C>> getIllegalVotes(){} | 返回不合法选票 |
| public Set<Vote<C>> getVotes(){} | 返回所有选票 |
| protected boolean checkLegal(Vote<C> vote){} | 判断选票是否合法 |
| protected void checkVotes(Set<Vote<C>> votes) throws VoteMore,VoteLess{} | 检查选票数量和投票人数量是否相同并抛出相应异常 |
| public GeneralPollImpl(){} | 构造函数 |
| public void setInfo(String name, Calendar date, VoteType type, int quantity){} | 设定投票活动的基本属性 |
| public void addVoters(Map<Voter, Double> voters){} | 设定投票人及其权重 |
| public void addCandidates(List<C> candidates){} | 设定候选人 |
| public void addVote(Vote<C> vote){} | 接收一个投票人对全体候选对象的投票 |
| public void statistics(StatisticsStrategy ss) throws VoteMore,VoteLess{} | 按规则计票 |
| public void selection(SelectionStrategy ss) {} | 按规则遴选 |
| public String result(){} | 输出遴选结果 |

表5 GeneralPollImpl方法列表

该类的RI，AF，Safety from Rep Exposure设计：

RI： name:长度大于一的字符串，开头字符不能为空格; date:非空; candidates:列表元素个数大于或等于一;voters:映射元素个数大于或等于一;quantity:大于等于一并且小于等于candidates列表元素个数的整数;votes:集合元素个数小于或等于voters元素个数;其余元素非空。

AF：name:投票活动的名称;data:投票活动发起的时间，在rep中已经说明。

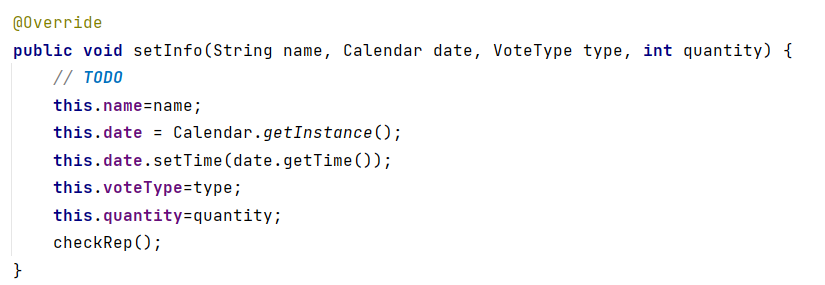
安全设计：ADT中的属性都采用private或protected修饰，并且构造函数及其他成员函数都没有将内部引用直接暴露给用户。

①GneralPollImpl的设计

1.setinfo设计

功能：设定投票活动的基本属性

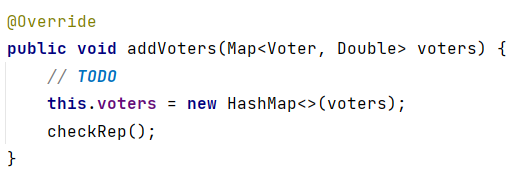
实现：将输入的参数赋值给类内属性即可，date做一下防御性拷贝。



2.addVoters的设计

功能：设定投票人及其权重

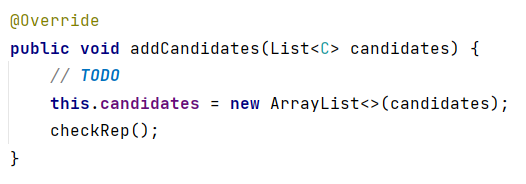
实现：防御性拷贝



3. addCandidates的实现

功能：设定候选人

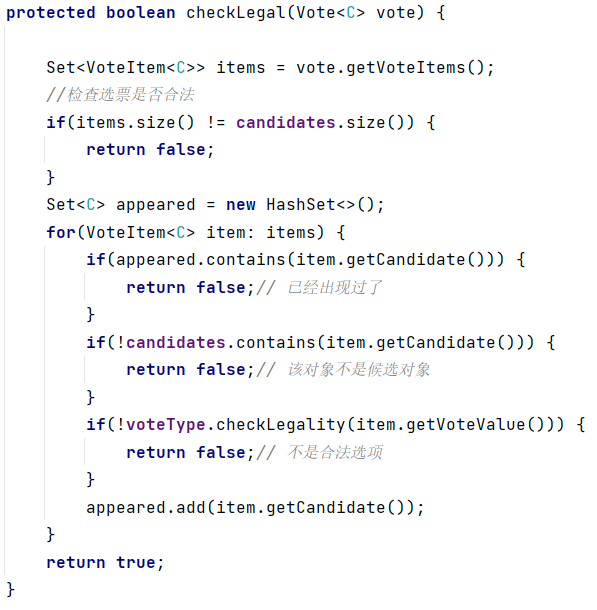
实现：防御性拷贝



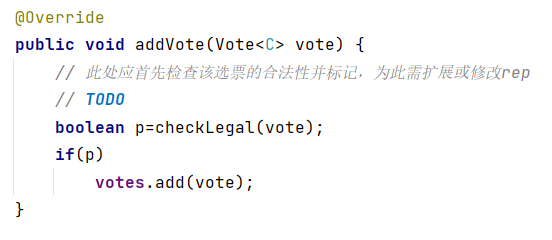
4.addVote的实现

功能：接收一个投票人对全体候选对象的投票

实现：需要先检测选票的合法性，若合法则接受。关于选票的合法性，有以下几点：①一张选票中没有包含本次投票活动中的所有候选人②一张选票中包含了不在本次投票活动中的候选人③一张选票中出现了本次投票不允许的选项值④一张选票中有对同一个候选对象的多次投票。因此可以在类内新建一个checkLegal方法来检查选票的合法性，其具体实现为：



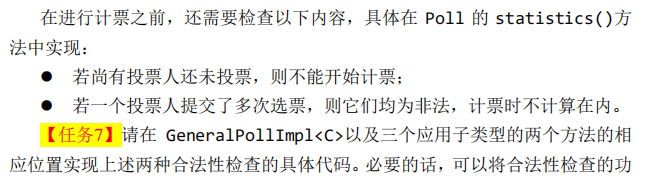
在addVote函数中，先调用checkLegal函数进行检查，若选票合法则接受投票。



5.statistics的设计

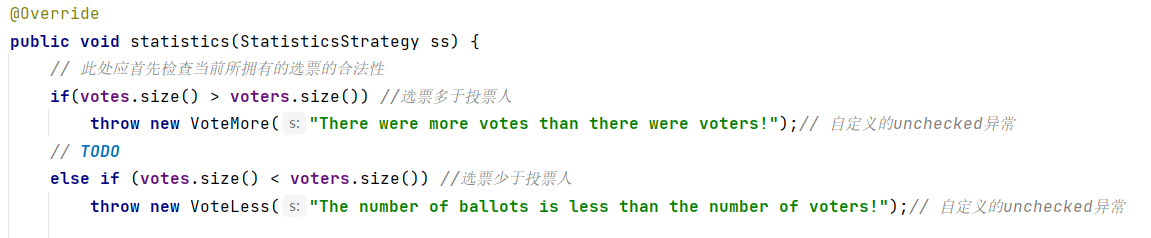
功能：按照规则计票

实现：



按照要求，还需要在计票前进行检查，具体要求见上图。

在匿名情况下，由于我们不知道投票人的信息，因此无法对上述两个要求的完整实现，只能简单的判断选票数量是否等于投票人数量，具体情况下的实现交由子类将该函数重写。

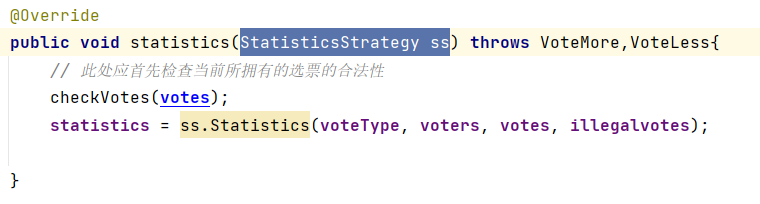


6.selection的设计

功能：按规则遴选

实现：

通过传入的StatisticsStrategy ss来进行实现，对于特定情况的遴选策略，交由子类重写。



### 任务6：投票活动Poll<C>的子类型

根据3.1对三种应用场景的分析，在继承GeneralPollImpl的基础上，需要对部分方法进行重写，以应用于不同的投票场景。

①商业决策场景

该场景需求特点：

⑴实名投票

⑵拟遴选出的对象数量：0或1(这部分在计票策略时实现)

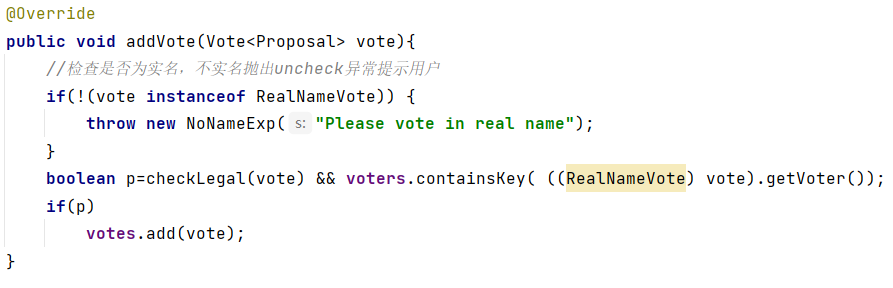
⑶投票具有权重(这部分在计票策略时实现)

⑷遴选规则：超过2/3支持票

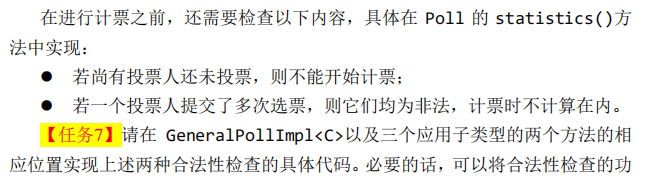
基于上述的需求特点，需要对原有的addVote方法和计票方法重写，以确保所有的选票都满足这些需求。

I addVote方法的重写

对于要添加的选票，首先判断其是否为实名，然后判断其实名是否属于投票人，最后再进行都要执行的选票有效性检查。



II 计票方法的重写

n

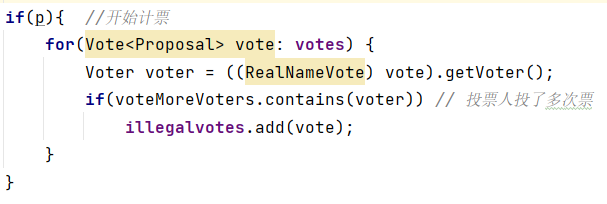
在GeneralPollImpl类中，计票方法的检查只是简单的实现了判断选票数量和投票人数量之间的检查，在这部分，需要基于实名投票，重写该判断。

1.首先判断是否所有人都投了票，若还有人尚未投票，则计票不能开始。

将投过票的合法投票人加入已投票集合，判断其大小是否与voters集合相等，判断是否所有人都已经投票，通过后开始计票



2.根据上面的原则，在计票时，投了多票的投票人，其所有投票均作废，在第一步时，我们还统计了多次投票的投票人，因此我们只需将这些人的票都放入非法投票当中。



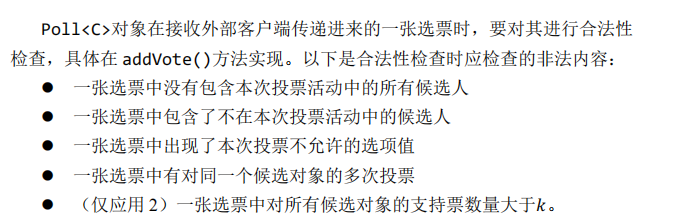
②代表选举场景

该场景需求特点：

⑴匿名投票

⑵拟遴选出k<=m名对象(这部分在计票策略时实现)

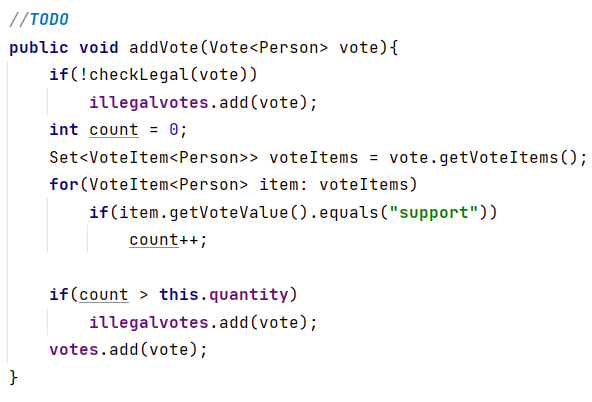
⑶投票无权重(这部分在计票策略时实现)



在addVote检查时，该场景还要求一张选票中对所有候选对象的支持票数量大于k。

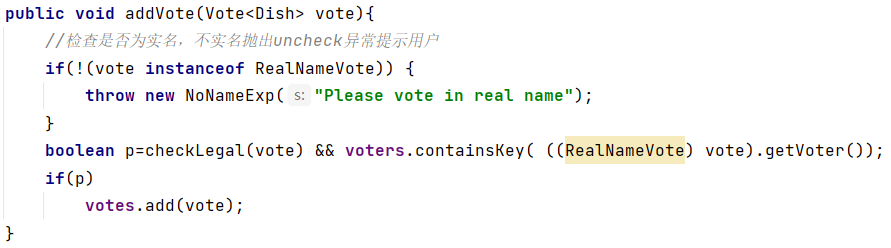
I addVote的重写

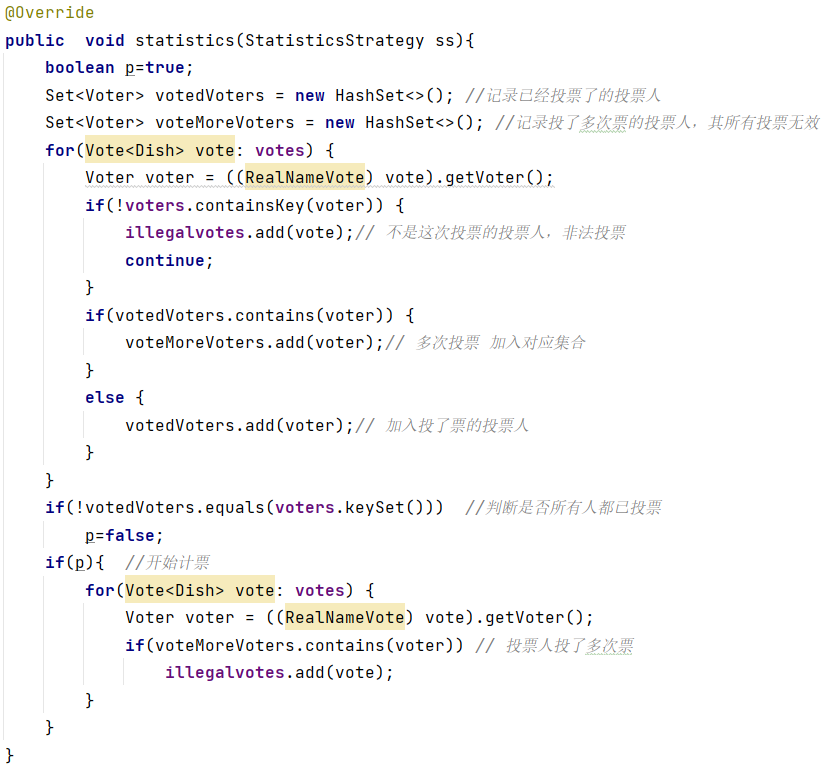
首先判断选票是否满足一般意义上的合法，然后进行该场景下的特殊合法性检测，即一张选票中对所有候选对象的支持票数量大于k



③聚餐点菜场景

这个场景的需求和企业决策大致相同，其设计代码如下：

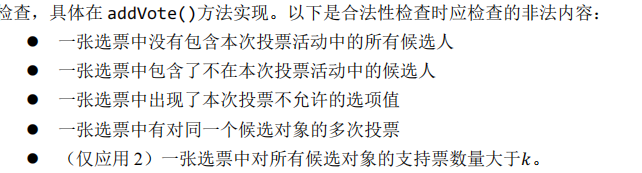




## ADT行为的设计与实现

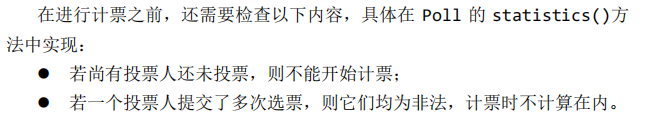
### 任务7：合法性检查

①addVote的合法性检查



这一部分在任务6 Poll<C>子类型的设计中已经有体现，在addVote对选票进行按上图要求的检查并抛出相应的异常，检查通过后再添加投票

②statistic的合法性检查



这部分同样在任务6 Poll<C>子类型的设计中已经有体现，对于匿名投票由于我们无法得知投票者的信息，因此只能简单的判断选票数量是否等于投票人数量；对于实名投票，能够严格执行上述检查。

### 任务8：采用Strategy设计模式实现灵活的计票规则

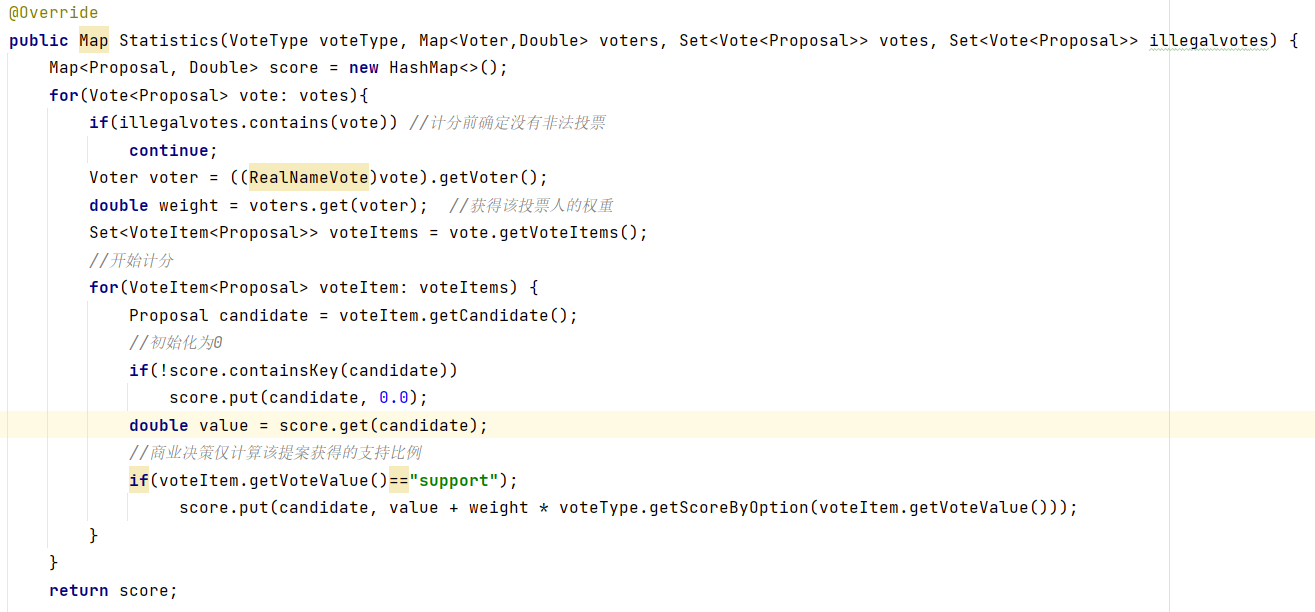
①商业决策场景

该场景的计票规则是仅根据权重，计算该提案获得支持票的比率



因此设计如下的实现方式：

首先确定没有非法票，然后依次遍历所有的投票人，每次遍历首先获得该投票人的权重，若投票为支持票，则在提案的原有支持率上加上这一票。



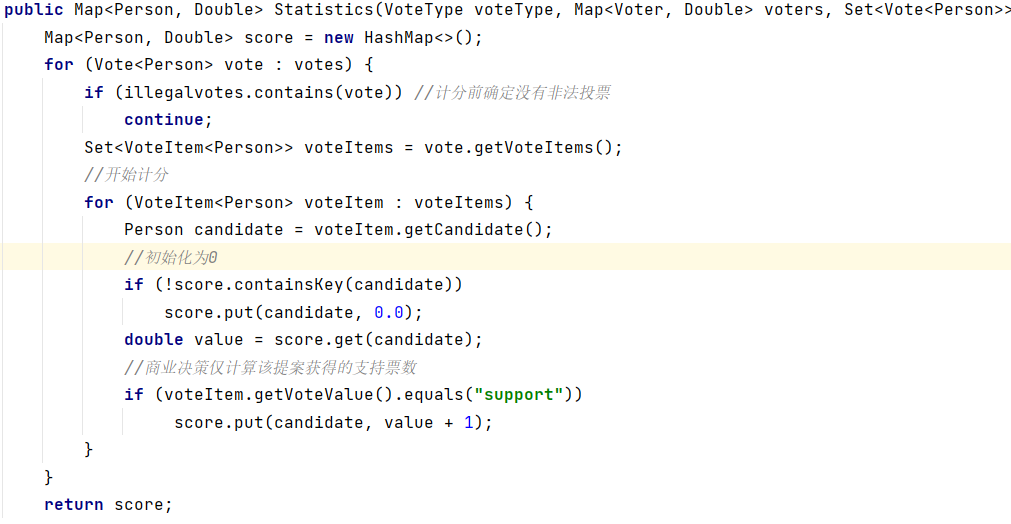
②代表选举场景

该场景的计票规则是投票无权重，计算每个候选人得到的支持票总数



因此设计如下的实现方式：

和其他场景的区别就是该场景是匿名投票，因此不需要计算各个投票人的权重



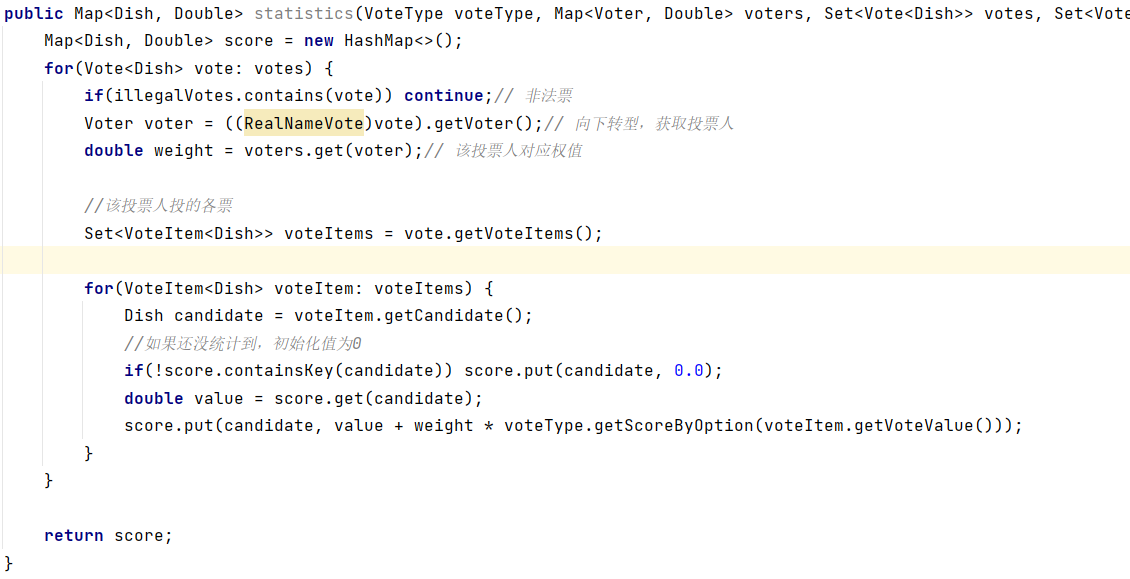
③聚会点餐场景

该场景的计票规则是仅根据权重，计算每个菜品综合得分



因此设计如下的实现方式：

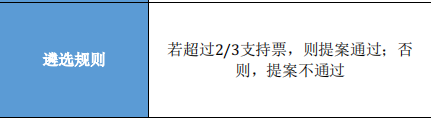
和其他两种的区别就是在分数计算的设计上，采用分数加权相加



### 任务9：采用Strategy设计模式实现灵活的遴选规则

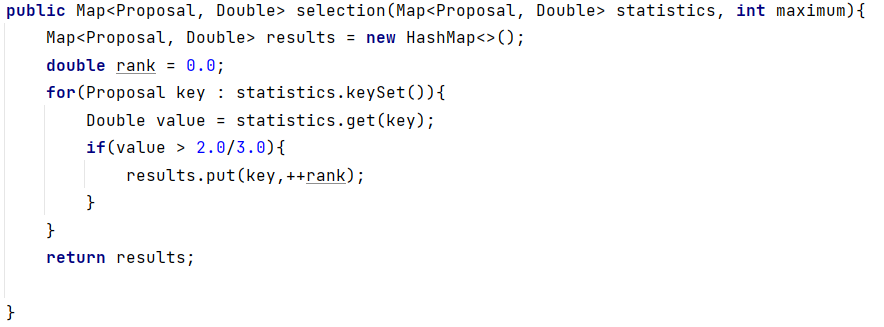
①商业决策场景

该场景的遴选规则如下：



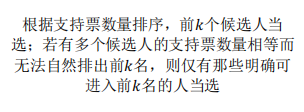
因此该场景下的遴选策略如下：

对于每个提案，提取其支持率，若超过2/3的支持率，则遴选通过



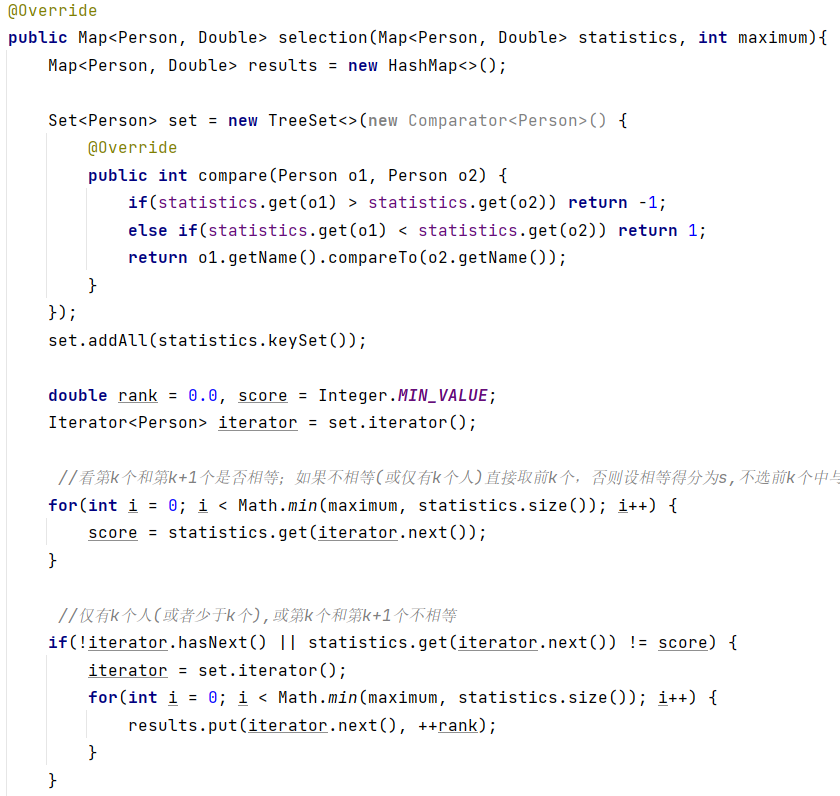
②代表选举场景

该场景的遴选规则如下：



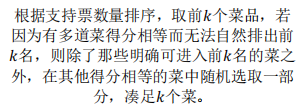
因此该场景下的遴选策略如下：

首先找到第k名的得分，然后查看第k+1名的分数，如果这两个分数不一样，则说明可以正好找出前k名；如果相等，则遴选出得分大于第k名得分的所有人。



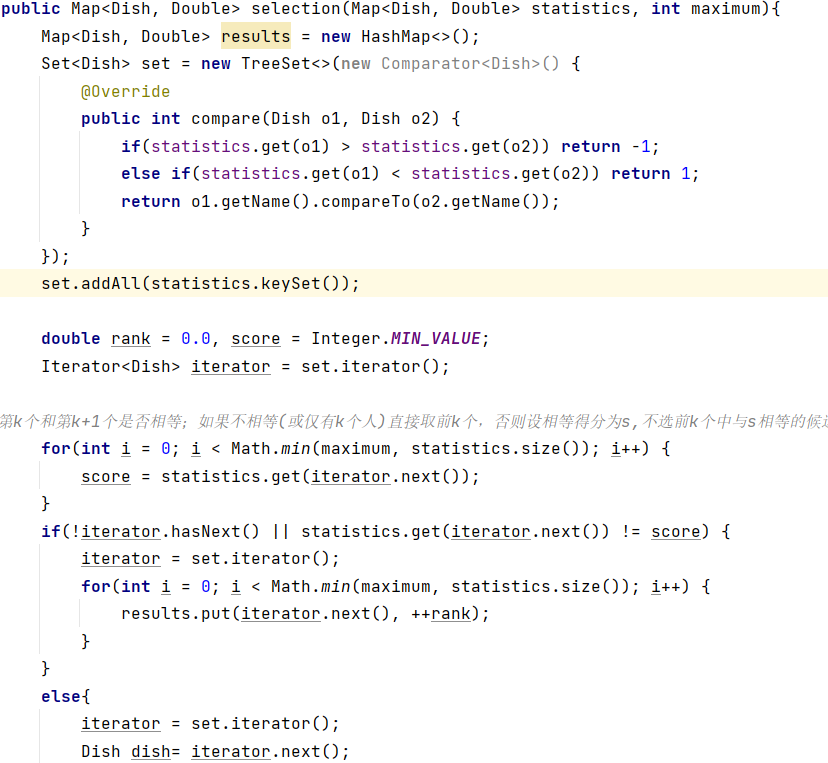
③聚会点餐场景

该场景的遴选规则如下：



因此该场景下的遴选策略如下：

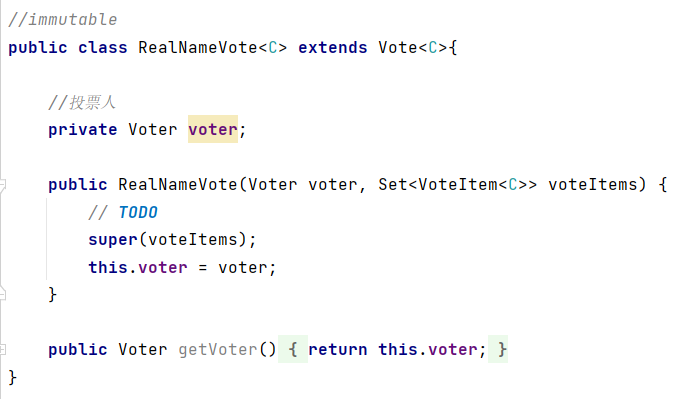
首先找到第k名的得分，然后查看第k+1名的分数，如果这两个分数不一样，则说明可以正好找出前k名；如果相等，则遴选出得分大于第k名得分的所有人，然后随机抽取与第k名得分相同的菜品以补充完整。



### 任务10：处理匿名和实名投票

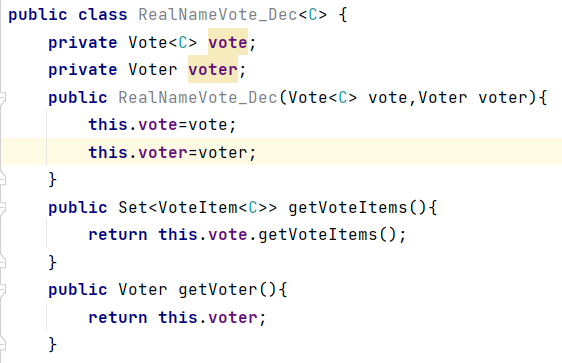
①RealNameVote设计

该类继承Vote类，在匿名投票的基础上，增加一个新的属性voter，用于存储投票人。在构造函数上，使用super输出化匿名投票信息，在此基础上初始化voter即投票人信息。



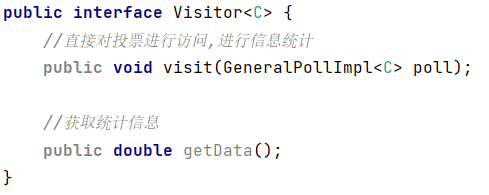
②Decorator设计模式

采用装饰器的设计模式，在RealNameVote\_Dec类中，新添加一个Vote类型的数据vote和一个Voter类型的数据voter，原先Vote类的功能全部委托给这个类内属性vote来实现，实名的功能通过属性voter得以添加。

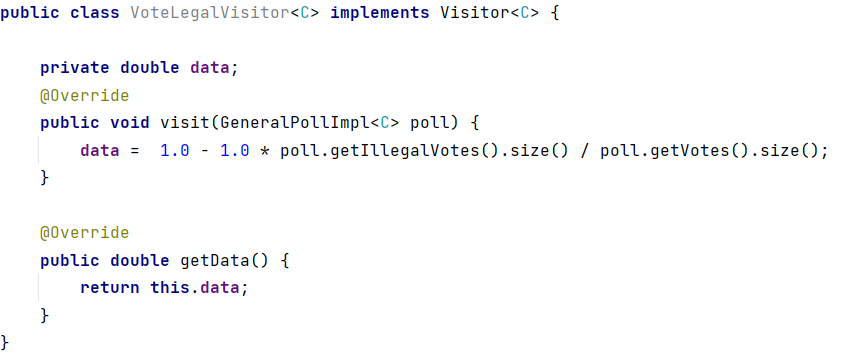


### 任务11：采用Visitor设计模式实现功能扩展

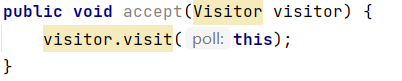
设置Vistitor接口：



由于之前在ADT内增加了属性存储非法投票，可以直接将Poll对象放到Visitor中。

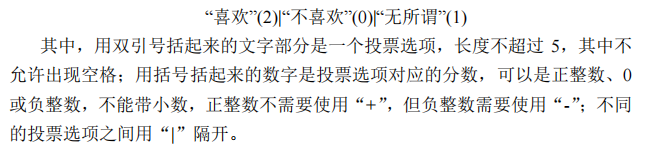


最后，在GeneralPollImpl中留一个accept方法



### 任务12：基于语法的数据读入

首先检查输入的字符串是否符合如下的输出格式：



对于输入的字符串，首先使用split将字符以|为界分割，首先判断要至少有两个选项。对于分割后的每个投票选项，使用正则表达式判断其是带权重还是不带权重的类型：





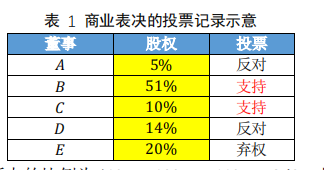
然后检查每个选项是否长度不超过5且不出现空格，都检查完毕之后，将其添加到options当中：



## 任务13：应用设计与开发

### 商业表决系统

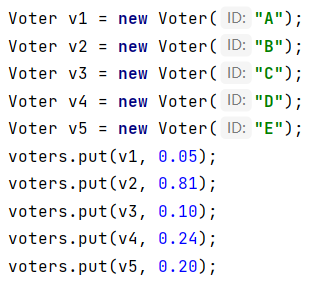
在该表决系统下，设计了一次模拟商业表决，数据直接采用了实验指导书上的例子。



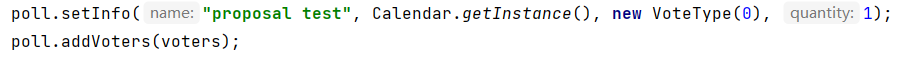
具体设计过程如下：

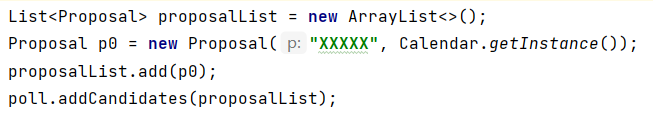
1.投票人权重设置

首先初始化5个Voter对象，然后将这些对象和其对应的权重put到voters集合当中。



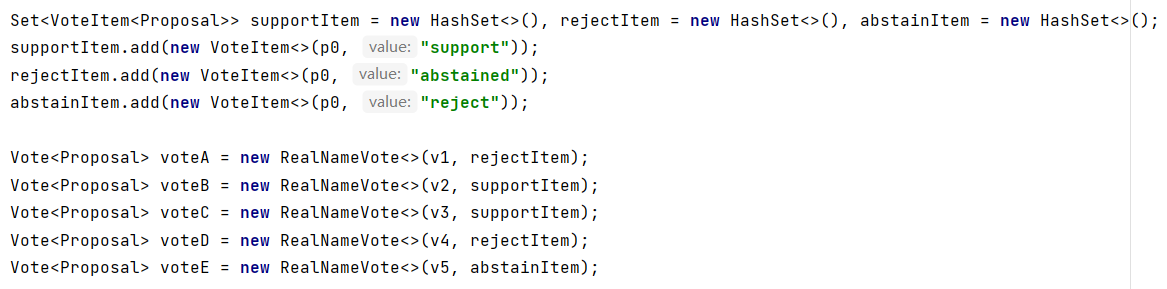
2.投票系统初始化





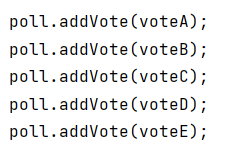
3.建立投票

由于商业表决是实名投票，因此需要使用RealNameVote类，具体投票信息均按照例子。

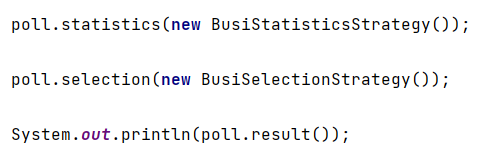


4.添加投票

将这些投票都加入到投票集合当中



5.分数统计及结果遴选

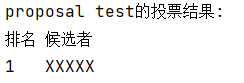


6.输出结果

符合预期结果，表示程序能够正确处理投票



将B的权重改为81%（虽然不合理），预期该提案能通过，再次运行程序

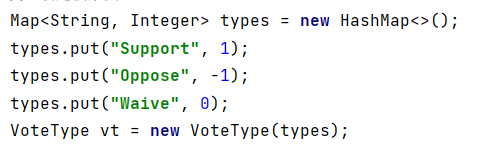


结果正确。

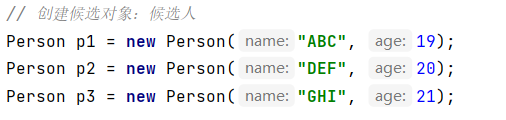
### 代表选举系统

这部分的大部分代码已经给出。

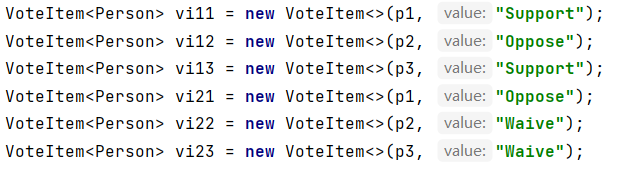
该系统有两位投票人vr1，vr2，投票类型如下：



共有三位候选人，其信息如下：

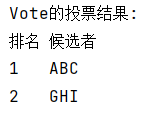


投票信息：



按照投票结果，预期选出：ABC GHI

运行程序：



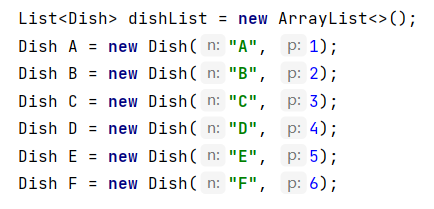
结果正确！

### 聚餐点菜系统

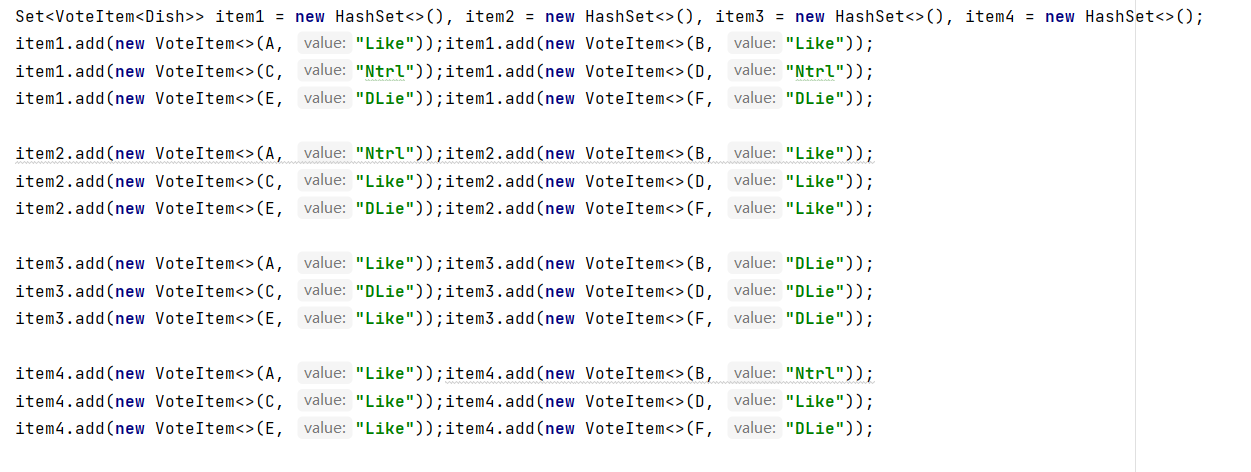
对于该系统的设计，同样数据直接采用了实验指导书上的例子：



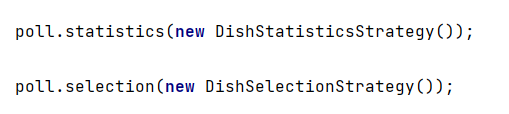
首先初始化6道菜品：



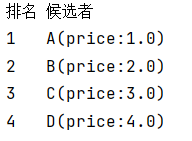
然后将建立各位成员的投票信息：



最后进行计票和遴选活动：



程序运行结果：

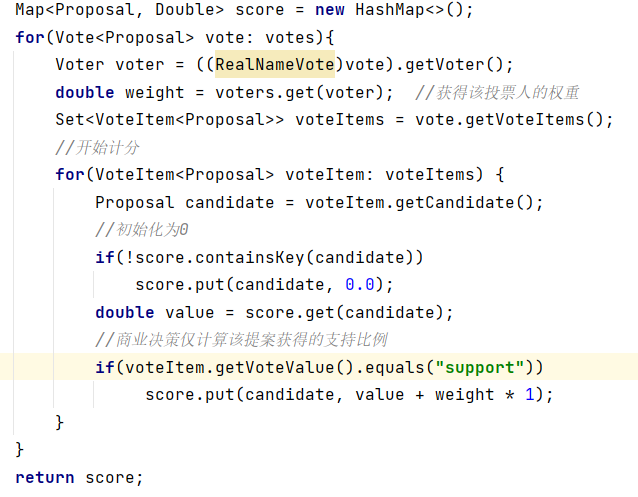


结果正确！

## 任务14：应对面临的新变化

### 商业表决应用：可以一次表决多个商业提案

新变化要求设计可以一次表决多个商业提案。实际上，在商业表决的原始设计当中，已经可以对多个候选对象进行投票，只是我们原来传入的候选提案集合中的数量一直为1，若要表决多个提案，只需要往集合当中添加元素。同时，在商业表决的计票设计当中，也是对提案的集合进行遍历：



因此无需对原有的设计再进行更改，即可完成多个提案的表决。

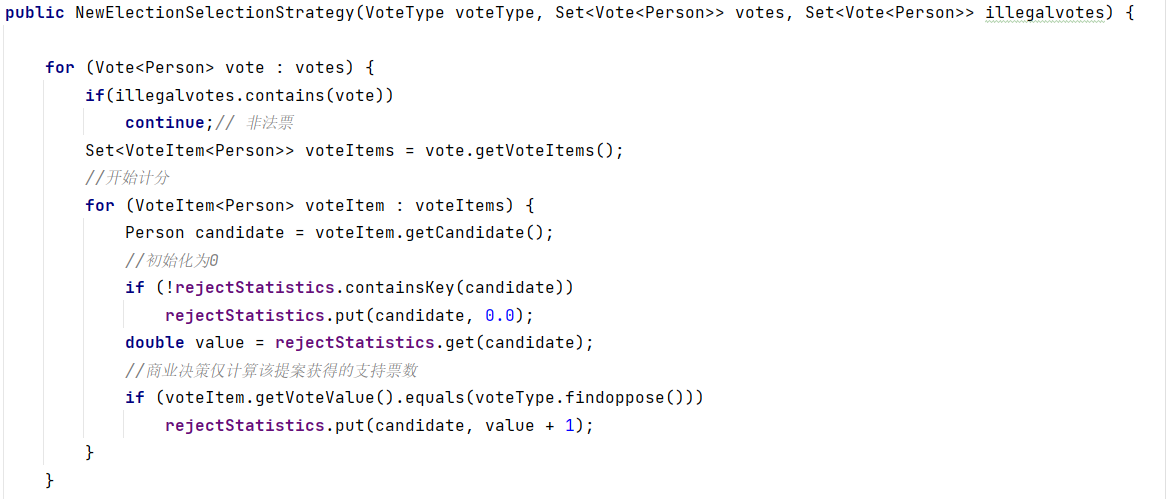
### 代表选举应用：遴选规则变化

新的遴选规则要求在相同赞同票的情况下，根据该候选人的反对票数量进行排位，若能得到前k个则遴选k个人，若仍无法明确得到前k个则只遴选出能够明确前k名的人。基于这样的遴选规则，原有的设计显然无法完成该需求，因此需要新创建一个策略NewElectionSelectionStrategy。

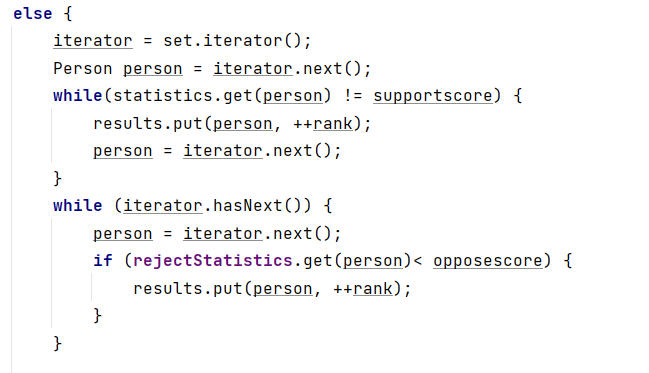
在这个策略里面，引入一个Map表用来存储各个候选人的反对票排序：



为了计算这一排序，在构造函数当中需要引入新的三个参数，以及在Votetype当中新增加一个findoppose方法用来找到反对票的名称。

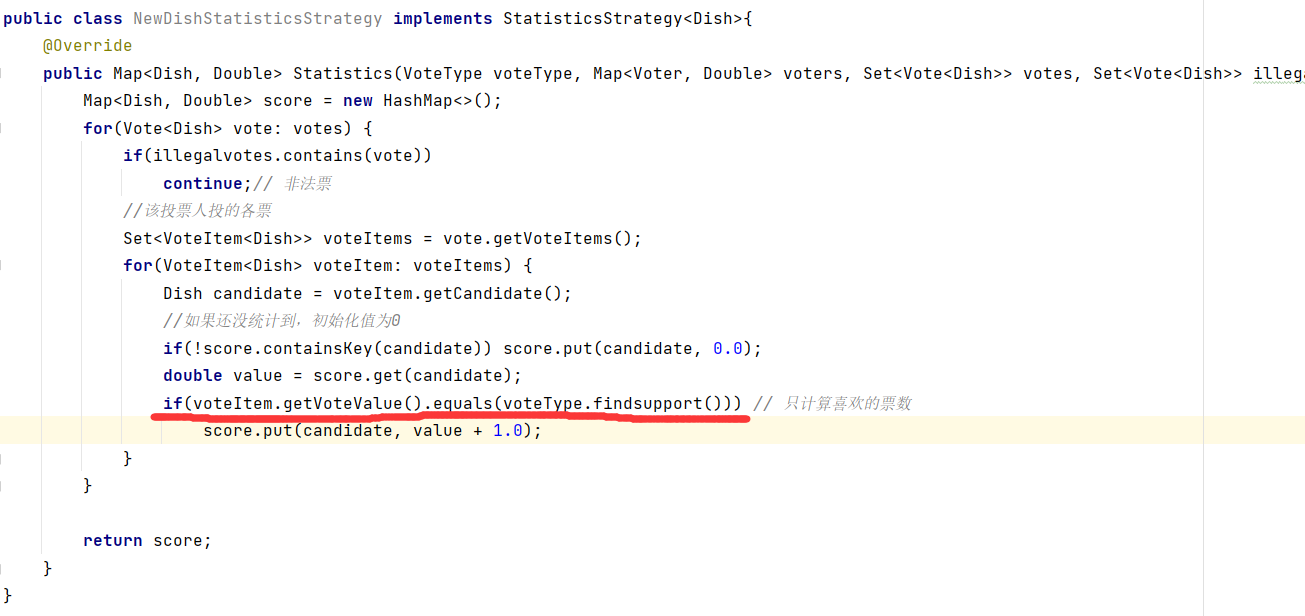


经过这样的设计，经过构造后类内得到了反对票序列Map，在遴选过程中，遵循着以下的逻辑：若第k个人和第k+1个人的赞同票数量不同，则能够明显分出前k个人，因此直接返回前k个人作为遴选结果；若第k个人和第k+1个人的赞同票数量相同，则需要一次比较同分人的反对票，直至遴选完成：



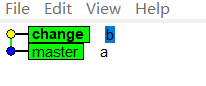
### 聚餐点菜应用：取消权重设置、只计算“喜欢”的票数

该场景的新变化只是简单的取消了权重设置，因此只需要创建一个新的计票策略，在原有的计票策略的基础上，取消计分时的加权，只统计赞成票，并不需要太多代价，只需要创建一个新的策略。



## Git仓库结构

请在完成全部实验要求之后，利用Git log指令或Git图形化客户端或GitHub上项目仓库的Insight页面，给出你的仓库到目前为止的Object Graph，尤其是区分清楚change分支和master分支所指向的位置。



# 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

每次结束编程时，请向该表格中增加一行。不要事后胡乱填写。

不要嫌烦，该表格可帮助你汇总你在每个任务上付出的时间和精力，发现自己不擅长的任务，后续有意识的弥补。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 6.21 | 13:22—22:30 | 任务1——任务7 | 顺利完成 |
| 6.22—6.23 | 13.40—22:50 | 任务7——任务13 | 顺利完成 |
| 6.24—6.25 | 16:15—23:05 | 任务12，任务14 | 顺利完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 对于这样一个较为庞大的框架，一时不知道何从下手 | 根据实验指导书，按照任务的先后顺序，理清了各个部分需要实现的功能与职责，最终顺利完成整个框架的设计 |
| 三个场景的计票和遴选策略设计较为困难 | 使用策略模式的树形结构，先设计一个默认的策略，在此基础上进行继承。就可以提高代码复用性，减少重复性。 |
| 在计票时很多票会被记成一张，经过仔细检查发现，连续的投票会导致candidate和voteitem不足以区分两张票 | 最终在Vote类中再次新增加一个属性，以实现区分短时间内连续投入的票 |

# 实验过程中收获的经验、教训、感想

## 实验过程中收获的经验和教训（必答）

在进行软件开发的设计阶段时就要考虑到软件的可维护性，安全性及可复用性，如果这些点没有做好，就会导致后续新增加或者调整软件功能时，需要修改大量的代码和结构，导致软件开发的代价过高。

## 针对以下方面的感受（必答）

1. 重新思考Lab2中的问题：面向ADT的编程和直接面向应用场景编程，你体会到二者有何差异？本实验设计的ADT在三个不同的应用场景下使用，你是否体会到复用的好处？

面向ADT的编程提高了代码的复用性，能够有效提高开发效率。面向场景编程显得应用面不如前者，思路受限制，面向ADT编程带来的复用极大提高了开发效率。

1. 重新思考Lab2中的问题：为ADT撰写复杂的specification, invariants, RI, AF，时刻注意ADT是否有rep exposure，这些工作的意义是什么？你是否愿意在以后的编程中坚持这么做？

用户无需知道ADT具体实现，而开发者可根据需要随时修改。并且保证表示独立性，防止表示泄露，使软件更安全。会坚持这么做。

1. 之前你将别人提供的ADT/API用于自己的程序开发中，本次实验你尝试着开发给别人使用的ADT/API，是否能够体会到其中的难处和乐趣？

难处：开发别人使用的ADT/API，更需要充分考虑整个结构的简单易懂性，不能只让自己看懂，其次还要满足健壮性和正确性，在未来发生变化时，要保证程序有好的可维护性。

乐趣：写代码十分快乐。

1. 你之前在使用其他软件时，应该体会过输入各种命令向系统发出指令。本次实验你开发了一个简单的解析器，使用语法和正则表达式去解析一个遵循特定规则的字符串并据此构造对象。你对语法驱动编程有何感受？

语法驱动以一种逻辑的形式来构造对象，和本学期学习的形式语言与自动机课程有很大的重合部分，其实质就是一种对本文的匹配。这大大降低了开发的代价。例如文本搜索中，这样做可以极大提高效率。

1. Lab1和Lab2的工作都不是从0开始，而是基于他人给出的设计方案和初始代码。本次实验中也提供了一部分基础代码。假如本实验要求你完全从0开始进行ADT的设计并用OOP实现，你觉得自己是否能够完全搞定？你认为“设计ADT”的难度主要体现在哪些地方？

类与类之间的关系很重要但也很复杂，在设计ADT时，如何将各个类有机联系在一起，互相补充是设计的一大难点。

1. “抽象”是计算机科学的核心概念之一，也是ADT和OOP的精髓所在。本实验的三个应用既不能完全抽象为同一个ADT，也不是完全个性化，如何利用“接口、抽象类、类”三层体系以及接口的组合、类的继承、委派、设计模式等技术完成最大程度的抽象和复用，你有什么经验教训？

只有充分理解类与类之间的关系以及设计模式，才能利用上这些抽象关系来提高开发的效率。可以使用委派，继承等手段，提高代码的复用性，减少开发成本，提高开发效率。

1. 关于本实验的工作量、难度、deadline。

工作量，难度适中，deadline合理。

1. 课程结束了，你对《软件构造》课程内容和任课教师的评价如何？

这门课程可以说是我第一次接触到实际的软件开发的流程，在这门课程中也学到了很多软件开发的知识，对我以后的工作，发展等都会有很大的帮助，任课的刘铭老师上课耐心讲解，课下积极回答学生问题，帮助学生理解知识，对我的帮助非常大。