**UML面向对象的系统分析与设计**

姓名：徐云哲 学号：15020031088 组别：第1小组

1. 根据自己小组完成的系统，运用OO分析及设计方法，画出相关的UML图。包括 ：类图、关联图、用况图、顺序图等。

**一、类图+关联图：**



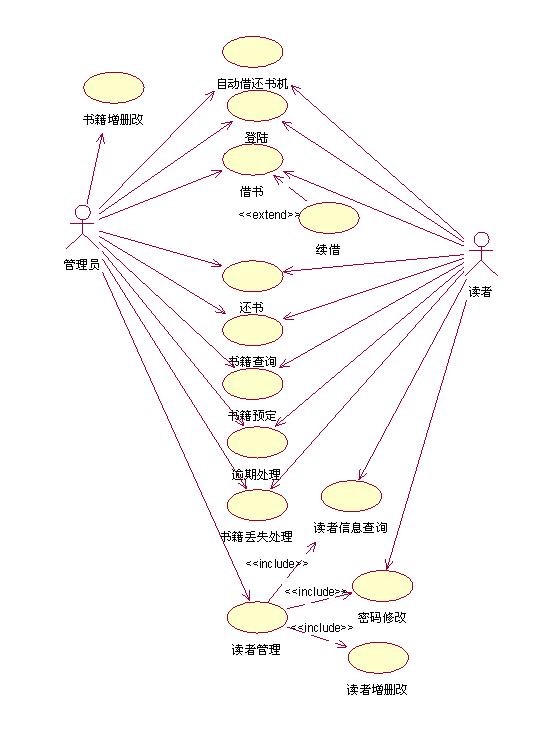


****

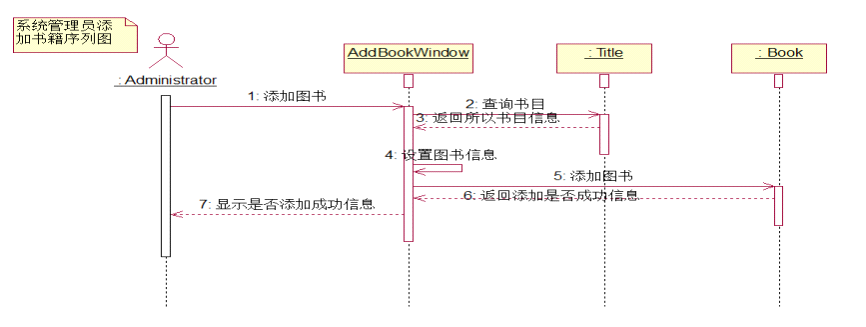
****

****

**二、用况图：**



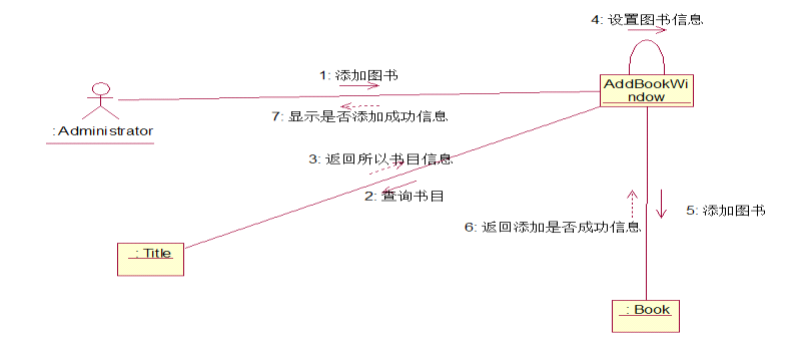
**三、顺序图+协作图：**

**1.1管理员添加书籍顺序图：**

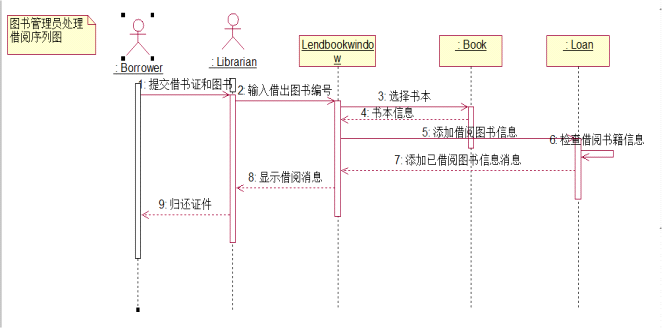
顺序图说明：

添加图书：添加新的图书的函数。

**1.2管理员添加书籍协作图：**



**2.1管理员处理书籍借阅顺序图：**



顺序图说明：

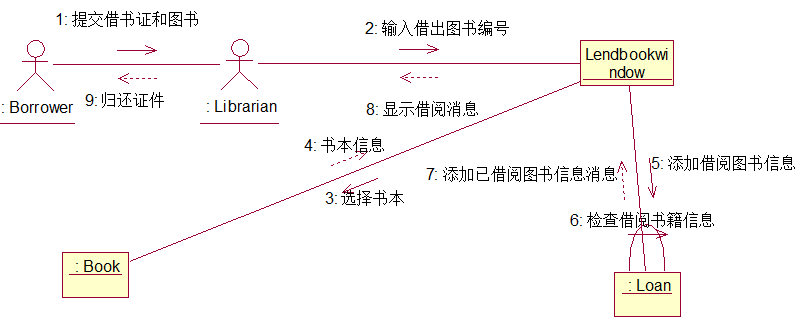
<1>查找书目：查找书目的函数。

<2>查找书籍：查找所要书籍的函数。

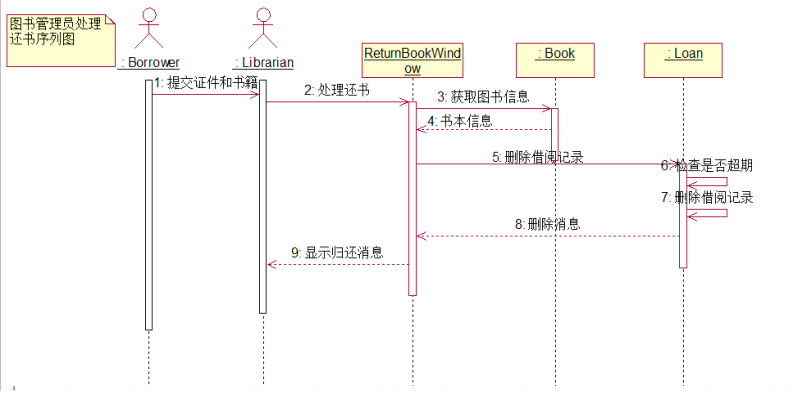
<3>输入借出图书编号：根据书的ID找到书籍信息的函数。

<4>检查借阅者信息：验证借阅者身份的函数。

**2.2管理员处理书籍借阅协作图：**



**3.1管理员处理书籍归还顺序图：**



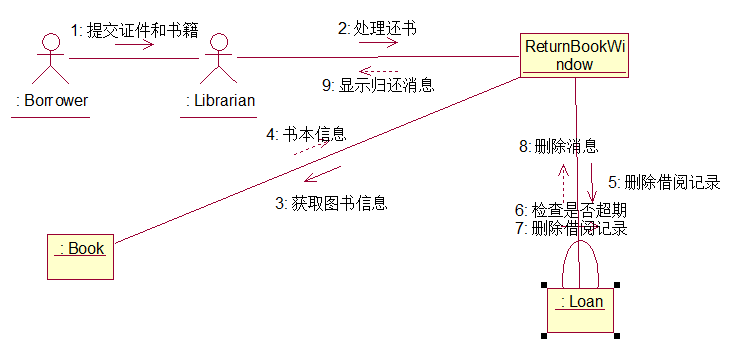
顺序图说明：

<1>检查书籍：检查书籍状况的函数。

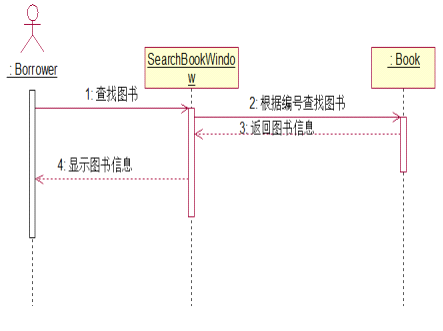
<2>获取图书信息：更新书籍类信息的函数。

<3>删除借阅记录：更新借阅者借阅信息的函数。

**3.2管理员处理书籍归还协作图：**



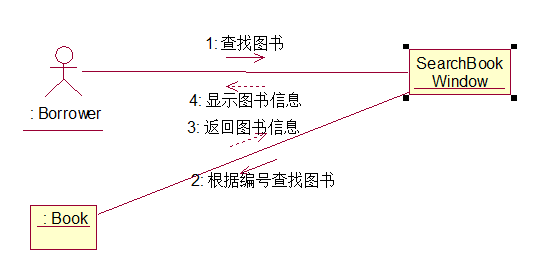
**4.1借阅者查询书籍信息顺序图：**



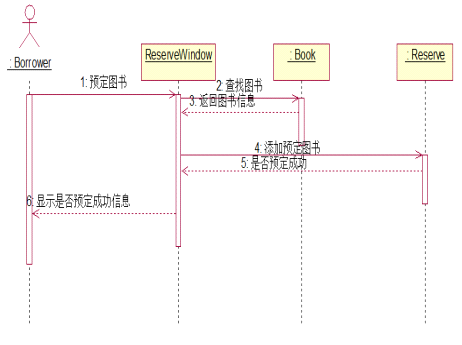
顺序图说明

查找图书：根据书目查找到相应的书籍信息。

**4.2借阅者查询书籍信息协作图：**



**5.1借阅者预约书籍时顺序图：**



顺序图说明

<1>登陆:登录系统的函数。

<2>查找书目：查找相应的书目的函数。

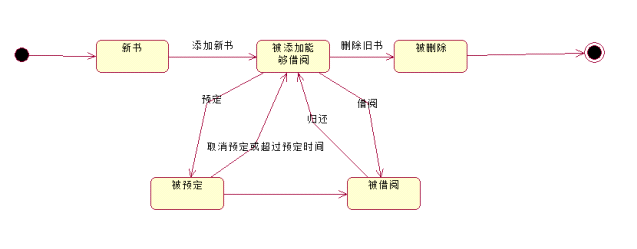
<3>预定图书：预约书籍的函数。

**5.2借阅者预约书籍协作图：**

## 

**四、状态图：**

**图书状态图：**



图书的状态图说明：

<1>当被预约的图书超过预约的期限或被借阅者取消预约时，转换为能被借阅的图书状态。

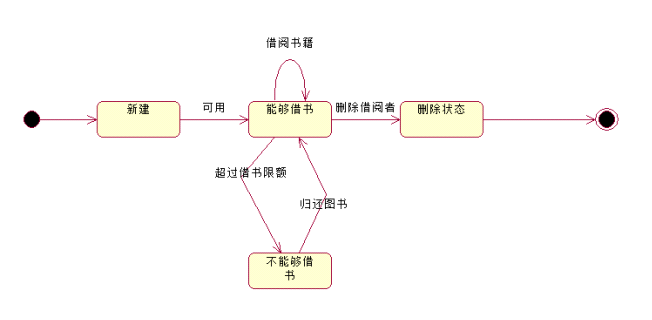
<2>被预约的图书可以被预约的借阅者借阅。

<3>图书被借阅后转换为被借阅状态。

<4>图书归还后转换为能被借阅状态。

<5>图书删除后转换为被删除状态。

**读者的状态图：**



借阅者的状态图说明：

<1>读者能够借阅一定数目的图书。

<2>当读者借阅图书的数目超过一定的限额时，不能够借阅图书。

<3>当读者处于不能够借阅图书时，读者归还图书，转换为能够借阅的状态。

<4>读者能够被管理员删除。

1. 以开发图书管理系统和程序设计为背景，论述面向对象的思想在解决复杂性问题方面的特点。并谈论自己对面向对象思想的认识。也可以从系统分析和设计的角度，讨论“现代图书管理系统“的设计、开发方法及方案。

**面向对象的思想在解决复杂性问题方面的特点：**

1. **在系统建模方面 :**

传统方法把软件开发看作为一种顺序的线性的过程，有的资料把这种过程建立的模型称之为瀑布式模型,瀑布式模型强调开发过程每一步 骤的完整性和独立性,后一步工作必须在前一步结束的基础上才能进行，这并不符合现实世界。而在OOM（面向对象模型）中由于对象的概念贯穿于整个开发过程之中,且无论什么样的具体的OOM法,总是采用喷泉式模型的，而喷泉式模型是比较符合现实世界的。

1. **在防止内外环境因素波动而造成的复杂性方面：**

一个软件的开发过程中及在投入使用之后,内外环境因素总是在不 停地波动之中,因而造成了问题的复杂性,这使许多软件工作者大伤脑筋, 如果采用OOM后则可能在下列诸点上减缓这种波动造成的影响:

1. 传统的软件开发方法强调从“系统”功能的分解为出发点,而 “系统”恰好是最不稳定的因素,就从我国企业而论,把企业当作一个系统,这个系统不是经常处于变动之中吗? OOM则不然,它从识别“对象” 和“对象的功能”为出发点,相比起整个系统来说,“对象”要稳定得 多,即便个别对象发生变化,对全局造成的影响要小得多。因而OOM事实上是在经常变动的“系统”之中去寻找那些比较稳定的因素。
2. 传统的软件开发方法在系统分析阶段总是试图确定问 题领域的“答案”,殊不知这种过早的确定反而对今后的变动无所适从,而00M则强调确定“问题领域”,强调系统分析是“学习过程”和迭代过程”,因而对今后系统的变动保持相当灵活的回旋空间。
3. 传统的软件开发方法在软件开发的前、中期均不考虑数据结构的问题,而将这一问题放置于后期去考虑,结果使得 围绕着数据而衍生的结构、功能等因素一直处于游离状态。 而OOM在定义对象时即考虑数据结构及方法,然后把它们封装于对象之中,再借助于类结构和继承性来衍生到其它类或对象之中,这就抓住了数据这一核心问题,既简化了问题又保证 了数据的一致性。从这一意义上来说,某些学者认为OOM实际上是一种数据驱动的系统分析和设计方法。
4. 传统的软件开发方法建立起来的系统模型和实际世界总是相差较大,而OOM自始至终以对象描述系统,对象就是实际世界中的实体,一一对 应,符合人们认识世界的思维过程, 而且系统开发的各个阶段间过渡得要平稳得多,因为“对象” 贯穿开发过程的始终。这同时又给今后的维护带来莫大的方便,即使系统作了较大的变动,为了完成新的功能,只需将对象和类重新拼装和组合。
5. 传统的软件开发方法总是强调系统的逻辑设计和物理 设计截然分开,逻辑设计解决系统要干什么的问题,而物理 设计才解决如何干的问题,这等于将有关系统的物理结构的决策推迟决定,因而有些资料称之为“延迟提交”。但一些场合下这种延迟决定往往事与愿违,因为不考虑具体硬件结构 的概念模型只能是开发人员的一种想象产物,一和具体的硬 件结构相结合便会产生许多问题,如:没有相合适的硬件结构 可以匹配概念模型,或即使有,但硬件成本太高,超过了用户可以支付的能力,因而开发人员不得不一再修改概念模型, 这在一定程度上增加了软件开发的复杂性。而OOM强调系统的迭代过程,并不将逻辑设计和物理设计分开,在决定对象 的概念模型时就开始考虑必要的物理因素,即是说并不总是实行“延迟决定”的策略,要看具体情况而定。这样做的好处 在于系统开发的每一 阶段都考虑项目实施的可能性,并不纯 粹追求一种理想的概念模型,而是在理想和现实之间取得较好的平衡。
6. **在辅助软件开发项目的管理方面：**

软件项目参与的人多,历时较长,整个开发过程人员往往变动,缺乏交流。传统方法在系统开发的每个阶段各有一套文档资料,开发人员之 间不便于交流,又由于信息模型不很贴近现实世界,也不便于软件开发 人员和用户之间的交流。而00M围绕着对象进行系统开发,既便于软件人 员之间的交流又便于软件人员和用户之间的交流。这在一定程度上有助于对软件项目自身的管理。

**自己对面向对象思想的认识：**

1：对象

关于对象的理解其实非常简单。在我们的身边，每一种事物都是一种对象，统筹一句话概括就是，对象是事物存在的实体。例如：电视、汽车、书桌、楼房包括我们人类自身都是一种对象。对象的话又划分为两种，即静态和动态两部分。静态：顾名思义，指的就是不能动的部分。就拿我们自身来说，每个人的你年龄、性别、身高、体重等都属于静态。而这些特征都可概括为属性，所有的对象都具备属性。动态：即具备这些属性的人可能做出的行为和动作（微笑、哭泣、行走、奔跑等）。人类通过探讨对象的属性和观察对象的行为来了解对象。

2：类

简单概括，类是封装对象属性和行为的载体，反过来说具有相同属性和行为的一类实体被成为类。举一个简单的例子，鸟类。鸟类封装了所有鸟的具体属性（嘴、翅膀、爪子）和具有的行为（飞行、捕食）。说到对象，不得不重点提一下对象的三个基本特征： 封装、继承、多态。

3：封装

定义：隐藏对象的属性和实现细节，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。

封装的目的：增强安全性和简化编程，使用者不必了解具体的实现细节，而只是要通过外部接口，特定的访问权限来使用类的成员。

封装的基本要求：把所有的属性私有化，在开发的时候经常要对已经编写的类进行测试。

4：继承（extends）

目的：实现代码的复用。

介绍：当两个类具有相同的特征（属性）和行为（方法）时，可以将相同的部分抽取出来放到一个类中作为父类，其它两个类继承这个父类。继承后子类自动拥有了父类的属性和方法，但特别注意的是，父类的私有属性（private）和构造方法不能被继承。另外子类可以写自己特有的属性和方法，目的是实现功能的扩展，子类也可以复写父类的方法，即方法的重写。子类不能继承父类中访问权限private的成员变量和方法。子类可以重写父类的方法，命名与父类同名的成员变量。有的时候我们需要这样的需求：我们需要将某些事物尽可能地对这个世界隐藏起来，但是允许子类的成员来访问他们这个时候就需要使用到protected。

用法：1、找出具有共同属性和行为的对象。

2、设计代表共同状态和行为的类。（父类）

3、决定子类是是否需要让某项行为具有特定不同的运作方式。

4、通过寻找使用共同行为的子类找出更多抽象化的机会。

5、完成类的继承层次化。

5：多态

概念：同一消息可以根据发送对象的不同而采用多种不同的行为方式，根据其实际类型调用其相应的方法。

实现方式：

（1）基于继承实现的多态

基于继承的实现机制主要表现在父类和继承该父类的一个或多个子类对某些方法的重写，多个子类对同一方法的重写可以表现出不同的行为。

（2）基于借口实现的多态

继承是通过重写父类的同一方法的几个不同子类来体现的，那么就是通过实现接口，覆盖接口中同一方法中不同的类来体现的。在接口的多态中，指向接口的引用必须是指定实现了该接口的一个类的实例程序。在运行时，根据对象引用的实际类型来执行对应的方法。继承都是单继承，只能为一组相关的类提供一直的服务接口，但是接口可以是多继承多实现，它能够利用一组相关或者不相关的接口进行组合与扩充，能够对外提供一致的服务接口。所以它相对于继承来说有更好的灵活性。

好处：程序的可扩展性及可维护性增强。

总结：

1.封装是为了对物与外界交流进行抽象。

2.继承是为了对物的信息进行抽象。

3.多态是为了对物的行为进行抽象。

6：最后，再补充一下关于抽象和接口一些相关信息。

抽象（abstract）

介绍：在面向对象的概念中，我们知道所有的对象都是通过类来描绘的，但是并不是所有的类都是用来描绘对象的，如果一个类中没有包含足够的信息来描绘一个具体的对象，这样的类就是抽象类。抽象类往往用来表示我们在对问题领域进行分析、设计中得出的抽象概念，是对一系列看上去不同，但是本质上相同的具体概念的抽象，我们不能把他们实例化（拿不出一个具体的东西），所以称之为抽象。

关于抽象方法：

1、抽象类中可以有抽象方法，也可以有非抽象方法。

2、抽象类中可以全都是非抽象方法。

3、一个抽象类中如果没有抽象方法，那么这个类不能实例化，反之

如果有抽象方法，那么这个类一定是抽象类。

接口（interface）

概念：把原有的事物与功能增强，在不改变类方法的前提下，进一步进行功能扩展。

接口归纳：

1、接口中只能定义抽象方法，不能有方法体，一定是public修饰。（接口中的方法都是抽象的）

2、接口中可以定义变量。

3、接口中不能定义静态方法。

抽象与接口的联系

1、一个接口可以被多个类实现，一个类也可以实现多个接口。

2、接口中的方法都是抽象方法，并且抽象的方法默认都是public abstract修饰的。不能用其他的修饰符修饰，可以不写。

3、接口中没有构造方法。

4、接口不是类。尤其不能使用new来实例化一个接口。但是可以声明接口的变量，这个变量可以指向实现了此接口的子类。

总而言之，抽象类是一种功能不全的类，接口只是一个抽象方法声明和静态不能被修改的数据的集合，两者都不能被实例化。从某种意义上说，接口是一种特殊形式的抽象类。