# CSE 162 Mobile Computing

Lab 3 MediaRecorder

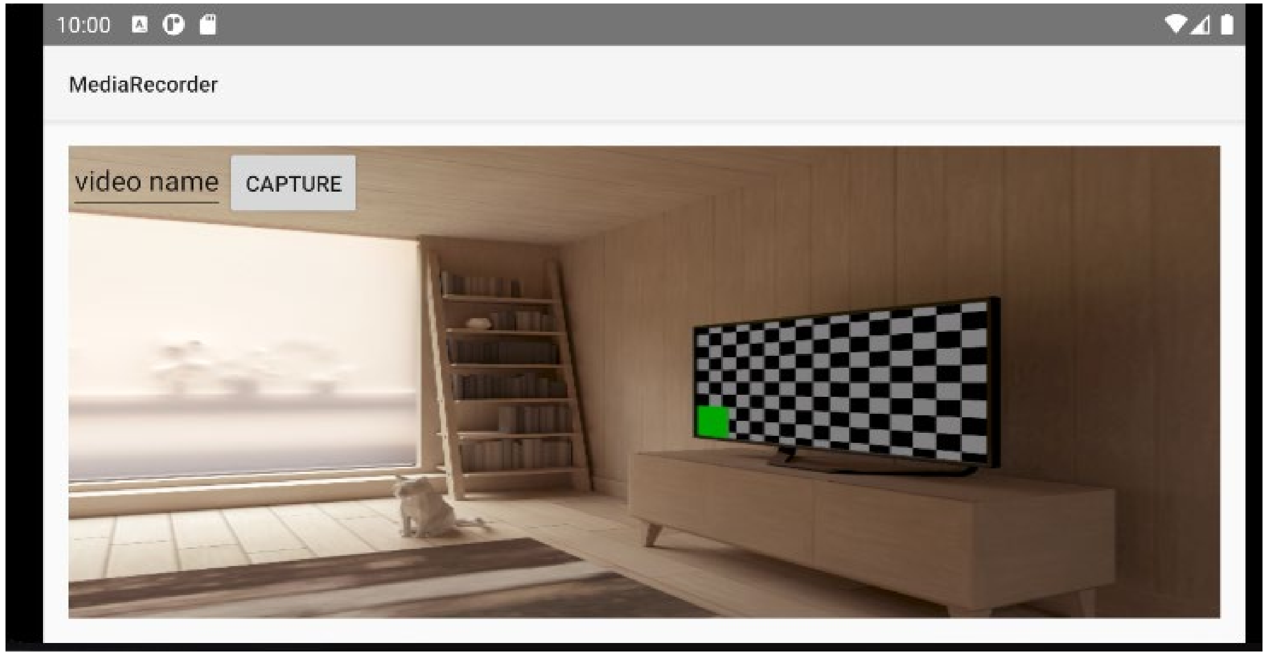
Department of Computer Science and Engineering
University of California, Merced, CA

## Goal: achieve the following features

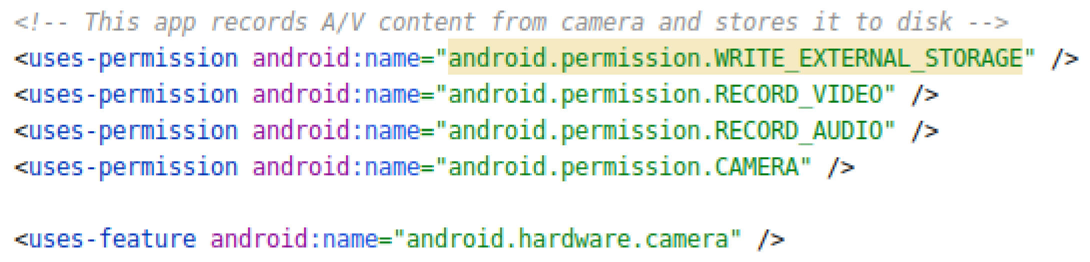
- Control the media recording capabilities
- Learn the MediaRecorder API
- Learn the Camera API

## Outline

- create an app to shoot video
- Push a button, the app begins to preview and record video
- Push the button again, save locally

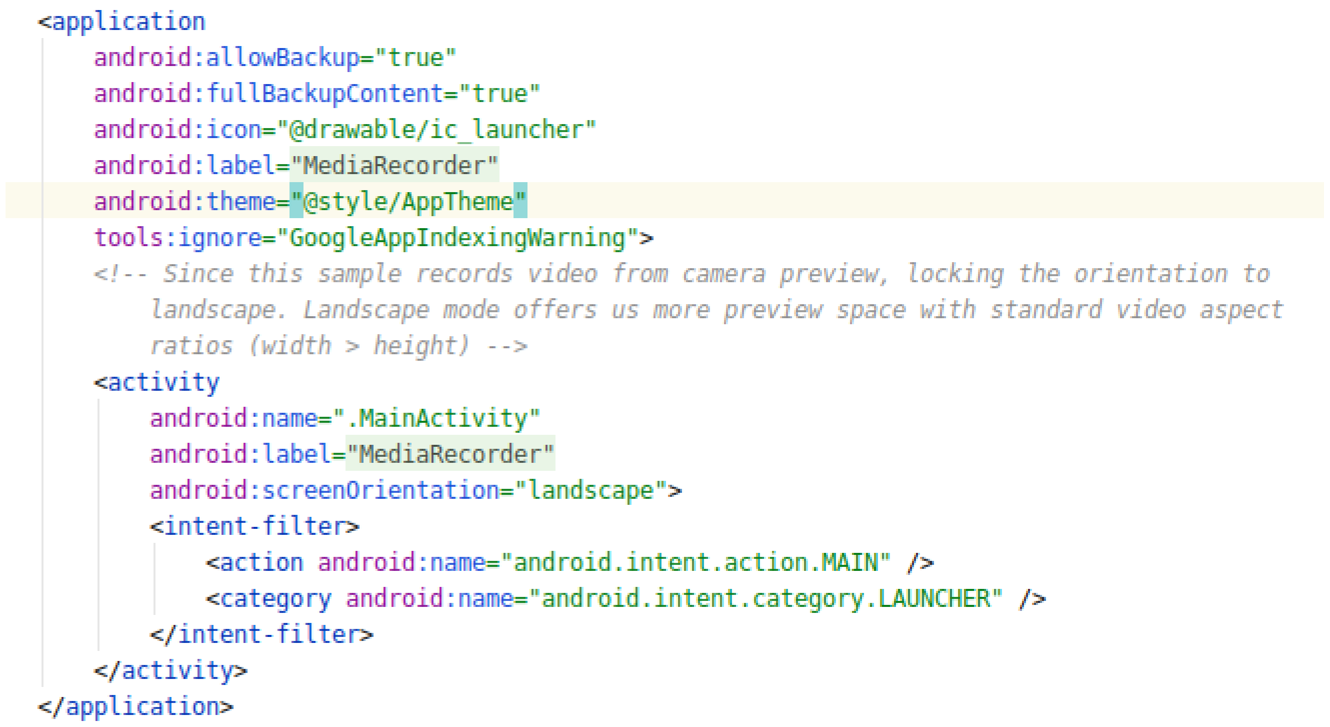


# permission in the manifest file



```xml
<!-- This app records A/V content from camera and stores it to disk -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_VIDEO" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.CAMERA" />

<uses-feature android:name="android.hardware.camera" />
```

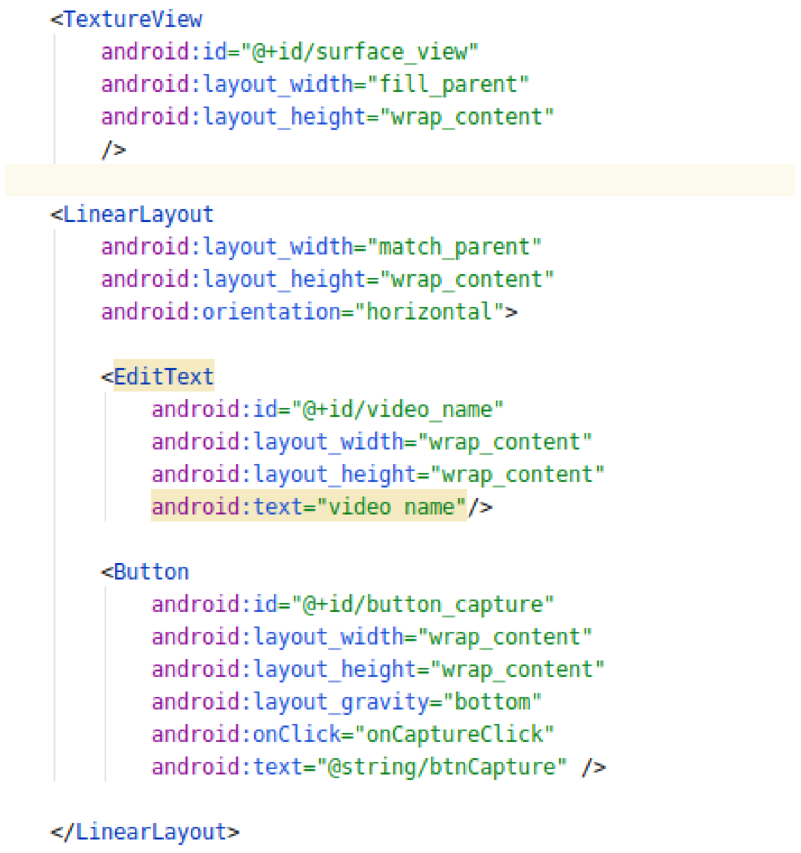
# other parts of the manifest



```
<application
    android:allowBackup="true"
    android:fullBackupContent="true"
    android:icon="@drawable/ic_launcher"
    android:label="MediaRecorder"
    android:theme="@style/AppTheme"
    tools:ignore="GoogleAppIndexingWarning">
    <!-- Since this sample records video from camera preview, locking the orientation to
        landscape. Landscape mode offers us more preview space with standard video aspect
        ratios (width > height) -->
    <activity
        android:name=".MainActivity"
        android:label="MediaRecorder"
        android:screenOrientation="landscape">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

## Prepare the UI

- in main_activity.xml, structured as follows
    - FrameLayout
        - Textureview
        - LinearLayout
            - EditText
            - Button

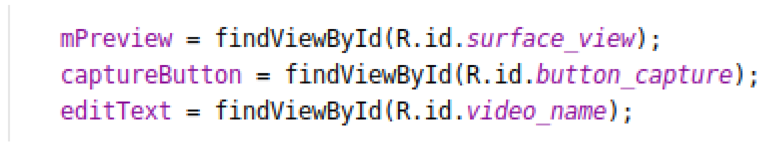


```xml
<TextureView
    android:id="@+id/surface_view"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <EditText
        android:id="@+id/video_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="video name"/>

    <Button
        android:id="@+id/button_capture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:onClick="onCaptureClick"
        android:text="@string/btnCapture" />

</LinearLayout>
```

## prepare the UI

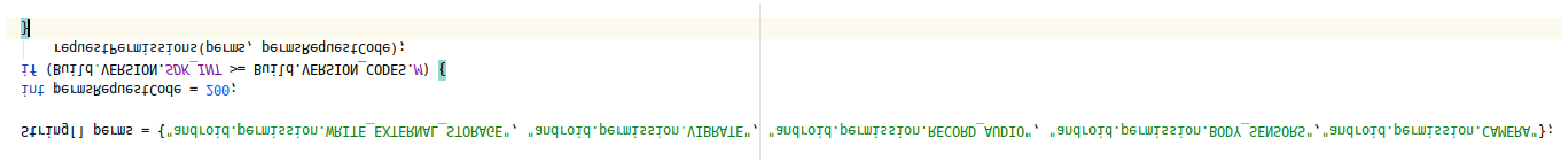
- Oncreate()

Obtain the views



```
mPreview = findViewById(R.id.surface_view);
captureButton = findViewById(R.id.button_capture);
editText = findViewById(R.id.video_name);
```

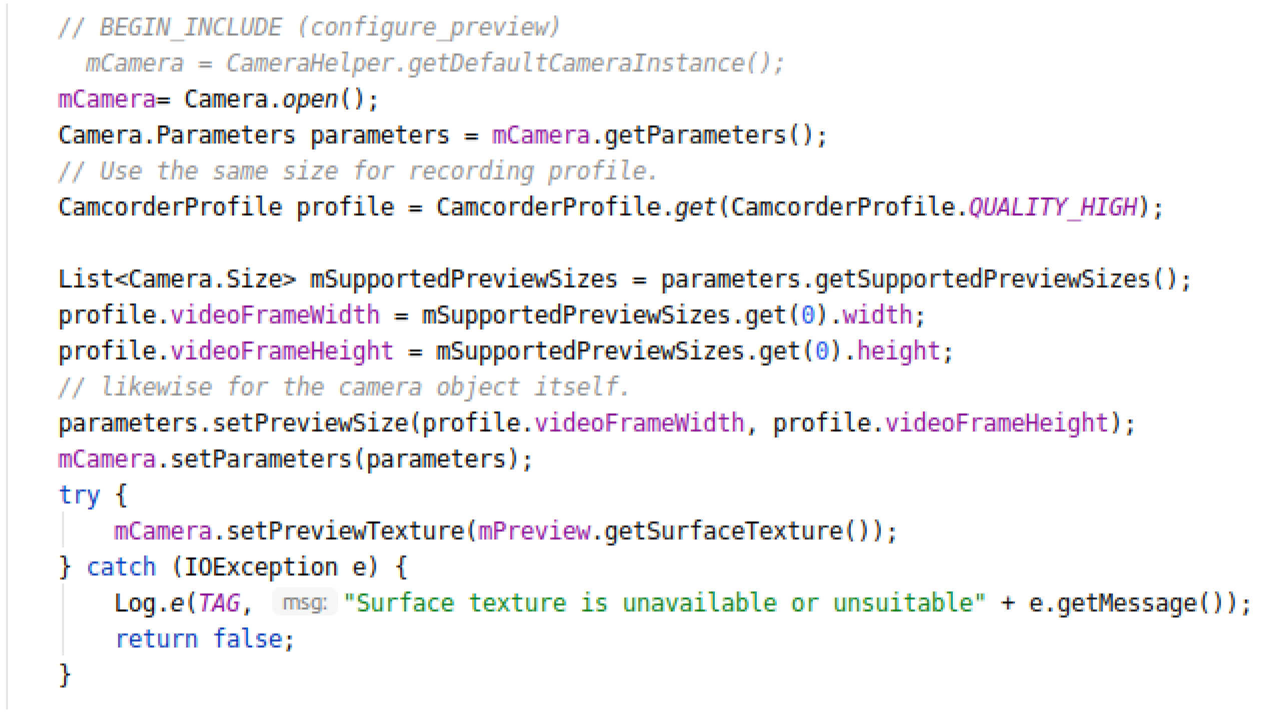
- obtain the permissions



```
String[] perms = {"android.permission.WRITE_EXTERNAL_STORAGE", "android.permission.VIBRATE", "android.permission.RECORD_AUDIO", "android.permission.BODY_SENSORS","android.permission.CAMERA"};

int permsRequestCode = 200;
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    requestPermissions(perms, permsRequestCode);
}
```

# prepare for video recording prepareVideoRecorder()

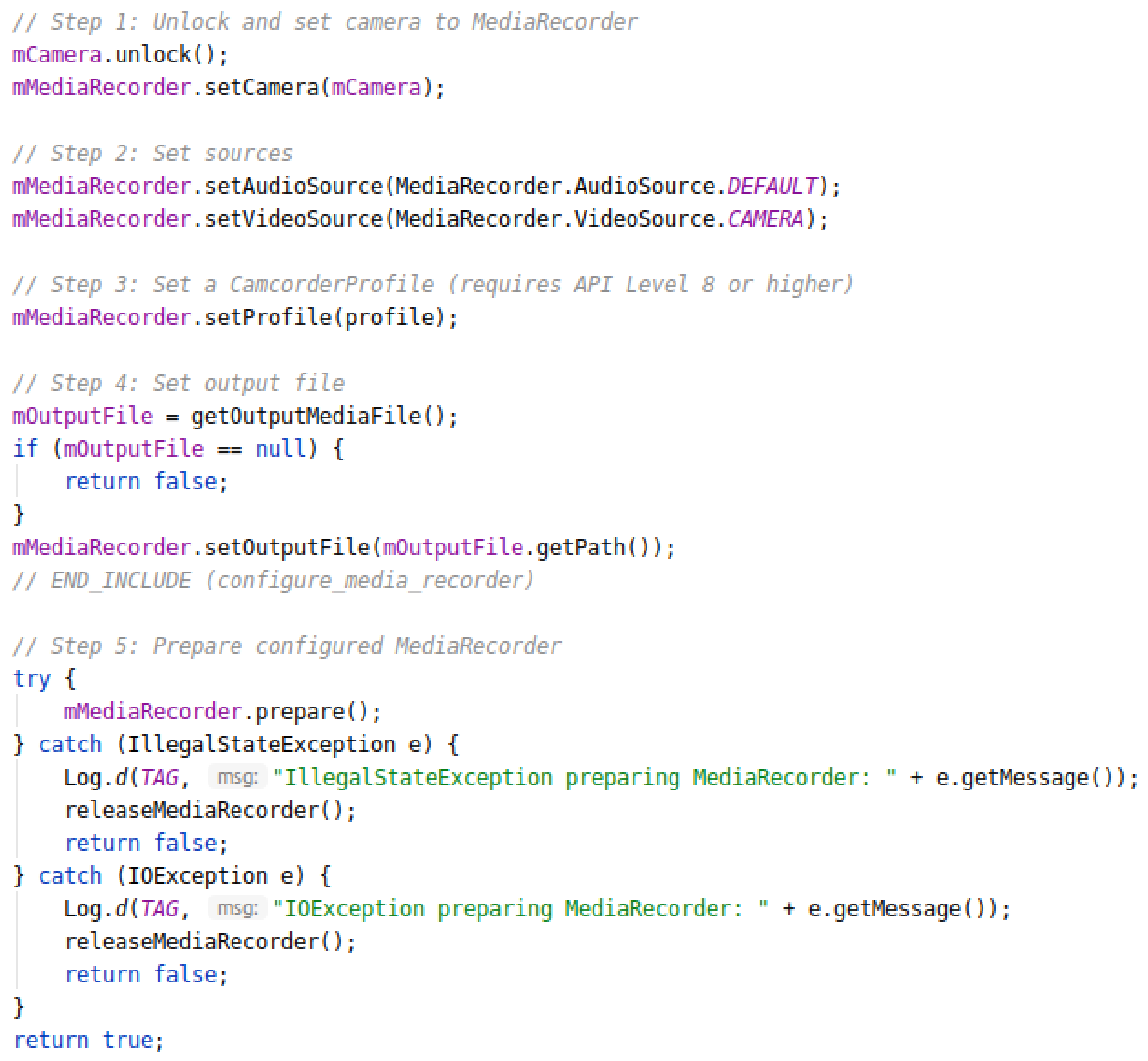
- Set the sizes of the video frame



```
// BEGIN_INCLUDE (configure_preview)
  mCamera = CameraHelper.getDefaultCameraInstance();
mCamera= Camera.open();
Camera.Parameters parameters = mCamera.getParameters();
// Use the same size for recording profile.
CamcorderProfile profile = CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH);

List<Camera.Size> mSupportedPreviewSizes = parameters.getSupportedPreviewSizes();
profile.videoFrameWidth = mSupportedPreviewSizes.get(0).width;
profile.videoFrameHeight = mSupportedPreviewSizes.get(0).height;
// likewise for the camera object itself.
parameters.setPreviewSize(profile.videoFrameWidth, profile.videoFrameHeight);
mCamera.setParameters(parameters);
try {
    mCamera.setPreviewTexture(mPreview.getSurfaceTexture());
} catch (IOException e) {
    Log.e(TAG, msg: "Surface texture is unavailable or unsuitable" + e.getMessage());
    return false;
}
```

# prepareVideoRecorder()

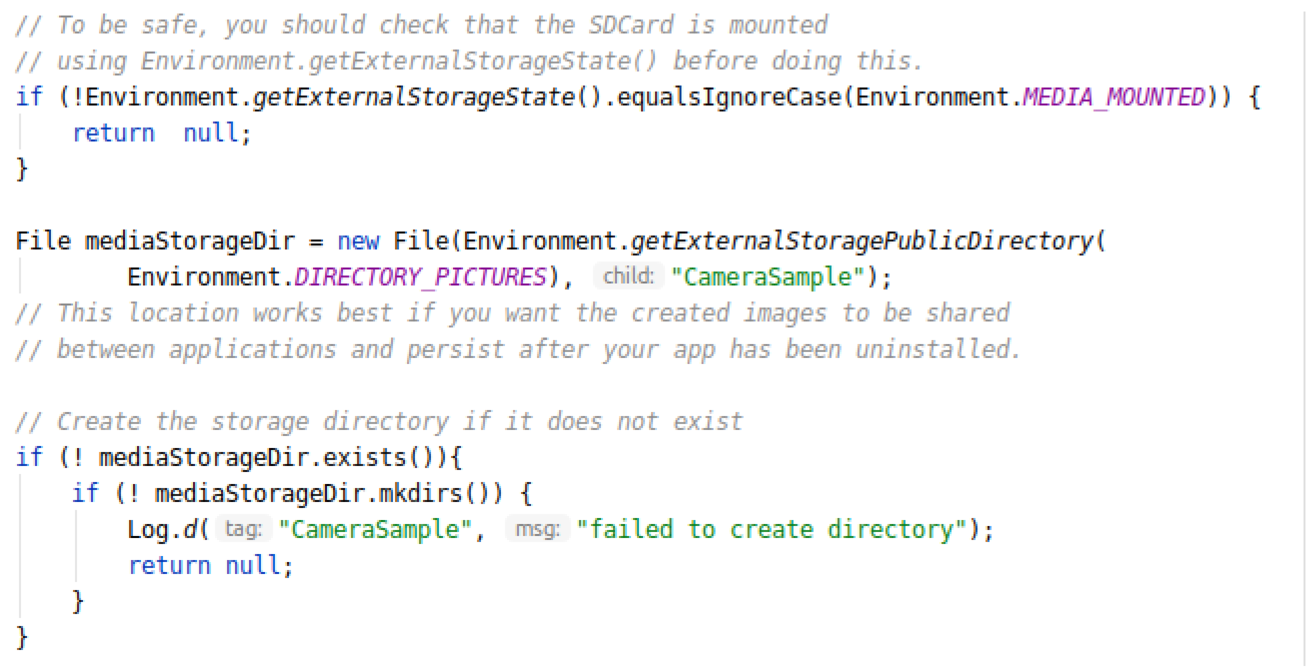
- configure the MediaRecorder



```
// Step 1: Unlock and set camera to MediaRecorder
mCamera.unlock();
mMediaRecorder.setCamera(mCamera);

// Step 2: Set sources
mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.DEFAULT);
mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);

// Step 3: Set a CamcorderProfile (requires API Level 8 or higher)
mMediaRecorder.setProfile(profile);

// Step 4: Set output file
mOutputFile = getOutputMediaFile();
if (mOutputFile == null) {
    return false;
}
mMediaRecorder.setOutputFile(mOutputFile.getPath());
// END_INCLUDE (configure_media_recorder)

// Step 5: Prepare configured MediaRecorder
try {
    mMediaRecorder.prepare();
} catch (IllegalStateException e) {
    Log.d(TAG, msg: "IllegalStateException preparing MediaRecorder: " + e.getMessage());
    releaseMediaRecorder();
    return false;
} catch (IOException e) {
    Log.d(TAG, msg: "IOException preparing MediaRecorder: " + e.getMessage());
    releaseMediaRecorder();
    return false;
}
return true;
```

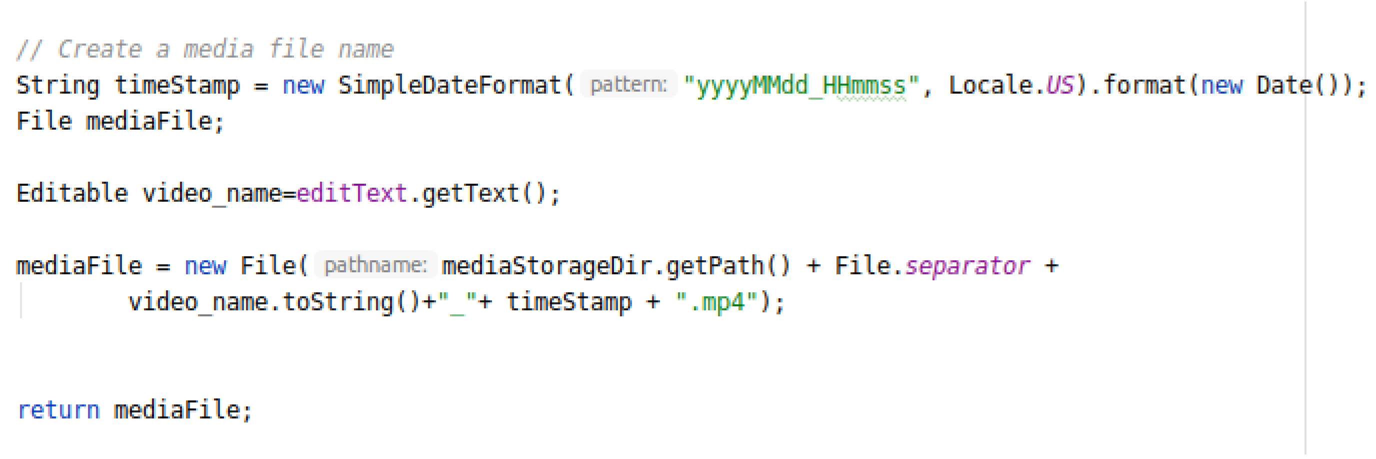
## Prepare for the file storage
## File getOutputMediaFile()

- permission, get the path, etc



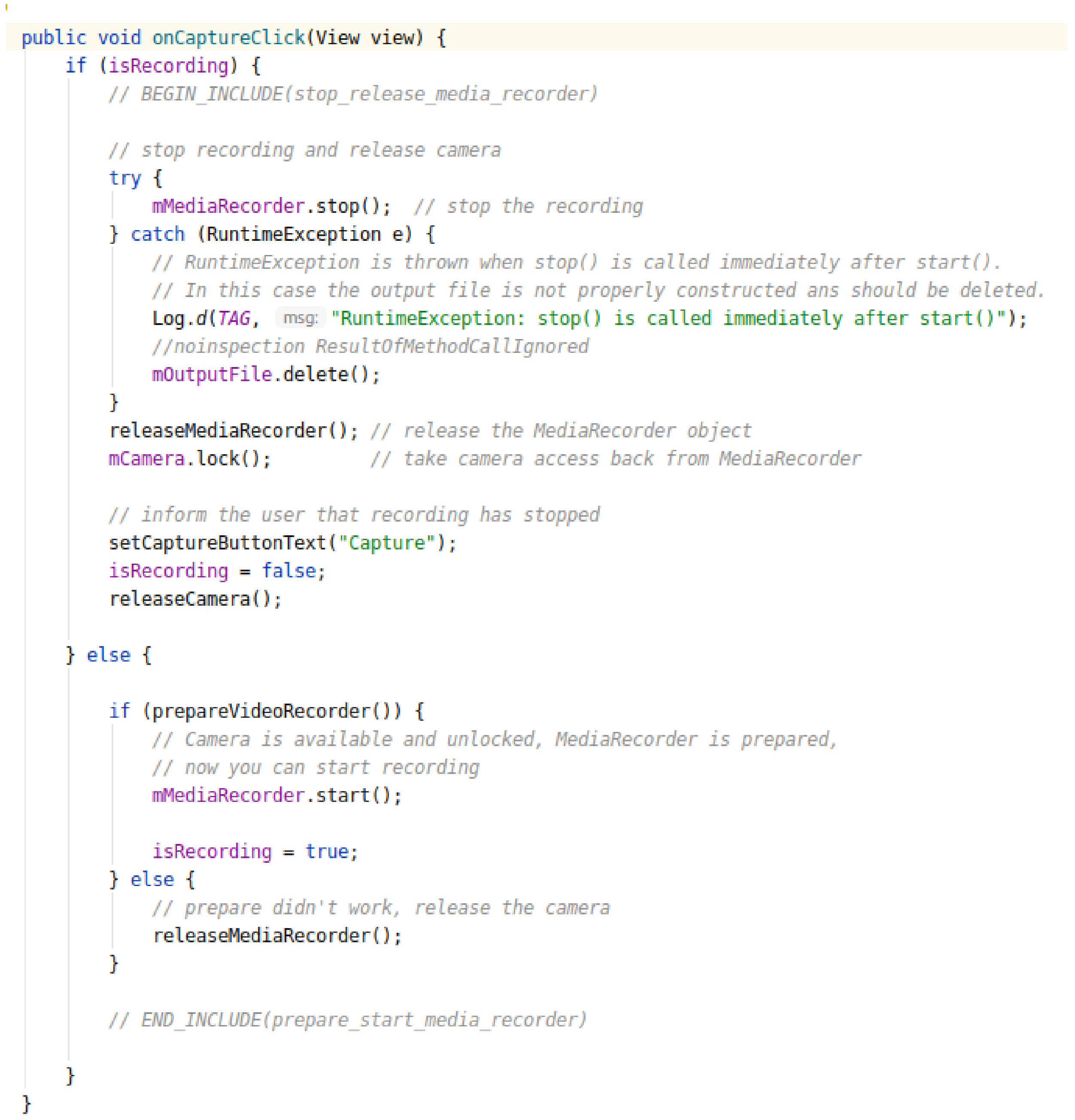
```
// To be safe, you should check that the SDCard is mounted
// using Environment.getExternalStorageState() before doing this.
if (!Environment.getExternalStorageState().equalsIgnoreCase(Environment.MEDIA_MOUNTED)) {
    return  null;
}

File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES),  child: "CameraSample");
// This location works best if you want the created images to be shared
// between applications and persist after your app has been uninstalled.

// Create the storage directory if it does not exist
if (! mediaStorageDir.exists()){
    if (! mediaStorageDir.mkdirs()) {
        Log.d( tag: "CameraSample",  msg: "failed to create directory");
        return null;
    }
}
```

## create the media file



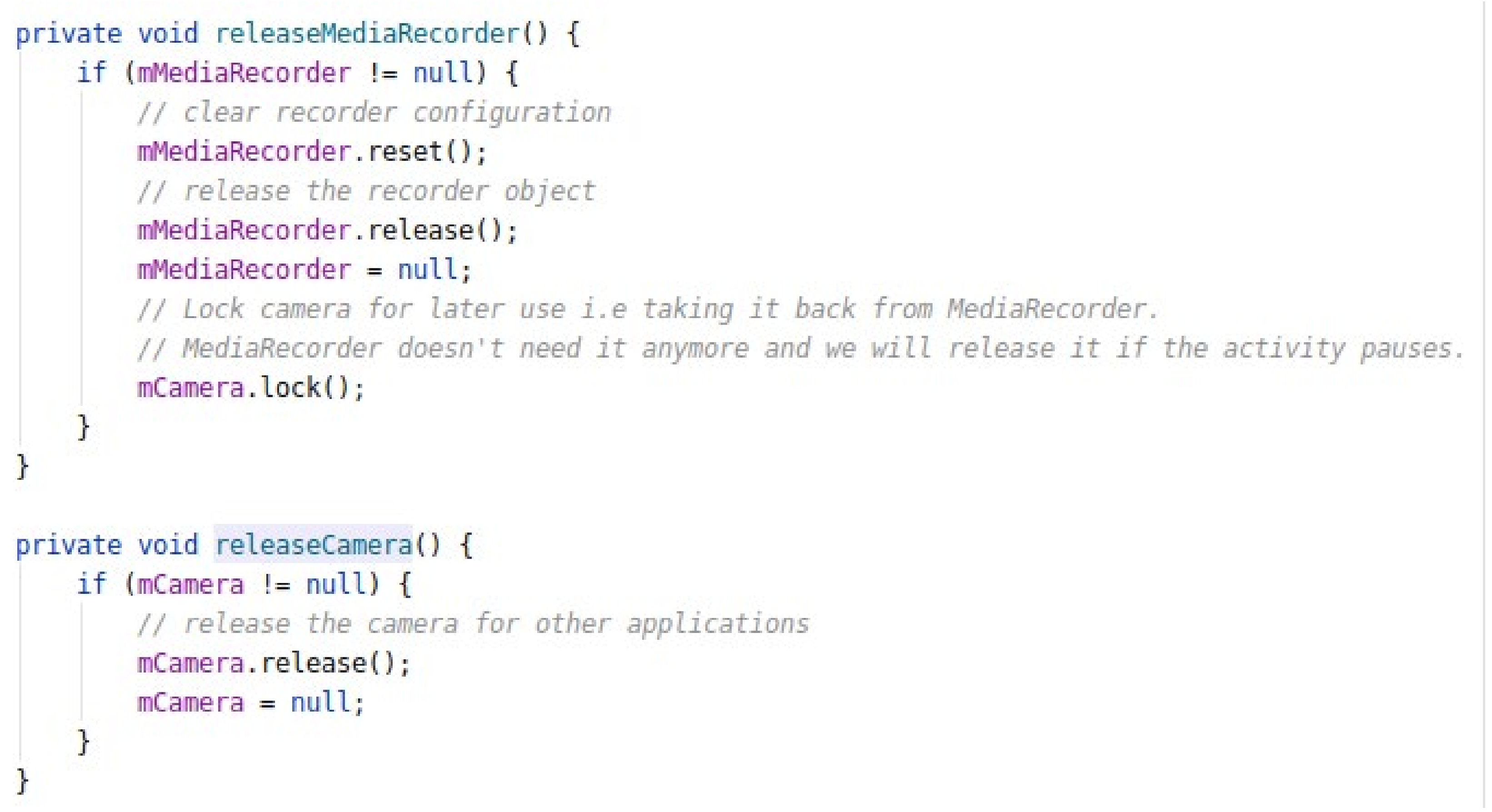
```
// Create a media file name
String timeStamp = new SimpleDateFormat( pattern: "yyyyMMdd_HHmmss", Locale.US).format(new Date());
File mediaFile;

Editable video_name=editText.getText();

mediaFile = new File( pathname: mediaStorageDir.getPath() + File.separator +
        video_name.toString()+"_"+ timeStamp + ".mp4");


return mediaFile;
```

## Use the media recorder

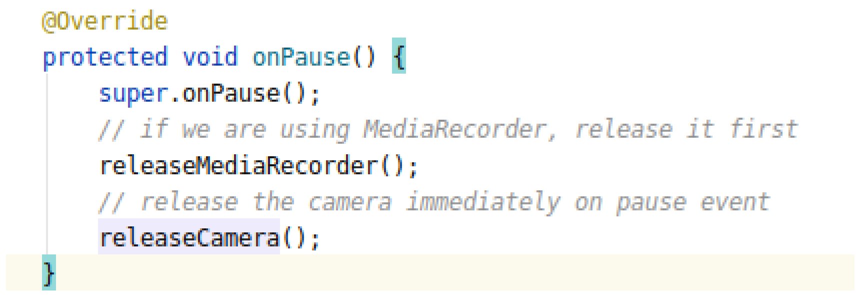


```java
public void onCaptureClick(View view) {
    if (isRecording) {
        // BEGIN_INCLUDE(stop_release_media_recorder)

        // stop recording and release camera
        try {
            mMediaRecorder.stop();  // stop the recording
        } catch (RuntimeException e) {
            // RuntimeException is thrown when stop() is called immediately after start().
            // In this case the output file is not properly constructed ans should be deleted.
            Log.d(TAG,  msg: "RuntimeException: stop() is called immediately after start()");
            //noinspection ResultOfMethodCallIgnored
            mOutputFile.delete();
        }
        releaseMediaRecorder(); // release the MediaRecorder object
        mCamera.lock();         // take camera access back from MediaRecorder

        // inform the user that recording has stopped
        setCaptureButtonText("Capture");
        isRecording = false;
        releaseCamera();

    } else {

        if (prepareVideoRecorder()) {
            // Camera is available and unlocked, MediaRecorder is prepared,
            // now you can start recording
            mMediaRecorder.start();

            isRecording = true;
        } else {
            // prepare didn't work, release the camera
            releaseMediaRecorder();
        }

        // END_INCLUDE(prepare_start_media_recorder)

    }
}
```

## When the recording stops, release the camera and the mediarecorder



```
private void releaseMediaRecorder() {
    if (mMediaRecorder != null) {
        // clear recorder configuration
        mMediaRecorder.reset();
        // release the recorder object
        mMediaRecorder.release();
        mMediaRecorder = null;
        // Lock camera for later use i.e taking it back from MediaRecorder.
        // MediaRecorder doesn't need it anymore and we will release it if the activity pauses.
        mCamera.lock();
    }
}

private void releaseCamera() {
    if (mCamera != null) {
        // release the camera for other applications
        mCamera.release();
        mCamera = null;
    }
}
```



```
@Override
protected void onPause() {
    super.onPause();
    // if we are using MediaRecorder, release it first
    releaseMediaRecorder();
    // release the camera immediately on pause event
    releaseCamera();
}
```

## Extra credit

- Add one more button on the UI. Click and play the most recently recorded video in the preview.