

**2020年春季学期  
计算机学院《软件构造》课程**

**Lab 4实验报告**

|  |  |
| --- | --- |
| 姓名 | 张乙 |
| 学号 | 1180301022 |
| 班号 | 1803010 |
| 电子邮件 | 1430268665@qq.com |
| 手机号码 | 15839452898 |

**目录**

[1 实验目标概述 1](#_Toc41566416)

[2 实验环境配置 1](#_Toc41566417)

[3 实验过程 2](#_Toc41566418)

[3.1 Error and Exception Handling 2](#_Toc41566419)

[3.1.1 处理输入文本中的三类错误 2](#_Toc41566420)

[3.1.2 处理客户端操作时产生的异常 5](#_Toc41566421)

[3.2 Assertion and Defensive Programming 7](#_Toc41566422)

[3.2.1 checkRep()检查rep invariants 7](#_Toc41566423)

[3.2.2 Assertion/异常机制来保障pre-/post-condition 9](#_Toc41566424)

[3.2.3 你的代码的防御式策略概述 10](#_Toc41566425)

[3.3 Logging 10](#_Toc41566426)

[3.3.1 异常处理的日志功能 10](#_Toc41566427)

[3.3.2 应用层操作的日志功能 11](#_Toc41566428)

[3.3.3 日志查询功能 11](#_Toc41566429)

[3.4 Testing for Robustness and Correctness 14](#_Toc41566430)

[3.4.1 Testing strategy 14](#_Toc41566431)

[3.4.2 测试用例设计 14](#_Toc41566432)

[3.4.3 测试运行结果与EclEmma覆盖度报告 16](#_Toc41566433)

[3.5 SpotBugs tool 16](#_Toc41566434)

[3.6 Debugging 17](#_Toc41566435)

[3.6.1 EventManager程序 17](#_Toc41566436)

[3.6.2 LowestPrice程序 19](#_Toc41566437)

[3.6.3 FlightClient/Flight/Plane程序 21](#_Toc41566438)

[4 实验进度记录 22](#_Toc41566439)

[5 实验过程中遇到的困难与解决途径 22](#_Toc41566440)

[6 实验过程中收获的经验、教训、感想 23](#_Toc41566441)

[6.1 实验过程中收获的经验和教训 23](#_Toc41566442)

[6.2 针对以下方面的感受 23](#_Toc41566443)

# 实验目标概述

本次实验重点训练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

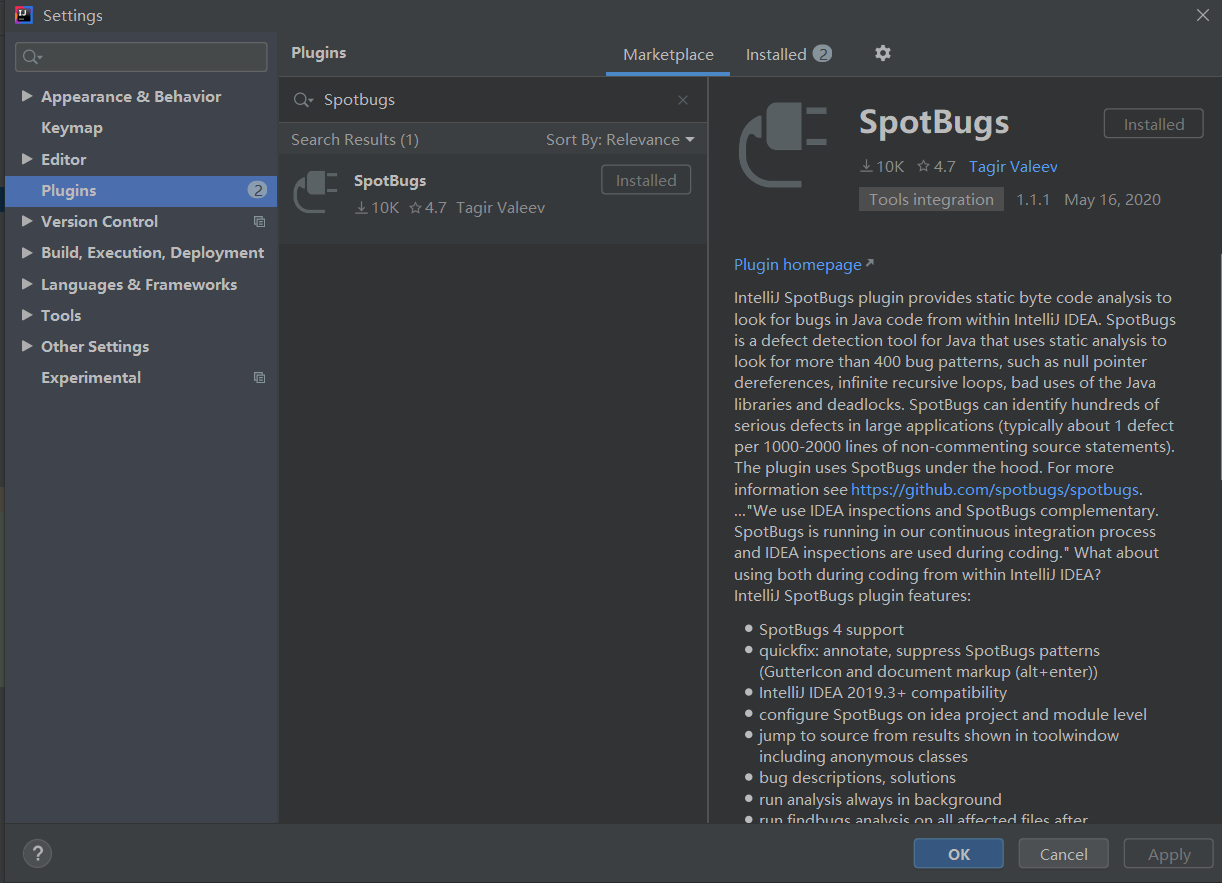
 日志

 调试技术

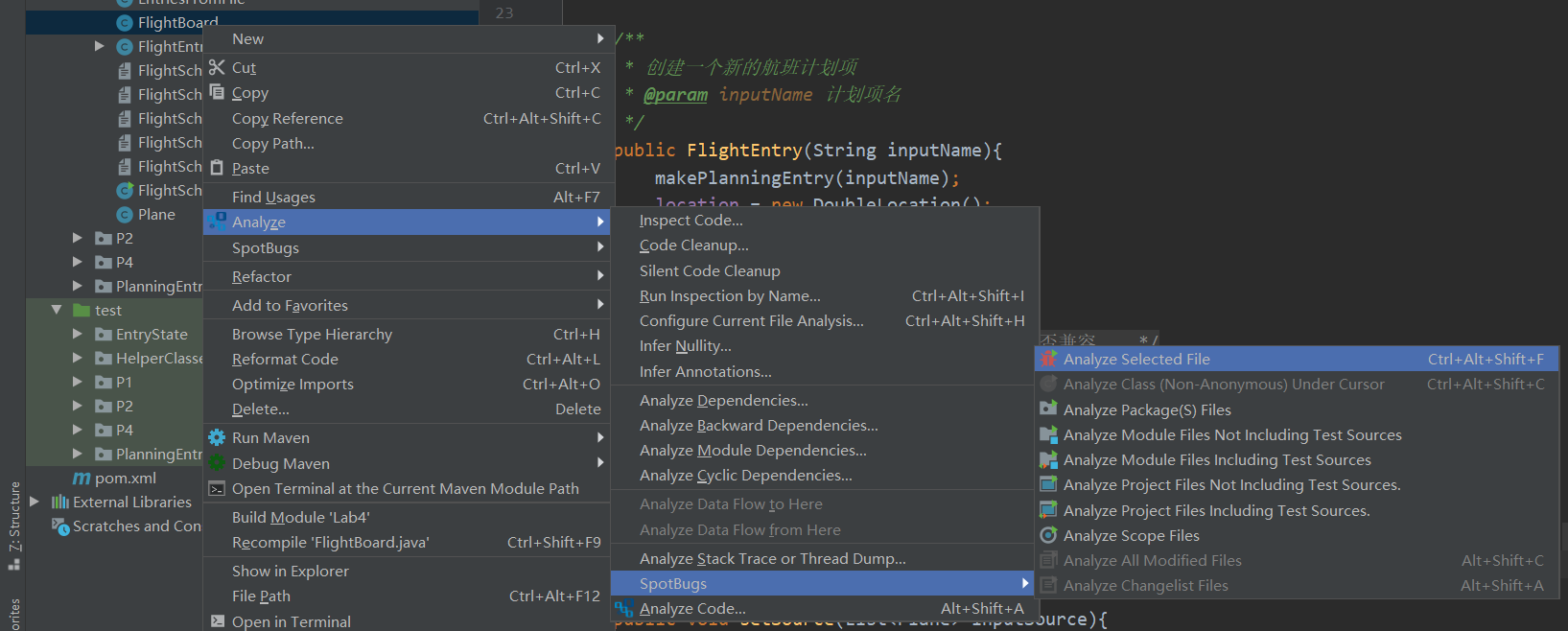
 黑盒测试及代码覆盖度

# 实验环境配置

本次实验需要安装SpotBugs插件，在插件界面搜索并安装即可。



重启IDE后，按照如下步骤即可正常使用。



在这里给出你的GitHub Lab4仓库的URL地址（Lab4-学号）。

https://github.com/ComputerScienceHIT/Lab4-1180301022

# 实验过程

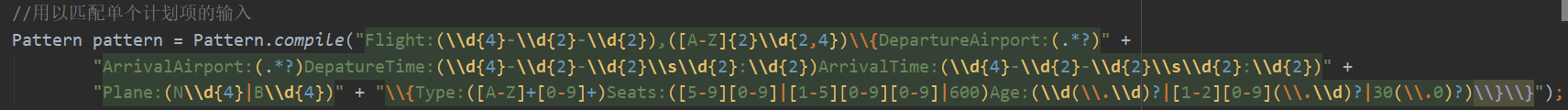
## Error and Exception Handling

### 处理输入文本中的三类错误

文本中的三种错误可以划分并解决如下：

1. 文件格式异常：输入文件中存在不符合语法规则的语句，例如某个元素定义的标签非法、元素定义的内容格式与语法规范不一致（例如最多 1 位小数但使用了 2位小数、要求 2 位大写字母和 2-4 位数字但却使用了非大写字母或超过4 位数字等）、元素定义中分量的数目或次序与规范不符合（例如航班号在日期前面出现，或缺少日期等）等。

解决方法：修改读入文件时的正则表达式，使其满足格式要求。



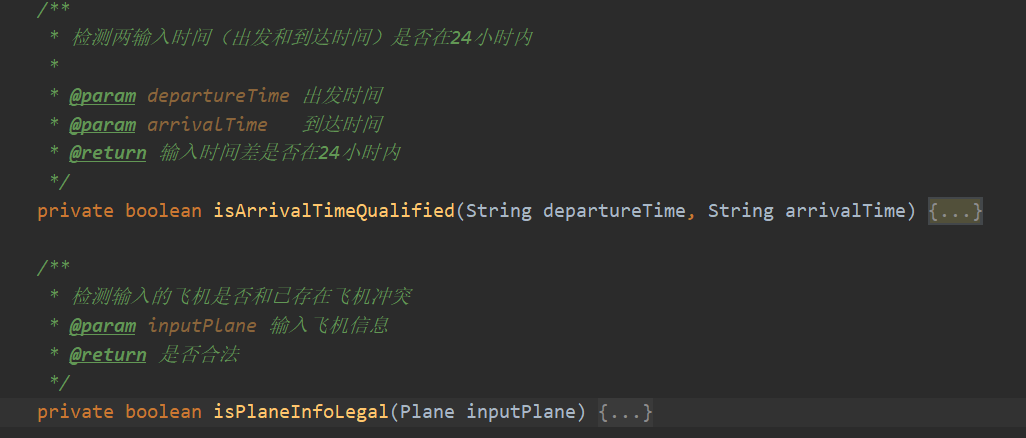
1. 相同标签异常：存在多个航班计划项的“日期,航班号”信息完全一样。

解决方法：创建计划项后，遍历已经加入集合的计划项，检测新计划项是否和每一项兼容。其中，兼容的检测包括对标签相同异常的检测。如果两计划项的名字相同，且出发日期相同，则不可兼容。

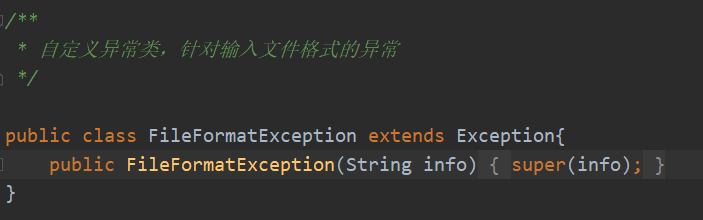
1. 依赖关系异常：第一行出现的航班日期与内部出现的起飞时间中的日期不一致；降落时间中的日期与航班日期差距大于 1 天；同一个航班号，虽然日期不同，但其出发或到达机场、出发或到达时间有差异；在不同航班计划项中出现编号一样的飞机，但飞机的类型、座位数或机龄却不一致。

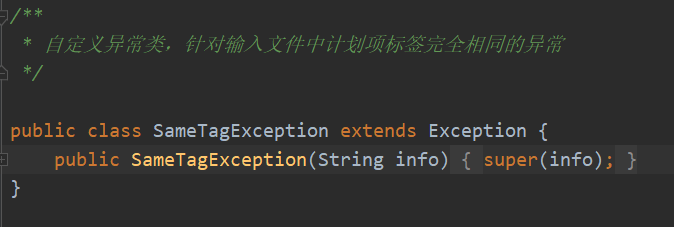
解决方法：同2，对新计划项进行兼容性检测。如果名字相同，出发/到达地点有任一个不同，则不可兼容；名字相同且出发、到达日期均不在同一天，起止时刻不相同，不可兼容。除此之外，设计方法检测每个新计划项是否满足降落时间中的日期与航班日期差距小于 1 天，以及编号相同的飞机是否信息相同。

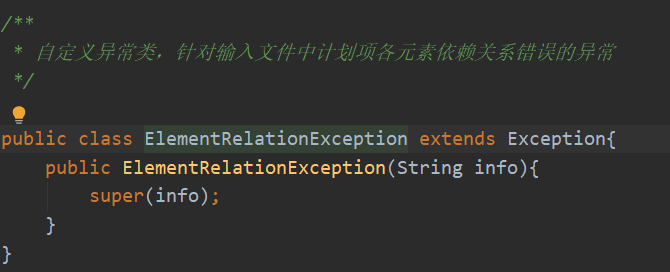




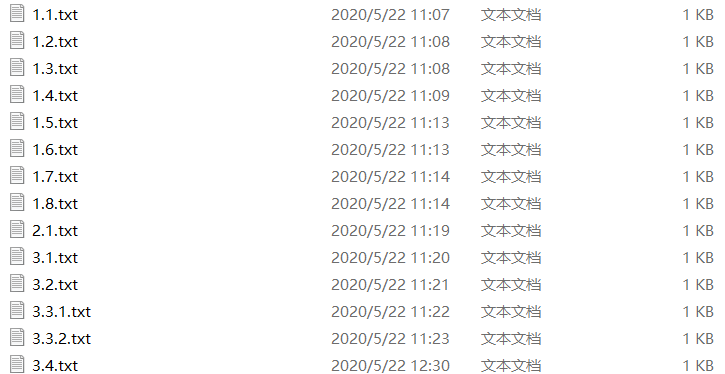
设计异常类型如下：







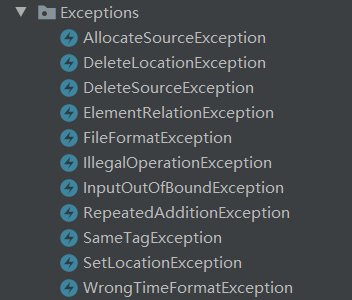
对于每种异常情况，构造了一系列文本，检测程序健壮性。每个文件对应一种异常情况。



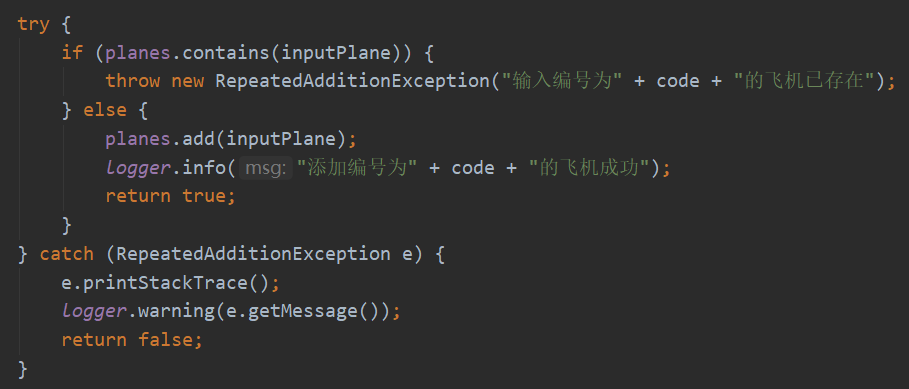
### 处理客户端操作时产生的异常

以FlightScheduleAPP为例说明，后两个问题不再赘述。

设计如下异常类型以供使用：

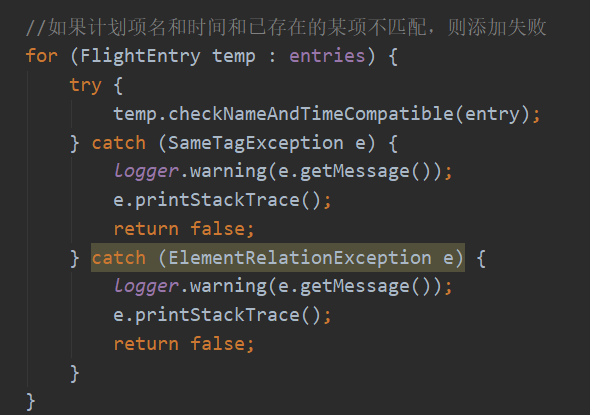


1.重复加入已有飞机时，抛出RepeatedAdditionException，捕获并处理。

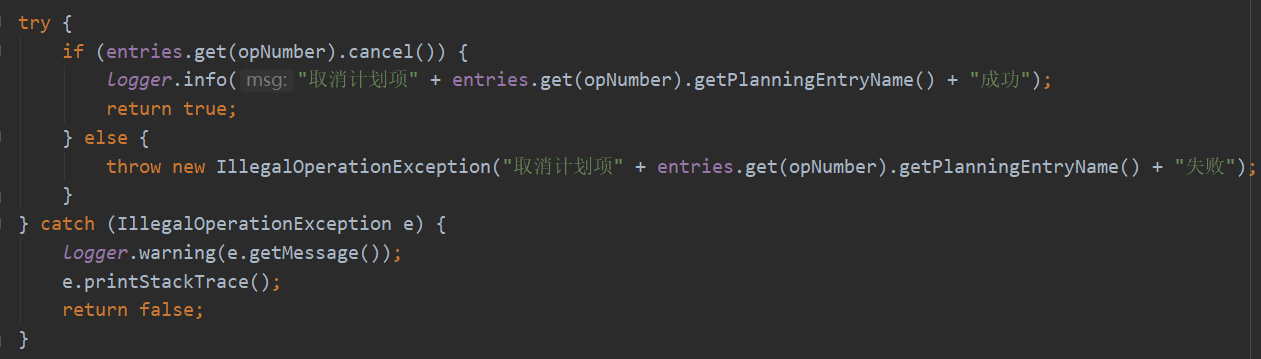


2.重复加入地点时，抛出、捕获、处理同上。

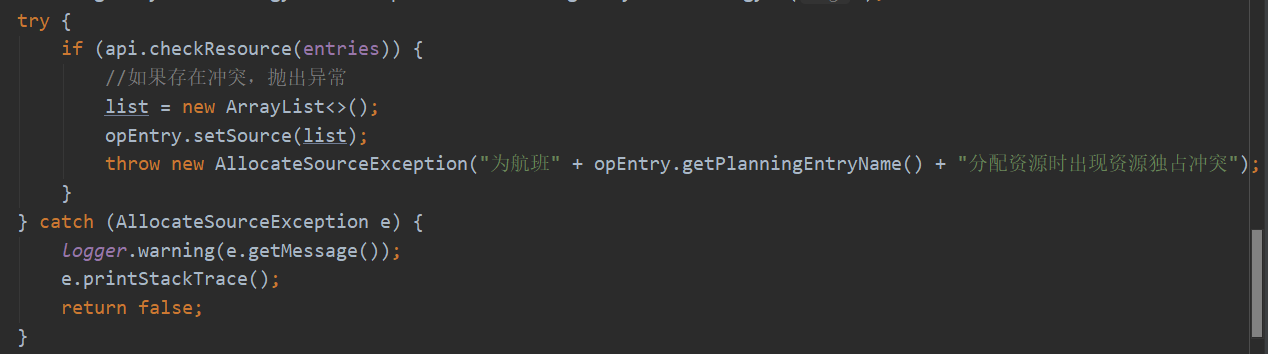
3.加入新计划项时，如果出现不兼容，则抛出对应异常并捕获处理。具体的异常抛出在调用方法中实现。



4.取消某航班时，如果不满足取消条件，抛出异常并捕获处理。



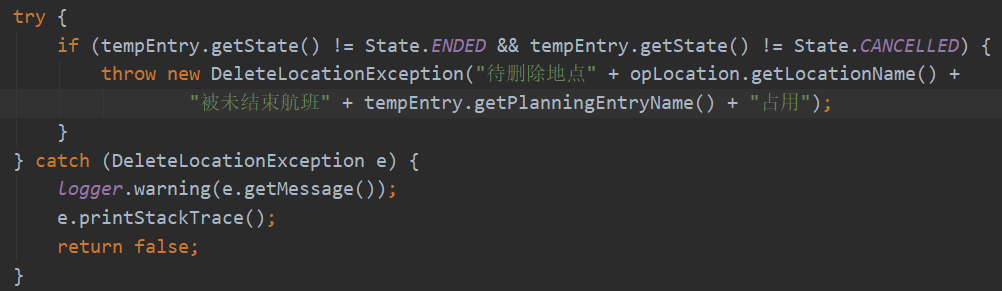
5.为某航班分配飞机，首先检测是否处于待分配状态。具体实现同4。如果处于待分配状态，则选择飞机并尝试进行分配。然后进行资源冲突检测。如果存在使用该飞机且时间有重合的航班，则抛出异常。



6.启动某计划项，同4。

7.结束某计划项，同4。

8.删除指定位置。删除之前首先检测是否存在占用该位置且未被取消/未结束的航班。如果有，则无法删除并抛出异常。



9.删除指定飞机。同8。

## Assertion and Defensive Programming

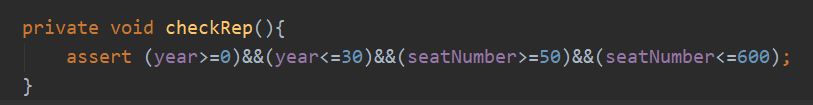
### checkRep()检查rep invariants

1. 对于Plane，要符合从文件中读取计划项时的格式要求。

AF(code, model, seatNumber, year) = 编号code，型号为model，座位数为seatNumber，机龄为double的飞机

RI：(year>=0)&&(year<=30)&&(seatNumber>=50)&&(seatNumber<=600)

Safety from rep exposure：所有成员域都用private final修饰，因为成员域是不可变类所以可以在方法中直接返回

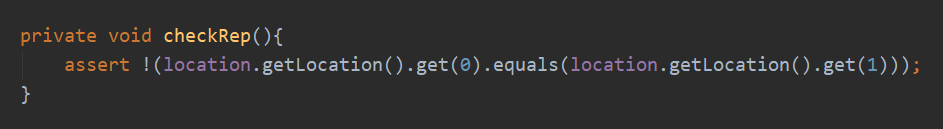


1. 对于FlightEntry，出发地点和到达地点不能相等。且出发时间不能晚于到达时间（在TimePair类中限制，所以FlightEntry类的checkRep不再重复）。

AF:名字为name，位置为location，状态为status，资源为source，时间为time的航班计划项

RI:出发地不能和目的地相等

Safety from rep exposure:成员域用private修饰，返回时采用防御性复制



TimePair类的checkRep，限制了出发和到达时间的关系。

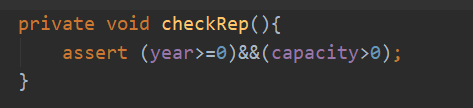


1. 在Train类中，要求类似Plane。

AF(number, type, capacity, year) = 编号number，类型为type，定员数capacity，使用时间为double的车厢

RI：(year>=0)&&(capacity>0)

Safety from rep exposure：所有成员域都用private final修饰，因为成员域是不可变类所以可以在方法中直接返回



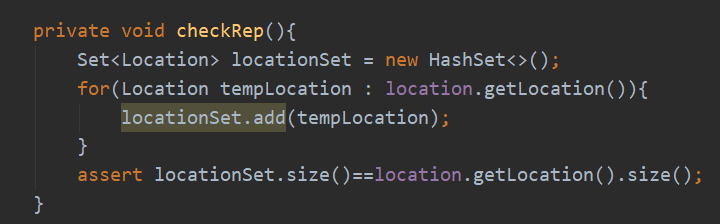
1. TrainEntry

AF:名字为name，位置为location，状态为status，资源为source，时间为time的列车计划项

RI:车次的位置不能重复

Safety from rep exposure:成员域继承自父类且用protected修饰，返回时采用防御性复制

采用的思想是，将location的各个地点加入到一个新的集合中，并检测集合中地点数目和location地点数目是否相等。因为重写了Location类的hashCode和equals方法，所以在同一个集合中不会出现两个同名的Location。



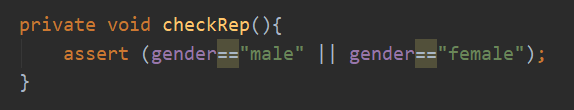
1. Teacher

在此限制性别必须为male或female

AF(name, id, gender, title) = 姓名name，身份证号为id，性别为gender，职称为title的教师

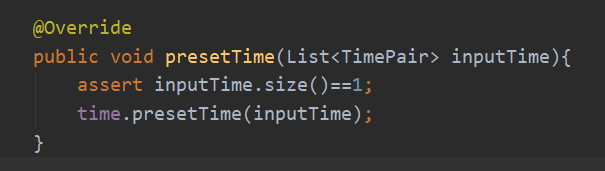
RI：gender=="male" || gender=="female"

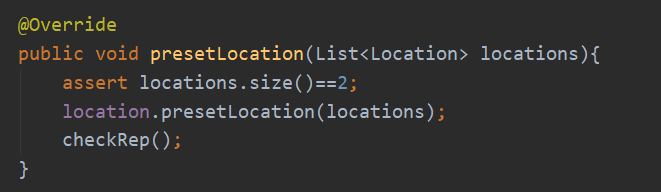
Safety from rep exposure：所有成员域都用private修饰，因为成员域是不可变类所以可以在方法中直接返回



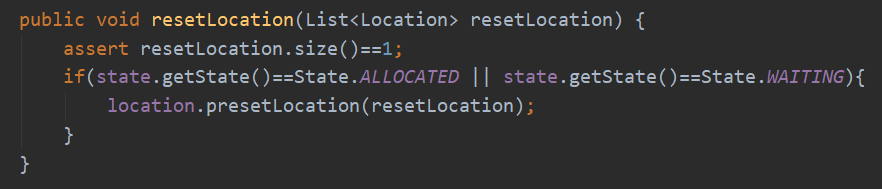
### Assertion/异常机制来保障pre-/post-condition

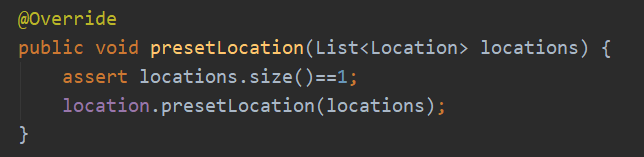
1. 在给FlightEntry分配位置和时间时，限制地点数为2，时间对数为1（即一个起始时间，一个到达时间）。

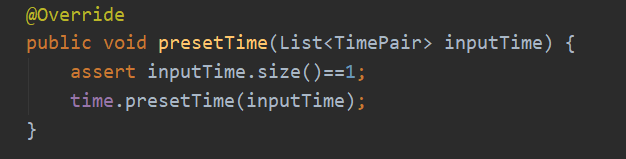




1. 为CourseEntry分配时间和位置，以及重新分配位置时，限制地点数为1，时间对数为1（即一个起始时间，一个到达时间）。







### 你的代码的防御式策略概述

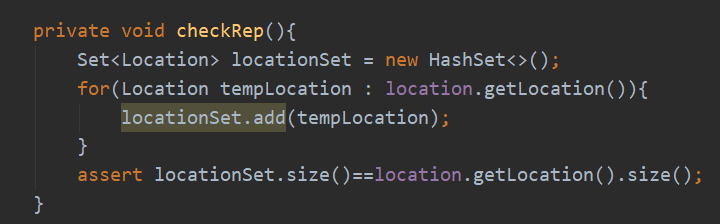
以为列车计划项分配位置为例：

用户在客户端操作时，在APP内会首先通过一系列的try和catch判断输入数据的合法性，如果输入非法，则抛出异常并给出提示，要求用户重新操作（1）。此外，在各个ADT内通过checkRep保持不变性，通过assert保证输入数据的正确性，防止非法数据对程序造成损害（2）。

（1）为列车计划项分配位置时，检测是否输入相同的地点。如果是，则抛出异常。



（2）检测列车计划项是否有重复地点



## Logging

### 异常处理的日志功能

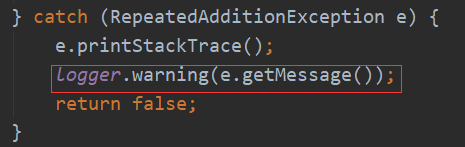
以FlightScheduleAPP的日志为例。

首先进行日志的配置。说明：为了获取yyyy-MM-dd HH:mm的时间格式，我修改了%JAVAHOME%-conf-logging.properties的配置文件，一并上传到仓库中。

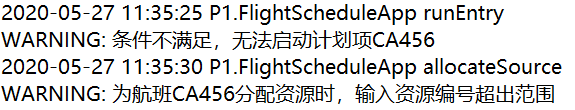


当用户操作出现错误时，将异常信息写入日志。

具体实现方法是，在每个catch块中，添加语句logger.warning(e.getMessage)来记录异常信息。



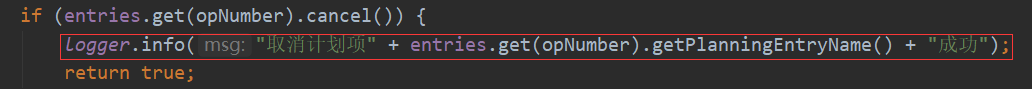
得到文本文件显示如下：



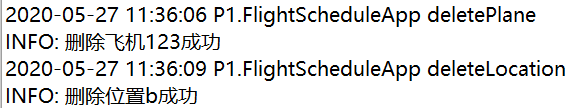
### 应用层操作的日志功能

在每个用户操作返回前添加一条记录语句，记录操作是否成功，以及操作的部分细节。

例如取消计划项时，如果操作成功，就在日志中记录该操作。否则记录操作的异常信息。



得到文本文件显示如下：

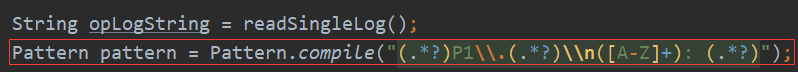


### 日志查询功能

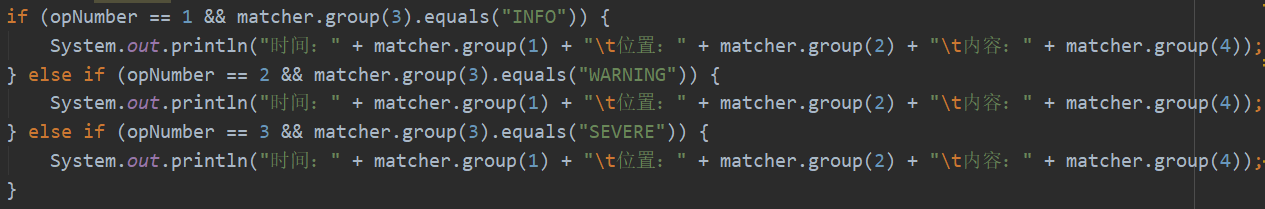
日志可以按照信息类型、操作类型、时间进行筛选。

1. 按照信息类型筛选

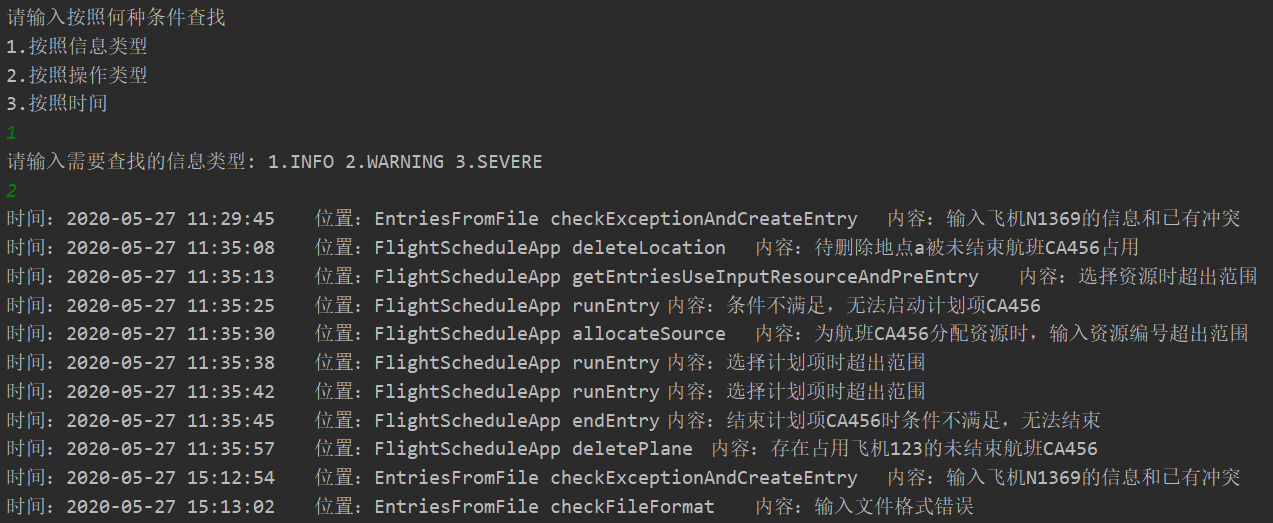
每次读入两行（一条信息），对其进行正则表达式的解析。



根据用户输入消息类型（INFO, WARNING, SEVERE），遍历并打印相应信息。

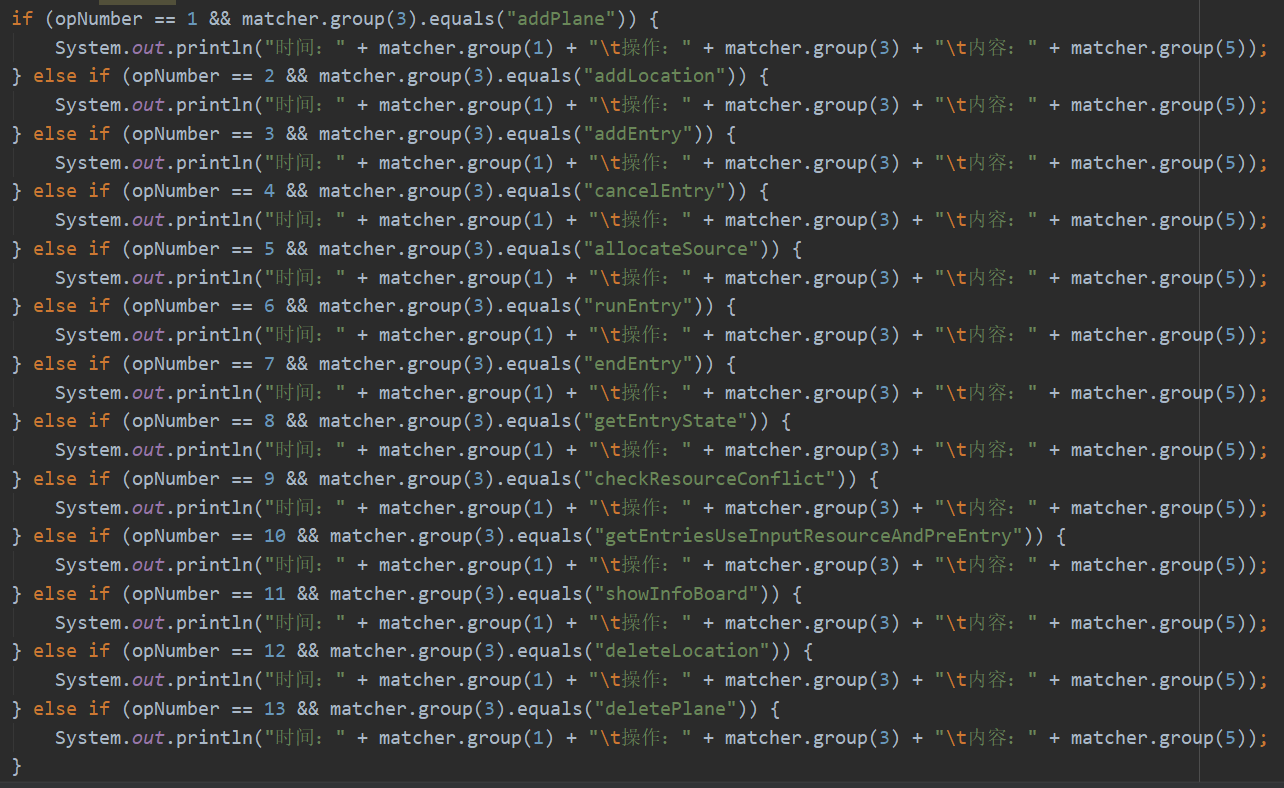


运行结果如下：



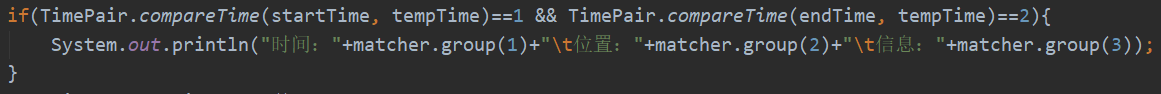
1. 按照操作类型筛选

类似1的情况，同样是读入一条日志的信息，获取其操作类型，并和输入进行比对，输出符合要求的日志信息。

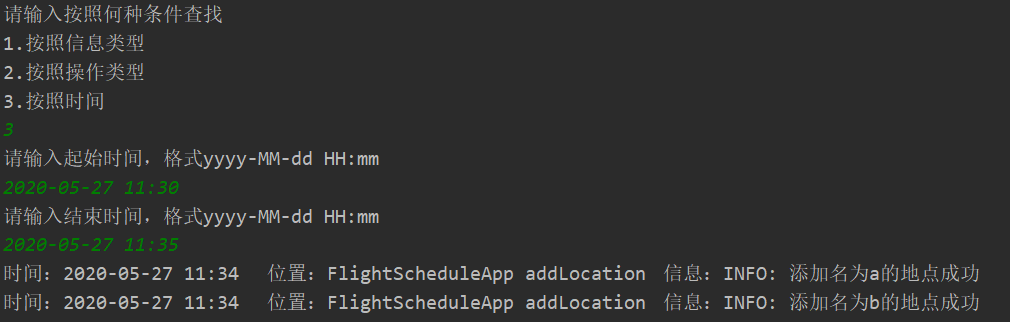


1. 按照输入时间段查找

首先对用户输入时间进行格式检查，并检查起始时间是否早于结束时间。然后对日志进行解析，得到每条日志的记录时间，并比对是否在输入时间段内。如果是则输出，否则考察下一条日志。



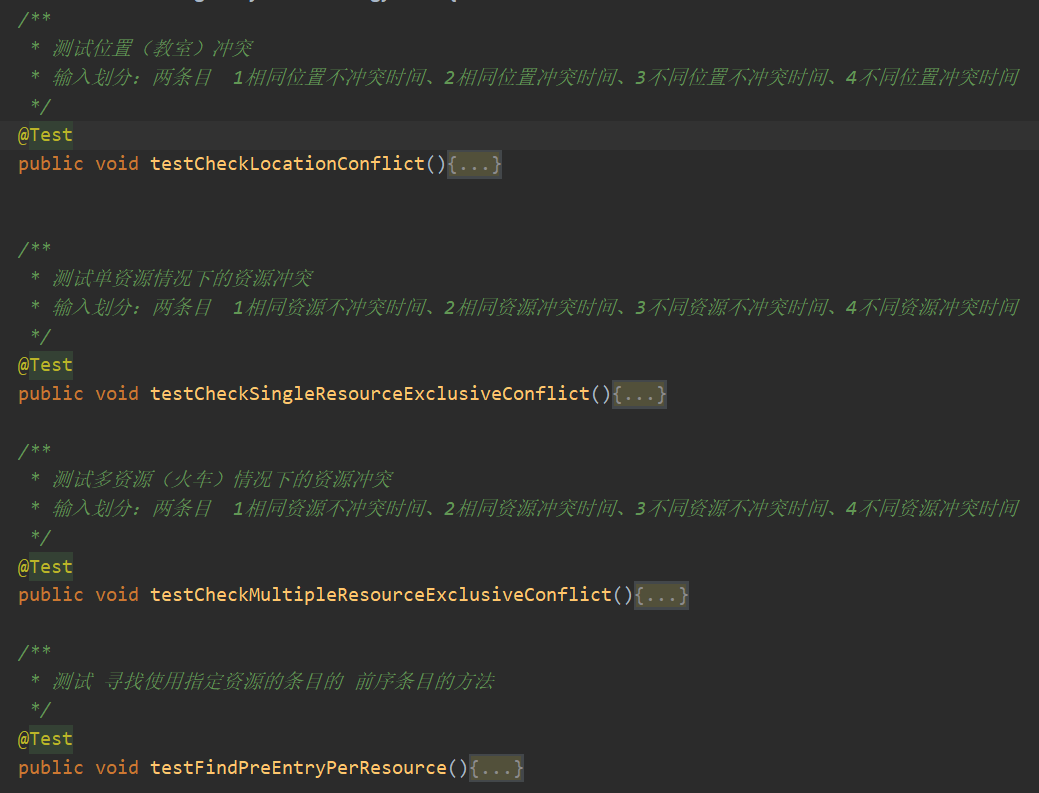
运行结果如下：



## Testing for Robustness and Correctness

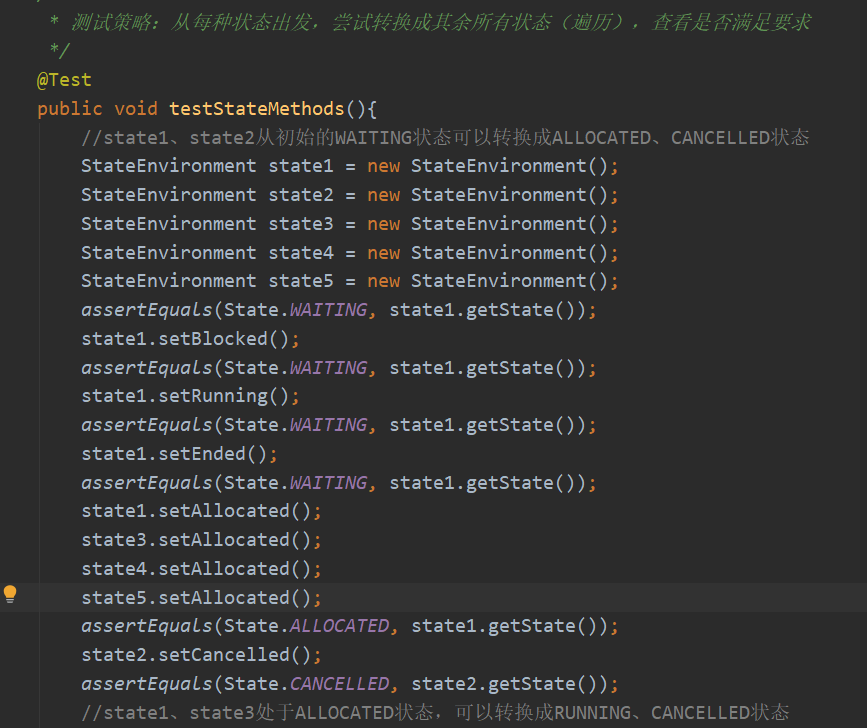
### Testing strategy

使用等价类和边界值的测试思想，为每个ADT设计测试数据。以可复用API测试为例，将输入数据作如下划分：



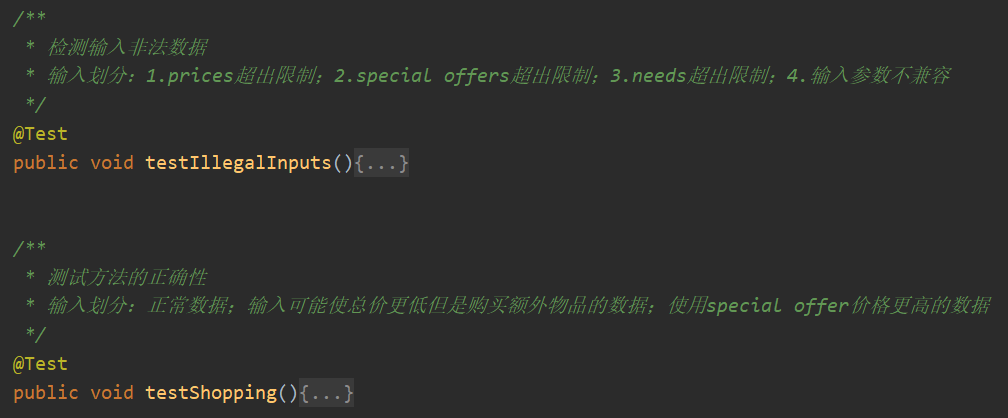
### 测试用例设计

以状态转移的类为例。尝试从每种状态转化成其余所有状态，并检查是否改变成功且是否符合要求。

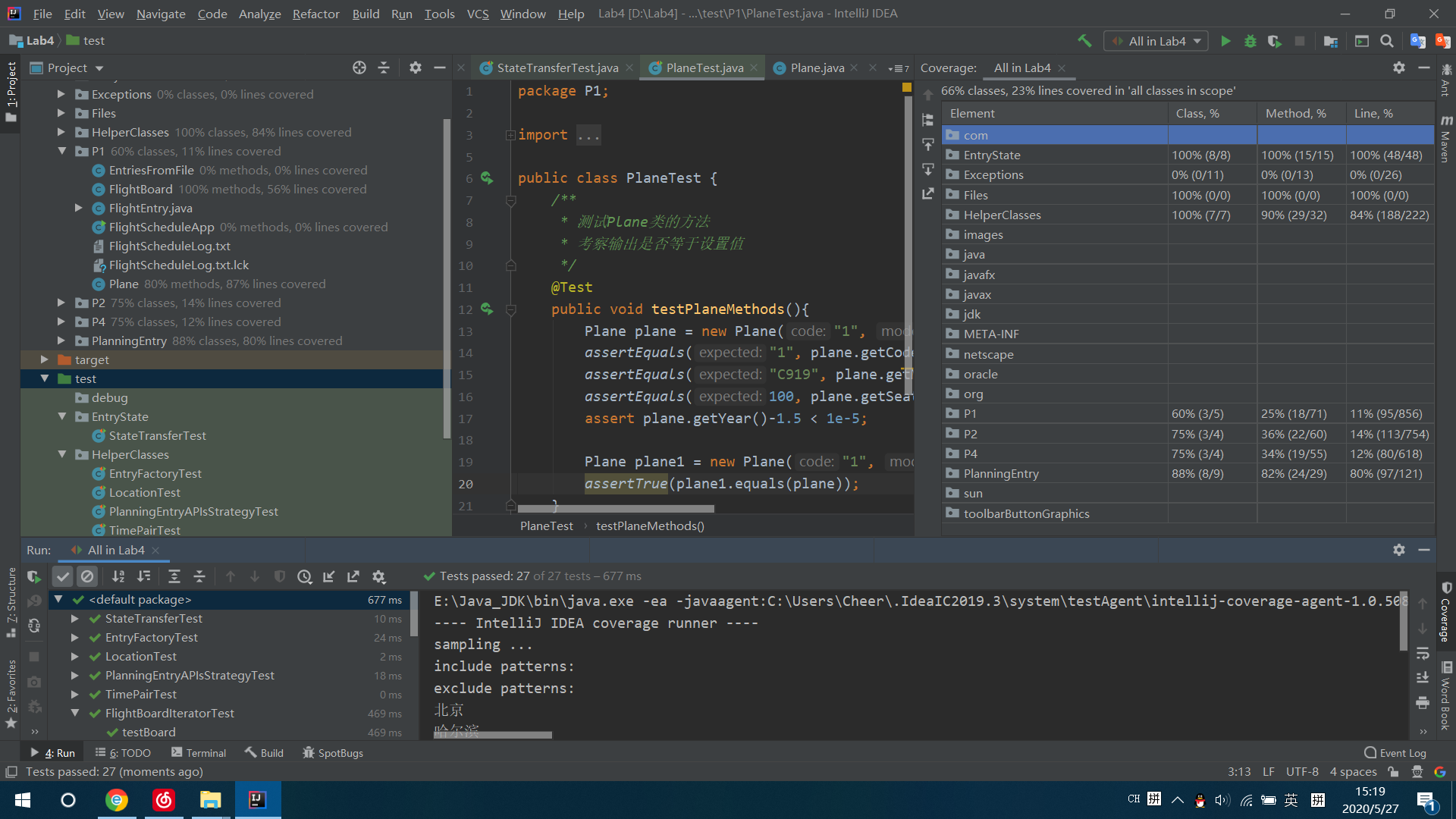


以下代码较长，不再赘述。最终实现了逐行代码的全覆盖。

对于其它类，采用笛卡儿积的方法，对每种情况进行选择，以较少的代码实现较高的覆盖度。



### 测试运行结果与EclEmma覆盖度报告

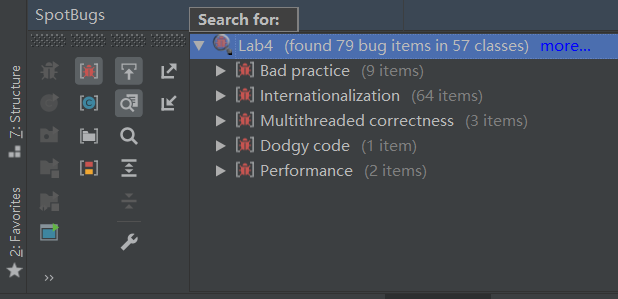


由于APP中代码较多，且都是private方法，无法进行测试，所以覆盖度偏低。

## SpotBugs tool

发现了哪些错误，每种错误代表什么不良的编程习惯

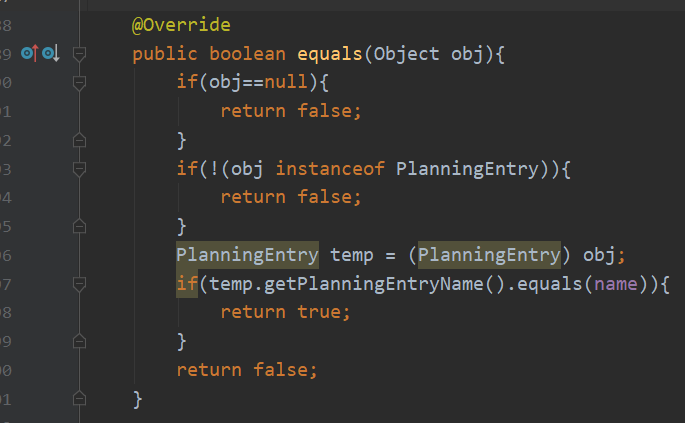
对代码修改，消除这些错误。



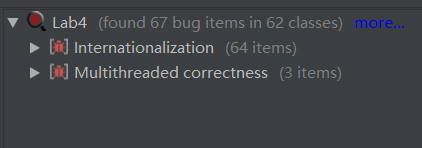
其中第二项和第三项由于调用外部库引起，无法消除。

bad practice中的错误是程序初期开发时处理非法数据调用的System.exit(-1)。

还有一处错误，未判断equals方法的输入是否为空，修改如下。



修改后的结果：



## Debugging

### EventManager程序

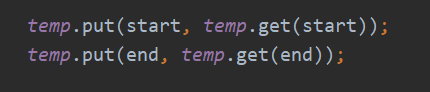
理解待调试程序的代码思想

将每个小时看作一个节点，每输入一个时间段，就将其中包含的节点的值+1（节点值代表当前时刻正在进行的时间数目）。然后遍历temp，寻找temp的最大值并输出即可。

发现并定位错误的过程

首先理解程序的思想，然后根据该思想修改代码中的明显错误的部分。

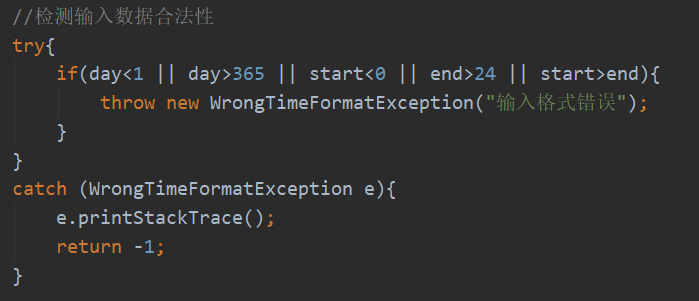
例如下面的原地传送行为：



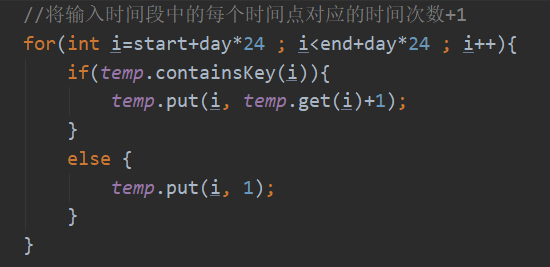
再通过debug一步步跟踪代码执行过程，查看细节部分。例如边界值问题/操作符问题。

你如何修正错误

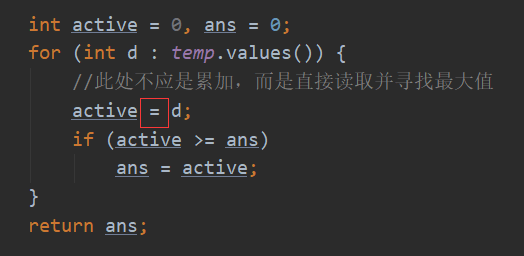
首先检测输入数据是否合法，即day, start, end是否符合要求，如果不符合就终止执行。



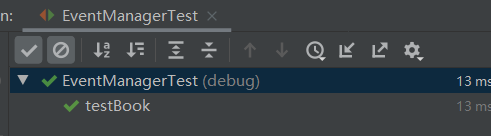
修改前的程序甚至没用到参数day。在此加入参数day的使用。



修改运算符，使用赋值而不是累加。



修复之后的测试结果



### LowestPrice程序

理解待调试程序的代码思想

给出一系列价格、special offer、需求，求购买需求物品的最低价格。首先默认每个物品都是单买，并储存其价格。然后遍历special offer，如果买了某一项offer的所有物品后，物品数都不超过需求，且价格更低，则将需求中的物品数目减去对应值，递归调用该方法。最终求得在所有special offer下的最低价格。

发现并定位错误的过程

显而易见的错误：出现边界值的问题。

C:\Users\Cheer\AppData\Roaming\Tencent\Users\1430268665\QQ\WinTemp\RichOle\BU{1~%A_6QY}5(M08UWB831.png

通过debug一步步跟踪，发现在某项special offer刚好满足物品数需求时，不会购买该offer。错误如下：

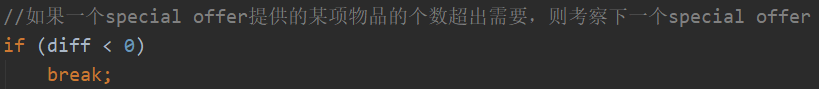
C:\Users\Cheer\AppData\Roaming\Tencent\Users\1430268665\QQ\WinTemp\RichOle\XGTNGLH6FSKS$GB1LWM(@TB.png

你如何修正错误

首先为程序添加检测非法输入的步骤，如果输入非法则不会执行程序。

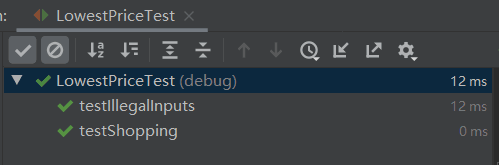


对上述错误进行简单修改：





修复之后的测试结果



### FlightClient/Flight/Plane程序

理解待调试程序的代码思想

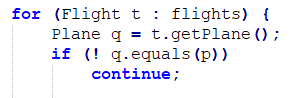
遍历计划项，并尝试给每个航班分配资源。具体方法是，利用随机数取出一个飞机，如果该飞机未被其它所有航班使用，则直接分配。遍历所有其它航班，如果被某个其它航班使用，检查两航班的时间是否有交集。如果有，则重新取飞机并重复上述操作；否则，继续遍历，直到遍历所有航班并无冲突，则可以分配。

发现并定位错误的过程

在使用随机数取飞机的循环中，可以明显看到如果所有飞机都无法分配给该计划项时，循环无法退出。因此要有退出循环的策略。

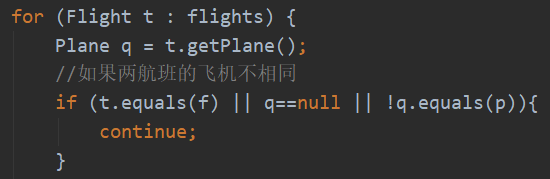
返回结果在循环中未发生改变，恒为true。

另外，根据调试结果，此处t有可能未分配资源，产生空指针异常。

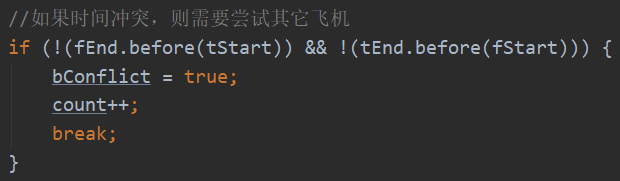


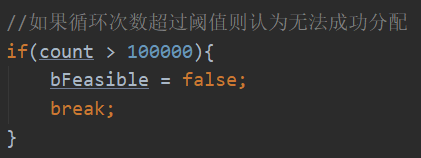
你如何修正错误

增加if语句的判断条件，使其具备判断空指针的功能。

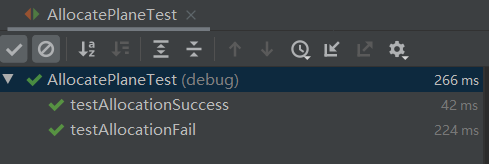


在为每个航班分配飞机时，添加一个count变量。每有一次不成功分配，count值+1。当超过设定值（此处设定为100000）且未分配成功时，说明所有飞机都无法分配给该计划项，返回false。





修复之后的测试结果



# 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 |
| 05-21 | 19：00-23：00 | 文件读入时的异常设计和捕获处理 | 按时完成 |
| 05-23 | 08：30-12：30 | 客户端操作异常捕获处理 | 提前一小时 |
| 05-24 | 14：00-18：00 | 程序的防御改进 | 按时完成 |
| 05-27 | 08：30-11：30 | 日志及其查找功能 | 超时半小时 |
| 05-27 | 14：00-17：30 | 测试策略、spot bugs | 按时完成 |
| 05-27 | 19：30-22：30 | debug环节 | 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 |
| Java logging的默认格式对于正则表达式的处理极其不友好。想改成yyyy-MM-dd HH:mm的格式。 | 查阅博客，修改了配置文件，使写入格式变得易于处理。 |
| 修改上述问题之后，直接把原来的日志删除，而不是删掉整个文件。导致文件开头有一个看不见的\ufeff，无法读取数据。 | 调试跟踪，发现第一行的第一个字符是空，删除日志文件并重新创建，问题解决。 |
| IDEA无法安装EclEmma。尝试用Eclipse打开项目后，程序甚至无法运行。用IDEA也无法运行。 | 幸好我留了个备份。在IDEA中使用run with coverage即有全面的代码覆盖率。 |

# 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

备份很重要！在进行较大规模的修改/换IDE之前一定要留一个备份，防止意外发生。

在开发时，不仅是正确性，还要对程序健壮性多作考虑。防止用户的恶意输入使程序崩溃。使用异常处理机制，让程序捕获异常并优雅地处理，提升使用体验。

不要在某处发生致命错误时直接System.exit()，否则在调试的时候会摸不着头脑。白白浪费很多时间。

坚持测试优先，在开发过程中多进行测试，防止最后需要一口气对于整个程序进行debug。

## 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？

正确性是开发程序的基本条件，一定要保证程序正确运行。否则该程序将失去意义。健壮性是为了抵挡外部的错误输入，保证程序能长出运行。程序员在开发过程中，要先保证正确性，再保证健壮性。

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？）

不划算。如果某错误或异常发生的概率很小，就不应将其作为重点来大写特写。将有关健壮性的大部分代码放在经常出现的异常部分比较合理。

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？

前者强调程序员开发时，对于程序健壮性的保证。后者是对客户增加更多的要求，如果输入错误数据将导致程序非正常运行/宕机。面对普通用户时采用前者思想，考虑到用户群体的区别。面对特定用户，即用户有责任保证输入数据的正确性时，采用后者思想。

1. 过分谨慎的“防御”（excessively defensive）真的有必要吗？你如何看待过分防御所带来的性能损耗？如何在二者之间取得平衡？

没有必要进行过分谨慎的防御。防御固然重要，但不可“喧宾夺主”，占用过分多的代码行。过分防御带来的性能损耗对于用户是非常痛苦的（例如某些游戏使用的D加密，导致游戏经常卡死）。要进行适当且必要的防御，而不是完完全全“滴水不漏”的防御。

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？

在异常处理部分（catch块）中记录异常信息，可以帮助定位错误位置和信息。对于程序员自己使用的方法（如private方法），使用assert判断输入数据，可以有效保证程序正确性。

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？

充分的测试指的是利用尽量精简的代码测试最全面的情况。代码覆盖度只是衡量测试的一个条件，代码覆盖度100%并不意味着100%充分的测试。

1. Debug一个错误的程序，有乐趣吗？体验一下无注释、无文档的程序修改。

有乐趣。debug可以发现藏在细微处的错误，用debug改出一个细小的错误的感觉非常爽。曾经我写程序没有写注释的习惯，那种感觉非常痛苦。现在我在关键部分增加注释说明代码块的目的，对于阅读非常有帮助。

1. 关于本实验的工作量、难度、deadline。

工作量合理。难度适中。ddl非常友好。

1. 到目前为止你对《软件构造》课程的评价和建议。

这门课是我这学期最喜欢的课，每节都会看回放并写博客记录。通过这门课程真的学到了很多很多东西。较为“硬核”的实验让我在实际操作中应用课堂所学知识，这种感觉非常棒。从一开始的一窍不通到现在的刚刚入门，我切实感觉到了这门课给我带来的收获。特别喜欢这门课！

1. 期末考试临近，你对占成绩60%的闭卷考试有什么预期？

这门课我从头到尾都认真学习并思考，整理了很多笔记，我对自己很有信心。