



哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2019 年春季学期
计算机学院《软件构造》课程

Lab 4 实验报告

姓名	林之浩
学号	1170300817
班号	1703008
电子邮件	630073498@qq.com
手机号码	18065053516

目录

1 实验目标概述	1
2 实验环境配置	1
3 实验过程	3
3.1 Error and Exception Handling	3
3.1.1 输入不合法	3
3.1.1.1 TrackGame	4
3.1.1.2 Atomstructure	5
3.1.1.3 Socialnetwork	5
3.1.2 标签一样错误	6
3.1.3 依赖关系错误	6
3.2 Assertion and Defensive Programming	7
3.2.1 checkRep() 检查 invariants	7
3.2.1.1 TrackGame:	7
3.2.1.2 SocialNetCircle:	8
3.2.1.3 针对各个 physicalObject 的子类:	8
3.2.2 Assertion 保障 pre-/post-condition	9
3.3 Logging	9
3.3.1 写日志	9
3.3.2 日志查询	10
3.4 Testing for Robustness and Correctness	13
3.4.1 Testing strategy	13
3.4.2 测试用例设计	13
3.4.2.1 文件读写过程:	13
3.4.2.2 不在文件读入过程发生的参数错误	14
3.4.2.3 健壮性测试	15
3.4.3 测试运行结果与 EclEmma 覆盖度报告	15
3.5 SpotBugs tool	16
3.5.1 readLine	16
3.5.2 类名大写开头	17
3.5.3 Readline 错误	17
3.6 Debugging	17

3.6.1 理解待调试程序的代码思想	17
3.6.1.1 findMedianSortedArrays	17
3.6.1.2 removeComments	18
3.6.1.3 TopVotedCandidate	18
3.6.2 发现并定位错误的过程	18
3.6.2.1 findMedianSortedArrays	18
3.6.2.2 removeComments	19
3.6.2.3 TopVotedCandidate	19
3.6.3 如何修正错误	20
3.6.3.1 findMedianSortedArrays	20
3.6.3.2 removeComments	20
3.6.3.3 TopVotedCandidate	21
3.6.4 结果	21
3.6.4.1 findMedianSortedArrays	21
3.6.4.2 removeComments	22
3.6.4.3 TopVotedCandidate	23
4 实验进度记录	23
5 实验过程中遇到的困难与解决途径	23
6 实验过程中收获的经验、教训、感想	24
6.1 实验过程中收获的经验教训	24
6.2 针对以下方面的感受	24

1 实验目标概述

本次实验重点训练学生面向健壮性和正确性的编程技能，利用错误和异常处理、断言与防御式编程技术、日志/断点等调试技术、黑盒测试编程技术，使程序可在不同的健壮性/正确性需求下能恰当的处理各种例外与错误情况，在出错后可优雅的退出或继续执行，发现错误之后可有效的定位错误并做出修改。

实验针对Lab 3中写好的ADT代码和基于该ADT的三个应用的代码，使用以下技术进行改造，提高其健壮性和正确性：

错误处理

异常处理

防御式编程

日志

调试技术

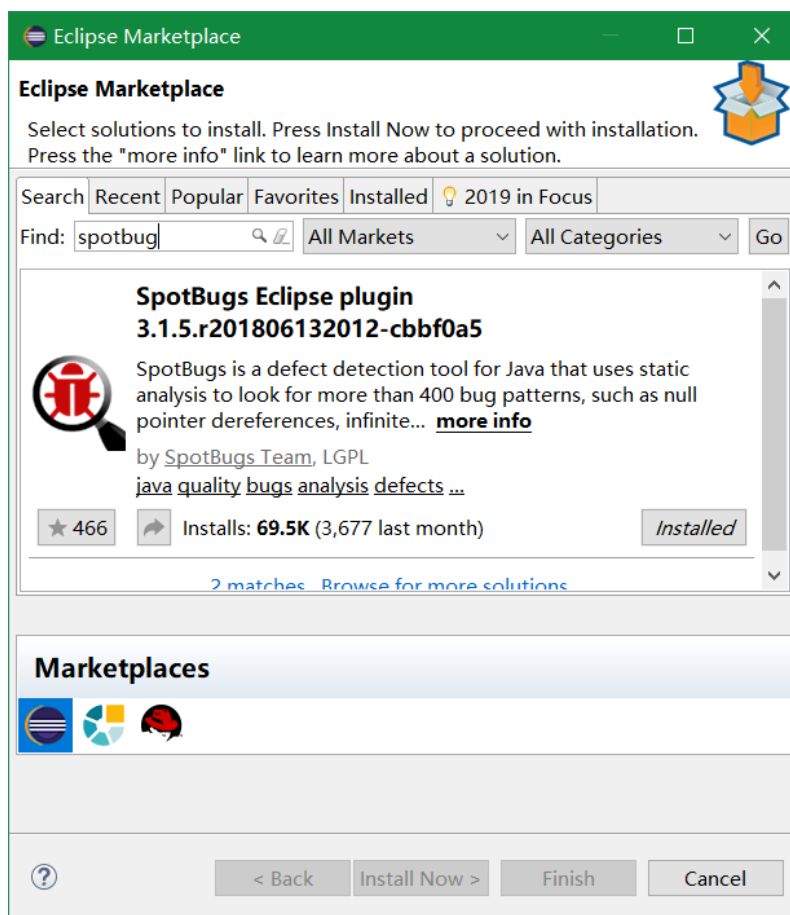
黑盒测试及代码覆盖度

2 实验环境配置

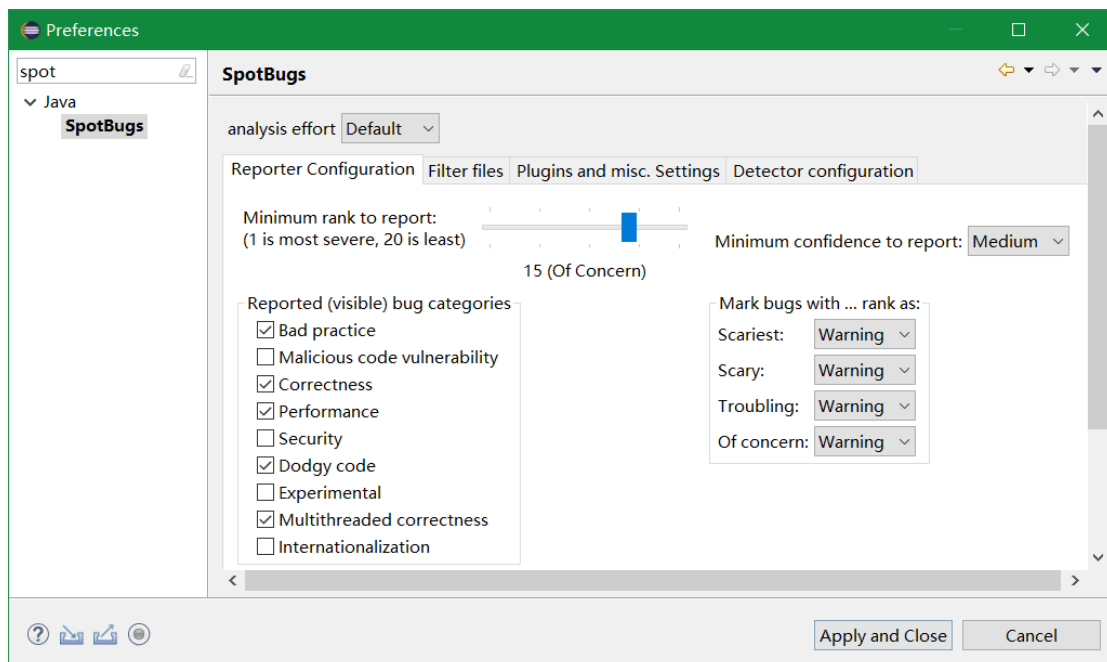
简要陈述你配置本次实验所需环境的过程，必要时可以给出屏幕截图。

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。

直接在 eclipse marketplace 中搜索 spotbugs 然后安装。



并且在 windows-preference 下的 spotbugs 菜单对其进行配置



在这里给出你的 GitHub Lab4 仓库的 URL 地址（Lab4-学号）。

<https://github.com/ComputerScienceHIT/Lab4-1170300817>

3 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！）。

3.1 Error and Exception Handling

下列所有错误出现后都会提示用户重新输入读取的文件的地址

以下省略开始选择菜单项 1 的步骤：

```
1. 读取文件构造系统
2. 可视化
3. 查询好友位置
4. 打印轨道结构
5. 增加新的朋友
6. 增加新的关系并重新整理
7. 删除一个关系并重新整理
8. 计算熵值
9. 计算信息扩散度
10. 计算两个用户之间的逻辑距离
end. 结束
1
输入需要读取的文件名：例如SocialNetworkCircle.txt
exception\SocialLocialErr.txt
```

3.1.1 输入不合法

截取到不合规定的文件输入之后，调用一个分析文本的方法 `analyzeInput`，以 `trackgame` 为例：

将输入拆分成若干小块，然后将他们分别与原来的正则表达式的每个小部分匹配，就能找到是哪个部分出现了何种类型错误

```

public String analyzeInput(String readLine) throws illegalTextGrammarException {
    String msgString = null;
    String[] arguments = readLine.trim().split("\\s");
    if (arguments[0].equals("Athlete")) {
        String[] parameter = arguments[2].split(",");
        if (parameter.length != 5) {
            throw new illegalTextGrammarException(readLine + ":运动员参数缺失");
        }
        if (!Pattern.matches("<[A-Za-z]+>", parameter[0])) {
            throw new illegalTextGrammarException(readLine + ":运动员名字错误");
        }
        if (!Pattern.matches("[A-Z]{3}", parameter[2])) {
            throw new illegalTextGrammarException(readLine + ":运动员国籍错误");
        }
        if (!Pattern.matches("\\d{1,2}\\.|\\d{2}>", parameter[4])) {
            throw new illegalTextGrammarException(readLine + ":运动员最好成绩错误");
        }
    } else if (arguments[0].equals("Game")) {
        if (!Pattern.matches("(100|200|400)", arguments[2])) {
            throw new illegalTextGrammarException(readLine + ":无效的比赛类型");
        }
    } else if (arguments[0].equals("NumOfTracks")) {
        if (!Pattern.matches("[4-9]|10", arguments[2])) {
            throw new illegalTextGrammarException(readLine + ":无效的轨道数目");
        }
    }
    return msgString;
}

```

测试文件位于

src > txt > exception

各异常触发情况如下： 分别按顺序读取异常文件

3.1.1.1 TrackGame

exception/TrackGame1.txt

Athlete ::= <1,JAM,38,9.94>:运动员参数缺失

exception/TrackGame2.txt

Athlete ::= <111,1,JAM,38,9.94>:运动员名字错误

exception/TrackGame3.txt

Athlete ::= <Bolt,1,JaAM,38,9.94>:运动员国籍错误

exception/TrackGame4.txt

Athlete ::= <Bolt,1,JAM,38,9.914>:运动员最好成绩错误

exception/TrackGame5.txt

Game ::= 300:无效的比赛类型

exception/TrackGame6.txt

NumOfTracks ::= 11 :无效的轨道数目

3.1.1.2 Atomstructure

exception/Atom1.txt

ElementName ::= :元素名字参数缺失

exception/Atom2.txt

ElementName ::= Rba:元素名字错误

exception/Atom3.txt

NumberOfTracks ::= :轨道数参数缺失
请重新读取文件

exception/Atom4.txt

NumberOfTracks ::= a:轨道数错误

exception/Atom5.txt

NumberOfElectron ::= /2;2/8;3/18;4/8;5/1:轨道电子参数错误

3.1.1.3 Socialnetwork

exception/Social1.txt

CentralUser ::= <TommyWong,30>:人物参数缺失

exception/Social2.txt

CentralUser ::= <Tommy鑄學ong,30,M>:人物名字错误

exception/Social3.txt

CentralUser ::= <TommyWong,3a0,M>:人物年龄错误

exception/Social4.txt

CentralUser ::= <TommyWong,30,L>:人物性别错误

exception/Social5.txt

SocialTie ::= <TommyWong, LisaWong,:社交关系参数缺失

exception/Social6.txt

SocialTie ::= <TommyW...ong, LisaWong, 0.98>:社交关系名字错误

exception/Social7.txt

SocialTie ::= <TommyWong, LisaWong, 1.98>:社交关系亲密度错误

3.1.2 标签一样错误

设计如下输入，获得以下输出

```
kGame6.txt x AtomSameLabel.txt x AtomicStructure.txt x SocialSameLabel.txt x
CentralUser ::= <TommyWong, 30, M>
SocialTie ::= <TommyWong, LisaWong, 0.98>
Friend ::= <LisaWong, 25, F>
Friend ::= <LisaWong, 25, F>
```

exception/SocialSameLabel.txt

LisaWong为名的对象已经存在

```
Athlete ::= <Bolt, 1, JAM, 38, 9.94>
Athlete ::= <Bolt, 1, JAM, 38, 9.94>
Game ::= 100
NumOfTracks ::= 11
```

exception/TrackGameSameLabel.txt

Bolt为名的对象已经存在

```
CentralUser ::= <TommyWong, 30, M>
SocialTie ::= <TommyWong, LisaWong, 0.98>
SocialTie ::= <TommyWong, LisaWong, 0.98>
Friend ::= <LisaWong, 25, F>
```

exception/SocialSameLabel2.txt

从TommyWong到LisaWong的边已经存在
请重新读取文件

3.1.3 依赖关系错误

设计如下输入，获得输出：

轨道数目不一致：

```
ElementName ::= Rb
NumberOfTracks ::= 2
NumberOfElectron ::= 1/2
```

```
exception\AtomLogicalErr.txt
```

```
轨道数前后不一致错误
请重新读取文件
```

关系中出现未定义的人:

```
CentralUser ::= <TommyWong, 30, M>
SocialTie ::= <TommyWong, aaa, 0.98>
Friend ::= <LisaWong, 25, F>
```

```
exception\SocialLocialErr.txt
```

```
关系中出现未定义的人:aaa
请重新读取文件
```

3.2 Assertion and Defensive Programming

3.2.1 checkRep()检查 invariants

3.2.1.1 TrackGame:

在 TrackCircularOrbit 修改 checkRep:

```
public void checkRep() {
    //中心物体为空
    assert this.getCentralObject() == null;
    //每个轨道最多一个人
    for(Entry<Track, List<Athlete>> e:OrbitMap.entrySet()) {
        assert e.getValue().size() <= 1;
    }
    //轨道数目在4-10之间
    assert OrbitMap.keySet().size() >= 4;
    assert OrbitMap.keySet().size() <= 10;
}
```

在 TrackGame 中则要对比赛每一轮对应的 TrackCircularOrbit 调用 checkRep, 所以再写一个

checkRep 检查每一轮的轨道系统：

```
public void checkRep() {
    for (TrackCircularOrbit Orbit : trackOrbitList) {
        Orbit.checkRep();
    }
}
```

对于一些其他条件例如：“前 $n-1$ 个轨道必须满编”通过安排竞赛的算法的正确性保证的，故不做检验。

3.2.1.2 SocialNetCircle:

对于：

在 `SocialNetworkCircle` 中，不管社交关系如何增加或删除，第 i 层轨道上的人与中心点的人之间的最短路径等于 i 。

这个要求，因为我的实现在安排轨道的结构的时候是先用 BFS 搜索算出物体到中心点的逻辑距离，再把这个物体放在对应逻辑距离对应的轨道上，所以，这个验证无异于“自己验证自己”，故没有实现，这一个要求是通过每次增删关系都重新调用一次重整轨道结构的 `reArrange` 方法，以及在 `reArrange` 方法中使用的 BFS 算法的正确性来保证的。

```
/**
 * 重整轨道结构
 */
public void reArrange() {
    // ...
}
```

3.2.1.3 针对各个 `physicalObject` 的子类：

用正则匹配的方式检验参数是否合法，不合法抛出异常，方法和文件读取阶段的异常处理无异：

举例：

```
public void checkRep() throws IllegalArgumentException {
    if (!Pattern.matches("<[A-Za-z]+)", this.name)) {
        throw new IllegalArgumentException(this.name + ":运动员名字错误");
    }
    if (!Pattern.matches("[A-Z]{3}", nationality)) {
        throw new IllegalArgumentException(nationality + ":运动员国籍错误");
    }
    if (!Pattern.matches("\\d{1,2}\\.|\\d{2}>", String.valueOf(bestRecord))) {
        throw new IllegalArgumentException(bestRecord + ":运动员最好成绩错误");
    }
}
```

3.2.2 Assertion 保障 pre-/post-condition

有的时候 pre-/post-condition 不是通过 assertion 来保证，而是通过抛出异常
例如：社交网络中要求加入的边不能有自己到自己，或者是 ab, ba 同时出现：

```
if (fromString.equals(toString)) {
    throw new sameLabelException("从" + fromString + "到" + toString + "的边无效");
}
for (relationKeeper k : keeperList) {
    if (k.getFromString().equals(toString) && k.getToString().equals(fromString)) {
        throw new sameLabelException("从" + fromString + "到" + toString + "的边已经存在");
    }
    if (k.getFromString().equals(fromString) && k.getToString().equals(toString)) {
        throw new sameLabelException("从" + fromString + "到" + toString + "的边已经存在");
    }
}
```

还有一些简单的使用 assert 保障 pre-/post-condition 的：举个例子：

向 ConcreteCircularOrbit 中添加新的点、轨道、关系、中心物体时，都要防止对象是一个 null。

```
public boolean addtrackRelation(E object1, E object2, double distance) {
    assert object1 != null;
    assert object2 != null;
    if (!trackRelationMap.containsKey(object1)) {
        trackRelationMap.put(object1, new HashMap<E, Double>());
    }
}
```

另外一些防御式的编程的例子比如使用防御性拷贝

```
/**
 * 返回与中心连接的物体
 *
 * @return 物体构成的集合
 */
public Set<E> getCentralConnectedObject() {
    return new HashSet<>(centralRelationMap.keySet());
}
```

3.3 Logging

3.3.1 写日志

导入 log4j 包，编写配置文件放置于 src 文件夹下（参考网上）
设置好控制台输出和文本输出，
配置文件如下：

```

### 设置###
log4j.rootLogger = debug,stdout,D,E

### 输出信息到控制台 ###
log4j.appender.stdout = org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target = System.out
log4j.appender.stdout.Threshold = DEBUG
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern = [%-5p] %d{yyyy-MM-dd HH:mm:ss,SSS} method:%l%n%n

### 输出DEBUG 级别以上的日志文件设置 ###
log4j.appender.D = org.apache.log4j.DailyRollingFileAppender
log4j.appender.D.File = out/debug.log
log4j.appender.D.Append = true
log4j.appender.D.Threshold = DEBUG
log4j.appender.D.layout = org.apache.log4j.PatternLayout
log4j.appender.D.layout.ConversionPattern = TYPE(%p) TIME(%-d{yyyy-MM-dd HH:mm:ss}) METHOD (%l) REASON(%m)%n

### 输出ERROR 级别以上的日志文件设置 ###
log4j.appender.E = org.apache.log4j.DailyRollingFileAppender
log4j.appender.E.File = out/error.log
log4j.appender.E.Append = true
log4j.appender.E.Threshold = ERROR
log4j.appender.E.layout = org.apache.log4j.PatternLayout
log4j.appender.E.layout.ConversionPattern = TYPE(%p) TIME(%-d{yyyy-MM-dd HH:mm:ss}) METHOD (%l) REASON(%m)%n

```

在每一个方法后面增加记录日志的操作:

对于所有的异常,都以 ERROR 级别记录,对于所有的操作提示和记录,都已 DEBUG 级别记录,分别输出到以下两个文件中

(E:) > eclipseworkspace > Lab4_1170300817 > out

名称	修改日期
debug.log	2019/5/19 星期...
error.log	2019/5/18 星期...

生成的日志文件如下 (以 trackgame 为例):

```

debug.log - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
TYPE(DEBUG) TIME(2019-05-19 00:51:52) METHOD (:applications.AtomStructure.AtomGame.gameMain(AtomGame.java:82)) REASON(文件读取成功)
TYPE(DEBUG) TIME(2019-05-19 00:58:06) METHOD (:applications.AtomStructure.AtomGame.gameMain(AtomGame.java:95)) REASON(跃迁成功:从track0到track2)
TYPE(DEBUG) TIME(2019-05-19 00:58:09) METHOD (:applications.AtomStructure.AtomGame.gameMain(AtomGame.java:106)) REASON(回退成功)
TYPE(DEBUG) TIME(2019-05-19 00:58:14) METHOD (:applications.AtomStructure.AtomGame.gameMain(AtomGame.java:117)) REASON(文字输出完成)
TYPE(DEBUG) TIME(2019-05-19 00:58:35) METHOD (:applications.AtomStructure.AtomGame.gameMain(AtomGame.java:113)) REASON(可视化完成)
TYPE(DEBUG) TIME(2019-05-19 00:58:41) METHOD (:applications.AtomStructure.AtomGame.gameMain(AtomGame.java:125)) REASON(轨道增加成功)
TYPE(DEBUG) TIME(2019-05-19 00:58:45) METHOD (:applications.AtomStructure.AtomGame.gameMain(AtomGame.java:137)) REASON(增加成功)
TYPE(DEBUG) TIME(2019-05-19 00:59:37) METHOD (:applications.AtomStructure.AtomGame.gameMain(AtomGame.java:125)) REASON(轨道增加成功)
TYPE(DEBUG) TIME(2019-05-19 00:59:44) METHOD (:applications.AtomStructure.AtomGame.gameMain(AtomGame.java:95)) REASON(跃迁成功:从track0到track5)
TYPE(DEBUG) TIME(2019-05-19 00:59:47) METHOD (:applications.AtomStructure.AtomGame.gameMain(AtomGame.java:106)) REASON(回退成功)

```

输出如下:

```

错误类型: DEBUG
类名: applications.AtomStructure.AtomGame
方法名: gameMain
时间: 2019-05-19T01:17:35Z
详细信息: 文件读取成功

```

3.3.2 日志查询

首先用正则表达式读取整个日志文件:

```

public logRecord(String line) {
    String regex = "TYPE\\((.*)\\) TIME\\((.*)\\) METHOD\\((.*)\\)\\.\\((.*)\\)\\.\\((.*)\\) REASON\\((.*)\\)"
    Matcher matcher = Pattern.compile(regex).matcher(line);
    if(matcher.find()) {
        logType = matcher.group(1);
        className = matcher.group(3);
        methodName = matcher.group(4);
        reason = matcher.group(5);
        String timeStr = matcher.group(2);
        time = Instant.parse(timeStr.replace(' ', 'T') + "Z");
    }
}

```

学习使用了 JAVA 8 的新特性用 stream 来筛选日志列表

```

public String getFilterLogs(Predicate<logRecord> predicate) {
    List<logRecord> logs = records.stream().filter(predicate).collect(Collectors.toList());
    StringBuilder sb = new StringBuilder();
    for (logRecord logRecord : logs) {
        sb.append(logRecord.toString());
    }
    return sb.toString();
}

```

只需传入一个 Predicate 即可完成筛选例如:

```

case "2":// 按类型
    System.out.println("请输入类型: ERROR或DEBUG");
    inputString = reader.readLine().trim();
    final String input1 = new String(inputString);
    System.out.println(LOGKEEPER.getFilterLogs(p -> p.getLogType().equals(input1)));
    System.out.println("查询完毕");
}

```

查询演示:

end. 结束

11

选择筛选条件:

1. 时间段
2. 按类型
3. 按类
4. 按方法

1

请输入开始时间, 格式参考: 2019-05-19 02:05:14

2019-05-19 01:17:38

请输入结束时间, 格式参考: 2019-05-19 02:05:14

2019-05-19 10:58:11

错误类型: DEBUG

类名: applications.AtomStructure.AtomGame

方法名: gameMain

时间: 2019-05-19T02:05:14Z

详细信息: 文件读取成功

2

请输入类型: ERROR或DEBUG

DEBUG

错误类型: DEBUG

类名: applications.AtomStructure.AtomGame

方法名: gameMain

时间: 2019-05-19T01:17:35Z

详细信息: 文件读取成功

错误类型: DEBUG

类名: applications.AtomStructure.AtomGame

方法名: gameMain

时间: 2019-05-19T01:17:38Z

详细信息: 跃迁成功:从track0到track2

错误类型: DEBUG

类名: applications.AtomStructure.AtomGame

方法名: gameMain

时间: 2019-05-19T02:05:14Z

详细信息: 文件读取成功

错误类型: DEBUG

类名: applications.AtomStructure.AtomGame

方法名: gameMain

时间: 2019-05-19T10:58:11Z

详细信息: 文件读取成功

选择筛选条件:

1. 时间段
2. 按类型
3. 按类
4. 按方法

3

请输入类名

applications.AtomStructure.AtomGame

错误类型: DEBUG

类名: applications.AtomStructure.AtomGame

方法名: gameMain

时间: 2019-05-19T01:17:35Z

详细信息: 文件读取成功

错误类型: DEBUG

类名: applications.AtomStructure.AtomGame

方法名: gameMain

时间: 2019-05-19T01:17:38Z

详细信息: 跃迁成功:从track0到track2

错误类型: DEBUG

类名: applications.AtomStructure.AtomGame

方法名: gameMain

时间: 2019-05-19T02:05:14Z

详细信息: 文件读取成功

11

选择筛选条件:

1. 时间段
2. 按类型
3. 按类
4. 按方法

4

请输入方法名

gameMain

错误类型: DEBUG

类名: applications.AtomStructure.AtomGame

方法名: gameMain

时间: 2019-05-19T01:17:35Z

详细信息: 文件读取成功

错误类型: DEBUG

类名: applications.AtomStructure.AtomGame

方法名: gameMain

时间: 2019-05-19T01:17:38Z

详细信息: 跃迁成功:从track0到track2

错误类型: DEBUG

类名: applications.AtomStructure.AtomGame

方法名: gameMain

时间: 2019-05-19T02:05:14Z

详细信息: 文件读取成功

3.4 Testing for Robustness and Correctness

3.4.1 Testing strategy

总体策略是尽可能提高代码的覆盖率，尽可能覆盖到每一个分支，所以需要等价类划分的思想。对于有 exception 的功能，必须测试到每一个 exception 和相应的处理是否正确执行，对 3.1 节编写的每一个 exception 都编写单独的测试文档来测试。

3.4.2 测试用例设计

3.4.2.1 文件读写过程：

针对每一个 exception 的每一种触发情况都编写对应的测试用例和测试文件：

目录：src/txt/exception

电脑 > 工作 (E:) > eclipseworkspace > Lab4_1170300817 > src > txt > exception

名称	修改日期	类型	大小
Atom1.txt	2019/5/17 星期...	TXT 文件	1 KB
Atom2.txt	2019/5/17 星期...	TXT 文件	1 KB
Atom3.txt	2019/5/17 星期...	TXT 文件	1 KB
Atom4.txt	2019/5/17 星期...	TXT 文件	1 KB
Atom5.txt	2019/5/17 星期...	TXT 文件	1 KB
AtomLogicalErr.txt	2019/5/18 星期...	TXT 文件	1 KB
Social1.txt	2019/5/17 星期...	TXT 文件	1 KB
Social2.txt	2019/5/17 星期...	TXT 文件	1 KB
Social3.txt	2019/5/17 星期...	TXT 文件	1 KB
Social4.txt	2019/5/17 星期...	TXT 文件	1 KB
Social5.txt	2019/5/19 星期...	TXT 文件	1 KB
Social6.txt	2019/5/19 星期...	TXT 文件	1 KB
Social7.txt	2019/5/19 星期...	TXT 文件	1 KB
SocialLocalErr.txt	2019/5/19 星期...	TXT 文件	1 KB
SocialSameLabel.txt	2019/5/19 星期...	TXT 文件	1 KB
SocialSameLabel2.txt	2019/5/19 星期...	TXT 文件	1 KB
TrackGame1.txt	2019/5/17 星期...	TXT 文件	1 KB
TrackGame2.txt	2019/5/17 星期...	TXT 文件	1 KB
TrackGame3.txt	2019/5/17 星期...	TXT 文件	1 KB
TrackGame4.txt	2019/5/17 星期...	TXT 文件	1 KB
TrackGame5.txt	2019/5/19 星期...	TXT 文件	1 KB
TrackGame6.txt	2019/5/19 星期...	TXT 文件	1 KB
TrackGameSameLabel.txt	2019/5/19 星期...	TXT 文件	1 KB

编写对应的 test 读取这些文件，检测是否出现了设计好应该出现的错误：

举例如下：


```

@Test
public void exception1Test() {
    try {
        builder.createFromFile("src/txt/exception/Atom1.txt");
    } catch (NumberFormatException | illegalTextGrammarException | IOException | logicalErrorException e) {
        assertTrue(e.getMessage().contains(":元素名字参数缺失"));
    }
}

@Test
public void exception2Test()
    throws NumberFormatException, illegalTextGrammarException, IOException, logicalErrorException {
    try {
        builder.createFromFile("src/txt/exception/Atom2.txt");
    } catch (NumberFormatException | illegalTextGrammarException | IOException | logicalErrorException e) {
        assertTrue(e.getMessage().contains(":元素名字错误"));
    }
}

@Test
public void exception3Test()
    throws NumberFormatException, illegalTextGrammarException, IOException, logicalErrorException {
    try {
        builder.createFromFile("src/txt/exception/Atom3.txt");
    } catch (NumberFormatException | illegalTextGrammarException | IOException | logicalErrorException e) {
        assertTrue(e.getMessage().contains(":轨道数参数缺失"));
    }
}

```

3.4.2.2 不在文件读入过程发生的参数错误

不在文件读入过程发生的参数错误，举例：

```

@Test
public void personeEX1Test() {
    try {
        Person.getInstance(",a", 16, "M");
    } catch (illegalParameterException | sameLabelException e) {
        assertTrue(e.getMessage().contains("人物名字错误"));
    }
}

@Test
public void personeEX2Test() {
    try {
        Person.getInstance("aas", 16, "S");
    } catch (illegalParameterException | sameLabelException e) {
        assertTrue(e.getMessage().contains("人物性别错误"));
    }
}

@Test
public void personeEX3Test() {
    try {
        Person.getInstance("aax", 1111, "M");
    } catch (illegalParameterException | sameLabelException e) {
        assertTrue(e.getMessage().contains("人物年龄错误"));
    }
}

```

3.4.2.3 健壮性测试

比如：移除目标的轨道上没有物体，移除物体不在轨道系统里等情况能否正常返回

举例：执行两次 transit，轨道上只有一个物体，第二次返回的应该是个 false

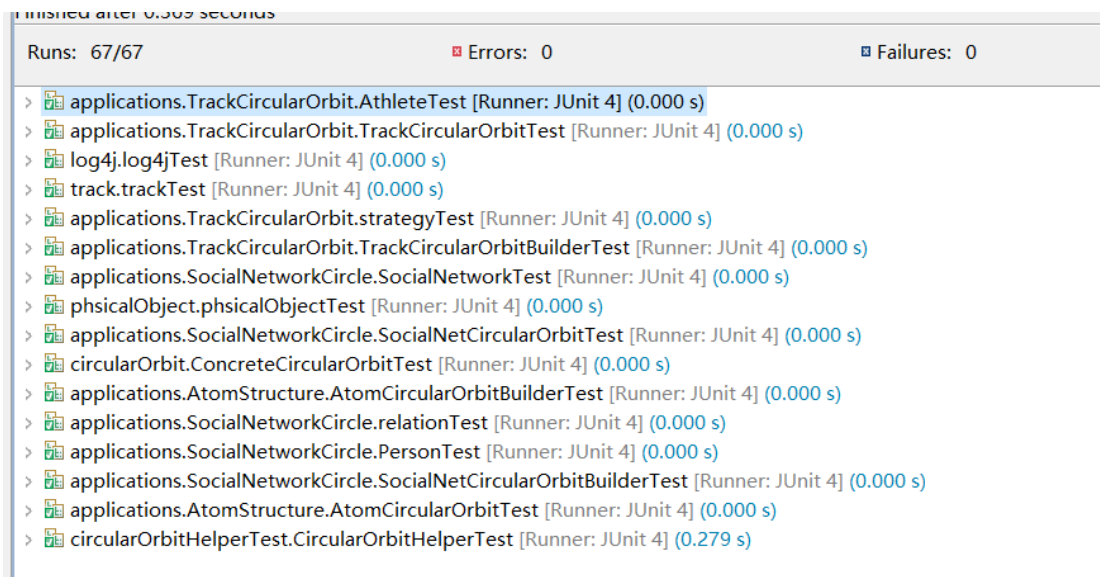
```
atomCircularOrbit.transit(t1, t2);
assertFalse(atomCircularOrbit.transit(t1, t2));
```

试图声明一些非法参数的物体：

检测 catch 到的异常的信息是否和错误原因一致

```
@Test
public void athleteEX1Test() {
    try {
        Athlete.getInstance("a", 16, "AAM", 12, 9.65);
    } catch (IllegalArgumentException | sameLabelException e) {
        assertTrue(e.getMessage().contains("运动员名字错误"));
    }
}
```

3.4.3 测试运行结果与 Eclemma 覆盖度报告



注：客户端代码和图形化的部分无法测试，所以以下的 AtomGame, SocialNetworkGame, TrackGame 无法覆盖，覆盖度为 0。

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
Lab4 1170300817		7,014	3,376	10,390
src	54.9 %	3,843	3,156	6,999
applications.AtomStructure	38.5 %	454	725	1,179
> AtomCircularOrbit.java	100.0 %	95	0	95
> AtomCircularOrbitBuilder.java	98.7 %	301	4	305
> AtomGame.java	0.0 %	0	721	721
> Memento.java	100.0 %	15	0	15
> Particle.java	100.0 %	18	0	18
> TransitCareTaker.java	100.0 %	25	0	25
applications.SocialNetworkCi	50.5 %	1,200	1,175	2,375
> Person.java	100.0 %	120	0	120
> relationKeeper.java	100.0 %	116	0	116
> SocialNetCircularOrbit.java	54.6 %	387	322	709
> SocialNetCircularOrbitBuil	98.0 %	577	12	589
> SocialNetworkGame.java	0.0 %	0	841	841
applications.TrackGame	36.9 %	524	895	1,419
> Athlete.java	100.0 %	155	0	155
> TrackCircularOrbit.java	87.7 %	128	18	146
> TrackCircularOrbitBuilder.j	90.9 %	241	24	265
> TrackGame.java	0.0 %	0	853	853
> applications.TrackGame.Strat	98.3 %	229	4	233
> centralObject	79.3 %	23	6	29
> circularOrbit	79.1 %	996	263	1,259
> circularOrbitHelper	83.3 %	15	3	18
> difference	100.0 %	111	0	111
> exception	100.0 %	24	0	24
> logRecord	94.0 %	202	13	215
> phsicalObject	100.0 %	9	0	9
> starter	0.0 %	0	60	60
> track	82.4 %	56	12	68
> test	93.5 %	3,171	220	3,391

3.5 SpotBugs tool

发现了哪些错误，每种错误代表什么不良的编程习惯
对代码修改，消除这些错误。

3.5.1 readLine

BufferedReader 类的 readLine() 在没有文本读取的情况下将 返回 null，在操作之前需要判断是否为 null。

旧:

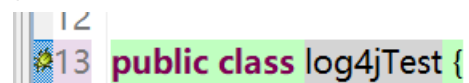
```
25 String input = reader.readLine().trim();
```

新:

```
;  
    String input = reader.readLine();  
    if (input != null) {  
        input = input.trim();  
    } else {  
        continue;  
    }  
}
```

3.5.2 类名大写开头

旧：



```
12  
13 public class log4jTest {
```

新：

3.5.3 Readline 错误

Readline 要判断非 null

旧：

```
inputString = reader.readLine();
```

新：

```
String input = reader.readLine();  
if (input != null) {  
    input = input.trim();  
} else {  
    continue;  
}
```

3.6 Debugging

3.6.1 理解待调试程序的代码思想

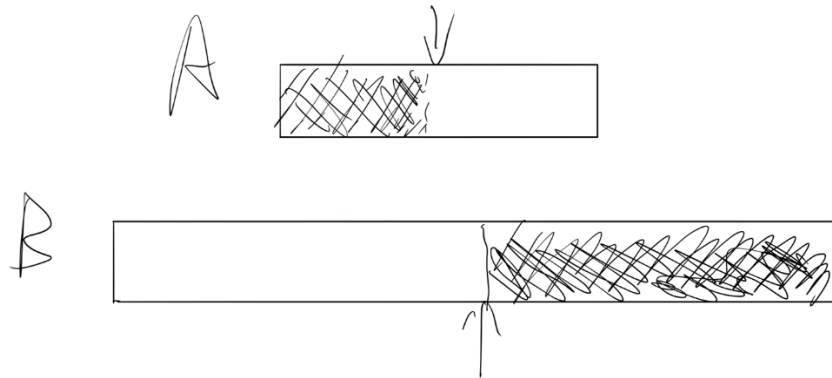
3.6.1.1 findMedianSortedArrays

目的：寻找两个有序数组合起来的中位数

不需要真正的合并而是找到两个特殊值 minRight 和 maxLeft 分别是合并以后的右半边最小值和左半边最大值，然后对这两个数求平均值。

算法本质上是寻找两个指针 i, j ，使得 $A[0-i]$ 和 $B[0-j]$ 合起来正好是 AB 合并之后的左半部分所以 $i+j$ 一定等于总长度的一半。

先把 A 数组换成短的数组，因为 A, B 数组有序，所以 A 的前半部分和 B 的后半部分是不用考虑的，因为 A 的前半肯定在合并后的左侧， B 的后半肯定在合并后的右侧。



随后一步步寻找这样的 i, j 使得 $B[j - 1] \leq A[i]$ ， $A[i - 1] \leq B[j]$ ，这样就找到了要求的 i, j 。

随后得到

$\text{maxLeft} = \max(B[j - 1], A[i - 1])$

$\text{minRight} = \min(B[j], A[i])$

取平均值即可。

3.6.1.2 removeComments

判断一段代码是否是在注释中，只需要判断代码之前是否有 `/*` 或者 `/**` 即可，如果不是注释代码，加入数组返回。

3.6.1.3 TopVotedCandidate

参考了网上的解答 (leetcode)

解法：设置一个目前得票最多的变量 count ，先循环得票人和时间，用 `hashmap` 或数组数组统计每个人的得票数，如果次数大于 count ，则更新 count ，用数组记录到目前时间得票最多的人(按时间顺序排列的目前得票多的人)。查找某一时间点得票最多的人时，用二分法查找按时间排序的数组。

3.6.2 发现并定位错误的过程

3.6.2.1 findMedianSortedArrays

根据算法要求的 i, j 的和的关系 ($A[i]$ 左边和 $B[j]$ 左边占总元素的一半) 可以知道，这里长度和 i 指针的计算有错误。

3.6.2.2 removeComments

```
if (!inBlock && i + 1 <= line.length() && chars[i] == '/' && chars[i + 1] == '*') {
    inBlock = true;
```

`i++;` // 新增 `i++` 以跳过 `*` 字符

这里如果没有 `i++` 就会使指针越界。

缺少处理单行注释的代码：

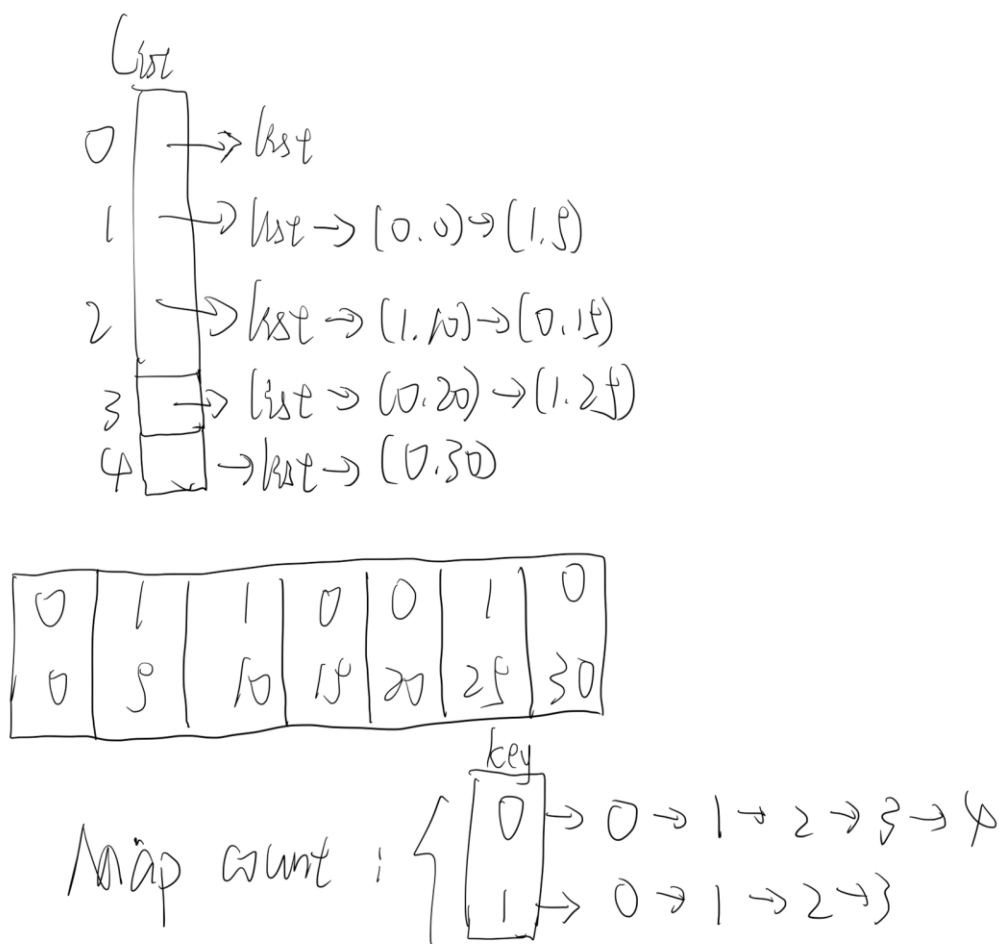
逻辑有误：

```
if (inBlock && newline.length() > 0) {
```

inblock 时不应该写入

3.6.2.3 TopVotedCandidate

分析流程和数据结构如下：



```
// int c = count.getOrDefault(p, 1); 初始为0, 在原来基础上+1
```

Count 记录投票次数, 显然初始值为 0, 每次加 1.

```

    int lo = 1, hi = A.size();
    while (lo < hi) {
        int mi = lo + (hi - lo) / 2;
        if (A.get(mi).get(0).time <= t)
//         lo = mi;
        lo = mi + 1;
    }

```

lo 类似搜索下界，当前的 mi 已经搜索过，可以改成 mi+1

```

    if (A.get(i).get(mi).time < t)

```

截止到某个时间点，所以是<=

```

//     int i = lo;
    int i = lo - 1;
和
//     int j = Math.max(lo, 0);
    int j = Math.max(lo - 1, 0);

```

3.6.3 如何修正错误

3.6.3.1 findMedianSortedArrays

修正 i, j 和 halfLen

```

//     int iMin = 0, iMax = m, halfLen = (m + n) / 2;
    int iMin = 0, iMax = m, halfLen = (m + n + 1) / 2;
    ...

int i = (iMin + iMax + 1) / 2;
int i = (iMin + iMax) / 2;
//         if ((m + n + 1) % 2 == 1) {
//             if ((m + n) % 2 == 1) {

```

3.6.3.2 removeComments

```

    if (!inBlock && i + 1 <= line.length() && chars[i] == '/' && chars[i + 1] == '*') {
        inBlock = true;
        i++; // 新增i++以跳过*字符
    }

```

新增：

```

    } else if (inBlock && i + 1 < line.length() && chars[i] == '/' && chars[i + 1] == '/') {
        i++;
        break; // 处理单行注释的情况
    }

```

修改逻辑：

```

    if (inBlock && newline.length() > 0) {
        if (!inBlock && newline.length() > 0) {

```

3.6.3.3 TopVotedCandidate

```

//      int c = count.getDefault(p, 1); 初始为0, 在原来基础上+1
//      int c = count.getDefault(p, 0)+1;

//      lo = mi;                          //      int i = lo;
//      lo = mi+1;                        //      int i = lo-1;

//      int mi = lo + (hi - lo) / 2;
//      if (A.get(i).get(mi).time < t)
//      if (A.get(i).get(mi).time <= t)//
//          lo = mi + 1;

//      int j = Math.max(lo, 0);
//      int j = Math.max(lo-1, 0);

```

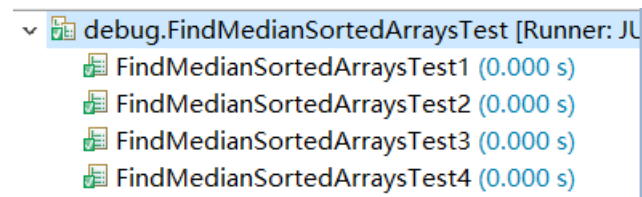
3.6.4 结果

3.6.4.1 findMedianSortedArrays

```

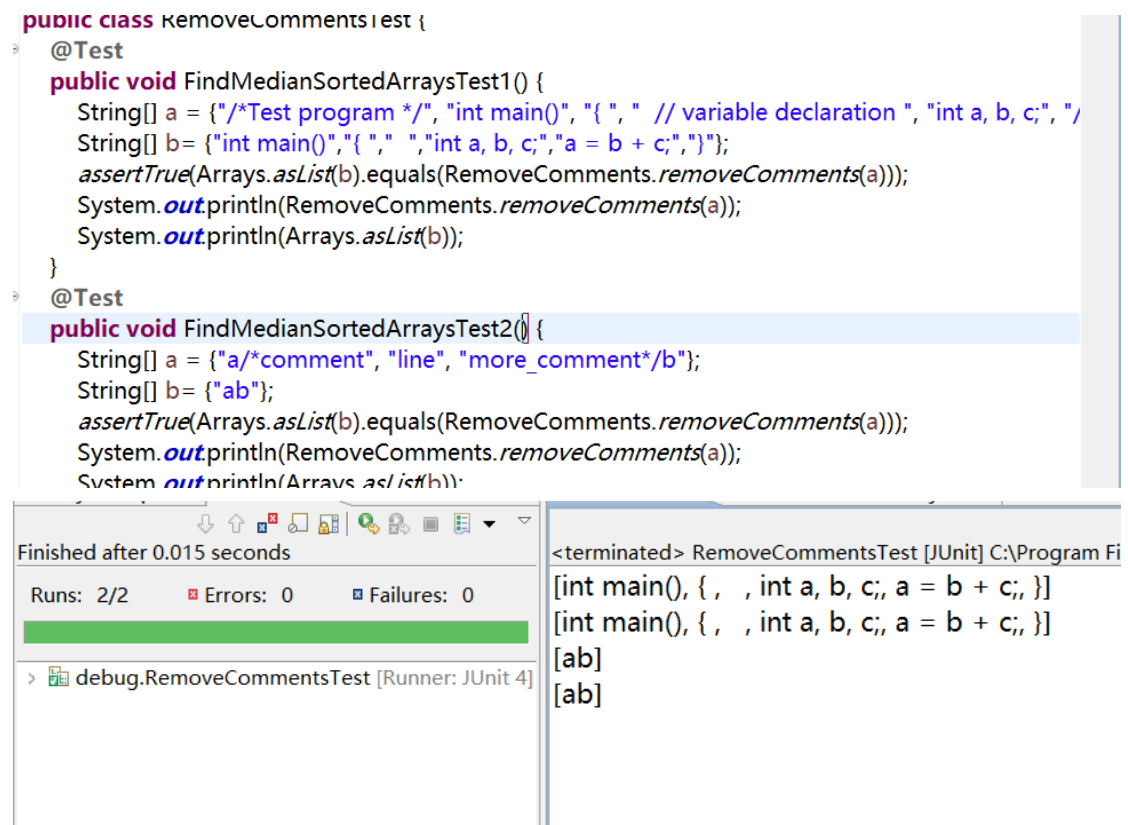
7 public class FindMedianSortedArraysTest {
3 @Test
9 public void FindMedianSortedArraysTest1() {
1     int[] a = { 1, 3 };
1     int[] b = { 2 };
2     assertTrue(FindMedianSortedArrays.findMedianSortedArrays(a, b) == 2.0);
3 }
4
5 @Test
5 public void FindMedianSortedArraysTest2() {
7     int[] a = { 1, 2 };
3     int[] b = { 3, 4 };
9     assertTrue(FindMedianSortedArrays.findMedianSortedArrays(a, b) == 2.5);
1    }
1
2 @Test
3 public void FindMedianSortedArraysTest3() {
4     int[] a = { 1, 1, 1 };
5     int[] b = { 5, 6, 7 };
5     assertTrue(FindMedianSortedArrays.findMedianSortedArrays(a, b) == 3.0);
7 }
3
9 @Test
1 public void FindMedianSortedArraysTest4() {
1     int[] a = {1,1};
2     int[] b = {1,2,3};
3     assertTrue(FindMedianSortedArrays.findMedianSortedArrays(a, b) == 1.0);
4 }

```

3.6.4.2 removeComments

直接放上测试代码:



3.6.4.3 TopVotedCandidate

```

7 public class TopVotedCandidateTest {
3   @Test
3   public void TopVotedCandidateTest1() {
0       int[] persons = { 0, 1, 1, 0, 0, 1, 0 };//投票给的人
1       int[] times = { 0, 5, 10, 15, 20, 25, 30 };//投票的时间
2
3       int[] input = { 3, 12, 25, 15, 24, 8 };//输入的时间点
4       int[] output = { 0, 1, 1, 0, 0, 1 };//输出的获胜者
5       TopVotedCandidate topVotedCandidate = new TopVotedCandidate(persons, times);
6       for (int i = 0; i < 6; i++) { //循环检验
7           assertEquals(topVotedCandidate.q(input[i]), output[i]);
3       }
3   }
}

```

debug.TopVotedCandidateTest [Runner: JUnit 4] (0.000 s)

TopVotedCandidateTest1 (0.000 s)

4 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
5.12	19:00-22:00	3.1	完成
5.13	19:00-22:00	3.2	完成
5.15	19:00-22:00	3.3	完成
5.17	19:00-22:00	3.3	完成
5.17	19:00-22:00	3.4	完成
5.18	19:00-22:00	3.5	完成
5.19	8:00-22:00	3.5+3.6	完成

5 实验过程中遇到的困难与解决途径

遇到的难点	解决途径
Log4j 不会用	上网找教程
Stream 新特性	翻看核心技术

6 实验过程中收获的经验、教训、感想

6.1 实验过程中收获的经验教训

6.2 针对以下方面的感受

- (1) 健壮性和正确性，二者对编程中程序员的思路有什么不同的影响？
健壮性使程序员考虑更多异常情况。
- (2) 为了应对 1%可能出现的错误或异常，需要增加很多行的代码，这是否划算？（考虑这个反例：民航飞机上为何不安装降落伞？）
具体看是什么样的系统，如果是很重要的系统，成本再大也要保证
- (3) “让自己的程序能应对更多的异常情况”和“让客户端/程序的用户承担确保正确性的职责”，二者有什么差异？你在哪些编程场景下会考虑遵循前者、在哪些场景下考虑遵循后者？
在外部接口处倾向于第一种，如果是内部接口的话倾向于第一种。
- (4) 过分谨慎的“防御”（*excessively defensive*）真的有必要吗？如果你在完成 Lab5 的时候发现 Lab5 追求的是 I/O 大文件时的性能（时间/空间），你是否会回过头来修改你在 Lab3 和本实验里所做的各类 *defensive* 措施？
如何在二者之间取得平衡？
不太了解，希望继续学习能找到答案。
- (5) 通过调试发现并定位错误，你自己的编程经历中有总结出一些有效的方法吗？请分享之。*Assertion* 和 *log* 技术是否会帮助你更有效的定位错误？
合理使用 *Assertion* 和 *log* 十分有益
- (6) 怎么才是“充分的测试”？代码覆盖度 100%是否就意味着 100%充分的测试？
不，有些边界条件测试覆盖度无法体现。
- (7) *Debug* 一个错误的程序，有乐趣吗？
被迫阅读别人代码说实话没有什么乐趣。
- (8) 关于本实验的工作量、难度、*deadline*。
工作量太大。
- (9) 到目前为止你对《软件构造》课程的评价和建议。

有的要求太严苛。

- (10)期末考试临近，你对占成绩 60%的闭卷考试有什么期望或建议？//请严肃的提出，杜绝开玩笑，教师会认真考虑你们的建议。
希望不是考概念背诵。