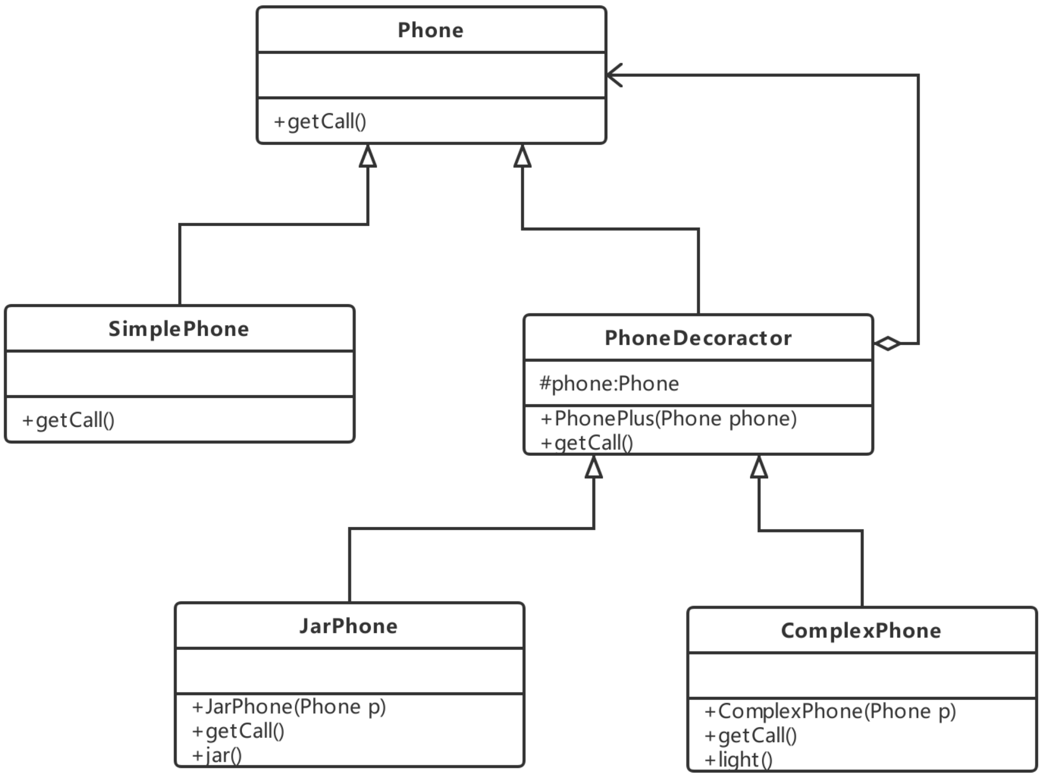
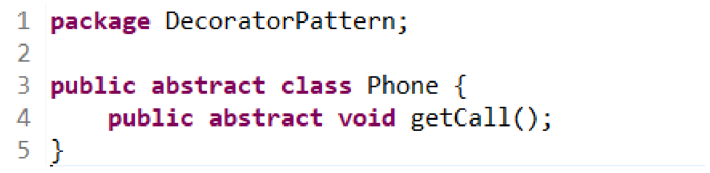
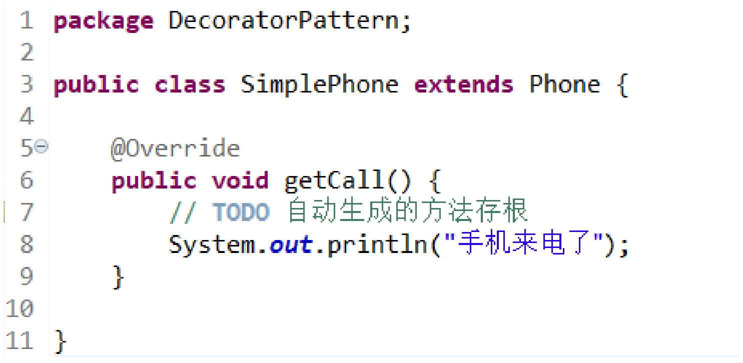


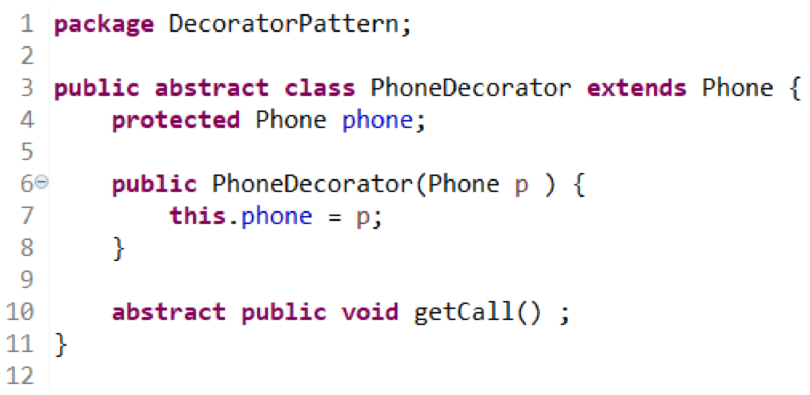
源码架构



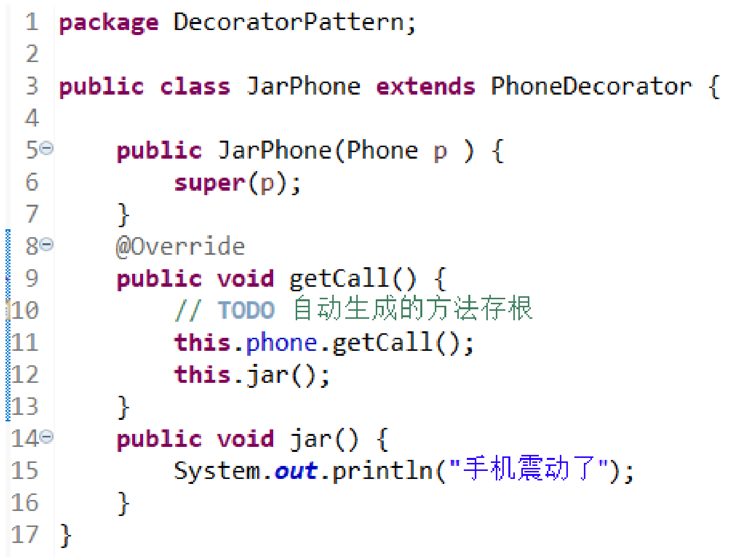
```java
package DecoratorPattern;

public abstract class Phone {
    public abstract void getCall();
}
```



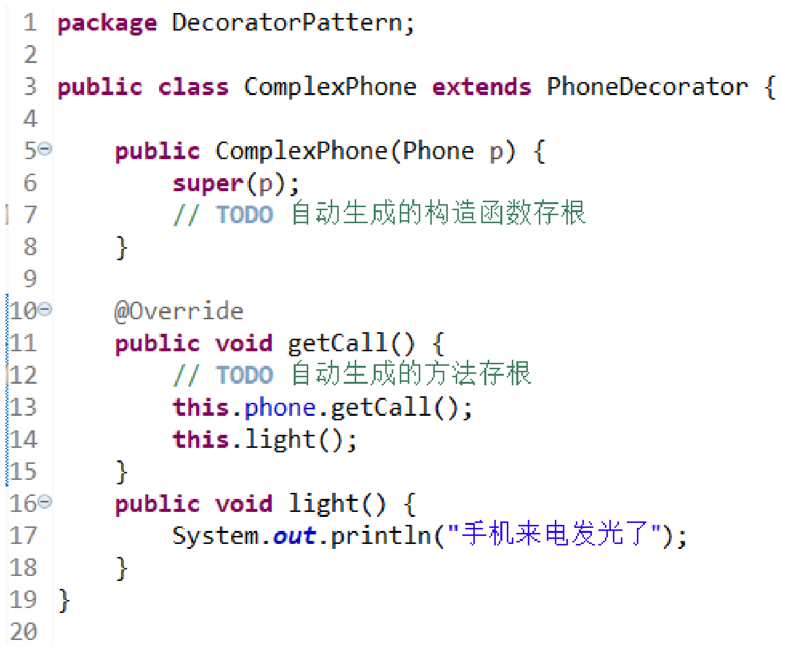
```java
package DecoratorPattern;

public class SimplePhone extends Phone {

    @Override
    public void getCall() {
        // TODO 自动生成的方法存根
        System.out.println("手机来电了");
    }

}
```



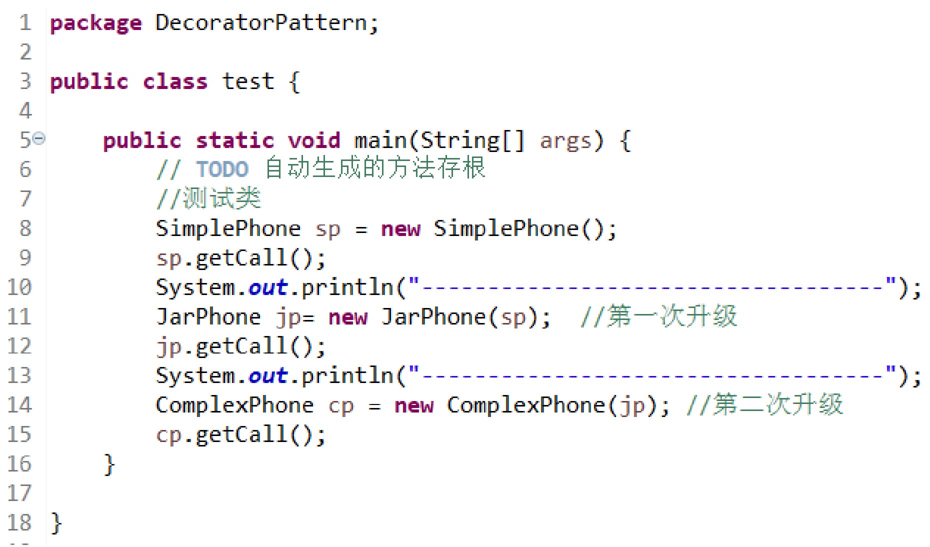
```java
package DecoratorPattern;

public abstract class PhoneDecorator extends Phone {
    protected Phone phone;

    public PhoneDecorator(Phone p ) {
        this.phone = p;
    }

    abstract public void getCall() ;
}

```



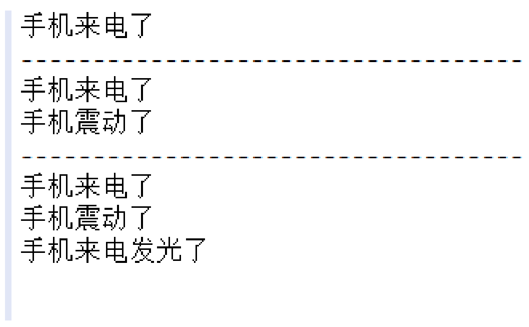
```java
package DecoratorPattern;

public class JarPhone extends PhoneDecorator {

    public JarPhone(Phone p ) {
        super(p);
    }
    @Override
    public void getCall() {
        // TODO 自动生成的方法存根
        this.phone.getCall();
        this.jar();
    }
    public void jar() {
        System.out.println("手机震动了");
    }
}
```



```java
package DecoratorPattern;

public class ComplexPhone extends PhoneDecorator {

    public ComplexPhone(Phone p) {
        super(p);
        // TODO 自动生成的构造函数存根
    }

    @Override
    public void getCall() {
        // TODO 自动生成的方法存根
        this.phone.getCall();
        this.light();
    }
    public void light() {
        System.out.println("手机来电发光了");
    }
}

```



```java
package DecoratorPattern;

public class test {

    public static void main(String[] args) {
        // TODO 自动生成的方法存根
        //测试类
        SimplePhone sp = new SimplePhone();
        sp.getCall();
        System.out.println("-----------------------------------");
        JarPhone jp= new JarPhone(sp);  //第一次升级
        jp.getCall();
        System.out.println("-----------------------------------");
        ComplexPhone cp = new ComplexPhone(jp); //第二次升级
        cp.getCall();
    }

}
```



```
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
```