

**2022年春季学期  
计算学部《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 郑旭然 |
| 学号 | 120L020719 |
| 班号 | 2003005 |
| 电子邮件 | zxrshawn@icloud.com |
| 手机号码 | 16645013218 |

**目录**

[1 实验目标概述 1](#_Toc107180162)

[2 实验环境配置 1](#_Toc107180163)

[3 实验过程 2](#_Toc107180164)

[3.1 待开发的三个应用场景 2](#_Toc107180165)

[**3.1.1 商业表决（BusinessVoting）** 2](#_Toc107180166)

[3.1.2 代表选举（Election） 2](#_Toc107180167)

[3.1.3 聚餐点菜（DinnerOrder） 3](#_Toc107180168)

[3.1.4 共性分析 3](#_Toc107180169)

[3.1.5 特性分析 4](#_Toc107180170)

[3.2 ADT识别与设计 5](#_Toc107180171)

[3.2.1 任务1：投票类型VoteType 5](#_Toc107180172)

[3.2.2 任务2：投票项VoteItem<C> 8](#_Toc107180173)

[**3.2.3 任务3：选票Vote** 10](#_Toc107180174)

[**3.2.4 任务4：投票活动Poll<C>的测试** 11](#_Toc107180175)

[**3.2.5 任务5：投票活动Poll<C>的实现类GeneralPollImpl** 11](#_Toc107180176)

[**3.2.6 任务6：投票活动Poll<C>的子类型** 13](#_Toc107180177)

[**3.3 ADT行为的设计与实现** 14](#_Toc107180178)

[**3.3.1 任务7：合法性检查** 14](#_Toc107180179)

[**3.3.2 任务8：采用Strategy设计模式实现灵活的计票规则，** 16](#_Toc107180180)

[**3.3.3 任务9：采用Strategy设计模式实现灵活的遴选规则** 18](#_Toc107180181)

[**3.3.4 任务10：处理匿名和实名投票** 19](#_Toc107180182)

[**3.3.5 任务11：采用Visitor设计模式实现功能扩展** 20](#_Toc107180183)

[**3.3.6 任务12：基于语法的数据读入** 21](#_Toc107180184)

[**3.4 任务13：应用设计与开发** 21](#_Toc107180185)

[**3.4.1 商业表决系统** 22](#_Toc107180186)

[**3.4.2 代表选举系统** 23](#_Toc107180187)

[**3.4.3 聚餐点菜系统** 24](#_Toc107180188)

[**3.5 任务14：应对面临的新变化** 26](#_Toc107180189)

[**3.5.1 商业表决应用：可以一次表决多个商业提案** 27](#_Toc107180190)

[**3.5.2 代表选举应用：遴选规则变化** 27](#_Toc107180191)

[**3.5.3 聚餐点菜应用：取消权重设置、只计算“喜欢”的票数** 27](#_Toc107180192)

[**3.6 Git仓库结构** 27](#_Toc107180193)

[**4 实验进度记录** 28](#_Toc107180194)

[**5 实验过程中遇到的困难与解决途径** 28](#_Toc107180195)

[**6 实验过程中收获的经验、教训、感想** 28](#_Toc107180196)

[**6.1 实验过程中收获的经验和教训（必答）** 28](#_Toc107180197)

[**6.2 针对以下方面的感受（必答）** 29](#_Toc107180198)

# 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：

1) 子类型、泛型、多态、重写、重载

2) 继承、委派、CRP

3) 语法驱动的编程、正则表达式

4) 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

在Project Structure中设置/src为source文件夹，/test为test文件夹。设置测试的依赖项为Junit测试，以及测试将依赖于Module src。

然后git init并git remote add origin即可。

电脑萤幕画面

描述已自动生成

我的GitHub Lab3仓库的URL地址：

<https://github.com/ComputerScienceHIT/HIT-Lab3-120L020719>

# 实验过程

## 待开发的三个应用场景

* + 1. **商业表决（BusinessVoting）**

面向某个商业公司，其内部成员提出某个商业提案，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。

### 代表选举（Election）

针对某次活动，需要从一群候选人中选出若干人，作为代表参加活动。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𝑘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𝑘，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，也就是说并列排名后超出选举名额的那些人没办法最终入选。

实验指导书给了一个很好的例子。例如，计划从5个提前确定的候选人（𝐴、𝐵、𝐶、𝐷、𝐸）中选出3个代表，投票人数为6，各自的选票如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 候选人 | 投票人1 | 投票人2 | 投票人3 | 投票人4 | 投票人5 | 投票人6 |
| A | 支持 | 反对 | 弃权 | 反对 | 反对 | 弃权 |
| B | 反对 | 反对 | 支持 | 反对 | 反对 | 支持 |
| C | 支持 | 支持 | 支持 | 反对 | 支持 | 支持 |
| D | 反对 | 支持 | 支持 | 反对 | 支持 | 反对 |
| E | 弃权 | 支持 | 支持 | 反对 | 支持 | 弃权 |

表 3‑1 代表选举的投票记录示意

可以看出，投票人 1、2、4、5、6 的选票都是有效选票，但投票人 3 投了 4票“支持”，超过了法定当选人数 3，故其选票无效。对有效选票进行统计，候选人𝐴-𝐸所得支持票的数目分别为：1、2、5、3、3，候选人𝐶、𝐷、𝐸的票数排名前三，故成功选出了 3 位代表。如果本次选举希望选出 2 个代表，那么因为𝐷和𝐸的得票相同，则只能选出𝐶，而𝐷和𝐸无法当选。

### 聚餐点菜（DinnerOrder）

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，点菜的数量为 [总人数, 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，权重也不同。所有人投票后，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。

例如，4 个人去吃饭，从菜单的 6 道菜中选择 4 道菜，各自偏好如下表所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **菜品** | **爷爷** | **爸爸** | **妈妈** | **儿子** | **总得分** |
| 4 | 1 | 2 | 2 |
| A | 喜欢 | 无所谓 | 喜欢 | 喜欢 | 17 |
| B | 喜欢 | 喜欢 | 不喜欢 | 无所谓 | 12 |
| C | 无所谓 | 喜欢 | 不喜欢 | 喜欢 | 10 |
| D | 无所谓 | 喜欢 | 不喜欢 | 喜欢 | 10 |
| E | 不喜欢 | 不喜欢 | 喜欢 | 喜欢 | 8 |
| F | 不喜欢 | 喜欢 | 不喜欢 | 不喜欢 | 2 |

针对菜品𝐶，有 2 个“喜欢”票、1 个“不喜欢”票，1 个“无所谓”票，其总得分为4×1+1×2+2×0+2×2=10。用相同的计算方式得到其他菜的得分，最终计算排名结果为：𝐴>𝐵>𝐶=𝐷>𝐸>𝐹，故选择𝐴𝐵𝐶𝐷四道菜。如果最终六道菜的总得分分别为17、12、8、8、8、6，那么前两道菜一定可以排入前 4 名，但第 3~5 道菜得分都是8，无法区分，此时从中随机选出 2 道菜，形成了最终入选的 4 道菜。

### 共性分析

在这三个应用中有很多共性的地方，因此可以设计一套统一的ADT来维护它们。

首先是这三个应用都属于一种投票活动，投票活动中都是一群人对着另一些对象表达自己的意见；投票项都是投票时的具体投票信息，例如对提案的表决、对被选举人的选票、对菜品的偏好。选票也有共性，体现在多张选票，即选票的集合上。

### 特性分析

三个应用都有各自的特性。

**候选对象不同。**分为商业提案、学生候选人、菜品。因此可以在 ADT 中引入泛型参数，表征不同应用中处理的不同候选对象类型，不同应用中可以替换为具体的类型（提案、候选人、菜品）。

**候选对象数量不同。**“= 1”是“≥ 1”的特例，是更强的约束条件。ADT 设计时可以按“≥ 1”设计，在“商业表决”应用的 ADT 子类中强化 RI 中的“≥ 1”为“= 1”。

**拟遴选出的候选对象数量k不同。**在后两个应用中，不管 k 需要满足什么条件，都是一个提前确定的值，在构造 ADT 对象的时候作为输入即可，但需要在 ADT 内部检查 k 的值是否合法。在应用 1 中，k 可以是 0 或 1，本质上也是确定的“1”（因为“0”可以看作是该提案没有满足遴选规则）。因此该维度可以一致化处理，只不过在各个应用的 ADT 子类中强化 RI 所需满足的不同条件。

**投票人类型**的话，不管是股东、学生还是就餐者，他们都是“人”，且三个应用中对它们管理的唯一要求是其身份ID，因此在设计时候无需考虑它们之间的差异，统一处理即可。

**投票类型**的话可以采用统一的方式进行抽象，存储投票的合法选项及其对应的分数。虽然应用 1 和应用 2 中，分数在后续计票和遴选中不起作用（主要是看“支持”票的数量或比例），但这种统一抽象不妨碍应用 1 和应用 2的实现。

**实名情况不同。**“实名”和“匿名”的区别在于投票记录 ADT 中是否保管投票者的“身份 ID”。可基于“匿名”设计父类型，然后在子类型上扩展“实名”特性。

**不同投票类型的数量限制**也需要考虑。该维度的限制主要是来自于应用 2。但违反该限制条件只是表明某选票是不合法的，但仍然是允许存在的。可以在合法性检查的时候进行判定，但不能作为 RI 中的条件。

**投票人权重情况不同。**该维度描述的“权重”是投票人参与到某个投票活动时产生的，不应作为“投票人”的固有属性。应用2中虽然所有投票人权重一样，但本质上相当于还是有权重，故可以“有权重”为基础设计父类型 ADT，在应用 2 的子类型 ADT 中强化 RI，确保所有投票人的权重一样。

**检查合法性的要求不同。**三个应用有共性的合法性检查要求；除此之外，应用2还需要检查额外的条件。

**计票规则不同。**该维度体现在 ADT 要提供的“计票”方法中，不同应用的计票方法内部的行为算法不同，可以采用设计模式或子类型的方式加以解决。

**遴选规则不同。**该维度体现在 ADT 要提供的“遴选”方法中，但实现机制不同。可以采用子类型 override 的机制，在子类型中按不同的遴选规则对相关方法进行重写。也可以采用某种合适的设计模式加以解决。

## 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

首先写一下checkRep()，限制是不能有空格，或者长度超过5也不行。。然后给Votetype()写构造方法，我这里一共写了两种，无参的和regex作为参数的。

文本

描述已自动生成

实际上做到后面发现，任务12中我们需要支持正则表达式，只需要把分数最高的那个值作为“赞成”，并且注意defensive copy。

后面的checkLegality()和getScoreByOption()直接查询即可。

文本

描述已自动生成

测试也是正常写即可，如下，给出一个片段：

文本

描述已自动生成

### 任务2：投票项VoteItem<C>

首先还是写checkRep()。下面这里candidate为泛型的，还有一个String value，表示给candidate投一个value的票。RI是照旧的。如果checkRep()不通过的话抛出异常即可。assert十分影响性能, 实际应用时要注释掉, 所以有了后面抛异常的方法检查前置条件。

文本

描述已自动生成

另外是在VoteItem中的equality判定时，要注意这一equality其实看的是VoteItem中Set的equality，然后对元素进行equals调用，因此如果只考虑candidate的equality可能会错把在同一集合中的但实际选项不同的选票判定为equals，所以针对这一点要进行区分。后面的get方法正常返回即可。

然后补全测试就行了。建立一个投票项，获得投票对象和投票项；尝试修改返回的内容；测试equals。

文本

描述已自动生成

* + 1. **任务3：****选票Vote**

这里需要我们补全Vote.java。构造方法和getter直接返回immutable field即可。这里的checkRep()直接返回true即可。另外不能有重复投票项，set容器将会保证这一点；candidate有且仅有全部候选人，不多也不少，以及value是合法投票项，这两点将由GeneralPollImpl保证。

文本

描述已自动生成

之后的创建选票对象应保证voteItem是immutable的，以及查询选票对象中的所有投票项时要保证没有表示泄露；candidateIncluded用来查询之前是否已经存了该candidate，遍历查询即可。

* + 1. **任务4：投票活动Poll<C>的测试**

这里为了完成测试还需要补充一些getter方法。

直接测试创建功能即可，其他的全在create()里。

文本

描述已自动生成

* + 1. **任务5：投票活动Poll<C>的实现类GeneralPollImpl**

这里是要实现三个子类中共性的地方。然后在各子类再重写即可。这里顺便说一下Poll接口，我们返回一个Election类型即可，这样就完成了Poll.java。

在写GeneralPollImpl.java的时候，因为需要知道哪些票合法，哪些票非法，又不能直接删除，我们新构造一个HashSet保存非法选票。另外，某些字段改为protected以便被子类访问。修改完效果如下：

文本

描述已自动生成

然后再去完成剩余的方法。有几个注意的地方，比如说setInfo()的date、addVoters()都需要defensive copy；addCandidates()中List里重复的候选人需要我们用HashSet来解决。

下面重点谈一下addVote()。这个方法是将投票加入到投票列表中。实现这里的时候一开始没有考虑全面，直到任务7中对addVote()提出了要求，因此又进行了修改。我们看一下任务7所规定的：

文本, 信件

描述已自动生成

addVote()时，首先检查该选票的合法性并标记，为此需扩展或修改rep，保证候选人名单不为空；已经重写了VoteItem<C>的equals方法，可以保证不出现对某一个人投多个相同票（即没有指向的候选人一样，并且投票的类型一样的票）；只需要判断没有对一个人的重复投票（即没有对某个候选人投多个类型不同的票），并且保证不出现其他候选人即可，这里默认选票合法。同时保证选票指向的候选人在候选人清单中，并且投票选项是给出的选项之一。

接下来则是检测票中有没有对同一个候选对象的多次投票项,全部检测还可以排除存在某个候选人漏掉的可能，第二次遍历，在找到相同元素后根据flag2进行操作。最后对可能的一些废选票作废。候选人名单为空则均为非法票。

然后VoterVote()分别处理投过票和没投过票的人。最后补上辅助的方法，这些其实在之后能看出来对statistics的完成是必要的，对ADT合法性有很大的帮助。

文本

描述已自动生成

然后是selection()方法。selection()计算出投票结果并存到results中。具体的计算还是委派给SelectionStrategy。最后results根据selection返回最终结果，是一个字符串。selection会告诉results候选对象及其得分，我们先根据value排序，然后便得出了排名，最后进行处理输出。

* + 1. **任务6：投票活动Poll<C>的子类型**

先开发三个子类型。回顾GeneralPollImpl.java，在BusinessVoting中需要重写addVote和checkVotes。前者需要我们进行实名投票，所以检查一下是否为RealNameVote，否则抛出异常。实名投票还需要考虑投票人是否在GeneralPollImpl.java的rep中，这样就完成了子类合法性的判断。实名投票时还需要考虑所有人是否都投票了，然后再按照votedVoters再判断一次，看两次是否相等。如果两次结果不一样的话返回false；另外将一个人投票不止一次的归到illegalVotes里即可。DinnerOrder几乎都是一样的，只是Proposal和Dish不同而已。

Election.java中只需要重写addVote即可。刚才我们看过下面这张图，现在看一下第五点，处理方法是看一下传进来的Vote里支持票有多少。

文本, 信件

描述已自动生成

通过supportVote()来得知赞成票数。

* 1. **ADT行为的设计与实现**
     1. **任务7：合法性检查**

这里我们在任务5中设计的时候已经谈过大部分了，现在回顾一下这一部分。代码中已经给出了addVote()和statistics()方法，现在需要我们对以下两点进行检查:

文本

描述已自动生成

addVote()时，首先检查该选票的合法性并标记，为此需扩展或修改rep，保证候选人名单不为空；已经重写了VoteItem<C>的equals方法，可以保证不出现对某一个人投多个相同票（即没有指向的候选人一样，并且投票的类型一样的票）；只需要判断没有对一个人的重复投票（即没有对某个候选人投多个类型不同的票），并且保证不出现其他候选人即可，

文本

描述已自动生成

这里默认选票合法。

文本

描述已自动生成

上图保证选票指向的候选人在候选人清单中，并且投票选项是给出的选项之一。

文本

描述已自动生成

接下来则是检测票中有没有对同一个候选对象的多次投票项,全部检测还可以排除存在某个候选人漏掉的可能，第二次遍历，在找到相同元素后根据flag2进行操作。最后对可能的一些废选票作废。候选人名单为空则均为非法票。

然后VoterVote()分别处理投过票和没投过票的人。最后补上辅助的方法，这些其实在之后能看出来对statistics的完成是必要的，对ADT合法性有很大的帮助。

文本

描述已自动生成

文本

描述已自动生成

* + 1. **任务8：采用Strategy设计模式实现灵活的计票规则，**

这里为三个应用分别构建 StatisticsStrategy 的子类型。我们在pattern包下新建类：BusinessStatisticsStrategy、DinnerStatisticsStrategy、ElectionStatisticsStrategy。我们以DinnerStatisticsStrategy为例：

首先是很长的一段用Map获取字段。

文本

描述已自动生成

对Statistics()重写，首先投票次数只能为1次，删去其他的。挑出剩下的选票，在有效票中确定一个票是否是合法票，投票人的投票次数为1,即查找对应投票人是否仍在有效集中，从实名投票集中找到投票人，查找投票人是否还在有效投票人集合中；删去非法票，最后flag为假则删掉票。

文本

描述已自动生成

在最后一个for循环中统计合法选票中支持票的数量，遍历每一道菜，然后把这道菜的评分都加到一起统计出结果即可。

文本

描述已自动生成

* + 1. **任务9：采用Strategy设计模式实现灵活的遴选规则**

这里为三个应用分别构建 SelectionStrategy 的子类型。我们在pattern包下新建类：BusinessSelection、DinnerOrderSelection、ElectionSelection。我们以ElectionSelection为例：

首先照例是Map获取字段。

文本

描述已自动生成

重写select，进行排序。这里没啥好说的，得善用getter方法才行。特别注意，如果是同票的情况会对结果有影响，需要进行下图的处理；最后返回候选人列表。

文本

描述已自动生成

* + 1. **任务10：处理匿名和实名投票**

匿名投票其实是基础，有匿名投票才有实名投票，实名投票是需要有身份的证明的，所以我们这里传进来一个voterID，以及投票项voteItems，然后获取一个new Voter。接着将这个实名投票者返回即可。

文本

描述已自动生成

* + 1. **任务11：采用Visitor设计模式实现功能扩展**

这个任务中我们需要给ADT构造一个Visitor接口来统计合法票数量。Visitor要想进行访问，就需要在GeneralPollImpl做准备，比如我们需要接受visit过来的数据，为此我们在GeneralPollImpl中留一个方法接受visitor为参数；实现的时候在Visitor接口中直接去访问Poll。

文本

描述已自动生成

现在我们回顾GeneralPollImpl，我们可以看到有两个Set<Vote<C>>，那么Visitor去visit的时候就不知道该访问哪一个了，也就是说很容易混淆，因此需要有参数才行。所以我们直接visit Poll，如下图所示，这样自由度就更高了。然后直接针对任务编写代码就行了，很巧的是我们之前对非法选票有过考虑了，这里简单写一下就行了。



不过这里Visitor设计模式在最终代码不一定在明面上会保留，因为注意到任务14的修改，可能会优先考虑将visitor融合进代码中来换任务14更好的实现，最后我考虑将visitor模式写作这一方法：

文本

描述已自动生成

* + 1. **任务12：基于语法的数据读入**

任务12考察的是正则表达式，限制就是没有空格，长度在5之内，并且支持了任务12所描述的两种输入。然后对于“喜欢”、“不喜欢”、“支持”、“反对”等等，用正则表达式去“迎合”，然后分成两种情况：带数字和不带数字的。写出来正则表达式，加上转义符之后发现很乱，其实就是先考虑肯定没有空白符，所以是\S+，然后有数字的就额外加上一个[+-]?，作为符号，然后是\d+即可。

后来感觉上面那种是比较麻烦的。在代码中我另外给出了这样一种正则表达式：翻译一下也可以写作下面这样。

文本

描述已自动生成

* 1. **任务13：应用设计与开发**

任务13看下来之后感觉更像是一种“测试”，验证一下之前的东西能否正确运行。下面就简单复述一下大致的过程，只需要按照步骤走一遍即可。

* + 1. **商业表决系统**

首先是创建候选对象，就是候选人。接下来创建投票项，然后我创建了两个投票者以及两个人的票。最后进行如下的例行步骤：创建一次投票，然后加入投票者，候选对象以及选票信息等，最后调用策略模式，输出投票结果。

文本

描述已自动生成

文本

描述已自动生成

* + 1. **代表选举系统**

这与商业表决系统的步骤差不多，依然是先创建候选人，创建投票项，最后进行如下的例行步骤：创建一次投票，然后加入投票者，候选对象以及选票信息等，最后调用策略模式，输出投票结果。

文本

描述已自动生成

文本

描述已自动生成

* + 1. **聚餐点菜系统**

聚餐点菜的实现过程略有不同。也是先创建投票人，对喜欢、不喜欢、无所谓赋值，对菜品进行定义；接下来创建投票项，依然是创建两个投票人选票，最后创建一次投票，然后加入投票者，候选对象以及选票信息等，最后调用策略模式，输出投票结果。

文本

描述已自动生成

文本

描述已自动生成

然后实名加入投票和投票项，在最后statistics进行计算，并给出选出的结果。

* 1. **任务14：应对面临的新变化**

这里先创建一下change分支。

手机屏幕的截图

描述已自动生成

任务14包含了三个子任务的修改：

1. 商业表决应用：可以一次表决多个商业提案；
2. 代表选举应用：在遴选阶段，如果因多个候选人的支持票数量相等而无

法自然排出前𝑘名，则考虑他们获得的反对票数量，反对票数量越少，其

排序越靠前。若考虑反对票之后仍然无法自然排出前𝑘名，则只遴选那些

明确可进入前𝑘名的人作为最终结果。

1. 聚餐点菜应用：不考虑就餐者身份的差异，取消权重设置，且只计算“喜

欢”的票数。

下面分三点修改叙述。（这一部分只是简单说明修改的部分，具体的细节在代码中实现；而且由于之前实验指导书所引导的各种策略模式、委派等等，这一部分至少会轻松一些；唯一麻烦一点的是代表选举应用）

* + 1. **商业表决应用：可以一次表决多个商业提案**

这个功能是在设计的时候本来就有的.我们注意poll中BusinessVoting设计成能够存储多个投票项即可；但如果设计只能存一个就要修改了。

* + 1. **代表选举应用：遴选规则变化**

这一点修改可以理解为在考虑反对票数量的基础上，其他逻辑不变。我们就试着从最开始重新统计数据，这样还可以不破坏原来的代码。这里我们新建NewElectionSelectionStrategy，然后在其构造方法里给出三个参数，计算一下反对票的信息。Selection里我们要调用新策略，并且重写一下selection的排序规则，也就是加上比较反对票的那一步。再之后就和之前的ElectionSelectionStrategy差不多，比较相邻两个人支持票和反对票数量。

* + 1. **聚餐点菜应用：取消权重设置、只计算“喜欢”的票数**

我们之前用到了Strategy模式，那么对于取消权重的修改就很简单了，我们创建NewDinnerStatisticsStrategy，然后将原来的策略替换掉即可。

* 1. **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。

图示

描述已自动生成

1. **实验进度记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 5.30 | 20:00-21:00 | Git clone；Git init以及配置Project structure；完成实验目标概述以及实验环境配置 | 按计划完成 |
| 6.24 | 13:00-17:30 | 完成immutable的ADT | 按计划完成 |
| 6.25 | 9:30-12:45 | GeneralPollImpl、三种子类型 | 按计划完成 |
| 6.25 | 17:00-22:00 | 三种子类型 | 按计划完成 |
| 6.26 | 8:30-13:00 | 初步完成任务11-14，修改三种子类型 | 按计划完成 |
| 6.26 | 14:45-18:30 | 完善任务11-14 | 按计划完成 |
| 6.26 | 21:00-22:00 | 完善报告 | 按计划完成 |

1. **实验过程中遇到的困难与解决途径**

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 给出的框架中存在表示泄露，例如Calendar date这里不好进行defensive copy。 | 实验中要求不能修改框架。我们使用Java11，其实可以使用java.time来代替Calendar date这些，而且是immutable的，因此没有safety from rep exposure。 |
| Auxiliary包中有很多mutable类，有很多风险 | 可尝试选择修改为immutable，但实验中我们无法修改代码框架，如果实在需要的地方可选择性修改。 |
| 任务14代表选举的修改方式的选取 | 当时想了两种方案，一种是接口里和Poll子类的statistics方法都要改，一种是重新统计。一开始按照前者改动感觉代价太大了，于是换成了后者 |

1. **实验过程中收获的经验、教训、感想**
   1. **实验过程中收获的经验和教训（必答）**
2. 为ADT撰写mutability/immutability、AF、RI、safety from rep exposure、方法的spec、为测试写testing strategy，很耗费时间但是十分值得。
3. 要加强代码的功底，小失误要少犯。
4. Spec很重要，在具体开始工作之前就要认真写每个方法的spec，spec还要写好方法细节之处的功能和注释，这样可以为后面的编写省去很多麻烦，也能明确责任。
5. 从ADT开始构思和设计进而完成一个项目；测试时先测试完ADT的正确性后再检查整个项目是否正确。在给出ADT的规约后就开始编写测试用例，可以在写测试时就将AF和RI考虑好，进一步完善spec。
   1. **针对以下方面的感受（必答）**
6. 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编程复用性会更强，可能在前期的设计与实现过程会比较繁琐，但在以后使用的时候会更灵活，更加模块化。面向应用场景编程的复用性相比之下不会很高。是的，体会到了复用的好处。这样的复用让工作更方便了。

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？

* 能够使程序正确性和鲁棒性得到保证，更大可能避免了不该发生的错误；是的，因为这其实是很有意义的，对于后面的测试有好处，虽然前期适应可能需要花费功夫。

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？

* 自己开发给别人使用的ADT或API需要我们考虑用户的感受，坚持写好各方法的spec，撰写AF、RI等，这样才能让用户正确而又便捷的使用我们的程序。

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？

* 这里需要我们从文本中总结出规律，并且抽象地表达成正则表达式，这需要我们有很强的逻辑能力。

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？

* 那样的话肯定会更加吃力。我认为设计ADT的难度主要体现在考虑实际问题是否全面，构思是否完善，而且可能会在开始写代码之后修改之前的ADT，思考问题一定要全面才行，我觉得这是难点；还有就是平常的时候就需要熟悉代码框架。

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？

* 抽象类不能抽象成同一个，这个时候我们就需要接口，将接口实现成各种抽象类，抽象类再去实现成具体的类，并且使用我们课上学到的各种设计模式，例如装饰器模式、工厂模式等等，考虑将复杂的任务委派到一些子类中进行，这样可以使得我们的工程逻辑更加清晰，修改和测试等等都会更加方便。我认为刚开始写时我还有很多东西没能考虑全面，导致很多地方需要返工，而完成最大程度的抽象和复用恰恰又是十分重要的，应该在平常的学习和工作中有意地去进行练习。

1. 关于本实验的工作量、难度、deadline。

* 工作量与难度均适中；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？

* 我很喜欢软件构造这门课程，它为我打开了java世界的大门，并且教会了我如何开发出一个初具规模的项目，学到了许多有用的东西和先进又高效的设计理念；徐老师的教学也清晰易懂、引人入胜，感谢徐老师的《软件构造》课程！