**题目：**

小王正在开发一套视频监控系统，考虑到Windows Media Player和Real Player是两种常用的媒体播放器，尽管它们的API结构和调用方法存在区别，但这套应用程序必须支持这两种播放器API，而且在将来可能还需要支持新的媒体播放器。请你针对上面的描述，帮助小王进行设计，给出设计思路及所采用的设计模式，画出类图，并写出关键代码。

**解答：**

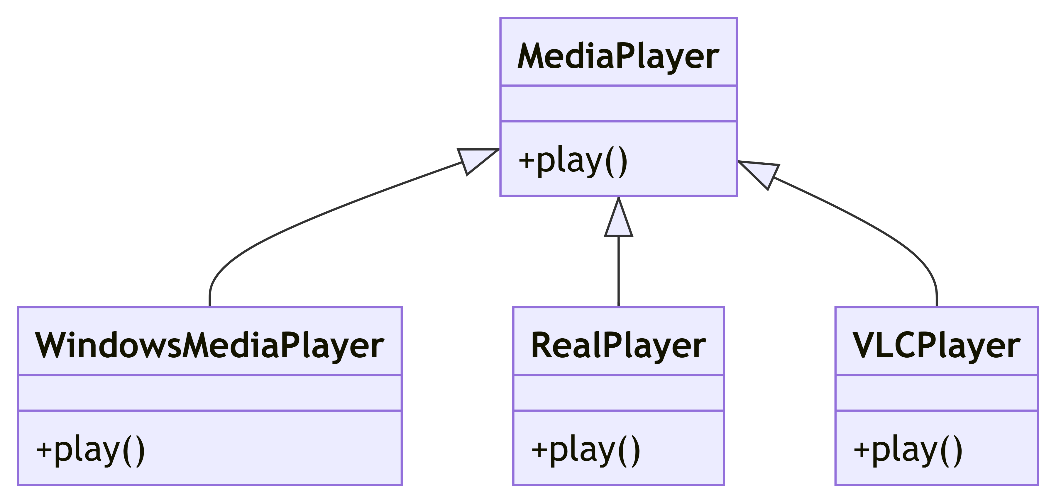
* 设计思路：

1. 定义播放器的抽象接口，提供统一的操作方法
2. 为不同的播放器实现具体的操作方法，实现各播放器的具体操作
3. 当需要支持新的播放器时，只需要增加新的实现类，而不需要修改现有的代码结构

* 模式：

使用**桥接模式**来设计这套系统。桥接模式的核心思想是将抽象部分与它的实现部分分离，使它们都可以独立变化。这样可以方便地添加新的播放器，而不需要修改原有代码。

* 类图：



|  |
| --- |
| * 核心代码： |
| from abc import ABC, abstractmethod  *# 抽象媒体播放器类*  class MediaPlayer(ABC):      @abstractmethod      def play(self):          pass  *#以Windows Media Player 和Real Player和VLC Player为例*  *# Windows Media Player具体实现*  class WindowsMediaPlayer(MediaPlayer):      def play(self):          print("Playing video using Windows Media Player")  *# Real Player具体实现*  class RealPlayer(MediaPlayer):      def play(self):          print("Playing video using Real Player")  *# VLC Player具体实现*  class VLCPlayer(MediaPlayer):      def play(self):          print("Playing video using VLC Player")  *#客户端*  def client\_code(player: MediaPlayer):      player.play()  *# 使用Windows Media Player播放*  windows\_player = WindowsMediaPlayer()  client\_code(windows\_player)  *# 使用Real Player播放*  real\_player = RealPlayer()  client\_code(real\_player)  *# VLC Player具体实现*  class VLCPlayer(MediaPlayer):      def play(self):          print("Playing video using VLC Player")  *# 增加新播放器：VLC Player*  vlc\_player = VLCPlayer()  client\_code(vlc\_player) |