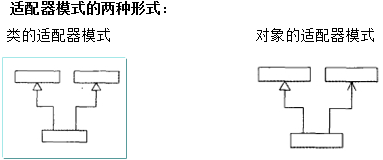
* 设计模式 – 适配器模式（Adapter）：

把一个类的接口变换成客户端所期待的另一种接口，从而使原本因接口不匹配而无法一起工作的两个类能够一起工作。



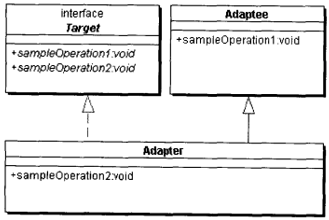
* 类的适配器模式：

使用一个具体类把源（Adaptee）适配到目标（Target）。

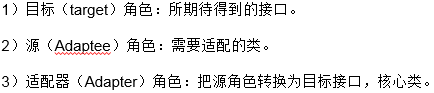
如果源（Adaptee）以及源的子类都使用此适配器类适配，就行不通了。

我们使用对象的适配器模式来解决。

类的适配器模式把适配的类的API转换为目标的API，静态类图如下：



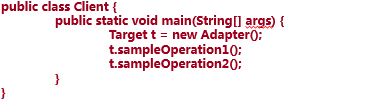
该模式涉及到的角色：



* 类的适配器模式Demo：



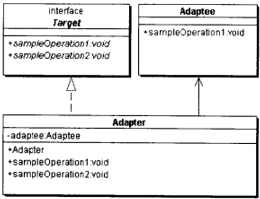




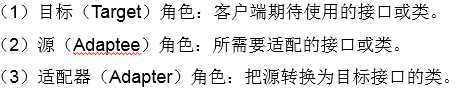
* 对象适配器模式：

与类的适配器模式一样，对象适配器模式把被适配的类的API转换为目标。

类的API，不同的是，对象适配器模式不是使用继承关系连接到Adaptee类，而是使用委派关系连接到Adaptee类。类图如下：



该模式涉及到的角色：

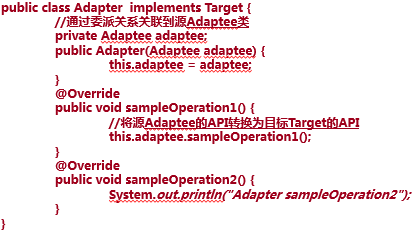


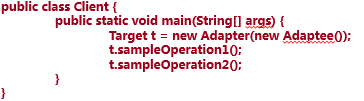
* 对象适配器模式Demo：





说明：Adapter类适配Adaptee及Adaptee的所有子类。





* 设计模式 – 缺省适配器模式（Default Adapter）：

为一个接口提供缺省实现，这样子类型可以从这个缺省实现类进行扩展，而不必从原有的接口进行扩展。通常这个缺省实现类可以是一个抽象类。

**好处**：很多时候一个接口的扩展类不需要实现所有的方法，而是需要某一两个方法即可，从原接口扩展必须实现所有方法，而从该缺省实现类扩展只需要重写需要的方法即可。

**案例**：AWT事件处理。

* 适配器模式总结：

一个类实现目标接口，同时指定目标类的类型作为成员变量，并且在接口的方法实

现中使用目标类型的成员变量。即把这个类和这个类的子类适配到目标接口上。

