一.命令模式（Command Pattern）的定义

将请求封装成对象，以便使用不同的请求、日志、队列等来参数化其他对象。命令模式也支持撤销操作。

1. 命令模式的四个角色

1.调用者（Invoker）角色

该角色负责调用命令对象的执行请求。

2.命令（Command）角色

该角色声明一个給具体命令类的抽象接口，定义需要执行的命令。

3.具体命令（Concrte Command）角色

该角色定义一个接收者和行为之间的弱耦合，实现命令方法，并调用接收者的相应操作。

4.接收者（Receive）角色

该角色负责具体实施和执行一个请求。

三.命令模式的优缺点

命令模式中的优点

1.类间解耦

调用者和接收者之间没有任何依赖，调用者实现功能只需要调用命令中的执行方法即可，不需要了解是哪个接收者来执行。

2.可扩展性

命令的子类非常容易扩展，而调用者和高层的模块不产生严重的代码耦合。

3.命令模式结合其他模式会跟优秀。

命令模式可以结合责任链模式，实现灵敏解析任务；

结合模板方法模式，可以减少Command子类的膨胀问题。

命令模式的缺点

使用命令模式可能会导致系统中出现过多的具体命令类，因此在项目中使用命令模式时需要慎重考虑。

四.命令模式的使用场景

1.使用命令模式作为“回调”在面向对象系统中的代替。“回调”就是将一个方法登记，方便在以后调用此方法。

2.需要在不同的时间指定请求、将请求排队。

3.系统需要支持命令的撤销（undo方法）。命令对象可以把状态存储起来，等到客户需要撤销时，可以调用undo（）方法，将命令所产生的效果撤销。

4.需要将系统中所有的数据更新操作保存到日志里，以便在系统崩溃时，可以根据日志读回所有的数据更新命令，重新调用execute（）方法一条一条执行这些命令，从而回复系统在崩溃前所做的数据更新。

5.一个系统需要支持交易。一个交易结构封装了一组数据更新命令。使用命令模式来实现交易结构可以使系统增加新的交易类型。