四种设计模式分别为：创建型模式、结构型模式、行为型模式、并发型模式。

1. 创建型模式
2. 举例

抽象工厂模式：比如一个抽象工厂类叫做DocumentCreator（文档创建器），此类提供创建若干种产品的接口，包括createLetter()（创建信件）和createResume()（创建简历）。其中，createLetter()返回一个Letter（信件），createResume()返回一个Resume（简历）。系统中还有一些DocumentCreator的具体实现类，包括FancyDocumentCreator和ModernDocumentCreator。这两个类对DocumentCreator的两个方法分别有不同的实现，用来创建不同的“信件”和“简历”（用FancyDocumentCreator的实例可以创建FancyLetter和FancyResume，用ModernDocumentCreator的实例可以创建ModernLetter和ModernResume）。这些具体的“信件”和“简历”类均继承自抽象类，即Letter和Resume类。客户端需要创建“信件”或“简历”时，先要得到一个合适的DocumentCreator实例，然后调用它的方法。一个工厂中创建的每个对象都是同一个主题的（“fancy”或者“modern”）。客户端程序只需要知道得到的对象是“信件”或者“简历”，而不需要知道具体的主题，因此客户端程序从抽象工厂DocumentCreator中得到了Letter或Resume类的引用，而不是具体类的对象引用。

1. 特点
   1. 优点

I.具体产品从客户代码中被分离出来

II.容易改变产品的系列

III.将一个系列的产品族统一到一起创建

* 1. 缺点

I.在产品族中扩展新的产品是很困难的，它需要修改抽象工厂的接口

1. 结构型模式
2. 举例

适配器模式：配器模式（Adapter）的定义如下：将一个类的接口转换成客户希望的另外一个接口，使得原本由于接口不兼容而不能一起工作的那些类能一起工作。适配器模式分为类结构型模式和对象结构型模式两种，前者类之间的耦合度比后者高，且要求程序员了解现有组件库中的相关组件的内部结构，所以应用相对较少些。

1. 特点
   1. 优点

I.客户端通过适配器可以透明地调用目标接口。

II.复用了现存的类，程序员不需要修改原有代码而重用现有的适配者类。

III.将目标类和适配者类解耦，解决了目标类和适配者类接口不一致的问题。

* 1. 缺点

I.对类适配器来说，更换适配器的实现过程比较复杂。

1. 行为型模式
2. 举例

责任链模式：责任链模式在面向对象程式设计里是一种软件设计模式，它包含了一些命令对象和一系列的处理对象。每一个处理对象决定它能处理哪些命令对象，它也知道如何将它不能处理的命令对象传递给该链中的下一个处理对象。该模式还描述了往该处理链的末尾添加新的处理对象的方法。

1. 特点
   1. 优点

I.降低请求发送者和接收者之间的耦合度

II.简化了对象、使对象不清除链的结构

III.增加了对象指派职责的灵活度、增强了可扩展性

IV.将多个条件语句进行分散到各个具体处理类中，增加代码的可阅读性。使代码更加清晰，责任更明确

* 1. 缺点

I.降低程序的性能。每个请求都是从链头遍历到链尾，当链比较长的时候，性能会大幅下降。

II.不易于调试。由于该模式采用了类似递归的方式，调试的时候逻辑比较复杂。

1. 并发型模式
2. 举例

线程池模式：这个模式首先启动一定量的工作线程。当没有请求的时候，所有线程都会处于等待状态，一旦有需求送达，就会唤醒某个线程来处理，处理后就会等待新的需求。

1. 特点
   1. 优点

I.抵消线程创建的开销，提高响应性；

I.封装了工作者线程生命周期管理；

III.减少销毁线程的开销。