Media Handling in Android Applications

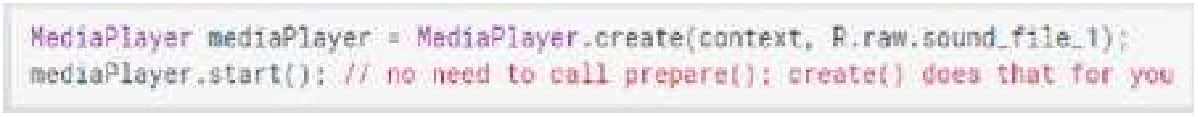
Media Player overview

- Android provide a multimedia frame work for images, videos and audio
- Here we can access content from resources from android file system or internet
- To do that we can use two android frameworks
    - MediaPlayer
        - This basically tell the android system to play the sound for an audio player and provide sounds and control the sounds of a playback
    - AudioManager
        - This manages audio sources and audio outputs of the device

Using MediaPlayer class

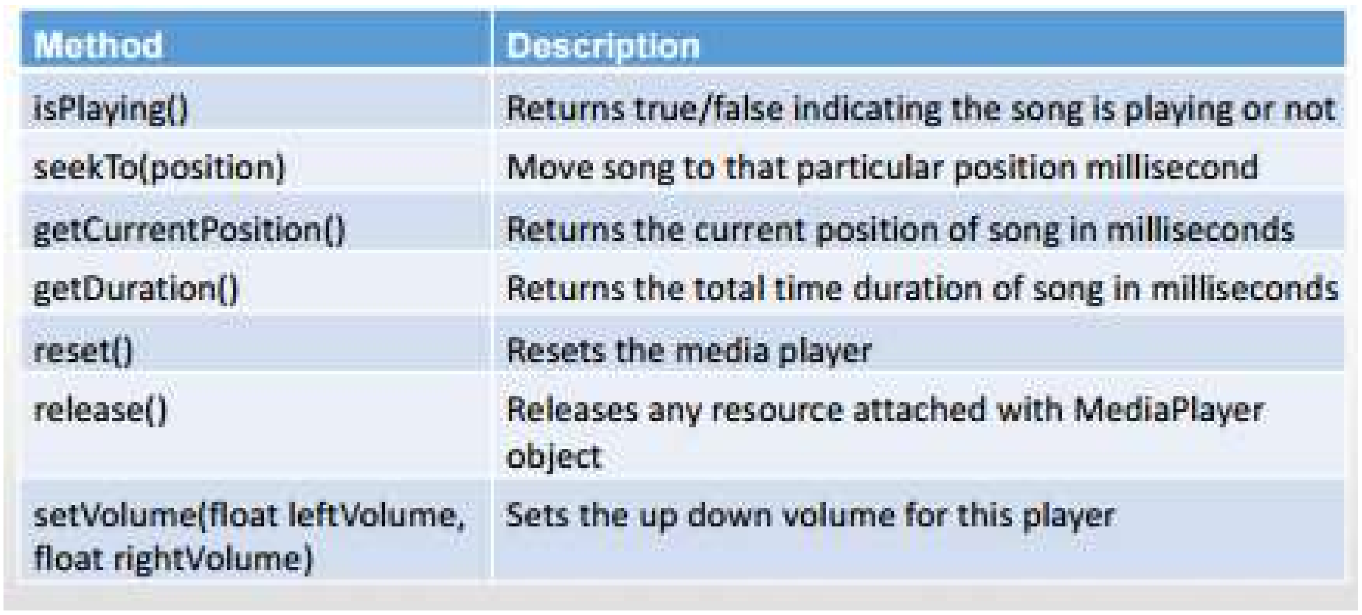
Below shows the code for the media player class

- Here we have call a static method call "create ()" there you have to pass the context and the file name as parameters



```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

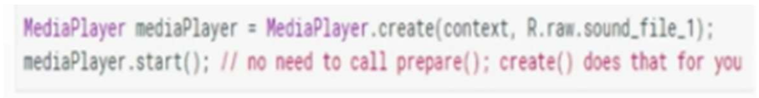
- We have to create a raw folder in res folder and in that folder we have to place the audio files
- After call this we can directly start the mediaPlayer using below command
- Below methods are useful to do several tasks



Method	Description
isPlaying()	Returns true/false indicating the song is playing or not
seekTo(position)	Move song to that particular position millisecond
getCurrentPosition()	Returns the current position of song in milliseconds
getDuration()	Returns the total time duration of song in milliseconds
reset()	Resets the media player
release()	Releases any resource attached with MediaPlayer object
setVolume(float leftVolume, float rightVolume)	Sets the up down volume for this player

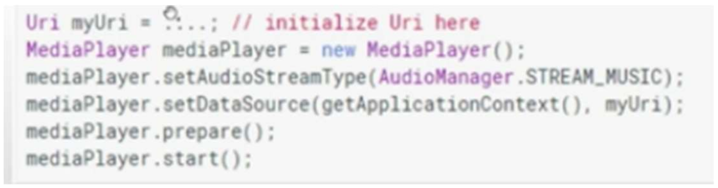
- MediaPlayer supports several different media sources such as:
  - Local resources
  - Internal URIs, such as one you might obtain from a Content Resolver
  - External URLs (streaming)

play audio using local recourses



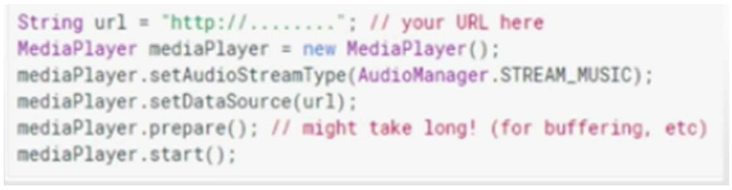
```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

play from a URI available locally in the system



```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

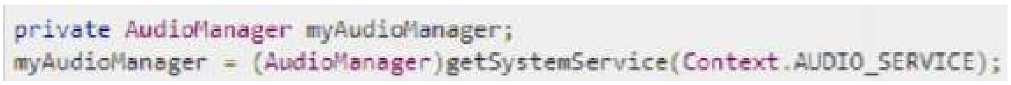
Playing from a remote URL via HTTP streaming



```
String url = "http://........"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

Audio manager

- Android provides AudioManager class that provides access to controls like ringer volume and ringer profile (silent, vibrate, loud)



```
private AudioManager myAudioManager;
myAudioManager = (AudioManager)getSystemService(Context.AUDIO_SERVICE);
```

Image Handling

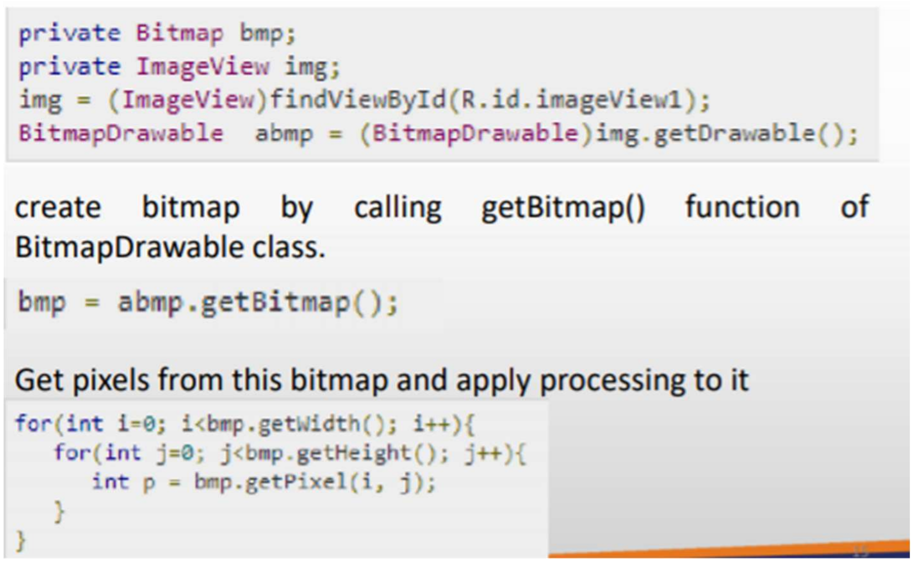
- When you need to display static images in your app, you can use the Drawable class

Create drawables from resource images

- Android provides Bitmap class to handle images. This can be found under android. graphics. Bitmap

- Supported file types are PNG (preferred), JPG (acceptable), and GIF (discouraged).

create a bitmap of image from the imageView



```
private Bitmap bmp;
private ImageView img;
img = (ImageView)findViewById(R.id.imageView1);
BitmapDrawable  abmp = (BitmapDrawable)img.getDrawable();
```

create bitmap by calling getBitmap() function of BitmapDrawable class.

```
bmp = abmp.getBitmap();
```

Get pixels from this bitmap and apply processing to it

```
for(int i=0; i<bmp.getWidth(); i++){
   for(int j=0; j<bmp.getHeight(); j++){
      int p = bmp.getPixel(i, j);
   }
}
```

Understand the Mobile Camera Handling Process

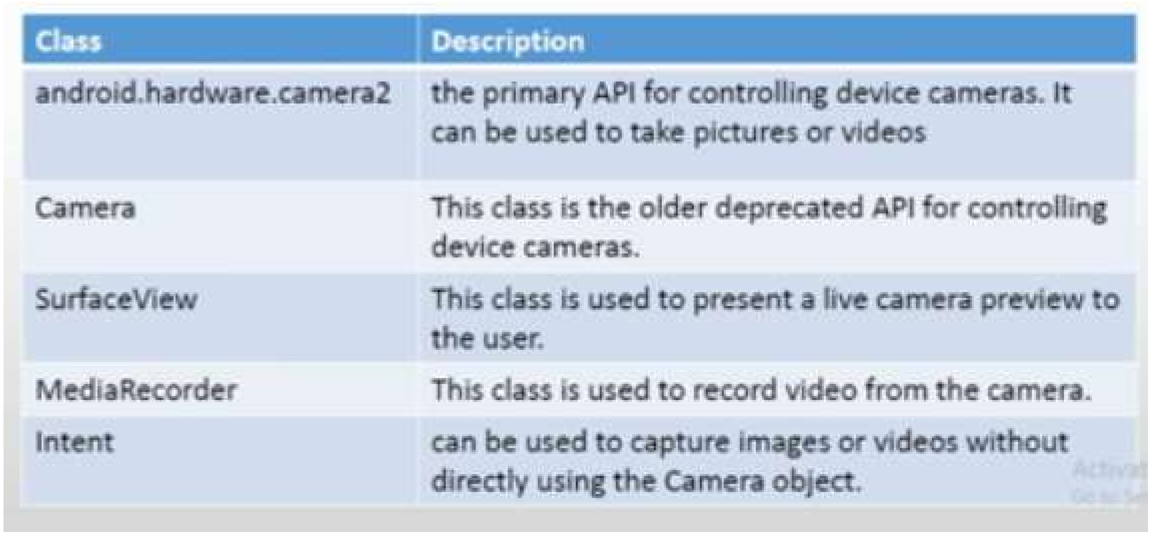
• Can use existing android camera application in your application

• Can directly using Camera API provided by android in application

Considerations

- Camera Requirement
  - We have to have a camera and we have to mention that in the manifest
  - If you are having an android device without a camera that applications have the capability of downloading those applications
- Quick Picture or customized camera
  - If your camera has only capability of capture what is in surrounding, you can get the picture and import that into the application and change

- Foreground service requirement
    - On Android 9 (API level 28) and later, apps running in the background cannot access the camera
- Storage



Class	Description
android.hardware.camera2	the primary API for controlling device cameras. It can be used to take pictures or videos
Camera	This class is the older deprecated API for controlling device cameras.
SurfaceView	This class is used to present a live camera preview to the user.
MediaRecorder	This class is used to record video from the camera.
Intent	can be used to capture images or videos without directly using the Camera object.

Manifest declaration

- Camera Permission Your application must request permission to use a device camera.
```
<uses-permission android:name="android.permission.CAMERA" />
```
- Camera Features
```
<uses-feature android:name="android.hardware.camera" />
```
- Storage Permission
    - If your application saves images or videos to the device's external storage (SD Card), you must also specify this in the manifest.
```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```
- Audio Recording Permission
    - For recording audio with video capture, your application must request the audio capture permission
```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```
.
- Location Permission
    - If your application tags images with GPS location information, you must request the ACCESS_FINE_LOCATION permission

Take a photo with a camera app

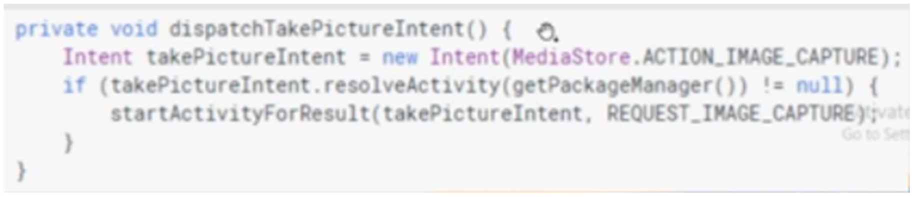
This process involves three pieces:

• The Intent

• A call to start the external Activity

• Some code to handle the image data when focus returns to your activity.

use MediaStore.ACTION_IMAGE_CAPTURE to launch an existing camera application installed on your phone



```
private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}
```

Building a camera app using camera Api

Steps for creating a custom camera interface for your application

• Detect and Access Camera

- Choose a camera in the device

• Create a Preview Class

- We have to see the preview

• Build a Preview Layout

- Here from preview class we take the frames/update the frames we have to display that as it is

• Setup Listeners for Capture

- We have to a design a button and from that button we have to capture the image

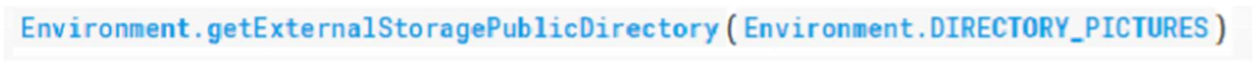
• Capture and Save Files

- In order to save the files, we have to have the storage permission

• Release the Camera

Save the full-size photo

First we take the external storage system after that we can store images in sd card



```
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES)
```

Camera features

• Can control with your camera application, such as picture format, flash mode, focus settings, and many more.

• Metering and focus areas

• Face detection

• Time lapse video