
哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

Lab Manuals for

# Software Construction

# Lab-3

# Reusability and Maintainability oriented Software Construction


Faculty of Computing

Harbin Institute of Technology

Spring 2022

# 目录

# 1 实验目标

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）。

# 2 实验环境

实验环境设置请参见 Lab-0 实验指南。

本次实验在 GitHub Classroom 中的 URL 地址为：

https://classroom.github.com/a/ZAJ8w2eC

请访问该 URL，按照提示建立自己的 Lab3 仓库并关联至自己的学号。

本地开发时，本次实验只需建立一个项目，统一向 GitHub 仓库提交。实验包含的多个任务分别在不同的包内开发，具体目录组织方式参见各任务最后一部分的说明。请务必遵循目录结构，以便于教师/TA 进行测试。

# 3 实验要求

本实验需要设计并实现抽象数据类型 `Poll`、`Vote` 和一组辅助 `ADT`，对现实中广泛存在的各类“投票”活动进行抽象，并在三个具体应用中使用它们。在设计 ADT 时，需要对其进行抽象和泛型化，从而使表达能力更强、适应现实中不同情况需求的能力更强。

考虑到你对 ADT 和 OOP 尚处于初学阶段，本次实验给你提供了基础代码框架，请沿着实验手册的要求，递进完成各个任务，在已有代码基础上进行扩展和修改。除非明确说明不得修改，否则你有自由对已有代码做出调整。代码框架可

从以下地址下载：

https://github.com/rainywang/Spring2022_HITCS_SC_Lab3

## 3.1 待开发的应用场景

要为以下场景开发 Java 程序，在设计和实现 ADT 的时候需充分考虑这些应用之间的相似性和差异性，使 ADT 有更大程度的复用和更容易面向各种变化。

(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（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。以下给出了针对某提案的表决结果示例：

表 1 商业表决的投票记录示意

| 董事 | 股权 | 投票 |
|---|---|---|
| $A$ | 5% | 反对 |
| $B$ | 51% | 支持 |
| $C$ | 10% | 支持 |
| $D$ | 14% | 反对 |
| $E$ | 20% | 弃权 |

那么，“支持”所占的比例为$51\% + 10\% = 61\% < 2/3$，故该提案表决未通过。

(2) **代表选举（Election）**：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$k$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——这意味着最终选出的人数可能小于$k$。

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：

表 2 代表选举的投票记录示意

| 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 |

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。

(3) **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。

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：

表 3 聚餐点菜的投票记录示意

| 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$4\times1+1\times2+2\times0+2\times2=10$。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。

## 3.2 面向可复用性和可维护性的需求分析

考虑上节给出的三个应用，其中都包含了各种各样的“一组投票人针对一组候选对象的投票、评价、打分”的现象，虽然评价对象的类型有差异、评价机制有差异、评价规则有差异，但仍存在很多共性。

为了提高软件构造的可复用性和可维护性，可为其设计和构造一套统一的 ADT，需仔细分析它们的共性和差异。

### 3.2.1 共性概念

表 4 三个应用中的共性概念

| 概念 | 含义 | 商业表决 | 代表选举 | 聚餐点菜 |
|---|---|---|---|---|
| 投票活动 | 一组投票人针对一组候选对象的投票、评价或打分 | 股东表决某个商业提案 | 学生选举大会代表 | 一家人聚餐点菜 |
| 候选对象 | 被选择/被评价/被打分的具体事物 | 商业提案 | 被选举的学生 | 菜品 |
| 投票人 | 根据主观意愿对各候选对象进行投票、评价或打分的人 | 股东 | 学生 | 就餐者 |
| 投票项 | 一个投票人对一个候选对象的具体投票信息 | 一个股东对一个商业提案的表决票 | 一个学生对一个被选举人的选票 | 一个人对某个菜品的偏好 |
| 选票 | 一个投票人对所有候选对象的投票项的聚合 | 一个股东对所有商业提案的表决票 | 一个学生对所有被选举人的投票 | 一个人对所有菜品的偏好 |

### 3.2.2 多个维度上的差异

除了上述共性的概念，实际应用中还会有多个维度上的差异：

表 5 三个应用体现差异的维度清单

| 维度 | 解释说明 |
|---|---|
| 候选对象的数量$m$ | 有多少个候选对象在一次投票活动中被投票？ |
| 拟遴选出的候选对象数量$k$ | 从所有候选对象中选择出多少个符合规则的候选对象作为最终结果？ |
| 投票类型 | 例如“支持、反对、弃权”、“ 喜欢、不喜欢、无所谓”，并分别映射到具体的数值。例如在上述应用中，“支持、反对、弃权”被量化为1、−1、0，“喜欢、不喜欢、无所谓”被量化为2、0、1。 |
| 实名与否 | 投票记录上是否记录投票人的身份信息？可以是实名，也可以是匿名。 |
| 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。 |
| 检查合法性 | 每张选票是否合法？以下为非法选票：<br>● 一张选票中没有包含本次投票活动中的所有候选人<br>● 一张选票中包含了不在本次投票活动中的候选人<br>● 一张选票中出现了本次投票不允许的投票选项<br>● 一张选票中有对同一个候选对象的多次投票项<br>所有选票整体是否合法？<br>● 若尚有投票人还未投票，则不能开始计票<br>● 若一个投票人提交了多次选票，则它们均为非法，计票时 |

| | |
|---|---|
| | 不计算在内 |
| **计票规则** | 如何将所有投票人的选票综合到一起，计算出每个候选对象的得票数或得分，对所有候选人进行排序？ |
| **遴选规则** | 根据何种规则从所有候选对象中选择出一个子集作为最终投票结果？ |

下图展示了一个示例说明：


| 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 |

图 1 相关概念的示意图

下表给出了三个具体应用在上述维度上的差异性说明：

表 6 三个应用在各维度上的异同

| | 商业表决 | 代表选举 | 聚餐点菜 |
|---|---|---|---|
| 候选对象类型 | 商业提案（例如“关于投资 xx 项目的提案”） | 有资格参选的学生（候选人） | 菜品 |
| 候选对象数量$m$ | 1 | $\geq 1$ | $\geq 1$ |
| 拟遴选出的候选对象数量$k$ | 0（表决未通过）或者 1（表决通过） | 提前确定，$k \leq m$ | 提前确定，$n \leq k \leq n+5 \leq m$，$n$为投票人数量 |
| 投票人类型 | 股东 | 学生 | 就餐者 |
| 投票人数量$n$ | 提前确定 | 提前确定 | 提前确定 |
| 投票类型 | 支持/反对/弃权（可看作$1/-1/0$分） | 支持/反对/弃权（可看作$1/-1/0$分） | 喜欢/不喜欢/无所谓（可看作$2/0/1$ 分） |
| 实名与否 | 实名 | 匿名 | 实名 |
| 一个投票人在投票时，不同投票选项的数量限制 | 无限制 | 支持票数量$\leq k$ | 无限制 |
| 投票人权重 | 有，取决于股东所占的股权大小 | 无，所有投票人的权重一样 | 有，就餐者的不同身份具有不同的影响力 |
| 检查合法性 | 表 5 中给出的合法性检查条件 | 除了表 5 中给出的合法性检查条件，还需要检查一个投票人投的支持票总数是否小于等于$k$ | 表 5 中给出的合法性检查条件 |
| 计票规则 | 考虑投票人权重，计算提案得“支持”票的比率 | 计算每个候选人得到的“支持”票总数 | 根据各就餐者的权重，加权求和计算得到每个菜品的总得分 |
| 遴选规则 | 若超过$2/3$支持票，则提案通过；否则，提案不通过 | 根据支持票数量排序，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 | 根据支持票数量排序，取前$k$个菜品，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。 |

### 3.2.3 对差异性的分析

上一节表格里列出的三个应用在各维度上的差异，有些是可以用统一的方式去处理，有些则无法一致化，需要通过子类型多态或设计模式等手段加以处理。下表给出了相关说明，供设计时参考。请借鉴课程中关于继承、重载、CRP 和设计模式的内容，做出最恰当的设计。

表 7 对各维度差异性是否可一致化设计的分析

| 维度 | 建议的 ADT 设计决策（供参考） | 对应章节 |
|---|---|---|
| 候选对象类型 | 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） | 3.3.2、3.3.4<br>3.3.6、3.3.7 |
| 候选对象数量 $m$ | “= 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” | 3.3.6、3.3.7 |
| 拟遴选出的候选对象数量 $k$ | 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。 | 3.3.6、3.3.7 |
| 投票人类型 | 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。 | 3.3.2 |
| 投票类型 | 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。 | 3.3.3、3.3.4 |
| 实名与否 | “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。 | 3.3.5、3.4.4 |
| 不同投票类型的数量限制 | 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。 | 3.4.1 |
| 投票人权重 | 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 | 3.3.6、3.3.7<br>3.4.2 |

| | | |
|---|---|---|
| | 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** | 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。 | 3.4.1 |
| **计票规则** | 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。 | 3.4.2 |
| **遴选规则** | 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。 | 3.4.3 |

## 3.3 ADT 识别与设计

### 3.3.1 ADT 识别

按照上一节的需求描述，从中抽取关键的“名词”，识别 ADT。以下是抽象出来的 ADT：

(1) 投票活动 `Poll<C>`：代表一次投票活动，`C` 是表征该投票活动中的“候选对象”类型的泛型参数。这是本实验中最大的 ADT。

(2) `GeneralPollImpl<C>`：是对 `Poll<C>`的一个实现类。

(3) 投票记录 `VoteItem<C>`：代表一个投票人针对一个候选对象的投票结果。

(4) 选票 `Vote`，分为匿名选票和实名选票两类：代表一个投票人对所有候选对象的投票结果，由多个投票记录构成。

(5) 投票人 `Voter`

(6) 投票类型 `VoteType`，用于描述在一次投票活动中，投票人对候选对象可能投出的选票类型。例如在商业表决应用中，投票类型包括“支持、反对、弃权”，分别对应于 1、0、0 分。在构造投票活动时，需要指定其采用的投票类型。

(7) 候选对象：对应于 `C`，在三个应用中分别需要构造以下 ADT：

a) 提案 `Proposal`

b) 候选人 `Person`

c) 菜品 `Dish`

实验代码中已经给出了上述 ADT 的代码框架，请仔细阅读并理解。为辅助理解，下图的 class diagram 描述了这些 ADT 之间的关系。

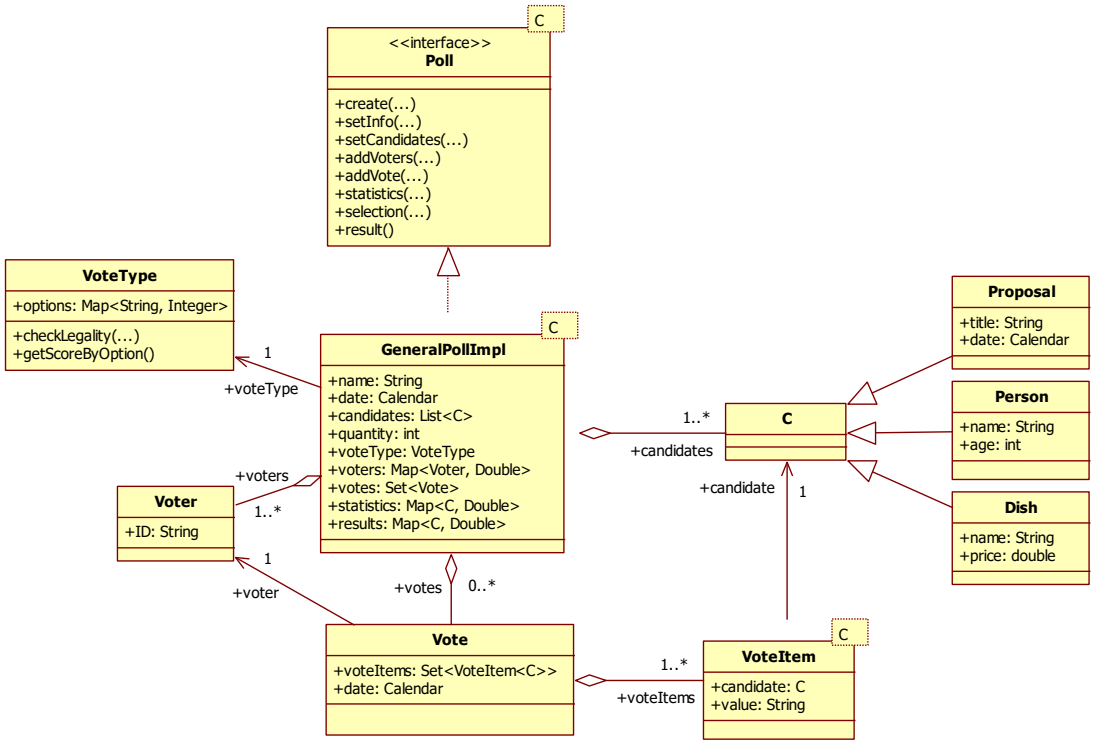


图 2 描述 ADT 之间关系的类图

以下各节，将按照从简单到复杂的次序，在已给出的部分代码框架基础上，逐步引导你构建和完善可复用的 ADT。

### 3.3.2 辅助 ADT

(1) 投票人 `Voter`：只需要能唯一标识身份的 ID 即可。似乎为了一个简单的 ID 无需设计一个 ADT、只需要用 `String` 表示即可。但考虑将来的扩展性，还是设计为一个独立的 ADT。

(2) 候选对象（泛型参数 `C`）：无需抽象，设计三个具体类：

- 提案 `Proposal`：提案名称、提案日期
- 候选人 `Person`：姓名、年龄
- 菜品 `Dish`：名字、价格

这些 ADT 都是 Immutable 的，其代码已经给出（见 `auxiliary` 包），请阅读，无需做任何修改。

### 3.3.3 投票类型 `VoteType`

该 ADT 用于管理一次投票活动中允许使用的合法投票选项，是 Immutable 的。例如在三个应用中的投票选项分别为：

- 支持/反对/弃权：对应于1、−1、0
- 支持/反对/弃权：对应于1、−1、0
- 喜欢/不喜欢/无所谓：对应于2、0、1

在实验所给的代码中（见 `vote` 包），该 ADT 使用 `Map<String, Integer> options` 作为其 rep，其中 key 为选项名、value 为选项名对应的分数。不要更改该 rep。

【任务1】请根据 spec 设计各个方法的测试用例，在 `VoteTypeTest.java` 中完成测试代码。撰写 AF、RI、Safety from rep exposure，完成相关方法的内部代码，具体要求参见代码中相关注释。如果在完成后续各项任务时，需要在 `VoteType` 中补充相关方法，可以自行补充，但需要补全其 spec 和代码。

### 3.3.4 投票项 `VoteItem<C>`

`VoteItem<C>`是一个 immutable 的 ADT，描述了一个“投票项”，表征了一个投票人对一个候选对象的具体评价。该 ADT 内部的数据和行为包括：

【数据】候选对象（类型为 `C`）

【数据】投票选项（例如“支持”）。

【行为】创建投票记录

注意：该 ADT 一定是 immutable 的，意味着一旦生成就不能再修改。该 ADT 的代码见 `vote` 包，代码中已经给出了 Rep，请不要更改。

【任务2】请根据 spec 设计各个方法的测试用例，在 `VoteItemTest.java` 中完成测试代码。撰写 AF、RI、Safety from rep exposure，补全相关方法的代码。如果在完成后续各项任务时，需要在 `VoteItem<C>`中补充相关方法，可以自行补充，但需要补全其 spec 和代码。

### 3.3.5 选票 `Vote`

这是代表一个投票人对所有候选对象的投票项的聚合体，是 Immutable 的。该 ADT 内部应维护的数据和行为包括：

【数据】对所有候选对象的投票记录（相当于多个 `VoteItem` 对象）

【数据】投票时间

【行为】创建选票

【行为】查询该选票中包含的所有投票项

该 ADT 的代码见 `vote` 包。代码中已经给出了 Rep，请不要更改。

【任务3】请根据 spec 设计各个方法的测试用例，在 `VoteTest.java` 中完成测试代码。撰写 AF、RI、Safety from rep exposure，补充相关方法的代码。如

果在完成后续各项任务时，需要在 Vote 中补充相关方法，可以自行补充，但需要补全其 spec 和代码。

### 3.3.6 投票活动 Poll<C>

Poll<C>是一个 mutable 的 ADT，设计为一个接口，管理一次投票活动的基本设定、投票记录、遴选结果等。C 是表征“候选对象”的泛型参数。

对 Poll<C>及其子类型来说，包含以下接口行为：

【行为】创建投票活动 create()：返回一个 Poll<C>对象，但尚未设定任何属性，也未包含任何投票数据。

【行为】设定投票活动的基本属性 setInfo()：将投票活动的名称、发起时间、投票类型、拟选出的候选对象数量等信息赋予一个 Poll<C>对象。

【行为】添加候选人 addCandidates()：将一个候选对象集合加入 Poll<C>对象。客户端可以多次调用该方法添加多次候选人。

【行为】添加投票人及其权重 addVoters()：将投票人集合及各自权重传入 Poll<C>对象。客户端可以多次调用该方法添加多次投票人。

【行为】接收一个投票人的选票 addVote()：将 Vote 对象传入 Poll<C>对象。

【行为】按规则计票 statistics()：首先判定选票的合法性，然后基于各个合法选票，按本次投票活动的规则进行计票，计算每个候选人的得分。

【行为】按规则遴选 selection()：基于计票结果，按本次投票活动的规则进行遴选，从中选出$k$个候选人作为最终结果。

【行为】展示投票结果 result()：将投票结果转为一个字符串，便于客户端对外呈现。

各接口方法的 spec 见 poll 包中的 Poll.java，任务过程中不能更改这些 spec。

【任务4】请根据 spec 设计各个方法的测试用例，在 PollTest.java 中完成测试代码。此时可暂不用考虑各应用所需要的 Poll 子类型。

GeneralPollImpl<C>是 Poll<C>的一个实现类，见 poll 包中的代码，其中已给出相应的 rep，其中包含的数据包括：

【数据】投票活动的名称

【数据】发起时间

【数据】候选对象的集合（例如 List<C> candidates）

【数据】投票人集合及各自的权重：本次活动由哪些投票人（类型为 Voter）、每个投票人的权重（类型为 Double）

【数据】拟遴选出的候选人数量$k$

【数据】本次投票可使用的投票类型（VoteType）

【数据】投票（即所有投票人的选票 Vote 的集合）

【数据】投票结果统计（根据投票结果，统计出所有候选对象的得分）

【数据】计票结果（根据规则和投票结果统计，从中选出$k$个对象作为结果）

【任务5】编写 RI、AF、Safety from Rep Exposure、checkRep()、toString()等，完成对各个方法的内部逻辑的编码。方法若有需要抛出的异常，可自定义异常类并在方法的 spec 中使用。请在给定的 Rep 和接口方法 spec 基础上编程。也可以根据自己的设计需要，对这个 rep 加以调整（增加、修改或删除），但要说清楚依据。在完成这些方法的代码时，可以结合 3.4 节的各项要求一起考虑。允许在当前设计的方法基础之上，增加其他的辅助性方法。代码完成后，执行测试用例，确保所有测试用例通过。

### 3.3.7 投票活动 Poll<C>的子类型

Poll<C>接口中定义的方法应当是 ADT 所有实例对象都统一具有的“行为”。如果某方法对某些实例对象是有价值的、对其他实例对象是不需要的“行为”，那么不建议放在接口里。请仔细分析三个应用在各维度上的异同，评估目前给出的各个方法是否都是“统一具有的行为”，进而进行子类型设计。

三个应用的子类型为：

- BusinessVoting：代表针对某个商业提案的一次表决
- Election：代表一次选举活动
- DinnerOrder：代表一次聚餐点菜活动

【任务6】针对从 Poll<C>出发所派生出的所有 ADT 子类型，目前没有给出任何代码，留给你自由发挥（见 poll 包），但注意从表 6、表 7 中借鉴有用的信息，使你的这些子类型能充分体现各应用的差异。三个应用的共性功能在 GeneralPollImpl 的方法中实现，各自的个性化功能在各子类中方法中通过 override 机制来实现。为这些子类撰写 AF、RI、Safety from rep exposure，以及每个方法的 spec、实现代码。为它们设计和编写 JUnit 测试用例：

- BusinessVotingTest.java
- ElectionTest.java
- DinnerOrderTest.java

若有必要，你也可以增加新的辅助 ADT。

## 3.4 ADT 行为的设计与实现

### 3.4.1 合法性检查

`Poll<C>`对象在接收外部客户端传递进来的一张选票时，要对其进行合法性检查，具体在 `addVote()`方法实现。以下是合法性检查时应检查的非法内容：

- 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
- （仅应用 2）一张选票中对所有候选对象的支持票数量大于$k$。

如果有上述情况出现，则该选票为不合法，需要做出标记，后续计票时不计算在内（注意：不是拒绝该选票）。为了记录一张选票是否合法，需要对 `GeneralPollImpl<C>`的 rep 做相应扩展或修改，请自行完成。特别的，针对上面最后一项中针对应用 2 的合法性条件，应考虑在应用 2 的子类型中相应方法加以检查。

说明：似乎会认为“选票的合法性难道不应该在 `Vote` 中加以检查吗”？仔细思考一下就会发现，一个 `Vote` 对象内部并不掌握某次投票活动的候选人信息、选票类型信息、允许的支持票数量信息，无法做出判定，因此要在 `Poll` 中进行检查。

在进行计票之前，还需要检查以下内容，具体在 `Poll` 的 `statistics()`方法中实现：

- 若尚有投票人还未投票，则不能开始计票；
- 若一个投票人提交了多次选票，则它们均为非法，计票时不计算在内。

【任务7】请在 `GeneralPollImpl<C>`以及三个应用子类型的两个方法的相应位置实现上述两种合法性检查的具体代码。必要的话，可以将合法性检查的功能 delegate 到单独的 ADT 去实现，从而支持未来可能出现的各种新的合法性检查条件（若你如此做，需要对 `addVote()`和 `statistics()`的 spec 做调整）。

### 3.4.2 计票规则

考虑到不同应用所使用的计票规则有显著差异，并考虑到未来可能出现的其他应用中会引入新的计票规则，请使用 Strategy 设计模式改造你的设计，使 ADT 可以支持不同的计票规则。

- 应用 1：统计获得支持票的数量；

- 应用 2：统计获得支持票的数量；
- 应用 3：统计每个菜品的总得分（加权求和）；

【任务8】在 `Poll<C>`的代码中，`statistics()`方法有一个参数 `ss`，其类型为 `StatisticsStrategy`，这是一个接口，请在此基础上使用 Strategy 设计模式，为三个应用分别构建 `StatisticsStrategy` 的子类型，分别完成三种不同的计票规则。在三个应用的客户端程序中，传入特定的 `StatisticsStrategy` 子类型对象，即可执行不同的计票方法。这本质上是一种 delegation 机制。

### 3.4.3 遴选规则

考虑到不同应用所使用的遴选规则有显著差异，并考虑到未来可能出现的其他应用中会引入新的遴选规则，请使用 Strategy 模式改造你的设计，使 ADT 可以支持不同的遴选规则。

- 应用 1：获得支持票超过合法选票的2/3，即表示表决通过；
- 应用 2：选择排名前$k$的候选人，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；
- 应用 3：选择排名前$k$的菜，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。

【任务9】在 `Poll` 的代码中，`selection()`方法有一个参数 `es`，其类型为 `SelectionStrategy`，这也是一个接口。同样的，在此基础上使用 Strategy 设计模式，为三个应用分别构建 `SelectionStrategy` 的子类型，分别完成三种不同的遴选规则。在三个应用的客户端程序中，通过传入特定的 `SelectionStrategy` 子类型对象，即可执行不同的遴选方法。这本质上是一种 delegation 机制。

### 3.4.4 匿名和实名投票

应用 1 和应用 3 是实名投票，而应用 2 是匿名投票。前面给出的 `Vote` 的 rep 中没有出现投票人信息，因此它是支持匿名投票的 ADT。为了支持实名投票，需要在选票 ADT 中额外记录投票人信息。具体方法有二：

- 构造 `Vote` 的子类型 `RealNameVote<C>`（见 `vote` 包）
- 使用 Decorator 设计模式，在 `Vote` 基础上进行扩展。代码中没有给出相应的类，需要自行设计。

【任务10】请实现这两种方式，并在三个应用的客户端程序中加以演示。

### 3.4.5 对功能的灵活扩展

【任务11】考虑到将来的投票应用可能出现更多的对投票结果的处理，请在 ADT 设计中引入 Visitor 设计模式，预留好接口扩展新功能。基于该设计模式，实现一个“计算合法选票在所有选票中所占比例”的扩展功能。在 `Poll` 目前代码基础上进行扩展，实现上述设计模式和扩展功能。

### 3.4.6 基于语法的数据读入

【任务12】你在 3.3.3 节中已经实现了 `VoteType` 的构造方法。在这里补充完成另一个构造方法 `public VoteType(String regex)`，它输入一个遵循特定语法的字符串，构造函数解析该字符串，进而构造出 `VoteType` 对象。该字符串的语法规则如下：

“喜欢”(2)|“不喜欢”(0)|“无所谓”(1)

其中，用双引号括起来的文字部分是一个投票选项，长度不超过 5，其中不允许出现空格；用括号括起来的数字是投票选项对应的分数，可以是正整数、0 或负整数，不能带小数，正整数不需要使用“+”，但负整数需要使用“-”；不同的投票选项之间用“|”隔开。

也可以用如下形式：

“支持”|“反对”|“弃权”

与上面的例子相比，区别是没有分数。这种情况表明各个投票选项的权重是一样的。

请自行设计覆盖各种符合和不符合上述语法规则的测试用例，对这个新的构造方法 `VoteType(String regex)`方法进行完备的测试，测试代码补充在 `VoteTypeTest.java` 中。

## 3.5 应用设计与实现

利用前面所开发的 ADT，为三个应用场景分别开发程序（见 `app` 包）：

- `BusinessVotingApp.java`
- `ElectionApp.java`
- `DinnerOrderApp.java`

【任务13】在各自的 `main()`中模拟以下行为：

- 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）
- 添加一个或多个候选对象

- 添加投票人及其权重
- 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

在 `ElectionApp` 中给出了一个简单的示例，仅供参考，但不完整。你可自由安排执行次序，模拟现实中各种可能。这几个程序作为你前面所开发的 ADT 的客户端，验证 ADT 的复用性和适应变化的能力。无需真实用户手工操作的 GUI 或命令行界面，在 `main()`中撰写代码即可。

## 3.6 新的变化

【任务14】前面开发出的各应用，面临着以下的几个功能变化。请考查原有的设计，是否能以较小的代价适应这些变化。修改原有设计，使之能够应对这些变化。在修改之前，请确保你之前的开发已经 commit 到 Git 仓库，然后创建新分支“change”，在该分支上完成本节任务。

- 商业表决应用：可以一次表决多个商业提案；
- 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。
- 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。

在具体修改之前，请根据前面的 ADT 与应用设计，先大致估算一下你的程序需要变化多大才能适应这些变化。修改之后，再相对精确的度量一下你在该节之前所做的 ADT 设计以多大的代价适应了该表中的每一个变化。这里的“代价”可用“代码修改的量”和“修改所耗费的时间”加以估计。

注意 1：除非这些变化修改了之前各节提及的某些功能，你针对这些变化所做的修改都不应破坏之前已经写好的功能（考虑 OCP 原则）。

注意 2：提交到仓库之后，master 分支需仍然指向本节任务之前的结果，TA 会到仓库的 change 分支上检查本节任务。

以下给出了 Git 的相关操作指南。

```
…最初在 master 上工作…
git checkout -b change 创建新分支
…按上面的要求进行代码修改…
```

```
git add *
git commit -m "change"在该分支上提交
git checkout master                    切换回 master 分支
…请不要使用 git merge change 进行合并修改
…请不要使用 git branch -d change 删除分支
```

你的 Git 仓库中的 Object Graph 应类似于下图所示。其中上方的 master 分支包含了本节之前的开发结果，下方的分支是针对本节的变化需求的开发结果。

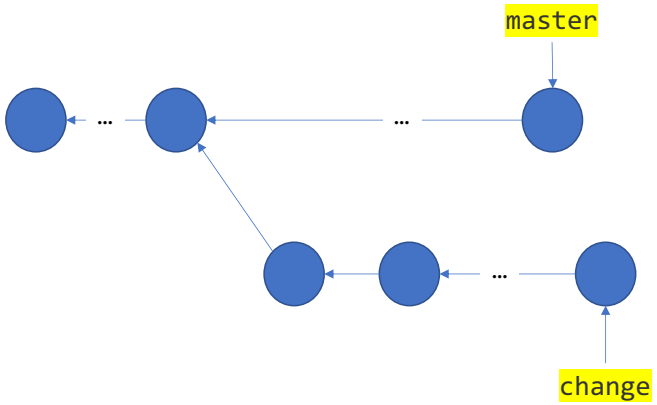


图 3 Git 仓库内的分支结构

## 3.7 项目结构

| | | |
|---|---|---|
| 项目名：HIT-Lab3-学号 | | |
| | src | java 文件，自行组织包结构，做到有序、清晰 |
| | test | JUnit 测试程序目录，与 src 结构保持一致 |
| | lib | 程序所使用的所有外部库文件 |
| | doc | 实验报告 |
| | 其他辅助目录 | |

注：如果手册中给出的接口/类无法满足你的设计需要，请自行增加相应的包、接口、类，但不要更改上述包结构。

# 4 实验报告

针对上述编程题目，请遵循**报告模板**，撰写简明扼要的实验报告。

**实验报告的目的是记录你的实验过程，尤其是遇到的困难与解决的途径。**不需要长篇累牍，记录关键要点即可，但需确保报告覆盖了本次实验所有开发任务。

注意：

- 实验报告不需要包含所有源代码，请根据上述目的有选择的加入关键源代码，作为辅助说明。
- 请确保报告格式清晰、一致，故请遵循目前模板里设置的字体、字号、

行间距、缩进；

- 实验报告提交前，请“目录”上右击，然后选择“更新域”，以确保你的目录标题/页码与正文相对应。
- 实验报告文件可采用 Word 或 PDF 格式，命名规则：Lab3-学号-Report。

# 5 提交方式

**截止日期：**夏季学期第 1 周周日（6 月 26 日）夜间 23:55。
**源代码：**从本地 Git 仓库推送至个人 GitHub 的 Lab3 仓库内。
**实验报告：**随代码仓库（doc）目录提交至 GitHub。

# 6 评分方式

Deadline 之后，教师和 TA 对学生在 GitHub 上的代码进行测试，阅读实验报告，做出相应评分。