设计原则：

1.找出应用中可能需要变化之处，把他们独立出来，不要和那些不需要变化的代码放在一起。

2.针对接口编程，而不是针对实现编程。

3.多用组合，少用继承。

4.为了交互设计之间的松耦合而努力。

5.类应该对扩展开放，对修改关闭。

6.要依赖抽象，不要依赖具体的类。（依赖倒置原则）

7.最少知识原则：只和你的密友谈话

策略模式：

定义了算法族，分别封装起来，让它们之间可以互相替换，此模式让算法的变化独立于使用算法的客户。

观察者模式：

定义了对象之间的一对多依赖，这样一来，当一个对象改变状态时，他的所有依赖者都会收到通知并自动更新。

装饰者模式：  
 动态地将责任附加到对象上。若要扩展功能，装饰者提供了比继承更有弹性的替代方案。

工厂模式：

定义了一个创建对象的接口，但由子类决定要实例化的类是哪一个，工厂方法让类把实例化推迟到子类。

抽象工厂模式：

提供一个接口，用于创建相关或依赖对象的家族，而不需要明确指定具体类。

单件模式：

确保一个类只有一个实例，并提供一个全局访问点。

命令模式：

将“请求”封装成对象。以便使用不同的请求、队列或者日志来参数化其他对象。命令模式也支持可撤销的操作。

适配器模式：

将一个类的接口，转换成客户期望的另一个接口。适配器让原本接口不兼容的类可以合作无间。

外观模式：

提供了一个统一的接口，用来访问子系统中的一群接口。外观定义了一个高层接口，让子系统更容易使用。

编程：

1. 良好的OO设计必须具备可复用，可扩充，可维护三个特性

2.