



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	
学号	
班号	
电子邮件	
手机号码	

目录

1 实验目标概述	1
2 实验环境配置	1
3 实验过程	1
3.1 Error and Exception Handling	1
3.2 Assertion and Defensive Programming	3
3.2.1 checkRep()检查 invariants	3
3.2.2 Assertion 保障 pre-/post-condition	3
3.3 Logging	3
3.3.1 写日志	3
3.3.2 日志查询	4
3.4 Testing for Robustness and Correctness	4
3.4.1 Testing strategy	4
3.4.2 测试用例设计	4
3.4.3 测试运行结果与 EclEmma 覆盖度报告	5
3.5 SpotBugs tool	5
3.6 Debugging	6
3.6.1 理解待调试程序的代码思想	6
3.6.2 发现并定位错误的过程	6
3.6.3 如何修正错误	6
3.6.4 结果	6
4 实验进度记录	6
5 实验过程中遇到的困难与解决途径	7
6 实验过程中收获的经验、教训、感想	7
6.1 实验过程中收获的经验教训	7
6.2 针对以下方面的感受	7

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：

- 1) 错误处理
- 2) 异常处理
- 3) Assertion
- 4) 和防御式编程
- 5) 日志
- 6) 调试技术
- 7) 黑盒
- 8) 测试
- 9) 及代码覆盖度

2 实验环境配置

2.1. 配置 FindBugs：因未找到最合适 IDEA 的 SpotBugs 的配置方法，所以使用 FindBugs 替代，直接在设置中的 plugins 里添加即可；

3 实验过程

3.1 Error and Exception Handling

1. 定义错误：

1.1.TrackGame 语法错误

- 1) 标签不存在：此处文件中只能存在三种标签：Athlete, Game, NumOfTracks, 其他的都是不合法的标签；
- 2) 顺序错误：每一个 Athlete 所在行的内容未按照姓名、编号、国家、年龄、最佳成绩排序；
- 3) 类型错误：年龄、编号必须是整形数，其他均不合法；

- 4) 语法不规范: 每行格式只可以是 `Game ::= int, Athlete ::= <String, int, String, int, double>, NumOfTracks ::= int`

1.2. AtomStructure 语法错误

- 1) 未知标签;
- 2) 结构不符
- 3) 电子数和轨道数为匹配
- 4) 数据类型不正确

1.3. SocialNetworkCircle 语法错误

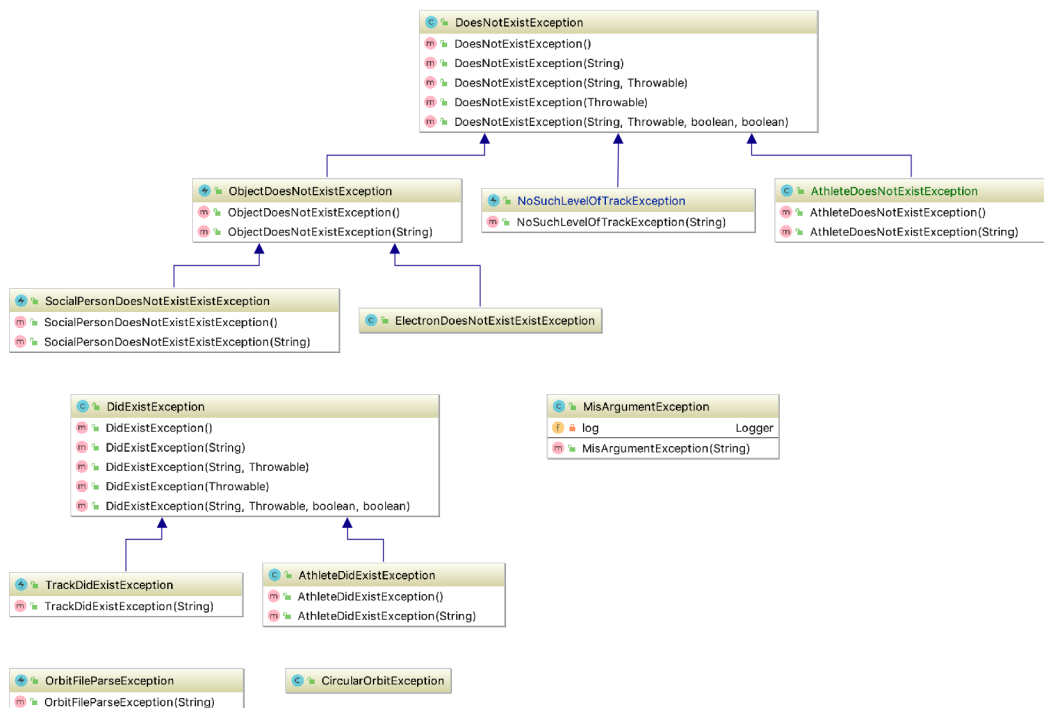
- 1) 出现未知标签
- 2) 文件内容缺失: 例如人际关系网络中, 中心人物是必不可少的, 但是如果文件中未定义中心人物, 则该错误属于文件内容缺失。
- 3) 参数不全
- 4) 数据类型不正确
- 5) 顺序不正确

1.4. 其他错误

- 6) 文件内容重复: 相同人名/标签重复出现等错误。
- 7) 文件不存在
- 8) 文件内容关系依赖冲突

2. 处理错误:

定义多个异常类并形成异常树, 如图所示:



3.2 Assertion and Defensive Programming

3.2.1 checkRep()检查 invariants

定义 RI:

- 1) AtomStructure: true
- 2) TrackGame: 每组比赛的人数不超过跑道数
- 3) SocialNetworkCircle: 图中第 i 层轨道上的人与中心点的逻辑距离为 i

3.2.2 Assertion 保障 pre-/post-condition

- 1) 部分方法没有使用 assert, 而是使用 Exception 告知用户, 例如人际关系网络中, 用户向网络中添加的关系中的来源是不存在的, 会以上面定义的 SocialPersonDoesNotExistException 异常抛出给客户端处理, 要求执行新的输入;
- 2) 每个方法都有条件检查, 例如输入名称不存在、或为 null 等任何妨碍程序正确运行的输入都会被阻挡, 不会执行后面的操作;
- 3) 在 ConcreteCircularOrbit 中创建了诸如 boolean containsObject(E object), boolean containsLevel(int level) 等方法辅助判断输入是否符合要求, 如果这个输入不合法但是能够重新输入, 则会以 Checked Exception 形式抛出给客户端, 如果错误不能挽回, 则会 assert false 或抛出 Unchecked Exception 结束程序;

3.3 Logging

3.3.1 写日志

采用 log4j 第三方库来实现日志功能, 该工具将日志分为多个等级: debug, info, warn, error 来区别重要度、记录信息或帮助 debug, 配置文件内容如下:

```
log4j.rootLogger=INFO, Console,file
```

```
log4j.appender.Console=org.apache.log4j.ConsoleAppender
log4j.appender.Console.Target=System.out
log4j.appender.Console.layout=org.apache.log4j.PatternLayout
log4j.appender.Console.layout.ConversionPattern=[%p] [%d{yyyy-MM-dd
HH\:mm\:ss,SSS}] [%l] %m%n
```

```
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=logs/lfasr-sdk-client.log
log4j.appender.file.MaxFileSize=128MB
log4j.appender.file.MaxBackupIndex=50
log4j.appender.file.Append=true
log4j.appender.file.Threshold=INFO
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%5p] %l - %m%n
```

3.3.2 日志查询

3.4 Testing for Robustness and Correctness

3.4.1 Testing strategy

为 ADT 和 Application 设计了如下的测试:



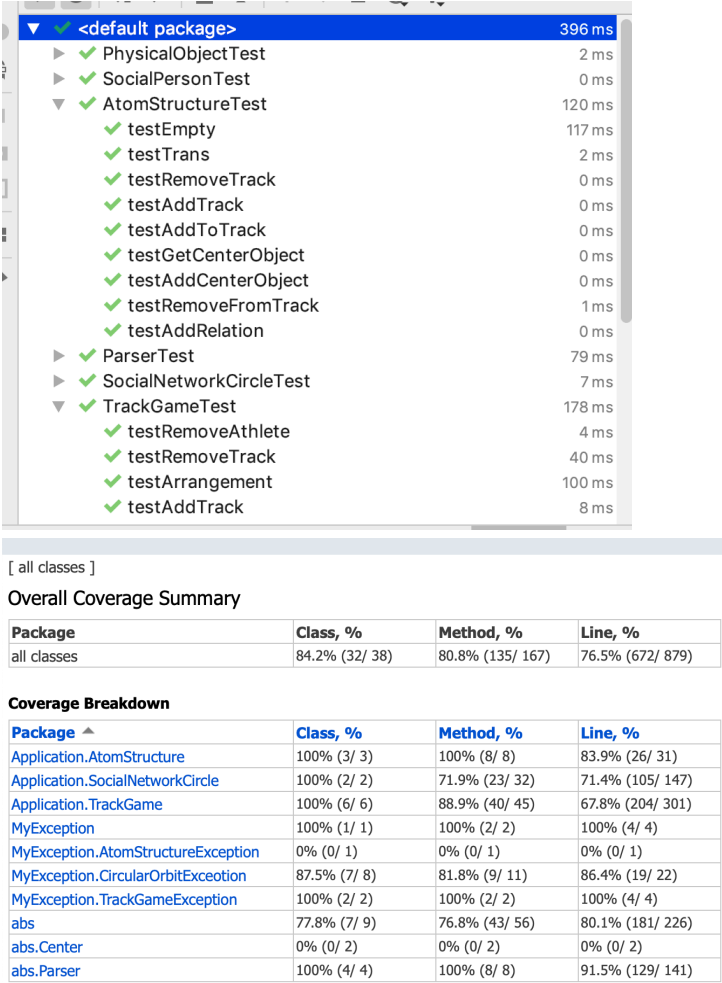
其中包括了 PhysicalObject、三个应用和 ADT、文件解析器的每个方法的测试

3.4.2 测试用例设计

总共包含 43 个测试，每个测试根据方法返回值使用 `assert` 断言，同时所有测试都尽可能包含正面和反面的测试用例，或考虑多种情况。例如原子轨道模型增加轨道方法中，第一次添加第 i 层轨道时，方法能够正确执行，返回 `true`，如果再次添加第 i 层轨道。会抛出 `TrackDidNotExistException`，如果添加的轨道层数是负数，会抛出 `IllegalArgumentException`。

3.4.3 测试运行结果与 EcEmma 覆盖度报告

测试结果如图，40 余个测试全部通过



3.5 SpotBugs tool

- 1) 判断奇偶数不适用于负数
- 2) 遍历 Map 时，使用 entrySet 比 keySet 效率更高
- 3) 储存只使用了一次的变量

就像(2)中的问题，如果没有仔细研究过 Java 对 Map 的实现的话，是根本不会知道的，而这个工具帮助我发现了这个潜在的影响效率的问题，因此在以后的学习和开发中，我会更加重视对这种工具的使用。

3.6 Debugging

3.6.1 理解待调试程序的代码思想

运用单步执行找出与预期不同的步骤，重新整理逻辑，修改代码，将它带向正确的步骤。

3.6.2 发现并定位错误的过程

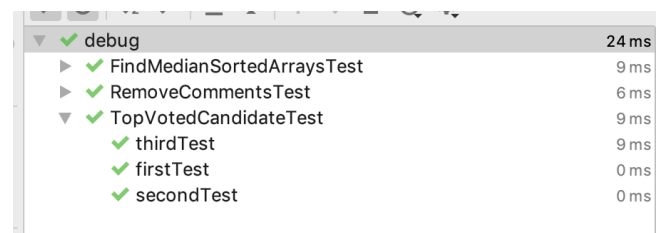
- (1) 发现 `findMedianSortedArrays` 方法不能获得正确的 `halfLen` 并且奇偶数判断不正确
- (2) `removeComments` 方法开始就存在容易越界的错误，然后运行可发现有的内容被错误的舍弃了
- (3) `TopVotedCandidate` 考虑的情况较少，且健壮性不足

3.6.3 如何修正错误

根据上述错误进行修正即可。

3.6.4 结果

测试全都通过



✓ debug	24 ms
▶ ✓ FindMedianSortedArraysTest	9 ms
▶ ✓ RemoveCommentsTest	6 ms
▼ ✓ TopVotedCandidateTest	9 ms
✓ thirdTest	9 ms
✓ firstTest	0 ms
✓ secondTest	0 ms

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
5.8	8:00-20:00	写完	只完成了异常部分
5.13	8:00-20:00	写完	完成了 log

5.14	8:00-20:00	写完	修复新发现的 bug
5.20	8:00-20:00	写完	修复 bug
5.21	8:00-20:00	写完	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
生成人际关系网络的速度过慢	重新设计了 Concrete 的 ADT，极大的改善了运行速度，将 Medium 的计算时间从 4-5 分钟减少到 1 秒钟。

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

- 1) 异常的强大：定义多种异常能够帮助我们区分问题，归类问题，以便更好地处理；

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？
正确性只要保证在正确输入下能得到正确结果，但是健壮性要求程序员预测出部分的非法输入，并且加以防范。
- (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？）
需要评估这个修复这个错误能否带来更大的收益，否则可能得不偿失，民航飞机就是这个道理，如果都加上降落伞，既浪费了空间，还大幅增加了成本。
- (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？
前者使程序员的职责更加繁重，要求能够对付部分不正确的输入，然后和客户端协商处理，而后者发现不正确的输入，退出就好了。当可能出现的错误比较容易预测时我会选择前者，如果代价较高时会选择后者。
- (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?

和 3 中所述有共同之处, 如果代价较高时, 这种防御得不偿失 (在保证正确性的前提下), 所以我会修改部分防御措施。

- (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? 多多利用方法调用栈的信息帮助快速定位问题发生的源头, 巧妙利用单步调试来重现问题发生的过程, 辅以 assertion 和 log 技术能够帮助还原问题发生的现场。

- (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?

充分的测试是要求考虑了每个等价类, 并且还得到了正确结果的测试, 覆盖度 100%仅代表代码 100%都运行了, 不代表没有隐含的错误。

- (7) Debug 一个错误的程序, 有乐趣吗?

很枯燥, 很无趣

- (8) 关于本实验的工作量、难度、deadline。

量适中, 难度适中, deadline 太早。

- (9) 到目前为止你对《软件构造》课程的评价和建议。

课程本身很好, 但模式上有种说不出的奇怪问题……

- (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。

支持包含纸质、闭卷的考试, 但分值过高。实验占用的时间远远远远超过了上课的时间 (时间和任务量不均衡), 但却只占用一小部分分数, 最后决定成绩的却是占用只两个小时的手写试卷, 给人的感觉就是: 努力了那么久, 写了成千上万行的代码在分数面前却微不足道, 有种丧失写实验的动力感觉。这是一堂理论与实践结合, 但更偏向实践的课程, 不应该遵循高考模式。