“软件系统设计与分析”论文

课程：软件系统设计与分析

学生：刘卓舟

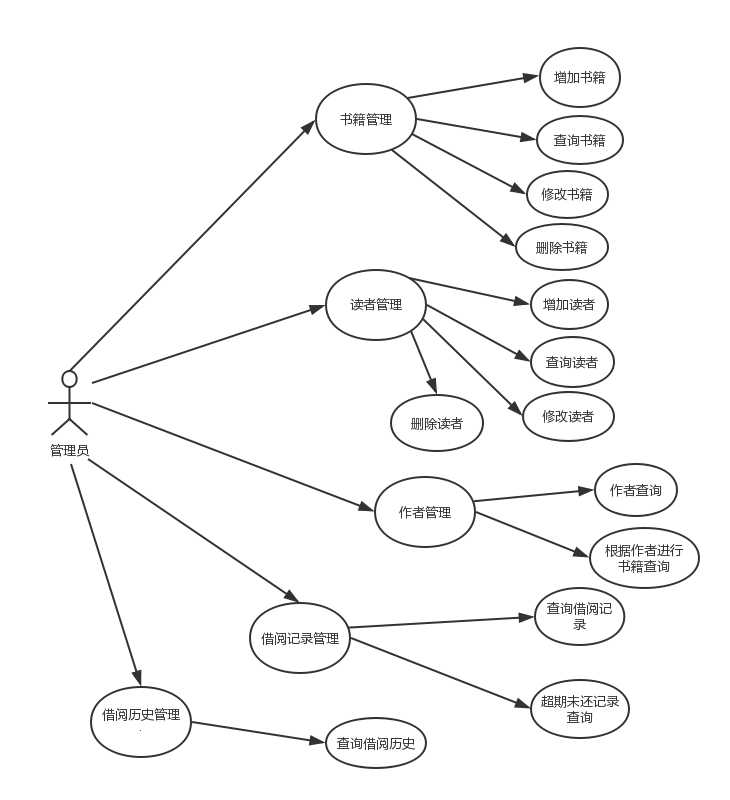
班级：2016计算机三班

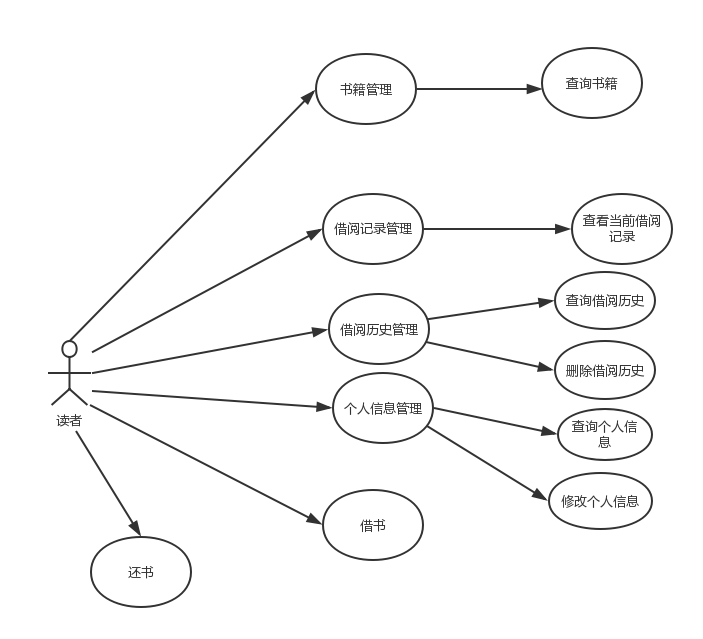
小组：第四组

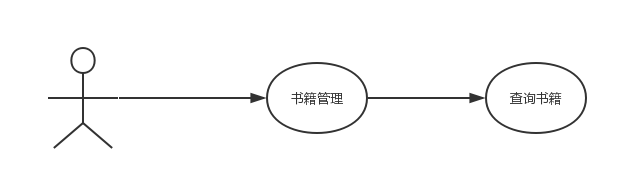
学号：16020031049

老师：张春海

1. **UML类图**
2. 用例图

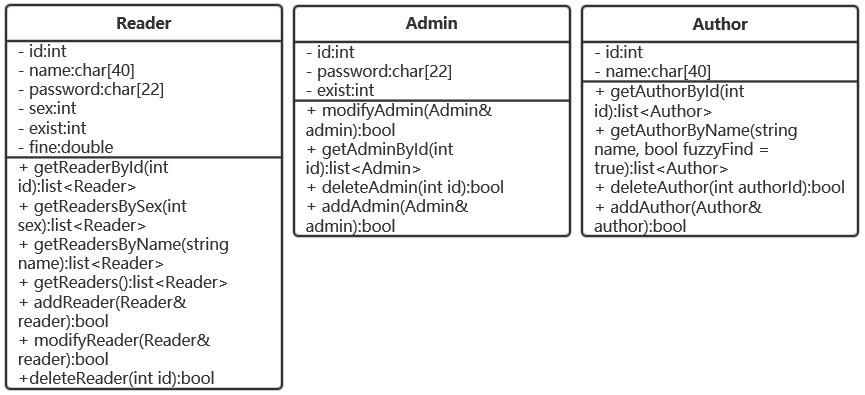






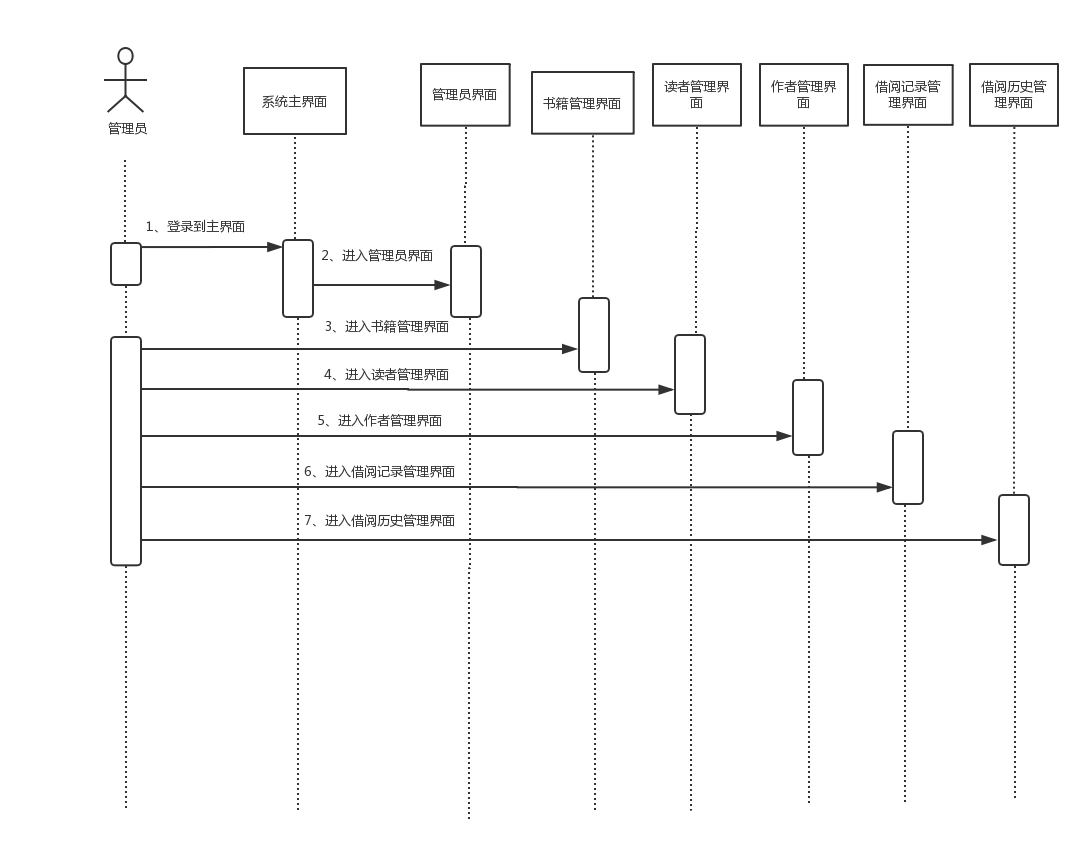
游客

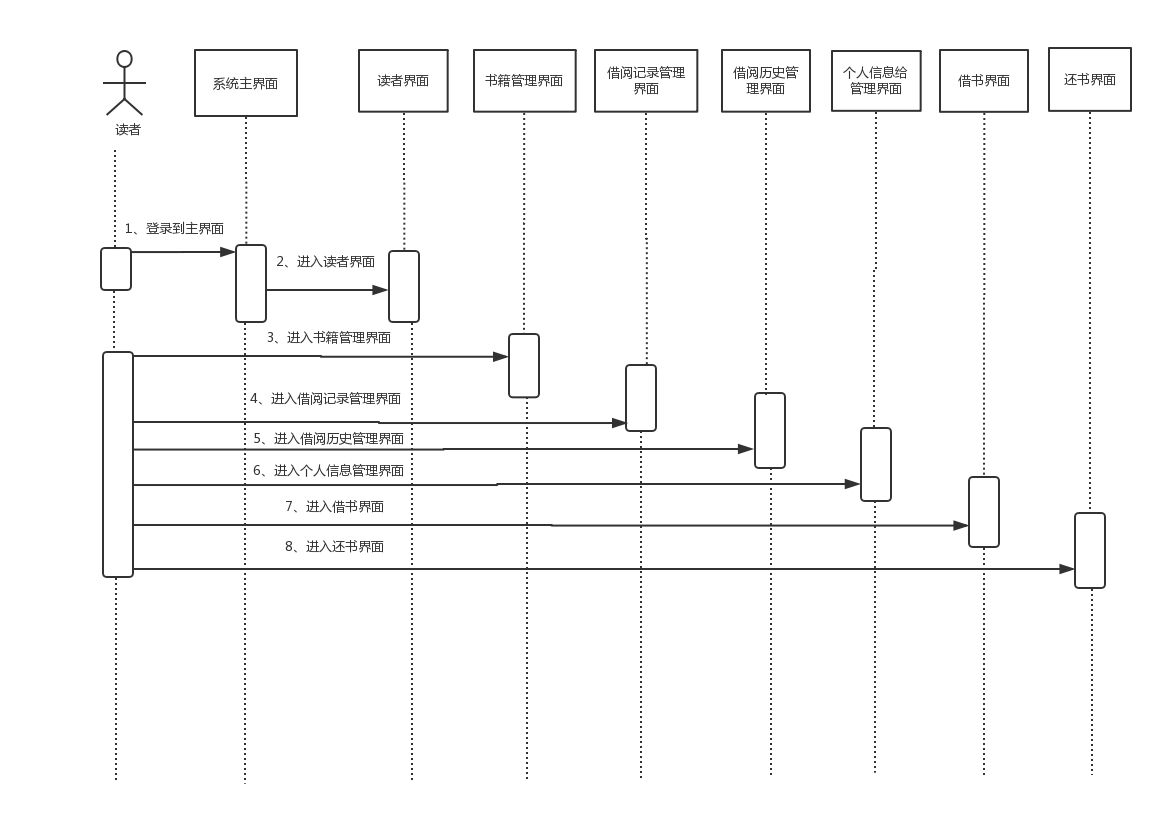
（2）类图

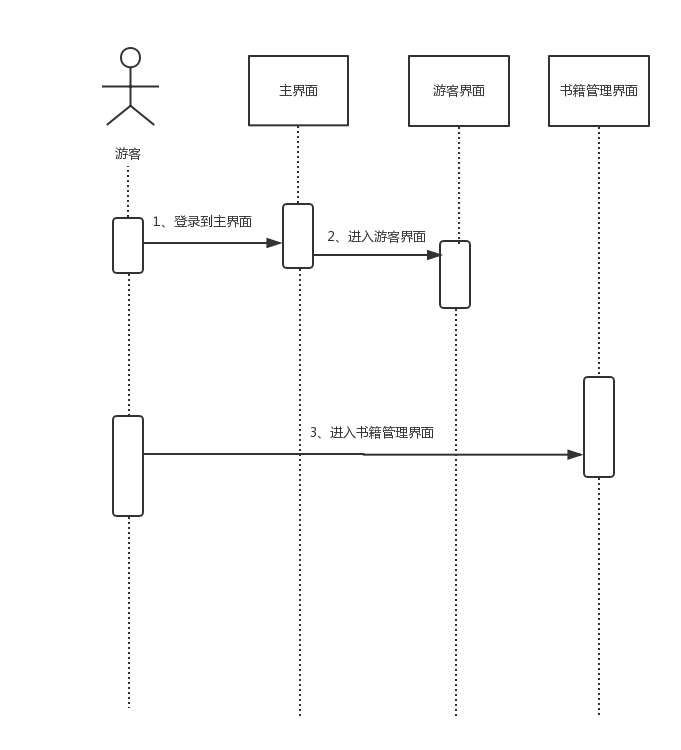




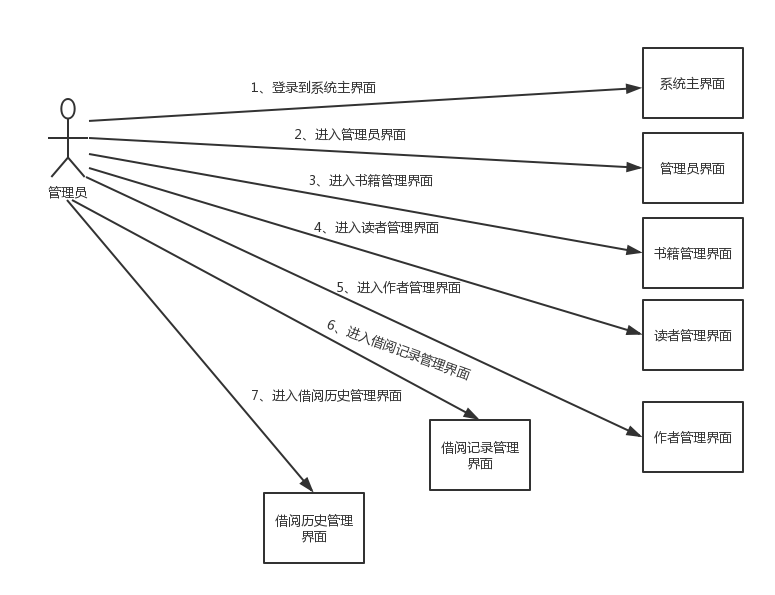
（3）顺序图

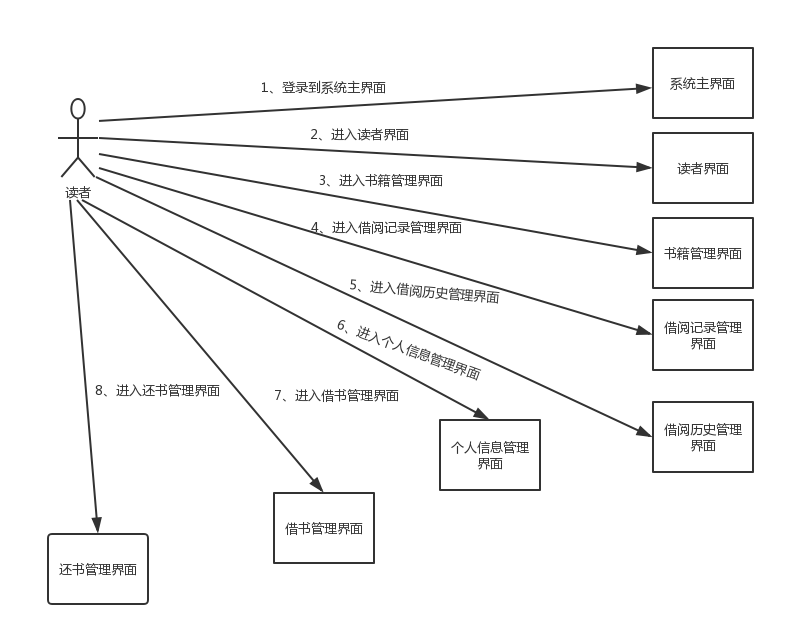


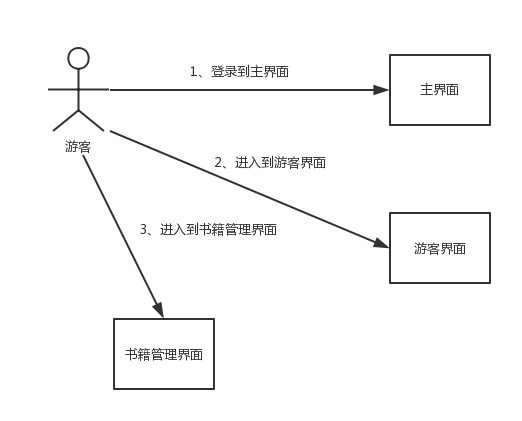




（4）协作图







1. **软件系统设计与分析**
2. **对软件工程的认识**

在我看来，软件工程涉及对人力、资源、时间的合理分配以及各种规范化。软件开发是一个软件工程师们将用户需求转变成软件需求，之后根据软件需求抽象为相应的软件设计，根据软件设计编写出相应的代码，再根据代码进行相应调试，最后拿出成品的过程。

对于一项软件工程来说，软件需求是必不可少的。我们首先要根据用户的需求将其转变为软件的需求，软件需求就是对用户的需求进行取其精华，正确理解，并选择相应的程序语言表达出来的过程。以这次所做的图书管理系统举例，首先我们明确用户的核心需求就是“借还书”，所以我们应该设计出一个以“借还书”为核心的图书管理系统，在此基础上延伸添加其他功能。

当软件需求确立后，我们就要开始软件设计了，UML类图就是一个很好地帮助我们软件设计的工具。首先我们应该大致确立软件系统的功能、流程、层次，并将其化为几个模块，明确其模块结构。在完成大致规划后，可以进行代码的粗略设计，并集中攻克发现的难题。像这次我们所做的图书管理系统一样，在小组集中讨论确立了四大层次（文件层，service层，servlet层，主界面层）以及系统的基本功能和流程之后，我们立即开始攻克“如何用文件存取数据”这一难题。

在软件设计完成之后，就要进行具体的代码编写了，在此过程中，可能会出现各种各样的问题，我们要一边进行代码的编写一边进行调试，并且还要考虑容错性与交互性。我们小组编写代码时，也出现过各种各样的问题，比如输入整数的容错性，如何对数据查重，但所幸大家都坚持下来了，集中攻克难题，大部分问题也得到了解决。

在代码编写完成之后，就要对软件进行测试了。一般情况下，测试过程中会暴露出一些或大或小的问题，有些严重的问题甚至会导致整个系统崩溃。在测试的过程中，首先应该保证系统基本功能的正确，在此基础上再去测试系统的拓展功能，健壮性以及交互性。在测试中，应该先对各个小模块进行测试，测试无误之后再对大模块乃至整个系统进行测试，这样的好处是减少了bug排查的范围，能较为快速地找到问题所在。在我们对图书管理系统的测试中，都是先测试图书管理系统“借还书”的核心功能，再去测试其他功能以及容错性。并且每个人写完自己所负责的代码后，都必须自己先测试，之后交由组内成员测试，确保没有问题之后才能开始下一步的工作。

当软件测试完成之后，整个软件工程的开发还剩下最后一个环节——对软件系统的维护，当然对于我们自己做的图书管理系统而言，由于不具备商业元素，自然不必维护。但是对于真正的软件工程来说，系统维护也是至关重要的。由于在用户的后续使用过程中，可能会暴露出各种各样的问题，并且用户的需求也是随时在变的，所以软件系统应该根据用户的需求相应地调整功能。

总而言之，软件工程具有传统工程的性质，同时，它也具有特性，比如复杂性与抽象性。软件工程是需要合作精神的，只有一个良好的团队才能做出一个好的软件系统。在软件工程中，分析和设计是重中之重，只有良好的分析设计才能奠定良好的基础，才能保证在开发过程中少走弯路。

(2)**面向对象模型的特征**

抽象：抽象是简化复杂的现实问题的途径，它可以为具体问题找到最恰当的类定义，它可以忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，暂时不用部分细节。在我看来，抽象的应用就是类的应用。以图书管理系统举例，书这个物体有很多特征，但是对于这个图书管理系统来说，只需要关注书的名字，作者，出版社，ISBN，类别等特征，不需要关注书的篇幅，内容，出版日期等特征。这时候运用抽象的方法，将书抽象为一个book类即可。

封装：在面向对象的编程中，封装，就是隐藏对象的属性和细节，仅对外公开接口，控制在程序中属性的读和修改的访问级别；将抽象得到的数据功能相结合，形成一个有机的整体，也就是将数据与操作数据的源代码进行有机的结合，形成“类”，其中变量和函数都是类的成员。在我看来，封装的应用主要在于类中属性的访问等级，一般情况下变量设置为private，函数设置为public。当然在实际使用中类中属性的权限等级视具体情况而定。封装可以更好地保障数据的安全，使程序更加具有结构性。

模块化: 在系统的结构中，模块是可组合、分解和更换的单元。模块化是一种将复杂系统分解成为更好的可管理模块的方式。它可以通过不同组件设定不同的功能，把一个问题分解成多个小的独立、互相作用的组件，来处理复杂、大型的软件。模块化具有独立的工作运行模式和分级启动功能。在我看来，模块化适用于大型工程，以图书管理系统举例，每个人独立地完成几个模块，例如查书模块，借书模块，这样既保证了高效性与相对独立性，又能迅速地发现并解决问题。

层次：层次是人们对复杂问题处理的基本方法，将总体要实现的很多功能分配在不同层次中，对每个层次要完成的服务及服务要求都有明确规定，高层使用低层提供的服务时，并不需要知道低层服务的具体实现方法。以图书管理系统举例，我们将整个系统分成了四个层次（文件层，service层，servlet层，主界面层），每个层次分别实现不同的功能，当程序开始执行时，由主界面层开始从上往下层层调用，高层次并不需要知道低层次的功能，只需要调用相应函数即可。这样的好处是使程序的逻辑结构更加清晰，但是弊端就是一定要确保底层功能没问题，否则整个系统就崩了。

（3）**面向对象的其他特征**

对象标识：对象标识即是每个对象所具有的唯一标识符，意味着面向对象中对象的唯一性。以图书管理系统举例，对于每本书，我们都设置了id作为它的唯一标识符以方便接下来的增删改查的操作。

分类性：分类性是指将具有一致的数据结构和行为的对象抽象成类。以图书管理系统为例，将所有书籍抽象成Book类，将所有作者抽象成Author类，将借阅记录抽象为BorrowRecord类。这样的好处是将同一种事物都抽象为同一类，方便接下来的代码编写。

持久性：持久性就是把在内存中的对象保存到文件里，使得程序重新加载后能够完全恢复到退出前的状态。以图书管理系统为例，对象都具有持久性，因为每一个对象所携带的数据经过一定的操作之后都将保存在文件中。这样的好处是使系统的生存周期更长，数据保存时间也更长。过一段时间再打开图书管理系统，里面的数据依旧存在，系统依然能够使用。

并发性：并发性是指在一个系统中，拥有多个计算，这些计算有同时执行的特性，而且他们之间有着潜在的交互。因此系统可进行的运行路径会有相当多个，而且结果可能具有不确定性。这样的好处是资源利用率更高并且程序运行更快，但是由于其具有不确定性，所以很容易导致死锁和资源匮乏等问题。

（4）**自我体会**

在这次课程的学习中，我深深地感受到了软件工程的魅力，进一步加深了对“面对对象”思想的理解。“面对对象”的核心就是“对象”二字，通过程序化思维对某一具体事物进行抽象，只保留我们所需要的属性，摒弃其他无用的属性。进一步强调了对象结构而不是程序结构。除此以外，面向对象的继承性实现了子类共享父类数据结构和方法的机制，在一个已有类的基础上添加新的内容。面向对象的多态性实现了不同的对象收到同一消息产生不同结果的机制，进一步提升了程序的简洁性与高效性。面向对象的封装性不仅有良好的数据隐藏功能，还能让外部接口与内部实现明显地分开，防止了程序相互依赖性而产生的变动影响。面向对象的对象唯一性保证了每个对象都有自身唯一的标识，通过这种标识，可找到相应的对象，在对象的整个生命周期中，它的标识都不改变。“面向对象”的思想可以说不同于以前的“结构化程序”思想，它更加灵活，更加友好。

在图书管理系统的开发过程中，我也收获了不少东西。首先是我学会了不少代码的编写方法，编写程序的能力得到了一定的提高，比如学会了如何用文件流存储、查找二进制数据。其次是初步认识了程序的模块化设计，以前自己编写小程序时，并没有很好的模块化设计，但是对于一个软件系统来说，必须分好层次、模块，先完成一个个小模块，再实现整体。以图书管理系统举例，我们将整个系统分为了四大层次以及大大小小的许多模块，这样使得程序的层次更加分明，编写代码的效率也显著提升。试想若是不采用模块化设计，那么所编写的程序很容易出现逻辑混乱，代码冗长的毛病。除此以外，我还学会了如何与小组成员间分工合作，每个组员所负责的工作独立却又紧密联系着。开发过程中可能会出现很多问题，小组成员间一定要团结，共同去解决问题，每个人都做好自己的分内工作，尽量帮助其他组员解决问题，当意见不合的时候，不妨冷静下来，各退一步，选择最优的那种方法。

在这次课程中，我学到了不少有益的东西，不仅仅是技术上的东西。希望自己以后能充分利用这些东西更进一步。