**Adapter Design Pattern**

**Assignment - 1**

Name - Heet Dobariya Roll No. - 22BCP177 Group - G5

* **Adapter Design Pattern :**

The adapter design pattern finds common application in software development across various scenarios. Here's how it can be put:

Usage scenarios:

1. Incorporating third-party libraries: When dealing with external libraries or APIs that don't align with the existing codebase's interfaces, the adapter pattern proves useful. It allows for creating intermediary wrappers that adapt the external interfaces to conform with the application's interface standards.

2. Integration with legacy systems: When modern applications need to interface with legacy systems, the adapter pattern bridges the gap between their disparate interfaces, facilitating seamless communication.

3. Standardizing interfaces: In complex systems with multiple subsystems or modules, each might adhere to its own interface conventions. Employing the adapter pattern helps in standardizing interfaces across various components, simplifying system maintenance and extension.

Advantages:

1. Reusability: Adapters serve as reusable components, capable of being employed across different parts of a system or even in separate projects, as they encapsulate the logic for interface adaptation.

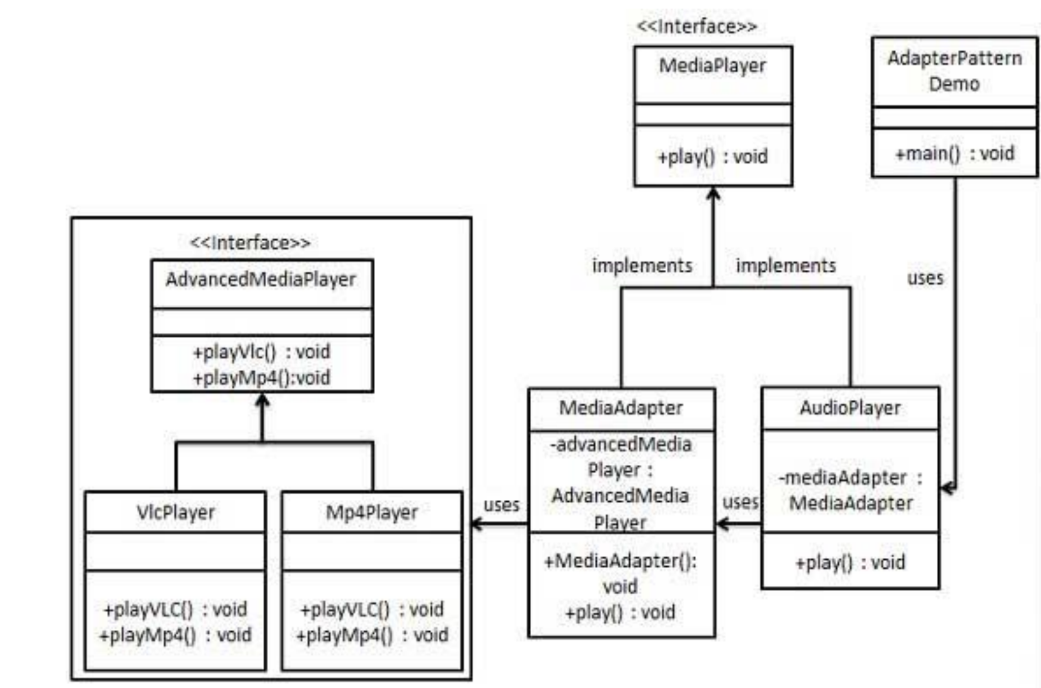
2. Flexibility: By facilitating integration between components boasting different interfaces, the adapter pattern fosters system flexibility, easing maintenance and evolution.

3. Interoperability: By harmonizing incompatible interfaces, the adapter pattern promotes seamless interaction among diverse components, systems, or modules.

4. Encapsulation: Adapters encapsulate the intricacies of interface adaptation, shielding client code from the nitty-gritty details. This enhances code maintainability and comprehensibility.

In summary, the adapter design pattern proves invaluable for seamlessly integrating disparate components or systems featuring incompatible interfaces, all while fostering code reusability and maintainability.

* **Program :** Implement adapter design pattern for Media Player example.
* **UML Diagram :**



* **Code :**

interface MediaPlayer{

    public void play(String audioType, String fileName);

}

interface AdvancedMediaPlayer{

    public void playVlc(String fileName);

    public void playMp4(String fileName);

}

class VlcPlayer implements AdvancedMediaPlayer{

    @Override

    public void playVlc(String fileName){

        System.out.println("Playing VLC file name : " + fileName);

    }

    public void playMp4(String fileName){

        // Do nothing

    }

}

class Mp4Player implements AdvancedMediaPlayer{

    @Override

    public void playVlc(String fileName){

        // Do nothing

    }

    public void playMp4(String fileName){

        System.out.println("Playing MP4 file name : " + fileName);

    }

}

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();

        }

    }

    public void play(String audioType, String fileName){

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

            advancedMusicPlayer.playVlc(fileName);

        }

        else if(audioType.equalsIgnoreCase("mp4")){

            advancedMusicPlayer.playMp4(fileName);

        }

    }

}

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 = new MediaAdapter(audioType);

            mediaAdapter.play(audioType,fileName);

        }

        else{

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

        }

    }

}

class Client{

    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.mp3");

        audioPlayer.play("avi" , "Mind Me.mp3");

    }

}

* **Output :**

