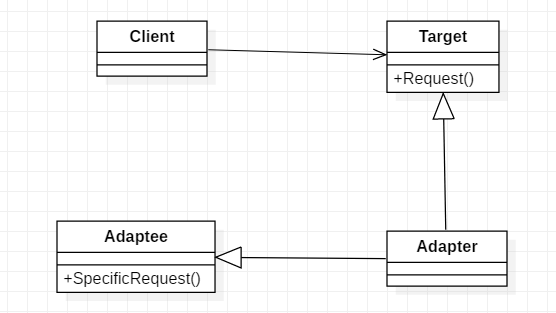
# 定义

假如有两个接口A、B，我们需要将这两个接口组合在一起，但是接口A、B是不相互兼容的，这时我们可以引入物体C，物体C既要适配A接口也要适配B接口，这样A和B通过C便可以组合在一起。

# **组成角色**

根据适配器类与适配者类的关系不同，适配器模式可分为对象适配器和类适配器两种，对象适配器中适配器与适配者之间是关联关系;类适配器模式中，适配器与适配者是继承(或实现)关系。



适配器模式包含的角色有：

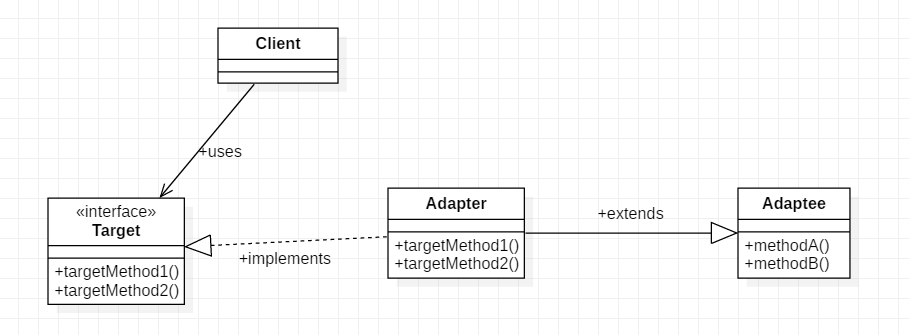
目标角色(Target)：该角色定义把其他类转换为我们期望的接口，可以实抽象类、具体类、接口。

适配器角色(Adapter)：适配器可以调用另一个接口作为一个转换器，对Adaptee和Target进行适配，适配器类通常是一个具体类。

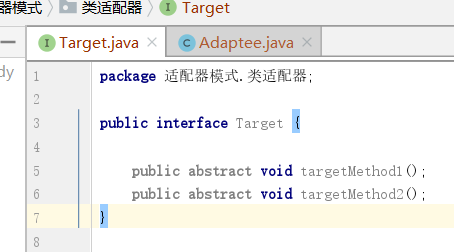
源角色(Adaptee)：你想把谁转换成目标角色，这个“谁”就是源角色。

请求者(Client): 负责使用Target定义的方法进行具体处理。

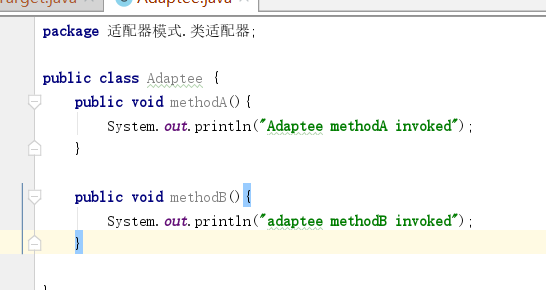
# **类适配器(使用继承)**



Target接口是我们要适配的目标接口



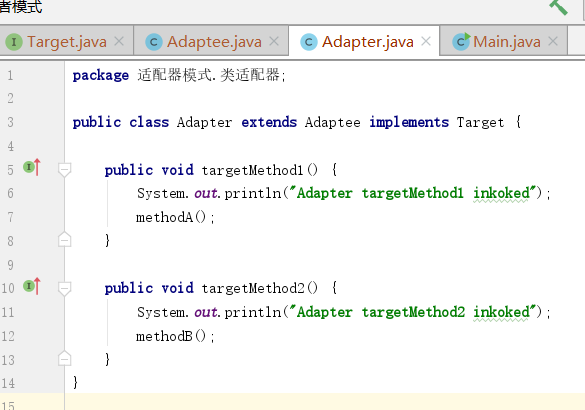
Adaptee是要被适配的接口



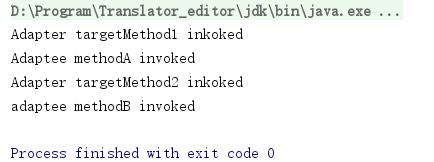
适配器



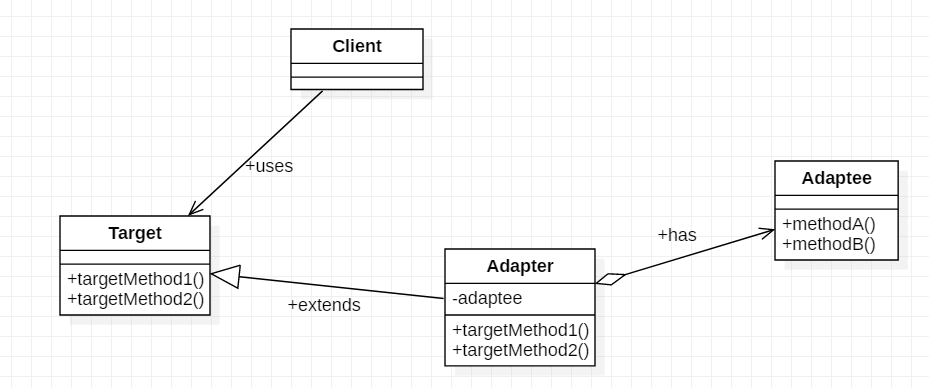
Client

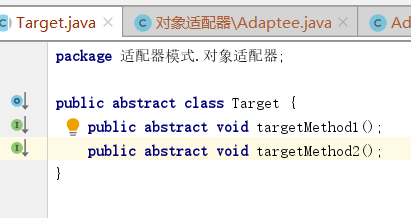


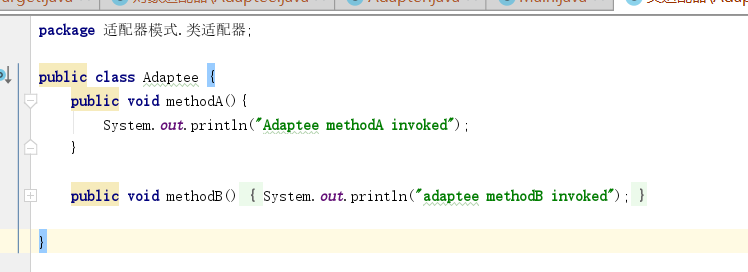
执行结果:



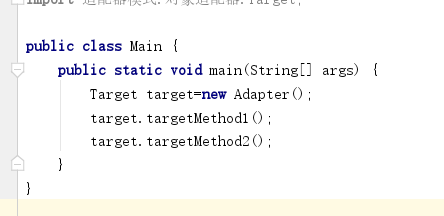
1. **对象适配器(使用委托)**











1. **优缺点**

优点：

\*降低适配接口的耦合性；

\*增加了类的透明性和复用性，将具体的业务实现过程封装在适配者类中，对于客户端类而 言是透明的，一个适配者类可以在多个不同的系统中复用；

\*可以将两个互不相干的类联系在一起；

\*增强系统灵活性；

缺点：

\*适配器对于java、c#等不支持多继承的语言，一次只能是配一个适配者类，不嫩恶搞同时适配多个适配者。

1. **适用场景**

适配器模式用于在两个对象或系统之间建立适配链接，使得前后系统衔接更加平滑，适配器模式的实现主要有继承方式的类适配器和委托方式的对象适配器。