# 适配器模式

## 引入

在软件开发过程中，可能会遇到这种情况：已有的类可以很好的满足客户的功能需求，但是它所提供的接口并不是客户所期望的（比如方法名不同），此时若想不改变已有的代码（包括已有类的代码和客户所提供的代码）来解决这个问题，就需要用到适配器模式，它和现实生活中的适配器扮演者同样的角色：将一个接口转换为另一个接口，以符合客户的期望。就如同电源适配器：可以将欧式插座经过转换变成美式插座，从而可以使美式插头可以使用该插座。当然，在转换过程中，电源适配器可能只是简单的改变插座的形状来匹配客户的插头，但是也有可能会经过复杂的变换以更好的适应新的需求。适配器模式也是如此，它的内部实现可能十分简单，也可能相当复杂，这要根据已有类和客户的需求来定。

## 定义

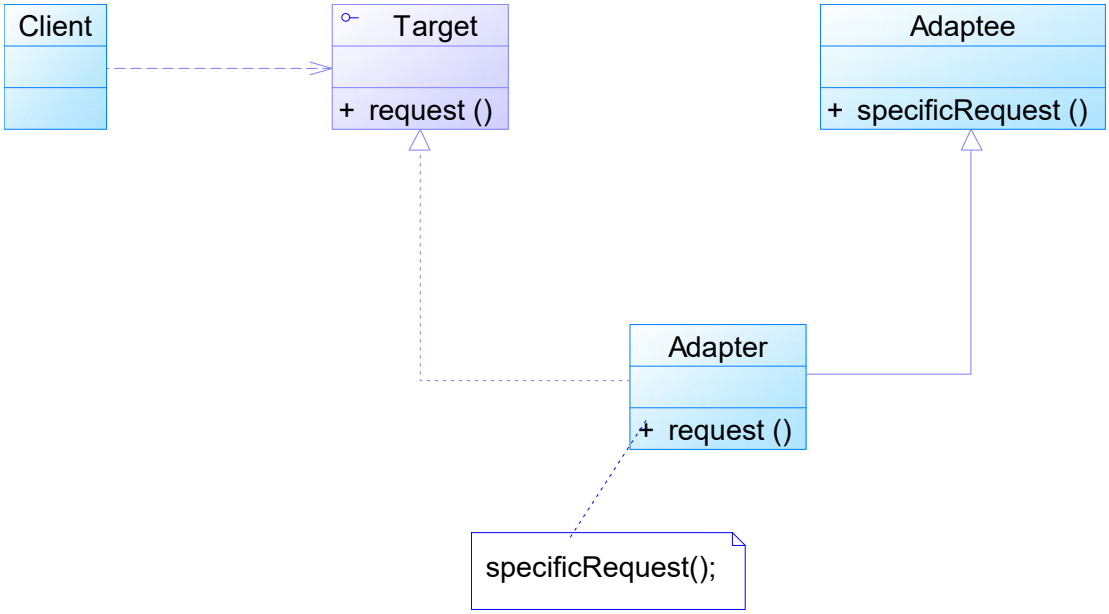
**将一个接口转换成客户期望的另一个接口。**适配器模式使接口不兼容的那些类可以一起工作。适配器模式既可以作为类结构型模式，也可以作为对象结构型模式。

## 使用方式

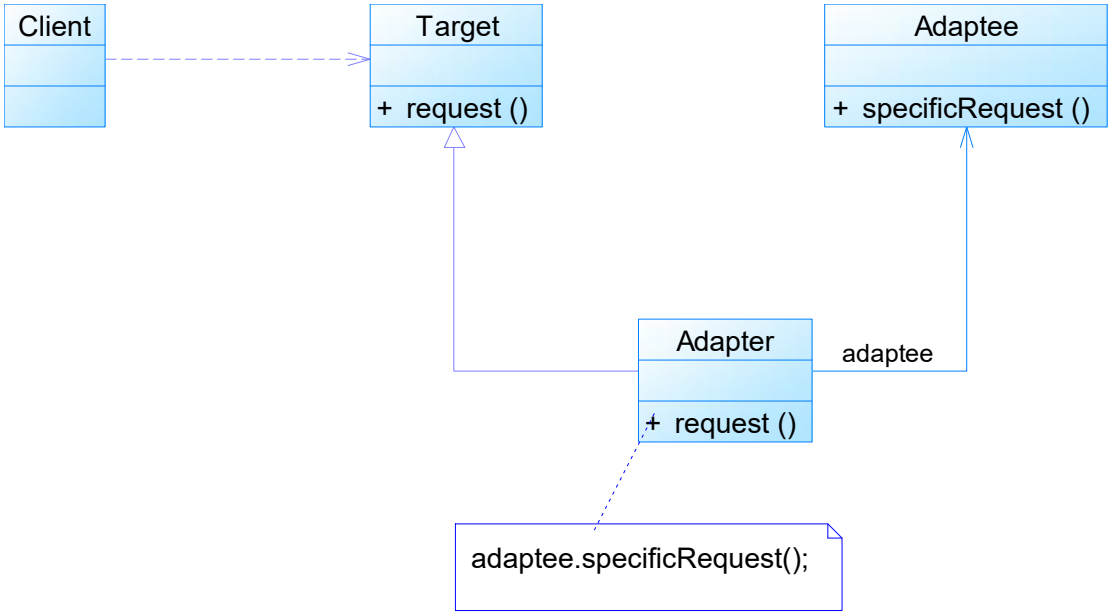
在适配器模式中可以定义一个包装类，包装不兼容接口的对象，这个包装类指的就是适配器(Adapter)，它所包装的对象就是适配者(Adaptee)，即被适配的类。适配器提供客户类需要的接口，适配器的实现就是把客户类的请求转化为对适配者的相应接口的调用。也就是说：当客户类调用适配器的方法时，在适配器类的内部将调用适配者类的方法，而这个过程对客户类是透明的，客户类并不直接访问适配者类。因此，适配器可以使由于接口不兼容而不能交互的类可以一起工作。

## 类图

类适配器（适配器类继承自适配者类）：




对象适配器（适配器类关联适配者类）：




# 优缺点

适配器模式的优点：

1）增加了类的透明性，将具体的实现封装在适配者类中，对于客户端类来说是透明的，将目标类和适配者类解耦。

2）增加了类的复用性，通过引入一个适配器类来重用现有的适配者类，而无须修改原有代码，提高了适配者的复用性。

3）灵活性和扩展性都非常好，可以很方便地更换适配器，也可以在不修改原有代码的基础上增加新的适配器类，完全符合“开闭原则”。

对于类适配器而言：

还具有如下优点：由于适配器类是适配者类的子类，因此可以在适配器类中置换一些适配者的方法，使得适配器的灵活性更强。

类适配器模式的缺点如下：对于 Java、C#等不支持多重继承的语言，一次最多只能适配一个适配者类，而且目标抽象类只能为抽象类，不能为具体类，其使用有一定的局限性，不能将一个适配者类和它的子类都适配到目标接口。

对于对象适配器而言：

还具有如下优点：一个对象适配器可以把多个不同的适配者适配到同一个目标，也就是说，同一个适配器可以把适配者类和它的子类都适配到目标接口。

对象适配器模式的缺点如下：与类适配器模式相比，要想置换适配者类的方法就不容易。如果一定要置换掉适配者类的一个或多个方法，就只好先做一个适配者类的子类，将适配者类的方法置换掉，然后再把适配者类的子类当做真正的适配者进行适配，实现过程较为复杂。