主管领导审核签字

哈尔滨工业大学 2021 学年春季学期

软件构造　　试　题

题号	一	二	三	四	五	六	七	八	九	总分
得分										
阅卷人										

片纸鉴心　诚信不败

密　封　线

授课教师＿＿＿＿　姓名＿＿＿＿　学号＿＿＿＿　院系＿＿＿＿

本试卷满分 100 分，按 60%计入总成绩。

一 单项选择题（每题 3 分，共 45 分。请将答案填入下表，否则不计分）

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	

1. 以下属于“内部质量属性”的是___
A 可复用性　　B 健壮性　　C 可扩展性　　D 代码可读性

2. 一个 Java 程序在执行过程中抛出了异常，运行界面上输出了发生异常的方法及其所在代码行、逐层调用该方法的其他方法及其代码行。考虑这些信息在多维度软件构造视图中所处的位置，以下正确的是____
A Component level, Run-time　　B Code level, Build-time
C Period, Code-level　　D Moment, Component level

3. 针对 Git 与软件配置管理系统，以下说法不恰当的是___
A Git 是一种分布式配置管理系统，有本地仓库和远程仓库
B `git commit -m "msg"`将本地工作目录中相对于上一次提交修改过的文件提交至本地仓库
C 不同 commit 中存储的文件指针，可能有指向磁盘上同一个文件的情况
D 将两个分支合并时，若存在冲突，需要人工介入，消除冲突后再试着合并

4. 以下关于 ADT 的说法，正确的是___
A 一个 mutable 的 ADT，因为其 rep 值可变，存在表示泄露也不会造成危害
B ADT 某个方法的返回值类型若不是 `void`，那么它不会是 mutator 方法
C RI 规定了 rep 所必须满足的约束条件，且在任何时刻都需保持为真

D　两个 ADT 具有相同 rep 和相同的 RI，那么在客户端程序员眼里它们是等价的

5. 以下代码段能够通过 dynamic checking 的是____

A	`final StringBuilder sb = new StringBuilder("abc");`
	`sb.append("d");`
	`sb = new StringBuilder("e");`

B	`List<String> strs = new ArrayList<>();`
	`strs.add("HIT");`
	`for(String s : strs)`
	`  if(s.startsWith("HIT"))`
	`  strs.remove(s);`

C	`List<String> list = new ArrayList<>();`
	`List<String> copy= Collections.unmodifiableList(list);`
	`copy.add("c");`

D	`String text = "hello" + new String("world");` `text = text + false + 1 + 0.1;`

6. 以下代码段中，返回结果为 true 的是____

A	`List<String> lst1 = new ArrayList<>();`
	`List<String> lst2 = new LinkedList<>();`
	`lst1.add("a");`
	`lst2.add(new String("a"));`
	`return lst2.equals(lst1);`

B	`String s1 = new String("abc");`
	`List<String> list = new ArrayList<String>();`
	`list.add(s1);`
	`s1 = s1.concat("d");`
	`return list.get(0).equals("abcd");`

C	`StringBuilder sb1 = new StringBuilder("ABC");`
	`StringBuilder sb2 = new StringBuilder("ABC");`
	`return sb1.equals(sb2);`

D	`Number a = new Double(3.0);`
	`Number b = new Integer(0);`
	`a = b;`
	`return a instanceof Double;`

7. 以下关于 overload 的说法，正确的是___

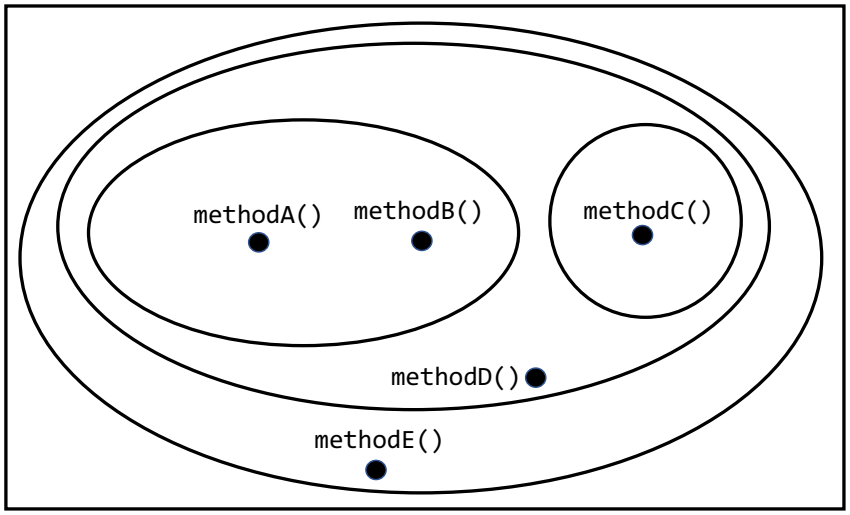
A　overload 是 OOP 中一种典型的参数化多态机制

B　两个 overload 的方法，要有不同的参数数目

C　两个 overload 的方法，返回值类型要遵循“协变”原则

D　父类型 A 和子类型 B 中可以存在 overload 的方法

8. 阅读下图的 specification diagram，其中黑点代表一个具体的方法实现，以下说法不正确的是___



A methodA()和 methodB()的 spec 对客户端程序员来说没有区别
B methodC()的 spec 的强度比 methodD()的 spec 的强度要大
C 若遵循 methodD()的 spec，客户端程序员可以无区别的使用 methodA()和 methodC()
D 在任何可以使用 methodD()的场合，都可以使用 methodE()而不会造成副作用

9. 关于以下 Java 代码段的说法，正确的是___

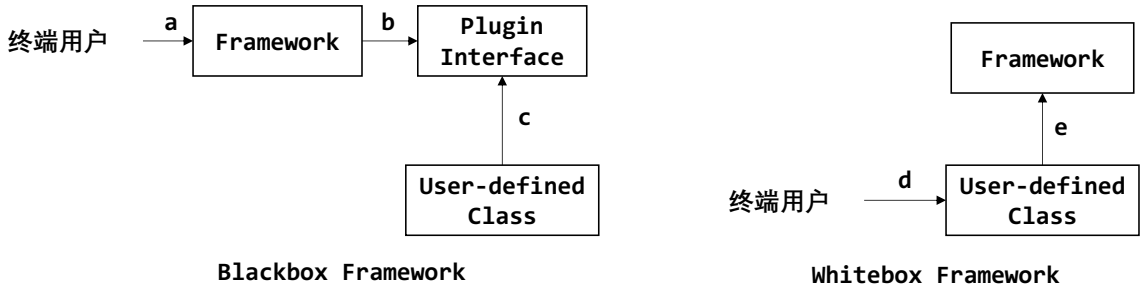
```
1  public interface A extends B {
2      public static A m() {...;}
3      List<Object> n(Number a);
4  }

5  public class C extends D implements A,E {
6      @Override
7      public List<String> n(Number d) {
8          ...
9      }

10     @Override
11     public boolean equals(B b) {...}
12 }
```

A 第 3 行最前面需增加 public 修饰符，否则无法通过静态检查
B 若把第 10 行删除，则第 11 行可以通过静态检查
C Java 不允许多重继承，故第 5 行 implements 之后不能同时出现 A 和 E
D 类 C 中的方法 n(..)是对接口 A 中方法 n(..)的合法 override，符合 LSP 原则

10. 以下关于 hashCode()的说法，最恰当的是___
A 如果不需要把 ADT 对象放入集合类中使用，那么该 ADT 无需重写 hashCode()
B 如果两个 ADT 对象不等价，那么它们的 hashCode()返回结果不应该相等
C 如果父类已重写了 hashCode()、子类的 rep 中没有引入新的属性且没有重写 equals(..)，那么子类中无需重写 hashCode()
D 在 hashCode()内部生成 hash code 时，要考虑 ADT 的 rep 中包含的所有属性

11. 以下关于 inheritance 和 delegation 的说法，不恰当的是___
    A 二者都可支持代码的复用
    B 如果类 A 和类 B 的功能 90%是一样的，但 B 包含了小部分个性化功能，A 中无个性化功能，那么可把 B 设计为 A 的子类
    C 如果类 A 通过 delegation 复用类 B 的功能，那么 A 的 rep 中必然有一个属性，其类型为 B，用于持久化的 delegation 关系，A 中的方法可通过该属性调用 B 的方法
    D inheritance 是发生在 class 层面，而 delegation 是发生在 object 层面

12. 图中表示 delegation 关系的是____



A a　　B b　　C c　　D d　　E e

13. 关于 visitor 模式，不恰当的是___
    A 该模式将一个 ADT 中定义的某个有多种实现方式的方法从 ADT 中抽取出来，单独放在 Visitor 接口及实现类中，以便于将来灵活复用
    B 给原 ADT 传递进不同的 visitor 实现类的实例，即可在无需改变原 ADT 的代码的情况下对 ADT 对象做不同的操作
    C 被 visit 的 ADT，需要提供 accept(Visitor v)这样的方法，便于将一个外部 visitor 实例传递进来
    D 被 visit 的 ADT，可以提供诸如 accept(Visitor v1, Visitor v2)的方法，即可支持对 ADT 做两个外部操作的组合

14. 以下关于代码健壮性和正确性的说法，不正确或不恰当的是____
    A 客户端程序调用方法 m()的时候可能违反 m()的 pre-condition，因此在 m()的最开始最好进行 pre-condition 的检查，以便于 fail fast
    B 满足 LSP 的父类型 A 和子类型 B，A 中有个方法 public void m() throws Exception，B 中对 m()进行重写时可以不抛出任何异常
    C 如果某方法实现中包含了 throw new xxException(...)的代码，则该方法的 spec 中一定会包含 throws xxException
    D 为 ADT 的某方法设计测试用例时，不应该使用任何与 ADT 的 rep 和方法实现相关的信息

15. 以下关于 assert 和 exception 的做法，最不恰当的是_____
    A 对 Java 中的 error 和 unchecked 异常，编程时最好不要用 try/catch 的方式来捕获
    B 一个方法根据传入的磁盘路径读入一个文件，该方法内部首先应该检查文件是否存在，若不存在，则用 assert false 语句终止程序执行
    C 在一个方法的最开始，最好应检查 pre-condition 是否满足，若不满足，用“抛出异常”的方式提示用户比用 assert 语句更合适
    D 在一个方法的 return 语句之前（若无 return 则在方法最后），应检查 post-condition 是否满足，若不满足，用 assert 语句比用“抛出异常”的方式更合适

应用设计题（55 分）

请从以下第二题至第九题中选择 55 分的题目作答即可。教师按从前往后的次序对前 55 分的已答题目进行评分。只要答题区有答案，即认为该题已选，请务必谨慎！

某客户需要开发一个“投票”系统，针对一组候选人和一组投票者，投票者按照特定规则为候选人投票，系统管理所有投票，并做统计分析。具体而言：

创建一次新的投票活动（`Poll`），需要指定一组候选人（`candidates`），并设定每个投票者允许投“支持票”的最大人数 `maxSupportNum`。

每个投票者对这组候选人进行投票。一张选票（`Vote`）上包含了投票者对每个候选人的投票（支持、反对、弃权）。需要检查每张选票的合法性，如果选票中支持的人数超过了 `maxSupportNum`，则为无效选票。

当所有投票者都已投票，可计算出每个候选人的总赞成数、总反对数、总弃权数，根据特定规则排序，选出不超过 `maxSupportNum` 个候选人作为最终结果。

为此，设计了两个 ADT：`Poll` 和 `Vote`。

二 （10 分）`Poll` 是一个接口，它有两个方法，代码见下页。

针对方法 `vote(...)`的 spec，为其设计测试用例。

(a) 按照等价类划分的思想，需要划分出哪些等价类？

(b) 针对你得到的每个等价类，分别设计一个测试用例，给出输出参数的取值和期望的效果。无需写出 JUnit 测试代码。

密 封 线

授课教师 _____ 姓名 _____ 学号 _____ 院系 _____

```
/*A mutable ADT*/
public interface Poll {
    /**
     * 初始化一次投票活动
     * @param candidates  一组候选人，candidates.size()>=1；每个元素代表一个候选人的姓名，
     *                    姓名由字母构成，首位大写，其内部不含空格
     * @param maxSupportNum 每个投票人可投支持票的最大数目，>=1、<=candidates.size()
     * @return 一个新的“投票活动”对象
     */
    static Poll initialize(Set<String> candidates, int maxSupportNum) {
        return new ConcretePoll(candidates, maxSupportNum);
    }

    /**
     * 读取一个投票人的投票，如果选票中不存在四种非法情况，将投票结果加入到记录中，否则抛出异常
     * @param voter 投票人的名字，无限定条件
     * @param votes Key代表候选人名字，Value代表对该候选人的投票结果
     * @throws NoEnoughCandidatesException 如果votes没有覆盖本次投票活动中的所有候选人
     * @throws InvalidCandidatesException 如果votes包含了不在本次投票活动中的候选人
     * @throws InvalidScoresException 如果votes.values()中包含了-1、0、1之外的值
     * @throws BeyondMaxSupportNumException 如果votes中值为1的数目超过了允许支持的最大人数
     */
    void vote(String voter, Map<String, Integer> votes)
            throws NoEnoughCandidatesException, InvalidCandidatesException,
                   InvalidScoresException, BeyondMaxSupportNumException;
```

三（5 分）如果把 vote(...)方法的 spec 中第三个@throws 去掉，在@param votes 那一行的末尾增加“-1 表示反对，0 表示弃权，1 表示支持，其他值不合法”。那么，修改后的 vote(...)的 spec 的强度，相比起修改前发生了什么变化？为什么？

四 （5 分）为了简化客户端代码，不用每次调用 `vote(...)`的时候都提前构造复杂的 `Map` 对象，而是 overload 新的 `void vote(String voter, String votes)`，其中 `votes` 是一个遵循特定语法的字符串，包含了 `voter` 的所有投票信息。

例如：`Jia(-1)Yi(0)Bing(1)`。该字符串开始是候选人的名字（其规则见上页 `initialize()`的 spec 里的说明），后面`()`里的数字是针对该候选人的投票（只能是 `-1/0/1` 之一），若有多个候选人，继续以同样的规则写在后面。上述例子中表示有三个候选人分别为 `Jia`、`Yi`、`Bing`，投票者反对 `Jia`（值为`-1`）、对 `Yi` 弃权（值为`0`）、支持 `Bing`（值为 `1`）。

请写出用于检查 `votes` 是否合法的正则表达式。

五 （10 分）类 `ConcretePoll` 实现了 `Poll` 接口。它的 rep 如下：

```
private Set<String> candidates = new HashSet<>(); // 一组候选人
private Set<Vote> votes = new HashSet<>(); // 投票记录
private int maxSupportNum; //每个投票人投票时允许的最大支持数
```

`Vote` 是一个 immutable ADT，表示投票人对一个候选人的投票，其 rep 为：

```
private String voter;  //投票人名字
private String candidate; //候选人名字
private int score; //投票人给候选人的分数，只能为-1、0、1 之一，
                   //分别表示反对、弃权、支持
```

基于上述 rep 和上页代码，分别写出 `ConcretePoll` 和 `Vote` 的 RI。

密 封 线

授课教师 姓名 学号 院系

六 （5 分）针对第五题给出的 ConcretePoll 的 rep，实现了它的 creator 方法，并增加了两个 observer 方法，分别返回某个候选人的所有得票结果、本次投票的所有候选人：

```
public ConcretePoll(Set<String> candidates, int maxSupportNum) {...}
public List<Integer> getVotesByCandidate(String candidate) {...}
public Set<String> listAllCandidates() {...}
```

不考虑它们内部具体如何实现，判断 ConcretePoll 是否可能存在表示泄露。如果可能存在，考虑在上述方法的具体实现代码中采取什么措施进行规避，以注释形式写出 ConcretePoll 的 safety from rep exposure。

七 （10 分）客户端代码如下，参照第五题给出的 rep，画出该代码运行之后的 snapshot diagram。

```
Set<String> candidates = new HashSet<>(Arrays.asList("Ding", "Wu"));
Poll poll = Poll.initialize(candidates, 2);
Map<String, Integer> votes = new HashMap<>();
votes.put("Ding", 1);
votes.put("Wu", 0);
poll.vote("Ji", votes);
```

八（10 分）在 Poll 的基础上，考虑各类现实中各类有差异的投票场景，因此对原有设计进行修改，增加子类型。

场景 a：投票人针对候选人的投票中不允许出现“弃权”。为其设计一个子类型：

```
class NoWaiverPoll extends ConcretePoll implements Poll
```

场景 b：投票结果不是简单的采用“支持、反对、弃权”，还要给出精确的分数，范围在[-10,10]之间，其绝对值越大，表示支持或反对的程度越大。为其设计另一个子类型：

```
class PrecisePoll extends ConcretePoll implements Poll
```

假设上述两个子类型相对于第五题给出的 ConcretePoll 的 rep 没有增加任何新属性，相比第 6 页 Poll 接口没有增加任何新方法，请分别判断这两个子类型相对于 Poll 来说是否符合 LSP？并简要说明理由。

院系＿＿＿＿＿＿ 学号＿＿＿＿＿＿ 姓名＿＿＿＿＿＿ 授课教师＿＿＿＿＿＿

密 封 线

九 （10 分）为 Poll 增加一个 getVoteResult()方法，以计算最终在投票中获胜的候选人及其支持率，其 spec 如下所示：

```
/**
 * 获取投票统计结果
 * @return key为候选人，仅包含符合“获胜”条件的候选人；
 *         value为key中候选人的支持率，值域为[0,1]
 */
Map<String, Double> getVoteResult();
```

在不同的投票活动中，统计获胜的规则可能有所差异。例如：

情况 1：根据支持票的比例，从大到小按次序选择 maxSupportNum 个候选人；

情况 2：不仅考虑支持票的比例，还要考虑支持票比例是否超过总投票人数的 2/3，从大到小次序选择最多 maxSupportNum 个候选人。

考虑未来应用中可能还会引入新的统计规则，故需要对该方法进行重新设计。你可以改其参数列表（但方法名和返回值不能改），也可以引入新的类。简要阐述你的设计思路，以及该思路如何支持客户端动态采用不同的统计规则。