GOF 设计模式小白教程之命令模式

作者：valarchie
原文链接：https://ld246.com/article/1567184643200
来源网站：链滴

命令模式（Command）

定义：

将一个请求封装为一个对象，使发出请求的责任和执行请求的责任分割开。这样两者之间通过命令对进行沟通，这样方便将命令对象进行储存、传递、调用、增加与管理。

通俗解释：

假设司机是命令发出方，而车子是命令接收方。司机与车子通过命令对象来传达指令操作车子从而进解耦。司机并不知道最终命令执行的是谁，车子也不知道是谁在下达命令。

代码：

命令接口类，持有命令的执行者Car

```
public abstract class Command {

    protected Car car;

    public Command(Car car) {
        this.car = car;
    }

    public abstract void execute();
}
```

实际的命令实现类，启动命令和加速命令

```
public class StartCommand extends Command {
    public StartCommand(Car car) {
        super(car);
    }
    @Override
    public void execute() {
        car.start();
    }
}


public class RunCommand extends Command {
    public RunCommand(Car car) {
        super(car);
    }

    @Override
    public void execute() {
        car.run();
    }
}
```

命令的实际执行者，车辆类

```
public class Car {

    public void start() {
        System.out.println("启动车子！");
    }

    public void run() {
        System.out.println("全速行驶！");
    }
}
```

命令的请求者，司机类，司机只需要一个requestCommand的方法，就可以执行不同的命令了。

```
public class Driver {

    public void requestCommand(Command command) {
        command.execute();
    }

}
```

测试命令模式

```
public class TestCommand {

    public static void main(String[] args) {

        Car car = new Car();

        Command start = new StartCommand(car);
        Command run = new RunCommand(car);

        Driver driver = new Driver();

        driver.requestCommand(start);
        driver.requestCommand(run);

    }
}
```

运行结果

```
启动车子！
全速行驶！
```

解析：

1. 降低系统的耦合度。命令模式能将调用操作的对象与实现该操作的对象解耦。

2. 增加或删除命令非常方便。采用命令模式增加与删除命令不会影响其他类，它满足“开闭原则”，扩展比较灵活。

3. 可以实现宏命令。命令模式可以与组合模式结合，将多个命令装配成一个组合命令，即宏命令。

4. 方便实现 Undo 和 Redo 操作。命令模式可以与后面介绍的备忘录模式结合，实现命令的撤销与复。

5. 缺点是可能产生大量具体命令类。因为计对每一个具体操作都需要设计一个具体命令类，这将增加统的复杂性。