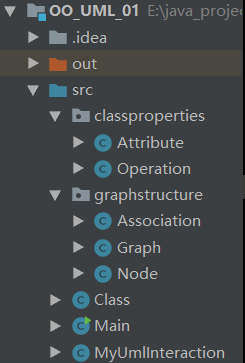
面向对象第四单元及博客总结作业

面向对象第四单元的学习也已经结束了。在本单元我们主要学习了UML对类图、状态图、顺序图的建模方法，并学会了用starUML工具进行UML建模，在代码作业上我们也完成了一整个UML分析工具，能够对一些简单的查询指令给出应答。总体上来讲这次作业仅仅分为两次并且每次都留了比较充足的时间，我们的发挥空间还是比较大的，各位同学们做的方法和架构好像也有很大的差别。下面的博客就将对本次作业我的设计架构进行分析说明，并且对本学期的内容进行一个简单的总结。

1. 本单元两次作业的架构设计

·UML第一次作业

第一次作业主要涉及类图，而在类图中又包括有类的属性（包括UMLOperation和UMLAttribute）以及类的关系。在一开始设计的时候我就认为不应该把他们混在一起写，应该达到一种图结构和节点属性彻底分离的状态（如图中的classproperties和graphstructure包）。当写入的时候分开写入，当查询的时候两部分功能反馈来的信息再在一个统一类（Class）中进行合并。这是整个架构的最基础想法。



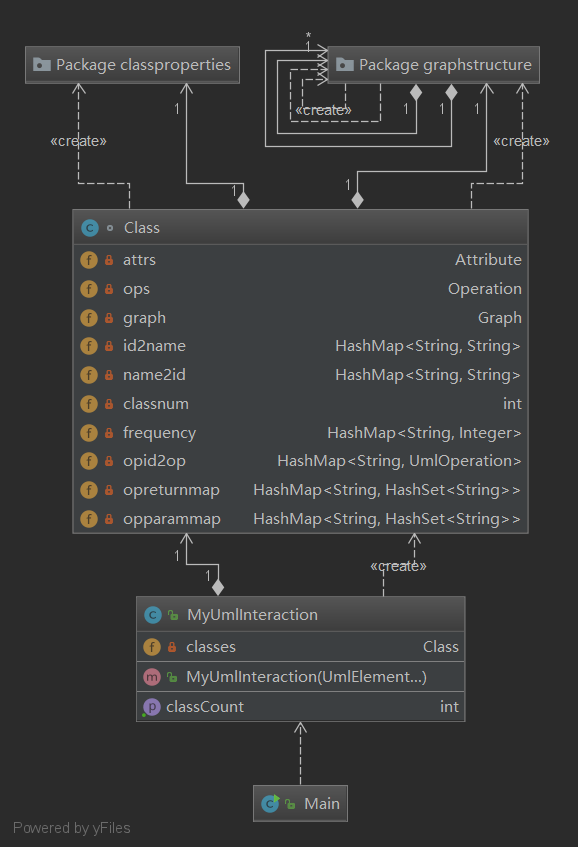
在图结构实现方面，Node类主要进行单个节点信息的反馈，主要保存了一些与此节点有关的信息；Graph则是整个图的一个总览，负责查询节点、查询节点关系、添加节点/添加关系、图算法的实现、缓存的实现（后面优化部分会再次提到）；Association是一个辅助类，适用于UMLAssociationEnd的对应ID还没有进行写入的情况。

在类的属性实现方面，Attribute和Operation类的功能和组成还是比较一致的。主要是用于属性查询功能、频率统计功能、对于某一特定类的特定属性的添加功能、缓存系统的实现，而对于Operation还有专门的参数统计功能。

Class类是管理以上5个类的一个界面，它向上与MyUmlInteraction相接，向下管理5个功能类。实际上信息分发和结果总结的一个类。

在优化阶段，为了防止评测的要求过高，我在每个功能类中设置了一定的缓存系统。主要针对的是有继承的查询。因为每次有继承的查询都需要使用向上迭代查找父节点，有些担心会不会因此报TLE，所以在Attribution/Operation/Graph中添加了对应的节点属性缓存和图结构缓存。每次添加缓存的时候是从此节点开始向上的所有父节点都可以添加对应的缓存。这样在查询密度比较大的情况下能够实现加速。但是其实以上的方法还是非常蠢的，在下一次作业中我将此优化方法进行了较大幅度的调整。届时将看到一个基于节点继承关系的一整个完整的缓存系统。

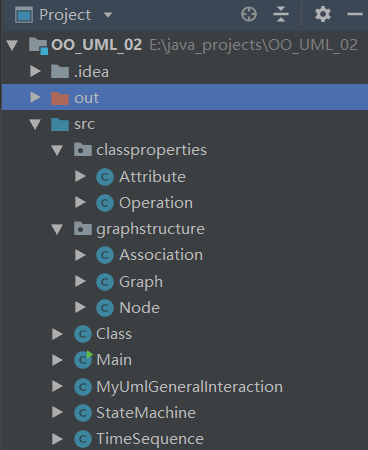
下图是我本次作业设计的UML类图。



本次作业因为自己构造了一些测试样例以及参考了一些同学的测试样例，将潜在的bug都消除了，自己也对一些特殊情况进行了检查。在强测中也没有发现问题。因此在第二次作业中与第一次作业中相同的部分还是沿用了这次作业的设计，基本没有改动。

·UML第二次作业

第二次作业在第一次作业的基础上添加了顺序图及状态转移图的解析和查询、以及几条基本规则的判断。首先说一下与上次作业无缝衔接的三条规则（UML002/UML008/UML009）。



针对UML002规则，我在attr内部添加了有关attr重名的存储单元，并且在graph类中也存储了有关“对端名称”的查询。这个功能的实现相对来说还是比较简单的。

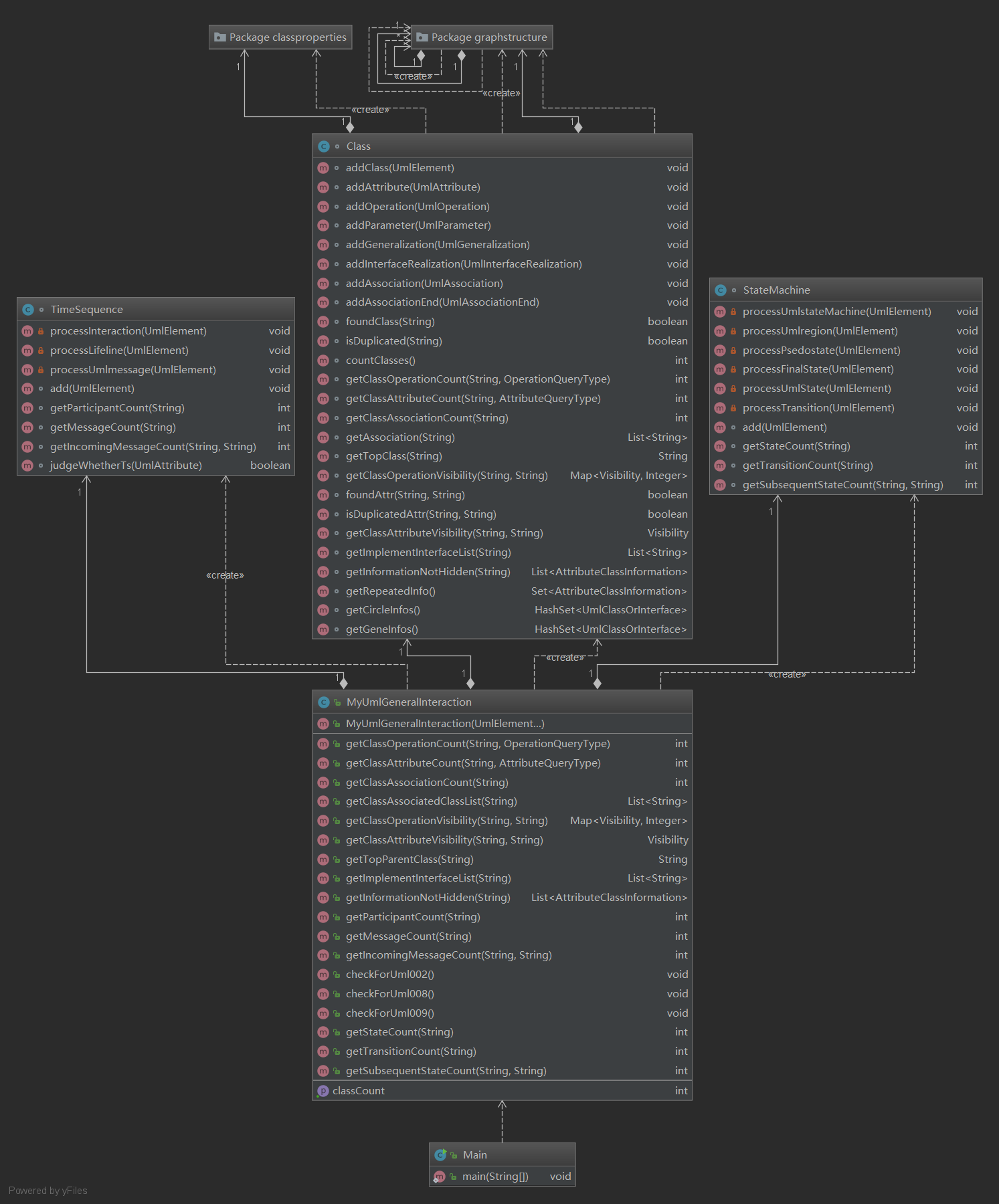
针对UML008规则，我对Node类中存储的信息进行了进一步的规范，这点在这里不再赘述。之后在Graph类中对于每一个节点使用dfs算法找环。这个思路也是比较平实的，再次不再赘述。

针对UML009规则，因为需要找到所有的重复继承接口。我使用了拓扑排序的算法对节点进行排序，从上到下将每个节点的祖先节点全部找到。而针对一个节点是否重复继承的标准比较的是其祖先节点集合的大小与祖先集合模数相加的大小关系。如果不等说明发生了重复继承。

需要注意一点的是，在以上的UML009过程中，实际上完成了一个节点所有祖先节点的查询。我在做祖先节点查询的时候边查询边对每一个节点实现了缓存系统，即一个节点中存储有所有祖先的相关信息，这样实际上我们可以重写在第一次作业中的使用递归方法找父节点的方法，从需要查询到直接可以从节点信息中获取祖先。这样也更进一步地提升了性能。

而对于顺序图和状态图查询的完成，一开始看到指导书之后可以说是云里雾里。因为确实有很多种歧义。并且看到同学们以及老师的解答也是越看越晕。好在助教同学及时出面为我们解释了评测机的数据组成，正如吴老师所言“真是最大程度的简化了”。所以这部分我认为在简化之后真的没有什么问题。我的设计也仅仅针对这么简单的数据进行构建，在此也不赘述。唯一需要说明的是在某一状态的所有后续状态查询中我使用了BFS。

下图是我本次作业的UML类图：



这次作业能够通过和后期助教同学能够公开数据的特征关系很大。在没有公开之前我甚至觉得这次作业也许要无法完成了。最后也是在经过大量测试之后验证没有问题，也成功通过了强测。再次感谢助教。

1. 四个单元中架构设计及OO方法理解的演进

第一单元：复杂多项式求导

架构设计：在第三次作业中封装了factor和term类，对一些简单情况进行了一定的优化。但是没有使用任何的设计模式、OO思想。

OO方法理解：OO的好处到底是什么？我为什么一定要把程序写成这个样子？这阶段还在摸索。

第二单元：多线程电梯

架构设计：对于封装成不同的类有了较好的认识，能够降低不同类之间的耦合度。并且能够保证每个类中的每个方法写的都功能比较单一。也使用了一些OO中的经典模式，比如单例模式、工厂模式，使用这些模式也简化了代码，使代码的管理更加简洁明了。总体来讲我认为在这一单元还是比第一单元有了一个质的提升。做的不好的地方是本次作业的可扩展性仍然一般。虽然第一第二次作业完成了可扩展性，但是很明显前两次作业的内容基本上都是一样的，也就不存在什么可扩展性了，到了第三次再次只能推到重来。究其原因还是因为有些编码太细节不够抽象。

OO方法理解：对于一些经典的设计模式有了深入的了解，逐渐明白了面向对象的意义和面向对象方法在多线程编程中的作用。

第三单元：地铁查询系统（JML建模）

架构设计：基本上可以划分为三个阶段——数据存储环节、算法实现环节、查询功能环节。在完成时注意了上下级的层次关系、各个类之间的耦合度以及方法功能的单一性。在存储的时候也使用了多种多样的手段和方法。并且使用了一些优化手段实现了缓存系统，这一过程也实现了算法复用。

OO方法理解：JML建模在工程开发中的作用、设计模式的灵活运用等。

第四单元：UML类图、状态图、顺序图查询系统

架构设计：总体上还是分为数据存储、算法实现、查询功能。

OO方法理解：应该主要是理解了UML建模，明白了这种建模手段的好处。并且能够使用一些工具进行UML建模并能够解析UML建模。

1. 四个单元中测试理解与实践的演进

第一单元：

基本上是靠自己编写测试数据+同学们共享测试数据完成测试的。针对一些可以优化的情况进行了高强度测试。还编写了相应的python程序来随机生成测试数据。

第二单元：

主要是使用python+管道的方法进行测试来模拟实时输入。并且在最后对于产生的数据使用python画图来看电梯的运行轨迹，以找到相应的问题。后续阶段也使用了一些随机数据的生成手段进行测试。并基于多线程程序的特点对同一测试数据进行超过10次的结果复现。这一单元我们还学会使用Jprofile检测CPU时间。

第三单元：

实现了两方面的测试。第一是JUNIT的测试，每次JUNIT测试单元都会随着情况的改变而改变；第二是编写了一定的测试数据对一些边界条件进行测试。

第四单元：

主要是用starUML建模，编写mdj文件，后转化成合适的测试样例进行大量测试。

1. 课程收获

本学期的OO课虽然有很多痛苦之处，但是也不乏探索的乐趣。回顾下来收获也颇丰，我认为OO课程在大三开设也有好处，因为他对于以往学过的知识都有大量的回顾，甚至将以前理论上的东西有了一定的实现。

1. 领悟了大部分面向对象编程思想的精髓，学会使用Java进行面向对象程序的开发；
2. 对面向对象的模式有了很多的了解；
3. 对于架构有了非常深入的了解，根据不同的需求进行了各种架构设计，并且考虑了更多有关架构扩展性等方面性能；
4. 学会了用各种手段构造测试样例、进行各种类型的测试，也学会了用一些工具（例如JUnit和Jprofile进行测试）；
5. 初步了解多线程编程，了解多线程的测试手段；
6. 再次强化了计算机专业多次强调的高内聚低耦合的设计思想；
7. 一再强化了的代码风格
8. 完成了各种之前学习过算法的具体实现，能够了解这些算法在运行时的表现。
9. 对面向对象课程的三点具体建议
10. 实验有时候感觉有点与上课内容脱节，有几节实验课有一种为了做实验在做实验的感觉。并且难度好像把握的不是特别好；
11. 最后一次作业指导书确实没说清楚，难以下手，本身说是留给我们两周的时间，实际上最后等所有点都明确了也许一周时间都不到了；
12. 理论和作业还是有一点点的衔接不上。老师课上讲的理论同学们的作业实现差别有点大。