안드로이드 앱을 개발할 때 스마트폰에 내장된 구글 기본 앱(주소록 앱, 갤러리 앱, 카메라 앱)과의 연동을 자주 사용한다. 앱 연동 기법은 구글 앱에 제한되는 것은 아니며, 외부 앱 연동 기법으로 모든 다양한 앱과 연동할 수 있다. 어떤 외부 앱이든 화면에 외부 앱의 액티비티를 띄우는 것은 인텐트 발생으로 되고, 그 외부 앱의 데이터를 가져오는 것은 콘텐츠 프로바이더로 한다.

● 주소록 앱 연동: 주소록 액티비티를 화면에 띄우고, 그 목록에서 사용자가 선택한 항목의 전화번호나 이메일을 가져와야 할 때 주소록 앱을 연동하여 이용한다. 인텐트 발생으로 넘어오는 식별자를 이용해서 다시 콘텐츠 프로바이더를 통해 원하는 데이터를 요청하는 방식이다.

1) 주소록 앱 목록을 화면에 띄우기: 인텐트 발생

주소록 앱을 연동하여 목록 화면을 띄우기 위해서는 암시적 인텐트를 발생시켜야 한다. Action 문자열을 Intent.ACTION_PICK, 데이터 정보를 ContactsContracts.Contacts.Phone.CONTENT_URI로 설정하여 인텐트를 발생시키면 주소록의 목록 화면이 뜨게 된다. 또한 유저가 선택한 주소록 목록 중 하나의 결과를 되돌려 받아야 하므로 ActionResultLauncher을 이용한다. 목록 액티비티가 실행된 후에는 사용자가 선택한 항목의 식별자 값이 Uri 객체 타입으로 전달된다.

```
// 액션 정보와 데이터 정보를 주소록 앱 쪽에서 선언되어 있는 것이랑 동일하게 해준다.
Intent intent = new Intent(Intent.ACTION_PICK);
// 액션 정보
intent.setData(ContactsContract.CommonDXataKinds.Phone.CONTENT_URI);
// 데이터 정보. Uri.parse(" ")로 주어도 된다. 상수 변수로 선언되어 있을 뿐.
contactsLauncher.launch(intent);
// 목록 화면에서 선택해서 데이터가 돌아왔을 때 사후처리가 필요하므로 registerForActivityResult
// 을 구현한 contactsLauncher 사용해서 intent를 발생시킨다
```

```
ActivityResultLauncher<Intent> contactsLauncher = registerForActivityResult (
    new ActivityResultContracts.startAcitivtyForResult(),
    new ActivityResultCallback<ActivityResult>() {
        @Override
        public void onActivityResult(ActivityResult result) {
            Uri data = result.getData().getData();
            // 주소록 앱에서 넘어오는 데이터. URL 문자열이다. 맨 마지막 path가 유저가 선택한
            // 이의 식별자 값이다.
            Log.d("kkang", "uri : " + data.toString());
        }
    }
```

넘어온 Uri 값을 출력해 보면 아래와 같다.

```
content://com.android.contacts/data/1
```

2) 주소록 앱의 데이터를 우리 쪽 앱으로 가져오기: 콘텐츠 프로바이더

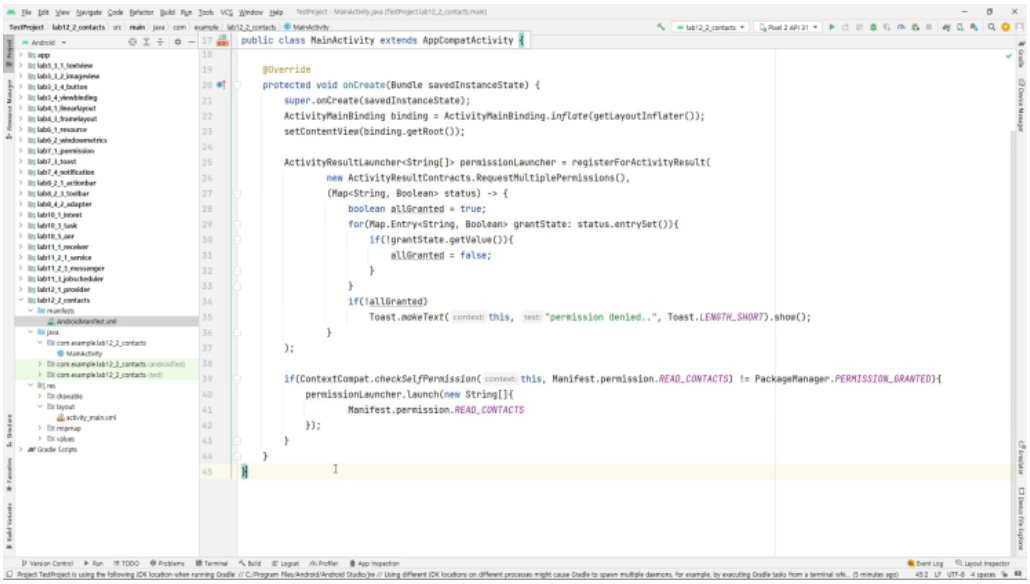
이 Uri 정보(식별자)를 조건으로 이용해 구체적으로 원하는 데이터, 전화번호나 이메일 등을 값을 다시 가져와야 하는데, 이 부분은 앱과 앱 간의 데이터 부분이므로 콘텐츠 프로바이더를 이용해야 한다. 주소록의 콘텐츠 프로바이더 이용은 퍼미션 등록이 필요하다.

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

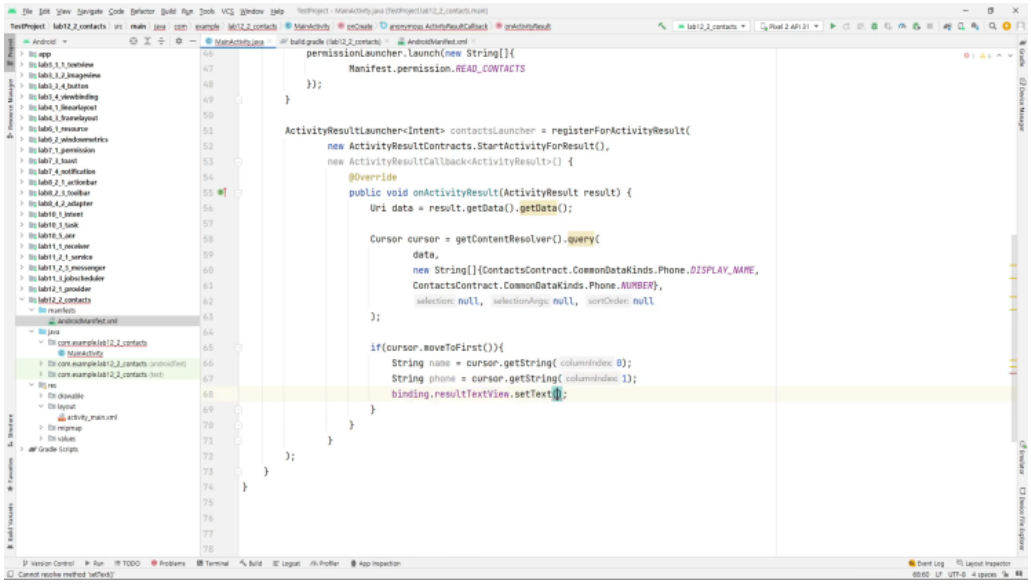
콘텐츠 프로바이더로 주소록 앱과 연동하여 전화번호를 획득하는 코드는 아래와 같다.

```
Cursor cursor = getContentResolver().query(result.getData().getData(),
                new String[]{ContactsContracts.CommonDataKinds.Phone.DISPLAY_NAME,
                ContactsContract.CommonDataKinds.Phone.NUMBER}, null, null, null);
                // 조건을 붙이지 않아도 URI에 조건이 담겨 있기 때문에
                // 얻고자 하는 데이터 이름과 번호만 표기해도 찾는 데이터가 넘어온다

// Cursor은 여러 데이터가 담긴 표같은 형식이다
// 그러므로 커서가 첫번째 row만 선택해 주어야 한다.
if (cursor.moveToFirst()) {
    String name = cursor.getString(0); // 첫번째 행에서 첫번째 데이터가 사람 이름
    String phone = cursor.getString(1); // 두번째 데이터가 전화번호
    binding.resultTextView.setText("name: " + name + ", phone: " + phone);
}
```



퍼미션 요청 관련 코드



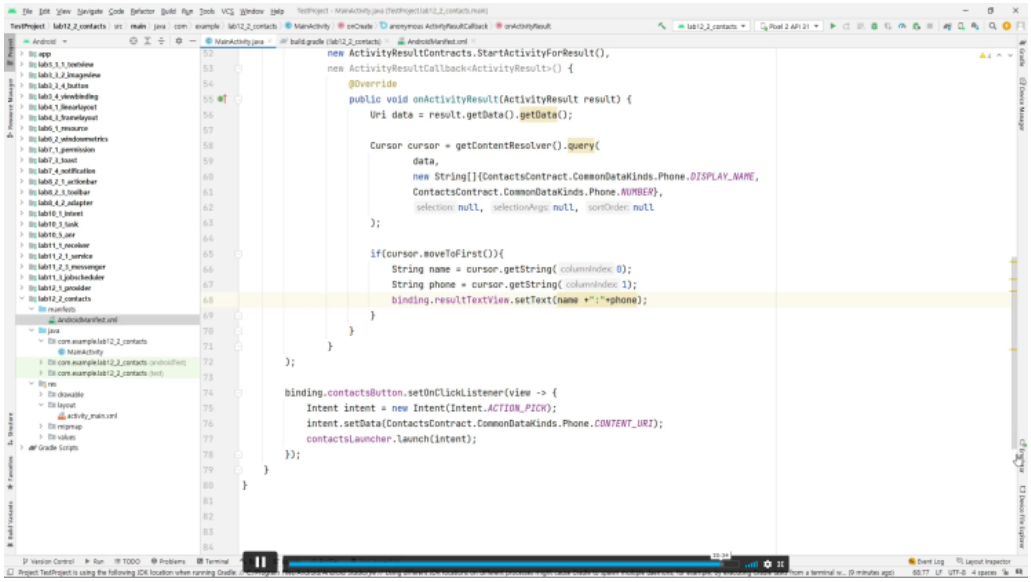
```
permissionLauncher.launch(new String[]{
        Manifest.permission.READ_CONTACTS
});
}

ActivityResultLauncher<Intent> contactsLauncher = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        new ActivityResultCallback<ActivityResult>() {
            @Override
            public void onActivityResult(ActivityResult result) {
                Uri data = result.getData().getData();

                Cursor cursor = getContentResolver().query(
                        data,
                        new String[]{ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
                        ContactsContract.CommonDataKinds.Phone.NUMBER},
                         selection: null,  selectionArgs: null,  sortOrder: null
                );

                if(cursor.moveToFirst()){
                    String name = cursor.getString( columnIndex: 0);
                    String phone = cursor.getString( columnIndex: 1);
                    binding.resultTextView.setText();
                }
            }
        }
);
}
}
```

Uri을 되돌려 받은 다음 콘텐츠 프로바이더로 넘겨주어 데이터 획득해서 저장하고 화면에 출력하기



```
        new ActivityResultContracts.StartActivityForResult(),
        new ActivityResultCallback<ActivityResult>() {
            @Override
            public void onActivityResult(ActivityResult result) {
                Uri data = result.getData().getData();

                Cursor cursor = getContentResolver().query(
                        data,
                        new String[]{ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
                        ContactsContract.CommonDataKinds.Phone.NUMBER},
                         selection: null,  selectionArgs: null,  sortOrder: null
                );

                if(cursor.moveToFirst()){
                    String name = cursor.getString( columnIndex: 0);
                    String phone = cursor.getString( columnIndex: 1);
                    binding.resultTextView.setText(name +":"+phone);
                }
            }
        }
);

binding.contactsButton.setOnClickListener(view -> {
    Intent intent = new Intent(Intent.ACTION_PICK);
    intent.setData(ContactsContract.CommonDataKinds.Phone.CONTENT_URI);
    contactsLauncher.launch(intent);
});
}
}
```

● 카메라 앱 연동: 개발하는 앱에 사진을 찍거나 동영상을 촬영하는 기능이 포함되어야 할 때, 카메라 앱을 연동하면 편리하다. 카메라 앱의 사진을 찍는 화면도 결국 액티비티이므로 인텐트를 발생시키는 것만으로도 쉽게 촬영 기능과 촬영된 데이터를 얻을 수 있다. 카메라 앱을 영동하여 촬영한 데이터를 획득하는 방법에는 섬네일로 받는 방법과 파일 공유 방법이 있다. 1) 섬네일 방법: 우리 앱에서 intent로 카메라 앱을 띄워서 사진을 찍으면 사진 data가 돌아오는 방식으로 사이즈가 아주 줄어들어서 들어온다. 2) 파일 공유 방법: 우리 앱에서 파일을 생성해서 파일 정보를 카메라 앱으로 보내면 카메라 앱이 사진을 찍고 사진을 file write하고 성공/실패를 넘겨주는 방식. 사진이 그대로 write 된다.

1. 섬네일로 결과 받기: 연동한 카메라 앱으로 촬영한 결과 데이터만 앱에서 전달받는다. 촬영된 데이터는 파일로 저장되지 않으며, 데이터 전달 방식이므로 사진 데이터의 크기가 작게 전달된다. 섬네일 이미지 데이터만 전달되는 정도이다.

Action 문자열을 MediaStore.ACTION_IMAGE_CAPTURE로 지정하고 인텐트를 발생시켜서 간단하게 카메라 앱의 액티비티를 실행할 수 있다.

```
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
cameraThumbnailLauncher.launch(intent);
```

결과값은 onActivityResult() 함수로 얻는다.

```
ActivityResultLauncher<Intent> cameraThumbnailLauncher = registerForActivityResult (
    new ActivityResultContracts.StartActivityForResult(),
    new ActivityResultCallback<ActivityResult>() {
        @Override
        public void onActivityResult(ActivityResult result) {
            Bitmap bitmap = (Bitmap) result.getData().getExtras().get("data");
        }
    }
);
```

촬영한 결과는 result.getData().getExtras().get("data")로 얻으며, 전달되는 타입은 bitmap이다.

2. 파일 공유 방법: 연동된 카메라 앱으로 촬영한 사진 데이터의 정보가 공유된 파일에 저장된다. 연동한 애보가 파일을 공유하고, 파일에 저장된 정보로 사진 데이터를 추출해야 하므로 코드가 다소 복잡하지만 촬영된 사진 파일 그대로 전달받을 수 있다. 파일 공유 방법은 개발한 앱에서 입의의 경로에 파일을 하나 만든다. 그리고 해당 파일의 경로를 카메라 앱에 전달하고, 카메라 앱에서 촬영 데이터 정보를 파일에 쓰고(write) 성공 여부를 반환하는 방식이다.

파일 만들어서 이 파일 정보를 외부 앱(카메라 앱)에 넘겨주기. 파일 정보를 외부 앱에 전달하기 위해서는 content://URI를 보내고, 이 URI에 대해 임시 액세시 권한을 부여해 주어야 하는데, 이를 쉽게 부여하려면 androidx에서 제공하는 콘텐츠 프로바이더인 FileProvider 클래스를 이용하면 된다. 이를 위해서는 res/xml 폴더에 임의의 이름으로 된 XML 파일을 만들어서 아래의 내용을 작성해야 한다.

```
// res/xml > .xml
<path xmlns:android="http://schemas.android.com/apk/res/android">
    // path=패키지명에 해당되는 폴더명/files/Pictures 파일을 공개로 설정
    <external-path name="myfiles" path="Android/data/com.exmaple.test4_12/files/Pictures
    // file/Pictures 경로에 있는 이 파일을 외부에 공개하겠다고 설정
</path>
```

<external-path> 태그는 외부 저장 공간의 파일을 공유하기 위해 사용되며, 내부 저장 공간에 대한 공유는 <file-path> 태그를 이용한다. 이렇게 작성한 XML 파일을 AndroidManifest.xml에서 FileProvider을 등록할 때 설정해 준다(FileProvider은 우리가 만든 프로바이더가 아니기 때문에 작성을 할 필요는 없지만 등록은 해 주어야 함). authorities 속성에 유일성이 확보된 식별자 문자를 하나 설정해 준다 (아래에서는 앱의 패키지명 이용). 그리고 <meta-data> 태그로 정의한 XML 파일의 정보를 설정한다.

```
// AndroidManifest.xml
<provider android:name="androidx.core.content.FileProvider"
    android:authorities="com.example.test4_12.fileprovider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml_file_paths"></meta-data>
        // 아까 만들어 놓은 xml 파일 등록
</provider>
```

이렇게 FileProvider을 등록한 후에 자바 코드에서 공유하려는 파일 정보의 Uri 값을 가져온다.

```
Uri photoURI = FileProvider.getUriForFile(this, "com.exmaple.test4_12.fileprovider", fil
```

얻은 uri 값을 카메라 앱 실행을 위한 인텐트의 Extra 데이터로 설정한다.

```
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
intent.putextra(MediaStore.EXTRA_OUTPUT, photoURI);
// 두번째 매개 변수로 파일의 식별자를 넘겨준다
cameraFileLuancher.launch(intent);
```

▶ 이미지 로딩으로 인한 OutOfMemoryException 문제: 크기가 큰 데이터를 로딩하다가 앱의 메모리 부족으로 실행 도중에 에러가 발생하는 경우가 있다. 서버로부터 내려받은 이미지나 카메라로 찍은 사진 이미지는 데이터의 크기가 클 수 있고, 이런 이미지를 화면에 출력하기 위해 로딩하면 OutOfMemoryException이 자주 발생한다. 이 문제를 피할려면 이미지의 크기를 줄여서 로딩해야 한다. 이미지 크기를 줄이는 방법은 API에서 제공한다.

Bitmap(이미지 데이터를 표현하는 객체)은 이미지를 표현하는 클래스로 BitmapFactory 클래스로 생성하고, BitmapFactory의 decodeXXX() 함수로 생성하여 이용한다.

BitmapFactory.decodeByteArray()	byte[] 배열로 Bitmap 생성
BitmapFactory.decodeFile()	파일 경로로 FileInputStream을 만들어서 decodeStream 이용
BitmapFactory.decodeResource()	Resource 폴더에 저장된 파일
BitmapFactory.decodeStream()	InputStream으로 Bitmap 생성

이때 옵션을 설정할 수 있다. 특히, Options 클래스의 inSampleSize 속성이 중요하다. 이 속성 값을 decodeXXX() 함수의 두 번째 매개변수에 option으로 지정하면, 이미지 크기를 자동으로 줄여서 로딩한다.

```
BitmapFactory.Options imgOptions = new BitmapFactory.Options();
imgOptions.inSampleSize = 10;
Bitmap bitmap = BitmapFactory.decodeFile(filePath.getAbsolutePath(), imgOptions);
```

위의 코드에서 이미지 크기를 10으로 지정하였는데, 이는 전달받은 이미지를 10분의 1로 줄여서 로딩하라는 의미이다. 이렇게 간단하게 이미지 크기를 줄여서 OutOfMemoryException 문제를 해결할 수 있다.

● 갤러리 앱 연동: 사진을 목록으로 보여주거나, 한 장의 사진을 크게 보여주는 액티비티를 인텐트로 실행할 수 있다. 또한 사진에 대한 각종 데이터를 콘텐츠 프로바이더를 이용해 얻을 수도 있다. (Intent로 발생시키고, 다시 URL 식별자로 id가 넘어오면, 다시 Provider에 식별자를 넘겨 구체적으로 원하는 데이터를 얻는 방식)

1) 인텐트를 발생시켜 갤러리 앱의 목록 액티비티를 띄운다. 인텐트에 Action 정보와 Type 정보, 그리고 Data정보를 준다.

```
Intent intent = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_
// 액션 문자열과 데이터 정보. 데이터 정보는 상수로 선언되어 있지만 Url.parse()로 직접 입력 가능
intent.setType("image/*");
galleryLauncher.launch(intent);
```

2) 사용자가 사진 한 장을 선택하면 다시 화면으로 돌아와야 하므로 ActivityResultLauncher을 이용한다. onActivityResult() 함수 내에서 갤러리 앱의 콘텐츠 프로바이더가 제공하는 InputStream 객체를 획득할 수 있다. 이 객체로 사용자 갤러리 앱에서 선택한 사진을 획득한다.

```
ActivityResultLauncher<Intent> galleryLauncher = registerForActivityResult (
    new ActivityResultContracts.StartActivityForResult(),
    new ActivityResultCallback<ActivityResult>() {
        @Override
        public void onActivityResult(ActivityResult result) {
            int calRatio = calculateInSampleSize (
                result.getData().getData(), // 갤러리 앱에서 선택한 사진 정보 넘기기
                getResources().getDimensionPixelSize(R.dimen.imgSize),
                getResources().getDimensionPixelSize(R.dimen.imgSize)
                // 화면에 뿌리기 위한 사이즈
                // 사진 정보와 화면에 뿌리기 위한 사이즈를 비교해서
                // 얼마나 줄여주어야 하는지 알려주는 개발자 함수 calRatio
            );

        BitmapFactory.Options option = new BitmapFactory.Options();
        option.inSampleSize = calRatio;

        try {
            InputStream inputStream =
                        getContentResolver().OpenInputStream(result.getData().getData())
                        // 갤러리 앱의 프로바이더에서 사진을 읽을 수 있는 스트림 제공
            Bitmap bitmap = BitmapFactory.decodeStream(inputStream, null, option);
            inputStream.close();
            if (bitmap != null) {
                // 비트맵을 이미지 뷰에 설정
                binding.resultImageView.setImageBitmap(bitmap);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    }
});
```

● 지도 앱: 앱에서 위치 정보 데이터인 위경고 데이터를 가지고 있을 때, 이 데이터로 지도를 띄울 수 있다. (ex. 회사 위치를 지도 앱으로 띄우고 싶을 때)

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("geo:37.5662952, 126.9779451"))
startActivit(intent);
```

Action 문자열을 Intent.ACTION_VIEW로 지정하며, 지도에서 가운데 위치로 이용할 위경도 값을 데이터 정보로 설정한다. 이때 URL의 scheme은 "geo"로 설정한다.

● 전화 앱: 앱에서 전화번호를 가지고 있을 때, 이 데이터와 전화를 거는 기능을 전화 앱 연동으로 연결할 수 있다. 퍼미션이 필요하다.

```
<uses-permission android:name="android.permission.CALL_PHONE"/>
```

```
Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:02-120"));
startActivity(intent);
```

"tel"을 scheme 정보로 설정하고, 그 뒤에 전화번호 정보를 추가해서 이용한다.

```
// file_path.xml

<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <external-path
        name="myfiles"
        path="Android/data/com.android.practiceconnectingwithgoogleapps/files/Pictures"/
</paths>
```

```
// AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.android.practiceconnectingwithgoogleapps">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.PracticeConnectingWithGoogleApps"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

        <provider
            android:authorities="com.android.practiceconnectingwithgoogleapps"
            android:name="androidx.core.content.FileProvider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/file_path"></meta-data>
                // 이 xml에 등록되어 있는 파일 경로가 FileProvider을 이용해서 외부 앱에 노출
        </provider>

    </application>
</manifest>
```

```
// MainActivity.java

package com.android.practiceconnectingwithgoogleapps;

import androidx.activity.result.ActivityResult;
import androidx.activity.result.ActivityResultCallback;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.content.FileProvider;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.graphics.Bitmap;
```

```
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.ContactsContract;
import android.provider.MediaStore;
import android.util.Log;
import android.widget.Toast;

import com.android.practiceconnectingwithgoogleapps.databinding.ActivityMainBinding;
import java.io.File;
import java.io.InputStream;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;

public class MainActivity extends AppCompatActivity {
    String filePath;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityMainBinding binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        // 카메라 앱 연동. 되돌아왔을 때 사후 처리를 위해 ActivityResultLauncher 준비
        ActivityResultLauncher<Intent> cameraFileLauncher = registerForActivityResult(
                // 인텐트를 처리해 줄 수 있는 객체
                new ActivityResultContracts.StartActivityForResult(),
                // 되돌아 왔을 때 콜백
                new ActivityResultCallback<ActivityResult>() {
                    @Override
                    public void onActivityResult(ActivityResult result) {
                        // 되돌아 왔을 때 처리 내용을 이 함수에 담아주면 된다

                        // 화면에 이미지의 사이즈를 줄여서 출력하기 위한 사이즈 계산
                        // 밑에 정의되어 있는 calculateInSampleSize 콜하기
                       int calRatio = calculateInSampleSize(

                                result.getData().getData(),
                               // 혹은 파일 url값 - 카메라 앱에 공개한 우리의 파일에 저장된
//                                Uri.fromFile(new File(filePath)),
                                // 화면에 찍고자 하는 사이즈
                                getResources().getDimensionPixelSize(R.dimen.imgSize),
                                getResources().getDimensionPixelSize(R.dimen.imgSize)
                        );

                        // calRatio 사이즈로 화면에 사진 출력하기
                        BitmapFactory.Options option = new BitmapFactory.Options();
                        option.inSampleSize = calRatio;
                        Bitmap bitmap = BitmapFactory.decodeFile(filePath, option);

                        if (bitmap != null) {
                            binding.resultImageView.setImageBitmap(bitmap);
                        }
                    }
                });

        // 카메라 버튼이 눌리면 카메라 앱 연동
        binding.cameraButton.setOnClickListener(view -> {
```

```
            try {
                // 파일 만들기. 시분초로 파일명 스트링 만들어서.
                // 그 스트링을 파일명으로 삼기
                String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Dat
                // 사진이 저장되는 외장 메모리 공간 지정(디렉토리 지정)
                File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
                // 파일 만들기. JPEG_시간날짜_.jpg 이름으로 디렉터리(우리가 지정한 디렉토리)에
                File file = File.createTempFile(
                        "JPEG_" + timeStamp + "_", ".jpg", storageDir
                );
                // 파일 경로 지정
                filePath = file.getAbsolutePath();
                // 외부 앱에 공개. 파일 프로파이더 이용. 공개하기 위한 Uri 만들기
                // 이 파일을 지칭하는 Uri값
                Uri photoURI = FileProvider.getUriForFile(this, "com.android.practiceconr

                Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                intent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
                cameraFileLauncher.launch(intent);
            } catch (Exception e) {
                e.printStackTrace();
            }

        });
```

```
// 갤러리 앱 연동. 사후 처리를 위한 ActivityResultLauncher.
        ActivityResultLauncher<Intent> galleryLauncher = registerForActivityResult(
                new ActivityResultContracts.StartActivityForResult(),
                new ActivityResultCallback<ActivityResult>() {
                    @Override
                    public void onActivityResult(ActivityResult result) {
                        int calRatio = calculateInSampleSize(
                                // 갤러리 앱에서 얻은 사진 데이터
                                result.getData().getData(),
                                getResources().getDimensionPixelSize(R.dimen.imgSize),
                                getResources().getDimensionPixelSize(R.dimen.imgSize)
                        );

                        BitmapFactory.Options option = new BitmapFactory.Options();
                        option.inSampleSize = calRatio;

                        try {
                            // 갤러리앱 쪽에서 제공하는 사진 읽는 스트림
                            InputStream inputStream = getContentResolver().openInputStre
                            Bitmap bitmap = BitmapFactory.decodeStream(inputStream, null
                            inputStream.close();
                            if (bitmap != null) {
                                binding.resultImageView.setImageBitmap(bitmap);
                            }
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    }
                });

        // 갤러리 버튼을 눌렀을 때 런쳐 구동
        binding.galleryButton.setOnClickListener(view -> {
```

```
            // 액션 정보, 데이터 정보, 타입 정보
            Intent intent = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTER
            intent.setType("image/*");
            galleryLauncher.launch(intent);
        });

    }


    // 몇 분에 몇으로 줄여야 하느냐를 계산해 주는 함수
    private int calculateInSampleSize(Uri fileUri, int reqWidth, int reqHeight) {
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inJustDecodeBounds = true;
        try {
            InputStream inputStream = getContentResolver().openInputStream(fileUri);

            //inJustDecodeBounds 값을 true 로 설정한 상태에서 decodeXXX() 를 호출.
            //로딩 하고자 하는 이미지의 각종 정보가 options 에 설정 된다.
            BitmapFactory.decodeStream(inputStream, null, options);
            inputStream.close();
            inputStream = null;
        } catch (Exception e) {
            e.printStackTrace();
        }
        //비율 계산.....................
        int width = options.outWidth;
        int height = options.outHeight;
        int inSampleSize = 1;

        //inSampleSize 비율 계산
        if (height > reqHeight || width > reqWidth) {

            int halfHeight = height / 2;
            int halfWidth = width / 2;

            while (halfHeight / inSampleSize >= reqHeight && halfWidth / inSampleSize >=
                inSampleSize *= 2;
            }
        }
        return inSampleSize;
    }
}
```

Resources: 깡쌤의 안드로이드 프로그래밍