Problem No: 05

Problem Name: Design and Implement the Adapter Design Pattern.

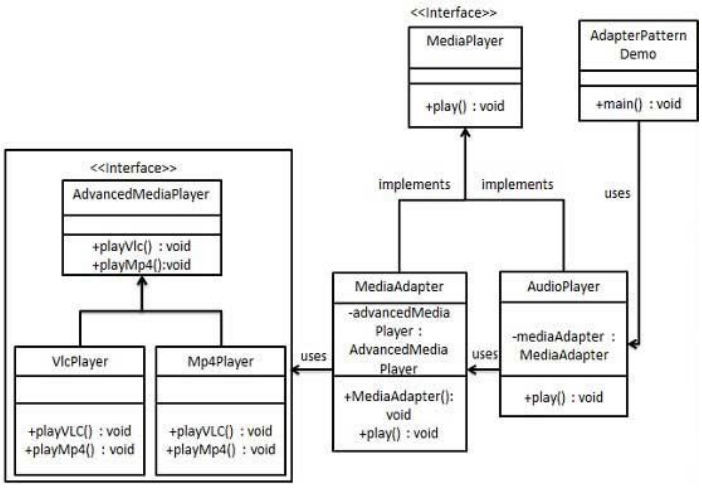
Objectives:

- To understand the Adapter Design Pattern and its purpose.
- To implement the Adapter Design Pattern using a class in an object-oriented language.
- To demonstrate how the Adapter pattern helps to convert one interface to another, making two incompatible interfaces compatible.

Theory:

The Adapter pattern is used when a class (or system) needs to interact with another class (or system) whose interface is not compatible. It provides a way to adapt the interface of one class so it can be used by another class without modifying the source code.

UML/User-defined Class Design:



Program (Java):

```
interface MediaPlayer {
   void play(String audioType, String fileName);
}
class AudioPlayer {
   public void playAudio(String fileName) {
      System.out.println("Playing audio file: " + fileName);
   }
}
```

```
class AudioAdapter implements MediaPlayer {
   private AudioPlayer audioPlayer;

   public AudioAdapter() {
      audioPlayer = new AudioPlayer();
   }

   @Override
   public void play(String audioType, String fileName) {
      if (audioType.equalsIgnoreCase("mp3")) {
         audioPlayer.playAudio(fileName);
      } else {
         System.out.println("Invalid audio type. Only MP3 supported.");
      }
   }
}
class MediaPlayerClient {
   public static void main(String[] args) {
      MediaPlayer player = new AudioAdapter();
      player.play("mp3", "song.mp3");
      player.play("mp4", "movie.mp4");
   }
}
```

Result and Discussion:

- The Adapter pattern allows us to make incompatible systems work together by converting the interface of one class to be compatible with the client's expectations.
- This pattern is useful when we want to integrate existing code or third-party libraries without changing the original classes. Instead, an adapter class is created to translate method calls from one format to another.
- In this example, the AudioPlayer class could be an existing class that we cannot modify. By using the AudioAdapter, we can adapt it to the MediaPlayer interface without changing the AudioPlayer class itself.