## 一句话概括：

**两个不兼容的接口之间的桥梁（不要与桥接模式的定义混淆哦）。**

## 补充介绍：

适配器模式（Adapter Pattern）涉及到一个单一的类，该类负责加入独立的或不兼容的接口功能，它好比一个插座或USB接口，将两个器件连接到一起。

适配器模式用于将一个类的接口转换成客户希望看到的另外一个接口，它使得那些原本因为接口不兼容而不能一起开心玩耍的类可以一起玩耍了。

适配器模式主要应用于将老的或者现存的对象放到一个全新的环境中，而新环境要求的接口时老对象不兼容的。这种模式在代码重构，加增新功能或者代码版本升级的时候有显著功效。

适配器通过继承或依赖已有的对象，实现想要的目标接口。

## 参与角色:

1）现有接口A

2）现有接口B （需要让A的实现类具有B的功能，但是A的方法定义不变）

3）B的实现类 （具有B的具体功能实现）

4）适配器类（实现接口A，并且引用B, 使之具有A的方法定义和B的功能）

5）A的实现类 （实现接口A，并且引用适配器，使之具备A的方法定义，以及A和B的所有功能）

## 优点：

1、可以让任何两个没有关联的类一起运行。 2、提高了类的复用。 3、增加了类的透明度。 4、灵活性好。

## 缺点：

1、过多地使用适配器，会让系统非常零乱，不易整体进行把握。比如，明明看到调用的是 A 接口，其实内部被适配成了 B 接口的实现，一个系统如果太多出现这种情况，无异于一场灾难。因此如果不是很有必要，可以不使用适配器，而是直接对系统进行重构。 2.由于 JAVA 至多继承一个类，所以至多只能适配一个适配者类，而且目标类必须是抽象类。

## 使用案例或场景：

**使用场景**：

代码重构、迁移、升级的时候，需要让现有的类提供新的接口。

**案例：**

现实中的案例：1）老的黑白电视机不支持视频 音频等接口，怎么播放VCD呢，这就需要一个适配器，即有两根视频音频线可以连接VCD又有一根射频线去连接黑白电视机。2）你用单反相机拍摄的精美照骗，需要先取出内存卡，然后通过一个读卡器与笔记本电脑连接起来，这个读卡器就是一个Adapter。

Java语言中的案例：1）Java中的jdbc; 2）Spring MVC中的MappingRequestHandleAdapter等

## 示例程序

需要源码的朋友可以前往github下载：

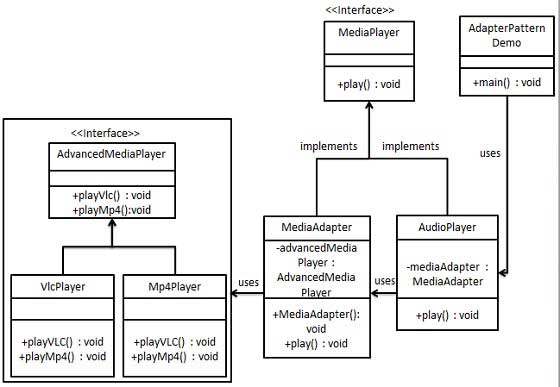
<https://github.com/aharddreamer/chendong/tree/master/design-patterns/demo-code/design-patterns>

**程序简介**：

现有一个MediaPlayer接口和一个实现了该接口的AudioPlayer类，默认情况下，AudioPlayer可以播放mp3格式的音频文件；

还有另外一个AdvancedMediaPlayer接口，以及实现了该接口的VlcPlayer和Mp4Player类，这两个类分别可以播放vlc和mp4格式的音频文件；

那么现在我们要让MediaPlayer接口来播放vlc和mp4格式的音频文件，需要一个实现了MediaPlayer接口的适配器类MediaAdapter, 并使用AdvancedMediaPlayer的对象来播放音频。



**代码：**

**public interface** MediaPlayer {  
 **void** play(String audioType, String fileName);  
}

**public interface** AdvancedMediaPlayer {  
 **void** playVlc(String fileName);  
 **void** playMp4(String fileName);  
}

**public class** VlcPlayer **implements** AdvancedMediaPlayer{  
 @Override  
 **public void** playVlc(String fileName) {  
 System.***out***.println(**"Playing vlc file. Name: "**+ fileName);  
 }  
  
 @Override  
 **public void** playMp4(String fileName) {  
 }  
}

**public class** Mp4Player **implements** AdvancedMediaPlayer{  
  
 @Override  
 **public void** playVlc(String fileName) {  
 *//什么也不做* }  
  
 @Override  
 **public void** playMp4(String fileName) {  
 System.***out***.println(**"Playing mp4 file. Name: "**+ fileName);  
 }  
}

**public class** MediaAdapter **implements** MediaPlayer {  
  
 AdvancedMediaPlayer **advancedMediaPlayer**;  
  
 **public** MediaAdapter(String audioType) {  
 **if** (audioType.equalsIgnoreCase(**"vlc"**)) {  
 **advancedMediaPlayer** = **new** VlcPlayer();  
 }**else if** (audioType.equalsIgnoreCase(**"mp4"**)) {  
 **advancedMediaPlayer** = **new** Mp4Player();  
 }  
 }  
  
 @Override  
 **public void** play(String audioType, String fileName) {  
 **if** (audioType.equalsIgnoreCase(**"vlc"**)) {  
 **advancedMediaPlayer**.playVlc(fileName);  
 }**else if** (audioType.equalsIgnoreCase(**"mp4"**)) {  
 **advancedMediaPlayer**.playMp4(fileName);  
 }  
 }  
}

**public class** AudioPlayer **implements** MediaPlayer {  
  
 MediaAdapter **mediaAdapter**;  
  
 @Override  
 **public void** play(String audioType, String fileName) {  
 *//播放mp3* **if** (audioType.equalsIgnoreCase(**"mp3"**)) {  
 System.***out***.println(**"Playing mp3 file. Name: "**+ fileName);  
 }  
 *//播放vlc或者mp4* **else if** (audioType.equalsIgnoreCase(**"vlc"**)  
 || audioType.equalsIgnoreCase(**"mp4"**)) {  
 **mediaAdapter** = **new** MediaAdapter(audioType);  
 **mediaAdapter**.play(audioType, fileName);  
 }  
 *//其他格式不支持* **else** {  
 System.***out***.println(**"Not supported given music file"**);  
 }  
 }  
}

**public class** AdapterPatternTest {  
 **public static void** main(String[] args) {  
 MediaPlayer player = **new** AudioPlayer();  
 player.play(**"mp3"**, **"i miss u.mp3"**);  
 player.play(**"vlc"**, **"i like u.vlc"**);  
 player.play(**"mp4"**, **"i love u.mp4"**);  
 player.play(**"avi"**, **"i hate u.avi"**);  
 }  
}

运行结果：

Playing mp3 file. Name: i miss u.mp3

Playing vlc file. Name: i like u.vlc

Playing mp4 file. Name: i love u.mp4

Not supported given music file

参考：

《建造者模式》菜鸟教程网站