通过文件检索，结合学校教学信息管理系统，谈一谈你进行系统设计时打算采用的主要设计模型，并对其进行比较、评价、验证，获得有效结论，形成需求报告

要求： 1.表述三个以上的设计模型，比较优缺点

2.结合学校教学管理系统，谈一谈你是如何从三个以上模型中选定你的设计模型的，表述比较、评价、验证你的模型选择过程

3.描述学校教学管理系统的需求报告

**基于代码层次设计的四种设计模式：**

外观模式：

外观模式（Facade Pattern）隐藏系统的复杂性，并向客户端提供了一个客户端可以访问系统的接口。这种类型的设计模式属于结构型模式，它向现有的系统添加一个接口，来隐藏系统的复杂性。  
这种模式涉及到一个单一的类，该类提供了客户端请求的简化方法和对现有系统类方法的委托调用。  
介绍：  
意图：为子系统中的一组接口提供一个一致的界面，外观模式定义了一个高层接口，这个接口使得这一子系统更加容易使用。  
主要解决：降低访问复杂系统的内部子系统时的复杂度，简化客户端与之的接口。我们经常使用的三层架构就可以理解为一个外观模式的实例，如果没有service层，我们的controller层就可能需要直接去关联多个dao才能实现功能，这样就会出现多个controller和dao交叉关联，使系统越来越复杂，这时我们就可以抽象出一层service和dao接口使得耦合程度降低，除此之外，在维护一个遗留的大型系统时，我们也可以抽象出一个接口，用这个接口去和这个旧系统交互，新系统只需要与外观类交互，这样可以使得代码维护的复杂程度大大降低。

工厂方法模式：

工厂方法模式是简单工厂的进一步抽象和推广，由于工厂方法模式具有多态性，工厂模式方法既保持了简单工厂的优点，同时又克服了它的缺点，简单工厂模式中的工厂类负责创建实例，那么如果有新的机型添加，就要修改工厂类，这样也不利于进一步松耦合。工厂方法模式中核心工厂类不负责所有实例的创建，而是将具体的创建工作交给了子类去完成，这个核心类成为了一个抽象工厂角色。

工厂方法模式设计到一下四个角色的概念：

1. 抽象工厂角色

工厂方法模式的核心，创建产品的类必须实现该接口

2. 具体工厂角色

该角色实现了抽象工厂接口，具体如何创建产品就是在该类中实现

3. 抽象产品角色

所有产品的超类，负责实现产品共性的抽象定义

4. 具体产品角色

该角色实现了抽象产品接口，负责具体不同产品的业务逻辑

承接上面的工作创建抽象工厂角色

桥梁模式

桥梁模式的用意是“将抽象化(Abstraction)与实现化(Implementation)脱耦，使得二者可以独立地变化”。这句话有三个关键词，也就是抽象化、实现化和脱耦。理解这三个词所代表的概念是理解桥梁模式用意的关键。

　　抽象化

　　从众多的事物中抽取出共同的、本质性的特征，而舍弃其非本质的特征，就是抽象化。例如苹果、香蕉、生梨、 桃子等，它们共同的特性就是水果。得出水果概念的过程，就是一个抽象化的过程。要抽象，就必须进行比较，没有比较就无法找到在本质上共同的部分。共同特征是指那些能把一类事物与他类事物区分开来的特征，这些具有区分作用的特征又称本质特征。因此抽取事物的共同特征就是抽取事物的本质特征，舍弃非本质的特征。 所以抽象化的过程也是一个裁剪的过程。在抽象时，同与不同，决定于从什么角度上来抽象。抽象的角度取决于分析问题的目的。

　　通常情况下，一组对象如果具有相同的特征，那么它们就可以通过一个共同的类来描述。如果一些类具有相同的特征，往往可以通过一个共同的抽象类来描述。

　　实现化

　　抽象化给出的具体实现，就是实现化。

　　一个类的实例就是这个类的实例化，一个具体子类是它的抽象超类的实例化。

　　脱耦

　　所谓耦合，就是两个实体的行为的某种强关联。而将它们的强关联去掉，就是耦合的解脱，或称脱耦。在这里，脱耦是指将抽象化和实现化之间的耦合解脱开，或者说是将它们之间的强关联改换成弱关联。

　　所谓强关联，就是在编译时期已经确定的，无法在运行时期动态改变的关联；所谓弱关联，就是可以动态地确定并且可以在运行时期动态地改变的关联。显然，在Java语言中，继承关系是强关联，而聚合关系是弱关联。

将两个角色之间的继承关系改为聚合关系，就是将它们之间的强关联改换成为弱关联。因此，桥梁模式中的所谓脱耦，就是指在一个软件系统的抽象化和实现化之间使用聚合关系而不是继承关系，从而使两者可以相对独立地变化。这就是桥梁模式的用意。

合成模式：

　　合成模式把部分和整体的关系用树结构表示出来。合成模式使得客户端把一个个单独的成分对象和由它们复合而成的合成对象同等看待。一个文件系统就是一个典型的合成模式系统。文件系统是一个树结构，树上长有节点。树的节点有两种，一种是树枝节点，即目录，有内部树结构，在图中涂有颜色；另一种是文件，即树叶节点，没有内部树结构。可以把目录和文件当做同一种对象同等对待和处理，这也就是合成模式的应用。

　　合成模式可以不提供父对象的管理方法，但是合成模式必须在合适的地方提供子对象的管理方法，诸如：add()、remove()、以及getChild()等。

　合成模式的实现根据所实现接口的区别分为两种形式，分别称为安全式和透明式。

安全式合成模式的结构

　安全模式的合成模式要求管理聚集的方法只出现在树枝构件类中，而不出现在树叶构件类中。

这种形式涉及到三个角色：

　抽象构件(Component)角色：这是一个抽象角色，它给参加组合的对象定义出公共的接口及其默认行为，可以用来管理所有的子对象。合成对象通常把它所包含的子对象当做类型为Component的对象。在安全式的合成模式里，构件角色并不定义出管理子对象的方法，这一定义由树枝构件对象给出。

树叶构件(Leaf)角色：树叶对象是没有下级子对象的对象，定义出参加组合的原始对象的行为。

　树枝构件(Composite)角色：代表参加组合的有下级子对象的对象。树枝构件类给出所有的管理子对象的方法，如add()、remove()以及getChild()。

　 树枝构件类(Composite)给出了addChild()、removeChild()以及getChild()等方法的声明和实现，而树叶构件类则没有给出这些方法的声明或实现。这样的做法是安全的做法，由于这个特点，客户端应用程序不可能错误地调用树叶构件的聚集方法，因为树叶构件没有这些方法，调用会导致编译错误。

安全式合成模式的缺点是不够透明，因为树叶类和树枝类将具有不同的接口。

透明式合成模式的结构：

　　与安全式的合成模式不同的是，透明式的合成模式要求所有的具体构件类，不论树枝构件还是树叶构件，均符合一个固定接口。

　 客户端无需再区分操作的是树枝对象(Composite)还是树叶对象(Leaf)了；对于客户端而言，操作的都是Component对象。

两种实现方法的选择

　　这里所说的安全性合成模式是指：从客户端使用合成模式上看是否更安全，如果是安全的，那么就不会有发生误操作的可能，能访问的方法都是被支持的。

　　这里所说的透明性合成模式是指：从客户端使用合成模式上，是否需要区分到底是“树枝对象”还是“树叶对象”。如果是透明的，那就不用区分，对于客户而言，都是Compoent对象，具体的类型对于客户端而言是透明的，是无须关心的。

　　对于合成模式而言，在安全性和透明性上，会更看重透明性，毕竟合成模式的目的是：让客户端不再区分操作的是树枝对象还是树叶对象，而是以一个统一的方式来操作。

　　而且对于安全性的实现，需要区分是树枝对象还是树叶对象。有时候，需要将对象进行类型转换，却发现类型信息丢失了，只好强行转换，这种类型转换必然是不够安全的。

　　因此在使用合成模式的时候，建议多采用透明性的实现方式。

描述学校教学管理系统的需求报告：

教学管理系统的总目标是:在计算机网络,数据库和先进的开发平台上,利用现有的软件，配置一定的硬件，开发-个具有开放体系结构的、易扩充的、易维护的、性能好的、安全性高的、具有良好人机交互界面的教学管理系统，系统界面应直观易懂，用户不需要专门培训即可使用。只有登陆用户可以使用本系统。系统根据用户的角色提供不同的功能。

根据可行性研究的结果和客户的要求，分析现有情况及问题，采用Web服务器结构,将教学管理系统划分为两个子系统: Web 端子系统,服务器端子系统。

要求学生可以查看作业布置和完成情况、上传作业、查看课件、查看作业成绩；老师可以批改作业、发布课件、答疑；管理员可以管理

教学管理系统的性能要求

系统处理的准确性和及时性、系统的开放性和系统的可扩充性、系统的易用性和易维护性、系统的标准性、系统的响应速度