GOF 设计模式小白教程之责任链模式

作者：valarchie
原文链接：https://ld246.com/article/1567173750512
来源网站：链滴

责任链模式（Chain of Responsibility）

定义：

为了避免请求发送者与多个请求处理者耦合在一起，将所有请求的处理者通过前一对象记住其下一个象的引用而连成一条链；当有请求发生时，可将请求沿着这条链传递，直到有对象处理它为止。

通俗解释：

这个模式非常简单。就好比小明在公司请假。先把假条给组长审批，组长再把假条给经理审批，经理把假条给老板审批。

代码：

抽象责任类Leader，拥有审批的技能。

```
public abstract class Leader {

    protected Leader next;
    public Leader getNext() {
        return next;
    }
    public void setNext(Leader next) {
        this.next = next;
    }
    // 审批
    public abstract void approve();

}
```

组长类、经理类、老板类依次往后传递

```
public class GroupLeader extends Leader {
    @Override
    public void approve() {

        System.out.println("组长审批通过！");

        if (getNext() != null) {
            getNext().approve();
        }

    }
}

public class Manager extends Leader {

    @Override
    public void approve() {
        System.out.println("经理审批通过！");
```

```
        if (getNext() != null) {
            getNext().approve();
        }
    }

}

public class Boss extends Leader {
    @Override
    public void approve() {
        System.out.println("老板审批通过！");
        if (getNext() != null) {
            getNext().approve();
        }
    }
}
```

测试责任链模式

```
public class TestChainOfResponsibility {

    public static void main(String[] args) {

        Leader groupLeader = new GroupLeader();
        Leader manager = new Manager();
        Leader boss = new Boss();

        groupLeader.setNext(manager);
        manager.setNext(boss);
        // 审批
        groupLeader.approve();

    }

}
```

运行结果：

```
组长审批通过!
经理审批通过!
老板审批通过!
```

解析：

1. 降低了对象之间的耦合度。责任对象不需要知道关心下一个对象是谁，只管传递即可。

2. 增强了系统的可扩展性。可以根据需要增加新的请求处理类，满足开闭原则。

3. 增强了给对象指派职责的灵活性。当工作流程发生变化，可以动态地改变链内的成员或者调动它们次序，也可动态地新增或者删除责任。

4. 责任分担。每个类只需要处理自己该处理的工作，不该处理的传递给下一个对象完成，明确各类的任范围，符合类的单一职责原则。