概述

将来自客户端的请求传入一个对象，从而使你可用不同的请求对客户进行参数化。用于“行为请求者”与“行为实现者”解耦，可实现二者之间与不变的因素

角色

- Command 定义命令的接口，声明执行的方法。
- ConcreteCommand 命令接口实现对象，是“虚”的实现；通常会持有接收者，并调用接收者的功能来完成命令要执行的操作。
- Receiver 接收者，真正执行命令的对象。任何类都可能成为一个接收者，只要它能够实现命令要求实现的相应功能。
- Invoker 要求命令对象执行请求，通常会持有命令对象，可以持有很多的命令对象。这个是客户端真正触发命令并要求命令执行相应接对象的入口。
- Client 创建具体的命令对象，并且设置命令对象的接收者。注意这个不是我们常规意义上的客户端，而是在组装命令对象和接收者，理解，因为真正使用命令的客户端是从Invoker来触发执行

代码

命令接收者Receiver： Tv

```
1 public class Tv {
2     public int currentChannel = 0;
3 
4     public void turnOn() {
```

```
        System.out.println("The televisino is on.");
    }

    public void turnOff() {
        System.out.println("The television is off.");
     }

     public void changeChannel(int channel) {
         this.currentChannel = channel;
         System.out.println("Now TV channel is " + channel);
    }
}
```

命令的接口

```
 public interface Command {
     void execute();
}
```

具体命令：开机

```
public class CommandOn implements Command {
    private Tv tv;

    public CommandOn(Tv tv) {
        this.tv = tv;
    }

    public void execute() {
        tv.turnOn();
     }
}
```

具体命令：关机

```
public class CommandOff implements Command {
    private Tv tv;

    public CommandOff(Tv tv) {
        this.tv = tv;
    }

    public void execute() {
        tv.turnOff();
     }
}
```

具体命令：切换频道

```
public class CommandChange implements Command {

     private Tv tv;

     private int channel;

```

```
    public CommandChange(Tv tv) {
            this.tv = tv;
    }

    public void setChannel(int channel){
            this.channel = channel;
    }

    public void execute() {
            tv.changeChannel(channel);
    }
}
```

命令发送者：遥控器

```
public class Control {

    private CommandOn onCommand;
    private CommandOff offCommand;
    private CommandChange changeChannel;

    public Control(CommandOn on, CommandOff off, CommandChange channel) {
            onCommand = on;
            offCommand = off;
            changeChannel = channel;
    }

    public void turnOn() {
            onCommand.execute();
    }

    public void turnOff() {
            offCommand.execute();
    }

    public void changeChannel(int channel) {
            changeChannel.setChannel(channel);
            changeChannel.execute();
    }
}
```

客户端

```
public class Main {

    public static void main(String[] args) {
            // 命令接收者Receiver
            Tv tv = new Tv();
            // 开机命令ConcreteCommond
            CommandOn on = new CommandOn(tv);
            // 关机命令ConcreteCommond
            CommandOff off = new CommandOff(tv);
            // 频道切换命令ConcreteCommond
```

```
            CommandChange channel = new CommandChange(tv);
            // 命令控制对象Invoker
            Control control = new Control(on, off, channel);

            // 开机
            control.turnOn();
            // 切换频道
            control.changeChannel(1);
            // 切换频道
            control.changeChannel(2);
            // 关机
            control.turnOff();
    }

}
```

总结

优点

1.降低对象之间的耦合度

2.新的命令可以很容易地加入到系统中

3.可以比较容易地设计一个组合命令

4.调用同一方法实现不同的功能

缺点

使用命令模式可能会导致某些系统有过多的具体命令类

命令模式与策略模式的区别——目标不同

策略模式是通过**不同的算法做同一件事情**：例如排序

而命令模式则是**通过不同的命令做不同的事情**