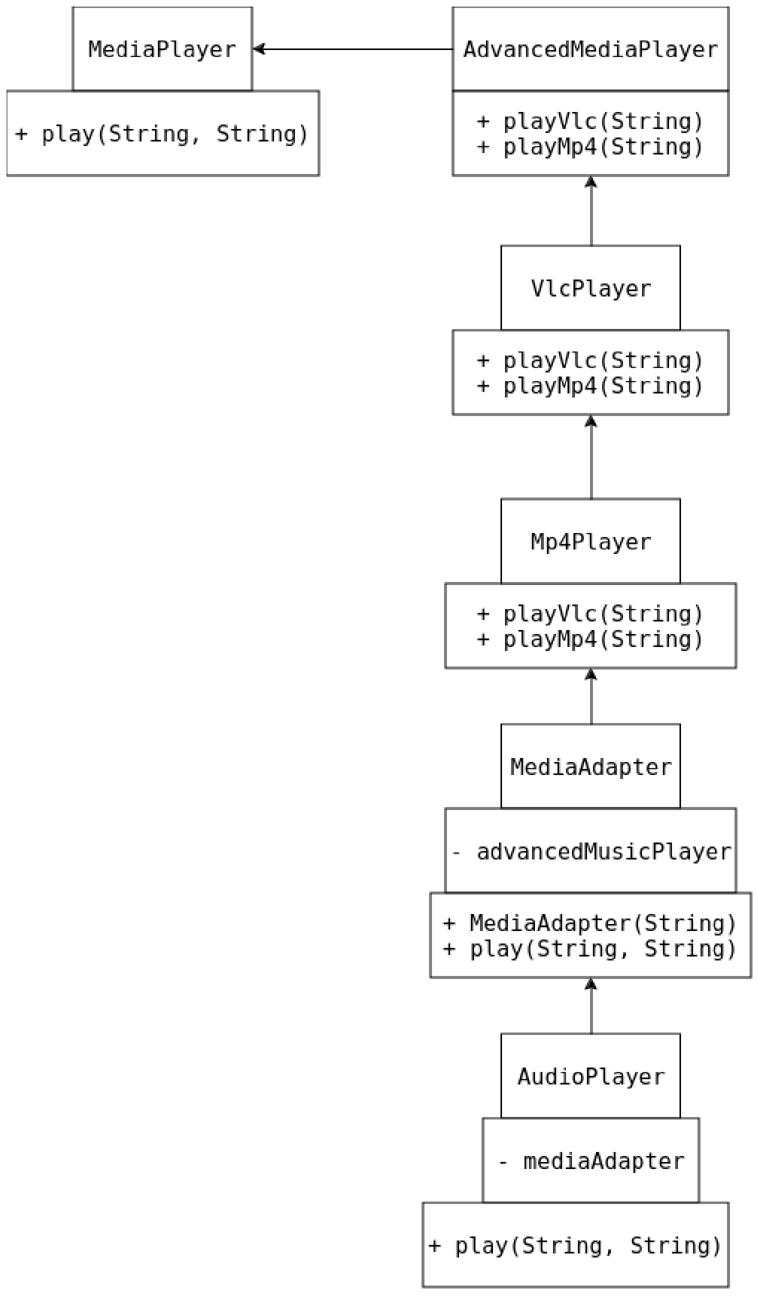
Adapter Design Pattern

Description:

The adapter design pattern is a structural pattern that allows incompatible interfaces between classes to work together. Acting as a bridge, the adapter pattern converts the interface of a class into another interface that a client expects. This enables classes with incompatible interfaces to collaborate without modifying their source code. Adapters typically wrap an existing class, providing a compatible interface to the client while delegating the actual functionality to the wrapped class. This pattern is particularly useful when integrating existing or third-party components into a system that expect different interfaces, allowing them to seamlessly communicate and interoperate. Additionally, adapters promote code reuse and maintainability by facilitating the integration of new components without requiring extensive modifications to existing codebases.

Class diagram:



Implementation:

1. MediaPlayer.java

```
package Adapter;

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

2. AdvancedMediaPlayer.java

```
package Adapter;

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

3. VlcPlayer.java

```
package Adapter;

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
	}
}
```

4. Mp4Player.java

```
package Adapter;

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);
        }
}
```

5. MediaAdapter.java

```
package Adapter;

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);
                }
        }
}
```

6. AudioPlayer.java

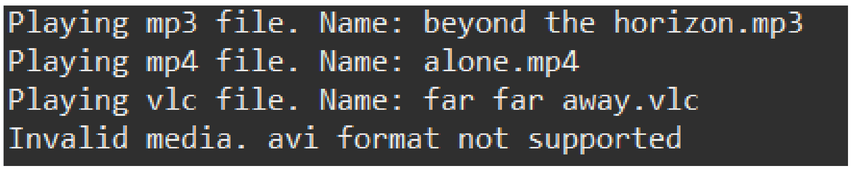
```
package Adapter;

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("mp4"))
                    {
                              mediaAdapter = new MediaAdapter(audioType);
                              mediaAdapter.play(audioType, fileName);
                    }
                    else
                    {
                              System.out.println("Invalid media. " + audioType + " format not supported");
                    }
          }
}
```

7. AdapterPatternDemo.java

```
package Adapter;

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");
          }
}
```

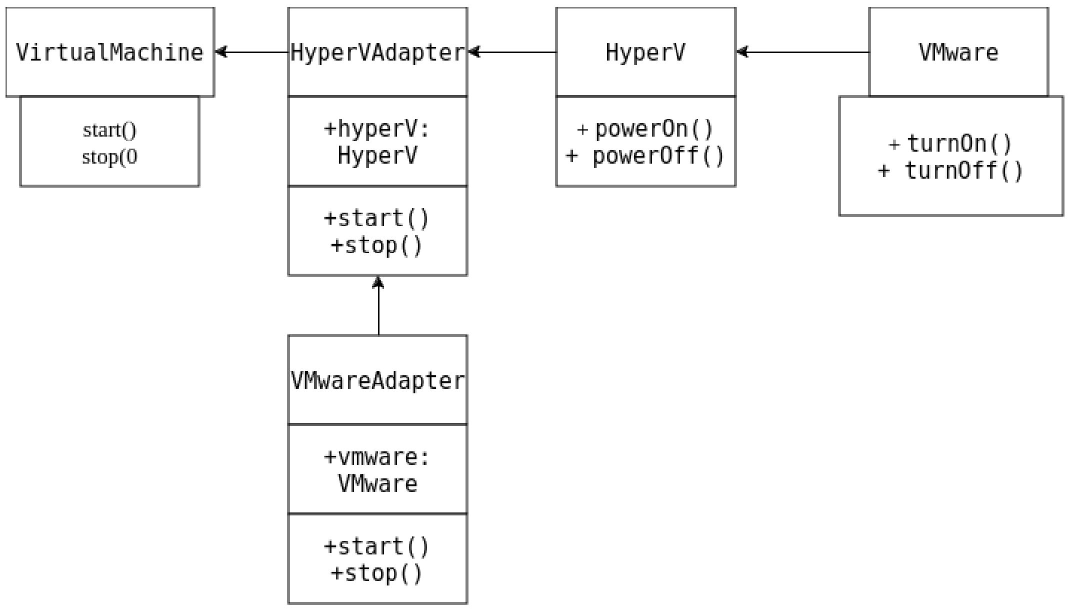
Output:



```
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
```

Adapter Design Pattern for different types of Virtual Machines

Class diagram:

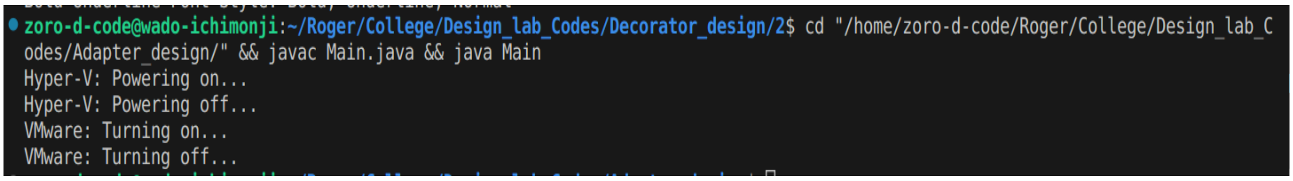


Implementation:

1. Main.java

```
// Target interface
interface VirtualMachine {
    void start();
    void stop();
}

// Adaptee 1
class HyperV {
    void powerOn() {
        System.out.println("Hyper-V: Powering on...");
    }

    void powerOff() {
        System.out.println("Hyper-V: Powering off...");
    }
}

// Adapter for Hyper-V
class HyperVAdapter implements VirtualMachine {
    private HyperV hyperV;

    public HyperVAdapter(HyperV hyperV) {
        this.hyperV = hyperV;
    }
    @Override
    public void start() {
        hyperV.powerOn();
    }

    @Override
    public void stop() {
        hyperV.powerOff();
    }
}

// Adaptee 2
class VMware {
    void turnOn() {
        System.out.println("VMware: Turning on...");
    }

    void turnOff() {
        System.out.println("VMware: Turning off...");
    }
}

// Adapter for VMware
class VMwareAdapter implements VirtualMachine {
    private VMware vmware;

    public VMwareAdapter(VMware vmware) {
        this.vmware = vmware;
    }

    @Override
    public void start() {
        vmware.turnOn();
    }

    @Override
    public void stop() {
        vmware.turnOff();
    }
}

// Client code
public class Main {
    public static void main(String[] args) {
        // Using the Hyper-V adapter
        HyperV hyperV = new HyperV();
        VirtualMachine hyperVAdapter = new HyperVAdapter(hyperV);
        hyperVAdapter.start();
        hyperVAdapter.stop();

        // Using the VMware adapter
        VMware vmware = new VMware();
        VirtualMachine vmwareAdapter = new VMwareAdapter(vmware);
        vmwareAdapter.start();
        vmwareAdapter.stop();
    }
}
```

Output:



```
zoro-d-code@wado-ichimonji:~/Roger/College/Design_lab_Codes/Decorator_design/2$ cd "/home/zoro-d-code/Roger/College/Design_lab_C
odes/Adapter_design/" && javac Main.java && java Main
Hyper-V: Powering on...
Hyper-V: Powering off...
VMware: Turning on...
VMware: Turning off...
```