MODUL 9
AKSES INTERNAL DEVICE KAMERA

A. Pendahuluan

Kamera merupakan salah satu perangkat keras pada Android dan dapat diakses dalam pengembangan aplikasi mobile. Android menyediakan akses penuh ke perangkat keras kamera.

Kerangka kerja Android mencakup dukungan untuk berbagai kamera dan fitur kamera yang tersedia di perangkat, memungkinkan untuk mengambil gambar di aplikasi. Lembar kerja ini membahas tentang cara mengakses kamera dan mengambil gambar dari aplikasi kamera yang dibuat.

Catatan: modul ini menjelaskan kelas android.hardware.camera, yang sebenarnya sudah tidak digunakan lagi. Untuk kelas kamera yang lebih baru terdapat camera2 dan cameraX. Namun penggunaannya lebih sulit untuk dipelajari, oleh karena itu yang digunakan adalah kelas camera.

B. Persiapan

Sebelum mengaktifkan aplikasi untuk menggunakan kamera pada perangkat Android, perhatikan beberapa hal berikut :

1. Camera – Smartphone yang digunakan harus memiliki minimal 1 kamera yang dapat diakses.
2. Kelas Kamera yang digunakan, *android.hardware.*camera sudah kuno dan sudah tidak disupport oleh android (*deprecated*) oleh karena itu, tampilan pada android studio dicoret namun tetap dapat digunakan.

   ```
   private Camera mCamera = null;
   ```
3. Izin untuk menggunakan kamera dan penyimpanan harus disetujui agar aplikasi dapat digunakan.

C. Mengembangkan aplikasi kamera

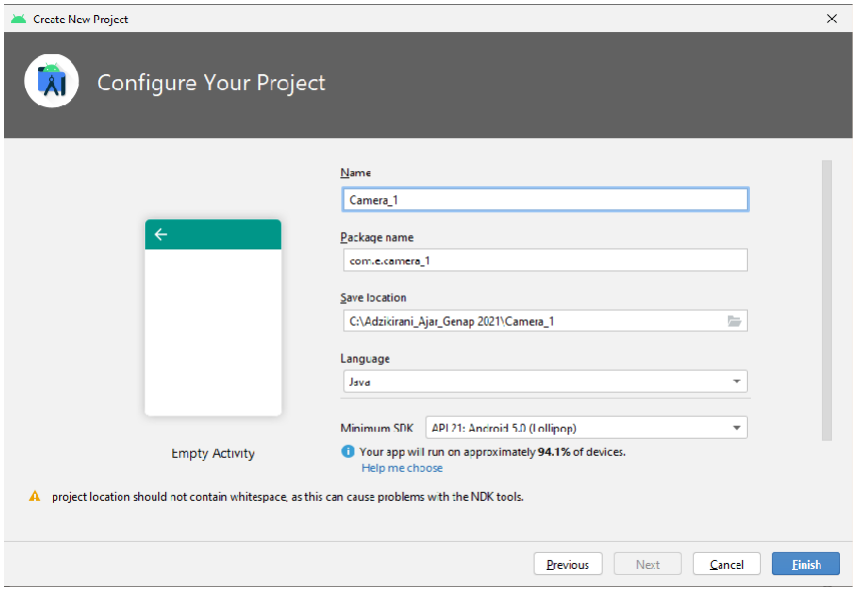
Langkah – Langkah umum untuk mengembangkan aplikasi kamera adalah sebagai berikut :

1. **Memeriksa & Meminta Izin**
2. **Mendeteksi dan Mengakses Kamera** - Buat kode untuk meminta akses dan memeriksa keberadaan kamera.
3. **Membuat *Class* Preview** - Buat *class* preview kamera yang memperluas SurfaceView dan mengimplementasikan antarmuka SurfaceHolder. *Class* ini mempreview gambar langsung dari kamera.
4. **Membuat Tata Letak Preview** - Setelah memiliki *class* preview kamera, buat tata letak tampilan yang menggabungkan preview dan kontrol antarmuka pengguna yang diiinginkan.
5. **Menyiapkan listener untuk Pengambilan Foto** - Hubungkan listener untuk kontrol antarmuka guna memulai pengambilan gambar atau video sebagai respons terhadap tindakan pengguna, seperti menekan tombol.
6. **Mengambil dan Menyimpan File** - Siapkan kode untuk mengambil gambar atau video dan menyimpan hasilnya.
7. **Melepas Kamera** - Setelah menggunakan kamera, aplikasi harus dirilis dengan benar untuk digunakan oleh aplikasi lain.

Hardware kamera adalah sumber daya bersama yang harus dikelola dengan hati-hati agar aplikasi tidak bertabrakan dengan aplikasi lain yang mungkin juga ingin menggunakannya. Bagian berikut membahas cara mendeteksi hardware kamera, cara meminta akses ke kamera, cara mengambil gambar atau video, dan cara melepas kamera setelah aplikasi selesai menggunakannya.

Perhatian: Ingatlah untuk melepas objek Camera dengan memanggil Camera.release() setelah aplikasi selesai menggunakannya! Jika aplikasi tidak merilis kamera dengan benar, semua upaya berikutnya untuk mengakses kamera, termasuk yang dilakukan oleh aplikasi itu sendiri, akan gagal dan dapat menyebabkan aplikasi atau lainnya error.

1. Buat New Project, Beri nama Camera_1, Gunakan minimum API 21 keatas (21+) agar dapat menggunakan semua fitur



Gambar 9.1 Konfigurasi Project Baru

2. Menambahkan Izin

   Langkah pertama yang perlu dilakukan adalah menambahkan izin untuk penggunaan kamera, dan izin untuk melakukan penyimpanan gambar atau video ke dalam perangkat pada file AndroidManifest.xml

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.Camera" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Gambar 9.2 Izin pada AndroidManifest.xml

Sehingga tampilan AndroidManifest.xml Menjadi seperti berikut

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.e.camera_1">
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.Camera" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Camera_1"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Camera_1">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Gambar 9.3 Tampilan AndroidManifest.xml setelah ditambahkan izin

3. Memeriksa Izin

Agar mendapatkan izin untuk menggunakan kamera, pada saat aplikasi pertama kali di buka, perlu ditambahkan permintaan izin untuk penggunaan kamera dan pengaksesan storage (menyimpan file).

Tambahkan sebuah method boolean untuk meminta izin dengan nama cekIzinDitolak() pada file MainActivity.Java

```java
private boolean cekIzinDitolak(){
    for(int i = 0; i< PERMISSIONS.length; i++){
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if(checkSelfPermission(PERMISSIONS[i]) != PackageManager.PERMISSION_GRANTED){
                return true;
            }
        }
    }
    return false;
}
```

Gambar 9.4 Pengecekan Izin dalam program

Method **cekIzinDitolak**() akan mengembalikan nilai berupa true jika izin belum di berikan, dan false jika sudah. dan *PERMISSION* adalah sebuah Variabel String Array yang berisi izin apa saja yang mau diminta.

```
private  static final  String[] PERMISSIONS = {
        Manifest.permission.CAMERA,
        Manifest.permission.WRITE_EXTERNAL_STORAGE
};
```

Gambar 9.5 Kode Permissions untuk mengecek izin

Dimana izin yang akan diminta yaitu untuk penggunaan camera dan penyimpanan.
Panggil callBack onResume, dan tambahkan code berikut untuk menggunakan method yang sudah di buat sebelumnya.

```
@Override
protected void onResume() {
    super.onResume();
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.M && cekIzinDitolak()){
        requestPermissions(PERMISSIONS, REQUEST_CODE);
        return;
    }
}
```

Gambar 9.6 Kode untuk method callback **onResume**

Pada saat callback onResume() dipanggil, aplikasi akan melakukan pengecekan izin, dan jika izin belum diberikan, akan memanggil fungsi requestPermissions() yang akan meminta izin kepada pengguna. Dan REQUEST_CODE adalah variable int yang berisi code yang digunakan sebagai code untuk melakukan request.

```
private static final int REQUEST_CODE = 1;
```

4. Mengakses kamera

Untuk mengakses camera, langkah pertama yang perlu dilakukan yaitu mendapatkan objek Camera, Untuk itu fungsi Camera.open() perlu dipanggil. Dan untuk memudahkan penggunaan, buat sebuah method Camera dengan nama getCamera()

```
public Camera getCamera(){
    Camera c = null;
    try {
        c = Camera.open(); // attempt to get a Camera instance
    }
    catch (Exception e){
        Toast.makeText(this, "Kamera tidak tersedia", Toast.LENGTH_SHORT).show();
    }
    return c; // returns null if camera is unavailable
}
```

Gambar 9.7 Fungsi Untuk memanggil kamera pertama kali

Memanggil Camera.open() akan memunculkan error/exception jika kamera sudah digunakan oleh aplikasi lain, Oleh karena itu ditambahkan dalam blok try-catch dan akan menampilkan sebuah Toast jika kamera tidak dapat diakses.

Keterangan : Fungsi Camera.open() dapat diberi sebuah parameter untuk memilih kamera berupa integer jika kamera yang tersedia lebih dari satu.
Contoh : Camera.open(1); //untuk menggunakan kamera depan
Untuk memanggil fungsi getCamera() panggil fungsi getCamera pada callback OnResume dan masukan pada sebuah variabel Camera dengan nama mCamera

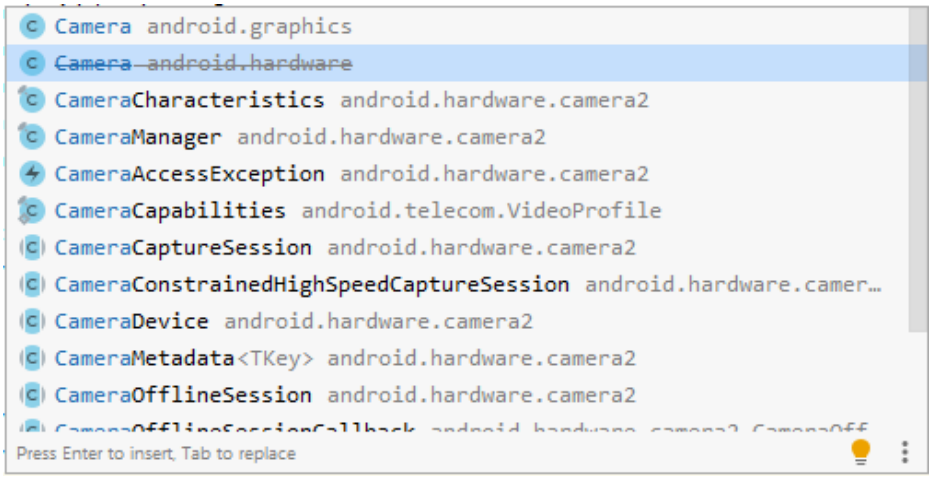
```
@Override
protected void onResume() {
    super.onResume();
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.M && cekIzinDitolak()){
        requestPermissions(PERMISSIONS, REQUEST_CODE);
        return;
    }
    mCamera = getCamera();
}
```

Gambar 9.8 Fungsi getCamera()

Variabel mCamera adalah sebuah variable Camera yang diisi dengan null untuk sementara.

```
private Camera mCamera = null;
```

Perlu diingat bahwa setiap pemanggilan variable Camera, yang digunakan(di import) adalah **android.hardware.Camera**



Gambar 9.9 Import library android.hardware.camera

```
import android.hardware.Camera;
```

5. Membuat Preview Kamera

Agar pengguna dapat mengambil gambar atau video secara efektif, mereka harus dapat melihat apa yang dilihat kamera perangkat. *Class* preview kamera adalah **SurfaceView** yang dapat menampilkan data gambar langsung yang berasal dari kamera, sehingga pengguna dapat membingkai dan mengambil gambar atau video.

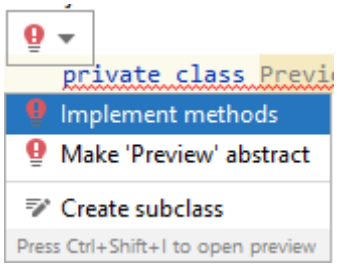
6. *Class* Preview

Untuk mulai menampilkan preview, diperlukan *class* Preview. Preview memerlukan implementasi **android.view.SurfaceHolder.Callback**, yang digunakan untuk membuat dan menutup tampilan kamera.

Tambahkan *Class* Preview berikut pada mainActivity sebagai local *Class*.

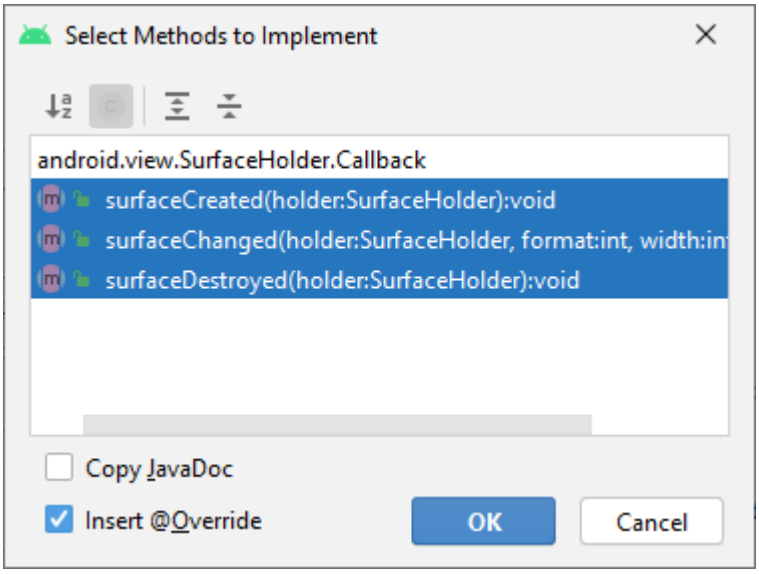
```
private class Preview extends SurfaceView implements SurfaceHolder.Callback {

}
```

Class yang dibuat akan dikenali sebagai error karena *class* yang mengimplementasikan SurfaceHolder.Callback wajib memiliki implementasi method surfaceCreated(), surfaceChanged(), dan surfaceDestroyed()
Untuk memengimplementasikan method tersebut, klik lampu merah pada bagian yang error / klik alt+enter



Gambar 9.10 Implement method (1)

Lalu pilih implement methods, pilih ketiga method lalu klik OK



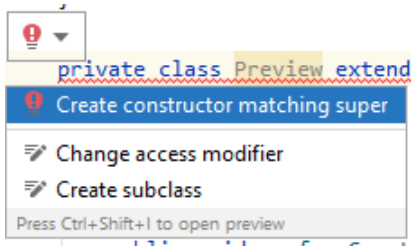
Gambar 9.11 Implement method (2)

Maka ketiga method tersebut akan secara otomatis diimplementasikan

```
@Override
public void surfaceCreated(@NonNull SurfaceHolder holder) {

}

@Override
public void surfaceChanged(@NonNull SurfaceHolder holder, int format, int width, int height) {

}

@Override
public void surfaceDestroyed(@NonNull SurfaceHolder holder) {

}
```

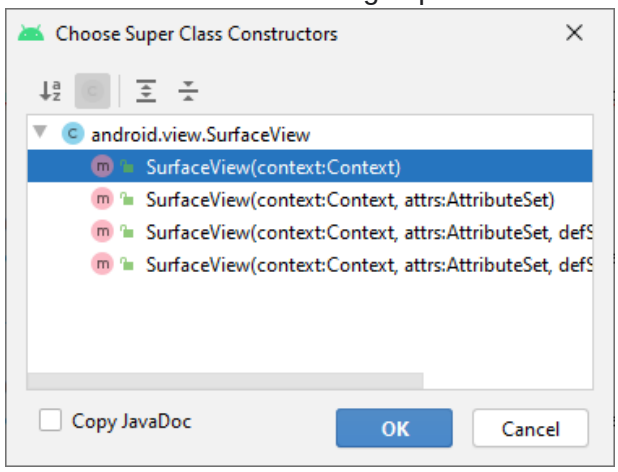
Gambar 9.12 Implement method (3)

Kemudian tambahkan lagi method constructor dengan menekan tombol lampu merah sekali lagi



Gambar 9.13 Create Constructor (1)

Pilih Create constructor matching super



Gambar 9.14 Create Constructor (2)

Dan pilih pilihan pertama lalu klik tombol Ok, dan ubah method Constructor yang dibuat dengan menambahkan parameter Camera,

```
public Preview(Context context, Camera camera) {
    super(context);
}
```

Dan isikan code berikut untuk menambahkan SurfaceHolder dan Kamera

```
SurfaceHolder mHolder;
private Camera PrevCamera;
public Preview(Context context, Camera camera) {
    super(context);
    PrevCamera = camera;
    mHolder = getHolder();
    mHolder.addCallback(this);
    mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
}
```

Gambar 9.15 Menambahkan SurfaceHolder Camera

Dimana surfaceHolder digunakan untuk memanggil fungsi ketika preview di buat ,dirubah atau ditutup.
Lalu pada method surfaceCreated, tambahkan code berikut untuk mengatur tampilan camera.

```
@Override
public void surfaceCreated(@NonNull SurfaceHolder holder) {
    try {
        mCamera.setPreviewDisplay(holder);
        mCamera.startPreview();
        mCamera.setDisplayOrientation(90);
    } catch (Exception e) {
        Toast.makeText(MainActivity.this, "Failed to open Camera "+e, Toast.LENGTH_LONG ).show();
    }
}
```

Gambar 9.16 Method SurfaceCreated

Dimana Fungsi :

- setPreviewDisplay digunakan untuk menentukan tampilan preview
- startPreview untuk menampilkan tampilan preview Camera
- dan setDisplayOrientation digunakan untuk mengatur orientasi kamera

Fungsi diletakan pada blok try-catch agar ketika camera tidak dapat digunakan aplikasi akan menampilkan toast pemberitahuan.

Method surfaceChanged digunakan untuk mengatur jika preview camera yang diinginkan dapat berubah tampilan atau berubah orientasi nya (rotate). Untuk sekarang biarkan kosong.

```
@Override
public void surfaceChanged(@NonNull SurfaceHolder holder, int format, int width, int height) {
    //Digunakan untuk mengatur tampilan Camera
}
```

Method surfaceDestroyed akan dipanggil ketika tampilan preview ditutup, dan diisi dengan camera.Release() atau menutup Camera agar dapat digunakan pada aplikasi lain.

```
@Override
public void surfaceDestroyed(@NonNull SurfaceHolder holder) {
    if (mCamera != null) {
        mCamera.stopPreview();
        mCamera.release();
        mCamera = null;
    }
}
```

Gambar 9.17 Method SurfaceDestroyed

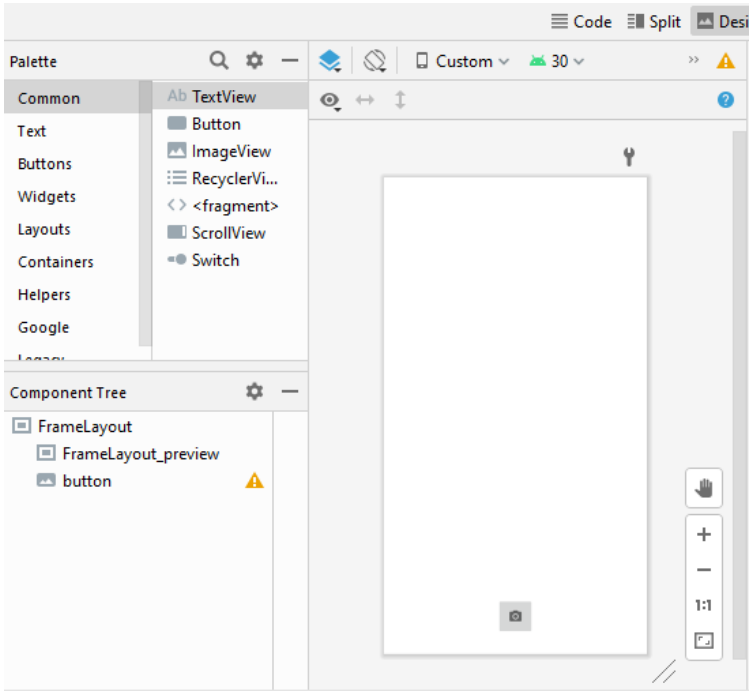
Class Preview yang telah dibuat sebelumnya, harus diletakan dalam suatu komponen View yang nantinya digunakan sebagai tempat untuk mengatur tampilan kamera. Dan dalam contoh ini, komponen yang digunakan adalah FrameLayout, yang digunakan sebagai Container untuk kelas preview kamera.

```
<FrameLayout
    android:id="@+id/FrameLayout_preview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Dan tambahkan sebuah Imagebutton yang nantinya digunakan untuk shutter camera.

```
<ImageButton
    android:id="@+id/button"
    android:layout_width="60dp"
    android:layout_height="60dp"
    android:src="@drawable/ic_baseline_camera_alt_24"
    android:layout_gravity="bottom|center_horizontal"
    android:layout_marginBottom="30dp"/>
```

Dimana android:src diisi dengan gambar camera yang perlu ditambahkan.



Gambar 9.18 Tampilan Awal FrameLayout untuk Camera

Lalu pada fungsi onResume, tambahkan *class* preview ke dalam komponen FrameLayout yang telah dibuat sebelumnya

```
preview = new Preview(this,mCamera);
frameLayoutpreview.addView(preview);
```

Dimana preview adalah variable objek dari local *class* preview yang telah dibuat sebelumnya.

```
private static  Preview preview;
```

Dan frameLayoutPreview adalah komponen frameLayout

```
private FrameLayout frameLayoutpreview;
```

```
frameLayoutpreview = findViewById(R.id.FrameLayout_preview);
```

Sehingga method onResume menjadi seperti berikut

```
@Override
protected void onResume() {
    super.onResume();
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.M && cekIzinDitolak()){
        requestPermissions(PERMISSIONS, REQUEST_CODE);
        return;
    }
    mCamera = getCamera();
    preview = new Preview(this,mCamera);
    frameLayoutpreview.addView(preview);

}
```

Gambar 9.19 Method OnResume

Coba Jalankan dan lihat apakah camera telah dapat digunakan.

7. Mengambil Foto

Setelah membuat *class* preview dan tata letak tampilan untuk menampilkannya, Aplikasi siap digunakan untuk mengambil gambar. Untuk mengambil gambar, gunakan metode Camera.takePicture(), metode ini mengambil tiga parameter ayng menerima data dari kamera. Untuk menerima data dalam format JPEG, harus diimplementasikan antarmuka Camera.PictureCallback untuk menyimpan gambar yang diterima dari kamera.

Sebelum mengatur method PictureCallback, tambahkan sebuah method File getOutputMediaFile() yang berfungsi

untuk menentukan letak penyimpanan dan nama file yang akan dibuat. Output dari method ini adalah sebuah file image yang masih kosong.

```
private File getOutputMediaFile(){
    //Menentukan Lokasi penyimpanan Foto pada folder Pictures
    String path = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES).getPath()+"/Aplikasi Kamera ";
    File mediaStorageDir = new File(path);
    // Membuat folder baru jika belum ada
    if (! mediaStorageDir.exists()){
        if (! mediaStorageDir.mkdirs()){
            Toast.makeText(this, "failed to create directory", Toast.LENGTH_SHORT).show();
            return null;
        }
    }
    //Scan file agar file yang dibuat dideteksi oleh aplikasi lain (Misal : agar foto yang diambil terlihat di Galerry
    sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, Uri.parse("file://"+path)));
    // Menentukan nama file foto yang dibuat
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    File mediaFile = new File(mediaStorageDir.getPath() + File.separator +
            "IMG_"+ timeStamp + ".jpg");

    return mediaFile;
}
```

Gambar 9.20 Fungsi getOutputMediaFile

Tambahkan code berikut pada fungsi onResume untuk memanggil method **PictureCallback**()
Dimana mPicture merumapan variable dari objek Camera.PictureCallback

```
private Camera.PictureCallback mPicture ;
```

```
mPicture = new Camera.PictureCallback() {
    @Override
    public void onPictureTaken(byte[] data, Camera camera) {
        //Mengambil File foto kosong yang telah dibuat menggunakan method getOutputMediaFile()
        File pictureFile = getOutputMediaFile();
        //Mengecek File
        if (pictureFile == null) {
            Toast.makeText(MainActivity.this, "Error creating media file, check storage permissions", Toast.LENGTH_SHORT).show();
            return;
        }
        //Memasukan foto yang telah diambil dan menyimpannya
        try {
            FileOutputStream fos = new FileOutputStream(pictureFile);
            fos.write(data);
            fos.close();
        } catch (FileNotFoundException e) {
            Toast.makeText(MainActivity.this, "File not found: " + e.getMessage(), Toast.LENGTH_SHORT).show();
        } catch (IOException e) {
            Toast.makeText(MainActivity.this, "Error Accesing File : " + e.getMessage(), Toast.LENGTH_SHORT).show();
        }
    }
};
```

Gambar 9.21 Fungsi onPictureTaken

Lalu tambahkan method setOnClickListener untuk menghandle button click dan memanggil method takePicture()

```
button_shutter.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        switch (previewState){
            case "BUSY":
                //menampilkan layar preview
                mCamera.startPreview();
                previewState = "PREVIEW";
                break;
            case "PREVIEW":
                //mengambil gambar
                mCamera.takePicture(null,null, mPicture);
                previewState = "BUSY";
                break;
        }
    }
});
```

Gambar 9.22 Fungsi Handle saat button shutter di click

Ketika button di click, aplikasi akan mengambil gambar (**takePicture**()) dan layar preview akan menampilkan gambar yang diambil dan tidak menampilkan preview kamera. Untuk menampilkan ulang tampilan preview camera, harus menampikan ulang preview dari camera dengan memanggil **camera**.**startPreview**()
Oleh karena itu switch case digunakan, aplikasi akan menampilkan ulang preview camera Ketika previewState bernilai **BUSY**, dan akan mengambil gambar Ketika previewState bernilai **PREVIEW**.
Dimana nilai default dari previewState adalah “**PREVIEW**”.

```
private static String previewState = "PREVIEW";
```

Coba jalankan, dan click tombol untuk mengambil gambar. Amati hasilnya

Untuk fungsi-fungsi camera lainnya dapat dilihat pada link berikut :

- https://developer.android.com/training/camera
- https://developer.android.com/training/camera/cameradirect
- https://developer.android.com/guide/topics/media/camera

Tugas

- Mengerjakan Latihan diatas untuk menggunakan kamera pada perangkat smartphone / emulator
- Ubah program diatas untuk kamera depan / kamera lainnya.