链滴

《Head First 设计模式》：适配器模式

作者：jingqueyimu

原文链接：https://ld246.com/article/1597756388581

来源网站：链滴

正文

一、定义

适配器模式将一个类的接口（被适配者），转换成客户期望的另一个接口。适配器让原本接口不兼容类可以合作无间。

要点：

- 适配器实现了目标接口，并持有被适配者的实例。
- 适配器使用被适配者的方法，把请求转换成被适配者的一个或多个方法。
- 客户通过目标接口调用适配器的方法对适配器发出请求。
- 客户与被适配者是解耦的，一个不知道另一个。
- 当需要使用一个现有的类而其接口并不符合你的需要时，就使用适配器。

二、实现步骤

1、创建被适配者接口

```
/**
 * 被适配者接口
 */
public interface Adaptee {

    public void specificRequest();
}
```

2、创建具体的被适配者，并实现被适配者接口

```
/**
 * 具体的被适配者
 */
public class ConcreteAdaptee implements Adaptee {

    @Override
    public void specificRequest() {
        System.out.println("ConcreteAdaptee is doing something...");
    }
}
```

3、创建目标接口

```
/**
 * 目标接口
 */
public interface Target {

    public void request();
}
```

4、创建适配器类，并实现目标接口

适配器持有被适配者，并在收到请求时调用被适配者的方法。

```
/**
 * 适配器
 */
public class Adapter implements Target {

    /**
     * 被适配者
     */
    Adaptee adaptee;

    public Adapter(Adaptee adaptee) {
        this.adaptee = adaptee;
    }

    @Override
    public void request() {
        // 调用被适配者方法
        adaptee.specificRequest();
    }
}
```

5、使用适配器转换被适配者

```
public class Test {
```

```
    public static void main(String[] args) {
        // 被适配者
        Adaptee adaptee = new ConcreteAdaptee();
        // 适配器
        Adapter adapter = new Adapter(adaptee);
        adapter.request();
    }
}
```

三、举个栗子

1、背景

你上班的公司做了一套模拟鸭子游戏。现在，假设你缺鸭子对象，想用一些火鸡对象来冒充。

显而易见，因为火鸡的接口不同，所以我们不能公然拿来用。

2、实现

使用适配器把火鸡的方法转换成鸭子的方法，这样表面上看起来就像是一只鸭子了。

（1）创建火鸡接口

```
/**
 * 火鸡接口（被适配者）
 */
public interface Turkey {

    /**
     * 咯咯叫
     */
    public void gobble();

    /**
     * 飞行
     */
    public void fly();
}
```

（2）创建具体的火鸡

```
/**
 * 野火鸡
 */
public class WildTurkey implements Turkey {

    @Override
    public void gobble() {
        System.out.println("Gobble gobble");
    }
```

```
    @Override
    public void fly() {
        System.out.println("I'm flying a short distance");
    }
}
```

（3）创建鸭子接口

```
/**
 * 鸭子接口（目标接口）
 */
public interface Duck {

    /**
     * 呱呱叫
     */
    public void quark();

    /**
     * 飞行
     */
    public void fly();
}
```

（4）创建火鸡适配器，并实现鸭子接口

```
/**
 * 火鸡适配器
 */
public class TurkeyAdapter implements Duck {

    /**
     * 火鸡
     */
    Turkey turkey;

    public TurkeyAdapter(Turkey turkey) {
        this.turkey = turkey;
    }

    @Override
    public void quark() {
        turkey.gobble();
    }

    @Override
    public void fly() {
        // 火鸡飞行距离比鸭子短，因此需要调用5次火鸡的fly()方法。
        for (int i = 0; i < 5; i++) {
            turkey.fly();
        }
    }
}
```

（5）使用火鸡适配器，将火鸡适配成鸭子

```
public class Test {

    public static void main(String[] args) {
        // 火鸡（被适配者）
        Turkey turkey = new WildTurkey();
        // 火鸡适配器
        TurkeyAdapter turkeyAdapter = new TurkeyAdapter(turkey);
        turkeyAdapter.quark();
        turkeyAdapter.fly();
    }
}
```