

**2022年春季学期  
计算学部《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 乔浩宇 |
| 学号 | 120L021712 |
| 班号 | 2003005 |
| 电子邮件 | 2859295586@qq.com |
| 手机号码 | 17763511569 |

**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 1](#_Toc102826663)

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

[3.2.2 任务2：投票项VoteItem<C> 1](#_Toc102826665)

[3.2.3 任务3：选票Vote 2](#_Toc102826666)

[3.2.4 任务4：投票活动Poll<C>的测试 2](#_Toc102826667)

[3.2.5 任务5：投票活动Poll<C>的实现类GeneralPollImpl 2](#_Toc102826668)

[3.2.6 任务6：投票活动Poll<C>的子类型 2](#_Toc102826669)

[3.3 ADT行为的设计与实现 2](#_Toc102826670)

[3.3.1 任务7：合法性检查 2](#_Toc102826671)

[3.3.2 任务8：采用Strategy设计模式实现灵活的计票规则 2](#_Toc102826672)

[3.3.3 任务9：采用Strategy设计模式实现灵活的遴选规则 2](#_Toc102826673)

[3.3.4 任务10：处理匿名和实名投票 2](#_Toc102826674)

[3.3.5 任务11：采用Visitor设计模式实现功能扩展 2](#_Toc102826675)

[3.3.6 任务12：基于语法的数据读入 2](#_Toc102826676)

[3.4 任务13：应用设计与开发 2](#_Toc102826677)

[3.4.1 商业表决系统 2](#_Toc102826678)

[3.4.2 代表选举系统 2](#_Toc102826679)

[3.4.3 聚餐点菜系统 2](#_Toc102826680)

[3.5 任务14：应对面临的新变化 2](#_Toc102826681)

[3.5.1 商业表决应用：可以一次表决多个商业提案 2](#_Toc102826682)

[3.5.2 代表选举应用：遴选规则变化 3](#_Toc102826683)

[3.5.3 聚餐点菜应用：取消权重设置、只计算“喜欢”的票数 3](#_Toc102826684)

[3.6 Git仓库结构 3](#_Toc102826685)

[4 实验进度记录 3](#_Toc102826686)

[5 实验过程中遇到的困难与解决途径 3](#_Toc102826687)

[6 实验过程中收获的经验、教训、感想 4](#_Toc102826688)

[6.1 实验过程中收获的经验和教训（必答） 4](#_Toc102826689)

[6.2 针对以下方面的感受（必答） 4](#_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：

⚫ 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。

1. 建立本地仓库
2. 从远程仓库将源代码从GitHub上clone下来

在这里给出你的GitHub Lab3仓库的URL地址（HIT-Lab3-学号）。 <https://github.com/ComputerScienceHIT/HIT-Lab3-120L021712.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

简要介绍四个应用。

分析四个应用场景的异同，理解需求：它们在哪些方面有共性、哪些方面有差异。

共性：都是完成投票系统，共同具有的属性是name表示投票项目的名称,data表示项目日期,candidates表示候选人,voters表示投票者,quantity表示候选者最大数量,voteType表示种类...通过这些属性映射不同的功能

差异：具体要求有差异，比如商业投票候选者是商业提案选举人是股东，要求每个股东权重之和等于1；代表选举候选者是候选代表，支持匿名；聚餐点菜候选者是dish，quantity由点菜人数决定...

## ADT识别与设计

### 任务1：投票类型VoteType

设计思路：这个类是一个HashMap到投票类型的映射，key为投票项字符串，value为投票项对应的分数。这个类的方法有判断一个投票项是否合法，获得投票项对应的分数

实现过程：构造函数接受一个key为String，value为Integer的Map，这个Map作为内部属性options。检查合法性就是遍历options的所有key，如果任何一个都不匹配就返回false。获得对应分数也是遍历。需要重写equals和hashCode方法

### 任务2：投票项VoteItem<C>

实现思路：这个类表示的是一个投票人对一个候选人的投票，包含候选者和投票项，但不包含投票人。

实现过程：这个类的方法由获得投票项和获得候选人，因为都是不可变数据类型，不需要防御式拷贝，所以直接return。添加一个设置有效性的功能，在其他的类中调用。、

3.2.3任务3：选票Vote

实现思路：一个Vote对象就是一个投票者对所有候选者的投票，用HashSet voteItems映射一个投票者对所有候选对象的投票项集合，date表示日期。

实现过程：这个类有两个构造函数，一个形参是candidates，表示匿名投票，另一个形参是candidates和voter，表示不匿名投票

candidateIncluded(C candidate)方法遍历整个voteItems的元素，如果有和candidate相同的就返回true，否则返回true

**3.2.4任务4：投票活动Poll<C>的测试**

设计思路：创建一个引用类型为poll的对象，设置基本信息，添加候选人和投票者

实现过程：1.创建投票项目的基本信息

2.添加候选人，类型String

3.添加投票者

4.添加投票者对候选人的投票信息

5.按规则计票

**3.2.5任务5：投票活动Poll<C>的实现类GeneralPollImpl**

**实现过程：**

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

依次添加，比如：this.name = name;

addVoters(Map<Voter, Double> voters)和addCandidates(List<C> candidates)直接添加

### 

### 3.2.6任务6：投票活动Poll<C>的子类型

设计测试用例的策略：

先创建选举对象，创建投票项，添加候选人，然后增加投票人对每个候选人的选择，再依次按照统计策略和选择策略执行，最后输出结果

子类中重写的方法：

BusinessVoting的result方法：如果提案得分大于0.66则输出通过，否则输出未通过

Election中的addVote方法：要先检查选票中支持数量是否大于quantity，然后再进行父类中的检查和添加vote

result方法：按行输出被选择候选者的排名和名字

DinnerOrder中的result：按行输出被选择的菜名的排名

## ADT行为的设计与实现

### 任务7：合法性检查

对GeneralPollImpl类中的addVote方法进行如下条件检查：

一张选票中没有包含本次投票活动中的所有候选人

一张选票中包含了不在本次投票活动中的候选人

一张选票中出现了本次投票不允许的选项值

一张选票中有对同一个候选对象的多次投票

HashSet<C> cs = new HashSet<>();  
 for (VoteItem<C> voteItem : vote.getVoteItems()) {  
 cs.add(voteItem.getCandidate());  
 if(!candidates.contains((C)voteItem.getCandidate())){  
 voteItem.setValid(false);  
 return;  
 }  
 if (!this.voteType.checkLegality(voteItem.getVoteValue())) {  
 voteItem.setValid(false);  
 return;  
 }  
 }  
 if(cs.size()!=vote.getVoteItems().size())  
 return;  
 votes.add(vote);

此外对子类Election增加一张选票中对所有候选对象的支持票数量大于quantity的检查

int num=0;  
for(VoteItem<Person> vi:vote.getVoteItems()){  
 if(vi.getVoteValue().equals("支持"))  
 num++;  
}  
if(num>quantity)  
 return;

### 任务8：采用Strategy设计模式实现灵活的计票规则

以Dinner为例：先遍历整个votes，得到每个投票者对每个菜的喜好；然后遍历每张选票，得到每个投票项，在乘以权重得到对应分值，加到菜上，最终循环结束后得到每个菜的分值

for(Vote<Dish> vote:votes){//遍历整个votes，得到每一个投票者对所有候选者的选项  
 for(VoteItem<Dish> vi:vote.getVoteItems()){//得到每一个投票项

其他与dinner处理方式相似

### 任务9：采用Strategy设计模式实现灵活的遴选规则

以dinner为例，其他和这个情况相似

传入的形参有

Map<Dish, Double> statistics, Map<Dish, Double> results, int quantity

创建一个数组存放菜，一个数组存放得分，菜数组和得分数组的第i位是对应的

然后进行从大到小的排序

排序之后循环的方式将前quantity个菜和得分放入map result中，并且注意必须达到quantity并且没有重复得分才能停止循环：

for(int i=0;i< dishes.size();i++){  
 if(i>=quantity && scores.get(i)!=scores.get(i-1))  
 break;  
 results.put(dishes.get(i), (double)i+1);  
}

### 任务10：处理匿名和实名投票

第一种方式：添加Vote的子类RealNameVote，增加属性String name和方法getName，重写equals和hashCode方法，增加对name是否相同的判断

第二种方式：装饰器模式，定义接口VoteInterface，让Vote继承这个接口，并且定义抽象类VoteDeracotar，初步实现接口中的基本功能（不包含名字）,定义类VoteRealName继承这个抽象类，然后添加name的功能

### 任务11：采用Visitor设计模式实现功能扩展

在Poll和它的实现类中增加getLegalVotes的方法、getVotes的方法和accept方法，以获得合法选票的集合、所有选票的集合，并接受visitor。创建一个接口Visitor和实现这个接口的类VoteLegalVisitor，添加属性double data，实现visit方法和getData方法，用来计算合法选票的比例和返回data。

使用如下：

Visitor visitor = new VoteLegalVisitor();  
election.accept(visitor);  
double result = visitor.getData();  
System.*out*.println("选票中合法选票的比例是："+result);

### 任务12：基于语法的数据读入

按照需要匹配的字符串的要求得如下正则表达式：

"(\\“[\\u4e00-\\u9fa5]+\\”(\\(-?[0-9]+\\))?\\|)+\\“[\\u4e00-\\u9fa5]+\\”(\\(-?[0-9]+\\))?"

创建Pattern对象和Matcher对象，匹配传入的字符串regex

如果匹配成功，给相关属性赋值

如果匹配失败抛出相关异常

## 任务13：应用设计与开发

### 商业表决系统

创建投票活动

设定投票活动基本属性（名称、发起时间、投票类型等）

添加一个候选的商业提案

添加投票人（股东）及其权重

每个投票人对每个候选对象进行投票，汇集形成选票

将各投票人的选票陆续加入到投票活动

按计票规则进行计票

按遴选规则进行统计，得到投票结果

展示投票结果

### 代表选举系统

和3.4.1类似，只是候选者不只有一个，在本例中有三个候选者，该app其他步骤代码几乎无差别

### 聚餐点菜系统

和3.4.2几乎一样

## 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

checkRep需要修改，之前的断言是quantity等于1，把这条断言注释掉

如果发生了这种变化，那么三种应用的选择在算法上就没有区别了。

修改设计：按照其他两种应用的思路来修改商业表决应用的遴选规则。

### 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

变化不小，但可以接受，需要创建新类

如何修改设计以应对变化

创建 NewElectionSelection继承SelectionStrategy，构造器形参有一个legalVote，要求传入选举活动的有效选票，来统计每一个候选者被反对的选票数（放入属性Map<Person, Double> rejectStatistics中）

然后在select方法中做出的改变是：

if(i>=quantity && scores.get(i)==scores.get(i-1)){  
 if(rejectStatistics.get(persons.get(i)) > rejectStatistics.get(persons.get(i-1)))  
 results.put(persons.get(i+1), (double)i+1);  
 else if(rejectStatistics.get(persons.get(i)) < rejectStatistics.get(persons.get(i-1)))  
 results.put(persons.get(i),(double)i+1);  
 break;  
}

### 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

只需要重新写一个继承StatisticsStrategy的方法，把乘以权重的操作去掉就可以

如何修改设计以应对变化

将之前的获取分值的式子改为：

double n=voteType.getScoreByOption(vi.getVoteValue());

## 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 |
| 2022-6-1 | 18:20-20:50 | 获得源码，完成前五个任务 | 按计划完成 |
| 2022-6-22 | 14:30-16:05 | 完成前五个任务的修改，完成任务六 | 按计划完成 |
| 2022-6-22 | 16:10-20:00 | 完成任务七到任务九 | 未按计划完成，到当天十一点，仍有一部分问题没有解决 |
| 2022-6-23 | 16:40-18:40 | 完成任务8/9未完成的部分 | 按计划完成 |
| 2022-6-23 | 19:40-21:40 | 任务十到任务十二 | 任务十一没完成，任务十二测试不出来 |
| 2022-6-24 | 15:00-16:20 | 任务十三 | 提前二十分钟完成 |
| 2022-6-24 | 18:30-22:30 | 任务十四、实验报告、提交 | 实际大约零点完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 任务14的代表选举应用 | 创建 NewElectionSelection继承SelectionStrategy，构造器形参有一个legalVote，要求传入选举活动的有效选票 |
|  |  |
|  |  |

# 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）

lab3可以说是整个软件构造课的集大成者，难度和复杂度以及工作量都比前两个实验都高了不止一个层级，很考验综合能力。不仅能让我在考试结束之后回顾软件的各种原则，还复习了设计模式。其中也有一些复用和扩展，有的实现起来很顺手，有的很困难，很考验对大局的把握能力，有的地方不注意扩展性和复用性，就会导致做后续的任务遇到很大麻烦。

## 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？

面向应用场景编程会有更多的变化，和完全注重可复用性和可扩展性的ADT设计不同，由于灵活的变化，因此需要用到委托、设计模式等手段来根据具体的应用场景进行扩展或者修改。

体会到复用的好处，减少了很多重复代码的编写，而且更有利于理解各个类之间的关系。

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？

防止表示泄露是为了避免客户端依赖于内部属性或者恶意修改内部属性，我十分愿意以后的编程中这么做。

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？

能体会到，太能了。这样客户端只需要声明对象并调用方法，就可以完成比较复杂的工作，而客户端也不知道内部具体是怎么实现的。

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？

语法驱动编程很重要的一个环节就是正则表达式的匹配，因此学习形式语言与自动机这门课也不是完全没用的，它能帮助我们设计出正则表达式来判断输入进来的字符串是否匹配，非常有用。语法驱动编程，确实增强了程序和用户之间的交互型，让用户感受到了更大的自由度。

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？

我觉得如果完全由自己从零开始会有更大的挑战，虽然定义类并不困难，但是很有可能在之后的修改和扩展中遇到只有大幅度修改类才能解决的问题。我觉得设计ADT的难度主要在于怎么分配类之间的分工，来实现高内聚低耦合，怎么让他们更加具有扩展性

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？

从接口到抽象类到类，抽象程度越来越低，要解决的问题越来越具体，应该根据工作的具体程度来选择使用哪一种。要实现类层次的复用时，可以考虑接口的组合，实现对象层级的复用时，考虑使用委派，而继承则是复用父类的方法，设计模式是接口组合、委派和继承三者的综合应用，在适当的情况下选择合适的设计模式很重要。

1. 关于本实验的工作量、难度、deadline。

工作量比较大，本打算考试前完成实验，但由于内容偏难偏多，所以只完成了大概三分之一。考试结束后也是花了十几甚至二十几小时来完成剩余的三分之二。还好deadline是在考试后的一周，要不然真的会更加痛苦。

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？

我觉得软件构造相比于其他课程来说更加面向实践，而且实验占的分值也比其他课程高很多。在此前的代码编写过程中，没有把编码的规则当做很重要的东西，但是软件构造这门课把规则放到了空前的高度，让我意识到只有遵守好相应的规则才能写出质量高、运行起来错误少、修改起来更加方便高效的代码。