



源码架构

```

1 package DecoratorPattern;
2
3 public abstract class Phone {
4     public abstract void getCall();
5 }
6
7 package DecoratorPattern;
8
9 public class SimplePhone extends Phone {
10
11     @Override
12     public void getCall() {
13         // TODO 自动生成的方法存根
14         System.out.println("手机来电了");
15     }
16 }

```

```

1 package DecoratorPattern;
2
3 public abstract class PhoneDecorator extends Phone {
4     protected Phone phone;
5
6     public PhoneDecorator(Phone p ) {
7         this.phone = p;
8     }
9
10    abstract public void getCall() ;
11 }
12

```

```

1 package DecoratorPattern;
2
3 public class JarPhone extends PhoneDecorator {
4
5     public JarPhone(Phone p ) {
6         super(p);
7     }
8     @Override
9     public void getCall() {
10        // TODO 自动生成的方法存根
11        this.phone.getCall();
12        this.jar();
13    }
14    public void jar() {
15        System.out.println("手机震动了");
16    }
17 }

```

```

1 package DecoratorPattern;
2
3 public class ComplexPhone extends PhoneDecorator {
4
5     public ComplexPhone(Phone p) {
6         super(p);
7         // TODO 自动生成的构造函数存根
8     }
9
10    @Override
11    public void getCall() {
12        // TODO 自动生成的方法存根
13        this.phone.getCall();
14        this.light();
15    }
16    public void light() {
17        System.out.println("手机来电发光了");
18    }
19 }
20

```

```

1 package DecoratorPattern;
2
3 public class test {
4
5     public static void main(String[] args) {
6         // TODO 自动生成的方法存根
7         //测试类
8         SimplePhone sp = new SimplePhone();
9         sp.getCall();
10        System.out.println("-----");
11        JarPhone jp= new JarPhone(sp); //第一次升级
12        jp.getCall();
13        System.out.println("-----");
14        ComplexPhone cp = new ComplexPhone(jp); //第二次升级
15        cp.getCall();
16    }
17
18 }

```

手机来电了

手机来电了

手机震动了

手机来电了

手机震动了

手机来电发光了