引言

本单元以一个图书馆管理系统为背景，不断迭代，并且要求我们在迭代开发的同时画出UML图，较好地锻炼了我们的UML建模能力。

正向建模与开发

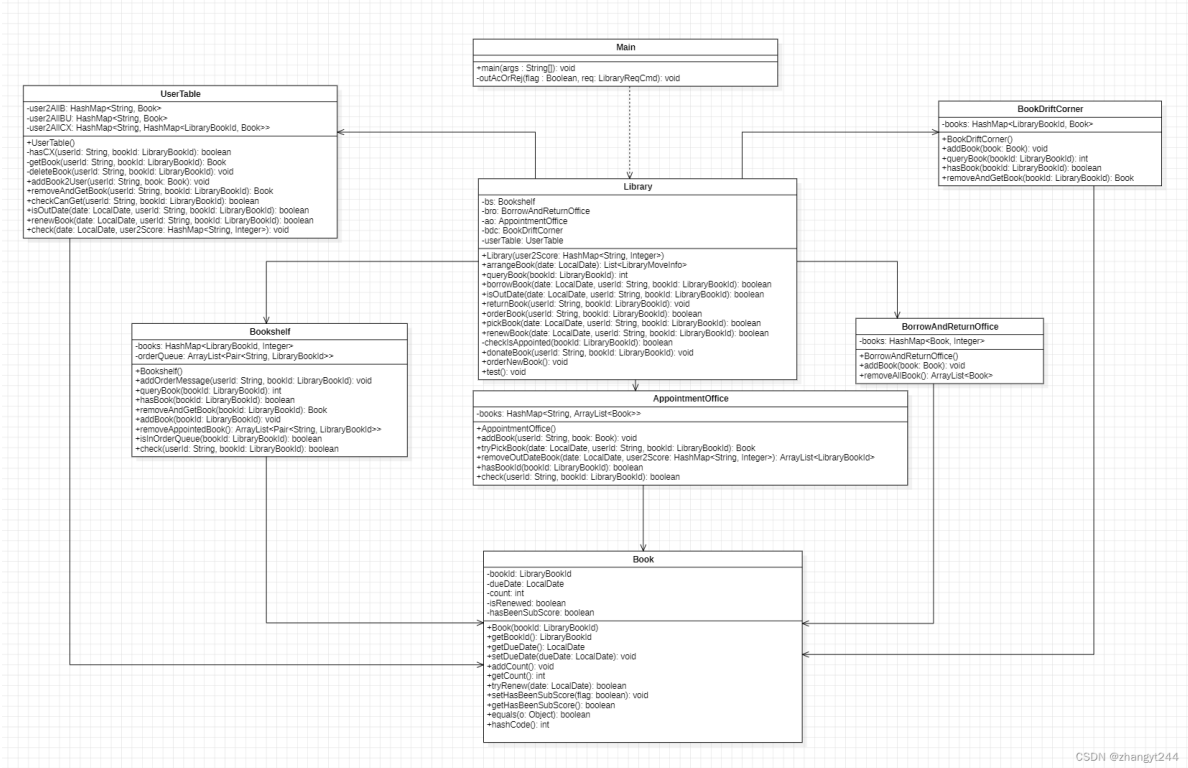
当我完成这三次迭代作业后，我认为本单元真正想让我们学会的不是一些编程的奇技淫巧，而是让我们学习正向建模与开放的方法论。正向建模与开发这种软件工程方法论通常用于设计与开发软件系统，重点放在系统的**设计**阶段，它通过创建模型来描述系统的结构和行为，然后才是将这些模型转化为实际的业务代码。

实际上，在做本单元第一次作业时，我对所谓的UML建模是持怀疑态度的，因为我感觉作业开发量并不大，完全可以一鼓作气写完。但是等到了第二次、第三次作业的时候，我发现由于题目量的增加，我无法在大脑中一次性地想出所有细节了，这时候我意识到了UML建模的合理性。UML建模可以对开发的项目做出高层的抽象，然后在我们实际编写代码时发挥指导的作用（当然老师上课也提到根据UML图可以实现代码的自动化生成以及项目的高效维护，不过可惜的是在OO课程上并没有体会到，但是我认为这些功能在将来工作的时候是会体验到的）。

当然，我在编写代码前绘制的图并非标准的UML图，而只是在纸上画一些极为粗糙的思维导图，不过我觉得这并非违反课程组的本意，相反这是十分贴合正向建模与开发概念的。如果直接在StarUML上绘制UML图的话，由于在编写代码时难免有改动，那最后修改UML图可能就要比重新画一个都麻烦了。

本单元架构设计

类图



架构分析

本单元由于我一开始采用了一个比较好的架构，所以在迭代过程中并未进行重构，所以在此以最后一次作业的架构进行展示。

- `Main`类主要负责输入与输出，并将所有请求传送给`Library`类。
- `Library`类可以说是本单元架构的核心类了，只有一个实例化对象，它负责将所有请求进行分析并将其分发具体的业务代码，从而实现特定的功能。
- `UserTable`，`Bookshelf`，`AppointmentOffice`，`BorrowAndReturnOffice`，`BookDriftCorner`类是图书馆的组成部分，与`LIbrary`一样，均是只有一个实例化对象。它们负责实现具体业务功能，管理具体的数据结构。
- `Book`类有多个实例化对象，内部封装了诸如`dueDate`、`bookId`之类的属性以及`addCount`之类的功能，使得其可以充分地表示大多数业务下的书籍信息，具有极强的复用性。

架构设计不足

虽然说从类图的角度来看，我的架构还算比较清晰。但是我还是觉得有几点遗憾的地方的：

- 在第二次迭代结束后，我才意识到可以实现一个`DataBank`类，用于存储`books`、`orderQueue`等属性，并用static进行修饰，这样可能会使代码结构更加清晰。
- 可以发现在我的代码中，大多数类都只有一个实例化对象，不如将它们全都实现单例模式，这样会使得整体代码更加安全、简单。

不过面向对象正如人生一样，总是没有止境的，对于不满意的地方，也不必非要强迫将其改到完美，这样总归是太伤身费神了。正所谓“何须多虑盈亏事，终归小满胜万全”。

OO课程架构设计思维演进

果不其然，课程组还是让大家对架构设计进行思考。当我与其他学校的高中同学交流面向对象时，我发现大多数学校的面向对象课程对架构设计的要求是非常低的，似乎他们的面向对象课程改名Java编程与设计更为合适。我在回答课程组给我们的问题前，我想揣测一下课程组为什么对我们的架构设计能力的培养这么重视。

当我们面对复杂系统性工程时，我们就不得不对问题进行分层。以开发一款飞机为例，总师在大多数情况下仅需要考虑飞机这个整体组成的对象以及如机翼、机身、发动机等各大组件之间的协同，而无需专门考虑一个"微小"零件对整体所产生的影响。这些零件的设计实际上已经是更低层次的事情了，并非总师所应当考虑的，而应当是其他工程师所应负责的事情。这种站在一个极高的视角设计整个大型工程的分工协作的能力本质上就是架构设计的能力。我相信北航想培养出来能堪此大任的毕业生而绝非一个个悲催的程序员，而这就需要培养我们的架构设计方面的能力。

言归正传，接下来我介绍一下自己在四个单元中架构设计思维的演进：

- 在U1中，我学习到了层次化设计的思想，这也是我认为我在OO课程中学习到的最有用的思想。由于在第一单元刚开始的时候，我和几位OO学得比较好的学长交流了许多，所以我在第一次作业中设计出来了比较优秀的架构。后来，由于作业复杂度的上升以及自己在最开始架构设计能力的不足，我的程序的耦合度逐渐有了变高的趋势。所幸我学习了一些分析耦合度等指标的工具，最终在迭代中将耦合度控制在了一个合理的范围。
- 在U2中，我学习了多线程程序的设计方法。与上一单元不同，在这单元的学习中并没有与学长有多少交流讨论，所以我在最开始的架构并不算多么优秀，写出来的代码的扩展性较低。经过几次作业的训练后，我写出来的代码大多具有了较好的扩展性。
- 在U3中，我学习了规格化设计。在这一单元我初次接触了JML这种规格化语言。在这之前，我很少认真写注释或者文档，通过对规格化设计的学习，我意识到了与他人协作时规格的重要性，注重起了代码的可读性以及注释的质量。

- 在U4中，我学习了正向建模。此前的几个单元，要么已经给出规格，要么重点学习某个知识（如多线程），要么指导书（在这里起到了甲方的作用）给出了具体明确的要求，很难模拟真实的开发场景。在第四单元，我们接手了一个图书馆管理系统的开发任务，极大地锻炼了我们对整体进行全面严密的设计思考的能力。

OO课程测试思维的演进

在OO课程的学习中，我的测试主要有四个阶段：

- 在最开始写完代码后，使用样例或者自己手搓几个普通的例子进行最基础的测试。
- 与朋友合作搭建评测机，并使用常量池，调参等技巧增加数据强度，对自己的代码进行大范围的测试。
- 自己手搓一些特别极端的数据，比如第三单元构造好几千条`qtvs`，对自己的代码进行压力测试与边界测试。
- 在完成上述所有测试后，与朋友互相走查代码，模拟互测流程，也就是进行所谓的白盒测试。

课程收获

在这学期16周的OO课程中，我主要有四点收获，首先我收获了许多新的知识，比如多线程程序的设计方法、UML图绘制方法、JML语法、Java语言设计方法。其次我的抗压能力得到了极大的提高，U1与U2时熬夜debug的场景我可能能够记一辈子；然后，我在OO课程中学到了很多有价值、有深度的思想，诸如层次化设计的思想、规格化设计的思想，与前两点收获相比，这一点收获显然更为重要；最后，我在学习OO的过程中，结交了许多朋友或者学长，如fzy、wxm、wxf、hj、hlq、nr、wsj、gpf、dhj等等等等，我想这种收获是无价的。