OO_Unit4总结

一、正向建模与开发

所谓正向建模与开发，就是一种结构化的软件开发方法，它从分析系统需求开始，通过创建抽象模型（尤其是像UML这样的面向对象模型）来指导和规范后续的编码实现工作。其本质是一个从抽象设计蓝图到具体代码实现的正向推进过程，旨在提高软件开发的规范性、可维护性和效率。

以本单元OO课程为例，可以分为以下步骤：

1. **需求分析**：指阅读OO课程的指导书，明确题目的要求和逻辑。
2. **系统设计**：设计系统的总体架构和核心模块，构思代码的设计文档。
3. **对象建模**：是整个任务的核心，将指导书中的内容抽象成一个个对象，绘制UML类图、状态图和顺序图，设计这些对象之间如何协作和交互。
4. **编程实现**：根据前述步骤中创建的模型和设计文档，使用Java编写代码，实现图书馆的模拟。
5. **测试与部署**：对编写的代码进行测试，确保代码符合指导书需求和设计，防止强测爆炸。

二、Unit4架构设计

本单元的主题是正向建模与开发，但惭愧的是，我只在第一次作业中进行了相关实践，后两次作业都是先编程再画图。

1. 第一次作业

第一次作业是这三次作业中相对难的，是从0到1的过程。

在这次作业中，我将`Library`类作为程序的核心，所有的请求在这个类中产生，在这个类中解决，同时，`AppointmentOffice`、`BorrowReturnOffice`、`BookShelf`作为部门负责处理相关的请求，由`Library`类向这些部门中发送相关请求，其中`BookShelf`较为特殊，所有其他部门都会涉及到“从书架取书”或是“向书架还书”的动作，因此`BookShelf`是其他所有部门的成员变量。

2. 第二次作业

第二次作业将书架分为热门书架和普通书架，增加了阅览室和相应的阅读、归还请求。

对于书架分类，我采取将原先`BookShelf`类中存储书籍的列表`books`改为两个列表`hotBooks`和`normalBooks`，其他部分几乎不需要做改动，这样方便热门书籍和普通书籍之间的交换，每次开馆时只需要将两类书籍在这两个列表之间进行转移即可。

对于阅览室的实现，只需模仿第一次作业中`appointmentOffice`和`BorrowReturnOffice`的实现即可。

3. 第三次作业

第三次作业增加了用户信用分和图书借阅期限限制。

对于用户信用分，为`User`类增加新的属性`creditScore`表示该用户的信用分，对于图书借阅期限，为`Book`类增加新的属性`borrowedDate`，表示该书籍剩余还书时间，然后根据指导书要求，在相关请求上对用户的信用分进行加减和作出限制即可。

三、大模型使用总结

关于大模型的使用，在第三单元的时候就已经讨论过，当时我得出这样的我认为可用的方法：

1. 首先向大模型提出自己的目的，将代码交给大模型（最好是直接上传代码文件），询问大模型有何想法。
2. 根据大模型给出的建议，选出你觉得最好的一条，然后让它给出实现步骤，并将实现步骤讲解给你，如果此时你发现了实现步骤是正确的，继续，否则，指出错误并让其修正。
3. 让大模型顺着实现步骤一步一步实现，如果是直接让大模型一次性解决大规模的任务，效果并不好。
4. 在本地实现大模型给出的结果，进行测试，如果出现问题，将错误内容反馈给大模型让其修改（最好是自己在本地调试一遍，尽可能将错误结果讲清楚），重复此步骤直到正确。关于测试，也可以由大模型实现。

但是经过最后一次上机实验课后，发现ROSES框架更好用，虽然不知道原理，由于第八次实验已经关闭，因此以下ROSES框架摘自网络：

> ROSES框架
>
> 关键结构字段说明：
>
> i. **角色(Role):**指定大模型的角色，比如，他应该扮演专家、助手或某个特定领域的顾问。
>
> ii. **目标(Objective):**描述您想要实现的目标或您想要大模型完成的任务。
>
> iii. **场景(Scenario):**提供与您的请求相关的背景信息或上下文。
>
> iv. **预期解决方案(Expected Solution):**描述您期望的解决方案或结果。
>
> v. **步骤(Steps):**询问实现解决方案所需的具体步骤或操作。

此外，我还在网络上找到了一些其他的框架，并不都适合编写代码，但是也许会在其他地方用到：一些AI框架

四、设计思维演进

- Unit1

  第一单元的主题是层次化，也是唯一重构过的单元。第一次作业由于时间短任务重便直接以OOpre时的第七次作业作为模板，直接在上面修改得到的，但是第一次作业最终的架构很难实现第二次作业的三角函数，因此便自己重构，分别增设单项式类`Mono`和多项式类`Poly`，初步实现了层次化。

- Unit2

这一单元的主题是多线程，感谢上机实验的馈赠，这一单元我所采用的架构全都是沿袭自第一次上机实验，代码架构完美实现"生产者-消费者"模型，调度器线程和输入线程作为生产者，候程表作为托盘，电梯线程作为消费者，三者合作实现作业要求，后续增加的临时调度和双轿厢电梯只需要看作电梯的特殊动作，对电梯线程进行修改即可。

- Unit3

  这一单元的主题是规格化，JML规格已经定死了架构，只要正确实现JML规格，就能有个很好的架构。

- Unit4

  这一单元的主题就是正向建模与开发，于是在第一次作业中我也采取了先建模后开发的方法，画完类图大纲之后编程效率确实提高很多，但是后两次作业由于迭代难度很低，便没有再采用先设计再编程的策略，而是直接编写完代码然后对着代码修改类图。

五、测试思维演进

遗憾的是，这个学期我没有搭建我自己的评测机，除了第三单元以外，其他单元都是根据指导书手动构造样例，或者是白嫖其他同学的评测机。在第三单元，除了指导书要求使用JUnit测试的方法，我还尝试过测试其他方法，得益于上机实验已经给出的JUnit模板，曾经在OOpre认为JUnit比较鸡肋的我也感受到了JUnit测试的强大之处。

研讨课上大佬分享了如何使用大模型来搭建评测机，也许我会在暑假尝试一下。

六、课程收获

整个学期下来，收获最大的当然是加深了对面向对象的认识，同时在高强度的任务下，抗压能力得到了很大进步，同时编程能力也进步了不少，学会了JUnit测试、认识了JML规格、初步了解了多线程、还了解了IDEA一些奇奇怪怪的用法……

在课程最后结束的时候，我并没有像上学期CO结束时“解脱”般的松了口气，而是有种“意犹未尽”或是不舍的感觉，大概是因为后两个单元冲淡了前两个单元的“罪恶”而带来的幻觉，也许吧。总之，感谢课程组的对这门课恰到好处的设计，感谢助教们对课程的奉献。

再见了，所有的OO。