Design Patterns - Adapter Pattern

Adapter pattern works as a bridge between two incompatible interfaces. This type of design pattern comes under structural pattern as this pattern combines the capability of two independent interfaces.

This pattern involves a single class which is responsible to join functionalities of independent or incompatible interfaces. A real life example could be a case of card reader which acts as an adapter between memory card and a laptop. You plugin the memory card into card reader and card reader into the laptop so that memory card can be read via laptop.

We are demonstrating use of Adapter pattern via following example in which an audio player device can play mp3 files only and wants to use an advanced audio player capable of playing vlc and mp4 files.

适配器模式作为桥梁，连接两个不兼容的接口。这种类型的设计模式来源于结构型模式，它具有结合两个相互独立的接口的能力。

这个模式涉及到单个类，该类负责接入独立的、不兼容的接口。一个现实生活的例子，比如说读卡器，它可能会在记忆卡和笔记本电脑之间扮演一个适配者的角色。首先把记忆卡插到读卡器上，在把读卡器插入笔记本上，然后我们就可以从笔记本读取记忆卡上的数据。

我们通过以下的例子来展示适配器模式。一个音频播放器设备只能播放mp3文件；而另一个比较先进的音乐播放器可以播放vlc和mp4文件。

Implementation

实现

We have a *MediaPlayer* interface and a concrete class *AudioPlayer* implementing the *MediaPlayer* interface. *AudioPlayer* can play mp3 format audio files by default.

我们有一个MediaPlayer接口和一个实现该接口的实体类AudioPlayer，这个AudioPlayer默认播放mp3格式的音频。

We are having another interface *AdvancedMediaPlayer* and concrete classes implementing the *AdvancedMediaPlayer* interface. These classes can play vlc and mp4 format files.

我们还有另外一个接口 AdvancedMediaPalyer 和实现该接口的实体类

这些实体类可以播放vlc和mp4格式的音频。

We want to make *AudioPlayer* to play other formats as well. To attain this, we have created an adapter class *MediaAdapter* which implements the *MediaPlayer*interface and uses *AdvancedMediaPlayer* objects to play the required format.

我们希望AudioPlayer也可以播放其他格式的文件。为了实现这个目标，我们创建了一个适配器类MediaAdapter，该类实现了接口MediaPlayer，并且使用AdvancedMediaPlayer的对象来播放需要的格式。

*AudioPlayer* uses the adapter class *MediaAdapter* passing it the desired audio type without knowing the actual class which can play the desired format.*AdapterPatternDemo*, our demo class will use *AudioPlayer* class to play various formats.

AudioPlayer 使用适配器类 MediaAdapter，通过它来播放所期望的音频类型，不需要知道实际是哪个类播放这个期望的音频类型。AdapterPatternDemo，我们的demo类将使用AudioPlayer类来播放各种格式的音频。



Step 1

第一步

Create interfaces for Media Player and Advanced Media Player.

创建MediaPlayer和AdvancedMediaPlayer接口。

*MediaPlayer.java*

public interface MediaPlayer {

public void play(String audioType, String fileName);

}

*AdvancedMediaPlayer.java*

public interface AdvancedMediaPlayer {

public void playVlc(String fileName);

public void playMp4(String fileName);

}

Step 2

第二步

Create concrete classes implementing the *AdvancedMediaPlayer* interface.

创建实体类实现AdvancedMediaPlayer接口。

*VlcPlayer.java*

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) {

//do nothing

}

}

*Mp4Player.java*

public class Mp4Player implements AdvancedMediaPlayer{

@Override

public void playVlc(String fileName) {

//do nothing

}

@Override

public void playMp4(String fileName) {

System.out.println("Playing mp4 file. Name: "+ fileName);

}

}

Step 3

第三步

Create adapter class implementing the *MediaPlayer* interface.

创建一个适配器类实现MediaPlayer接口

*MediaAdapter.java*

public class MediaAdapter implements MediaPlayer {

AdvancedMediaPlayer advancedMusicPlayer;

public MediaAdapter(String audioType){

if(audioType.equalsIgnoreCase("vlc") ){

advancedMusicPlayer = new VlcPlayer();

}else if (audioType.equalsIgnoreCase("mp4")){

advancedMusicPlayer = new Mp4Player();

}

}

@Override

public void play(String audioType, String fileName) {

if(audioType.equalsIgnoreCase("vlc")){

advancedMusicPlayer.playVlc(fileName);

}

else if(audioType.equalsIgnoreCase("mp4")){

advancedMusicPlayer.playMp4(fileName);

}

}

}

Step 4

第四步

Create concrete class implementing the *MediaPlayer* interface.

创建实体类实现MediaPlayer接口。

*AudioPlayer.java*

public class AudioPlayer implements MediaPlayer {

MediaAdapter mediaAdapter;

@Override

public void play(String audioType, String fileName) {

//inbuilt support to play mp3 music files

//内置支持播放MP3类型的音乐

if(audioType.equalsIgnoreCase("mp3")){

System.out.println("Playing mp3 file. Name: " + fileName);

}

//mediaAdapter is providing support to play other file formats

else if(audioType.equalsIgnoreCase("vlc") || audioType.equalsIgnoreCase("mp4")){

mediaAdapter = new MediaAdapter(audioType);

mediaAdapter.play(audioType, fileName);

}

else{

System.out.println("Invalid media. " + audioType + " format not supported");

}

}

}

Step 5

第五步

Use the AudioPlayer to play different types of audio formats.

使用AudioPlayer播放不同种类的音频格式。

*AdapterPatternDemo.java*

public class AdapterPatternDemo {

public static void main(String[] args) {

AudioPlayer audioPlayer = new AudioPlayer();

audioPlayer.play("mp3", "beyond the horizon.mp3");

audioPlayer.play("mp4", "alone.mp4");

audioPlayer.play("vlc", "far far away.vlc");

audioPlayer.play("avi", "mind me.avi");

}

}

Step 6

Verify the output.

校验输出。

Playing mp3 file. Name: beyond the horizon.mp3

Playing mp4 file. Name: alone.mp4

Playing vlc file. Name: far far away.vlc

Invalid media. avi format not supported