第十一次博客作业

1. 规格化设计的发展历史进程

首先，在查阅资料的过程中，我发现规格化设计这个名词在百度上东西少的可怜，基本没有资料，而查jsf出来的东西更是驴头不对马嘴



因此，我认为这个历史指代的英文原词语应该不是JSF，而是别的某个专有概念。一经查找，我发现了JML，全称*Java Modeling Language*，它是一种遵循契约样式的规格化语言，包含了Hoare style的前后置条件，不变式等。

*The Java Modeling Language (JML) is a specification language for Java programs, using Hoare style pre- and postconditions and invariants, that follows the design by contract paradigm. Specifications are written as Java annotation comments to the source files, which hence can be compiled with any Java compiler.*

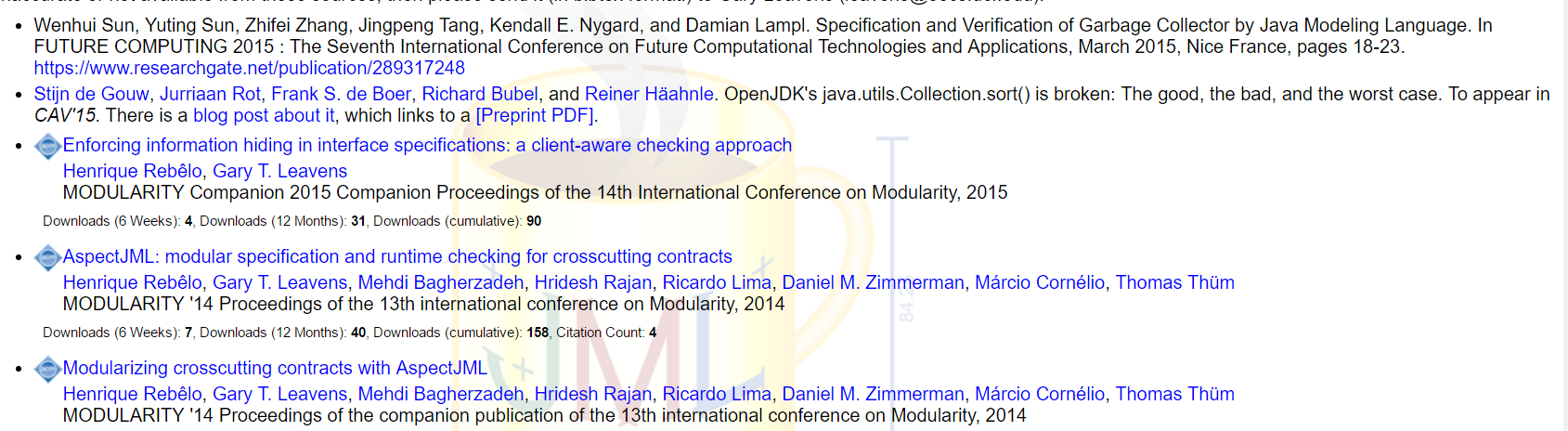
*Various verification tools, such as a runtime assertion checker and the Extended Static Checker (ESC/Java) aid development. [1]*

进一步地，分析JML的历史渊源。最早关于JML的文章已经不可考，但是早在2002年就已经出现了自动校验JML规范性和根据JML检查代码的自动化程序。根据一篇综述性文章Design by Contract with JML [2] ，JML起源于DBC设计思想（最早描述并将DBC规范化变成可实现状态的是Hoare在1969年的一篇关于形式化验证的文章[3]），也就是**D**esign **B**y **C**ontract，这种契约式思想可以说规范了类和调用者的权利和义务[5]，一则明确了实现者的管辖范围，二则引导了调用者的正确使用。JML继承了遵循DBC设计的多种规范化语言的设计思想精髓——比如面向模型规范化语言的丰富表意性和，包括Eiffel[6], Larch, the Refinement Calculus等，在以上语言的基础上，JML构建了一套完整的语义学模型来对Java模型的行为进行描述，以避免代码行为与设计者的意图出现含混不清的情况。之后，人们不断开发出了自动检查JML格式和根据JML自动在代码中插入assertion检查，形式化证明检查，不变式和状态检查，运行时检查等[4]的相关工具包括jmlc，jmlunit，jmldoc等等。

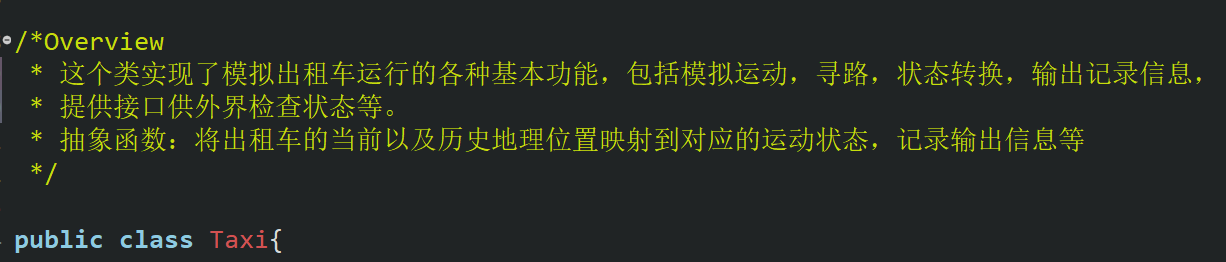
至于人们为什么重视JML，从根本原因上来说，相对于java语言，JML是一种更加高级的语言。它屏蔽了部分细节，进行了进一步抽象，使得设计中调用者需要关心的和可以期望得到的部分得以精确地描述和定义，这一点，不仅满足了对调用者权利和义务的范围界定，也方便了设计者实现。同时，JML引入量词等形式化语言拓展了JAV表达式，规范化的语言格式在满足了表意丰富的情况下，提供了多种工具检查的支持，这也是以往DBC形式化语言不能做到的。

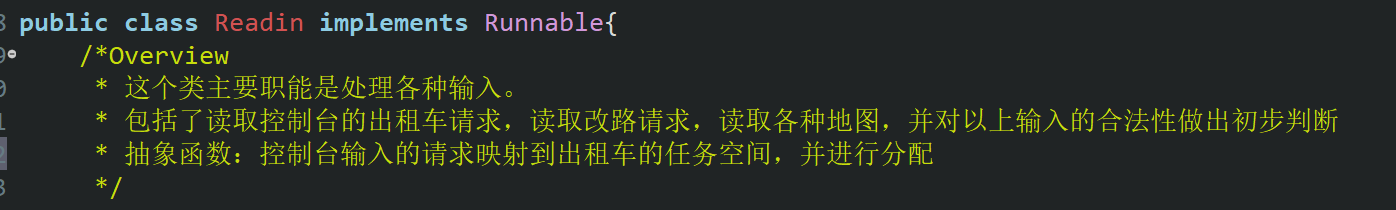
举个例子来说，如果一个开平方的函数，计算出来小数位保留到七位，那么我们直接阅读代码可能会让调用者迷惑，为什么是7位，以后可以改变吗？我只需要4位可以吗？而这些问题如果放大到更加复杂的例子中，会很影响实际工程中上下游的调用关系设计，因此，在使用JML以后，我们可以定义这个函数就是计算开平方的，位数最高（而不是只能）提供7位精确度（给以后代码重写留下解释余地）

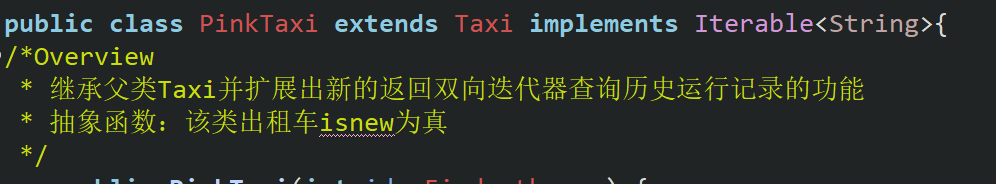
因此，JML也越来越受到人们的重视。下面是一张JML相关论文近年来的研究情况，可以看到热度不减当年。



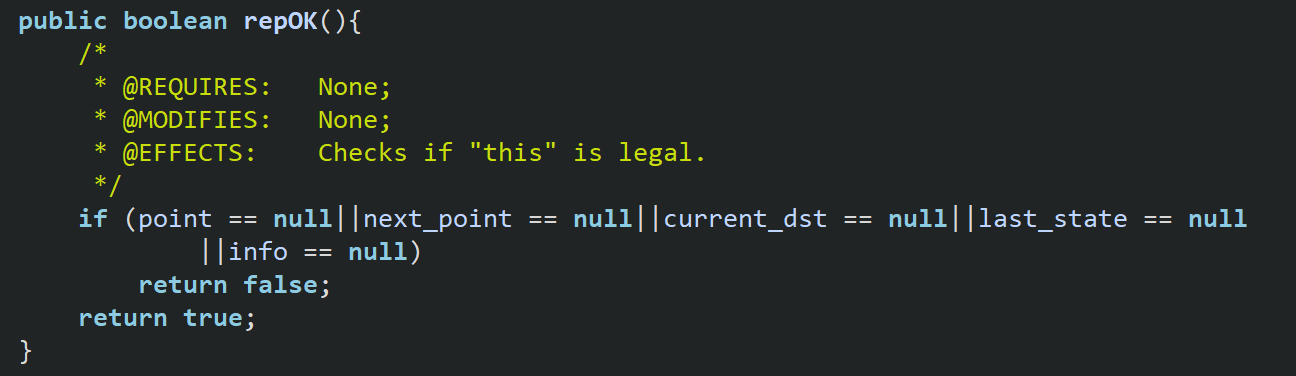
1. 仅仅只是 ***自认为*** 不错的类规格\*3和数据规格\*3
   1. 类规格

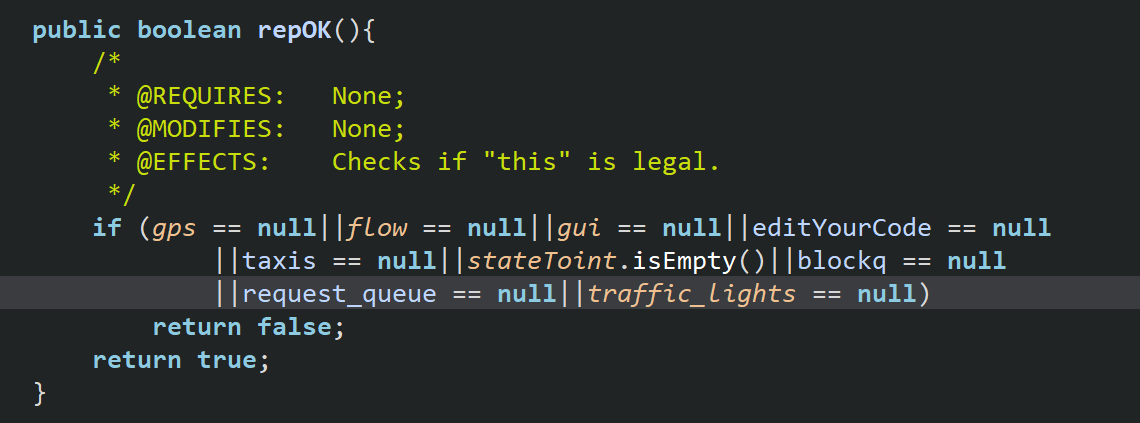


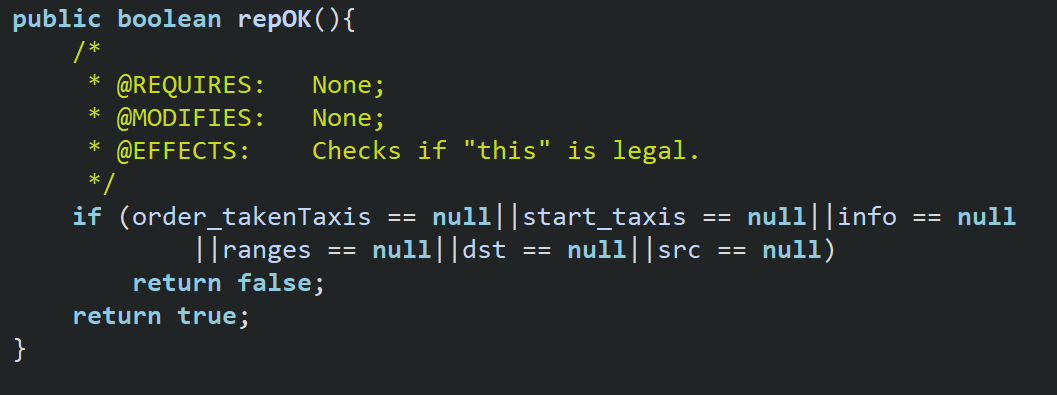




* 1. 数据规格

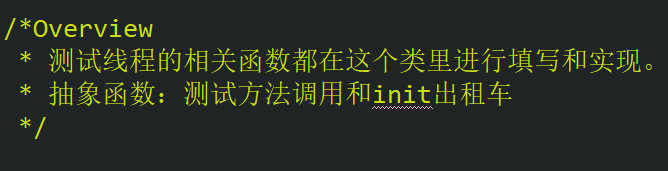


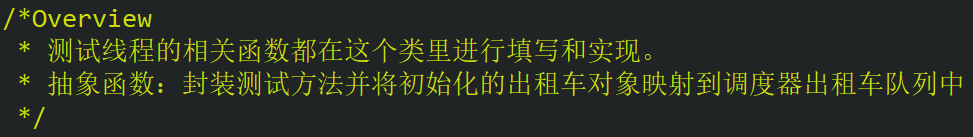




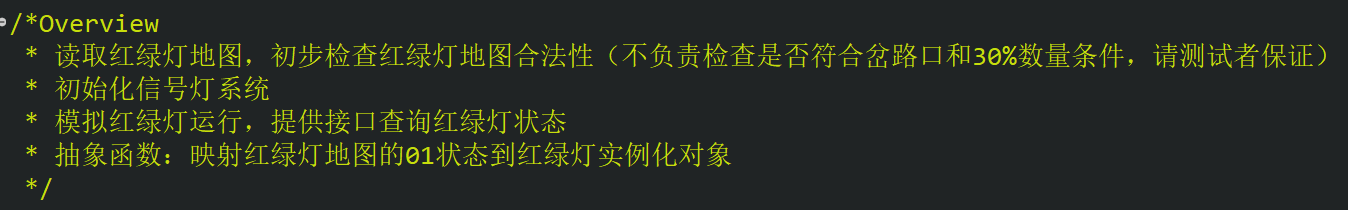
1. 写的***确实不咋地***的过程规格\*5和数据规格\*5及他们的改进
   1. 类规格

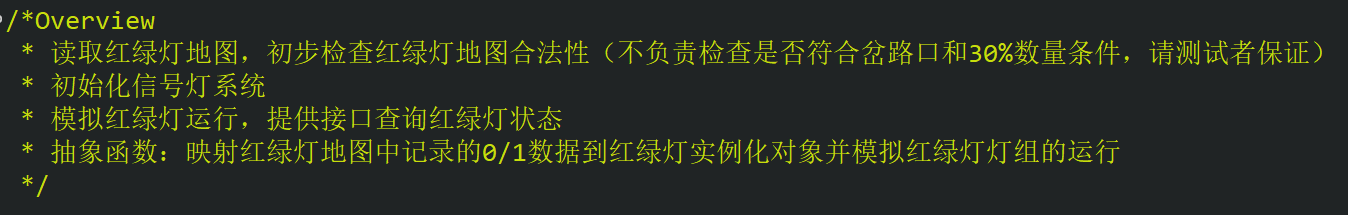
来自于EditYourCode



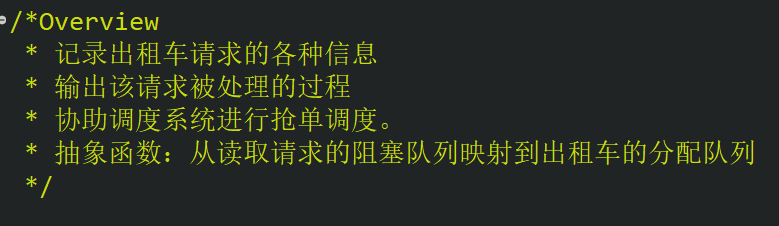


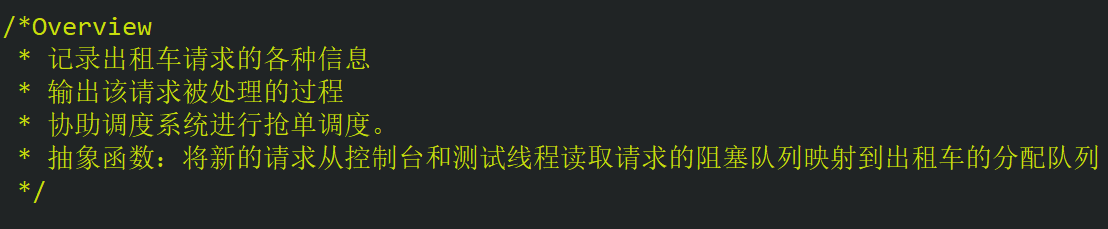
来自于Light



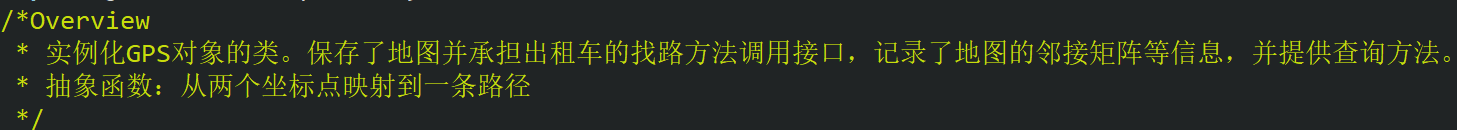


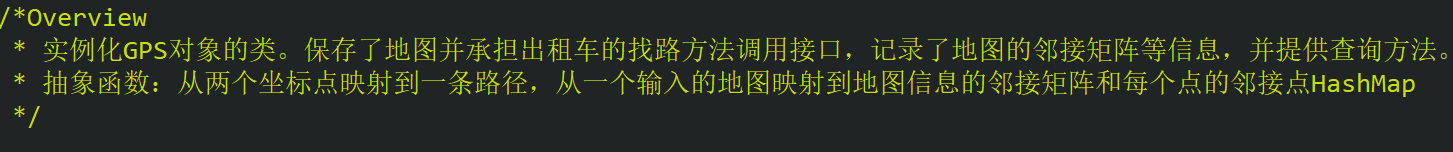
来自于Request





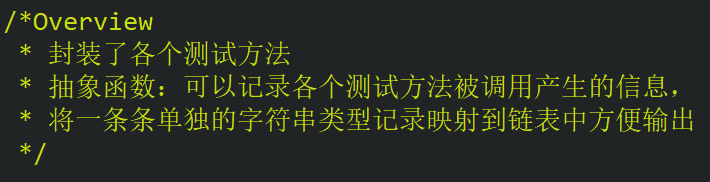
来自于FindPath





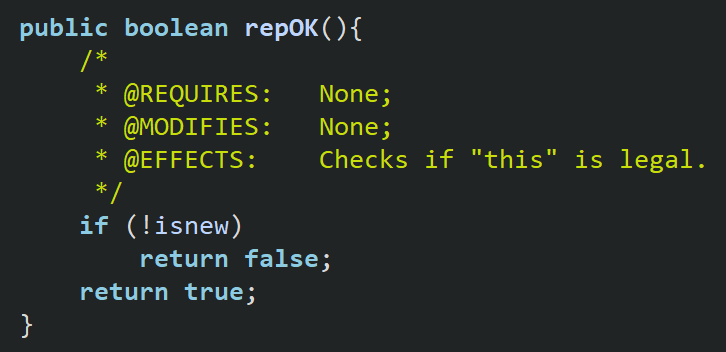
来自于TestOperation

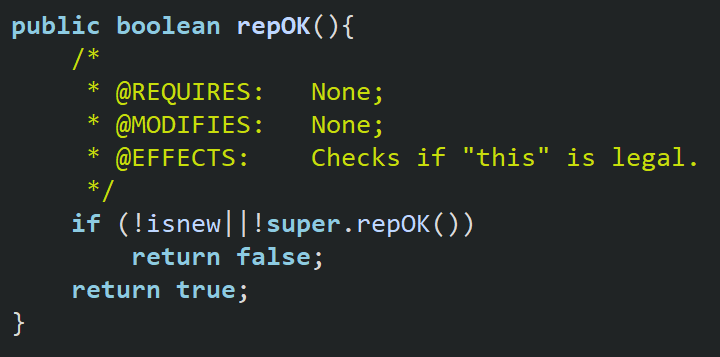




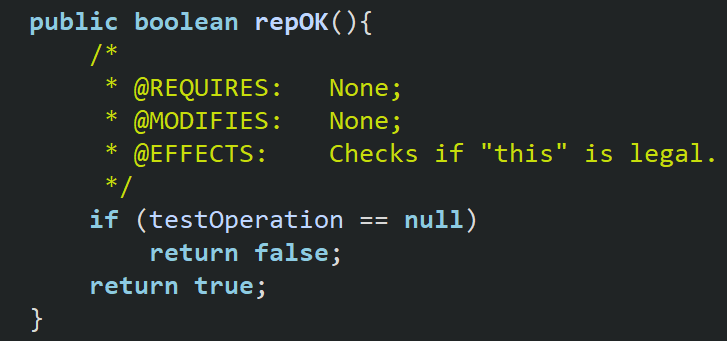
* 1. 数据规格

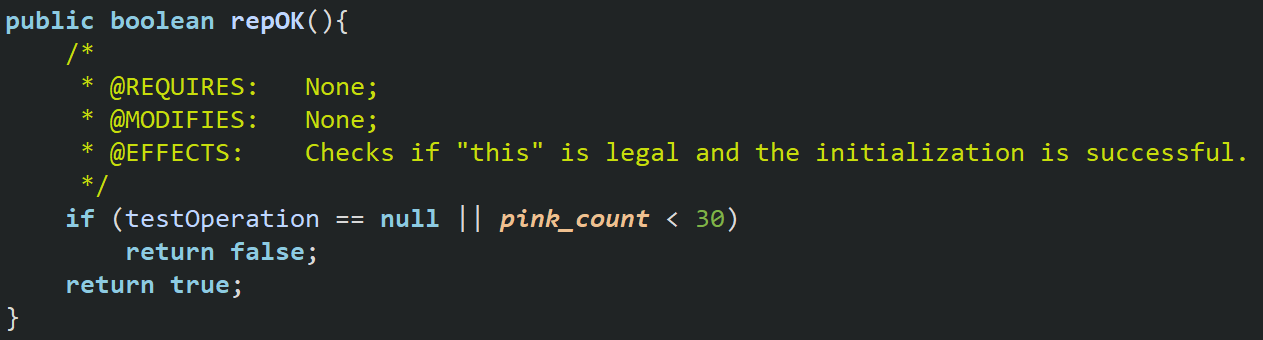
来自于PinkTaxi类



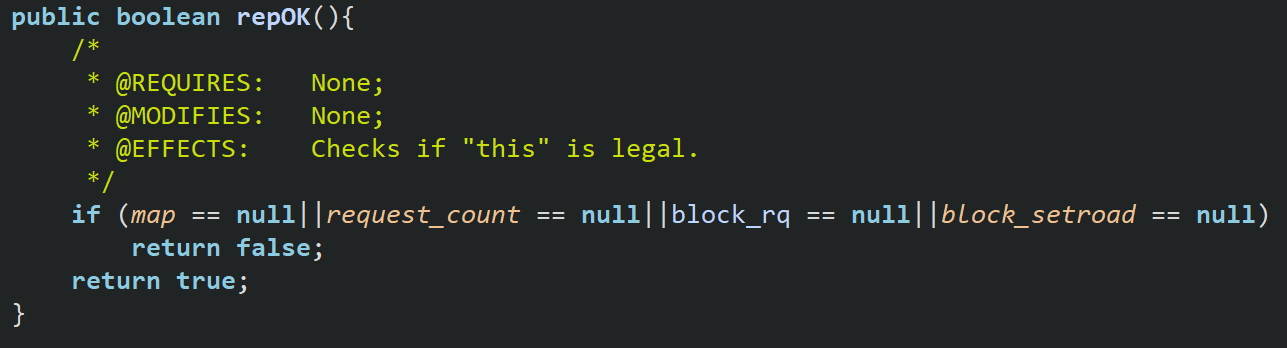


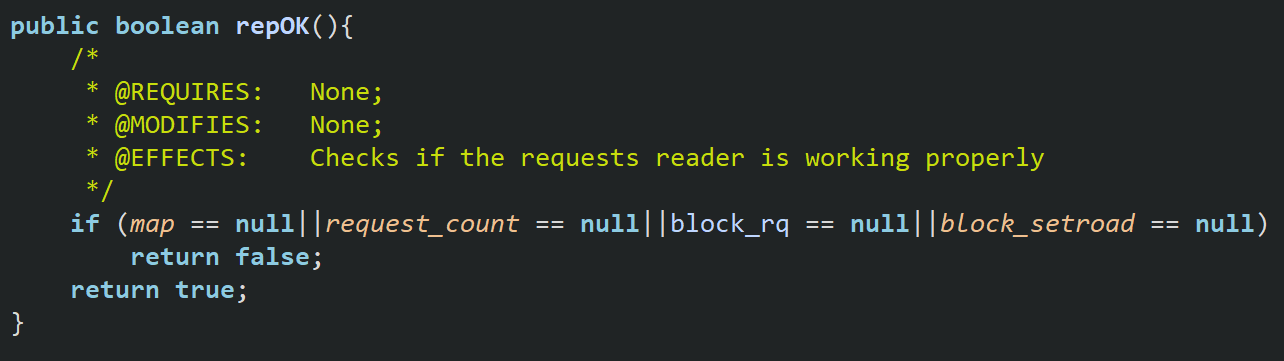
来自于EditYourCode



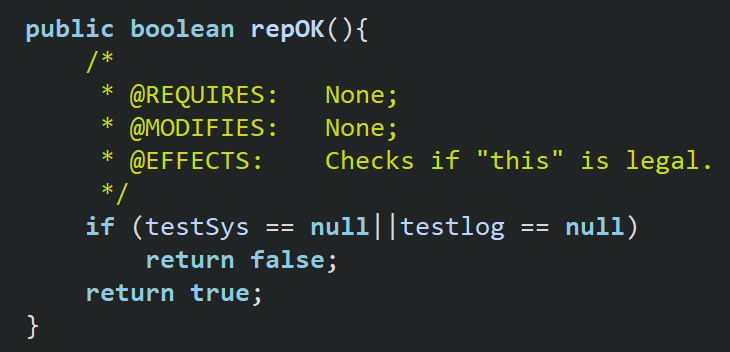


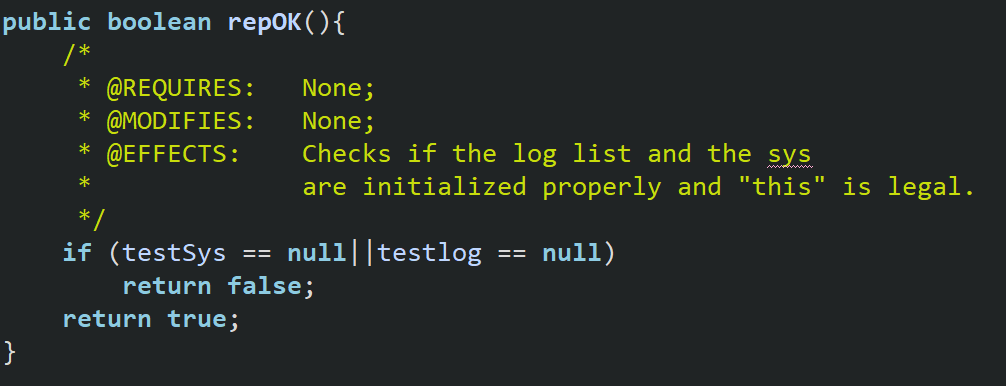
来自于Readin



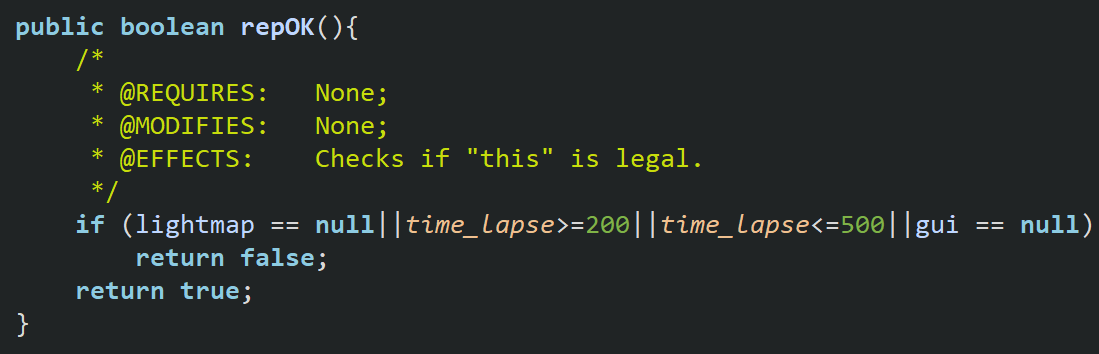


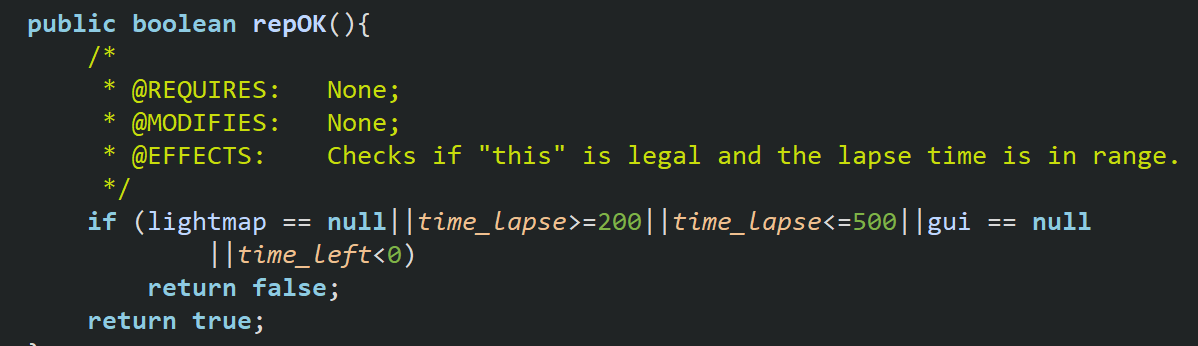
来自于TestOperation类





来自于Light





以上数据规格和类规格写的不够完善，经分析，主要原因我认为有以下几点：

1. 先写了代码后补写规格
2. 表示对象实现概念不够清晰，导致抽象函数写得不好，不知道管理的东西是什么
3. bug中反映出的过程规格质量问题

（三次作业一共两个bug，另外一个纯粹个人笔误）

* 1. 由于出租车随机走动算法问题，一段时间后车辆聚集到地图上方

这个bug最主要的原因是前期设计细节没有考虑周全，在循环当前坐标点的邻接点时直接找了其中第一个流量最小的方向，一般就是上。然而在规格书写中由于先写代码后写规格导致完全忽略了这个设计逻辑上的错误。之后将所有流量最小且相同的点都加入队列随机返回，遂解决。

**附录：引用文章来源**

[1] <https://en.wikipedia.org/wiki/Java_Modeling_Language>

[2] Leavens G T, Cheon Y. Design by contract with JML[J]. URL, 2004, 32(8):571 - 586.

[3] C. A. R. Hoare. An axiomatic basis for computer programming. Communications of the ACM, 12(10):576–583, October 1969.

[4] Gary T. Leavens, Yoonsik Cheon, Curtis Clifton, Clyde Ruby, and David R. Cok. How the design of JML accommodates both runtime assertion checking and formal verification. Science of Computer Programming, 55(1-3):185– 208, March 2005.

[5] Bertrand Meyer. Applying “design by contract”. Computer, 25(10):40–51, October 1992.

[6] Bertrand Meyer. Eiffel: The Language. Object Oriented Series. Prentice Hall, New York, NY, 1992.