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.



//step 1

//create interfaces for Media player and advanced Media player

interface MediaPlayer{

public void play(String audioType,String fileName);

}

interface AdvancedMediaPlayer{

public void playVlc(String fileName);

public void playMp4(String fileName);

}

//step 2

//create concrete classes implementing the advancedMediaPlayer interface

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

}

}

class Mp4Player implements AdvancedMediaPlayer{

public void playMp4(String fileName) {

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

}

@Override

public void playVlc(String fileName) {

//do nothing

}

}

//step 3

//create adapter class implementing the mediaplayer interface

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);

}

}

}

//step 4

//create concrete class implementing the MediaPlayer interface

//Audio player java

class AudioPlayer implements MediaPlayer{

MediaAdapter mediaAdapter;

@Override

public void play(String audioType, String fileName) {

//inbuilt support to play mp3 music files

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

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

}else

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

//mediaadapter is providing support to play other file formats

mediaAdapter = new MediaAdapter(audioType);

mediaAdapter.play(audioType, fileName);

}else {

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

}

}

}

//step5

//use the audioplayer to play different types of audio formats

public class AdapterPatternDemo {

public static void main(String[] args) {

AudioPlayer audioPlayer = new AudioPlayer();

audioPlayer.play("mp3", "beyon\_the\_horizon.mp3");

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

audioPlayer.play("vlc", "far\_far\_away.vlc");

audioPlayer.play("avi", "mind\_me.avi");

}

}