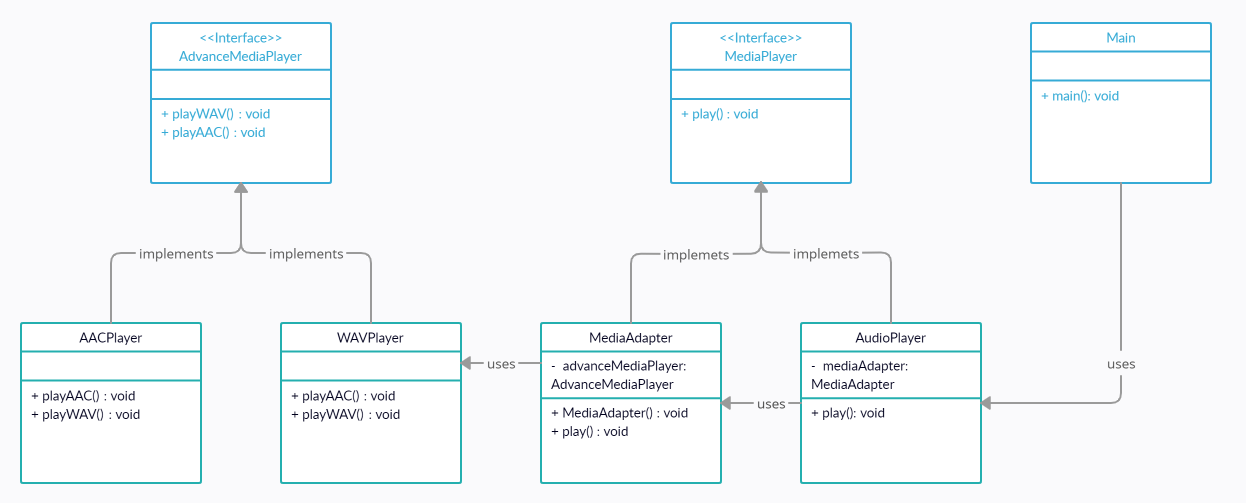
**Assignment 7: Adapter Design Pattern**

**What is Adapter Design Pattern?**

Adapter is a structural design pattern that allows objects with incompatible interfaces to collaborate.

**Structure (Class Diagram)**

****

A Car has an audio player. It plays a different types of audio files. To play those we need an **Adapter Class** to convert those files to mp3 file and play the audio. I have created a program which supports only **“.mp3”, “.aac” & “.wav”** formats. If any other file format is used then the code gives a message that the **“File format is not supported”**.

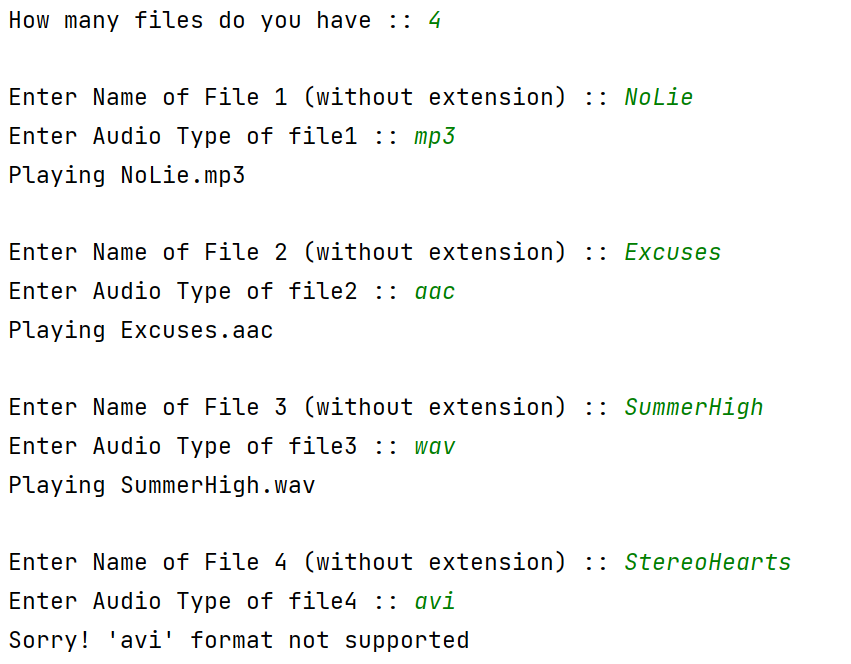
**Implementation (Code)**

interface *MediaPlayer* {  
 public void *play*(*String audioType* ,*String fileName*);  
}  
  
interface *AdvanceMediaPlayer* {  
 public void *playAAC*(*String audioType*, *String fileName*);  
 public void *playWAV*(*String audioType*, *String fileName*);  
}  
  
class *AACPlayer* implements *AdvanceMediaPlayer* {  
 public void *playAAC*(*String audioType*, *String fileName*) {  
 *System*.out.*println*("Playing " + *fileName* + "." + *audioType*);  
 }  
 public void *playWAV*(*String audioType*, *String fileName*) {  
 *// Do Nothing* }  
}

class *WAVPlayer* implements *AdvanceMediaPlayer* {  
 public void *playAAC*(*String audioType*, *String fileName*) {  
 *// Do Nothing* }  
 public void *playWAV*(*String audioType*, *String fileName*) {  
 *System*.out.*println*("Playing " + *fileName* + "." + *audioType*);  
 }  
}  
  
class *MediaAdapter* implements *MediaPlayer* {  
 *AdvanceMediaPlayer* advMusicPlay;  
  
 public *MediaAdapter*(*String audioType*){  
 if (*audioType*.*equalsIgnoreCase*("aac")) {  
 advMusicPlay = new *AACPlayer*();  
 } else if (*audioType*.*equalsIgnoreCase*("wav")) {  
 advMusicPlay = new *WAVPlayer*();  
 }  
 }  
  
 public void *play* (*String audioType*, *String fileName*) {  
 if (*audioType*.*equalsIgnoreCase*("aac")) {  
 advMusicPlay.*playAAC*(*audioType*, *fileName*);  
 } else if (*audioType*.*equalsIgnoreCase*("wav")) {  
 advMusicPlay.*playWAV*(*audioType*, *fileName*);  
 }  
 }  
}  
  
class *AudioPlayer* implements *MediaPlayer* {  
 *MediaAdapter* mediaAdapter;  
  
 public void *play*(*String audioType*, *String fileName*) {  
 if (*audioType*.*equalsIgnoreCase*("mp3")) {  
 *System*.out.*println*("Playing " + *fileName* + "." + *audioType*);  
 }  
 else if (*audioType*.*equalsIgnoreCase*("aac") || *audioType*.*equalsIgnoreCase*("wav")) {  
 mediaAdapter = new *MediaAdapter*(*audioType*);  
 mediaAdapter.*play*(*audioType*, *fileName*);  
 }  
 else {  
 *System*.out.*println*("Sorry! '" + *audioType* + "' format not supported");  
 }  
 }  
}

**Output**

public class *Main* {  
 public static void *main*(*String*[] *args*) {  
 *AudioPlayer* audioPlayer = new *AudioPlayer*();  
 *Scanner* sc = new *Scanner*(*System*.in);  
  
 *System*.out.*print*("How many files do you have :: ");  
 int n = sc.*nextInt*();  
  
 for (int i=1 ; i<=n ; i++) {  
 *System*.out.*print*("\nEnter Name of File " + i + " (without extension) :: ");  
 *String* fileName = sc.*next*();  
 *System*.out.*print*("Enter Audio Type of file" + i + " :: ");  
 *String* audioType = sc.*next*();  
  
 audioPlayer.*play*(audioType, fileName);  
 }  
 }  
}



**Applicability**

1. **Use the Adapter class when you want to use some existing class, but its interface isn’t compatible with the rest of your code.**

The Adapter pattern lets you create a middle-layer class that serves as a translator between your code and a legacy class, a 3rd-party class or any other class with a weird interface.

1. **Use the pattern when you want to reuse several existing subclasses that lack some common functionality that can’t be added to the superclass.**

You could extend each subclass and put the missing functionality into new child classes. However, you’ll need to duplicate the code across all of these new classes.