Firstly, I have created the abstract classes for the **target interfaces:**

- **MP4MediaPlayer**
- **VLCMediaPlayer**

```python
from abc import ABC, abstractmethod
class MP4MediaPlayer(ABC):
    Tabnine | Edit | Test | Explain | Document | Ask
    def play_mp4(self, filename):
        Processing your request... | Cancel request
        pass

class VLCMediaPlayer(ABC):
    Tabnine | Edit | Test | Explain | Document | Ask
    def play_vlc(self, filename):
        pass
```

These classes extend the ABC class, which makes them abstract classes that have no implementations. But it defines the **functionality that the clients expect** from the media player, by defining the **function signature**.

Then I created the **MediaPlayer** class that only plays mp3 files, otherwise **throws an error**:

```python
class MediaPlayer:
    Tabnine | Edit | Test | Explain | Document | Ask
    def __init__(self):
        pass
    Tabnine | Edit | Test | Explain | Document | Ask
    def play_mp3(self, filename):
        #check if filename ends with .mp3 extension
        if filename.split(".")[1]=="mp3":
            print(f"Playing song: {filename}")
        else:
            print(f"Unsupported file format: {filename}")
```

Then I have created 2 separate adapters for handling VLC and MP4 file extension and to make them compatible with the MediaPlayer class:

- **MP3MP4Adapter**
- **MP3VLCAdapter**

They extend the respective mp4 and vlc player classes and override their funcitons

The two adapter classes have the following implementations:

```
class MP3MP4Adapter(MP4MediaPlayer):
    Tabnine | Edit | Test | Explain | Document | Ask
    def __init__(self, media_player: MediaPlayer):
        self.media_player = media_player
    Tabnine | Edit | Test | Explain | Document | Ask
    def play_mp4(self, filename):
        #convert mp4 file to mp3 format
        print("Converting mp4 file to mp3 format...")
        name = filename.split(".")
        name[1] = "mp3"
        converted_filename = (".").join(name)
        self.media_player.play_mp3(converted_filename)
```

```
class MP3VLCAdapter(VLCMediaPlayer):
    Tabnine | Edit | Test | Explain | Document | Ask
    def __init__(self, media_player: MediaPlayer):
        self.media_player = media_player
    Tabnine | Edit | Test | Explain | Document | Ask
    def play_vlc(self, filename):
        #convert mp4 file to mp3 format
        print("Converting VLC file to mp3 format...")
        name = filename.split(".")
        name[1] = "mp3"
        converted_filename = (".").join(name)
        self.media_player.play_mp3(converted_filename)
```

Testing the code:

Here we create a single **media_player** instance from the **MediaPlayer** class and using it to seamlessly play all file formats (**.mp4** and **.vlc**) using the adapters and respective functions.

```python
if __name__ == "__main__":
    mp3_player = MediaPlayer()
    mp3_player.play_mp3("title_song.mp3")

    mp4_player = MP3MP4Adapter(mp3_player)
    mp4_player.play_mp4("random_song.mp4")

    vlc_player = MP3VLCAdapter(mp3_player)
    vlc_player.play_vlc("video_game.vlc")
```