Ch-07 设计模式

设计模式：对被用来在特定场景下解决一般设计问题的类和相互通讯的对象的描述

设计模式

外观模式：为子系统中的一组对象提供统一接口，使系统更易于使用

桥接模式：将抽象部分与它的实现部分分离，使它们都可以独立地变化

单例模式：确保一个类只有一个实例

备忘录模式：捕获一个对象的内部状态并在该对象之外保存这个状态，需要时能将对象恢复到原先保存的状态

策略模式：定义一系列算法，并封装每个算法，并使它们可以互换；策略模式可以让算法独立于使用它的客户端

工厂方法模式：把实例化的操作单独放到一个类中，这个类就成为简单工厂类

抽象工厂模式：提供一个接口，用于创建一个对象家族

代理模式：给某对象提供一个代理以控制对该对象的访问，常用于扩展原实际对象的行为

迭代器模式：提供一个顺序访问聚合对象元素的方法，并且不暴露聚合对象的内部表示

访问者模式：将作用于某种数据结构中的各元素的操作分离出来封装成独立的类，使其在不改变数据结构的前提下可以添加作用于这些元素的新的操作，为数据结构中的每个元素提供多种访问方式

观察者模式：当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新

命令模式：将一个请求封装成一个对象，使发出请求的责任与执行请求的责任分隔开

模板方法模式：定义一个操作中的算法骨架，而将算法的一些步骤延迟到子类中，使得子类可以不改变该算法结构的情况下重定义该算法的某些特定步骤

空对象：使用什么都不干的空对象来代替 NULL

适配器模式：定义一个转换器，把一个类接口转换成另一个用户需要的接口

依赖注入模式：对象声明自己的依赖，而该依赖由外部注入的形式为其提供