

Project Report

Gunish 2017BCS0023

Adapter Design Pattern

Adapter pattern works as a bridge between two incompatible interfaces. This type of design pattern comes under structural pattern as this pattern combines the capability of two independent interfaces. This pattern involves a single class which is responsible to join functionalities of independent or incompatible interfaces. A real life example could be a case of card reader which acts as an adapter between memory card and a laptop. You plugin the memory card into card reader and card reader into the laptop so that memory card can be read via laptop.

Application with Problem:

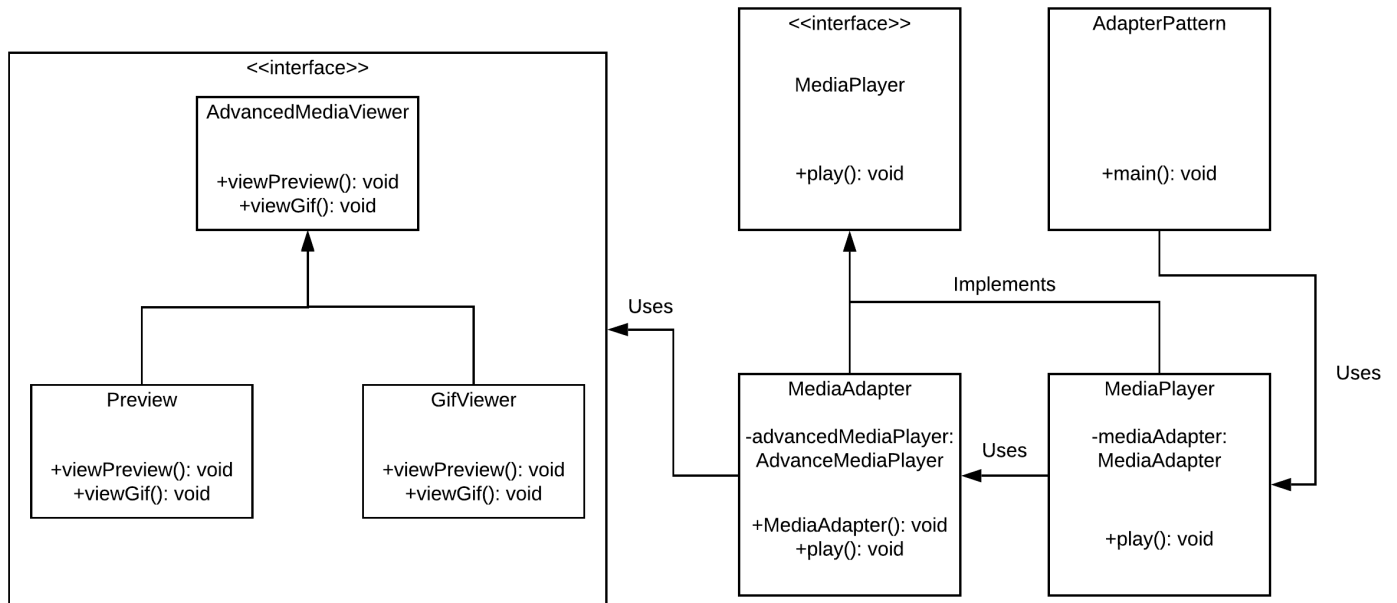
- An image viewer can view only .jpg files
- All Image viewers are not same.
- There are compatibility issues.

Objective

- 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.
- We are demonstrating use of Adapter pattern via following example in which an image viewer device can view .jpg files only and wants to use an advanced viewer capable of viewing .png and .gif files.

Design

UML Diagram



IMPLEMENTATION

We have an image viewer interface and a concrete class *MediaPlayererr* implementing the *MediaPlayer* interface. *MediaPlayererr* can view jpg format image files by default.

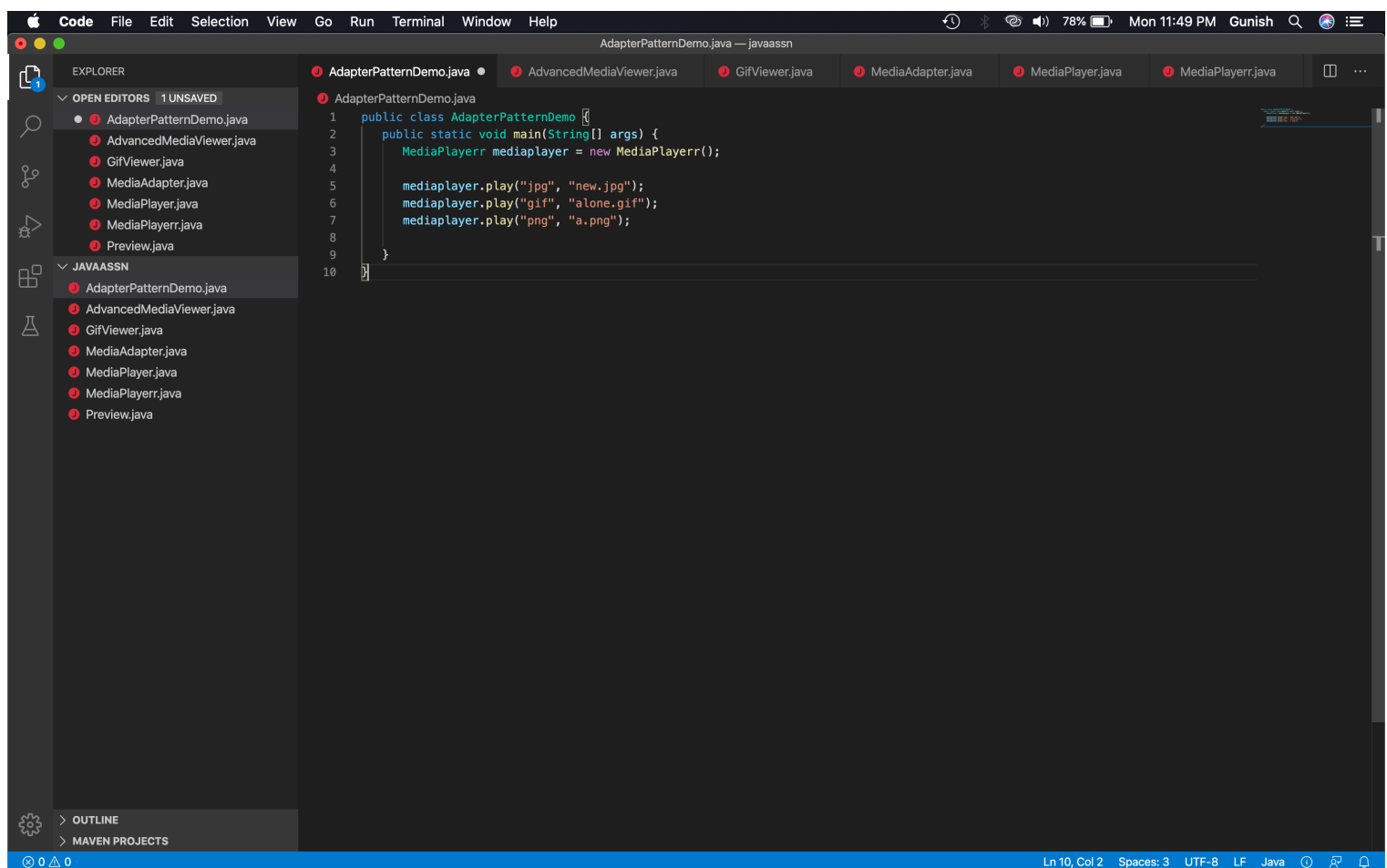
We are having another interface *AdvancedMediaPlayer* and concrete classes implementing the *AdvancedMediaPlayer* interface. These classes can view .gif and .gif format files

We want to make *MediaPlayer* to play other formats as well. To attain this, we have created an adapter class *MediaAdapter* which implements the *MediaPlayer* interface and uses *AdvancedMediaViewer* objects to view the required format

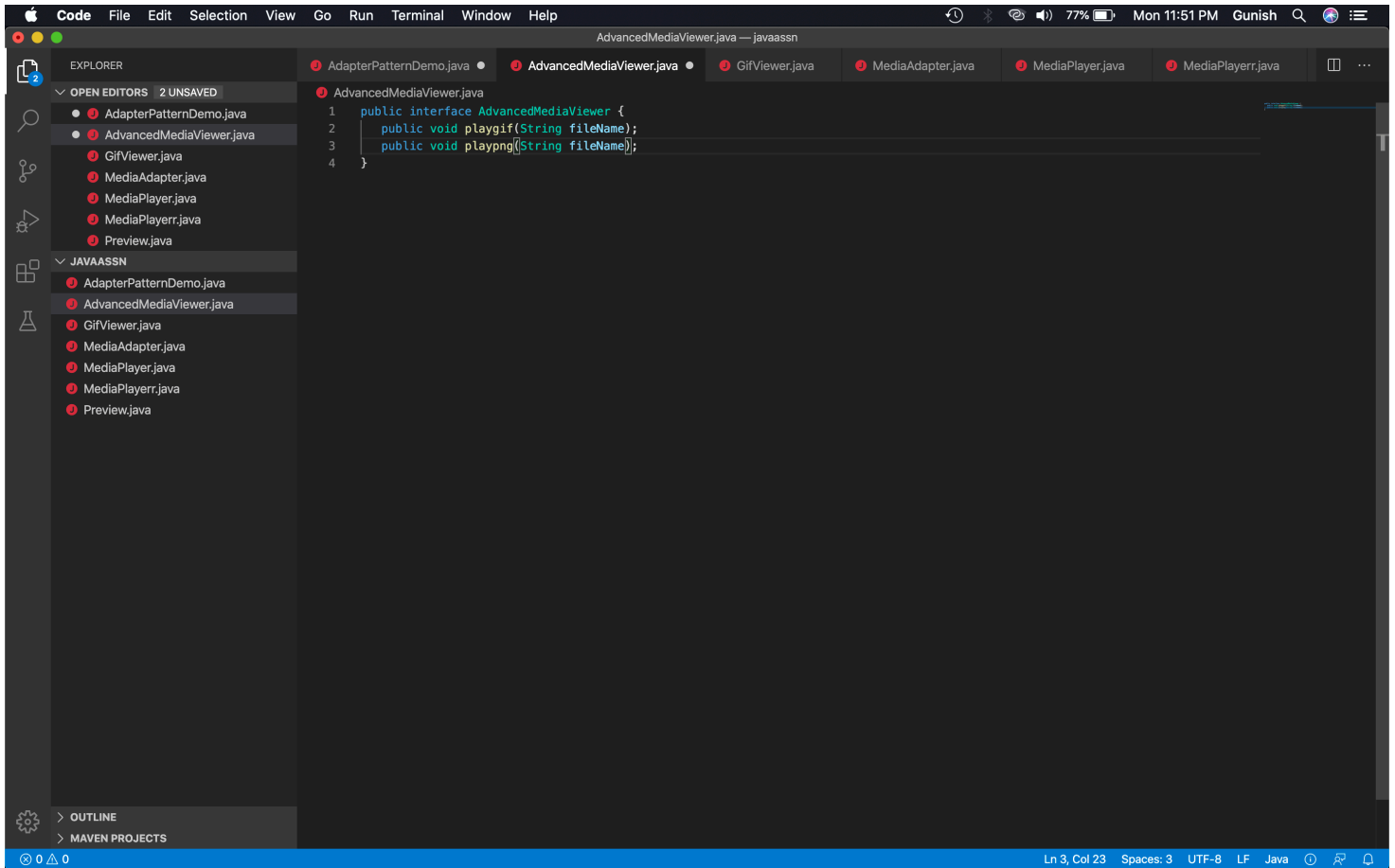
MediaPlayer uses the adapter class *MediaAdapter* passing it the desired image type without knowing the actual class which can view the desired format. *AdapterPatternDemo*, our demo class will use *MediaPlayer* class to view various formats.

CODE

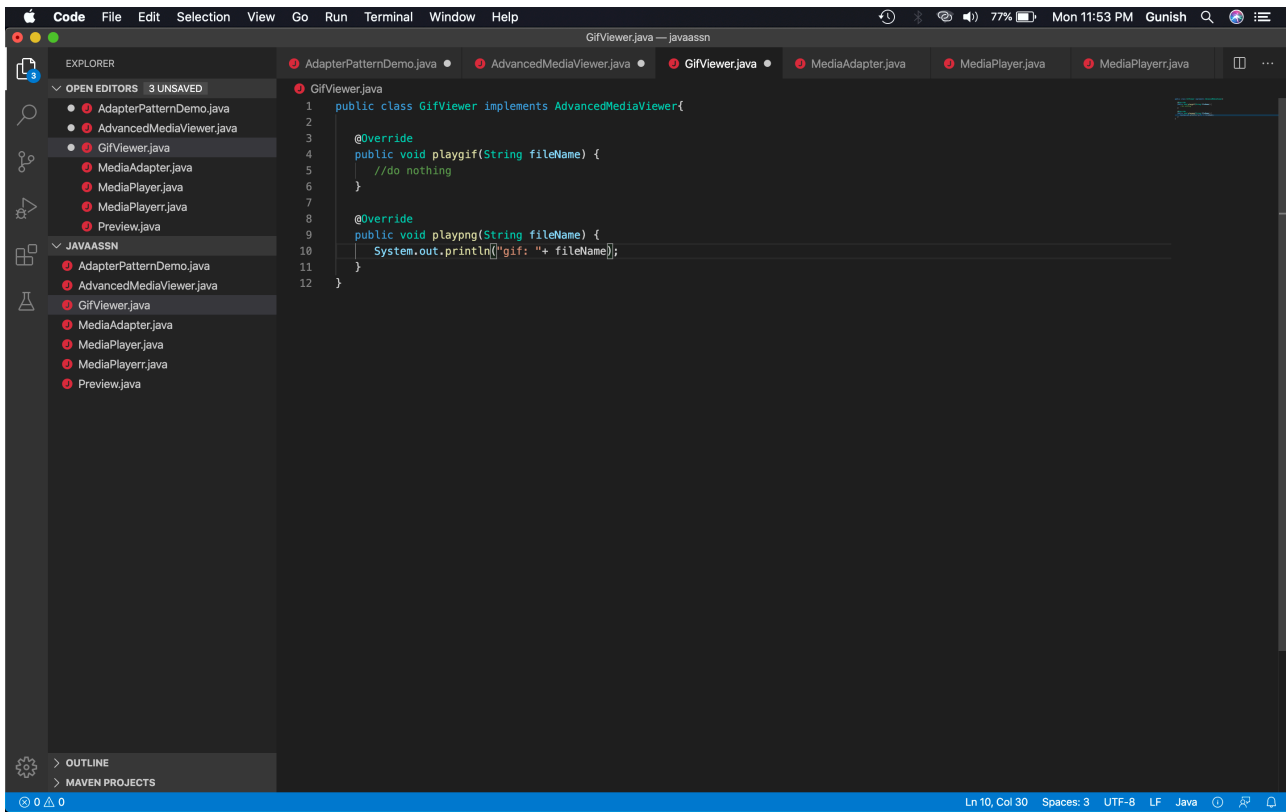
1. AdapterPatternDemo.java



2. AdvanceMediaPlayer.java

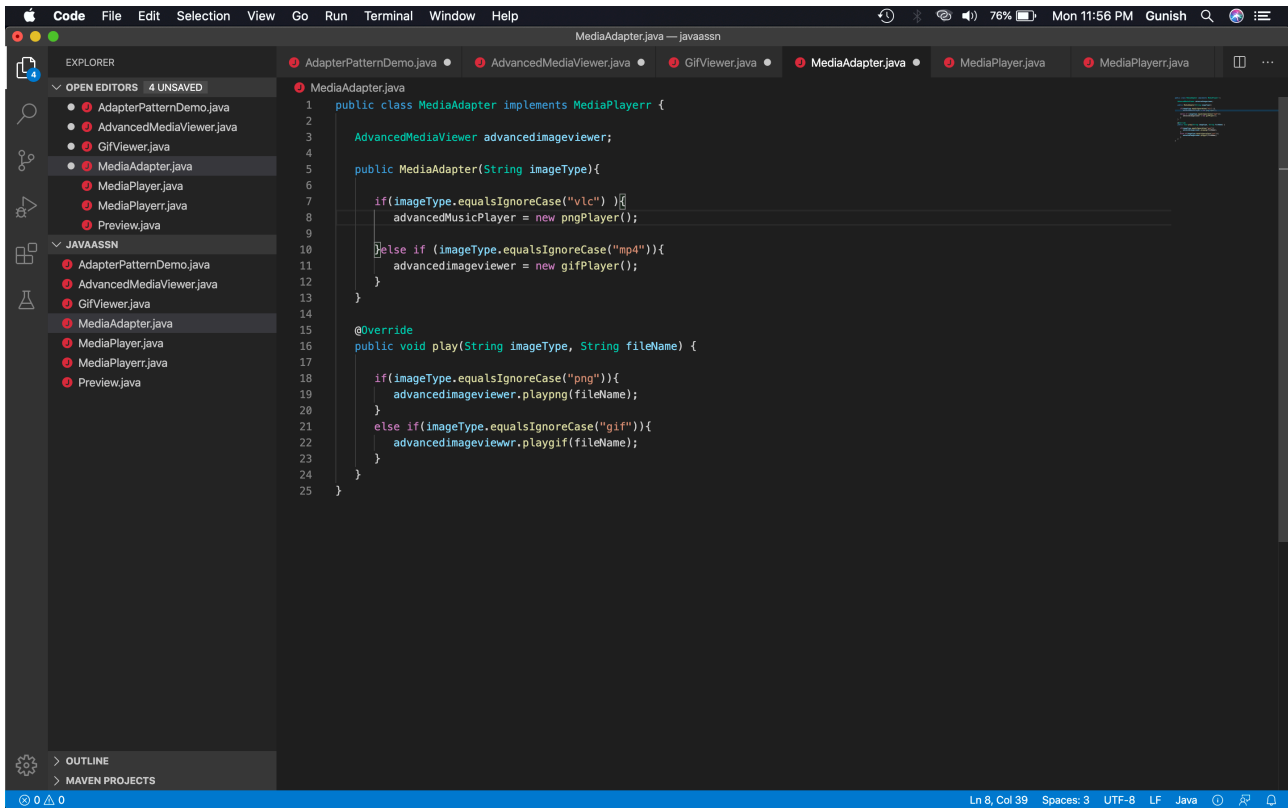


3. GifViewer.java



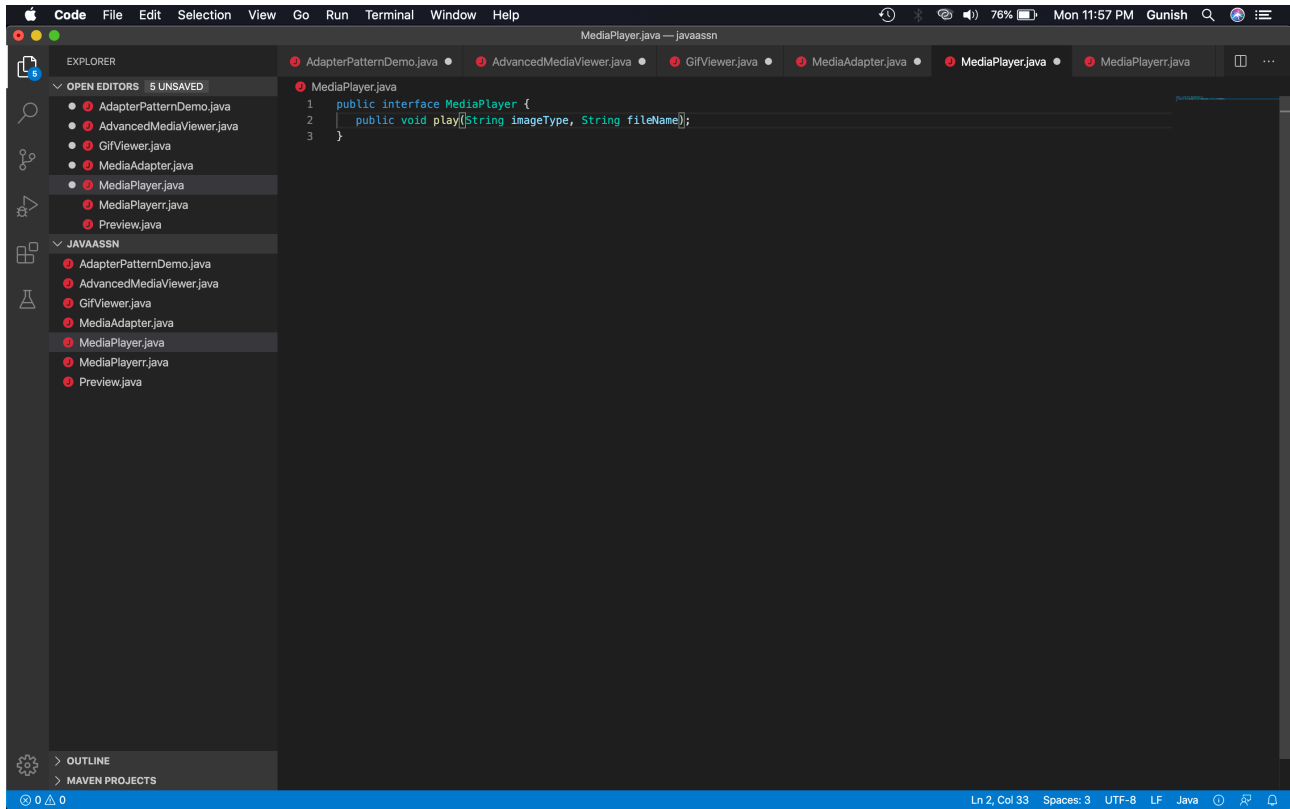
```
1 public class GifViewer implements AdvancedMediaViewer{
2
3
4     @Override
5     public void playgif(String fileName) {
6         //do nothing
7     }
8
9     @Override
10    public void playpng(String fileName) {
11        System.out.println("gif: " + fileName);
12    }
13 }
```

4.MediaAdapter.java

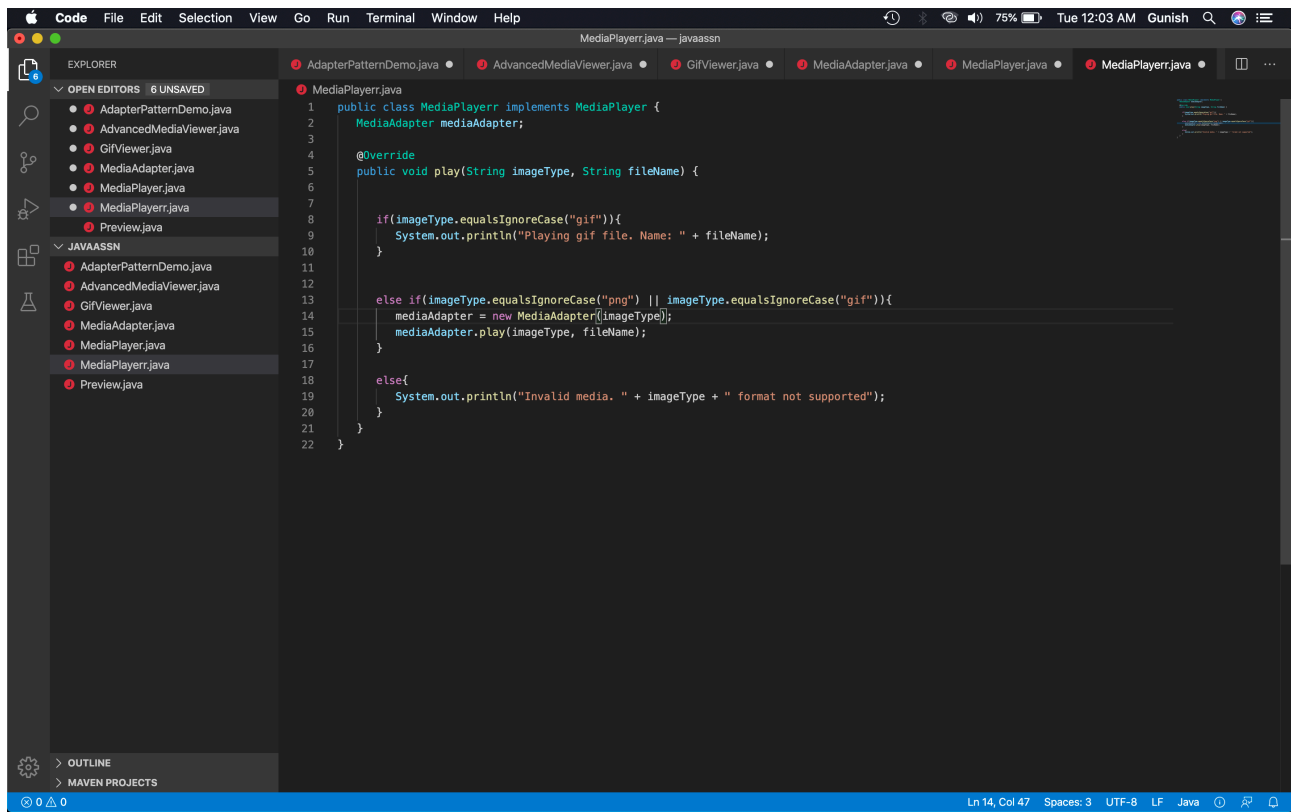


```
1 public class MediaAdapter implements MediaPlayerr {
2
3     AdvancedMediaViewer advancedImageViewer;
4
5     public MediaAdapter(String imageType){
6
7         if(imageType.equalsIgnoreCase("vlc")){
8             advancedMusicPlayer = new pngPlayer();
9         }
10        else if (imageType.equalsIgnoreCase("mp4")){
11            advancedimageViewr = new gifPlayer();
12        }
13    }
14
15    @Override
16    public void play(String imageType, String fileName) {
17
18        if(imageType.equalsIgnoreCase("png")){
19            advancedimageViewr.playpng(fileName);
20        }
21        else if (imageType.equalsIgnoreCase("gif")){
22            advancedimageViewwr.playgif(fileName);
23        }
24    }
25 }
```

5. MediaPlayer.java



6. MediaPlayer



```
1 public class MediaPlayer implements MediaPlayer {
2     MediaAdapter mediaAdapter;
3
4     @Override
5     public void play(String imageType, String fileName) {
6
7         if(imageType.equalsIgnoreCase("gif")){
8             System.out.println("Playing gif file. Name: " + fileName);
9         }
10
11         else if(imageType.equalsIgnoreCase("png") || imageType.equalsIgnoreCase("gif")){
12             mediaAdapter = new MediaAdapter(imageType);
13             mediaAdapter.play(imageType, fileName);
14         }
15
16         else{
17             System.out.println("Invalid media. " + imageType + " format not supported");
18         }
19     }
20 }
21
22 }
```