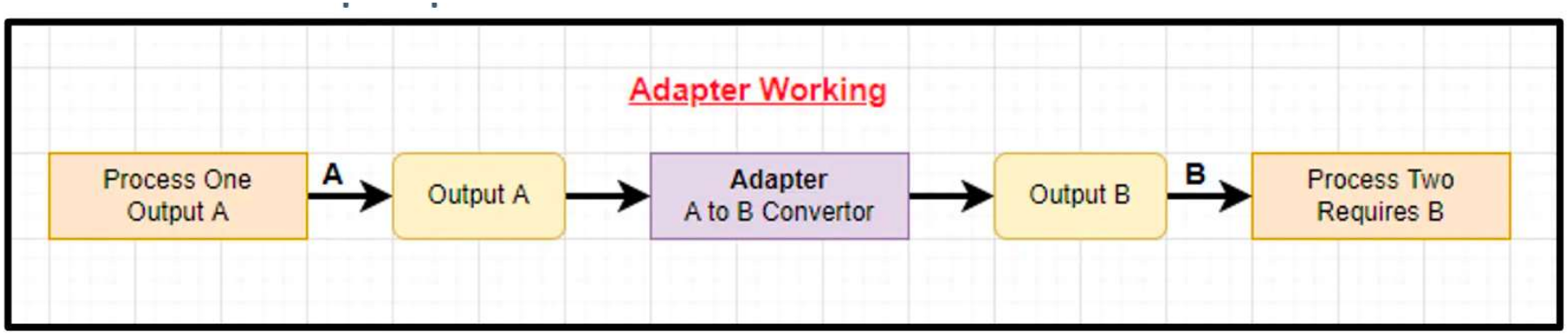
Adapter Design Pattern


TRAVEL
UNIVERSAL
ADAPTOR
SURGE
PROTECTOR

Working


Adapter Working
Process One
Output A
A
Output A
Adapter
A to B Convertor
Output B
B
Process Two
Requires B

Adapter Design Pattern

An Adapter Pattern says that just "converts the interface of a class into another interface that a client wants".

In other words, to provide the interface according to client requirement while using the services of a class with a different interface.

The Adapter Pattern is also known as Wrapper.

Advantage of Adapter Pattern

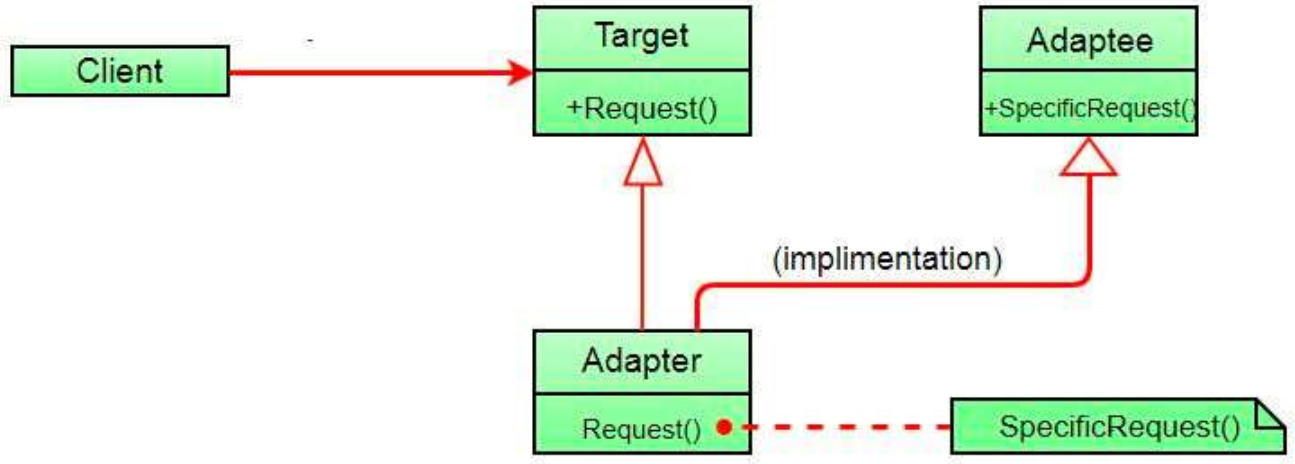
It allows two or more previously incompatible objects to interact.

It allows reusability of existing functionality.

Usage

- When an object needs to utilize an existing class with an incompatible interface.
- When you want to create a reusable class that cooperates with classes which don't have compatible interfaces.
- When you want to create a reusable class that cooperates with classes which don't have compatible interfaces.

Major Components



Major Components

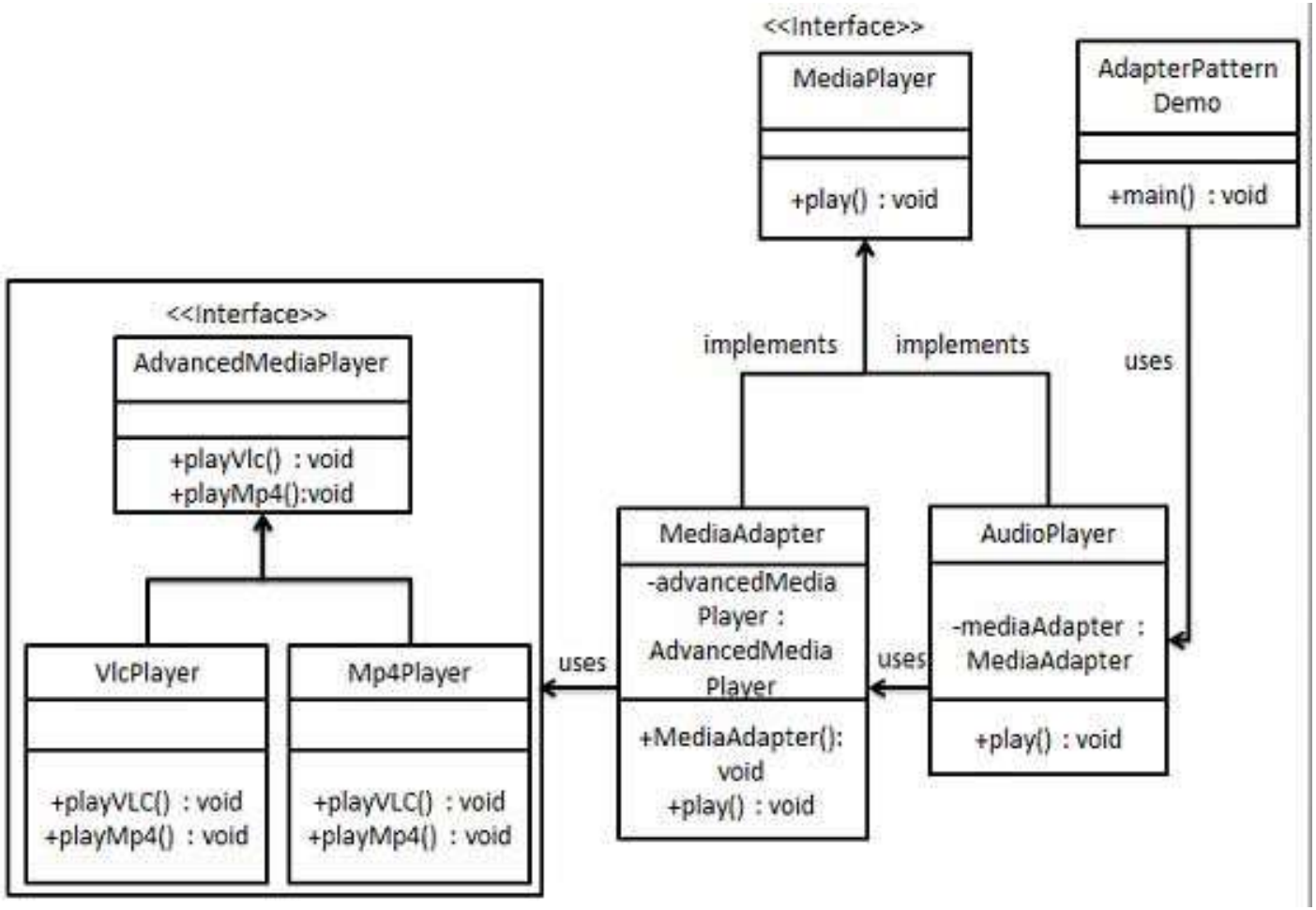
Target Interface: This is the desired interface class which will be used by the clients.

Adapter class: This class is a wrapper class which implements the desired target interface and modifies the specific request available from the Adaptee class.

Adaptee class: This is the class which is used by the Adapter class to reuse the existing functionality and modify them for desired use.

Client: This class will interact with the Adapter class.

Example


<<Interface>>
MediaPlayer
+play() : void
AdapterPattern Demo
+main() : void
implements
implements
uses
<<Interface>>
AdvancedMediaPlayer
+playVlc() : void
+playMp4():void
MediaAdapter
-advancedMedia Player : AdvancedMedia Player
+MediaAdapter(): void
+play() : void
AudioPlayer
-mediaAdapter : MediaAdapter
+play() : void
uses
uses
VlcPlayer
+playVLC() : void
+playMp4() : void
Mp4Player
+playVLC() : void
+playMp4() : void

MediaPlayer.java

```java
public interface MediaPlayer {
   public void play(String audioType, String fileName);
}
```

AdvancedMediaPlayer.java

```java
public interface AdvancedMediaPlayer {
   public void playVlc(String fileName);
   public void playMp4(String fileName);
}
```

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

```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);
    }
}
```

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);
      }
   }
}
```

Create concrete class implementing the *MediaPlayer* interface.

AudioPlayer.java

```
public 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);
      }

      //mediaAdapter is providing support to play other file formats
      else if(audioType.equalsIgnoreCase("vlc") || audioType.equalsIgnoreCase('
         mediaAdapter = new MediaAdapter(audioType);
         mediaAdapter.play(audioType, fileName);
      }

      else{
         System.out.println("Invalid media. " + audioType + " format not suppo
      }
   }
}
```

Use the AudioPlayer to play different types of audio formats.

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.

```
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
```