

**2022年春季学期  
计算学部《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 李世轩 |
| 学号 | 120L022109 |
| 班号 | 2003007 |
| 电子邮件 | 1146887979@qq.com |
| 手机号码 | 15234117960 |

**目录**

[1 实验目标概述 1](#_Toc106899067)

[2 实验环境配置 1](#_Toc106899068)

[3 实验过程 1](#_Toc106899069)

[3.1 待开发的三个应用场景 2](#_Toc106899070)

[3.2 ADT识别与设计 2](#_Toc106899071)

[3.2.1 任务1：投票类型VoteType 2](#_Toc106899072)

[3.2.2 任务2：投票项VoteItem<C> 4](#_Toc106899073)

[3.2.3 任务3：选票Vote 5](#_Toc106899074)

[3.2.4 任务4：投票活动Poll<C>的测试 5](#_Toc106899075)

[3.2.5 任务5：投票活动Poll<C>的实现类GeneralPollImpl 6](#_Toc106899076)

[3.2.6 任务6：投票活动Poll<C>的子类型 10](#_Toc106899077)

[3.3 ADT行为的设计与实现 11](#_Toc106899078)

[3.3.1 任务7：合法性检查 11](#_Toc106899079)

[3.3.2 任务8：采用Strategy设计模式实现灵活的计票规则 12](#_Toc106899080)

[3.3.3 任务9：采用Strategy设计模式实现灵活的遴选规则 13](#_Toc106899081)

[3.3.4 任务10：处理匿名和实名投票 13](#_Toc106899082)

[3.3.5 任务11：采用Visitor设计模式实现功能扩展 16](#_Toc106899083)

[3.3.6 任务12：基于语法的数据读入 17](#_Toc106899084)

[3.4 任务13：应用设计与开发 18](#_Toc106899085)

[3.4.1 商业表决系统 18](#_Toc106899086)

[3.4.2 代表选举系统 19](#_Toc106899087)

[3.4.3 聚餐点菜系统 20](#_Toc106899088)

[3.5 任务14：应对面临的新变化 21](#_Toc106899089)

[3.5.1 商业表决应用：可以一次表决多个商业提案 21](#_Toc106899090)

[3.5.2 代表选举应用：遴选规则变化 22](#_Toc106899091)

[3.5.3 聚餐点菜应用：取消权重设置、只计算“喜欢”的票数 22](#_Toc106899092)

[3.6 Git仓库结构 23](#_Toc106899093)

[4 实验进度记录 23](#_Toc106899094)

[5 实验过程中遇到的困难与解决途径 24](#_Toc106899095)

[6 实验过程中收获的经验、教训、感想 24](#_Toc106899096)

[6.1 实验过程中收获的经验和教训（必答） 24](#_Toc106899097)

[6.2 针对以下方面的感受（必答） 25](#_Toc106899098)

# 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：

⚫ 子类型、泛型、多态、重写、重载

⚫ 继承、委派、CRP

⚫ 语法驱动的编程、正则表达式

⚫ 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-Lab3-120L022109.git

# 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

简要介绍三个应用。

分析三个应用场景的异同，理解需求：它们在哪些方面有共性、哪些方面有差异。

共性：

都有一定数量的候选对象（数量>=1）

都对拟遴选对象数量有一定要求（这个数量是提前确定的）

投票人数量都是提前确定的

都有属于自己的投票类型

差异

投票人类型不同

投票类型不同

是否实名

对选票的合法性判断规则不同

计票规则不同

遴选规则不同

## ADT识别与设计

该节是本实验的核心部分。

### 任务1：投票类型VoteType

对VoteType的设计



方法测试

checkLegality

测试不合法输入

测试合法输入

测试不包含的输入

getScoreByOption方法实现

测试不合法输入

测试包含的type

测试不包含的type

方法实现

CheckRep

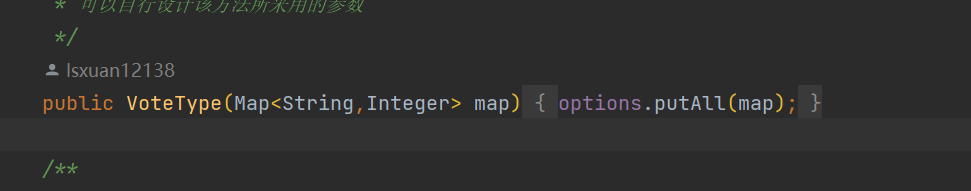
更具RI检查

检查options是否为空

检查options中的key和value是否为空

构造方法

将传入的map全部加入到options中



checkLegality(String option)

若option为null抛出异常

若options中包含option，返回true

反之返回false

getScoreByOption(String option)

若option为null抛出异常

若option不是合法的，抛出异常

否则返回options.get(option)

### 任务2：投票项VoteItem<C>

对VoteItem的设计



方法测试

getVoteValue

创建一个voteItem，

测试其getVoteValue是否与设定的相同

检查相同的情况

检查相同的情况

getCandidate

创建一个voteItem，

测试其getCandidate是否与设定的相同

检查相同的情况

检查相同的情况

方法实现

checkRep

根据RI

检查candidate是否为null

检查value是否为null

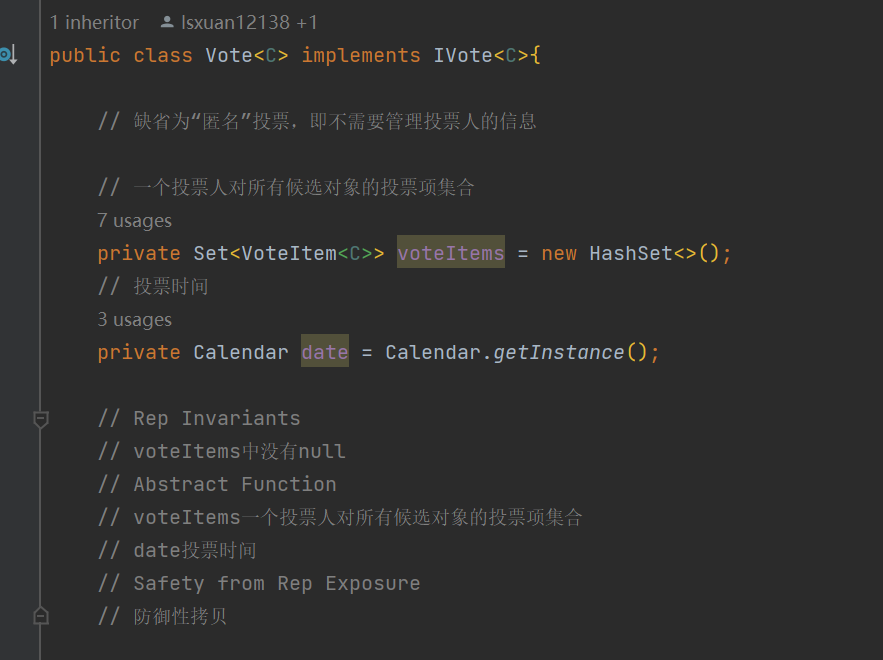
构造方法

因为rep都是不可变的类型，直接将传入的属性赋值给rep即可

getVoteValue和getCandidate同理直接返回即可

### 任务3：选票Vote

Vote设计



方法测试

getVoteItems

检测相同的输入（与设定相同的输入）

检查不同的输入

candidateIncluded

检查包含的输入

检查不包含的输入

方法实现

checRep

检查voteItems中是否包含null

构造方法

将传入的set中的元素全部加入voteItems

getVoteItems

防御性拷贝，返回voteItems的副本

candidateIncluded

若candidate为null抛出异常

检查voteItems中是否包含candidate和输入相同的元素

### 任务4：投票活动Poll<C>的测试

测试addCandidates

测试一般情况（添加的元素中无重复）

测试重复元素

测试addVoters

测试一般情况

测试重复添加同一个元素

测试setInfo

利用observer方法测试设定的值，是否与设定的相同

测试addVote

测试添加合法的选票

测试添加不合法的选票

测试statistics

测试现有投票活动的统计结果与预期是否相同

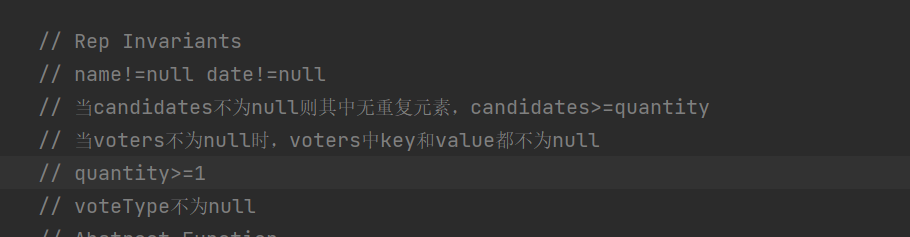
测试selection和result

测试经过遴选的结果，使用result方法输出结果，检查其与预期是否相同

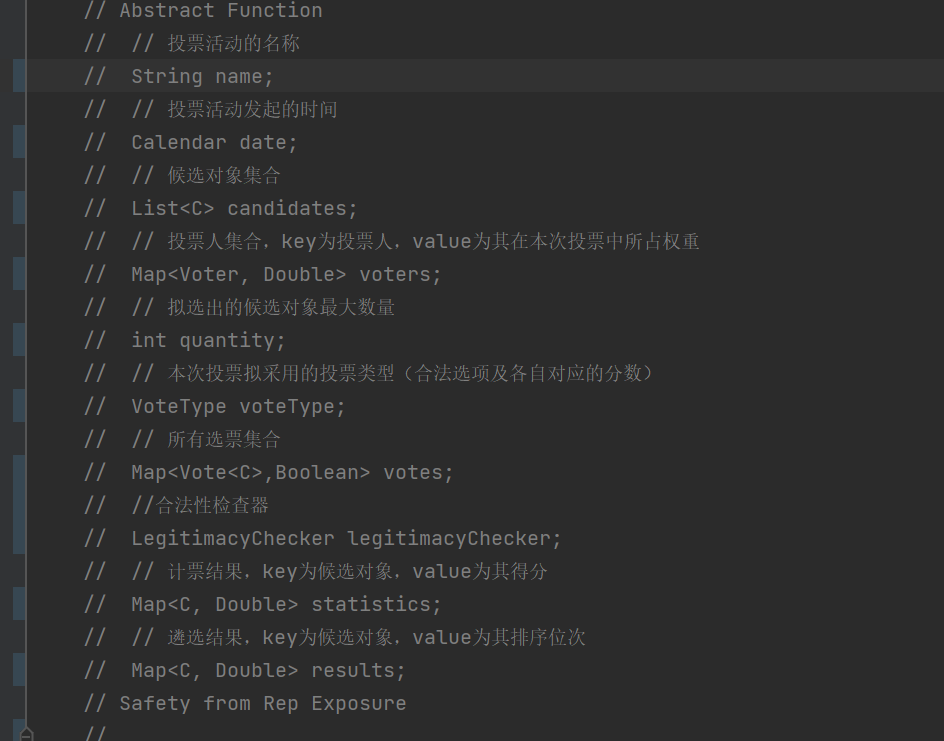
### 任务5：投票活动Poll<C>的实现类GeneralPollImpl

对GeneralPollImpl的设计

RI



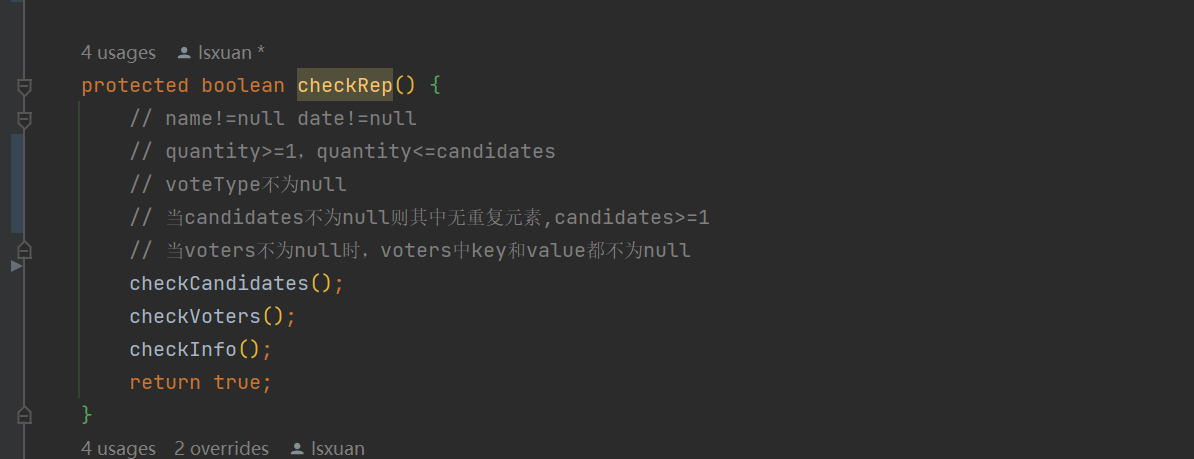
AF



CheckRep

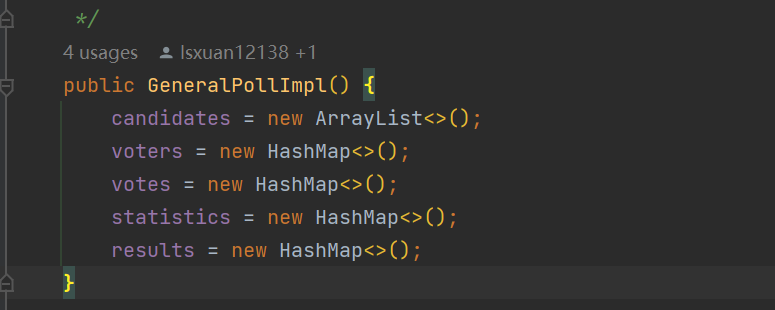
将checkRep拆分成三部分

分别根据RI检查





构造方法



public void setInfo(String name, Calendar date, VoteType type, int quantity)

首先检查输入数据合法性，不合法则抛出异常

将输入的属性不可变类型直接赋值，可变类型防御性拷贝

public void addVoters(Map<Voter, Double> voters)

将voters中的元素全部加入到this.votes中

public void addCandidates(List<C> candidates)

将candidates中元素遍历，若this.candidates中不包含则加入进去

public void addVote(IVote<C> vote)

检查选票合法性并标记

为此将rep中的votes改变为Map<Vote<C>,Boolean>

将检查合法性的功能delegate到另外的ADT中

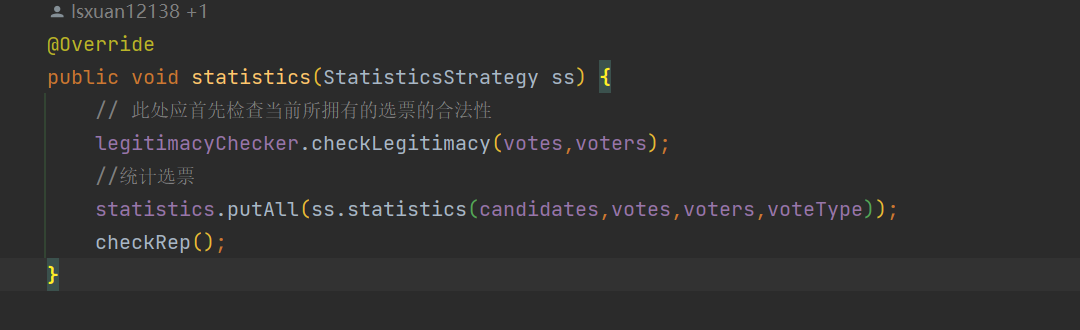


public void statistics(StatisticsStrategy ss)

先检查现有选票集的合法性

再统计选票

将这两给功能delegate到其他ADT中



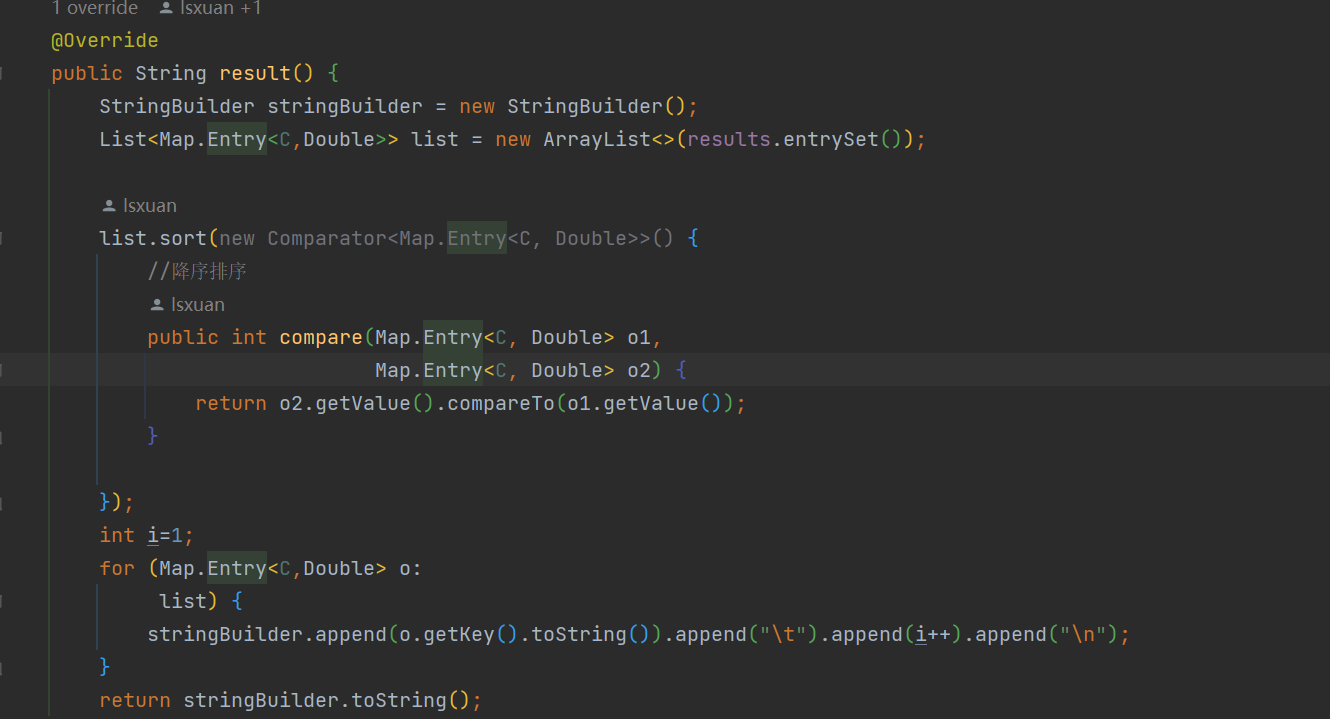
public void selection(SelectionStrategy ss)

将遴选结果放入result中

将遴选功能delegate到其他ADT中

public String result()

将遴选结果排序并输出



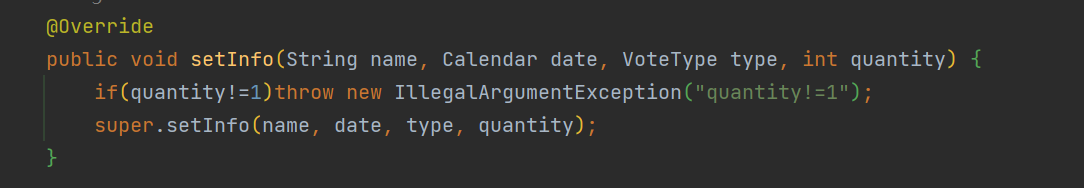
### 任务6：投票活动Poll<C>的子类型

因为在GeneralPollImpl设计时将大部分差异化的功能delegates出去，这里只需要对RI进行一些处理即可

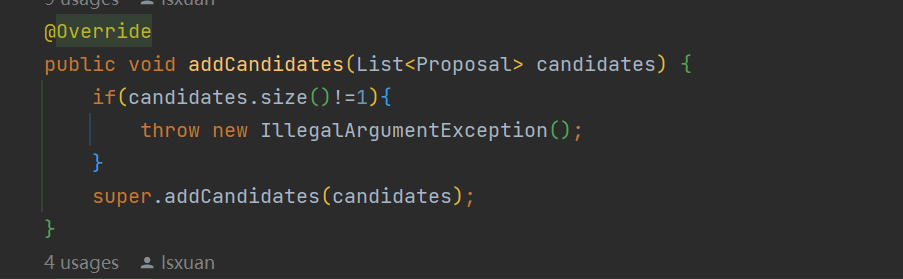
BusinessVoting

针对这个子类型，规定quantity强制为1，且candidates.size()=1

重写父类的setInfo

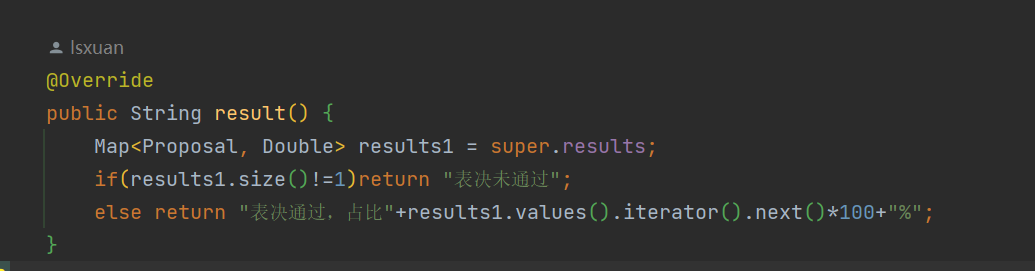


重写父类的addCandidates



另外其对结果的输出应该是表决是否通过

重写result

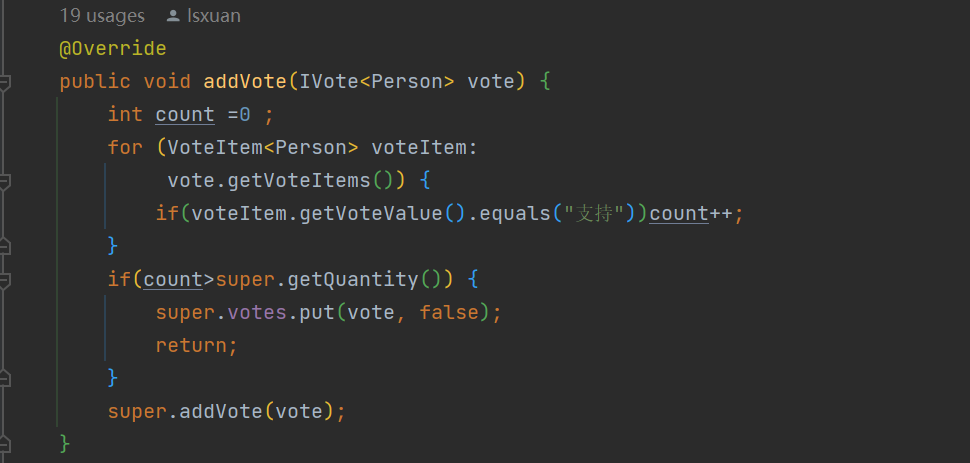


Election

针对这个子类型，不需要修改其RI，父类已经全部满足

但是对其选票的合法性有其他要求

重写addVote



因为这个要求是该应用特有的，不再单独建立ADT，只在这里处理

DinnerOrder

对这个子类型，比较特殊的是其quantity满足

voters.size()<=quantity<=voters.size()+5<=candidates.size()

那么需要对quantity和candidates做出限制

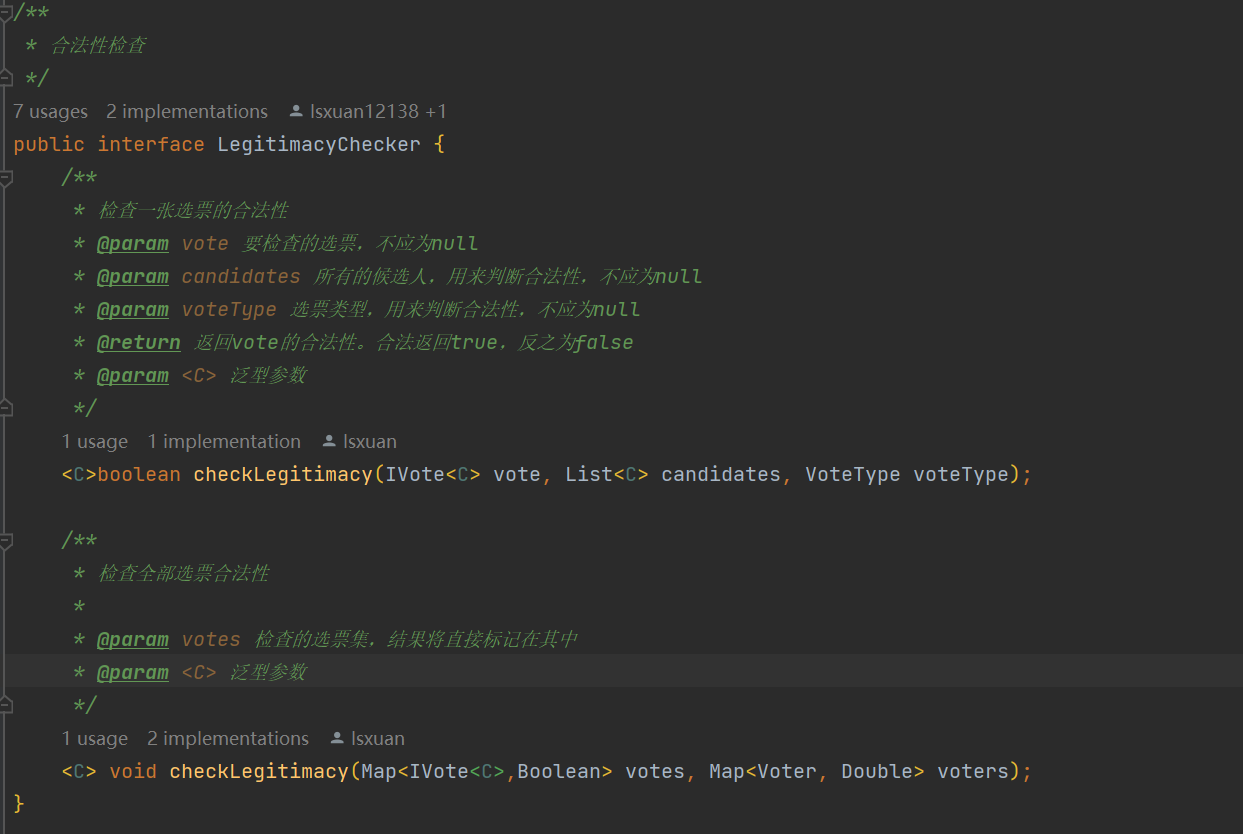


## ADT行为的设计与实现

### 任务7：合法性检查

在前面任务中，将合法性检查的功能delegate到其他ADT中，那么在这个任务中，我们就需要一套对应的ADT

首先，设定接口，其中有两个方法，分别检查单张选票的合法性和选票集的合法性（如图）



然后需要构建其对应的子类

对于不记名投票来讲构建AnonymousGeneralLegitimacyChecker

检查单张选票

检查是否包含不在本次投票活动中的候选人

检查是否出现蹦出投票不允许出现的选项值

检查是否出现对同一个候选对象的多次投票

检查一张选票中是否出现本次投票中的所有候选人

检查选票集是否合法

若尚有投票人还未投票，则不能开始计票；

因这是不记名投票，无法确定是否有人未投票或者多投票

只对选票数精选判断

对于记名投票，构建RegisteredGeneralLegitimacyChecker，继承AnonymousGeneralLegitimacyChecker

检查单张选票，其父类中方法依然适用，不再重写

检查选票集，父类中方法不再适用，重写方法

若有人还未投票则不能开始计票

若有人提交了多张票，将其均标记为非法

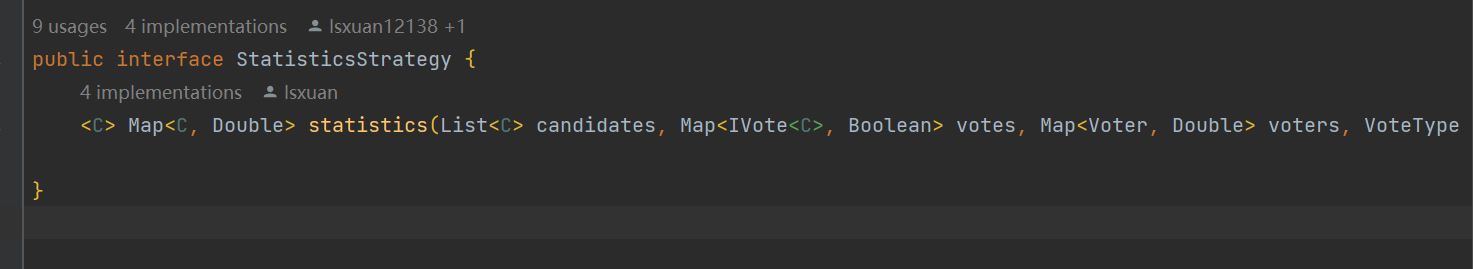
应用一和三都适用与记名投票的合法性检查

应用二适用与不记名投票的合法检查，另外针对应用二的特殊合法性检查已经在3.2.6中详述

### 任务8：采用Strategy设计模式实现灵活的计票规则

设计一套ADT来承担计票的规则

首先，设计统一的接口，其中方法用于计票



针对不记名投票AnonymousGeneralStatisticsStrategy

直接计算对应的选票的分值和

对于记名投票RegisteredGeneralStatisticsStrategy

计算分值时考虑投票人的权重

对于应用一BusinessVoting

实现自己的特殊子类BusinessVotingStatistics，只对其中“支持”计票但需要考虑记名投票的权重

对于应用二Election

实现自己的特殊子类ElectionStatistics只对其中“支持”计票且不计算权重

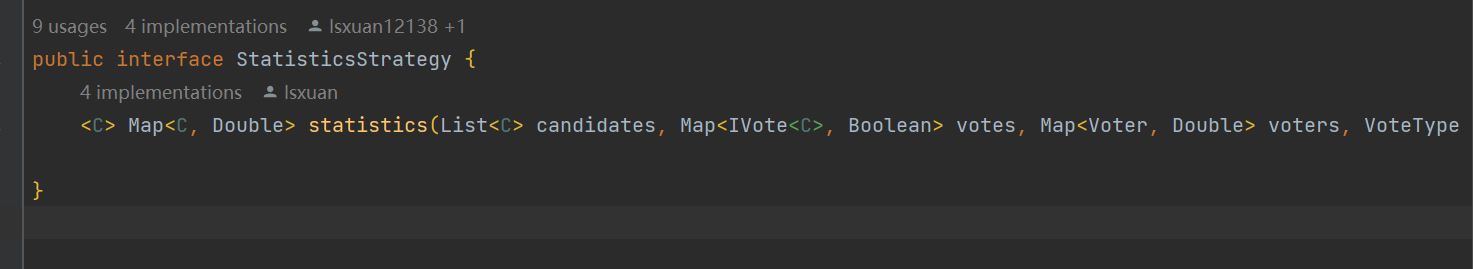
对于应用三

适用与记名投票的规则，不再构建自己的类型

### 任务9：采用Strategy设计模式实现灵活的遴选规则

设计一套ADT来承担遴选的规则

首先，设计统一的接口，其中方法用于计票



通用的遴选规则

对统计结果降序排序，选出前quantity个，若分值相同则排名随机

对于应用一BusinessVoting

实现自己的特殊子类BusinessVotingSelection，只有分值>2/3时才将其放入结果

对于应用二Election

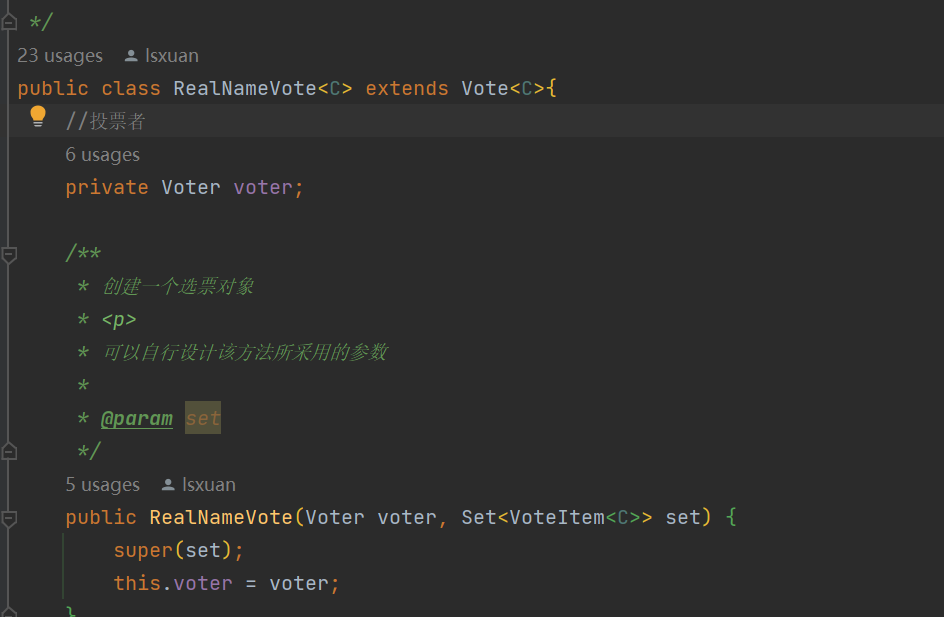
实现自己的特殊子类ElectionSelection，先将统计结果降序排序，将前quantity个放入result，若最后一个放入的分值与还未放入的下一个分值相同，则删除result中与其分值相同的元素

对于应用三

适用与通用的规则，不再构建自己的类型

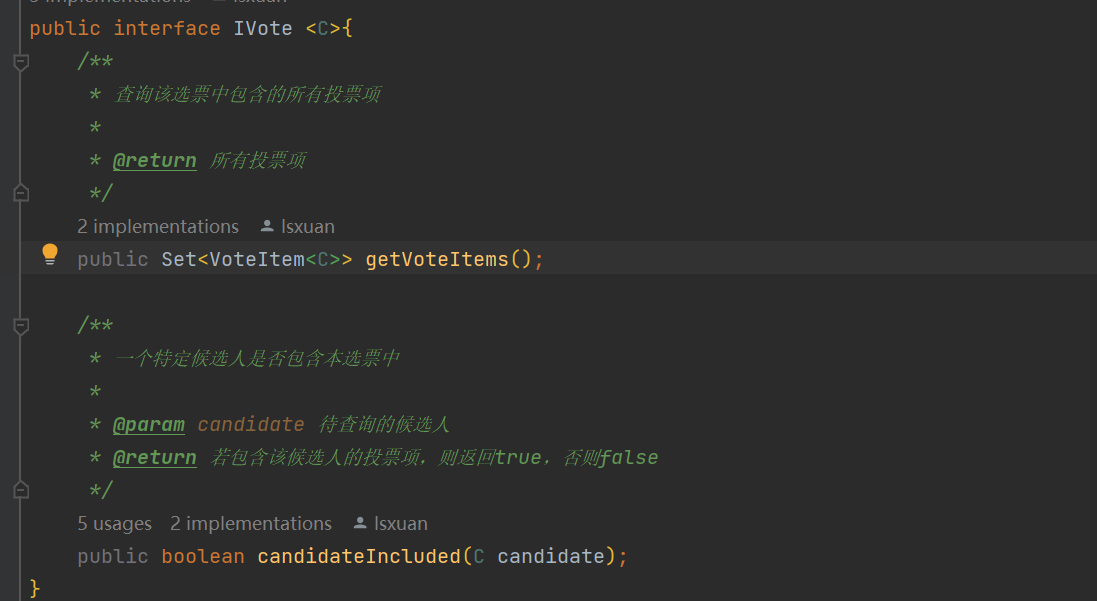
### 任务10：处理匿名和实名投票

使用子类方式实现



使用装饰器模式实现

首先抽象出一个接口

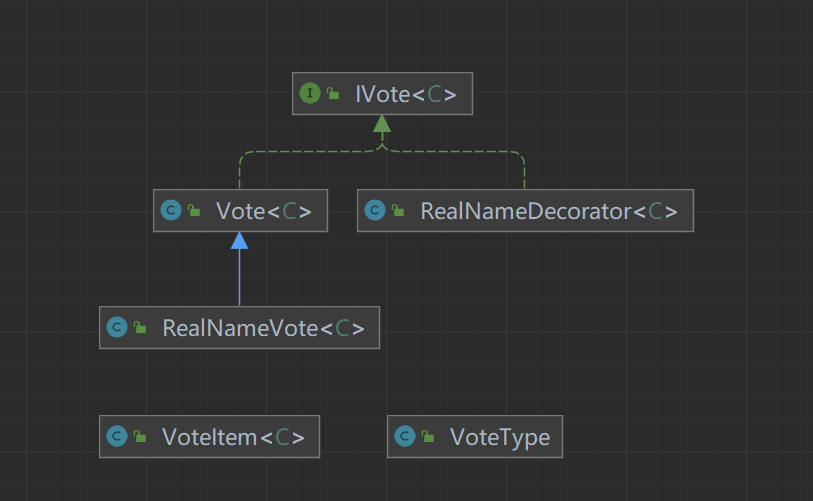


令Vote实现这个接口

再实现一个装饰器类

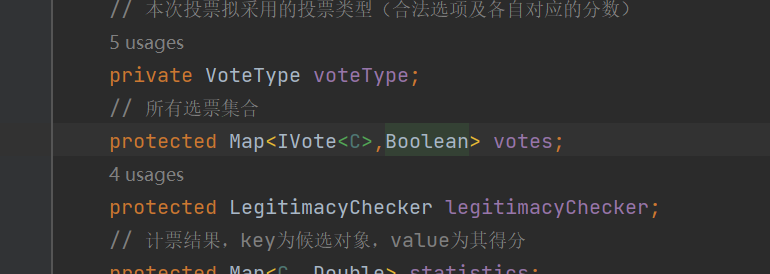


最终实现的类结构如图



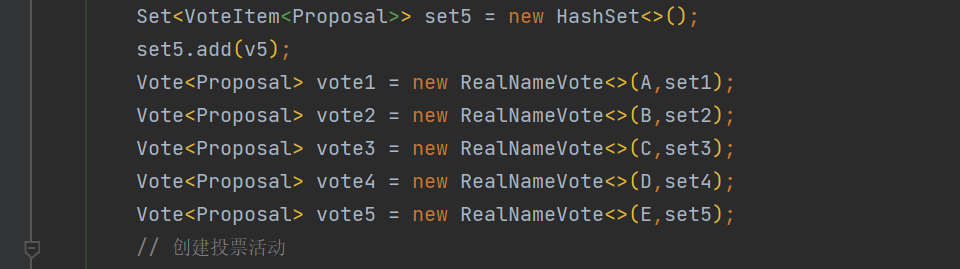
这里还需要对前面实现的结构和rep等进行修改，使用抽象的IVote来取代原有的Vote

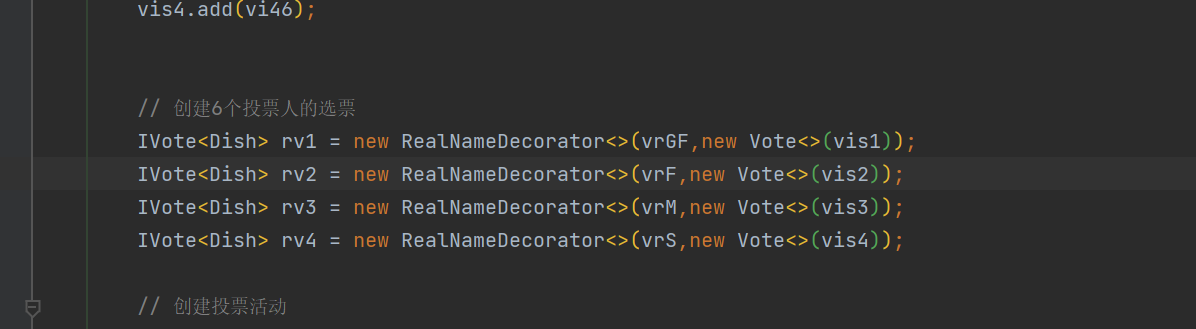
例如GeneralPollImpl类中



还有其他如这样的修改，不一一赘述

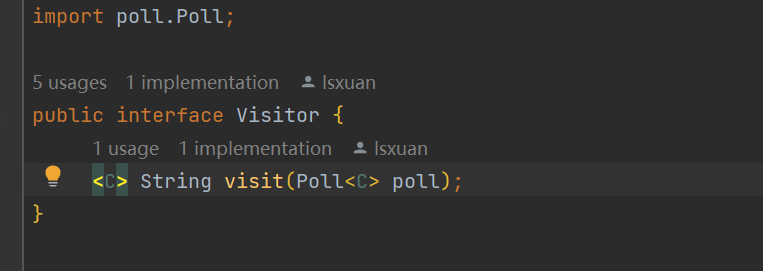
对记名投票的演示放在应3.4中，这里只简单截图





### 任务11：采用Visitor设计模式实现功能扩展

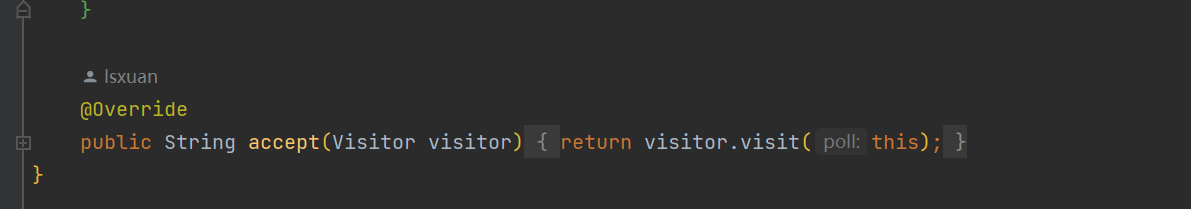
先构建一个接口Visitor



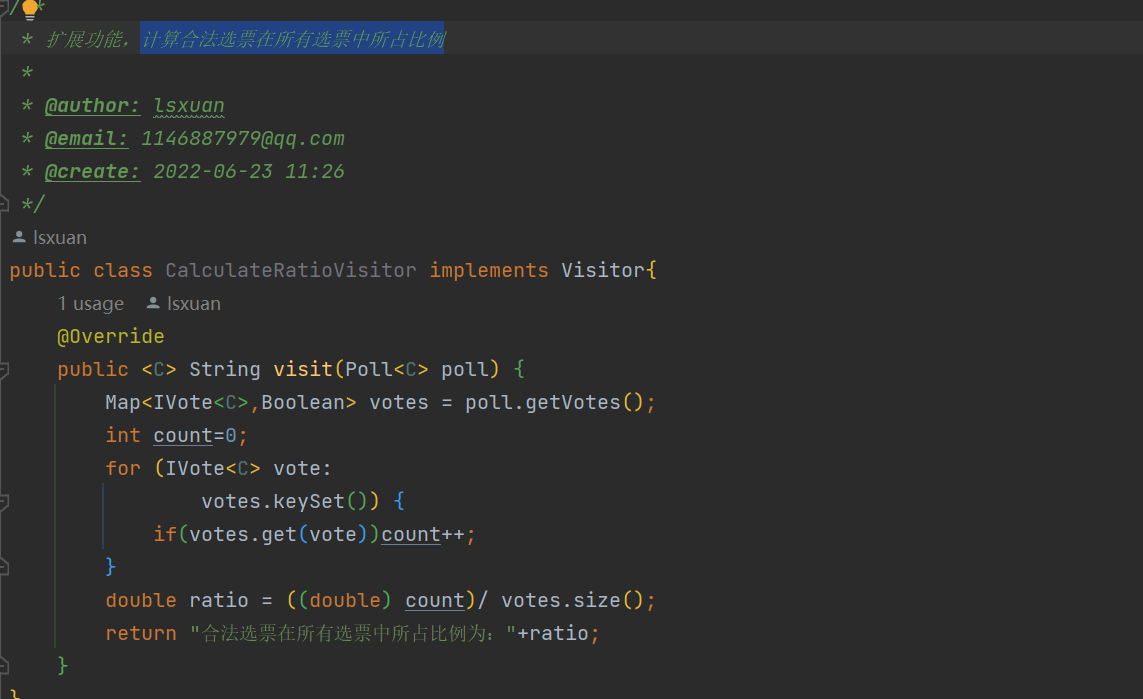
在Poll中增加一个方法



接下来在实现类中实现这个方法

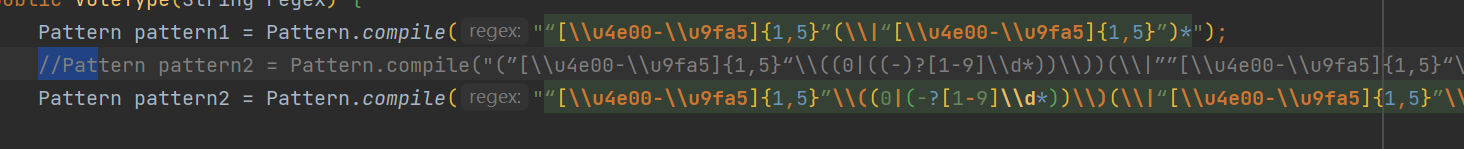


然后实现一个具体的visitor类，实现计算合法选票在所有选票中所占比例的功能



### 任务12：基于语法的数据读入

因为要求支持两种方式的读入，这里准备两种方式的正则表达式

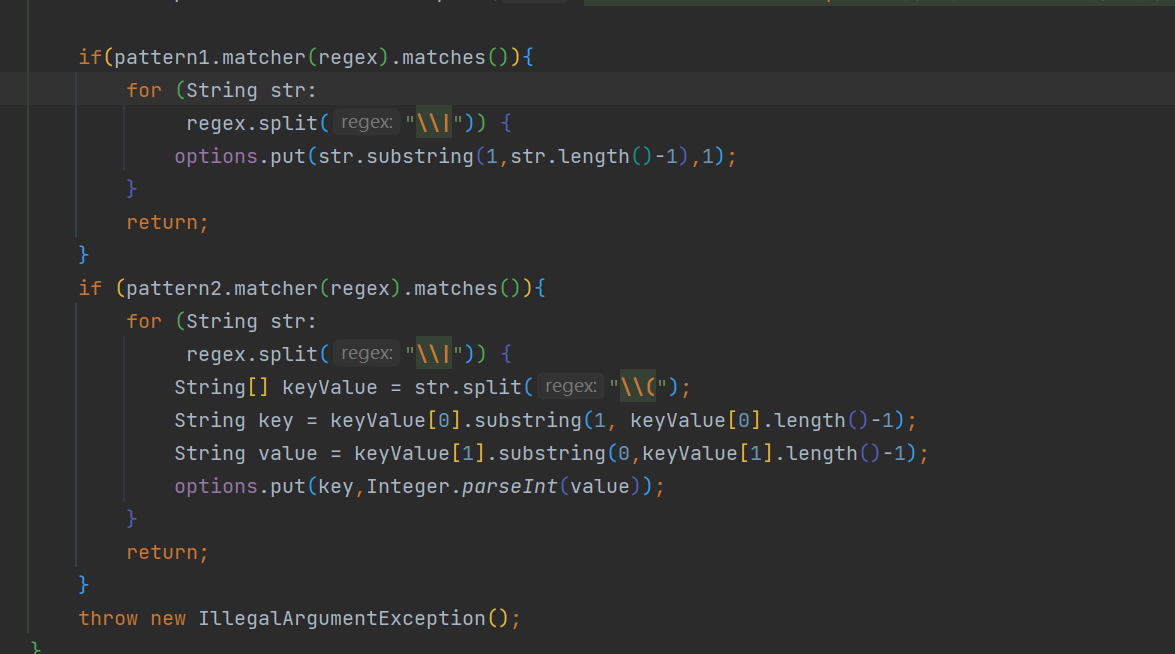


分别为

“[\u4e00-\u9fa5]{1,5}”(\|“[\u4e00-\u9fa5]{1,5}”)\*匹配符合“支持”|“反对”的字符串

“[\u4e00-\u9fa5]{1,5}”\((0|(-?[1-9]\d\*))\)(\|“[\u4e00-\u9fa5]{1,5}”\((0|(-?[1-9]\d\*))\))\*匹配符合“支持”(-1)|“反对”(0)的字符串

当两个都没有匹配到时，抛出异常，若匹配到，则采用对应方式将其解析



对该构造方法的测试

测试符合表达式“支持”(-1)|“反对”(0)的情况

只有一项的情况

有多项的情况

测试“支持”|“反对”的情况

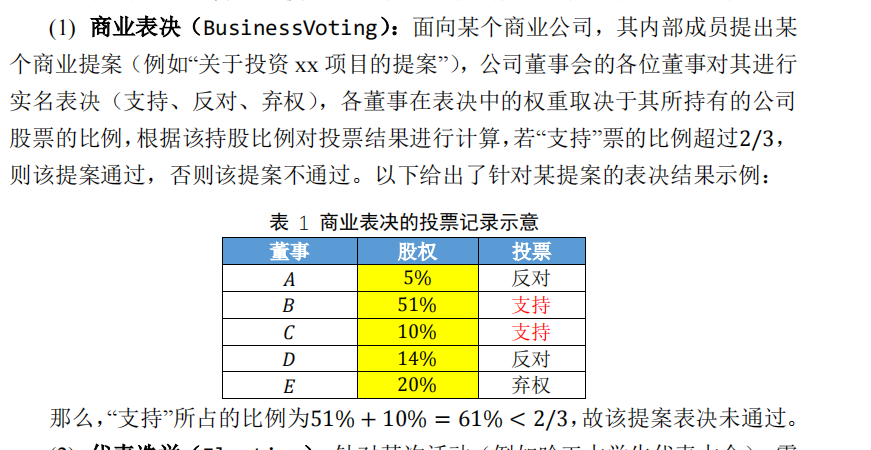
只有一项的情况

有多项的情况

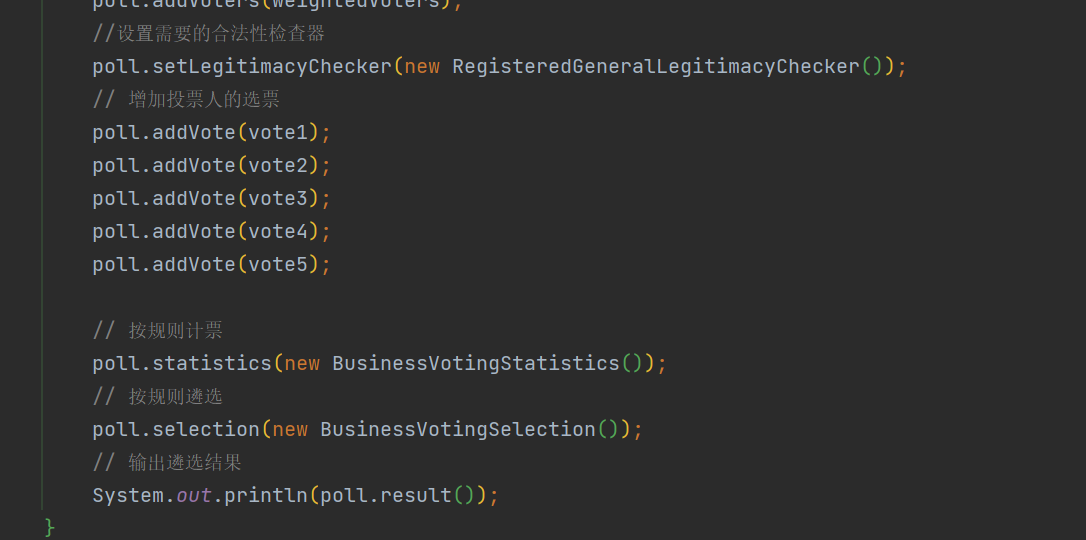
## 任务13：应用设计与开发

### 商业表决系统

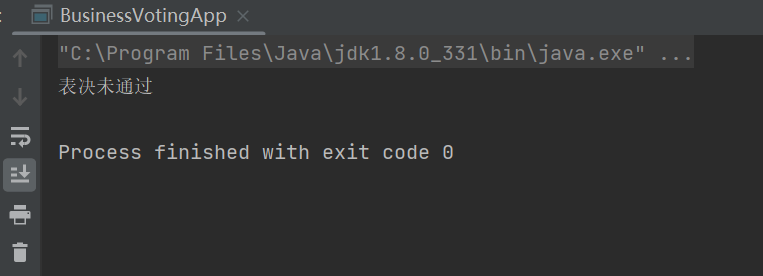
构造如图的一个商业投票



并设定对应的合法性检查，计票规则，遴选规则

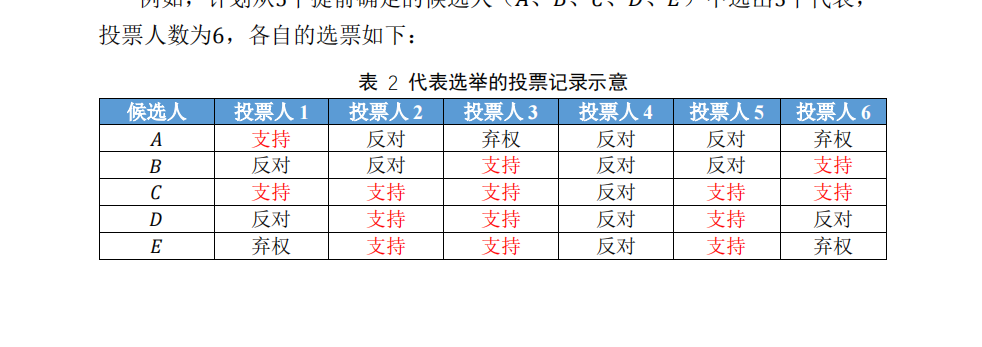


输出结果同预测一样

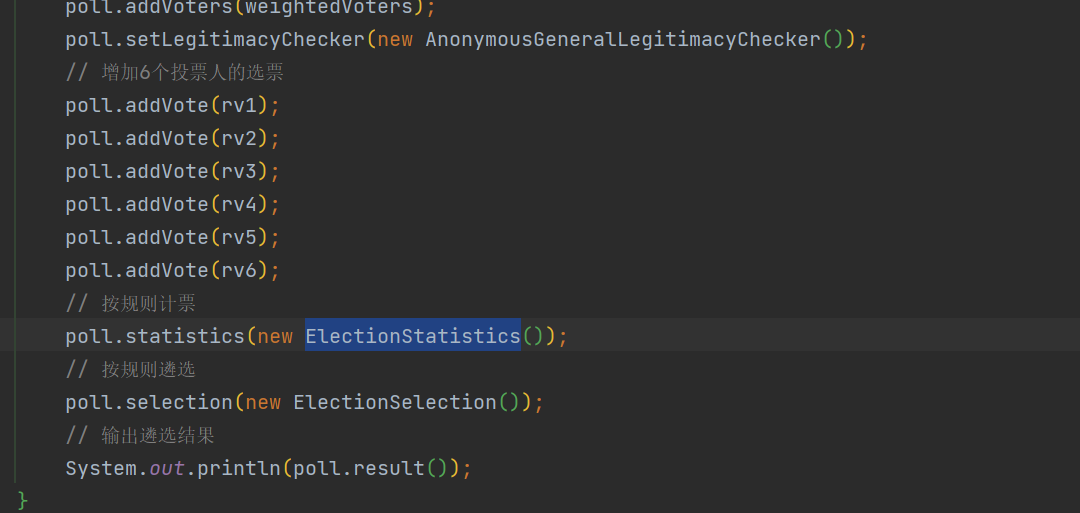


### 代表选举系统

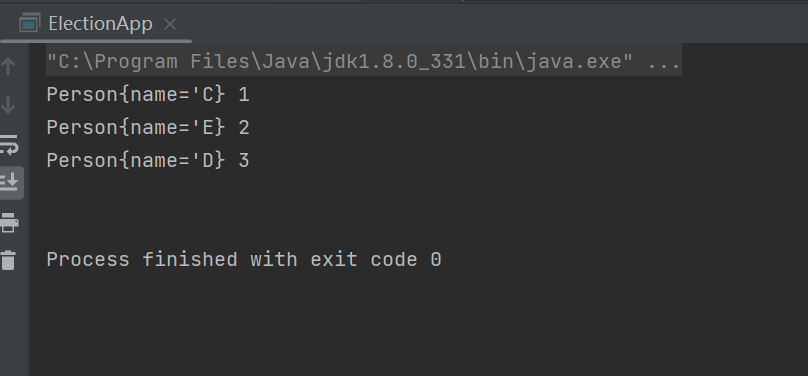
构造一个如图的代表选举



并设定对应的合法性检查，计票规则，遴选规则



输出结果同预测一样

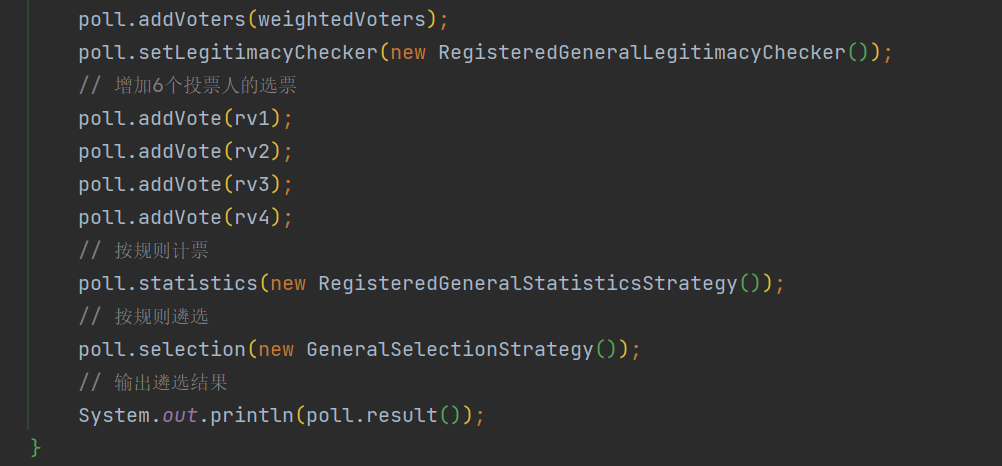


### 聚餐点菜系统

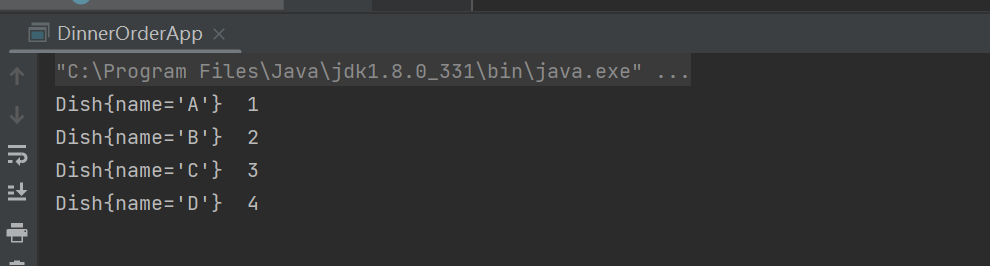
构建一个如图的点菜投票



并设定对应的合法性检查，计票规则，遴选规则



输出结果同预期一致



## 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

评估之前的设计是否可应对变化、代价如何

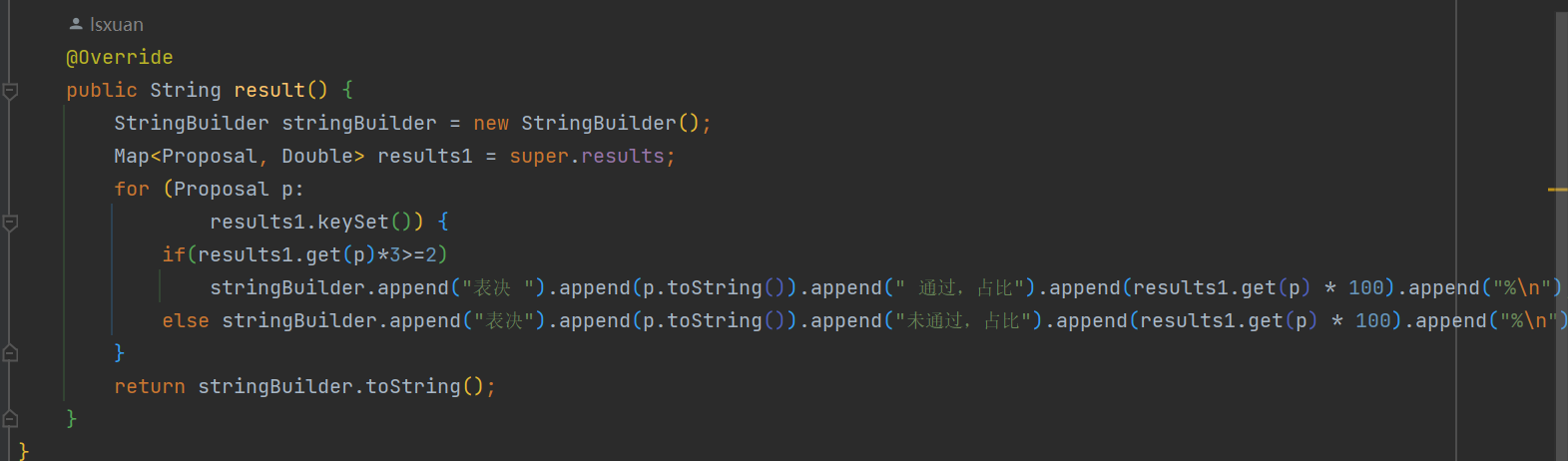
可以应对变化，代价很小

如何修改设计以应对变化

只需要修改BusinessVotingSelection中的实现和BusinessVoting的result()方法即可

BusinessVotingSelection中直接返回统计结果，不再选择

Result方法



### 代表选举应用：遴选规则变化

评估之前的设计是否可应对变化、代价如何

可以应对变化，代价较高

如何修改设计以应对变化

只需要修改ElectionSelection即可



### 聚餐点菜应用：取消权重设置、只计算“喜欢”的票数

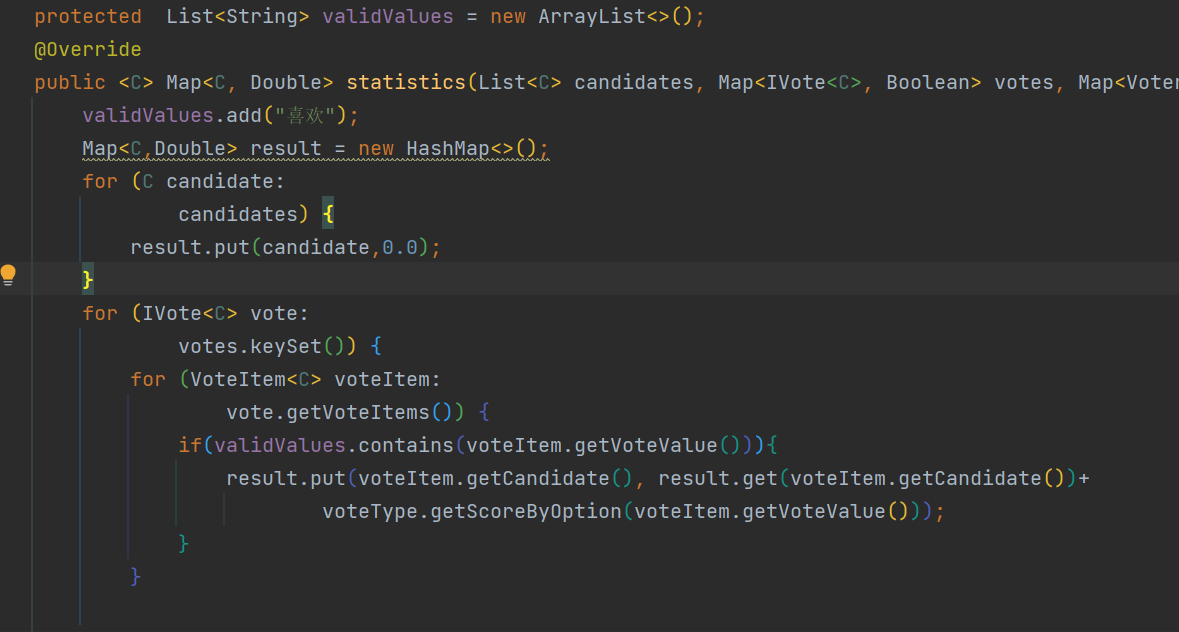
评估之前的设计是否可应对变化、代价如何

可以应对变化，代价很小

如何修改设计以应对变化

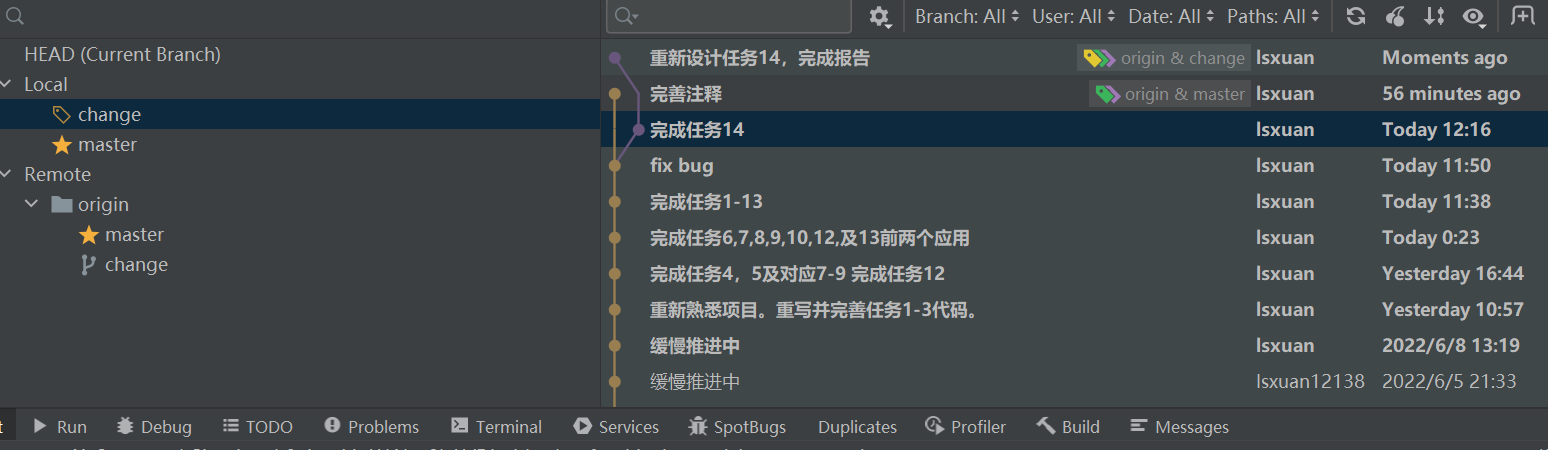
首先将投票人的权重设为一样的值

然后未新建一个计票规则即可



## 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 |
| 2022-06-04 | 13:30-14:45 | 任务1-3 | 完成 |
| 2022-06-04 | 16:50-18:45 | 任务4,5及3.4中任务 | 未完成，缓慢进行 |
| 2022-06-05 | 18:30-22:00 | 任务4,5及3.4中任务 | 未完成，缓慢进行 |
| 2022-06-22 | 10:30-11:00 | 重新熟悉项目，重写并完善任务1-3中代码 | 完成 |
| 2022-06-22 | 12:00-16:45 | 完成任务4，5及对应7-9，完成12 | 完成 |
| 2022-06-22 | 18:20-17:00 | 完善任务12 | 完成 |
| 2022-06-22 | 21:00-24:00 | 完成任务6,7,8,9,10及13前两个应用 | 完成 |
| 2022-06-23 | 10:00-11:40 | 完成任务1-13 | 完成 |
| 2022-06-23 | 11:40-12:15 | 完成任务14 | 完成 |
| 2022-06-23 | 14:00-17:15 | 完善注释，完成报告，修改bug | 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 |
| 对于三个子类型的实现重写方法时checkRep会造成循环调用，并栈溢出 | 拆分GeneralPollImpl的checkRep，并重写规划其中方法对rep的影响，使用适当的拆分后的方法 |
| 对与记名投票的实现，并根据其实现合法性检查，选票统计 | 重写之前大部分代码，将具体类型Vote换成IVote，并根据两种不同的实现方法，对方法重写编写 |
| 对代表选举应用的修改，其新的遴选规则需要取到反对票数据 | 使用类似与visitor模式的方法，将poll对象直接传给ElectionSelection |

# 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）

熟悉Java中子类型、泛型、多态、重写、重载等语法规则

基本掌握继承、委派、CRP等常用的软件构造技术

自学正则表达式并熟悉其基本使用

熟悉几种设计模式，并将其应用与代码编写中

## 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的编程更加利于代码层面，甚至组件层面的复用，在面对高度相似的使用场景下，面向ADT编程能够节省大量的代码量和时间。

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？

编写specification, invariants, RI, AF，时刻注意ADT是否有rep exposure等等可以大大减少出现错误的可能，且在方便自己当前开发的同时，也方便了自己和他人的复用

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？

在开发一套ADT的过程中，我们需要考虑各种各样的情况，因为我们并不知道ADT的用户将会怎样使用我们的ADT，要将其永远想象为恶意的，我们必须要确保ADT在spec规定的情况下可以正常的工作。

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？

语法驱动的编程是需要一定的代价去学习的，且在使用过程中也会有很多的坑。但是他是好用个的，如果对其熟练掌握，就可以以简单的语句实现复杂的功能。

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？

如果完全由自己实现，首先这是非常有难度的，虽然可能也许能够实现，但是可能实现的结果并不好用，不够美观。“设计ADT”的主要难度是在对使用场景中各种事务的共性与个性的分析抽象划分。当我们能很好的划分出共性与个性，那么将其实现也不会非常困难。

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？

首先，对于一个抽象的类型（抽象类或者接口）其中所依赖的方法等等，最好也使用抽象类型，这可以方便委派和设计模式等等软件构造方法的使用。

然后关于类的继承，我们可以根据共性来抽象父类，再根据个性来创建子类

1. 关于本实验的工作量、难度、deadline。

本次实验工作量比较大，难度也很大，deadline比较合理

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？

软件构造课程是对我过去所学的编程的一次飞跃，他不再仅仅试试面向一个具体的场景一个过程，而是面向ADT的等等更加抽象，更加高层次。他给了我许多经验教训和启发，对以后编程时会有很大帮助。

任课教师授课时专业不失幽默，回答问题及时而清晰。