2. 안드로이드 앱의 기본 구조

1. 안드로이드 소개
2. 안드로이드 앱 개발의 특징
3. 앱 구성 파일 분석

1. 안드로이드 소개

안드로이드 특징

- 안드로이드는 공개 운영체제인 리눅스를 기반으로 .
- 안드로이드 앱은 자바나 코틀린 언어를 이용해 개발.
- 안드로이드 운영체제의 주요 부분과 라이브러리, 구글에서 만든 앱 등의 코드는 대부분 공개되어 있음
- 안드로이드 스마트폰은 구글뿐 아니라 여러 제조업체에서 만들 수 있음.
- 안드로이드 앱은 구글의 플레이 스토어뿐만 아니라 다양한 방법으로 사용자에게 배포할 수 있음.
- 안드로이드 플랫폼에서는 모든 응용 프로그램이 평등하다는 사상을 바탕으로, 모바일에 기본으로 탑재된 앱과 개발자가 만든 앱이 똑같은 환경에서 똑같은 API를 이용.

1. 안드로이드 소개

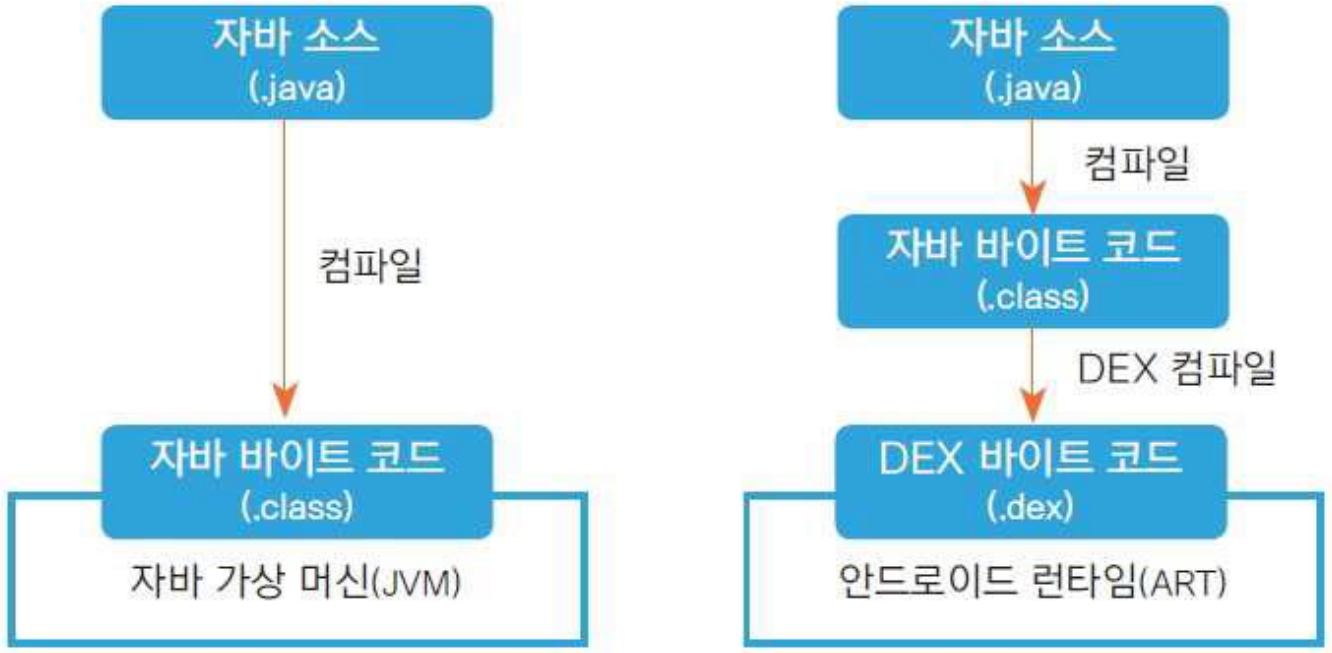
안드로이드 운영체제의 구조

- 리눅스 커널(Linux kernel) : 안드로이드는 리눅스에 기반을 둔 오픈 소스 소프트웨어 스택
- 하드웨어 추상화 레이어(hardware abstraction layer, HAL) : 상위의 자바 API 프레임워크에서 하드웨어 기능을 이용할 수 있게 표준 인터페이스를 제공
- 안드로이드 런타임(Android runtime) : ART라고 하며 앱을 실행하는 역할
- 네이티브 C/C++ 라이브러리 : 네이티브 C/C++ 라이브러리를 이용할 수 있으며 이를 안드로이드 NDK(native development kit)라고 함.
- 자바 API 프레임워크 : 앱을 개발할 때 사용하는 자바 API

1. 안드로이드 소개

▪ 안드로이드 운영체제의 구조

- 안드로이드는 자바 클래스를 런타임 때 그대로 실행하지 않고 DEX 파일로 컴파일.
- DEX 파일을 해석하는 ARTAndroid runtine에서 실행



1. 안드로이드 소개

안드로이드 버전

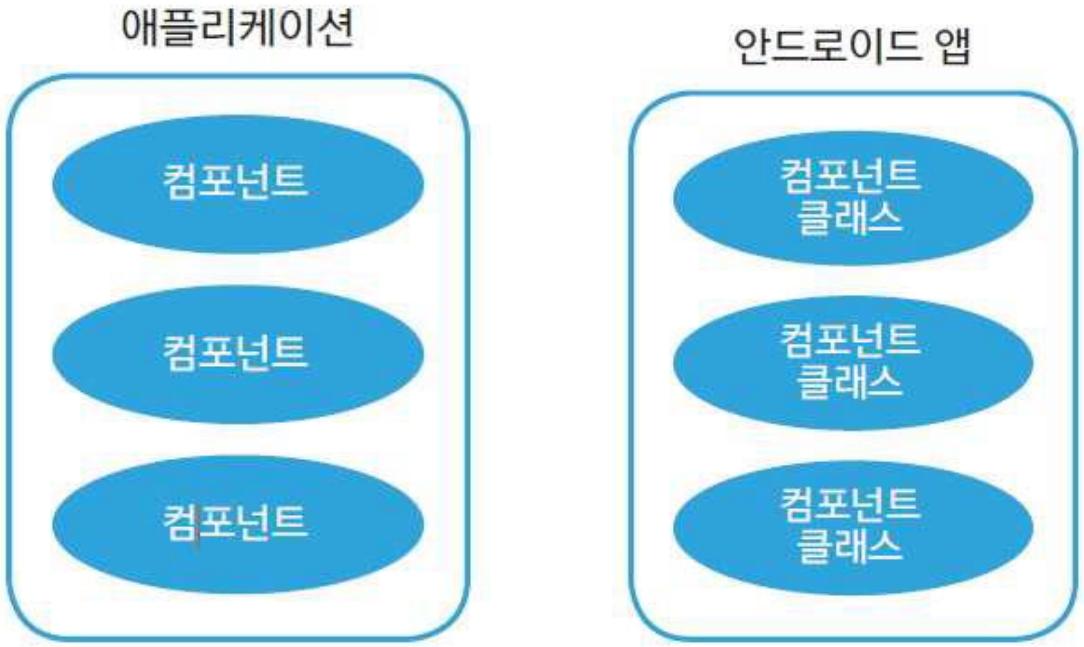
- 안드로이드 버전은 11.0, 12.0처럼 운영체제 버전을 가리키지만 앱을 개발할 때 사용하는 버전은 API 레벨(SDK 버전)사용

버전	코드명	API 레벨	출시 연도
Android 1.0	-	1	2008
Android 1.1	Petit Four	2	2009
Android 1.5	Cupcake	3	2009
Android 1.6	Donut	4	2009
Android 2.0.x ~ 2.1	Éclair	5~7	2009
Android 2.2.x	Froyo	8	2010
Android 2.3.x	Gingerbread	9~10	2010
Android 3.x	Honeycomb	11~13	2011
Android 4.1.x ~ 4.3.x	Jelly Bean	16~18	2012
Android 4.4.x	KitKat	19~20	2013
Android 5.x	Lollipop	21~22	2014
Adnroid 6.x	Marshmallow	23	2015
Android 7.x	Nougat	24~25	2016
Adnroid 8.x	Oreo	26~27	2017
Adnroid 9.0	Pie	28	2018
Android 10.0	-	29	2019
Android 11.0	-	30	2020
Android 12.0	-	31	2021

2. 안드로이드 앱 개발의 특징

■ 컴포넌트를 기반으로 한 개발

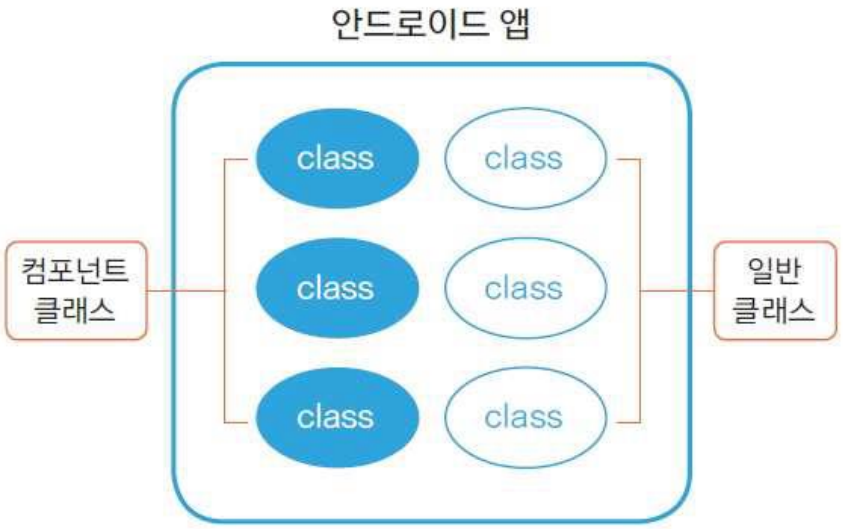
- 컴포넌트는 애플리케이션의 구성 요소
  - 컴포넌트를 한마디로 정의하면 애플리케이션의 구성 요소라고 할 수 있음.
  - 안드로이드에서는 클래스로 컴포넌트를 개발.



2. 안드로이드 앱 개발의 특징

▪ 컴포넌트를 기반으로 한 개발

- 안드로이드 앱을 구성하는 클래스는 모두 컴포넌트인가?
  - 앱은 여러 클래스로 구성되는데 크게 컴포넌트 클래스와 일반 클래스로 구분.
  - 클래스의 객체 생성부터 소멸까지 생명주기 관리를 개발자 코드에서 한다면 일반 클래스
  - 생명주기를 안드로이드 시스템에서 관리한다면 컴포넌트 클래스.

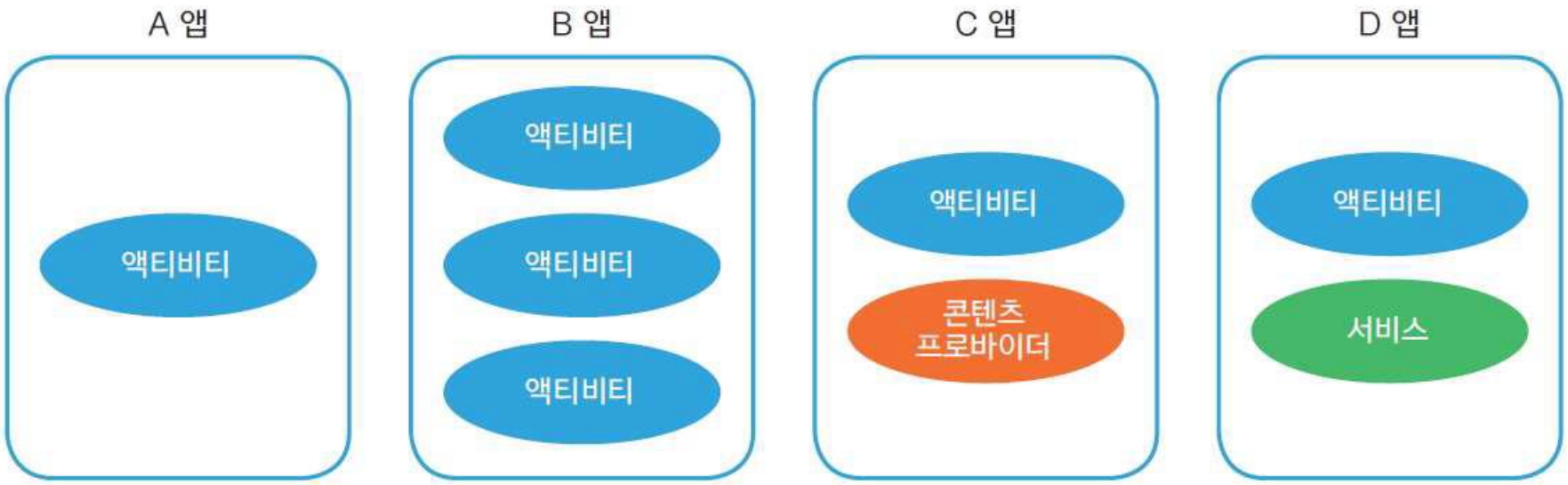


2. 안드로이드 앱 개발의 특징

- **안드로이드 컴포넌트는 4종류**
  - 액티비티 : 화면을 구성하는 컴포넌트.
  - 서비스 : 백그라운드 작업을 하는 컴포넌트
  - 콘텐츠 프로바이더 : 앱의 데이터를 공유하는 컴포넌트
  - 브로드캐스트 리시버 : 시스템 이벤트가 발생할 때 실행되게 하는 컴포넌트
- **4가지 컴포넌트를 어떻게 구분하는가?**
  - 컴포넌트 클래스를 만들 때는 지정된 클래스를 상속받아야 하는데 이 상위 클래스를 보고 구분.
  - 액티비티 : Activity
  - 서비스는 : Service
  - 콘텐츠 프로바이더 : ContentProvider
  - 브로드캐스트 리시버: BroadcastReceiver
- 클래스를 상속받아서 만듬

2. 안드로이드 앱 개발의 특징

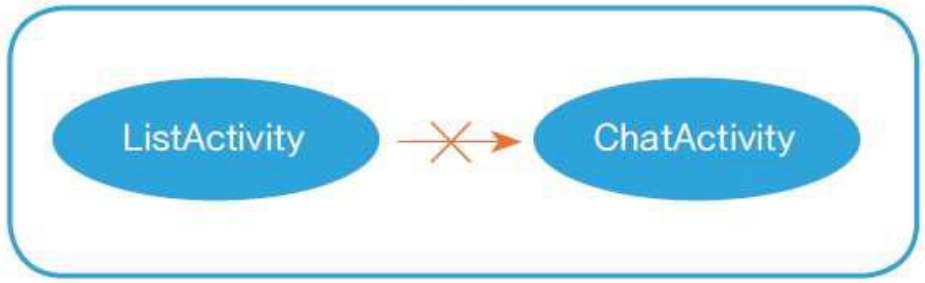
- **안드로이드 컴포넌트는 4종류**
  - 앱을 개발할 때 컴포넌트를 어떻게 구성해야 하는가?
    - 앱의 기능과 화면 등을 고려해 필요한 만큼 구성



2. 안드로이드 앱 개발의 특징

안드로이드 컴포넌트는 4종류

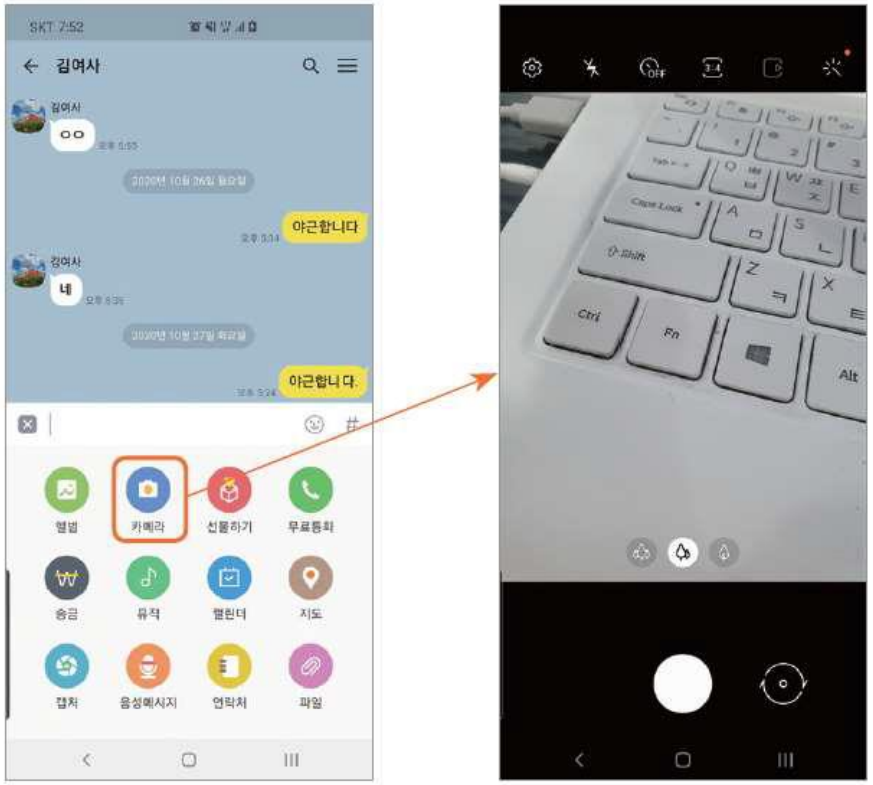
- 컴포넌트는 앱 안에서 독립된 실행 단위
  - 컴포넌트끼리 서로 종속되지 않아서 코드 결합이 발생하지 않는다는 의미



- 앱 실행 시점이 다양하다
  - 컴포넌트가 앱 내에서 독립해서 실행되는 특징 덕분에 앱의 실행 시점이 다양할 수 있음.
  - 안드로이드 앱에는 메인 함수 main function 개념이 없음

2. 안드로이드 앱 개발의 특징

- **애플리케이션 라이브러리를 사용**
  - 다른 애플리케이션을 라이브러리처럼 이용하는 것을 말함.



2. 안드로이드 앱 개발의 특징

▪ 리소스를 활용한 개발

- 리소스란 코드에서 정적인 값을 분리한 것
- 문자열 이외에 색상, 크기, 레이아웃, 이미지, 메뉴 등 많은 요소를 리소스로 활용할 수 있음
- 이미지 등 몇몇을 제외하면 대부분 리소스는 XML 파일로 작성함

• 문자열을 리소스로 등록하기

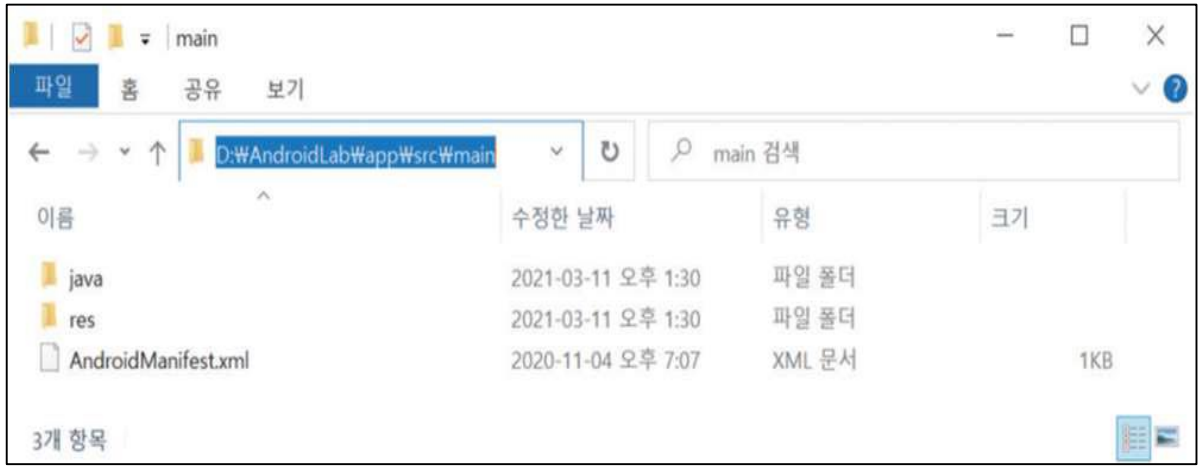
```
<string name="mytxt">
    동해 물과 백두산이 마르고 닳도록
    하느님이 보우하사 우리나라 만세
    무궁화 삼천리 화려 강산
    대한 사람, 대한으로 길이 보전하세
</string>
```

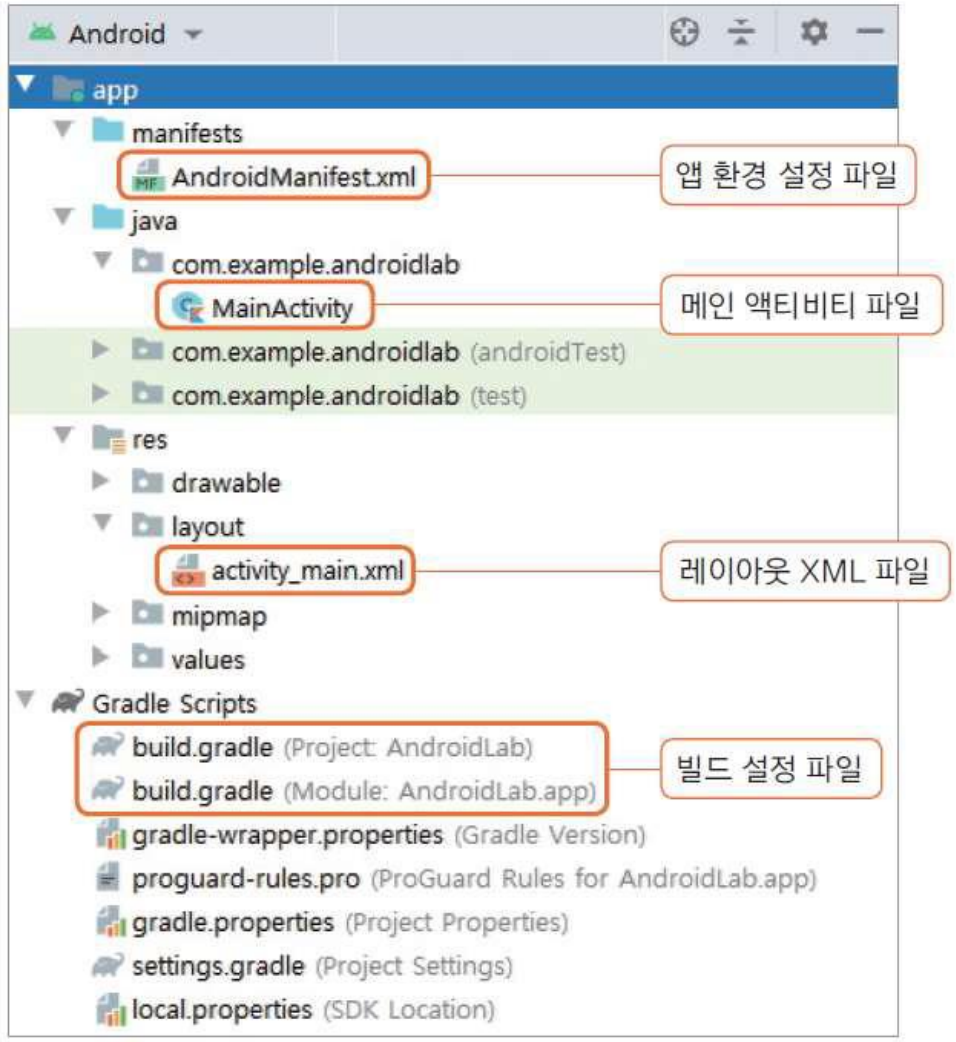
• 문자열 리소스 사용 예

```
textView.text = resources.getString(R.string.mytxt)
```

3. 앱 구성 파일 분석

- **프로젝트의 폴더 구성 알아보기**
  - 프로젝트 폴더에서 [모듈명 → src → main]





3. 앱 구성 파일 분석

모듈의 폴더 구성 알아보기

- 그래들 빌드 설정 파일
  - 그래들은 안드로이드 앱의 빌드 도구입니다.
  - 그래들의 설정 파일이 바로 build.gradle
  - 프로젝트 수준의 build.gradle (Project: AndroidLab)과 모듈 수준의 build.gradle (Module: AndroidLab.app)

이름	설명
build.gradle	빌드 설정 파일
AndroidManifest.xml	앱의 메인 환경 파일
res	리소스 폴더
activity_main.xml	레이아웃 XML 파일
MainActivity.kt	메인 액티비티 파일

3. 앱 구성 파일 분석

- 모듈의 폴더 구성 알아보기

• 플러그인 선언

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
}
```

• 컴파일 버전 설정

```
compileSdk 31
```

• 앱의 식별자 설정

```
applicationId "com.example.androidlab"
```

• SDK 버전 설정

```
minSdk 21
targetSdk 31
```

• 앱의 버전 설정

```
versionCode 1
versionName "1.0"
```

• 컴파일 옵션

```
compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}
kotlinOptions {
    jvmTarget = '1.8'
}
```

3. 앱 구성 파일 분석

▪ 모듈의 폴더 구성 알아보기

- targetSdk에 명시한 안드로이드 SDK는 기본으로 적용되지만 그 외에 개발자가 추가하는 오픈소스 라이브러리나 구글의 androidx 라이브러리 등 SDK 라이브러리가 아닌것들은 모두 dependencies에 선언해야 함.

• 라이브러리 설정

```
dependencies {
    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.4.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.2'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

라이브러리 버전은 시기에 따라 책과 다를 수 있어요.

3. 앱 구성 파일 분석

- **메인환경 파일**
  - AndroidManifest.xml : 안드로이드 앱의 메인환경파일
  - URL이 http://schemas.android.com/apk/res/android로 선언되었다면 안드로이드 표준 네임스페이스

• 네임스페이스 선언

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.androidlab">
```

3. 앱 구성 파일 분석

메인 환경 파일

- <application> 태그는 앱 전체를 대상으로 하는 설정
- icon속성 : 이곳에 지정한 이미지가 앱을 설치한 사용자의 폰에 보이는 실행 아이콘
- label속성 : 앱의 이름을 등록
- theme 설정 : 앱에 적용해야하는 테마를 설정

• 네임스페이스 선언

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.AndroidLab">
    (... 생략 ...)
</application>
```

3. 앱 구성 파일 분석

메인 환경 파일

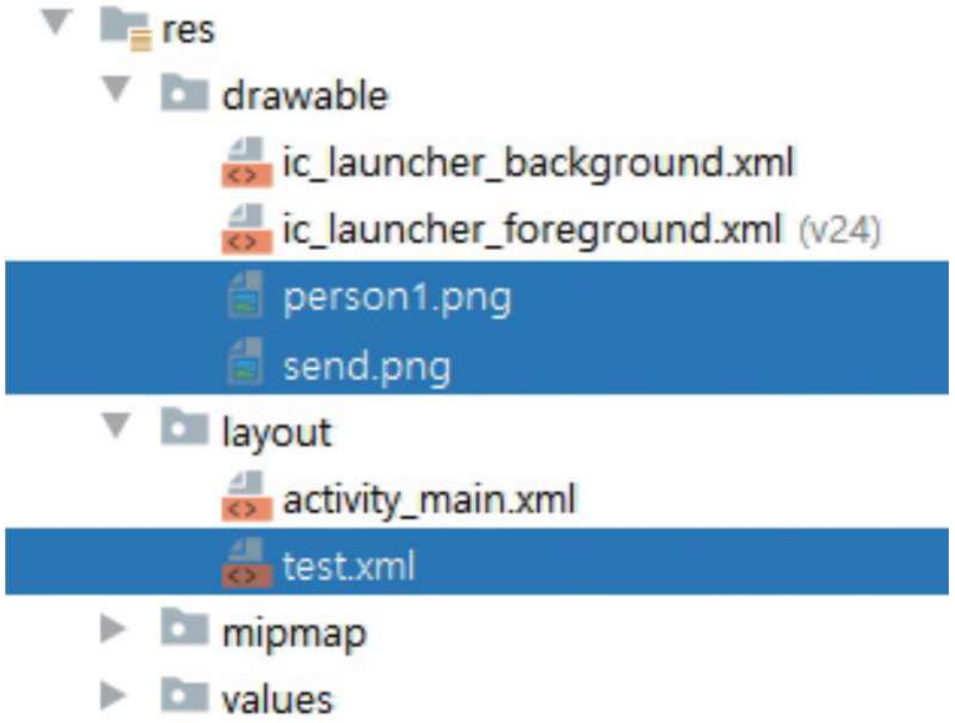
- 액티비티는 <activity> 태그로, 서비스는 <service> 태그로, 브로드캐스트 리시버는 <recei ver> 태그로, 콘텐츠 프로바이더는 <provider> 태그로 등록
- name 속성 : 클래스이름 등록
- <intent-filter> : 앱을 실행했을 때 처음 실행되는 액티비티라는 의미

• 액티비티 선언

```
<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>
```

3. 앱 구성 파일 분석

- **리소스 폴더**
  - drawable: 이미지 리소스
  - layout: UI구성에 필요한 XML 리소스
  - mipmap: 앱 아이콘 이미지
  - values: 문자열 등의 값으로 이용되는 리소스
  - 리소스를 식별하기 위한 int형 변수가 R.java 파일에 등록
  - res/layout/test.xml 파일이라면 R.layout.test라고 이용
  - res 하위의 폴더명은 지정된 폴더명을 사용해야 함.
  - 각 리소스 폴더에 다시 하위 폴더를 정의할 수는 없음
  - 리소스 파일명은 자바의 이름 규칙을 위배할 수 없음
  - 리소스 파일명에는 알파벳대문자를 이용할 수 없음



3. 앱 구성 파일 분석

- **레이아웃 XML 파일**
  - 화면을 구성하는 레이아웃 xml 파일

• 레이아웃 XML 파일

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. 앱 구성 파일 분석

■ 메인 액티비티 파일

- setContentView() 함수는 매개변수에 지정한 내용을 액티비티 화면에 출력
- R.layout.activity_main 지정했을 때 res/layout/activity_main.xml 파일에 구성한 내용을 화면에 출력

• 메인 액티비티 파일

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```