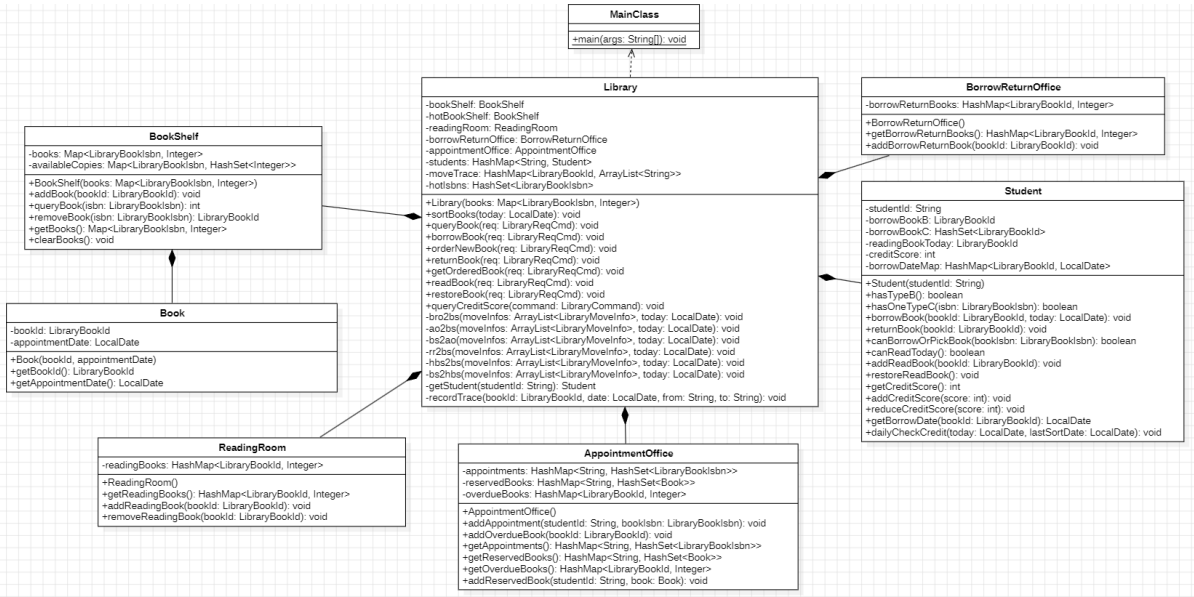
BUAA_OO_第四单元总结

漫长精彩的OO之旅经历一学期终于要结束了。在此总结第四单元正向建模的开发经验，并全面总结四个单元的整体内容。

整体架构

第四单元模拟图书馆系统的最终 UML 类图如下，在正向建模中，由于是先确定类的结构再实际写代码，因此我在践行面向对象基本编程思维、高内聚低耦合的封装思想、单一职责原则等思想上均有了非常大的提高，写出了简洁而清晰的高质量代码：



- `MainClass` 类负责输入解析
- `Library` 实现图书馆的核心功能，对于不同指令进行分发处理，并拥有负责书架、热门书架、用户、借还台、预约处的实例，负责它们之间的交互。
- `Bookshelf` 等具体的地点类封装该处特定的功能
- `Student` 类封装用户的行为，更加体现单一职责原则。

正向建模与开发

就作业而言，我们使用了类图、顺序图、状态图等 UML 语言绘图。他们分别由各自的功能，分别描述程序的**整体架构分工职责**、**类与类之间的消息路径**、**某个对象生命周期内的状态迁移**。这三张图绘制结束后，应该对代码已经有了非常深入的思考，或者可以说**代码几乎已经可以唯一确定**。在 OOpre 课程和前三个单元中，我对 UML 图的功能认识仅限于总结自己的设计方案，但从该单元开始，正向建模的挑战在于必须清楚整个架构中每个功能，设计好协作关系后进行开发。

我认为最终的代码设计和UML模型设计之间的追踪关系如下：

- 如果说 JML 语言像翻译任务，那么UML正向开发更像给出了一个作文每段的核心句，还给出了各个段落之间的逻辑关系，要求补全整篇作文。因此，即使具体代码实现不是确定的，但每个类的各个方法功能已经完全确定了，代码几乎已经可以完全确定。
- 状态图中书本的状态是可以自定义的，也反映了开发之前需要对书的流向和可能所处的状态做出明确的判断和确定。代码中特定的方法在某些条件下会实现具体的实例的状态的迁移。
- 顺序图侧重于描述各个类之间的消息传递关系，在写代码之前完全精准画出有一定困难。
- 因此，我的思路是：对于类图，完全可以在设计架构（具体实现之前）就完全准确画出，因为类图和代码最为接近，分析需求时能完全提炼出每个类需要什么样的功能；而对于顺序图和状态图，由于首次接触并不熟悉，在设计阶段往往不能完全准确画出需要完成代码后进行微调，更宜于总结分析时使用。

大模型辅助正向建模

几年来随着 LLM 的兴起，已经极大影响了教育和科研领域，当时在实际工程应用中（例如阿里等互联网大厂）应用率不足 20%。为什么会这样？除了公司规定和安全性的问题，最重要的是大模型对于长输入和复杂需求以及多轮迭代出现的**幻觉**。

对于高效辅助正向建模，我总结出以下注意事项：

- **拥有一个强大的 LLM。**GPT-4o 和 Claude 是良好的选择。
- **注意 prompt 的方法学设计。**本单元的创新实验中已经教会了我们许多 prompt 设计方法，同时也要在实践中探索迭代方法。
- **不要把代码当作自然语言投喂给 LLM。**在代码量特别大的情况下（比如迭代任务，已经有了规模的许多类的代码），如果使用的不是 API 而是网页前端，不宜把代码和自然语言需求同时输入到对话框中。因为LLM如果把代码当作自然语言进行处理，做注意力机制那些操作，效果会欠佳。正确的做法是使用 `repomix` 等工具将大规模代码转化为 **AI-friendly 的格式（例如xml文件）**，然后将该文件发给 LLM，需求通过自然语言描述通过对话框投喂给 LLM。

在本单元作业中，类图的设计我并没有依赖大模型，而是通过自己的经验实行，能够**体现我个人的编程习惯和设计理念，而不是完全被 AI 取代。**大模型辅助了方法设计，提高了开发效率。

四个单元中架构设计思维的演进

在 OOpre 课程多次迭代中，我在架构方面吃了苦头，导致后续迭代过程中困难重重。因此，在正课中，我在架构设计中投入了大量的精力和成本，产生了良好的效果。

- 第一单元：**递归下降，架构为先。**第一单元是对架构和面向对象设计思想最为精妙的体现。具体而言，涉及到许多因子：三角函数因子和自定义递推函数因子等。有些做法把函数当作预处理阶段进行，即在输入解析时进行展开。这样的写法似乎更为直接和容易实现，但是往往会导致后续解析深度太深导致超时。因此，我坚持将这些结构全部按照指导书的要求，放到因子层处理，体现了我设计结构的合理性。
- 第二单元：**线程协作，同步互斥。**第二单元接触新概念：多线程编程，也是我收获最大的一单元，其中困难的点就在于线程之间的**通信**、线程之间的**同步和互斥。**因此，从架构的角度，**生产者-消费者模型**发挥了重要作用，必须清晰地将线程对象和调度类的功能相分离，各自实现合适的功能。
- 第三单元：**规格约束，优化算法。**第三单元不涉及自己设计架构，核心在根据具体的 JML 规格进行八仙过海各显神通的具体实现。因此更偏向于算法维度，降低复杂度是核心。
- 第四单元：**正向建模，注重设计。**我认为改单元也是非常注重架构设计的一个单元，但是由于开发和迭代没有第一单元大，可能优势并不如第一单元明显。但是，经历了三个单元的经验，我对于架构设计已经有了充足的训练和理解（从复杂的大段需求中提炼出类和方法），能够较为科学设计出面向对象的框架。

四个单元中测试思维的演进

在正课中，我对第二、第三单元几次作业均自行搭建了自动化评测和对拍程序，对测试有较大的理解提升。我认为我的测试有以下几方面演进：

- 第一单元：仅仅有对于**边界测试**的感性认识，通过**手工构造**具有一定复杂度和边界条件的测试数据进行测试，通过代值的方式和同学对拍。好处是人工构造数据具有导向性，会对一些边界条件进行有意识的测试（例如 `sin(0)^0`、`((((((((((x^8)^8)^8)^8)^8)^8)^8)^8)^8)^8)^8` 等）。
- 第二单元：花费相当多成本搭建了评测机。**数据生成器**践行了**压力测试**，同时通过不同阈值限制控制不同位置数据流的内容，实现**一定的人为控制**，而不是纯粹随机；**评测程序**相当于用另一种思维（面向结果的思维）把作业重写了一遍，花费了大量的时间，de 评测机的bug也花费了很大的经历，但最后实际投入使用过程中效果非常理想。
- 第三单元：**单元测试**、**集成测试**和**回归测试**。每次作业中对于较高复杂度和容易出错的方法进行 Junit 单元测试，同时对系统功能集成测试，在迭代中同时验证上一次作业强测的功能，防止干扰之前行为。由于功能过于复杂，复现功能不现实，因此我构造了**数据生成器**和**对拍机**，用于和同学对拍或者和 java 标程对拍。
- 第四单元：融合了之前的测试策略，但由于架构较好，并没有在测试中出现很严重的错误。

课程收获

这门课虽然是一门开发课，但是我认为是我求学生涯中上过的最令人印象深刻的课（没有之一）。不论是内容上高强度作业的创新和精彩，还是形式上强测互测的有效有趣，体验感都是所有专业课包括中学课程中最精彩、收获最大的。尤其是在代码内功角度，无论是C语言还是java等各类语言，现在在看以往或者大一写的代码，能明显感受到有质的提升。

然而，这门课和所有计算机开发课程一样，并不是一帆风顺的，一定是负反馈远远高于正反馈。虽然我整体上能力有很大提升，但是也遭遇过许多挫折，尤其是第二单元。第二单元第二次作业正值清明假期，我精心设计、反复打磨，总共花费了20多个小时在细节的性能分争取上。然而，却因为量子电梯卡时间戳太死，导致强测0分。把量子电梯关闭后即可全部通过。同时，我也花费了很高的成本和精力尝试影子电梯等方法，在实践中全面了解了这些进阶方法的所有细节和优劣。然而，从结果的角度，确实给我了很大的打击，影响了我的情绪，在那几天都非常失落。从现在回看，这何尝不是一次收获的机会，让我更加辩证全面理解了正确性和性能之间的权衡关系，让我在多线程编程中的能力和技巧均得到了质的提升。

总结来看，这门课我影响最深刻的内容如下：

- **实验**的创新设计。优秀的助教们打破了传统实验的局限，主动拥抱时代大势，创新性地将 LLM 辅助设计融入实验中，在实验设计中下了很大的功夫，教会了我许多和大模型的交互方法，也逐步培养了**正确合理使用大模型辅助开发的意识**。
- **调试**能力的提升。我能够熟练使用条件断点、多种执行方式profiler 分析器等高阶调试方法，极大提升了 debug 的效率，尤其是第二单元多线程设计中的分析和纠错，极大充足了我的开发和测试经验。
- **测试**方法的进步。从本门课开始，我主动搭建科学的自动化评测机，在强测和互测中取得了非常好的效果（周日评测一开然后出去玩回来收割太爽了）。评测机分为几个部分：
  - 数据生成器：从一开始完全随机生成，到后来有意识地进行参数和阈值控制，实现更有强度和合理的数据质量。
  - 评测程序：可以重新实现逻辑，也可以进行对拍，两种我均有实现。
- **大模型**辅助开发能力的提升。主要分为以下两方面：

- 第一，我逐步认识到大模型在开发中的正确地位。对于整体架构和重复性劳动，大模型可以极大提升开发效率和科学性；然而，对于复杂的任务需求和高效的算法，大模型还无法替代优秀开发人员的功能。同时，在未来实际进入产业界时，要特别注意安全性问题。
- 第二，我使用大模型更具有方法学和科学性。主要体现在 prompt 的有效设计和代码输入形式的改进（上述提到的 `repomix`），都促进了我是用大模型的效率和质量。

总之，感谢课程组各位老师和助教，让我遇到了最精彩的课程、作业和实验，让我的开发能力、测试能力、大模型使用能力得到了质的飞跃。