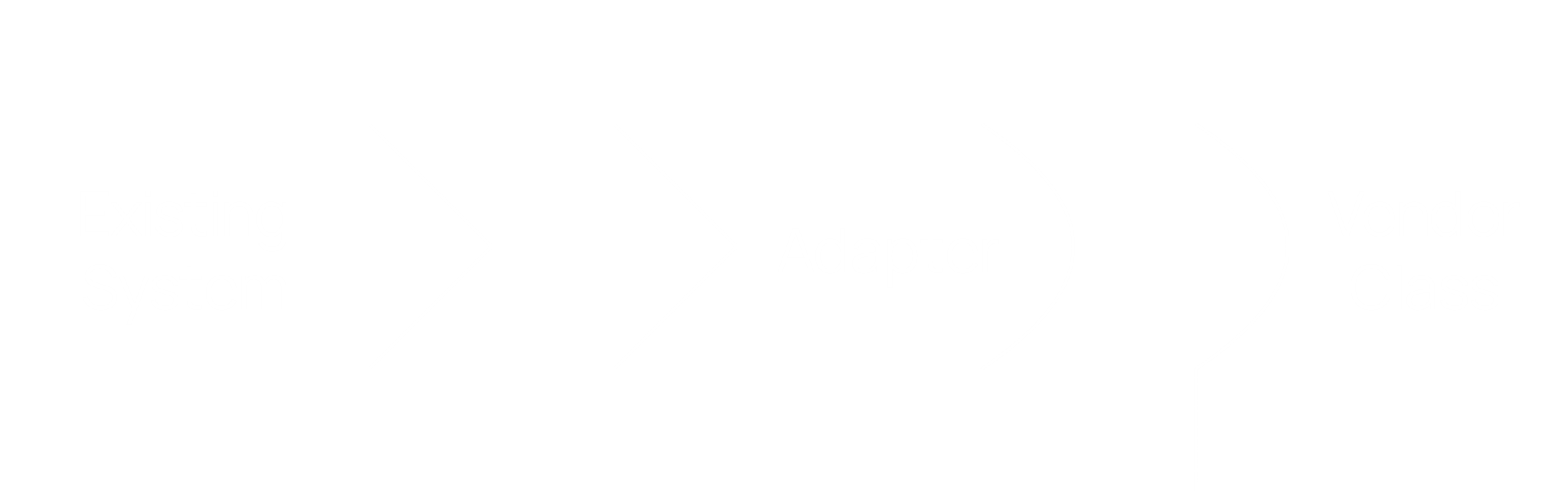
Adapter Pattern

The **Adapter Pattern** is a **structural pattern** that works as a bridge between two incompatible interfaces. Suppose we have access to some functionality using an interface, but the interface is not compatible with our existing code. To get around this issue, we can create an adapter that will allow our code to interact with the interface.



The Adapter Pattern allows us to follow the **Dependency Inversion Principle**:

**Dependency Inversion Principle**: High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend upon abstractions.

Example:

Suppose there is a complicated library that allows us to play music. It expects us to call a different method based on the type of audio we want to play.

public interface *AdvancedMediaPlayer* {  
 public void playVLC(String filename);  
 public void playMP4(String filename);  
}

JAVA

We don’t want to get overwhelmed with all of this information, so we decide to make a simpler interface.

public interface *MediaPlayer* {  
 public void play(String audioType, String fileName):  
}

JAVA

The **Adapter Pattern** allows us to connect these two interfaces so that we can use our simpler interface instead of having to deal with the complicated one.

public class VLCPlayer implements *AdvancedMediaPlayer* {  
 @Override  
 public void playVLC(String filename) {  
 // code  
 }  
  
 @Override  
 public void playMP4(String filename) {  
 // do nothing  
 }  
}  
  
public class MP4Player implements *AdvancedMediaPlayer* {  
 @Override  
 public void playVLC(String filename) {  
 // do nothing  
 }  
  
 @Override  
 public void playMP4(String filename) {  
 // code  
 }  
}

public 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);  
 }  
 }  
}

public class AudioPlayer implements *MediaPlayer* {  
 MediaAdapter mediaAdapter;  
  
 @Override  
 public void play(String audioType, String fileName) {  
 mediaAdapter.play(audioType, fileName);  
 }  
}

JAVA

Finally, we can present the client with our simple audio player.

public class AdapterPatternDemo {  
 public static void main(String[] args) {  
 AudioPlayer audioPlayer = new AudioPlayer();  
 audioPlayer.play("mp4", "video.mp4");  
 }  
}

JAVA