

**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 | 张钟升 |
| 学号 | 1170300431 |
| 班号 | 1703004 |
| 电子邮件 |  |
| 手机号码 | 13029801568 |

**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 2](#_Toc3922821)

[3.2 Assertion and Defensive Programming 2](#_Toc3922822)

[3.2.1 checkRep()检查invariants 2](#_Toc3922823)

[3.2.2 Assertion保障pre-/post-condition 2](#_Toc3922824)

[3.3 Logging 2](#_Toc3922825)

[3.3.1 写日志 2](#_Toc3922826)

[3.3.2 日志查询 2](#_Toc3922827)

[3.4 Testing for Robustness and Correctness 2](#_Toc3922828)

[3.4.1 Testing strategy 2](#_Toc3922829)

[3.4.2 测试用例设计 2](#_Toc3922830)

[3.4.3 测试运行结果与EclEmma覆盖度报告 2](#_Toc3922831)

[3.5 SpotBugs tool 2](#_Toc3922832)

[3.6 Debugging 3](#_Toc3922833)

[3.6.1 理解待调试程序的代码思想 3](#_Toc3922834)

[3.6.2 发现并定位错误的过程 3](#_Toc3922835)

[3.6.3 如何修正错误 3](#_Toc3922836)

[3.6.4 结果 3](#_Toc3922837)

[4 实验进度记录 3](#_Toc3922838)

[5 实验过程中遇到的困难与解决途径 3](#_Toc3922839)

[6 实验过程中收获的经验、教训、感想 4](#_Toc3922840)

[6.1 实验过程中收获的经验和教训 4](#_Toc3922841)

[6.2 针对以下方面的感受 4](#_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。

<https://github.com/ComputerScienceHIT/Lab4-1170300431.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

重载了两个Exception, ExceptionGroup和LogicErrorException.

* ExceptionGroup

1. **public** **class** ExceptionGroup **extends** RuntimeException **implements** Iterable<Exception>{
2. **private** List<Exception> exs;
4. **public** **boolean** join(Exception ex);
6. @Override
7. **public** String getMessage();
9. @Override
10. **public** String getLocalizedMessage();
12. @Override
13. **public** **void** printStackTrace();
15. **public** **int** size();
17. **public** Exception get(**int** index);
19. **public** **boolean** merge(ExceptionGroup other);
21. **public** **void** clear();
23. **public** **boolean** isEmpty();
25. @Override
26. **public** Iterator<Exception> iterator();
27. }

接口和方法如上. 显然, ExceptionGroup是作为一个Exception的容器出现的, 它主要适用于从文件读取的时候, 汇集输入错误信息.

* LogicErrorException

参考C++的logic\_error....主要用于汇报用户输入中产生的逻辑异常, 如前后矛盾等等. 它继承RuntimeException.

* IllegalArgumentException

程序中主要的错误就是参数错误, 比如不能将字符串转为数字, 抛出NumberFormatException, 它是IllegalArgumentException的子类; 不能转为枚举, 直接抛出IllegalArgumentException; 除此之外, 任何输入的参数不合法均可视为IllegalArgumentException并加入ExceptionGroup实例化的对象, 并最后抛出它.

## Assertion and Defensive Programming

### checkRep()检查invariants

在实验3中,checkRep已上传到源代码.

首先CircularOrbit留出接口checkRep, 并由子类StellarSystem, AtomStructure, SocialNetworkCircle实现该接口. 对StellarSystem, 需要检查两颗相邻行星半径小于它们的两条轨道间的距离差. 这里距离差也就是高轨道短半轴减低轨道长半轴. 另外发现似乎有行星运行在恒星内部.... 对于AtomStructure, 给一个空实现... 对SocialNetworkCircle, 利用已经存在的getLogicalDistance, 计算每个User与`中心用户的逻辑距离, 断言它等于该User的轨道半径. (与中心用户不连通的点轨道半径为-1).

### Assertion保障pre-/post-condition

使用assert和annotation分别保证运行时和静态检查. 使用assert的情况, 比如setRelation函数, 可以为PhysicalObject a和b设置关系. 此时用assert !a.equals(b)来保证a和b不相等就是应当的. 再如addTrack函数, 应当断言传入的参数个数大于等于1. 除此之外assert更多的用于查询(CircularOrbit.query, Collection.get)后的NotNull保证.

## Logging

### 写日志

用Java Logger写日志. 比较值得注意的有:

* 添加FileHandler以输出到文件;
* 重载Formatter以控制输出格式;
* getLogger得到的Logger对象都有父对象LoggerManager.root(可能吧), 这个Logger有默认的ConsoleHandler. 获取这个Handler并置Level.OFF就可以实现静默输出日志;
* 如果希望用附加的方式打开文件, 可以用相关的构造函数;
* 要注意FileHandler是应当关闭的.

### 日志查询

1. 实现两个parser类来解析info和warning的文本.
2. 分别读取两个log文件, 用两个parser得到List<List>类型的参数矩阵.
3. 用户输入查询类型和查询条件, 查找符合条件的List, 将这些参数join后就还原回了日志记录, 将这些日志记录显示出来即可.

## Testing for Robustness and Correctness

### Testing strategy

根据等价类和边界条件设计testing strategy. 要单独注意的是对抛出异常的条件测定, 有的异常并不容易出现, 尽可能地复现大多数的异常. 借助coverage工具可以发现未被覆盖的分支, 反复分析, 增加合适的测试用例.

### 测试用例设计

对大多数返回布尔值的修改操作, 比如addObject, addTrack, 可以调用多次, 一次使它返回真, 多次返回假(操作成功往往只有一种情况, 而操作失败可能有多种情况), 往往可以完全覆盖. 一些observer方法, 如findTrack, isEmpty一类的方法, 根据它们所获取的对象类型设计测试用例. 对CircularOrbitAPI中的方法, 事先确保其所调用的方法的测试通过, 再设计一个或多个情形作为测试用例.

值得注意的有两点, 一是一些GUI操作(包括main函数)无法测试; 二是对于一些protected和private方法, 一般情况下测试模块无法直接测试, 可能的解决方案有两种. 一是测试他们的公有的调用者; 当公有的调用者不便测试, 或者方法本身的功能比较独立, 对外部条件依赖不大的时候, 可以用反射的方式直接调用, 完成测试.

### 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。

## Debugging

### 理解待调试程序的代码思想

### 发现并定位错误的过程

### 如何修正错误

### 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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
|  |  |
|  |  |
|  |  |

# 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

## 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？
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？）
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？
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？
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？
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？
7. Debug一个错误的程序，有乐趣吗？
8. 关于本实验的工作量、难度、deadline。
9. 到目前为止你对《软件构造》课程的评价和建议。
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。