

**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 | 陈鋆 |
| 学号 | 1170300513 |
| 班号 | 11703005 |
| 电子邮件 | 295951960@qq.com |
| 手机号码 | 18059077917 |

**目录**

[1 实验目标概述 1](#_Toc3922818)

[2 实验环境配置 1](#_Toc3922819)

[3 实验过程 1](#_Toc3922820)

[3.1 Error and Exception Handling 1](#_Toc3922821)

[3.2 Assertion and Defensive Programming 19](#_Toc3922822)

[3.2.1 checkRep()检查invariants 19](#_Toc3922823)

[3.2.2 Assertion保障pre-/post-condition 24](#_Toc3922824)

[3.3 Logging 26](#_Toc3922825)

[3.3.1 写日志 26](#_Toc3922826)

[3.3.2 日志查询 27](#_Toc3922827)

[3.4 Testing for Robustness and Correctness 29](#_Toc3922828)

[3.4.1 Testing strategy 29](#_Toc3922829)

[3.4.2 测试用例设计 30](#_Toc3922830)

[3.4.3 测试运行结果与EclEmma覆盖度报告 36](#_Toc3922831)

[3.5 SpotBugs tool 36](#_Toc3922832)

[3.6 Debugging 38](#_Toc3922833)

[3.6.1 理解待调试程序的代码思想 38](#_Toc3922834)

[3.6.2 发现并定位错误的过程 39](#_Toc3922835)

[3.6.3 如何修正错误 40](#_Toc3922836)

[3.6.4 结果 44](#_Toc3922837)

[4 实验进度记录 46](#_Toc3922838)

[5 实验过程中遇到的困难与解决途径 47](#_Toc3922839)

[6 实验过程中收获的经验、教训、感想 47](#_Toc3922840)

[6.1 实验过程中收获的经验和教训 47](#_Toc3922841)

[6.2 针对以下方面的感受 48](#_Toc3922842)

# 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

⚫ Assertion 和防御式编程

⚫ 日志

⚫ 调试技术

⚫ 黑盒测试及代码覆盖度

# 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。

在这里给出你的GitHub Lab4仓库的URL地址（Lab4-学号）。

https://github.com/ComputerScienceHIT/Lab4-1170300513.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！）。

## Error and Exception Handling

**⚫ 输入文件中存在不符合语法规则的语句：**

**语法错误的定义：**

**AtomStructure：**

1. 元素名称不符合要求。如：



1. 轨道数目非正整数。如：



1. ::= 的前缀或后缀不符合要求。如：





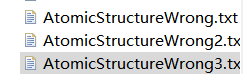
1. 轨道号非整数。如：

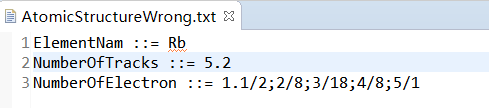


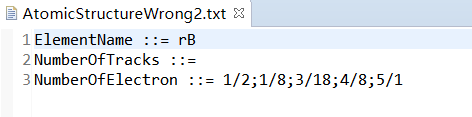
1. 电子数目非整数。如：

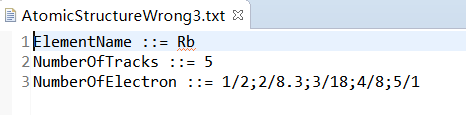


错误文档的构造：









**StellarSystem：**

1. 恒星的半径、质量不符合number的定义。如：



1. 行星的行星半径、轨道半径、公转速度不符合number的定义。如：



1. 初始角度不在[0,360)的区间内，如：



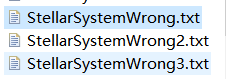
1. 公转方向不属于CW或CCW中的一个。如：

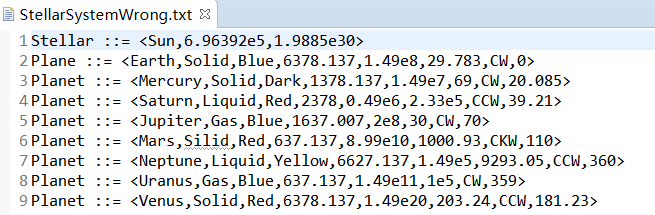


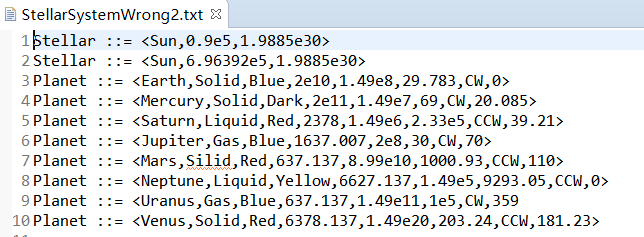
1. ::= 的前缀或后缀不符合要求。如：

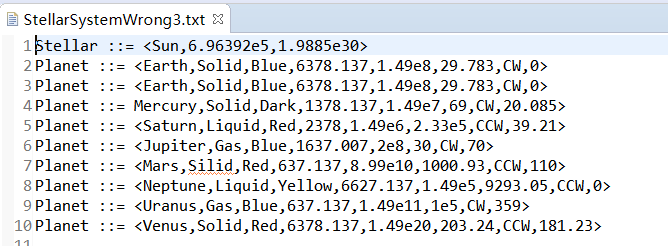


错误文档的构造：









**SocialNetworkCircle：**

1. ::= 的前缀或后缀不符合要求。如：



1. CentralUser的年龄非正整数。如：



1. CentralUser的性别不属于M或F之一。如：



1. Friend的年龄非正整数。如：



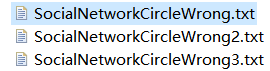
1. Friend的性别不属于M或F之一。如：

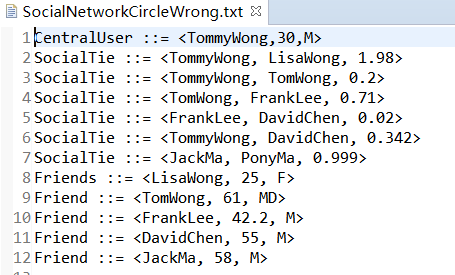


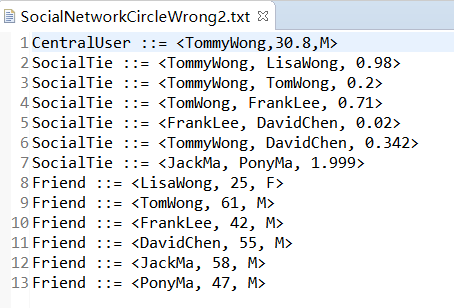
1. 社交亲密度的值不在[0,1]的区间上。如：

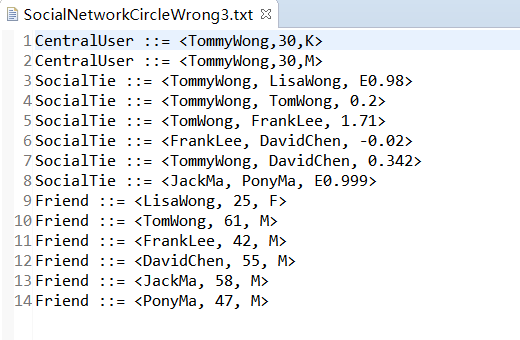


错误文档的构造：





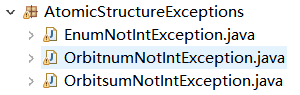




**Exception的设置与捕获：**

**AtomStructure：**

异常设置：





异常的捕获：

在使用正则表达式进行文本解析的过程中（每一行每一行地单独进行解析），程序一旦发现上面所描述的语法异常，应捕获该异常，进行异常处理。

另外，在读文件的while循环内进行try-catch，使得程序能够汇总所有出现的错误。在该try-catch结构内如果发现问题就抛出相应异常，交由该结构统一处理。

对异常的统一处理以及while内的try-catch结构代码如下：

**while**((line = br.readLine()) != **null**)

{

**try** {

Matcher m1 = Pattern.*compile*(format1).matcher(line);

Matcher m2 = Pattern.*compile*(format2).matcher(line);

Matcher m3 = Pattern.*compile*(format3).matcher(line);

**while**(m1.find())

{

elename = m1.group(2);

flag1 = 1;

}

**while**(m2.find())

{

**try**

{

orbitnum1 = Integer.*valueOf*(m2.group(2));

}**catch** (Exception e) {

**throw** **new** OrbitsumNotIntException();

}

flag2 = 1;

}

**while**(m3.find())

{

String[] couples = m3.group().split("NumberOfElectron ::= |;");

orbitnum2 = couples.length - 1;

**for**(**int** i = 1 ; i < couples.length ; i++)

{

String[] numbers = couples[i].split("/");

**int** num1 = 0, num2 = 0;

**try**

{

num1 = Integer.*valueOf*(numbers[0]);

}**catch** (Exception e) {

**throw** **new** OrbitnumNotIntException();

}

Track t = Track.*Roundcreator*(num1);

**super**.addorbit(t);

**try** {

num2 = Integer.*valueOf*(numbers[1]);

} **catch** (Exception e) {

**throw** **new** EnumNotIntException();

}

**for**(**int** j = 1 ; j <= num2 ; j++)

{

PhysicalObject p = PhysicalObject.*Q3creator*();

**super**.addtoorbit((E)p, t);

}

}

}

m1 = Pattern.*compile*(format1).matcher(line);

m2 = Pattern.*compile*(format2).matcher(line);

m3 = Pattern.*compile*(format3).matcher(line);

**if**((!m1.find()) && (!m2.find()) && (!m3.find()))

**throw** **new** SyntaxErrorException(linecounter,line);

linecounter++;

}**catch** (TheSameTagRException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : " + e.showwrong());

}**catch** (SyntaxErrorException e) {

eflag = 1;

String suf = "line " + e.getlinenum() + ": " + e.getline() + " : ";

String[] words = e.getline().split(" ");

**if**(words[0].equals("ElementName"))

log.severe(suf + "Element names do not meet grammatical requirements.");

**else** **if**((!words[0].equals("NumberOfTracks")) && (!words[0].equals("NumberOfElectron")))

log.severe(suf + "Prefix name mismatch.");

**else**

log.severe(suf + "Suffix mismatch.");

}**catch** (OrbitsumNotIntException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : The number of orbits is not an integer.");

}**catch** (OrbitnumNotIntException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : Orbital number is not an integer.");

}**catch** (EnumNotIntException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + ": The number of electrons is not an integer.");

}**finally** {

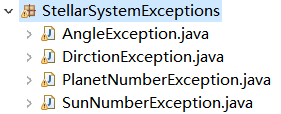
linecounter++;

}

}

**StellarSystem：**

异常设置：





异常的捕获：

在使用正则表达式进行文本解析的过程中（每一行每一行地单独进行解析），程序一旦发现上面所描述的语法异常，应捕获该异常，进行异常处理。

另外，在读文件的while循环内进行try-catch，使得程序能够汇总所有出现的错误。在该try-catch结构内如果发现问题就抛出相应异常，交由该结构统一处理。

对异常的统一处理以及while内的try-catch结构代码如下：

**while**((line = br.readLine()) != **null**)

{

**try**

{

Matcher m1 = Pattern.*compile*(format1).matcher(line);

Matcher m2 = Pattern.*compile*(format2).matcher(line);

Matcher m3 = Pattern.*compile*(format3).matcher(line);

**while**(m1.find())

{

String[] couples = m1.group().split("CentralUser ::= <|,|>");

**int** num;

**try** {

num = Integer.*valueOf*(couples[2]);

} **catch** (Exception e) {

**throw** **new** CUAgesException();

}

**if**( (!couples[3].equals("M")) && (!couples[3].equals("F")))

**throw** **new** CUSexException();

center = CentralObject.*Q5creator*(couples[1], couples[3], num);

**super**.addcenterobject((L)center);

}

**while**(m2.find())

{

String[] couples = m2.group().split("Friend ::= <|, |>");

**int** num;

**try** {

num = Integer.*valueOf*(couples[2]);

} **catch** (Exception e) {

**throw** **new** FriendAgesException();

}

**if**( (!couples[3].equals("M")) && (!couples[3].equals("F")))

**throw** **new** FriendSexException();

physicalObjects.add(PhysicalObject.*Q5creator*(couples[1], couples[3], num));

}

m1 = Pattern.*compile*(format1).matcher(line);

m2 = Pattern.*compile*(format2).matcher(line);

m3 = Pattern.*compile*(format3).matcher(line);

**if**((!m1.find()) && (!m2.find()) && (!m3.find()))

**throw** **new** SyntaxErrorException(linecounter,line);

linecounter++;

}**catch** (SyntaxErrorException e) {

eflag = 1;

String suf = "line " + e.getlinenum() + " " + e.getline();

String[] words = e.getline().split(" ");

**if**((!words[0].equals("CentralUser")) && (!words[0].equals("Friend")) && (!words[0].equals("SocialTie")))

log.severe(suf + " : Prefix name mismatch.");

**else**

log.severe(suf + " : Syntax mismatch.");

}**catch** (CUAgesException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : The CentralUser's number of ages is not an integer.");

}**catch** (FriendAgesException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : The Friend's number of ages is not an integer.");

}**catch** (CUSexException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : The CentralUser's sex is illegal.");

}**catch** (FriendSexException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : The Friend's sex is illegal.");

}**finally** {

linecounter++;

}

}

//第二遍扫描再读SocialTie

file = **new** File(pathname);

reader = **new** InputStreamReader(**new** FileInputStream(file));

br = **new** BufferedReader(reader);

linecounter = 1;

**while**((line = br.readLine()) != **null**)

{

**try**

{

Matcher m1 = Pattern.*compile*(format1).matcher(line);

Matcher m2 = Pattern.*compile*(format2).matcher(line);

Matcher m3 = Pattern.*compile*(format3).matcher(line);

**while**(m3.find())

{

String[] couples = m3.group().split("SocialTie ::= <|, |>");

**if**(couples[1].equals(center.getname()))

{

**double** v = 0;

**try** {

v = Double.*valueOf*(couples[3]);

} **catch** (Exception e) {

**throw** **new** IntimacyNumException();

}

**if**(v < 0 || v > 1)

**throw** **new** IntimacyNumException();

**int** flag = 0;

**for**(PhysicalObject p : physicalObjects)

{

**if**(p.getname().equals(couples[2]))

{

myaddcontactco((L)center, (E)p, v);

flag = 1;

}

}

**if**(flag == 0)

**throw** **new** SocialTieLogicalException(couples[2]);

}

**else**

{

**double** v = 0;

**try** {

v = Double.*valueOf*(couples[3]);

} **catch** (Exception e) {

**throw** **new** IntimacyNumException();

}

**if**(v < 0 || v > 1)

**throw** **new** IntimacyNumException();

**int** flag1 = 0 , flag2 = 0;

**for**(PhysicalObject p1 : physicalObjects)

{

**if**(p1.getname().equals(couples[1]))

{

flag1 = 1;

**for**(PhysicalObject p2 : physicalObjects)

{

**if**(p2.getname().equals(couples[2]))

{

flag2 = 1;

myaddcontactoo((E)p1, (E)p2, v);

}

}

**if**(flag2 == 0)

**throw** **new** SocialTieLogicalException(couples[2]);

}

}

**if**(flag1 == 0)

**throw** **new** SocialTieLogicalException(couples[1]);

}

}

linecounter++;

}**catch** (SocialTieLogicalException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : " + e.showwrong());

}**catch** (IntimacyNumException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : The intimacy is illeagl.");

}**catch** (NullPointerException e) {

eflag = 1;

log.severe("line " + linecounter+ " : Lack of CentralUser.");

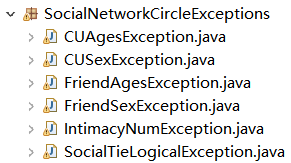
}**finally** {

linecounter++;

}

**SocialNetworkCircle：**

异常设置：





异常的捕获：

在使用正则表达式进行文本解析的过程中（每一行每一行地单独进行解析），程序一旦发现上面所描述的语法异常，应捕获该异常，进行异常处理。

另外，在读文件的while循环内进行try-catch，使得程序能够汇总所有出现的错误。在该try-catch结构内如果发现问题就抛出相应异常，交由该结构统一处理。

对异常的统一处理以及while内的try-catch结构代码如下：

**while**((line = br.readLine()) != **null**)

{

**try**

{

Matcher m1 = Pattern.*compile*(format1).matcher(line);

Matcher m2 = Pattern.*compile*(format2).matcher(line);

**while**(m1.find())

{

String[] couples = m1.group().split("Stellar ::= <|,|>");

number r = **null**, quality = **null** ;

**try** {

r = **new** number(couples[2]);

quality = **new** number(couples[3]);

} **catch** (Exception e) {

**throw** **new** SunNumberException();

}

CentralObject center = CentralObject.*Q2creator*(couples[1], r.doubleValue(), quality.doubleValue());

**super**.addcenterobject((L)center);

}

**while**(m2.find())

{

String[] couples = m2.group().split("Planet ::= <|,|>");

number v = **null**, objectr = **null** , orbitr = **null**;

**try**

{

objectr = **new** number(couples[4]);

orbitr = **new** number(couples[5]);

v = **new** number(couples[6]);

} **catch** (Exception e) {

**throw** **new** PlanetNumberException();

}

**double** angle\_360 = Double.*valueOf*(couples[8]) , angle;

**if**(angle\_360 >= 0 && angle\_360 < 360)

{

angle = angle\_360 ;

}

**else**

{

**throw** **new** AngleException(angle\_360);

//angle = (((angle\_360%360)+360)%360) ;

}

**if**((!couples[7].equals("CW")) && (!couples[7].equals("CCW")))

**throw** **new** DirctionException();

Track t = Track.*Roundcreator*(orbitr.doubleValue());

**super**.addorbit(t);

PhysicalObject object = PhysicalObject.*Q2creator*(couples[1], couples[2], couples[3], objectr.doubleValue(), v.doubleValue(), couples[7]);

addtoorbit((E)object, t);

Position<E> p = **new** Position<E>((E)object , orbitr.doubleValue() , angle);

**super**.positions.add(p);

}

m1 = Pattern.*compile*(format1).matcher(line);

m2 = Pattern.*compile*(format2).matcher(line);

**if**((!m1.find()) && (!m2.find()))

**throw** **new** SyntaxErrorException(linecounter, line);

linecounter++;

}**catch** (TheSameTagRException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : " + e.showwrong());

}**catch** (SyntaxErrorException e) {

eflag = 1;

String suf = "line " + e.getlinenum() + ": " + e.getline();

String[] words = e.getline().split(" ");

**if**((!words[0].equals("Stellar")) && (!words[0].equals("Planet")))

log.severe(suf + " : Prefix name mismatch.");

**else**

log.severe( suf + " : Syntax mismatch.");

}**catch** (AngleException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : " + e.getangle().doubleValue() + " is not in [0,360).");

}**catch** (SunNumberException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : Radius and quality do not meet the number's requirement.");

}**catch** (PlanetNumberException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line +

" : Amounts of planets that do not meet the number's requirement.");

}**catch** (DirctionException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line + " : Direction does not meet the requirements.");

}**catch** (UnsupportedOperationException e) {

eflag = 1;

log.severe("line " + linecounter + ": " + line +" : The add operation causes the system to be illegal.");

}**finally** {

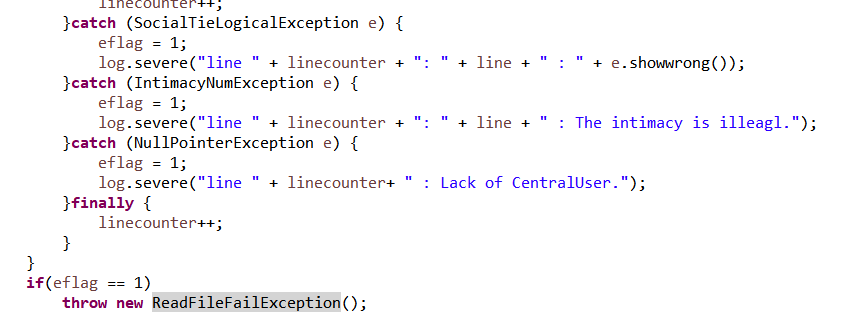
linecounter++;

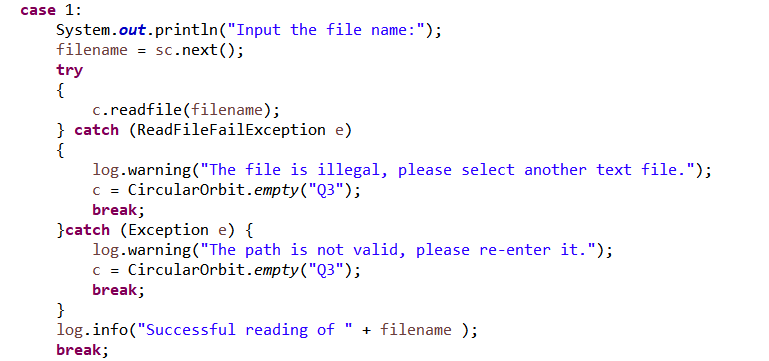
}

**异常的处理：**

三种应用均一致：结束此文件的读取，将当前所遇到的不合法之处汇总起来提示给用户，并让用户选择其他文本文件。

具体到代码层面的处理，设置eflag变量，一旦出现上述语法错误则改变其值，在读完文档后抛出ReadFileFailException交由main函数来提示用户选择其他文本文件。





**⚫ 存在标签完全一样的元素。**

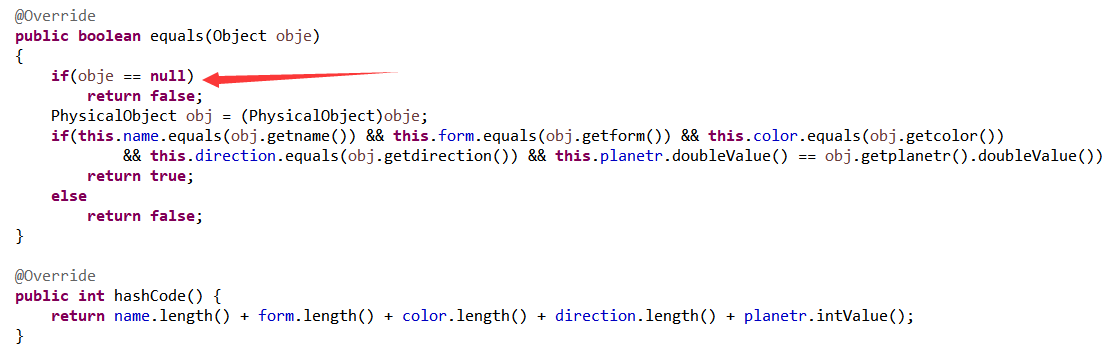
三种应用均一致。

设置异常：

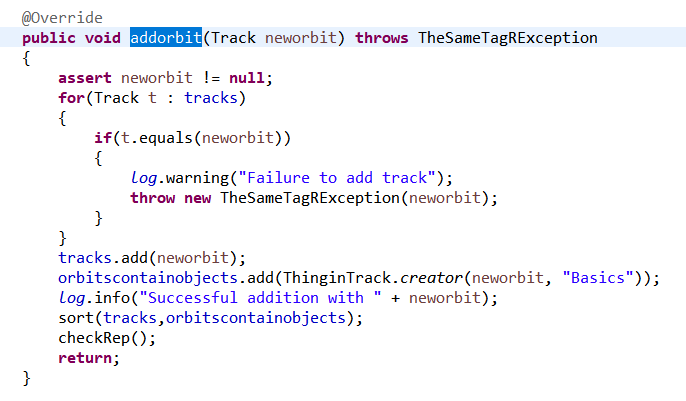


并在ConcreteCircularOrbit.java的添加轨道、轨道物体、中心物体的方法中添加相应的equal判断，如存在标签相同情况，则抛出异常。并且该方法也throw此异常，将其交由readfile方法统一进行捕获处理。

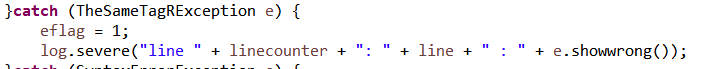
equals方法（需要添加相应的Hashcode）：



异常的抛出（部分例子）：



异常的捕获处理：

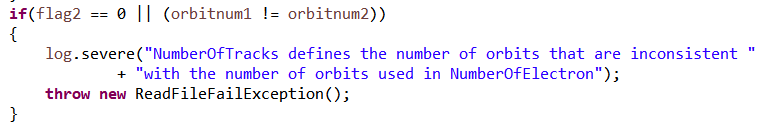


**⚫ 文件中各元素之间的依赖关系不正确。**

**AtomStructure：**

在AtomStructure中，NumberOfTracks定义的轨道数量和 NumberOfElectron 中使用的轨道数量不一致。

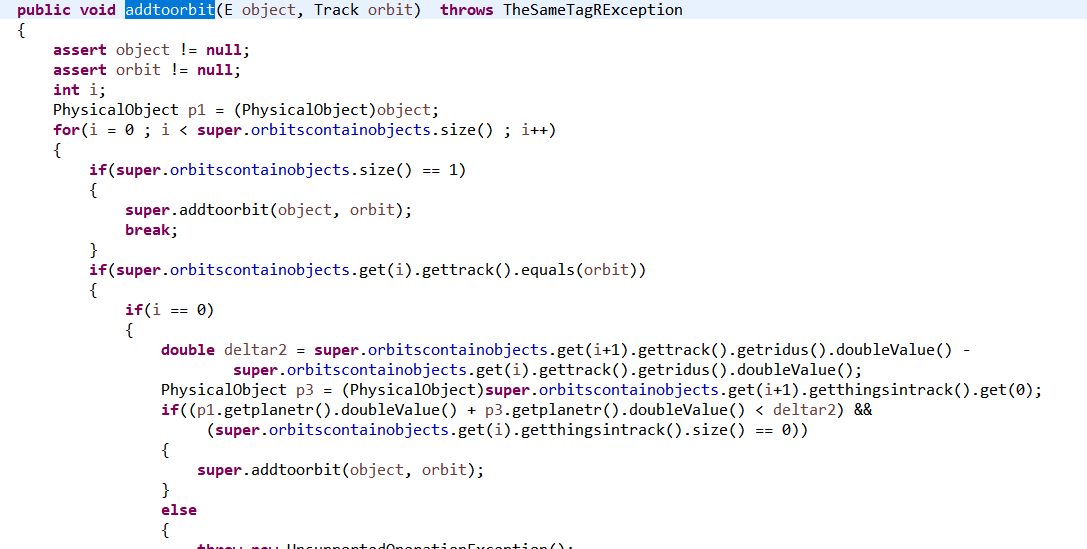
在代码实现上，读文件时记录两个数据，读完后进行比对，如果不一致则进行提示并抛出ReadFileFailException交由main函数来提示用户选择其他文本文件。



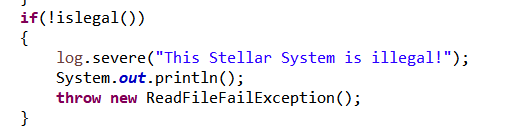
**StellarSystem：**

在StellarSystem中，中心点必须有一颗恒星；一个轨道上只能有一个行星且不能没有行星；相邻轨道的半径之差不能小于两颗相应行星的半径之和。

在代码实现上，在addtoorbit方法中如果发现上述问题，则抛出异常，在islegal中则是通过返回值为false或true来判断是否符合要求，最后在readfile方法中进行提示并抛出ReadFileFailException交由main函数来提示用户选择其他文本文件。



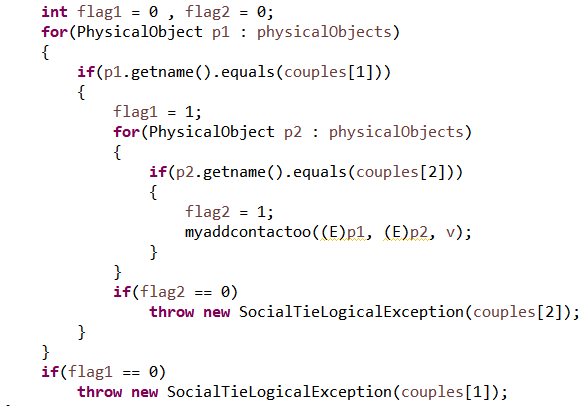


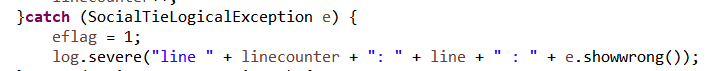


**SocialNetworkCircle：**

在SocialNetworkCircle 中没有用 Friend 定义就在 SocialTie中使用。

在代码实现上，直接在readfile方法内判断是否存在没有用 Friend 定义就在 SocialTie中使用的情况。

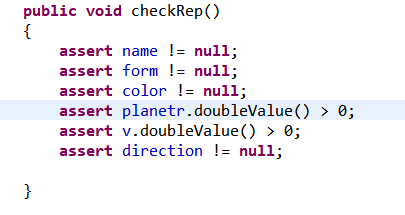


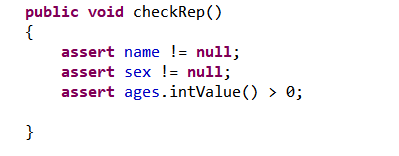


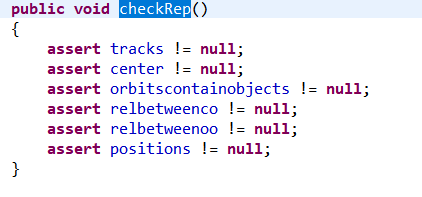
## Assertion and Defensive Programming

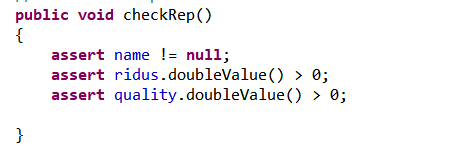
### checkRep()检查invariants

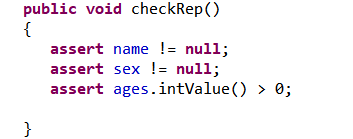
在大多数的类中，只需要保证类中的变量非空以及相应的数字在区间范围内即可。故checkRep函数如下（仅取部分，具体AF、RI已写于代码中）：





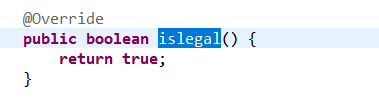






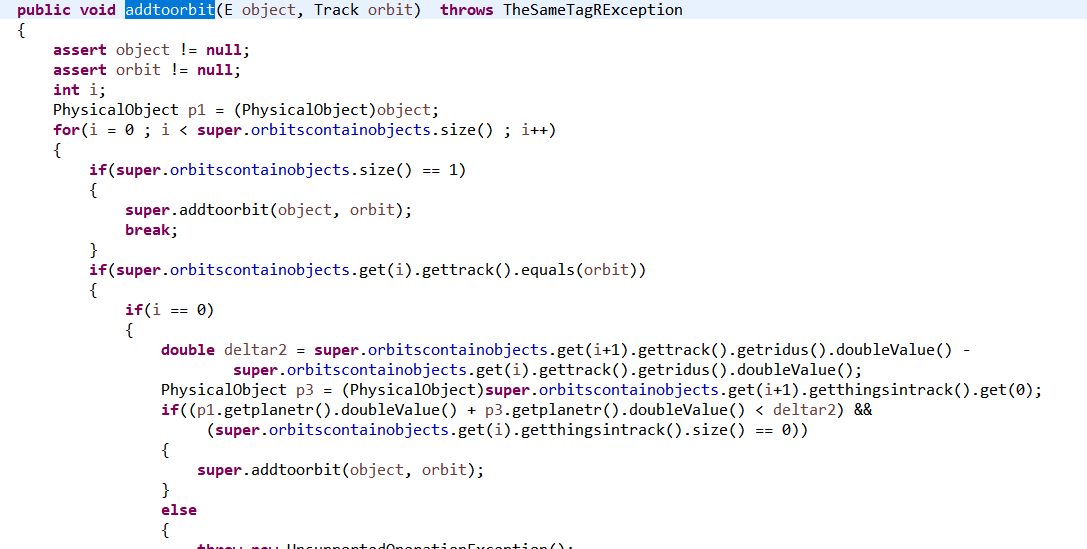
另外，针对三个应用的一些固有性质，在Lab3中已经做了足够多的非checkRep方法的“防御工作”。

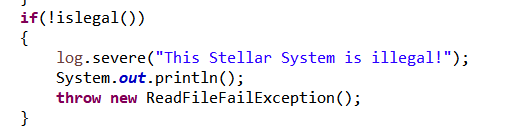
AtomStructure：无。



StellarSystem：中心点必须有一颗恒星；一个轨道上只能有一个行星且不能没有行星；相邻轨道的半径之差不能小于两颗相应行星的半径之和。

在addtoorbit方法中如果发现上述问题，则抛出异常，在islegal中则是通过返回值为false或true来判断是否符合要求，最后在readfile方法中进行提示并抛出ReadFileFailException交由main函数来提示用户选择其他文本文件。





SocialNetworkCircle：如果某个人出现在第 n 条轨道上，那么他和中心点的人之间的最短路径是 n。如果某个人与中心点用户不连通，则不应出现在轨道系统中。

在readfile方法的添加中进行相应限制，并且自定义reset方法，在添加、删除操作后使用该方法，使得整个系统始终符合RI。

**private** **void** reset()

{

**int** max = 0 , i , j;

**for**(i = 0 ; i < objects.size() ; i++)

{

E e = objects.get(i);

**int** logic;

**if**((logic = getminlogicdisfromc(e)) != -1 && max < logic)

{

max = logic;

}

}

**for**(i = 1 ; i <= max ; i++)

{

**int** flag = 0;

**for**(j = 0 ; j < **super**.tracks.size() ; j++)

{

**if**(**super**.tracks.get(j).getridus().intValue() == i)

{

flag = 1;

**break**;

}

}

**if**(flag == 0)

{

Track t = Track.*Roundcreator*(i);

**try** {

**super**.addorbit(t);

} **catch** (TheSameTagRException e) {

System.***out***.println(e.showwrong());

}

}

}

**for**(i = 0 ; i < objects.size() ; i++)

{

E e = objects.get(i);

**int** logic;

**if**( (logic = getminlogicdisfromc(e)) == -1)

{

**super**.deleteorbitobj(e);

i--;

}

**else**

{

transit(e, **super**.tracks.get(logic - 1));

}

}

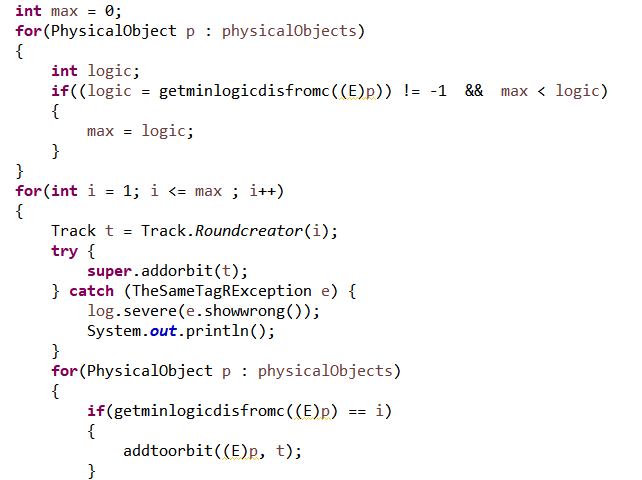
**for**(i = max ; i < **super**.tracks.size() ; i++)

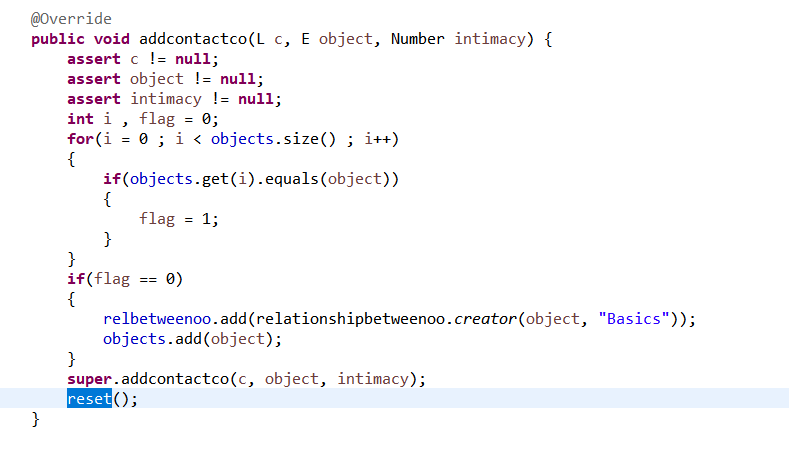
{

**super**.removeorbit(**super**.tracks.get(i));

}

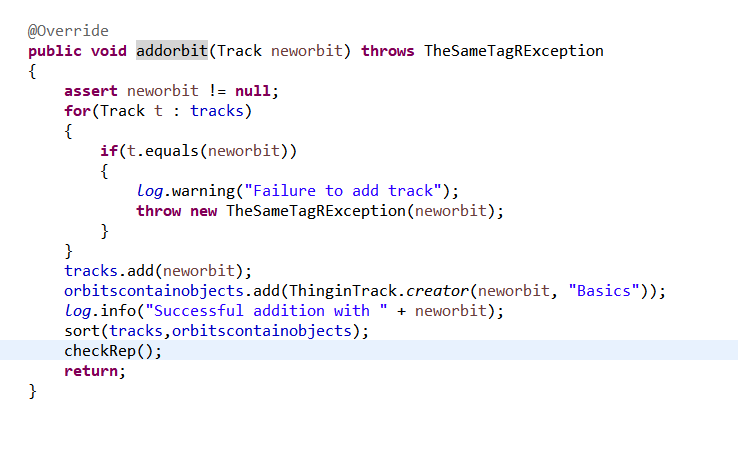
}

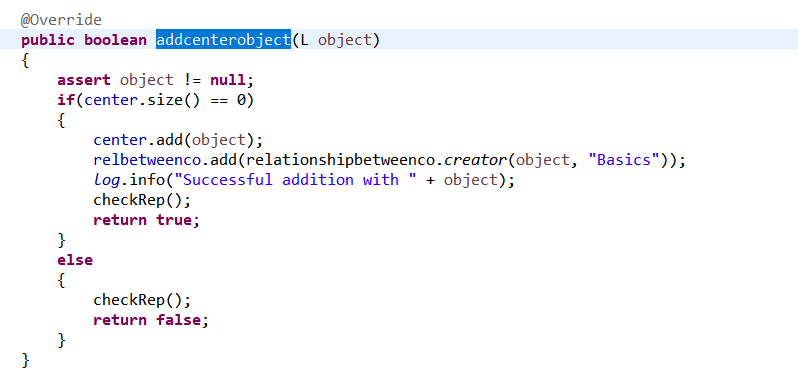


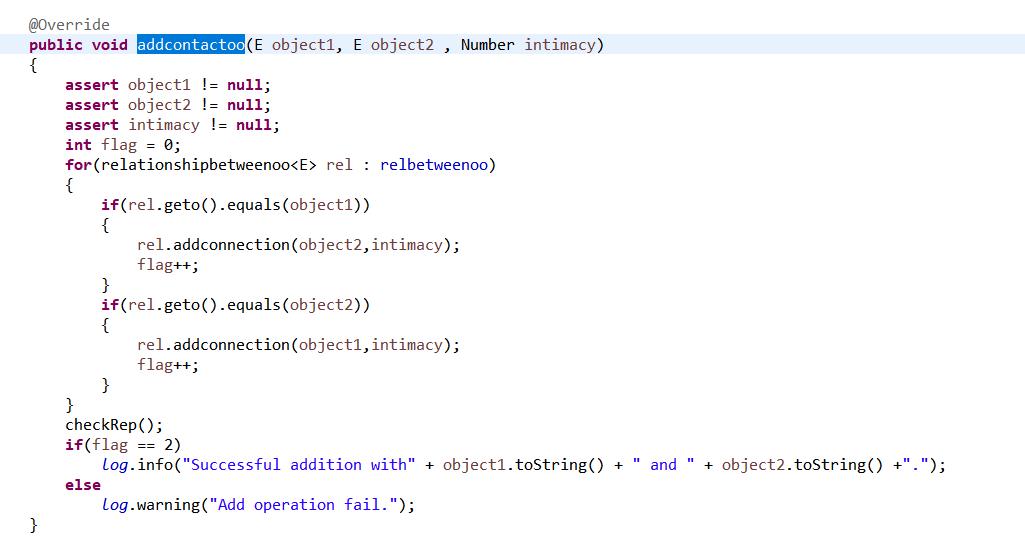


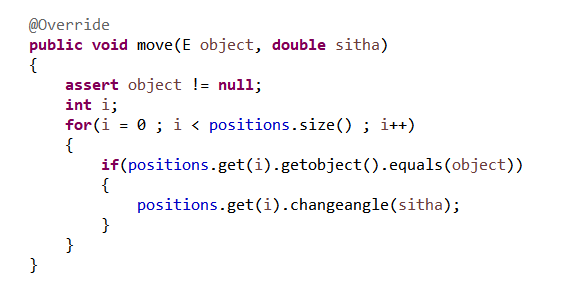
### Assertion保障pre-/post-condition

在大多数的方法中，前置条件只需要保证参数非空即可，后置条件则交由checkRep方法来检验，下面值列举出部分例子。









## Logging

### 写日志

使用 java logging，为 3.1和 3.2 节经过异常处理、错误处理、断言处理的程序增加日志功能。日志需要记录以下信息：

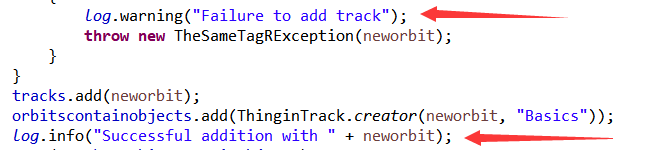
1. 所有的异常/错误：发生的时间、异常/错误类型、异常/错误发生的类名和方法名，异常/错误的具体信息、异常/错误处理的结果；
2. 对多轨道系统的所有操作，包括：读取文件、增加/删除轨道、增加中心点物体、增加/删除轨道上的物体、增加/删除物体之间的关系、跃迁等

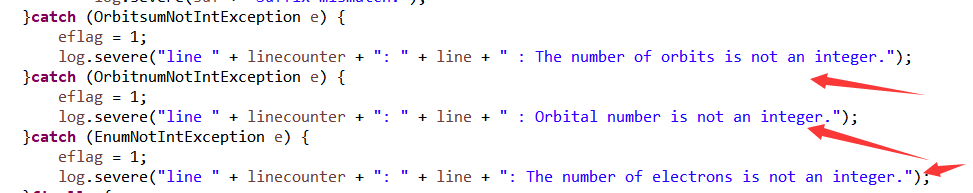
由于在Lab3中对相应的操作都以使用了打印操作来提示，故只需要将打印操作改为相应的log日志输出即可。

定义在三个应用父类ConcreteCircularOrbit类中的logger：



相应的log日志输出：





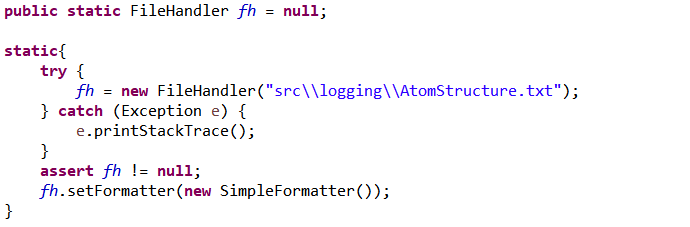
### 日志查询

为应用添加日志查询功能，用户可输入过滤条件（例如按时间段、按类型、按类、按方法、按操作类型）进行日志查询。查询结果不要直接显示原始日志记录，需要具有良好的可理解性。

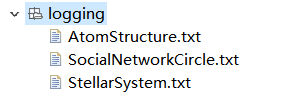
首先，为logger添加相应的Handler（调整格式后的二），使之能以相应格式输出到相应源路径的文本文件中，起到记录作用。

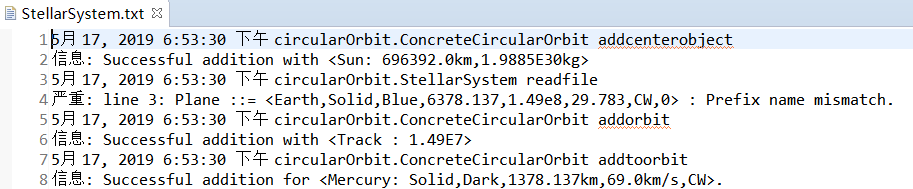
在查询操作时，从文本文件中读取内容，并且按过滤条件（例如按时间段、按类型、按类、按方法、按操作类型）进行日志查询（主要运用正则表达式与字符串匹配等操作来进行过滤）。

Handler的创建：

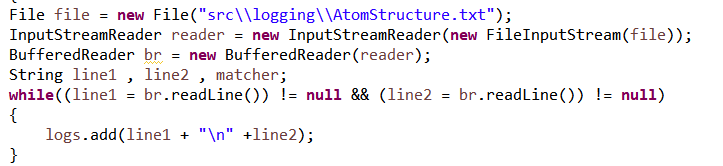


生成的相应文件：

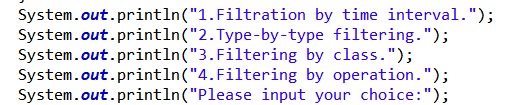


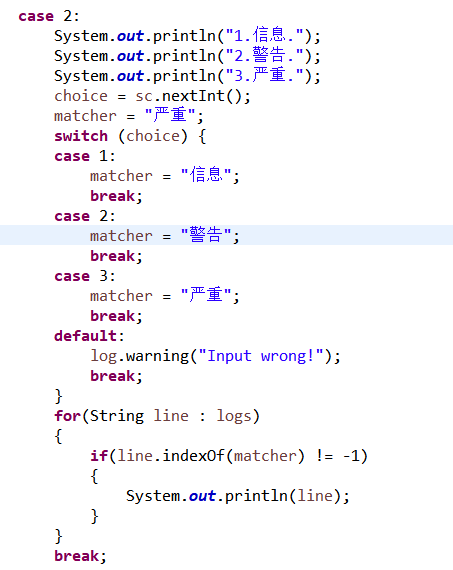


文件的读取：



筛选处理：

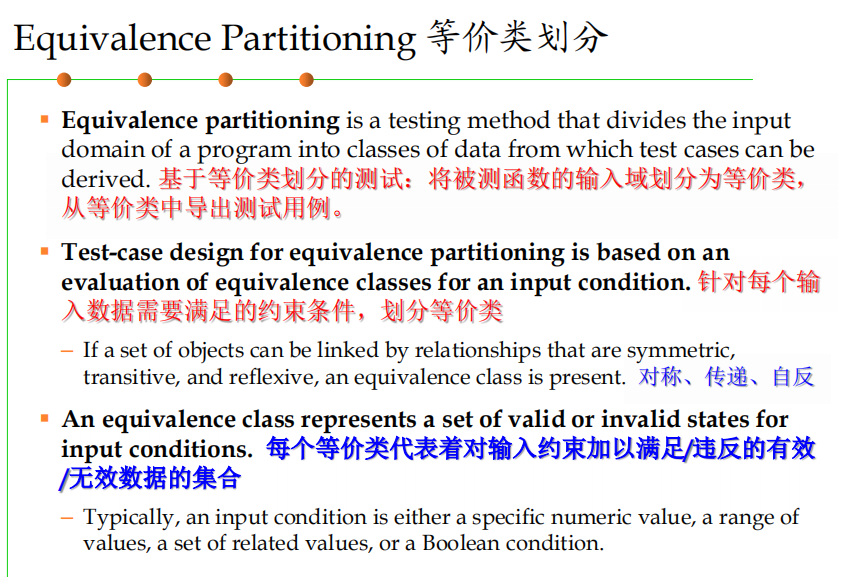


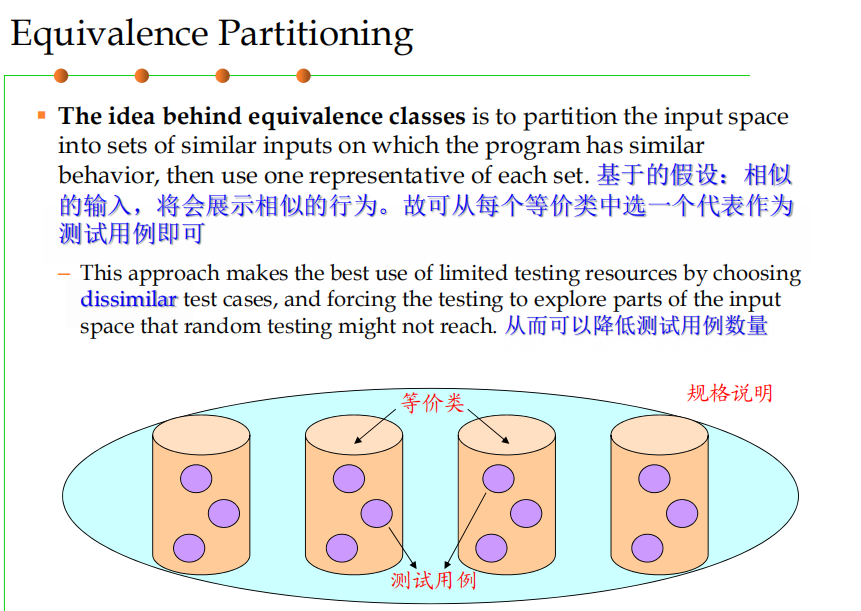


## Testing for Robustness and Correctness

### Testing strategy

使用等价类和边界值的测试思想。



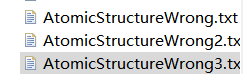


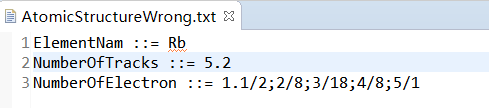
鉴于新增的情况主要出现在三个具体应用的实现CircularOrbit接口的类中，此处具体对这三个类的测试策略进行描述。

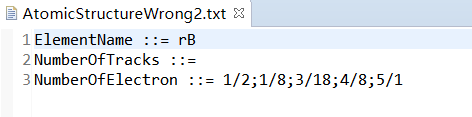
三个类中按3.1中定义的三种Exception的类型进行等价类划分，并对这些异常写了相应的三个.txt测试文件。通过读文件时捕获异常并对异常的类进行比对、对日志文件中记录的情况进行字符串比对等操作，确定原java文件是否正确。

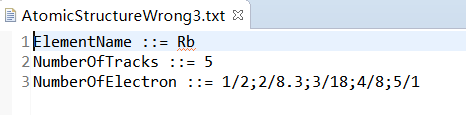
### 测试用例设计

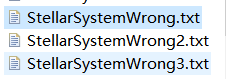
错误文档的构造：

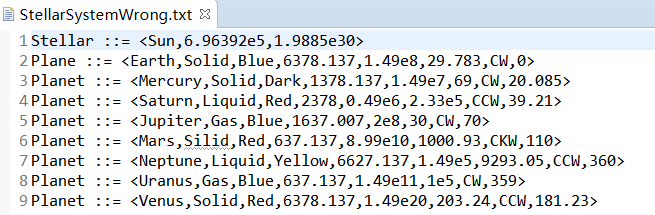


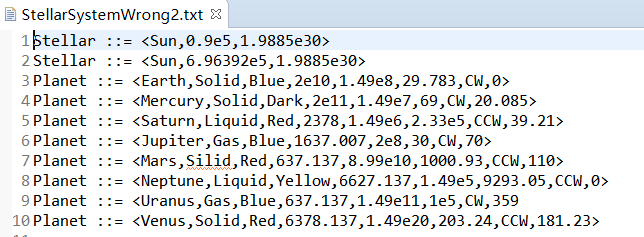


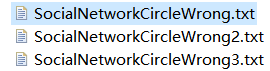


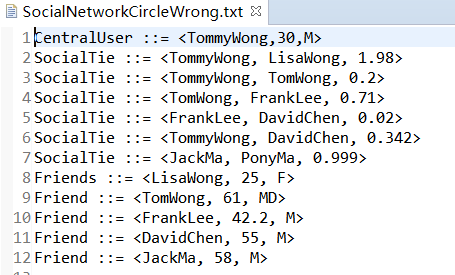


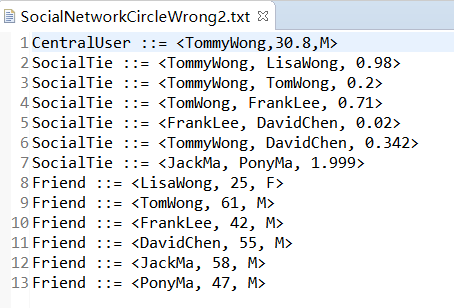


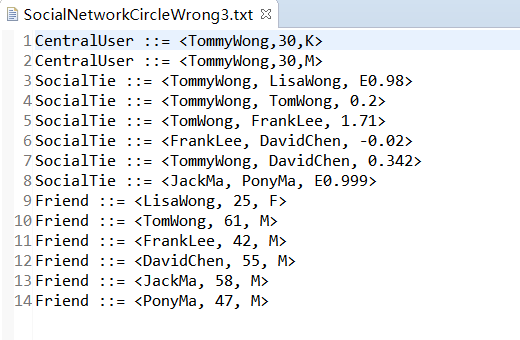












@Test

**public** **void** testWrongtxt()

{

CircularOrbit<CentralObject, PhysicalObject> cir = CircularOrbit.*empty*("Q3");

**try** {

cir.readfile("test AtomicStructureWrong.txt");

} **catch** (ReadFileFailException e) {

ReadFileFailException exp = **new** ReadFileFailException();

*assertEquals*(exp.getClass(), e.getClass());

}**catch** (Exception e) {

System.***out***.println("The path is not valid, please re-enter it.");

}

cir = CircularOrbit.*empty*("Q3");

**try** {

cir.readfile("test AtomicStructureWrong2.txt");

} **catch** (ReadFileFailException e) {

ReadFileFailException exp = **new** ReadFileFailException();

*assertEquals*(exp.getClass(), e.getClass());

}**catch** (Exception e) {

System.***out***.println("The path is not valid, please re-enter it.");

}

cir = CircularOrbit.*empty*("Q3");

cir.islegal();

**try** {

cir.readfile("test AtomicStructureWrong3.txt");

} **catch** (ReadFileFailException e) {

ReadFileFailException exp = **new** ReadFileFailException();

*assertEquals*(exp.getClass(), e.getClass());

}**catch** (Exception e) {

System.***out***.println("The path is not valid, please re-enter it.");

}

}

@Test

**public** **void** testWrongtxt()

{

CircularOrbit< CentralObject, PhysicalObject> cir = CircularOrbit.*empty*("Q5");

**try** {

cir.readfile("test SocialNetworkCircleWrong.txt");

} **catch** (ReadFileFailException e) {

ReadFileFailException exp = **new** ReadFileFailException();

*assertEquals*(exp.getClass(), e.getClass());

}**catch** (Exception e) {

System.***out***.println("The path is not valid, please re-enter it.");

}

**try** {

cir.islegal();

} **catch** (Exception e) {

// **TODO**: handle exception

}

cir = CircularOrbit.*empty*("Q5");

**try** {

cir.readfile("test SocialNetworkCircleWrong2.txt");

} **catch** (ReadFileFailException e) {

ReadFileFailException exp = **new** ReadFileFailException();

*assertEquals*(exp.getClass(), e.getClass());

}**catch** (Exception e) {

System.***out***.println("The path is not valid, please re-enter it.");

}

cir = CircularOrbit.*empty*("Q5");

**try** {

cir.readfile("test SocialNetworkCircleWrong3.txt");

} **catch** (ReadFileFailException e) {

ReadFileFailException exp = **new** ReadFileFailException();

*assertEquals*(exp.getClass(), e.getClass());

}**catch** (Exception e) {

System.***out***.println("The path is not valid, please re-enter it.");

}

cir = CircularOrbit.*empty*("Q5");

**try** {

cir.addcenterobject(**null**);

} **catch** (Throwable e) {

// **TODO**: handle exception

}

**try** {

cir.deletecentralobj(**null**);

} **catch** (Throwable e) {

// **TODO**: handle exception

}

**try** {

cir.deletecontactco(**null**, **null**);

} **catch** (Throwable e) {

// **TODO**: handle exception

}

**try** {

cir.deleteorbitobj(**null**);

} **catch** (Throwable e) {

// **TODO**: handle exception

}

**try** {

cir.deletecontactoo(**null**, **null**);

} **catch** (Throwable e) {

// **TODO**: handle exception

}

**try** {

cir.addtoorbit(**null**, **null**);

} **catch** (Throwable e) {

// **TODO**: handle exception

}

**try** {

cir.addcontactoo(**null**, **null**, **null**);

} **catch** (Throwable e) {

// **TODO**: handle exception

}

**try** {

cir.addcontactco(**null**, **null**, **null**);

} **catch** (Throwable e) {

// **TODO**: handle exception

}

}

@Test

**public** **void** testWrongtxt()

{

CircularOrbit< CentralObject, PhysicalObject> cir = CircularOrbit.*empty*("Q2");

**try** {

cir.readfile("test StellarSystemWrong.txt");

} **catch** (ReadFileFailException e) {

ReadFileFailException exp = **new** ReadFileFailException();

*assertEquals*(exp.getClass(), e.getClass());

}**catch** (Exception e) {

System.***out***.println("The path is not valid, please re-enter it.");

}

cir = CircularOrbit.*empty*("Q2");

**try** {

cir.readfile("test StellarSystemWrong2.txt");

} **catch** (ReadFileFailException e) {

ReadFileFailException exp = **new** ReadFileFailException();

*assertEquals*(exp.getClass(), e.getClass());

}**catch** (Exception e) {

System.***out***.println("The path is not valid, please re-enter it.");

}

cir = CircularOrbit.*empty*("Q2");

**try** {

cir.readfile("test StellarSystemWrong3.txt");

} **catch** (ReadFileFailException e) {

ReadFileFailException exp = **new** ReadFileFailException();

*assertEquals*(exp.getClass(), e.getClass());

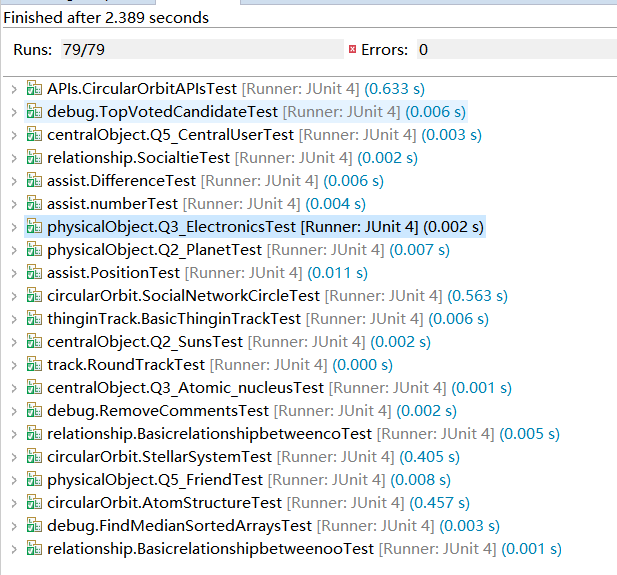
}**catch** (Exception e) {

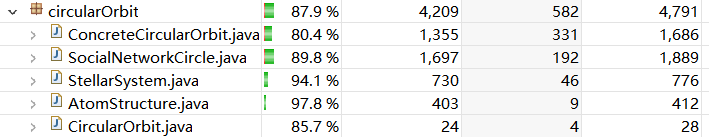
System.***out***.println("The path is not valid, please re-enter it.");

}

}

### 测试运行结果与EclEmma覆盖度报告

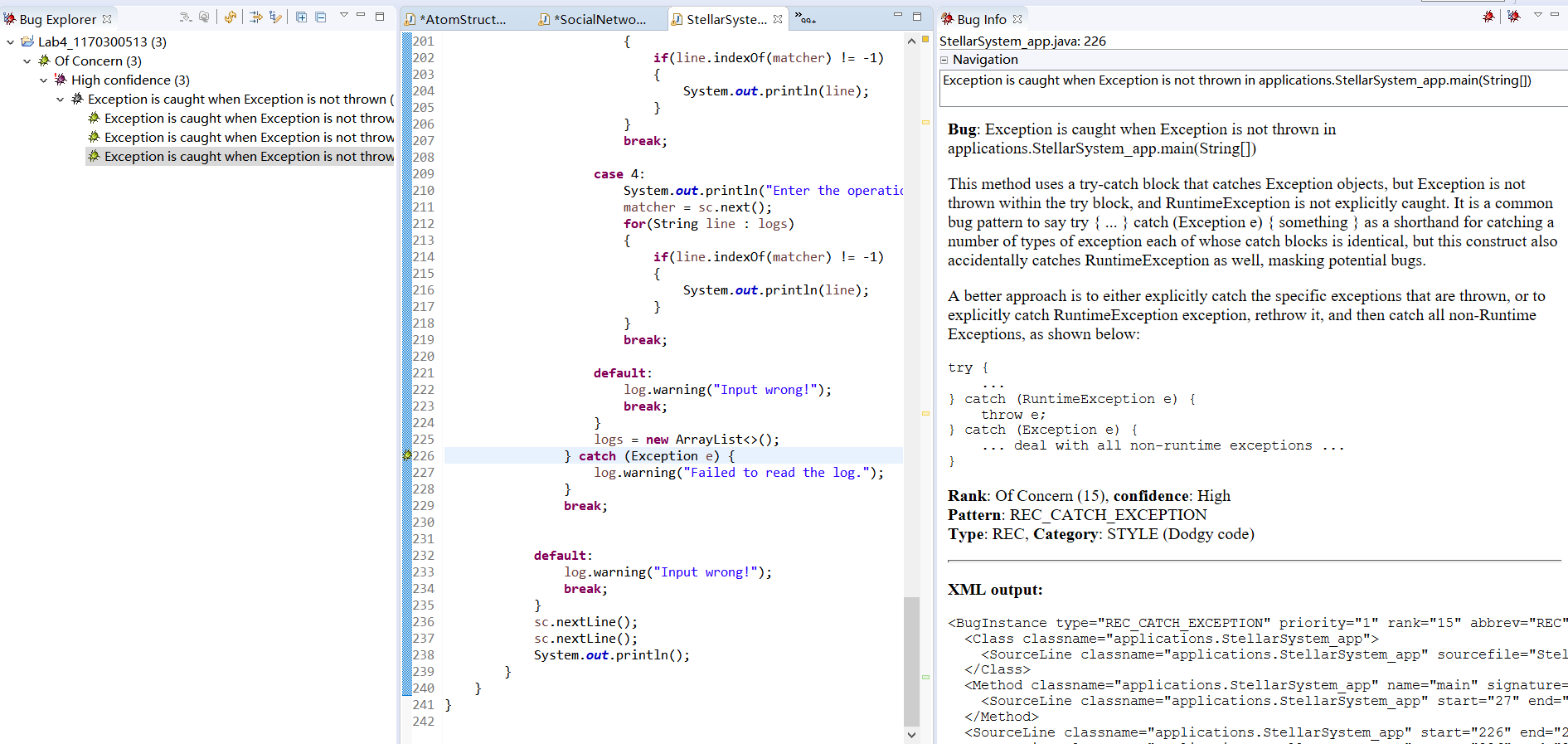




## SpotBugs tool

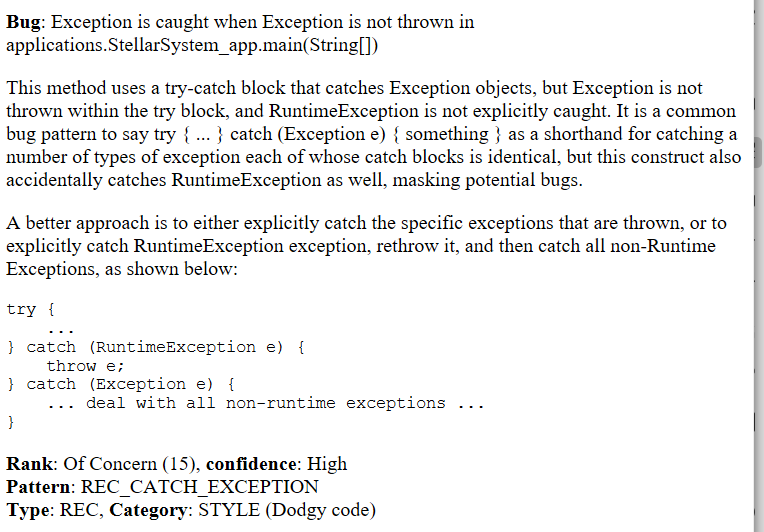
发现了哪些错误，每种错误代表什么不良的编程习惯

对代码修改，消除这些错误。



整体检测后发现这一个错误（出现在三个不同的应用中）

bug描述：



说明RuntimeException与Exception的处理最好要分开进行，否则会有很大的安全隐患，因为RuntimeException多是不可处理的，程序代码中出现的错误，而Exception是可以处理的，可预见的错误，二者不可同日而语。

## Debugging

### 理解待调试程序的代码思想

**FindMedianSortedArrays.java：**

Given two ordered integer arrays nums1 and nums2, with size m and n

Find out the median (double) of the two arrays.

You may suppose nums1 and nums2 cannot be null at the same time.

简而言之，是要我们求两个已排序好数组的中位数。

代码思想：通过二分查找在两数组间进行比较，并分偶数和奇数两种情况，最后定为到中位数的值

**RemoveComments.java：**

This program is used for removing all the comments in a program code.

这道题目要求我们去除代码中的注释，得到新的一段代码。

代码思想：通过bool型变量inBlock作为开关，一旦检测到注释开始的信号（注释符号），则关闭新代码的生成，检测到注释结束的信号后，则开启新代码的生成，以此生成不含注释的新代码。

**TopVotedCandidate.java：**

In an election, the i-th vote was cast for persons[i] at time times[i].

Now, we would like to implement the following query function:

TopVotedCandidate.q(int t) will return the number of the person that was leading the election at time t.

Votes cast at time t will count towards our query. In the case of a tie, the most recent vote (among tied candidates) wins.

简而言之，就是要我们输入某个时间查询此时哪个人所得的票数更多。

代码思想：先生成一个投票类Vote类列表的列表（记录某个时间的投票情况），再利用二分法查询相应时间点前的投票最多情况。

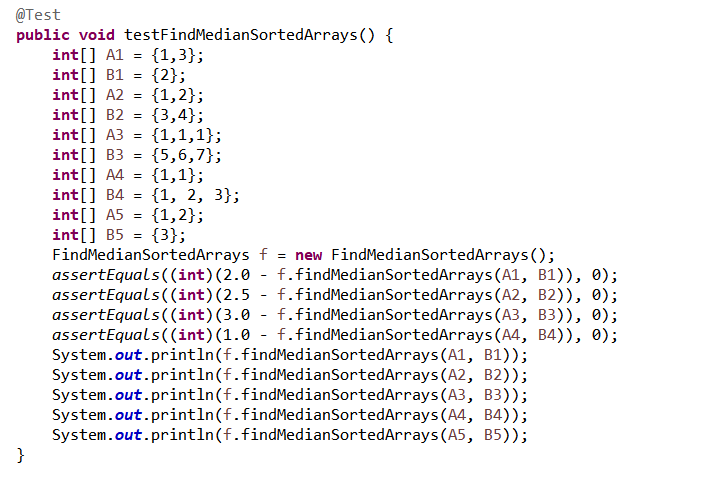
### 发现并定位错误的过程

**FindMedianSortedArrays.java：**

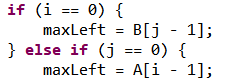
肉眼逻辑检查，发现此处按逻辑应是奇数时返回。



带入测试用例

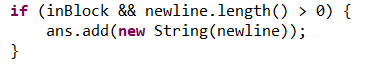


发现A5,B5、A1,B1两组用例均输出错误结果，带入代码中逻辑运行一遍发现是在此处存在边界问题。



**RemoveComments.java：**

肉眼逻辑检查，发现一个添加时开关处存在逻辑问题。



带入测试用例，发现“//”表示的注释并未去除，缺了某段代码。

并且“/\*\*/”类注释最后还会剩下一个‘/’符号。

定位问题，发现此处存在问题：

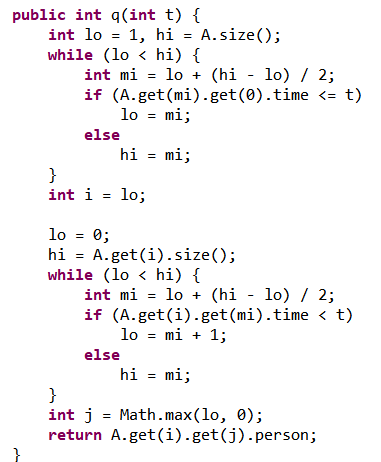


**TopVotedCandidate.java：**

肉眼观察代码逻辑，发现此处未对投票次数进行增加操作：



有测试用例可知，二分法操作亦存在边界问题（同FindMedianSortedArrays.java原理）：



### 如何修正错误

**FindMedianSortedArrays.java：**

由3.6.2中发现的错误：

奇数偶数判断问题：

修改前：

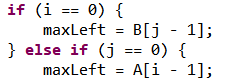


修改后：

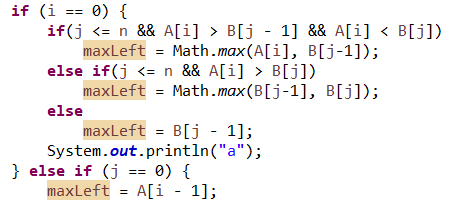


边界问题：

修改前：



修改后：

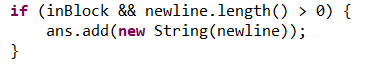


**RemoveComments.java：**

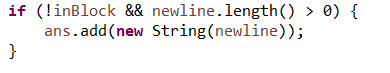
由3.6.2中发现的错误：

开关处逻辑问题：

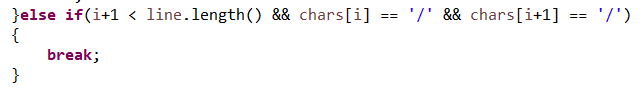
修改前：



修改后：



缺少的去除“//”注释问题：

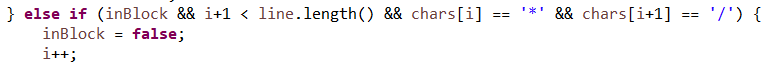


多余’/’问题：

修改前：



修改后：



**TopVotedCandidate.java：**

由3.6.2中发现的错误：

增加操作问题：

修改前：

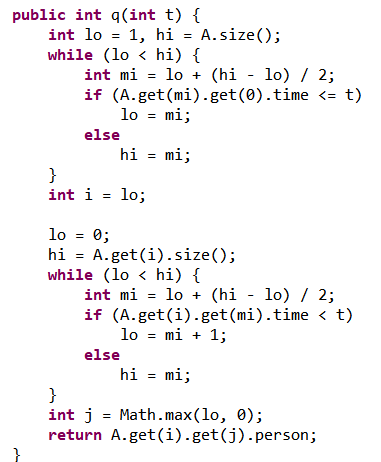


修改后：

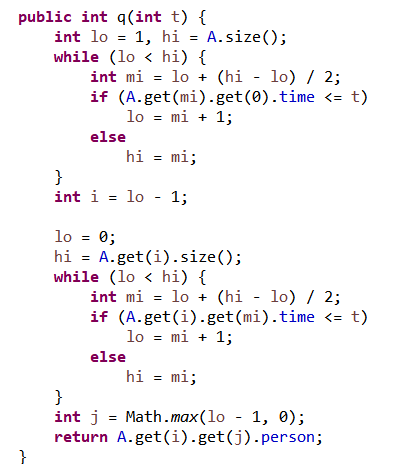


二分法问题：

修改前：



修改后：



### 结果

**FindMedianSortedArrays.java：**

test文件代码：

**public** **class** FindMedianSortedArraysTest {

@Test

**public** **void** testFindMedianSortedArrays() {

**int**[] A1 = {1,3};

**int**[] B1 = {2};

**int**[] A2 = {1,2};

**int**[] B2 = {3,4};

**int**[] A3 = {1,1,1};

**int**[] B3 = {5,6,7};

**int**[] A4 = {1,1};

**int**[] B4 = {1, 2, 3};

**int**[] A5 = {1,2};

**int**[] B5 = {3};

FindMedianSortedArrays f = **new** FindMedianSortedArrays();

*assertEquals*((**int**)(2.0 - f.findMedianSortedArrays(A1, B1)), 0);

*assertEquals*((**int**)(2.5 - f.findMedianSortedArrays(A2, B2)), 0);

*assertEquals*((**int**)(3.0 - f.findMedianSortedArrays(A3, B3)), 0);

*assertEquals*((**int**)(1.0 - f.findMedianSortedArrays(A4, B4)), 0);

System.***out***.println(f.findMedianSortedArrays(A1, B1));

System.***out***.println(f.findMedianSortedArrays(A2, B2));

System.***out***.println(f.findMedianSortedArrays(A3, B3));

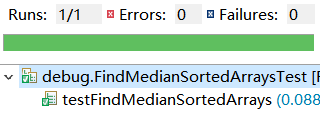
System.***out***.println(f.findMedianSortedArrays(A4, B4));

System.***out***.println(f.findMedianSortedArrays(A5, B5));

}

}

测试结果：



**RemoveComments.java：**

test文件代码：

**public** **class** RemoveCommentsTest {

@Test

**public** **void** testRemoveComments() {

String[] source1 = {

"/\*Test program \*/",

"int main()",

"{",

" // variable declaration",

"int a, b, c;",

"/\* This is a test",

"multiline" ,

"comment for" ,

"testing\*/" ,

"a = b + c;",

"}"

};

RemoveComments f = **new** RemoveComments();

List<String> answer = f.removeComments(source1);

String line1 = "int main()";

String line2 = "{";

String line3 = " ";

String line4 = "int a, b, c;";

String line5 = "a = b + c;";

String line6 = "}";

*assertEquals*(6, answer.size());

*assertEquals*(answer.get(0), line1);

*assertEquals*(answer.get(1), line2);

*assertEquals*(answer.get(2), line3);

*assertEquals*(answer.get(3), line4);

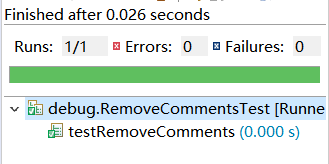
*assertEquals*(answer.get(4), line5);

*assertEquals*(answer.get(5), line6);

}

}

测试结果：



**TopVotedCandidate.java：**

test文件代码：

@Test

**public** **void** testTopVotedCandidate() {

**int**[] persons = {0,1,1,0,0,1,0};

**int**[] times = {0,5,10,15,20,25,30};

TopVotedCandidate t = **new** TopVotedCandidate(persons, times);

*assertEquals*(t.q(3), 0);

*assertEquals*(t.q(12), 1);

*assertEquals*(t.q(25), 1);

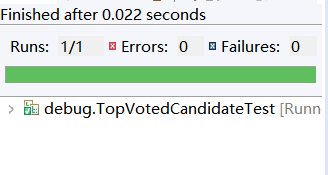
*assertEquals*(t.q(15), 0);

*assertEquals*(t.q(24), 0);

*assertEquals*(t.q(8), 1);

}

测试结果：



# 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 |
| 2019-05-06 | 15:45-18:00 | 复习课堂Exception内容并阅读实验要求 | 按时完成 |
| 2019-05-07 | 13:00-15:00 | 思考并定义3.1中的语法错误 | 部分未完成 |
| 2019-05-08 | 18:00-20:00 | 完成昨日欠缺，并完成3.1中标签操作的定义 | 按时完成 |
| 2019-05-09 | 18:00-22:00 | 在AtomStructure的readfile方法中实现异常的抛出、执行、捕获 | 按时完成 |
| 2019-05-10 | 13:00-15:00 | 在AtomStructure\_app中进行相应测试检验 | 按时完成 |
| 2019-05-11 | 18:00-23:00 | 在剩余两应用的readfile文件中实现异常的抛出、执行、捕获，并进行相应测验 | 按时完成 |
| 2019-05-12 | 18:00-22:00 | 完成3.2内容 | 按时完成 |
| 2019-05-13 | 15:45-18:00 | 研究学习3.3log日志内容 | 按时完成 |
| 2019-05-14 | 18:00-23:00 | 实现log日志的记录、查询，并检验 | 按时完成 |
| 2019-05-15 | 18:00-23:00 | 实现3.4test文件 | 按时完成 |
| 2019-05-16 | 18:00-19:00 | 完成3.5内容 | 按时完成 |
| 2019-05-17 | 18:00-23:00 | 完成3.6的debug任务 | 部分未完成 |
| 2019-05-18 | 18:00-19:00 | 完成昨日欠缺 | 按时完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| log日志的使用 | 上网查询相应的使用方法简介 |
| Exception的统筹使用 | 通过上课所学到的几种方法以及实验中的使用练习来加强这方面的能力 |
|  |  |

# 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

1.在开始进行实验前一定要完整地阅读一遍所有要求！本次实验中我就有多次为看清信息（如log日志的要求）而浪费大量时间重写已有代码。

2.不懂得知识要多上CSDN寻找相应的内容，并且大胆进行实验，例如本次的log日志的使用，其实并不难，但未知的恐惧还是让我犹豫了太久！

## 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？

健壮性需要程序员考虑各种可能出现的错误情况并将其修正，需要更周全的思路；而正确性为了保证其正确，需要程序员有更严谨、严格的思路想法。

1. 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？）

还是要综合成本、客户实际需求、错误的重要程度等条件，如果客户需求时间紧，时间成本高，那可能会不划算；但如果错误会造成严重后果（波音737MAX8的程序造成坠机），那就非常有必要添加。

1. “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？

“让自己的程序能应对更多的异常情况”：严于律己，宽于待人；“让客户端/程序的用户承担确保正确性的职责”：宽于待己，严于律人。前者：客户是不专业的小白之类的，并且知识文化水平较低，没有阅读spec的习惯；后者：客户专业程度较高，知识文化水平较高，愿意遵守spec。

1. 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？

没必要；会；看客户更注重什么，如果是军事国防等对安全要求很高的项目，更要注重防御；如果是追求速度的项目，则适当地提高性能。

1. 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？

之前都是逻辑推测以及打印来定为错误；这两个方法感觉总体上来说跟打印差不太多。

1. 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？

各种不同的边界条件都有尝试。不代表，可能程序没覆盖所有情况，或者测试用例不包括异常之类的情况。

1. Debug一个错误的程序，有乐趣吗？

如果是Debug自己写的程序思路较为清晰流畅，完成后有成就感，有乐趣；但Debug不是自己写的程序，就需要费大量时间了解作者思路，体验极差！

1. 关于本实验的工作量、难度、deadline。

工作量大，难度较高，deadline紧迫。

1. 到目前为止你对《软件构造》课程的评价和建议。

实验占用时间太多，并且有部分是跟本课程关联度较小的（如本次实验中的debug部分），以至于没时间复习课程内容，望改进！

1. 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。

能多出些与实验内容相关的内容，一来能够检验同学们实验做得是否认真、独立；二来也能增加对完成实验的积极性。