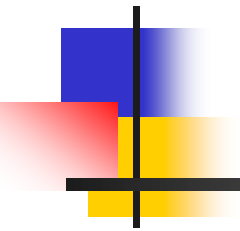
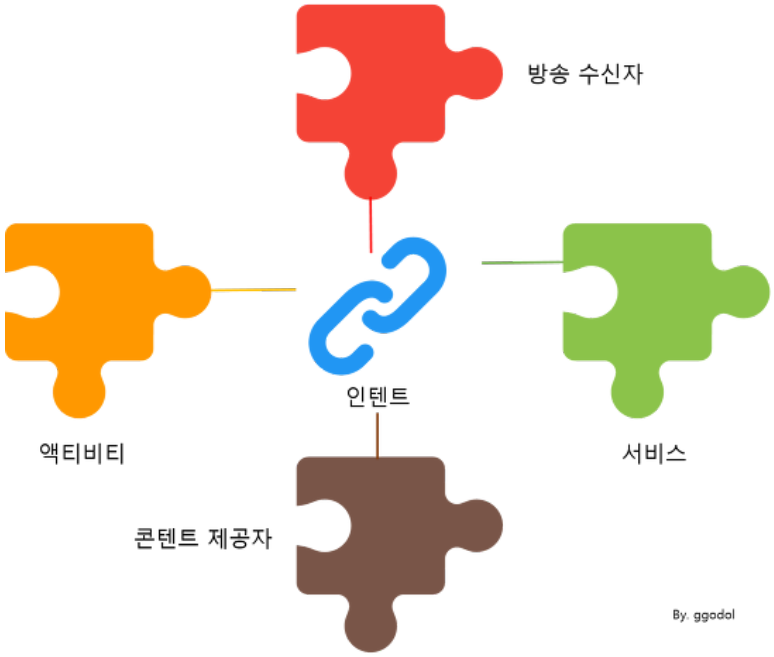


View 익히기 실습

배 희호 교수
경복대학교
소프트웨어융합과

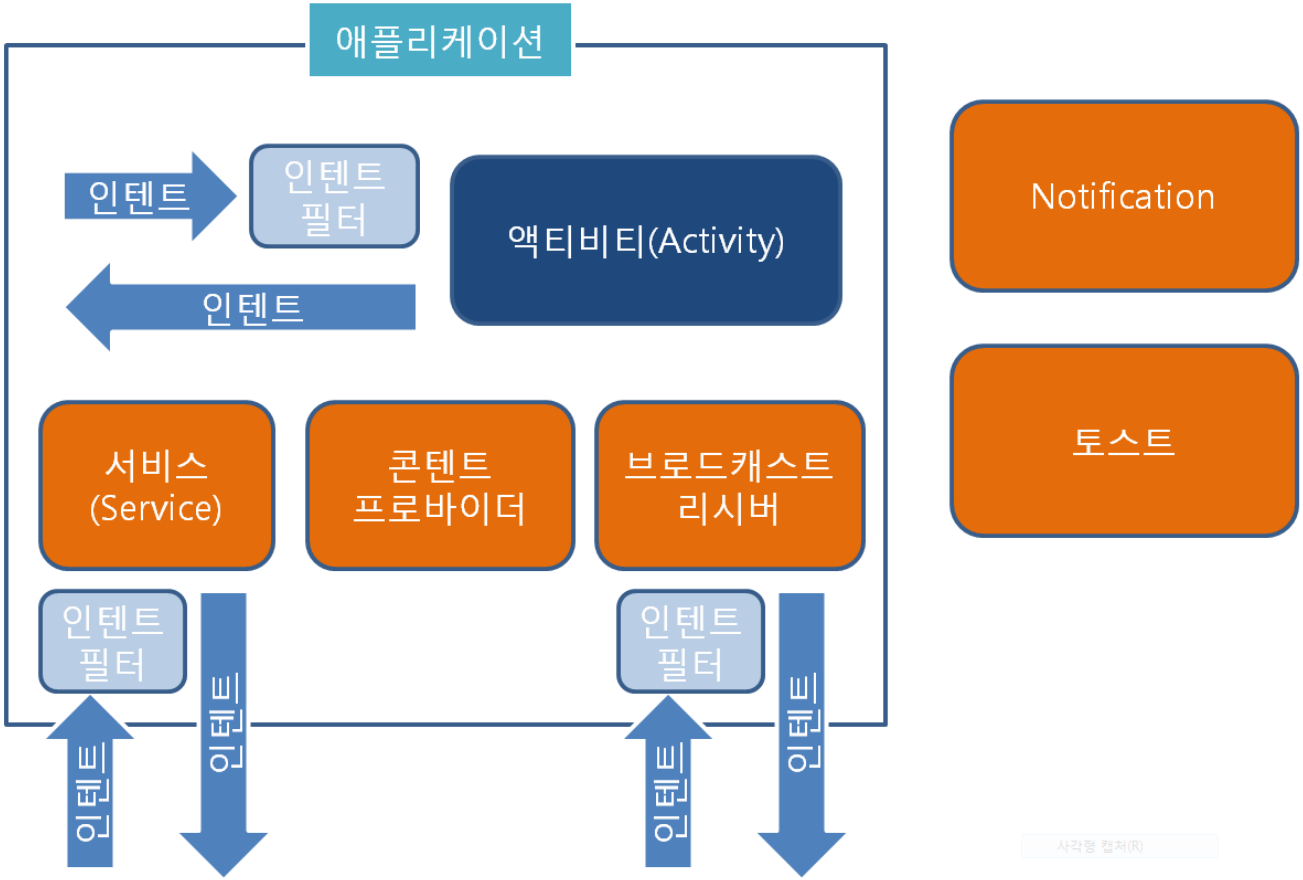
Component

- Android Application(애플리케이션)은 Component로 구성
  - Activity(액티비티)
  - Service(서비스)
  - BroadcastReceiver(방송 수신자)
  - Content provider(콘텐트 제공자)
- Intent를 통하여 다른 Application의 Component를 활성화시킬 수 있음



Component

- 각 Component들은 하나의 독립된 형태로 존재하며, 정해진 역할을 수행



Component

- Activity
  - 사용자 Interface 화면을 가지며 특정한 작업을 담당하는 Component
  - 일반적으로 UI를 갖는 하나의 Screen을 나타냄
  - Android Application은 반드시 하나의 Activity를 가지고 있어야 함
  - 각 Activity는 Manifests File에 등록되어 있어야 함
  - 하나 이상의 View를 가질 수 있음

Component

- 보통 UI에서 하나의 Screen
  - User Interface(UI) Component를 화면에 표시하고 System이나 사용자의 반응을 처리함
    - Application이 UI를 가진다면 하나 혹은 그 이상의 Activity를 가짐
    - 기존의 Activity는 같은 기능을 하는 새로운 것으로 대체될 수 있음
    - 개개의 Activity는 다른 Application에서 호출될 수 있음
- Activity는 일종의 기능들이 모여 있는 기본 단위
  - 하나의 클래스로 생성, 잘 정리된 하나의 단위
  - apk위의 Process에서 돌아감
  - Window와 연결됨

Component

- Activity의 예
  - Activity들이 모여서 Application이 됨

이메일 애플리케이션

이메일 애플리케이션은 3개의 액티비티로 이루어진다.

Component

- Service
  - Background에서 실행되는 Component로 오랫동안 실행되는 작업이나 원격 Process를 위한 작업을 할 때 사용
  - UI가 없음
  - 한번 시작된 Service는 Application이 종료되고 다른 Application으로 이동해도 계속 Background에서 실행
  - 모든 Service는 Service 클래스를 상속받아 작성
  - Network를 통하여 Data를 꺼내 올 수도 있음
  - 예) 배경 음악을 연주하는 작업

Component

- Service
  - Background에서 긴 작업을 처리
    - Activity와 거의 유사하게 동작함
  - Service 클래스 상속
    - onCreate(), onStart(), onDestroy()
  - Service가 IPC를 사용한 경우 AIDL 정의 필요
    - Android Interface Definition Language(AIDL)이 필요한 Code 자동 생성
  - Binder에 의해서 처리
  - UI가 없이 다른 Application의 요청을 처리할 때
  - Text-To-Speech Library

Component

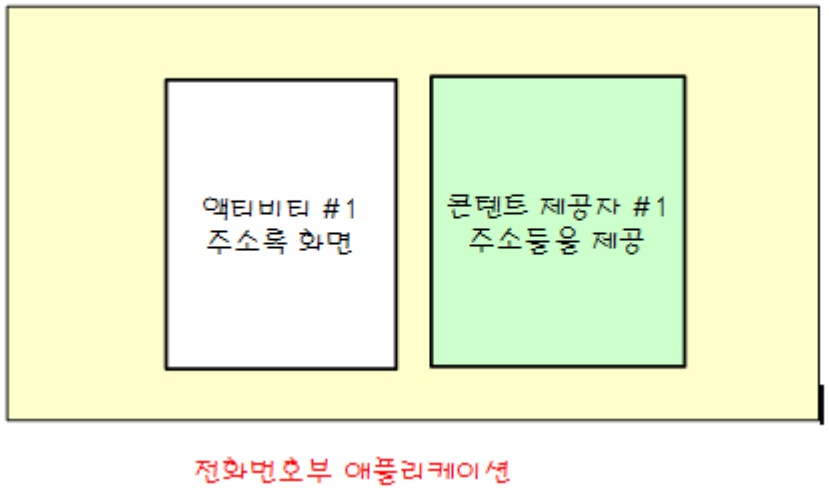
- BroadcastReceiver
  - Android 단말기에서 발생하는 다양한 Event/정보(방송)를 받고 반응하는 Component
  - 단말기에서 발생하는 일 중에서 Application이 알아야 하는 상황이 발생하면 Broadcast(방송)을 함
  - System Booting, Battery 부족, 전화/문자 수신, Network 끊김을 알려주는 것이 방송
  - BroadcastReceiver를 통해 상황을 감지하고 적절한 작업을 수행
  - 일반적으로 UI가 없음

Component

- 언제 BroadcastReceiver를 사용하는가?
  - Application이 System이 보내는 Event를 처리할 때
  - 예)
    - 특정 시간이 되었을 때
    - 위치 이동이 있을 때
    - 전화가 왔을 때
    - SMS가 왔을 때
- 어떻게 사용하는가?
  - BroadcastReceiver 상속
  - onReceive() 메소드 구현
  - Manifest의 Receiver에 intent-filter 기술

Component

- ContentProvider
  - Data를 관리하고 다른 Application Data를 제공하는 Component
    - Data는 File System이나 SQLite 데이터베이스, Web상에 저장될 수 있음
    - ContentProvider를 통해서 다른 Application의 Data를 Query(질의)하거나 변경 가능

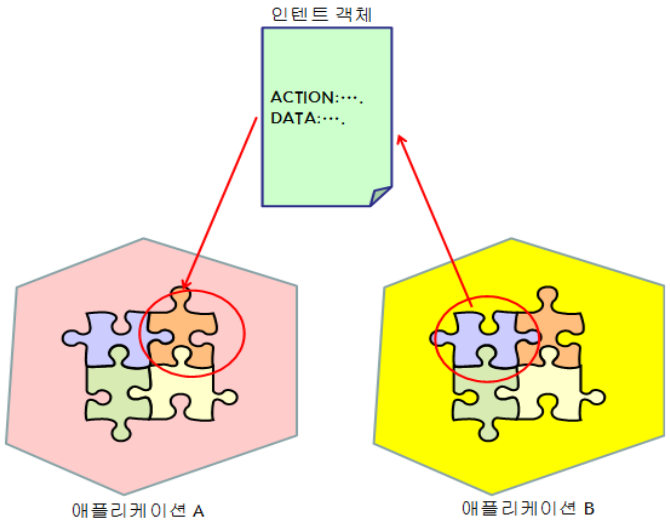


Component

- ContentProvider
  - Data를 처리하고 다른 Application에게 제공
    - 왜 ContentProvider를 사용하는가?
    - Android Application간의 Data 공유
    - Data에 접근하기 위한 표준 메소드의 정의
    - 어떤 형태의 Data Storage도 Backend로 사용될 수 있음
  - SQLite, Files, Memory Hash map, Remote Storage …
    - Data 관리를 위해 URI를 사용
    - 쿼리, 삽입, 수정, 삭제
  - Standard Prefix “content://”
    - 예) Contacts, MediaStore, Settings …

Component

- Intent
  - 서로 독립적으로 동작하는 4가지 Component들 간의 상호 통신을 위한 장치 (Component간의 통신 수단)
  - Intent를 통하여 다른 Application의 Component를 활성화시킬 수 있음
  - Application의 의도를 적어서 Android에 전달하면 Android가 가장 적절한 Component를 찾아서 활성화하고 실행

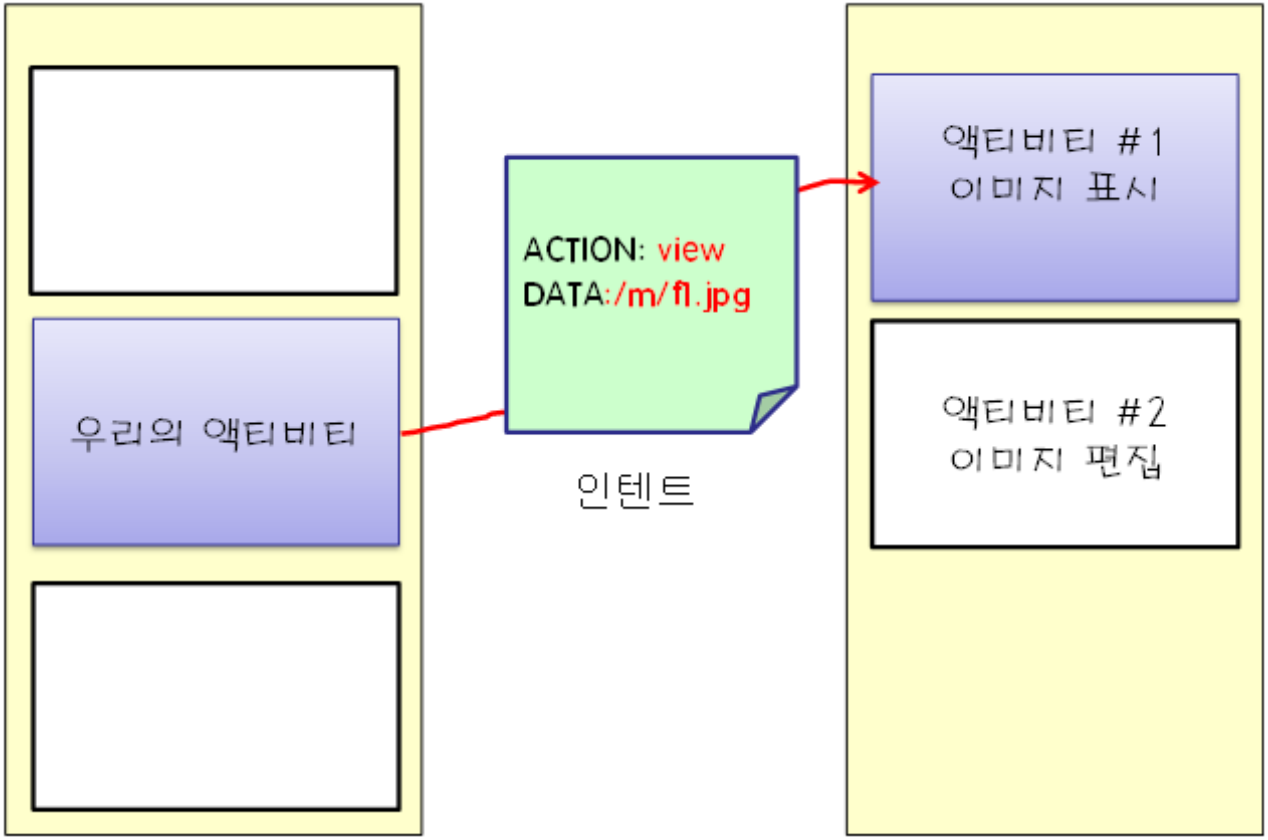


Component

- Intent
  - Message Object
    - 실행할 Operation을 가지고 있는 Object
    - Activity간의 연결고리 역할
    - Activity Manager가 해당 Intent를 처리
  - Intent는 여러 Data와 Action으로 구성
    - Action : VIEW, EDIT, DIAL…
      - Data : URI 형태로 되어있는 연관 Data
      - Category : 추가적인 정보 포함
      - 예) ACTION_DIAL, content://contacts/1
    - Data 중심의 경계 없는 Application 구조
      - ACTION_VIEW, http://www.google.com
  - 해당 Action을 처리하는데 App간 차별 없음
  - 사용자의 선택

Component

- Intent 사용의 예



Component

- Intent Filter
  - Activity, Service, BroadCast Receiver에 설정
    - 어떤 Intent를 처리할 것인지 결정

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Component

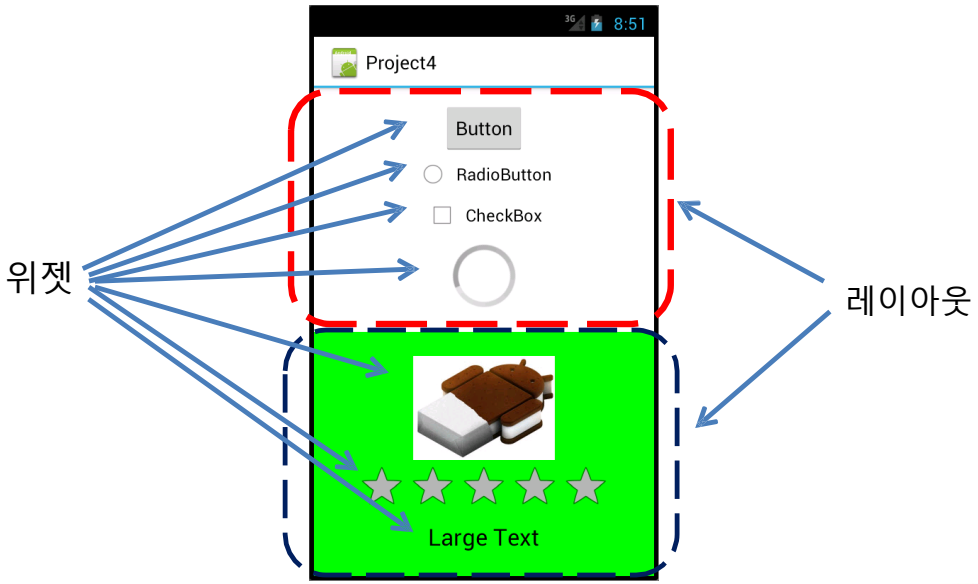
- Notification
  - 알림 메시지(Notification)는 알림 바(Notification Bar)나 알림 패널(Notification Panel), 소리, 진동, LED 점멸 등을 통해 사용자에게 특정 Event를 알릴 때 사용
  - 사용자 Interface가 없는 Service
  - Broadcast Receiver가 사용자에게 작업 완료 등을 알리기 위해 주로 사용

Component

- Toast
  - Toast 또한 Notification처럼 특정 Event를 사용자에게 알릴 때 사용하며, 주로 간단한 Message를 표시
  - Toast는 잠시 화면에 나타났다 사라지기 때문에 Toast를 통해 표시되는 내용은 Focus(표시되는 요소를 선택하는 동작)를 받을 수 없음

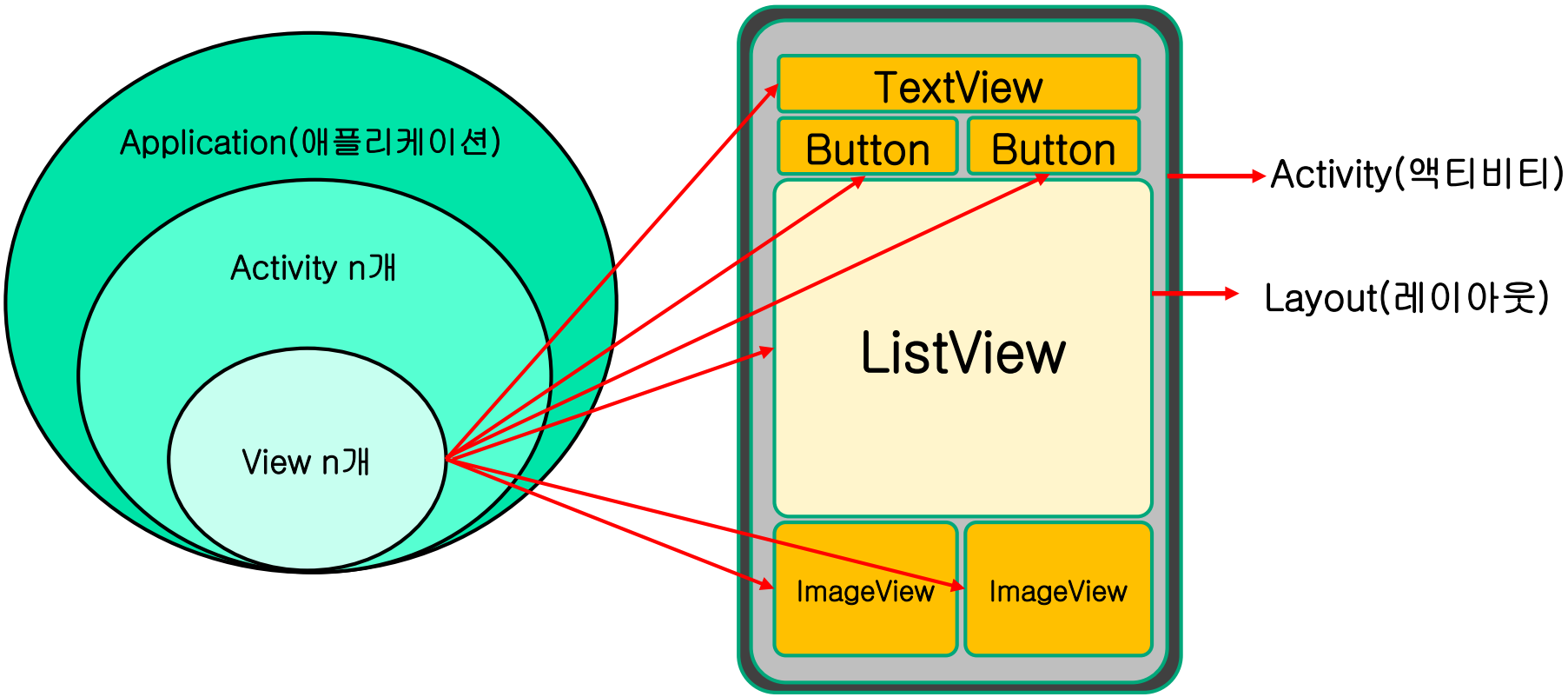
App 화면 구성 요소

- Android 화면에서 실제로 사용되는 것은 모두 View 클래스에서 상속
  - 이러한 클래스를 모두 'Widget'(또는 View)이라고 부름
- Widget중에서 다른 Widget을 담을 수 있는 것들은 주로 ViewGroup 클래스의 하위에 존재
  - 이렇게 다른 Widget을 담을 수 있는 클래스를 Layout이라고 함



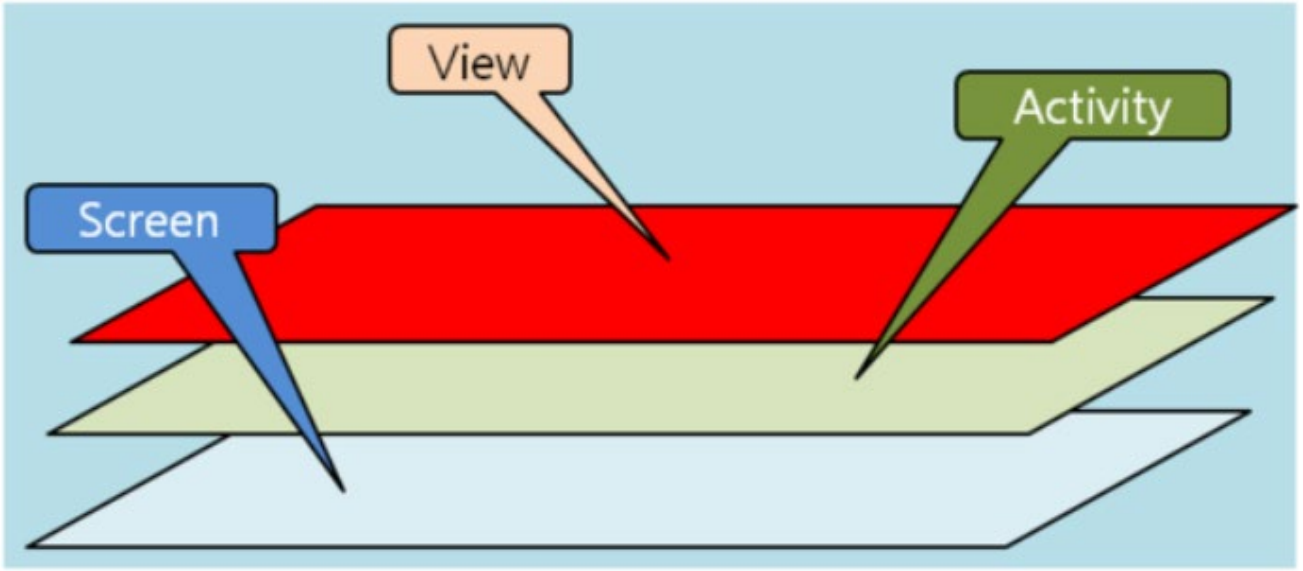
App 화면 구성 요소

- Activity와 View



App 화면 구성 요소

- Activity와 View

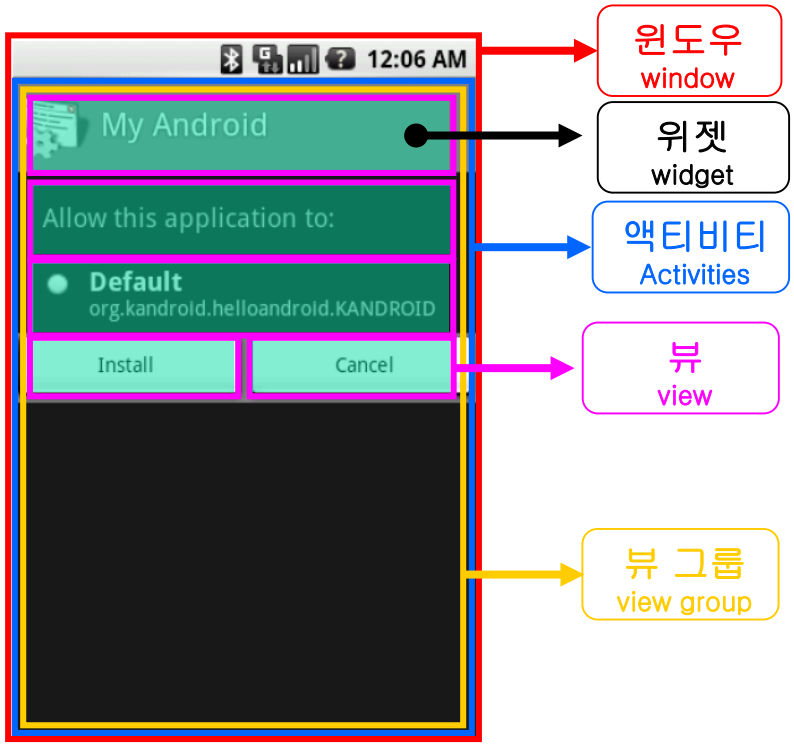


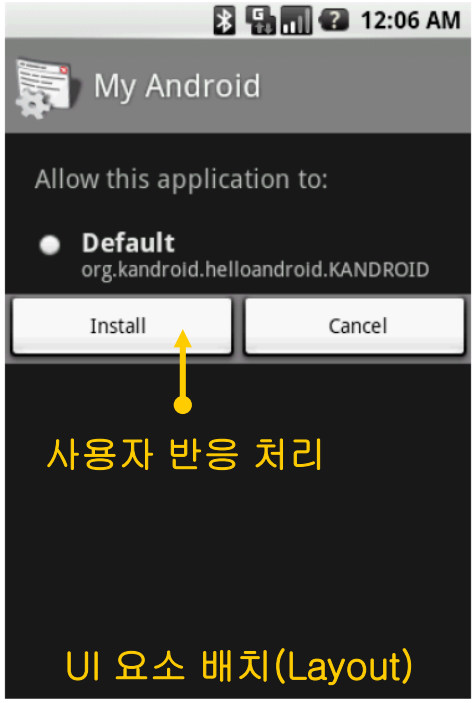
App 화면 구성 요소

- Window
  - 일반적으로 App의 화면
  - 그림을 그릴 수 있는 화면을 관리하고 사용자로부터의 입력(Touch, Key등)을 받을 수 있음
- Surface
  - Window가 가지고 있으면서 화면에 그림을 그릴 때 도화지와 같은 역할을 하는 객체
- View
  - 사용자의 눈에 보이는 App 화면의 구성 요소
  - View는 Window의 Surface를 이용하여 화면에 그림을 그리는 역할과 Window에 들어온 입력(Touch, Key등)을 어떻게 처리할 것인지에 대한 기능을 구현하고 있는 객체
  - Button, TextView, EditText 등은 모두 View를 상속하여 구현

App 화면 구성 요소

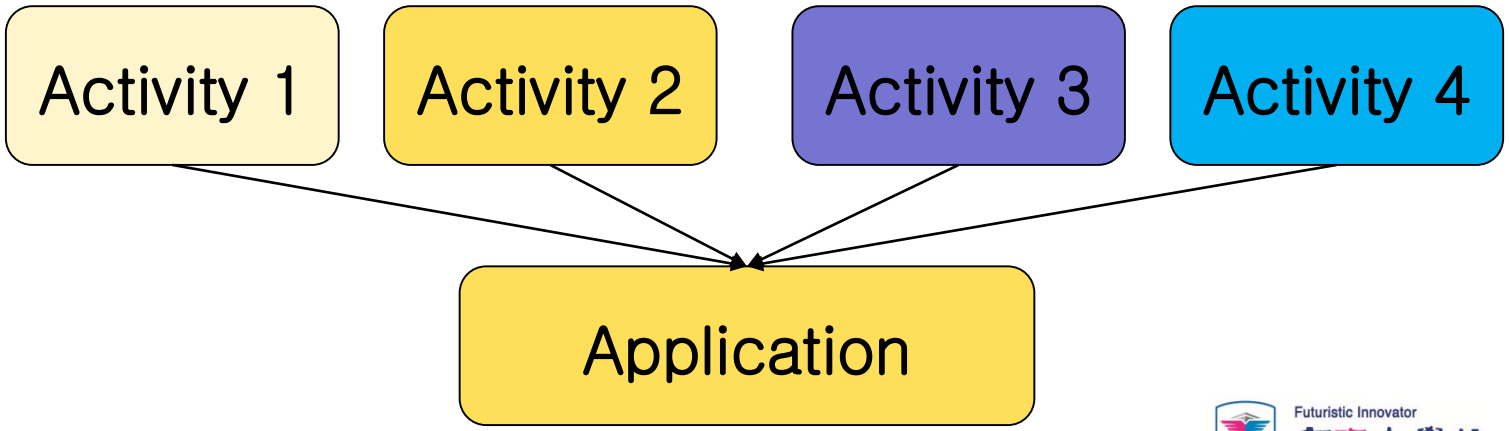
- Activity
  - Application 하나의 Screen 또는 화면을 일컫는 말
  - UI Component를 화면에 표시하고, System이나 사용자의 반응을 처리할 수 있음
  - Windows의 창의 개념과 유사함





App 화면 구성 요소

- Activity
  - Android Application의 화면을 구성하는 주요 단위인 Activity는 화면에 직접적으로 보이지 않으며, Activity 안의 View가 사용자를 대면하는 실체임
  - Application이 UI를 가진다면 하나 혹은 그 이상의 Activity를 가지며 기존의 Activity는 같은 기능을 하는 새로운 것으로 대체 될 수 있음
  - 여러 개의 View가 모여 하나의 Activity를 구성하고, 이러한 Activity가 모여 하나의 Application이 됨

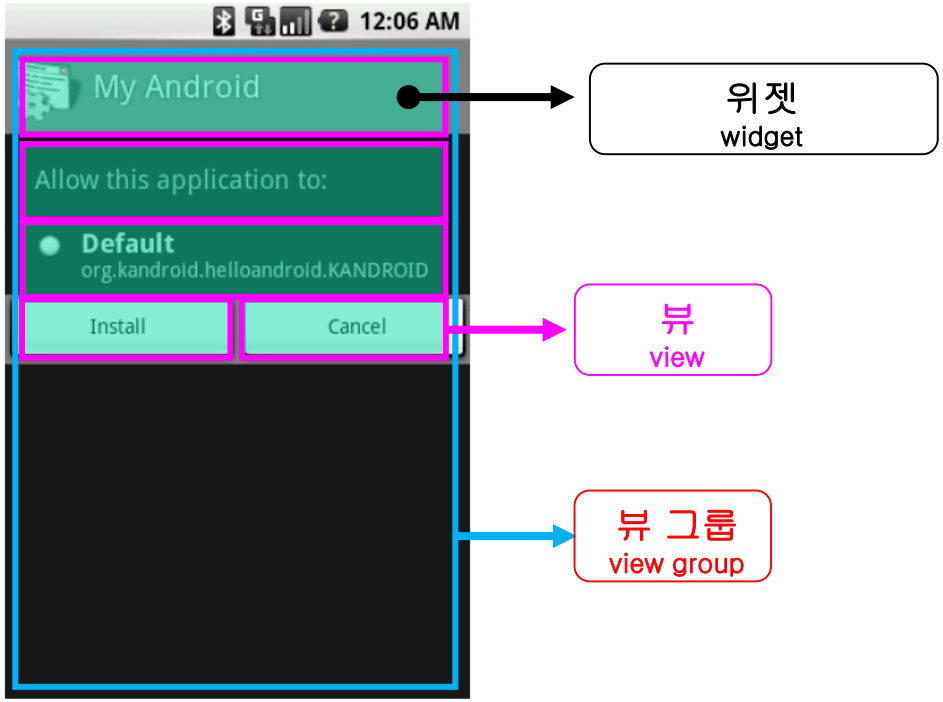


App 화면 구성 요소

- Widget
  - 직접적으로 보이며 사용자 Interface를 구성하며 Control 이라고도 함
  - 사용자의 입력을 받거나 화면에 Data를 표시해주는 것들
- ViewGroup
  - View를 담는 Container 역할을 하며 이 부류의 클래스들을 Layout이라고 함
- 출력 설정
  - Activity 클래스에서 setContentView(View의 id 또는 View 객체)에 Activity의 Main View를 설정하며 Main View안에 다른 View를 배치
- View는 원칙적으로 Layout XML File에서 생성 가능하지만, JAVA Code로도 생성 가능함

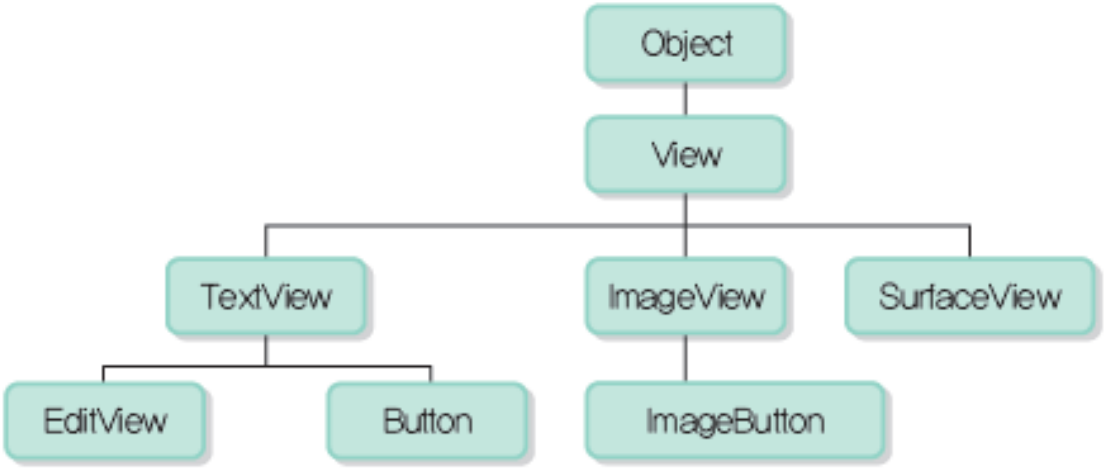
App 화면 구성 요소

- Android Application의 가장 기본적인 구성 단위는 Activity라는 클래스
- Activity는 한 화면을 나타내지만 그 자체로는 아무 것도 보여줄 수 없고, Activity에 View와 ViewGroup 클래스를 사용해야만 비로소 화면에 무엇인가를 표시할 수 있음



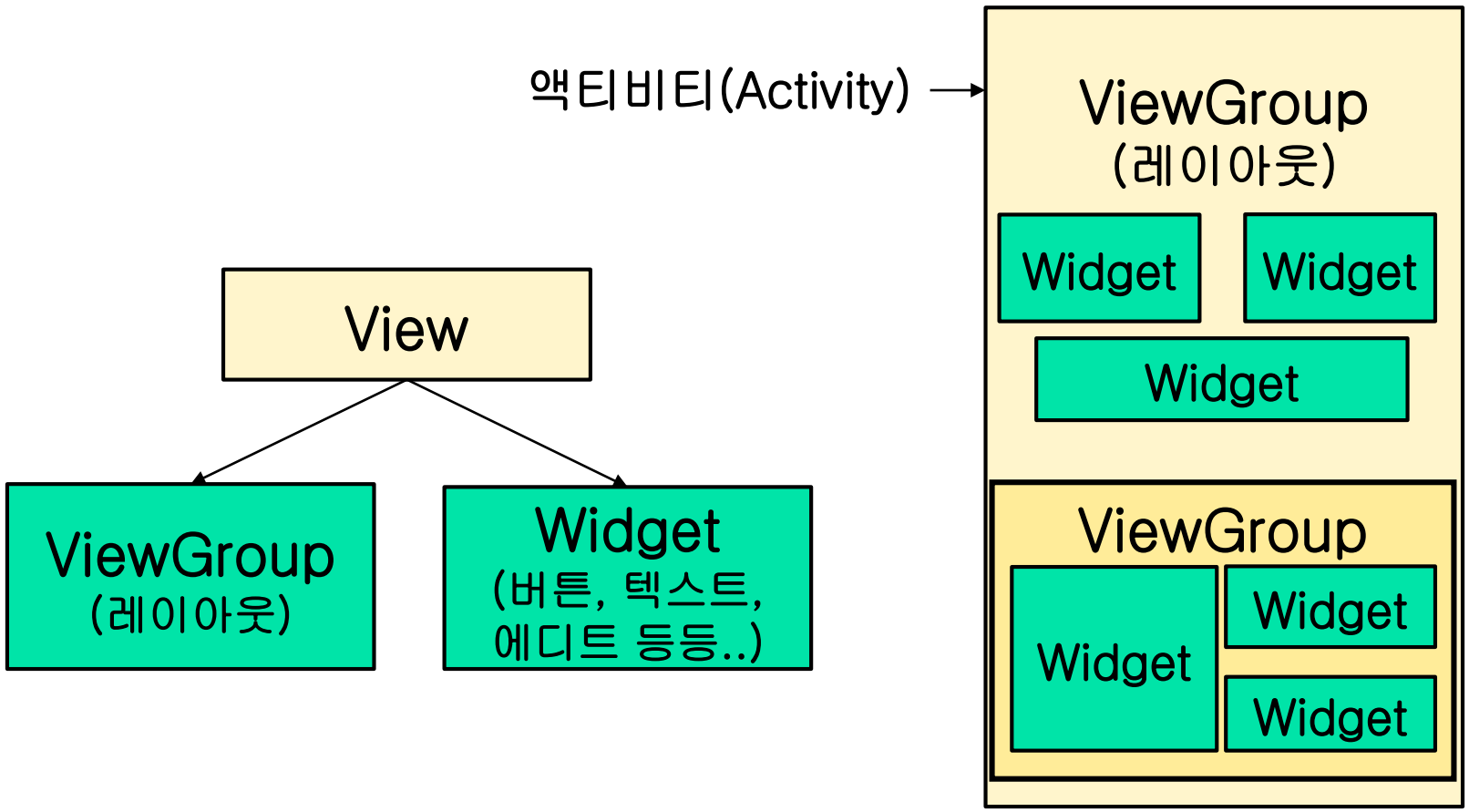
App 화면 구성 요소

- View
  - View는 Button, TextView 뿐만 아니라 이것들을 포함하는 눈에 보이지 않는 영역을 포함
  - 일반적으로 Control, Widget이라고 불리는 UI 구성 요소
  - View 클래스는 모든 View들의 부모 클래스
  - View 클래스가 가지고 있는 Field나 Method는 모든 View에서 공통적으로 사용할 수 있음



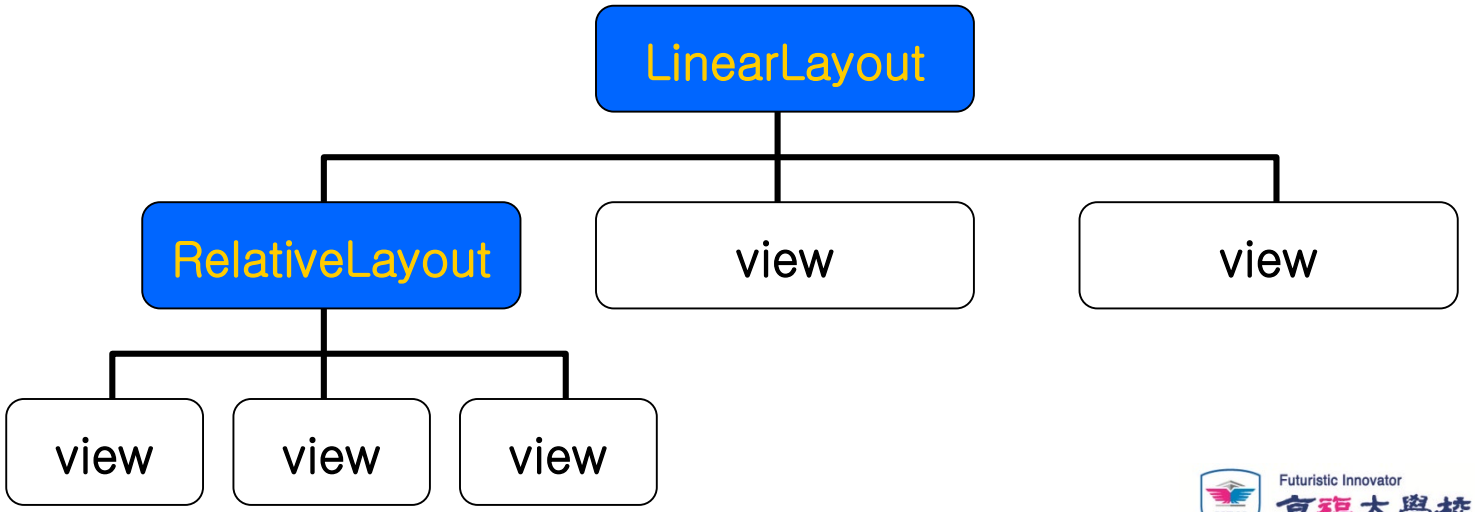
App 화면 구성 요소

- View와 ViewGroup



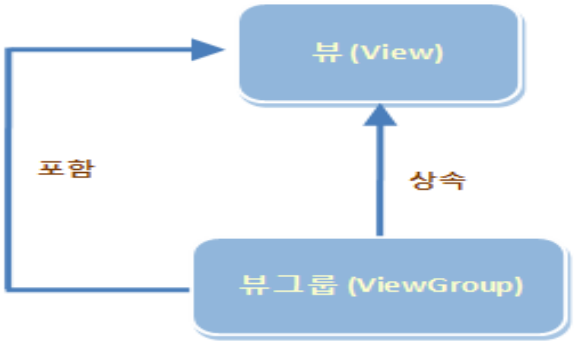
App 화면 구성 요소

- ViewGroup
  - View를 담는 Container 역할을 하며, 이 부류의 클래스들을 Layout이라고 함
  - 여러 Widget(View)을 포함할 수 있도록 확장(extends)한 것으로 ViewGroup도 View
  - ViewGroup에서 상속받아 Layout 클래스를 선언하므로 Layout 클래스도 View
- ViewGroup 구성



App 화면 구성 요소

- Layout
  - Widget들을 어떠한 방식으로 화면에 배치해줄지를 결정해주는 Container 역할
  - View의 효율적인 배치를 위해서는 Layout의 개념과 종류를 이해해야 함
  - Layout은 Activity안에 View를 배치하는 기법을 말함
  - View는 Composite pattern 구조로 이루어져 있음



뷰와 뷰 그룹의 관계

App 화면 구성 요소

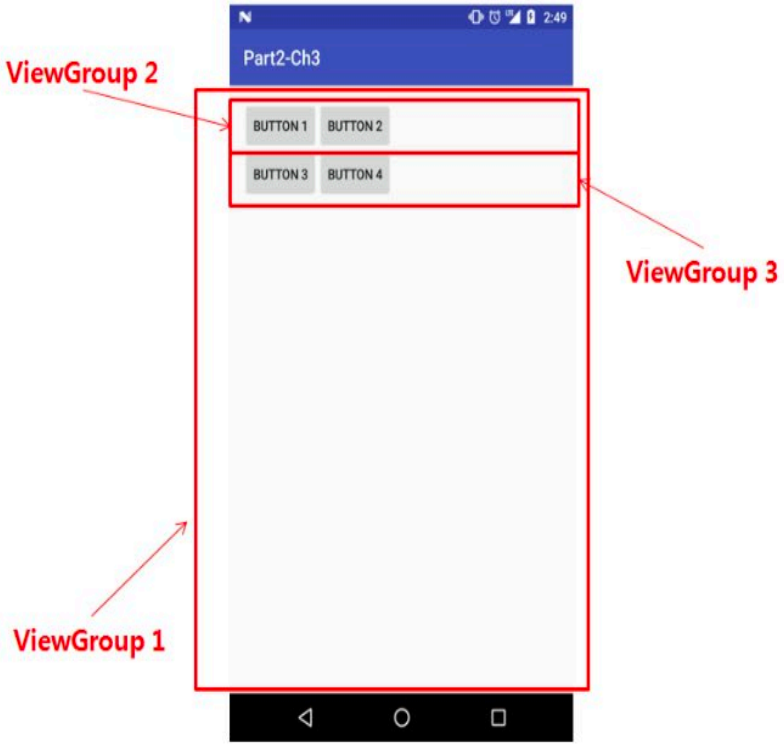
- Widget과 View
  - 앞으로 Widget과 View를 서로 차이점을 가지고 부르게 될 것인데, 쉽게 말 하자면 Widget과 View의 차이는 부르는 위치에 따라 다르다는 것
  - Widget
    - 화면에서 보이는 Button이나 Layout, CheckBox 등등을 사용자 입장에서 부를 때 Widget이라고 함
  - View
    - 개발자 입장에서 Widget에 해당하는 Code를 View라고 함

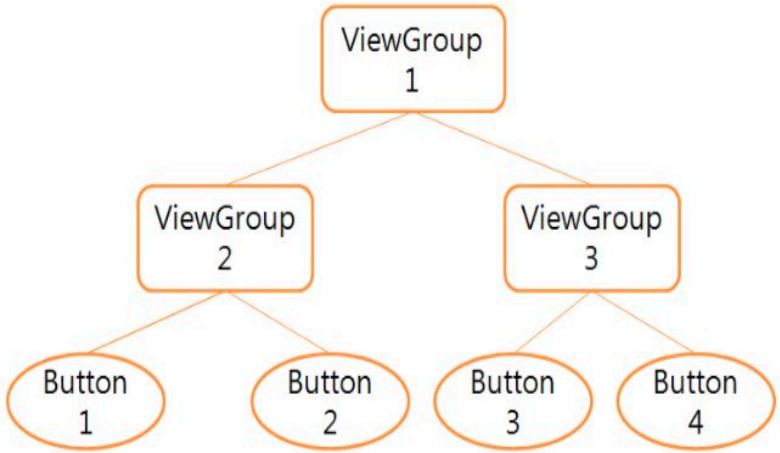
View Architecture

- View의 계층구조
  - ViewGroup이면서도 Widget처럼 사용되기도 하는 클래스도 있으며, 특정 Widget을 상세히 알고 싶다면 그 Supper 클래스들부터 연구해야 함
    - Sub 클래스는 Supper 클래스의 모든 속성을 상속받음
- Widget의 계층
  - View도 JAVA 클래스의 일종이므로 Root인 Object로부터 파생
  - View로부터 직접 파생되는 모든 클래스가 바로 Widget이며 스스로를 그릴 수 있는 능력을 가지고 있음

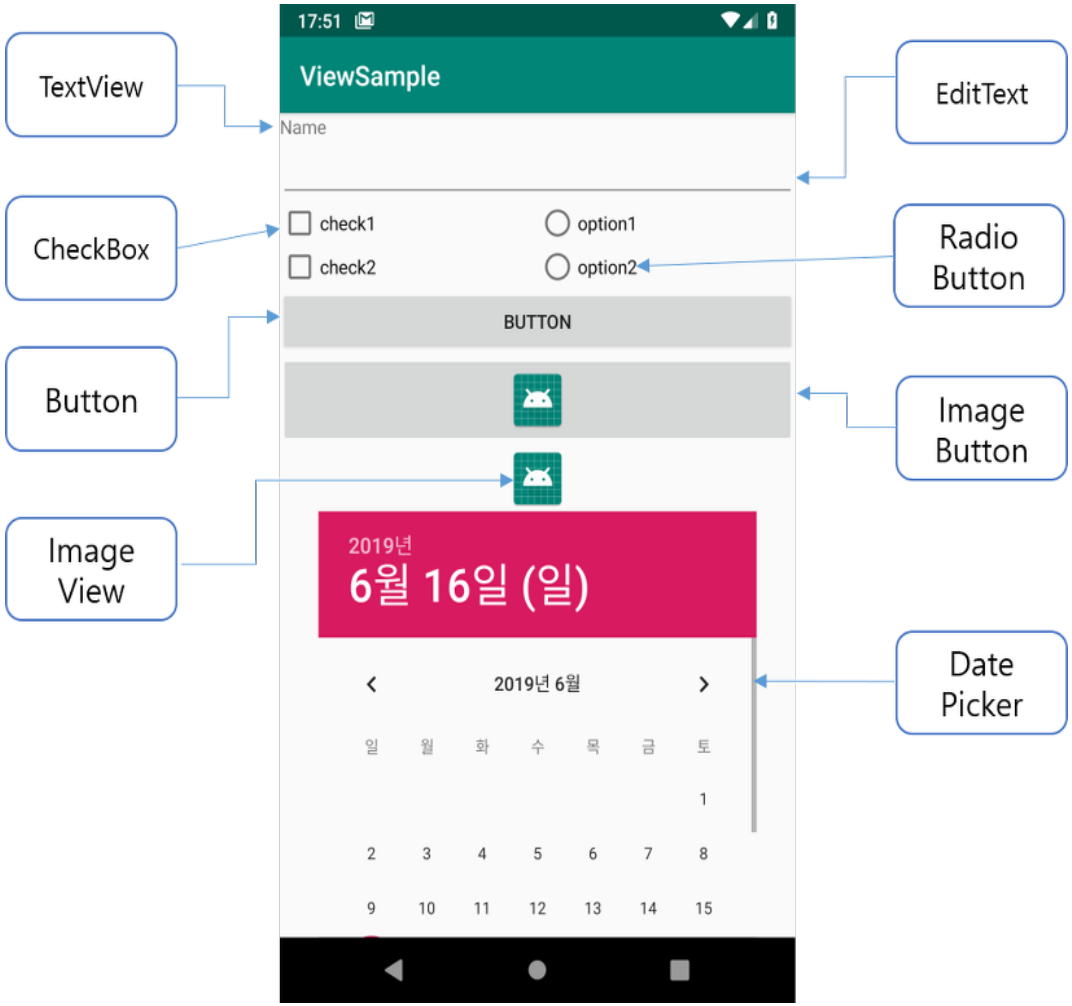
View Architecture

- View Group의 계층
  - View로부터 파생된 ViewGroup의 Sub 클래스
  - 다른 View들을 차일드로 포함하며 차일드를 정렬하는 기능을 가짐





View Architecture



View 속성

- View의 속성 설정
  - XML에서 설정
    - XML에서 View를 생성해서 설정할 수 있고 Layout 창에서 생성해서 속성 창에서 입력이 가능
  - JAVA Code 상에서 직접 설정 가능
    - JAVA Code에서는 set- 메소드를 이용해서 설정하고, get- 메소드를 이용해서 값을 반환

View 속성

- layout_width, layout_height 속성
  - View의 폭(width)과 높이(height)를 지정하며, 수평, 수직 각 방향에 대해 크기를 지정할 수 있음
  - 속성값으로 아래의 3가지 중 하나의 값을 지정
    - match_parent, fill_parent
      - 부모의 주어진 크기를 다 채움
    - wrap_content
      - 내용물의 크기 만큼만 채움
    - 정수 크기
      - 지정한 크기에 맞춤
      - 지정한 크기가 액면대로 다 받아들여지지 않으며, 주위 다른 Widget들의 크기에 영향을 받음
      - 명시적인 크기 지정 시 정수 하나와 단위를 지정하는 예약어를 같이 사용하며, 이 단위는 크기를 지정하는 모든 속성에 공통적으로 적용

View 속성

- fill_parent와 match_parent의 차이점
  - layout_width나 layout_height에 지정할 수 있는 속성이 Android 2.1까지는 wrap_content와 fill_parent만 존재했지만, Android 2.2부터는 match_parent가 추가됨
  - 이 속성이 추가된 이유는 기존에 존재하던 fill_parent가 실제로 부모 View를 가득 채우는 것이 아닌 부모 View의 내부 여백(padding)을 제외한 만큼만 가득 채우기 때문임
  - 그래서 fill_parent 대신에 사용할 수 있는 좀 더 직관적인 match_parent를 추가한 것
  - 두 속성이 같으므로 어느 것을 사용해도 무방하다고 생각할 수 있지만 fill_parent는 비 권장 속성이므로 match_parent를 사용하는 것이 맞음

View 속성

- View의 크기 지정에 사용되는 단위

단위	설명
px(pixels)	✓ pixel을 의미 ✓ 화면의 실제 pixel을 나타냄 ✓ 화면의 밀도와는 상관없는 치수 ✓ pixel은 권장되는 단위는 아닌데 그 이유는 장치마다 화면의 밀도가 다르기 때문임
dip(dp)	✓ dip는 Density-Independent Pixel의 약자 ✓ 여러 밀도의 화면에서 일정한 크기를 보여줄 수 있도록 제공되는 치수 (dip를 줄여서 dp라고도 씀) ✓ pixel과 dip의 관계는 pixels = dips * (density/160)로 정의 ✓ 160dpi 화면에서는 1dip는 1pixel이고, 240dpi 화면에서는 1dip가 1.5pixel이 됨 ✓ dp로 View의 크기를 지정하면 화면의 밀도가 다르더라도 항상 동일한 크기로 표시
sip(sp)	✓ sp는 Scale-independent Pixel의 약자 (dp와 유사한 기능) ✓ 사용자 설정에 따라 변경되는 Text 크기를 반영함 ✓ 이 단위는 font 크기를 지정하는 경우에 추천

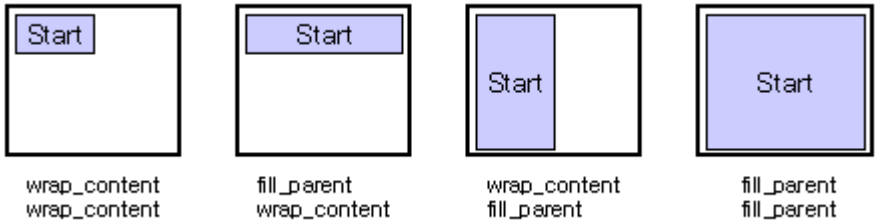
View 속성

- View의 크기 지정에 사용되는 단위

단위	설명
pts	✓ 포인트(points)의 약자 ✓ 1pts는 1/72인치
in	✓ 인치(inches)의 약자 ✓ 1인치로 된 물리적 길이
mm	✓ 밀리미터(millimeters)의 약자 ✓ 1밀리미터로 된 물리적 길이
em	✓ Text 크기 ✓ 글꼴과 상관없이 동일한 Text 크기 표시

View 속성

- layout_width, layout_height 속성
  - 예) “Start”라는 Caption을 가지는 Button 배치



View 속성

- id 속성
  - View의 ID를 지정함
  - XML Layout에서 정의한 View를 JAVA Code에서 참조하는 데 사용
  - XML Layout안에서 다른 View를 참조하는 데 사용하므로 유일하고, 직관적인 이름을 사용하는 것을 권장
  - 형식 : @[+]id/ID
    - @ : id를 Resource(R.java)에 정의하거나 참조한다는 뜻이며 무조건 붙임
    - + : id를 새로 정의한다는 뜻이며 참조 시는 생략
    - id : 예약어
    - / : 뒤에 원하는 이름을 작성하되 사용자 정의 규칙에 맞아야 하며 Activity안에서의 중복 불가
  - 예) android:id="@+id/name" : Text View에 name이라는 id를 부여

View 속성

- id 속성
  - XML 문서에 id를 지정하면 이 이름이 R.java에 정수형 상수로 선언됨
  - Activity나 View 객체가 자신에게 속한 View를 참조할 때 findViewById() 메소드 호출
    - 인수로 참조할 View의 id를 전달하는데 자신의 Main View 안에 속해있는 View만 호출 가능
  - 모든 View에 id를 의무적으로 지정할 필요는 없으며 Code에서 참조할 필요가 없는 Widget은 생략 가능

View 속성

- background 속성

[Format]
#RGB
#ARGB
#RRGGBB
#AARRGGBB

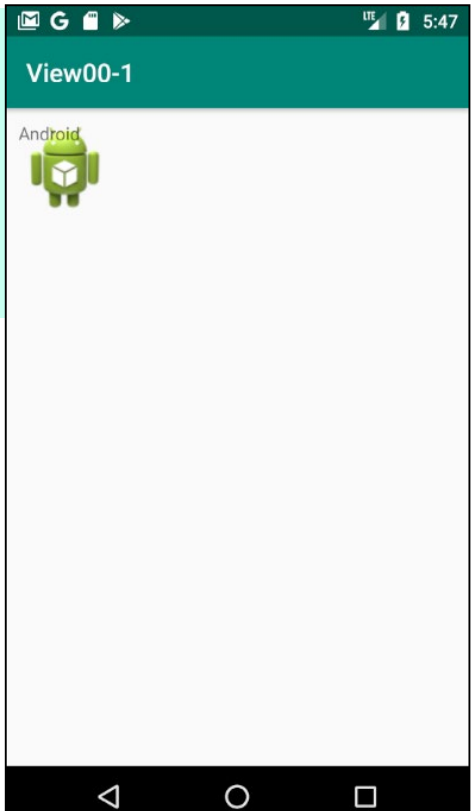
[배경색에 알파값을 적용하여 투명도를 조절하는 경우]

- XML Layout에서 색상을 지정할 때는 '#' 기호를 앞에 붙인 후, ARGB(A : Alpha, R : Red, G : Green, B : Blue)의 순서대로 색상의 값을 기록함
- 16진수 값을 지정할 때는 4가지 Format이 적용되며, 배경뿐만 아니라 색상을 지정하는 모든 속성에 적용
- 예) #ff0000 (#f00): 빨간색, #0000ff : 파란색

View 속성

- background 속성
  - View의 배경을 지정하며, 색상 및 Image등의 여러 가지 객체로 지정 가능
  - 예) background Image 사용
    (/res/drawable-00 폴더에 이미지 저장)

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Android"
    android:background="@drawable/ic_launcher"/>
```



View 속성

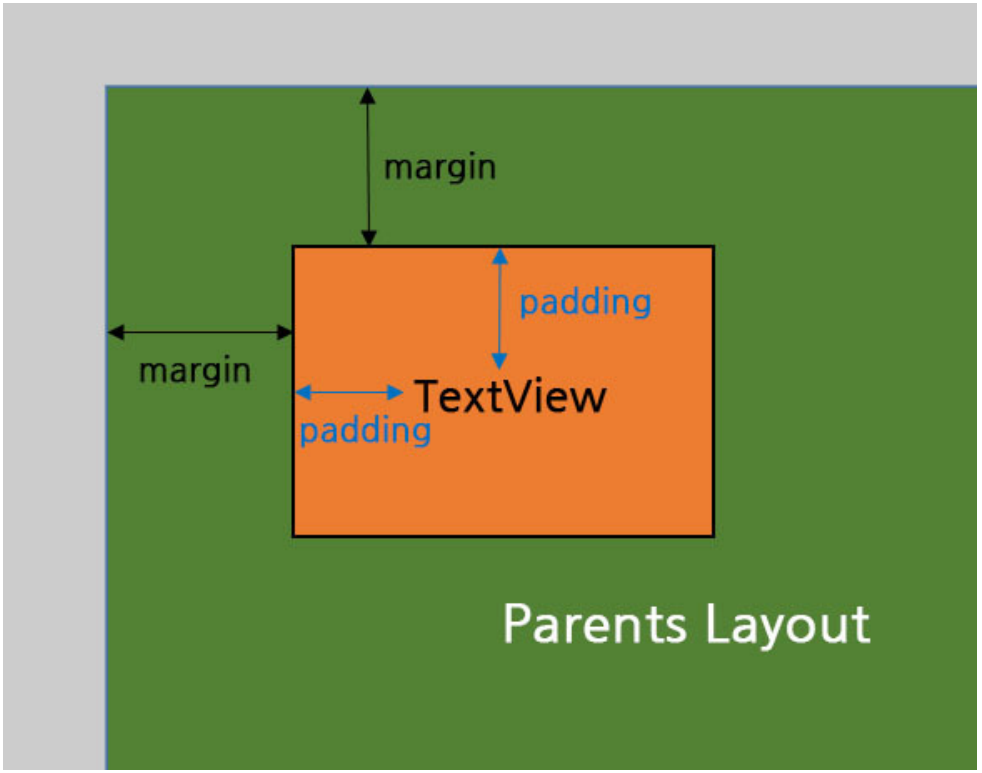
- padding 속성
  - Widget의 경계선으로부터 Widget안의 요소가 떨어지도록 설정
  - View와 내용물 간의 간격을 지정 (안쪽 여백)
  - padding 속성 값으로 4 방향의 여백을 조절할 수 있음
  - 속성값
    - padding – 4 방향에 동일한 여백이 적용
    - paddingLeft – 왼쪽 변에 대해 여백이 적용
    - paddingRight – 오른쪽 변에 대해 여백이 적용
    - paddingTop – 위쪽 변에 대해 여백이 적용
    - paddingBottom – 아래쪽 변에 대해 여백이 적용

View 속성

- margin 속성
  - View와 View사이의 간격을 지정(바깥쪽 여백)
  - margin에 하나의 값을 지정하면 4 방향에 대한 여백을 동시에 설정
  - 속성값
    - layout_margin
    - layout_marginLeft
    - layout_marginRight
    - layout_marginTop
    - layout_marginBottom

View 속성

- margin과 padding의 차이점
  - Android 개발 시 Widget을 위치할 때, margin과 padding을 사용하여 임의의 수치만큼 여백을 둘 수 있음



View 속성

```
<LinearLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:background="#00FF00"
    android:layout_margin="10dp"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="100dp"
        android:text="superdroid"
        android:layout_marginLeft="100dp"
        android:paddingLeft="100dp"
        android:background="#FFFF00"/>
</LinearLayout>
```

View 속성

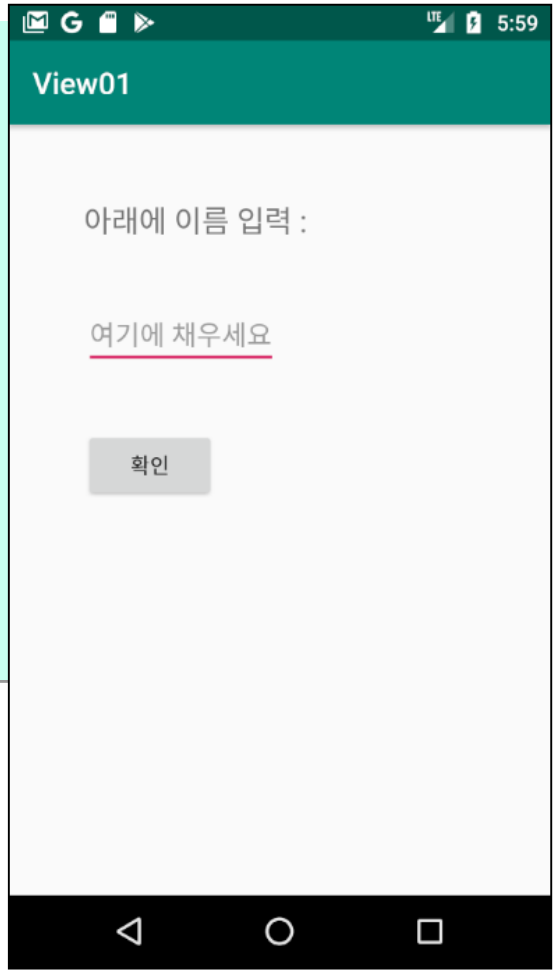
- layout_margin
  - Widget과 Widget 사이에 여유를 두고 싶다면 layout_margin 속성을 사용

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="30dp"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:textSize="20dp"
        android:text="아래에 이름 입력 : "/>
```

View 속성

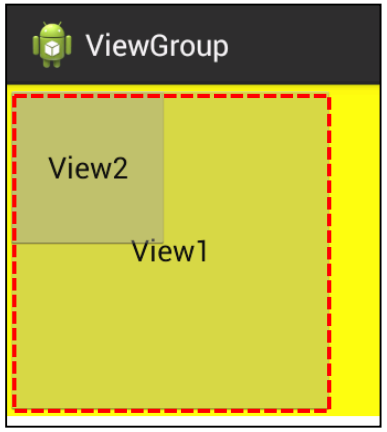
- layout_margin

```
    <EditText
        android:layout_margin="20dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="여기에 채우세요"/>
    <Button
        android:layout_margin="20dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="확인"/>
</LinearLayout>
```

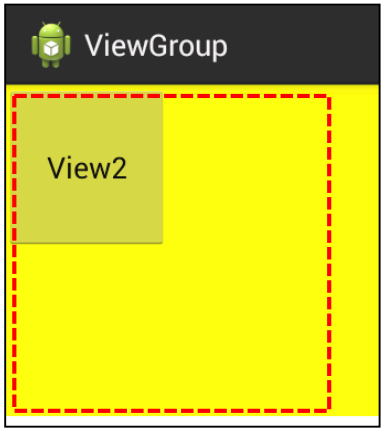


View 속성

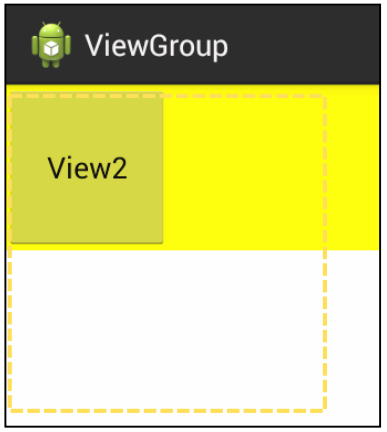
- visibility 속성
  - View의 표시 유무를 지정
  - 이 속성을 지정하면 Design 시기에 숨겨두었다가 실행 시 필요한 때만 보이도록 할 수 있음
  - 속성값(3 가지)
    - visible : 보이는 상태임
    - invisible : 숨겨진 상태이되 자리는 차지함
    - gone : 숨겨지며 자리도 차지하지 않음



android:visibility= "*visible*"



android:visibility= "*invisible*"



android:visibility= "*gone*"

View 속성

- clickable, longClickable 속성
  - Click Event를 받을 것인지, Long Click Event를 받을 것인지를 지정
    - Click
      - 손가락으로 View를 누름
    - LongClick
      - 손가락으로 View를 누른 채 잠시 기다림
  - boolean Type이므로 true 또는 false 둘 중 하나의 값을 지정

- enabled 속성
  - Widget의 동작 여부

View 속성

- focusable 속성
  - Keyboard Focus를 받을 수 있는지를 지정
  - default 값으로 false가 설정되어 있으며 필요 시 속성을 true로 변경
  - EditText나 Button처럼 사용자의 입력이 필요한 클래스는 default로 true가 지정되어 있음
- LayoutParams
  - LayoutParams는 View의 자체의 속성이 아니라 View가 배치되는 Layout에 따라 달라지는 속성
  - LayoutParams로 설정된 속성은 View가 아니라 View가 배치된 Layout에서 사용함
- layout_gravity
  - View의 정렬 값

View 속성

- rotation 속성
  - Widget을 회전시켜서 출력
  - Android 3.0 부터 지원됨
  - 속성값은 각도(degree)로 지정

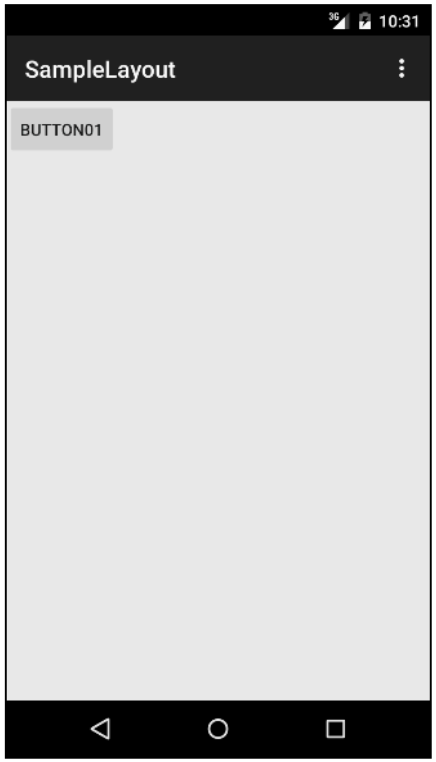
View 속성

```
<LinearLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10dp"
    android:orientation="vertical">
    <TextView
        android:id="@+id/id_test_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="안녕 안드로이드" />
</LinearLayout>
```

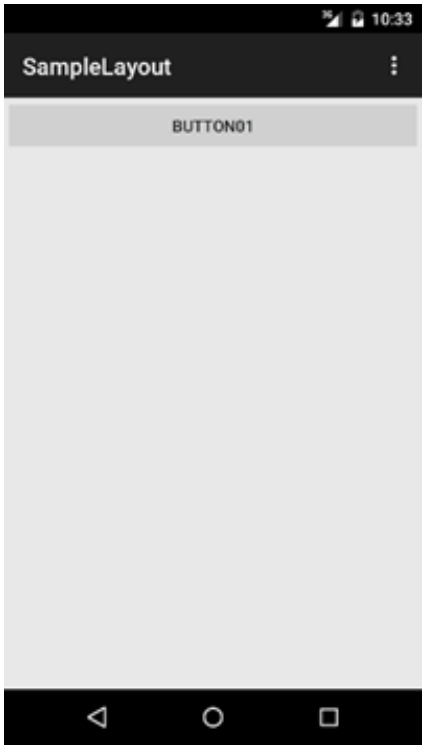
```
public class MainActivity extends AppCompactActivity{
    @Override
    protected void onCreate( Bundle savedInstanceState )    {
        super.onCreate( savedInstanceState );
        setContentView(R.layout.activity_main);

        TextView textView = findViewById( R.id.id_test_view );
        textView.setText( "Hello World!" );
    }
}
```

View 속성

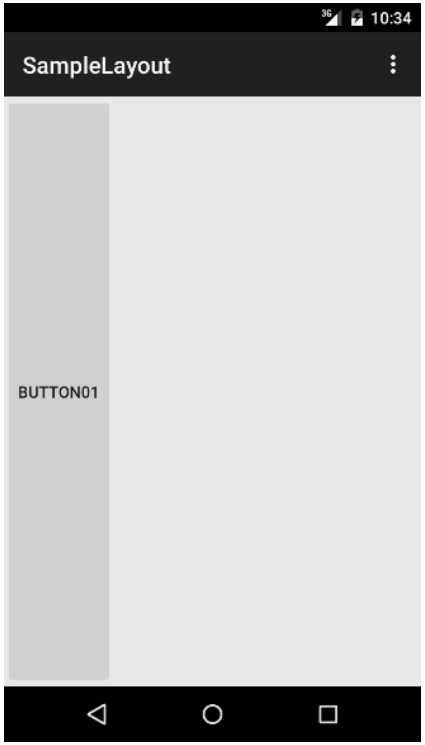
- match_parent와 wrap_content 값을 폭과 넓이에 적용한 예



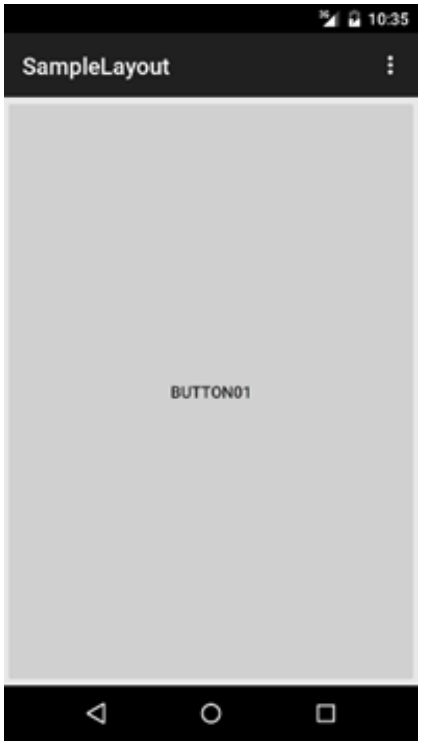
wrap_content
wrap_content



match_parent
wrap_content



wrap_content
match_parent



match_parent
match_parent

View 속성

- gravity 속성

속성값	내용
center	해당 위젯을 정중앙에 위치 시킨다 이때 사이즈의 변화는 없다
fill	해당 위젯을 가로/세로 길이를 부모 뷰 그룹의 사이즈에 맞게 늘려 채워준다
clip_vertical	해당 위젯의 세로 길이가 부모 뷰 그룹보다 클 경우 넘어서는 부분은 잘라낸다.
clip_horizontal	해당 위젯의 가로 길이가 부모 뷰 그룹 보다 클 경우 넘어서는 부분은 잘라낸다
start	해당 위젯을 부모 뷰그룹의 시작점에 위치 시켜 준다 이때 사이즈의 변화는 없다
end	해당 위젯을 부모 뷰그룹의 마지막에 위치 시켜 준다 이때 사이즈의 변화는 없다

View 속성

- gravity

속성값	내용
top	해당 위젯을 윗쪽에 위치 시킨다 이때 사이즈에 변화는 없다
bottom	해당 위젯을 하단에 위치 시킨다 이때 사이즈에 변화는 없다
left	해당 위젯을 왼쪽에 위치 시킨다 이때 사이즈에 변화는 없다
right	해당 위젯을 오른쪽에 위치 시킨다 이때 사이즈에 변화는 없다
center_vertical	해당 위젯을 세로 중앙에 위치 시킨다 이때 사이즈에 변화는 없다
fill_vertical	해당 위젯의 세로를 부모 뷰 그룹의 사이즈에 맞게 늘려 채워준다
center_horizontal	해당 위젯을 가로 중앙에 위치 시킨다 이때 사이즈에 변화는 없다
fill_horizontal	해당 위젯의 가로를 부모 뷰 그룹의 사이즈에 맞게 늘려 채워준다

속성 관련 메소드

XML 속성	관련 메소드	비고
alpha	setAlpha(float)	0(투명)과 1(불투명) 사이의 값으로 불투명도를 지정
background	setBackgroundResource(int)	리소스의 그림을 배경으로 지정
clickable	setClickable(boolean)	이 뷰를 클릭하면 이벤트가 반응 할 것인가 아닌가를 지정
contentDescription	setContentDescription (CharSequence)	텍스트 내용을 간략하게 설명하는 텍스트를 지정
drawingCacheQuality	setDrawingCacheQuality(int)	
duplicateParentState		직접적인 부모로부터 그리기 스타일을 가져올 것 인지를 지정

속성 관련 메소드

XML 속성	관련 메소드	비고
fadeScrollbars	setScrollbarFadingEnabled (boolean)	사용하지 않을때 화면은 어둡게 할것인지를 지정
fadingEdgeLength	getVerticalFadingEdgeLength()	페이딩 가장자리의 길이를 지정
filterTouchesWhen Obscured	setFilterTouchesWhenObscured (boolean)	화면이 다른창으로 이동할 때 터치 필터링 여부를 지정
fitsSystemWindows	setFitsSystemWindows (boolean)	화면에 레이아웃을 표시할때 폰의 상태표시줄을 같이 표시할 것인지의 여부를 지정
focusable	setFocusable(boolean)	뷰가 포커스를 취할 수 있는지의 여부를 지정

속성 관련 메소드

XML 속성	관련 메소드	비고
focusableInTouchMode	setFocusableInTouchMode (boolean)	터치모드에서 포커스를 취할 수 있는지의 여부를 지정
hapticFeedbackEnabled	setHapticFeedbackEnabled (boolean)	보기와 같이 긴 프레스등의 이벤트를 사용할 햅틱 피드백을 가져야 할지의 여부를 지정
id	setId(int)	이 뷰를 참조 하기 위한 식별자(id)를 부여
importantForAccessibility	setImportantForAccessibility (int)	이 뷰의 이벤트가 중요한 시스템을 접근할 때 이것을 실행시킬지의 여부를 사용자에게 물어보는 창을 띄움
isScrollContainer	setScrollContainer(boolean)	이 뷰를 이동하거나 축소 확대가 가능하고, 스크롤이 있는 컨테이너인지를 지정

속성 관련 메소드

XML 속성	관련 메소드	비고
keepScreenOn	setKeepScreenOn(boolean)	뷰를 볼 동안 화면은 유지할지를 지정
layerType	setLayerType(int,Paint)	이 레이어의 유형을 지정
layoutDirection	setLayoutDirection(int)	레이아웃의 방향을 설정
longClickable	setLongClickable(boolean)	long클릭시 이벤트의 반응 여부를 지정
minHeight	setMinimumHeight(int)	뷰의 최소 높이를 지정
minWidth	setMinimumWidth(int)	뷰의 최소 너비를 지정
nextFocusDown	setNextFocusDownId(int)	다음 포커스가 되는 뷰를 지정
nextFocusForward	setNextFocusForwardId(int)	
nextFocusLeft	setNextFocusLeftId(int)	
nextFocusRight	setNextFocusRightId(int)	
nextFocusUp	setNextFocusUpId(int)	

속성 관련 메소드

XML 속성	관련 메소드	비고
onClick		뷰를 클릭하면 컨텍스트의 메소드를 호출한다
padding	setPaddingRelative(int,int,int,int)	사방의 가장자리 여백을 픽셀단위로 설정
paddingBottom	setPaddingRelative(int,int,int,int)	아래쪽 여백을 설정
paddingEnd	setPaddingRelative(int,int,int,int)	각 위치의 여백 설정
paddingLeft	setPadding(int,int,int,int)	
paddingRight	setPadding(int,int,int,int)	
paddingStart	setPaddingRelative(int,int,int,int)	
paddingTop	setPaddingRelative(int,int,int,int)	

속성 관련 메소드

XML 속성	관련 메소드	비고
requiresFadingEdge	setVerticalFadingEdgeEnabled (boolean)	
rotation	setRotation(float)	각도만큼 회전한다
rotationX	setRotationX(float)	각도만큼 x축 회전
rotationY	setRotationY(float)	각도만큼 y축 회전
saveEnabled	setSaveEnabled(boolean)	뷰에 지정된 id에 뷰 상태가 저장된다
scaleX	setScaleX(float)	x축의 스케일을 지정
scaleY	setScaleY(float)	y축 방향의 스케일을 지정
scrollX		가로스크롤의 초기위치를 픽셀단위로 지정
scrollY		세로스크롤의 초기위치를 픽셀단위로 지정

속성 관련 메소드

XML 속성	관련 메소드	비고
scrollbarAlwaysDrawHorizontalTrack		가로 스크롤을 항상 표시할것인지를 지정
scrollbarAlwaysDrawVerticalTrack		세로스크롤바를 항상 표시할 것인지를 지정
scrollbarDefaultDelayBeforeFade	setScrollBarDefaultDelayBeforeFade(int)	대기할때 화면이 어두워지기전의 시간을 지정
scrollbarFadeDuration	setScrollBarFadeDuration(int)	스크롤바가 사라지는데 걸리는 시간을 지정
scrollbarSize	setScrollBarSize(int)	스크롤바의 높이와 너비를 지정
scrollbarStyle	setScrollBarStyle(int)	스크롤바의 위치와 스타일을 지정
scrollbarThumbHorizontal		가로스크롤바의 당김을 지정(손가락)
scrollbarThumbVertical		세로스크롤바의 당김을 지정(손가락)

속성 관련 메소드

XML 속성	관련 메소드	비고
scrollbarTrackHorizontal		가로스크롤바의 당김을 지정(트랙)
scrollbarTrackVertical		세로스크롤바의 당김을 지정(트랙)
scrollbars		스크롤 여부에 따라 스크롤바를 표시할 것인지를 지정
soundEffectsEnabled	setSoundEffectsEnabled(boolean)	이벤트에 대한 사운드를 사용할 것인지를 지정
tag		나중에 검색 할 수 있도록 뷰의 문자열을 포함하여 이 뷰에 대한 태그를 제공한다
TextAlignment	setTextAlignment(int)	텍스트의 정렬을 지정
textDirection	setTextDirection(int)	텍스트의 방향을 지정. 위에서 아래로 쓸 것인가, 좌 혹은 우로 쓸 것인가의 방향
transformPivotX	setPivotX(float)	뷰가 회전하고 확장할 수 있는 기준 점의 x위치

속성 관련 메소드

XML 속성	관련 메소드	비고
transformPivotY	setPivotY(float)	뷰가 회전하고 확장할 수 있는 기준 점의 y위치
translationX	setTranslationX(float)	뷰의 x축을 변환
translationY	setTranslationY(float)	뷰의 y축을 변환
visibility	setVisibility(int)	뷰를 보이는 것을 지정 보이게 할 수 있고 안보이게 할 수 있다

속성 관련 메소드

- View를 숨기고, 감추고, 보여지게

```
view.setVisibility(View.GONE);
  // View를 숨긴다. (공간차지 X)
view.setVisibility(View.INVISIBLE);
  // View를 감춘다. (공간차지 O)
view.setVisibility(View.VISIBLE);
 // View를 보여 준다. (공간차지 O)
```

- 설정된 View의 Visibility 속성을 확인

```
if(view.getVisibility() == View.VISIBLE) {

}
```

속성 관련 메소드

- View의 배경화면(Background)를 설정

```
view.setBackgroundColor(Color.GREEN);
  // 백그라운드 색상 설정
view.setBackgroundResource(resid);
 // 리소스로 백그라운드 설정
view.setBackground(background);
 // Drawable로 백그라운드 설정
```

속성 관련 메소드

- 설정된 View의 LayoutParams을 얻어와서 다시 셋팅

```
View view = (View) findViewById(R.id.view_color);
  // View를 얻오옴
ViewGroup.LayoutParams mParams =
      view.getLayoutParams();
  // View의 Parent 설정 속성인 (LayoutParams)을 얻어옴
mParams.width =
ViewGroup.LayoutParams.MATCH_PARENT;
  // View의 width를 MATCH_PARENT로 변경
view.setLayoutParams(mParams);
  // View에 새로운 속성을 적용
```

속성 관련 메소드

- View의 각종 활성화 비활성화 상태
  - Selector를 이용하여 같이 사용하면 됨

```
view.setEnabled(true); // 사용가능 여부
view.isEnabled();
view.setSelected(true); // 선택여부
view.isSelected();
view.setFocusable(true); // 포커스 여부
view.isFocusable();
view.setPressed(true); // 눌림 여부
view.isPressed();
```

속성 관련 메소드

- 수정된 View 갱신하기
  - View의 Layout 변경 작업등을 한 후에 반영이 안되는 경우가 생길 수 도 있기 때문에 View의 invalidate() 메소드를 호출해서 다시 반영해 주는 것이 좋음

```
view.invalidate()
```

- 강제로 View에 Focus 주기
  - EditText나 셀렉터(Selector) 등을 이용할 때 사용하면 좋음

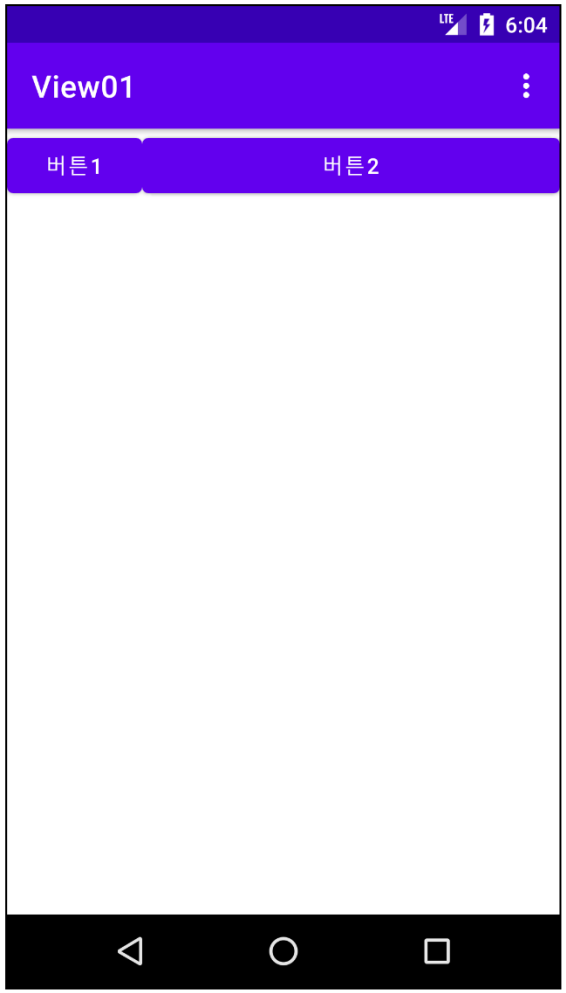
```
view.requestFocus()
```

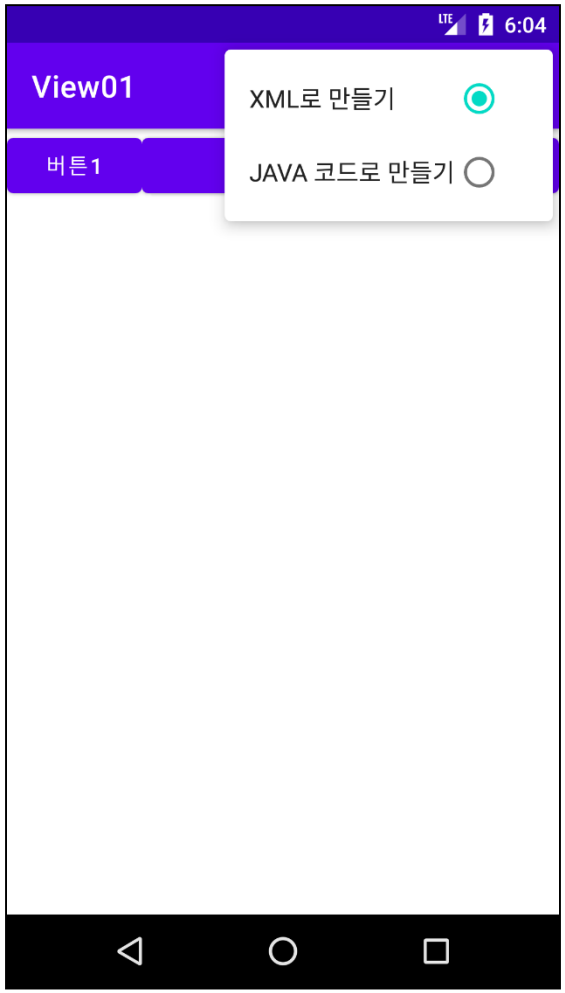
- View의 Click(Touch)음 설정하기

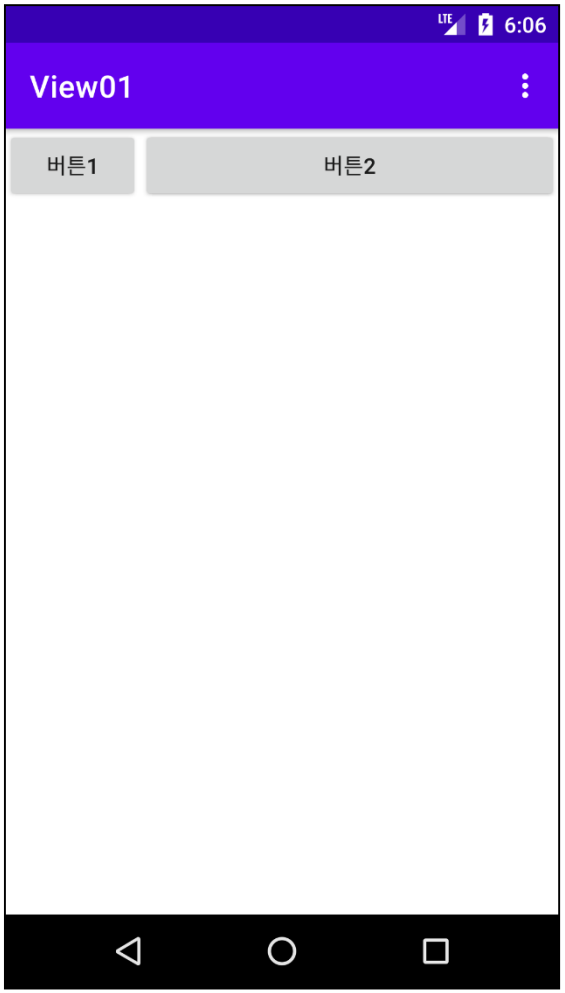
```
view.setSoundEffectsEnabled(true);
```

View 예제 1

- 다음과 같이 Activity에 Button Widget을 2개 배치하여보자







View 예제 1

- View 생성 방법
  - XML로 View 생성하기 (권장)
  - 코드로 View 생성하기

View 예제 1

- XML의 장점
  - View를 재사용하거나 다른 Layout에서 가져다 쓰기 쉬움
  - View가 Program과 분리되어 있어서 장기적으로 관리하기 편리함
  - 미리보기를 통해 어떤 View를 나타내는지 미리 파악할 수 있음

- JAVA Code로 그릴 때의 장점
  - Program의 상태에 따라 유동적으로 View를 조정할 수 있음
  - 별도로 Resource File을 생성할 필요가 없음

View 예제 1

- res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼1" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="버튼2" />
</LinearLayout>
```

View 예제 1

- menu/menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group
        android:checkableBehavior="single">
        <item
            android:id="@+id/item1"
            android:checked="true"
            android:title="XML로 만들기" />
        <item
            android:id="@+id/item2"
            android:title="JAVA 코드로 만들기" />
    </group>
</menu>
```

View 예제 1

- MainActivity.JAVA

```
public class MainActivity extends AppCompatActivity {
    int index = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    protected void onStart() {
        super.onStart();
        if (index == 1) {
            setContentView(R.layout.activity_main);
        } else {
```

View 예제 1

- MainActivity.JAVA

```
        LinearLayout layout = new LinearLayout(this);

        Button button1 = new Button(this);
        button1.setText("버튼1");
        layout.addView(button1);

        Button button2 = new Button(this);
        LinearLayout.LayoutParams buttonlayout =
          new LinearLayout.LayoutParams(MATCH_PARENT, WRAP_CONTENT);
        button2.setText("버튼2");
        layout.addView(button2, buttonlayout);
        setContentView(layout);
    }
}
```

View 예제 1

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu, menu);
        return super.onCreateOptionsMenu(menu);
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.item1:
                index = 1;
                break;
            case R.id.item2:
                index = 2;
        }
        item.setChecked(true);
        onStart();
        return true;
    }
}
```

View 예제 1

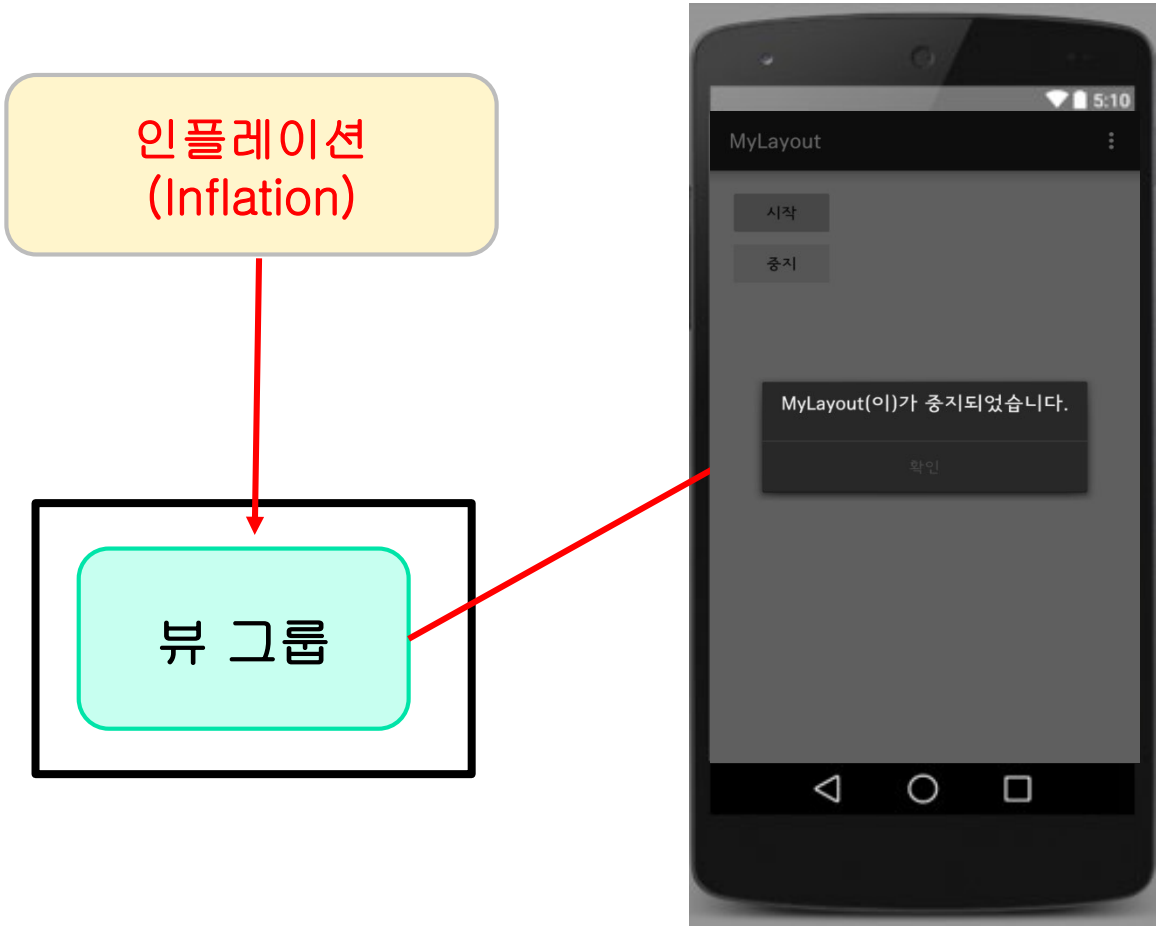
- XML을 이용한 View의 Architecture
  - 계층 구조를 표현하기 위해 Sub Tag로 표현

View 예제 1

- JAVA 코드를 사용한 View의 Architecture
  - addView() 메소드 이용

View 예제 1

- Inflation이란?



View 예제 1

- Inflation이란?
  - XML Layout은 단순히 XML로 정의된 File
  - XML Layout을 Application에서 사용하려면 객체화 과정을 거쳐야 됨
  - XML Layout에 정의된 내용이 Memory상에 객체화 되는 과정을 Inflation(인플레이션)이라고 함
  - XML Layout File의 내용이 Project가 Build되는 시점에 Binary Code로 Compile되어 Application에 포함됨
  - 실행 시점에 Compile된 Binary Code가 Memory상에 객체화 됨

View 예제 1

- Inflation이란?
  - Layout의 내용은 Android Project를 생성했을 때 자동으로 onCreate() 메소드에 정의되어 있는 setContentView() 메소드를 통해 Layout을 불러오게 됨

```
setContentView(R.id.activity_main)
```

  - setContentView()로 전달할 수 있는 Parameter에는 View 이외에도 XML Layout이 정의된 Resource, View 등이 되는데 R.id.activity_main이라고 정해진 기본 Resource를 전달하는 Code에서도 해당 XML File의 View를 보여주라는 것을 의미
  - 이 메소드는 화면에 나타날 View(Layout)를 지정하고(XML), 해당 View(XML)의 내용을 Memory상에 객체화하는 역할을 함

View 예제 1

- setContentView() 메소드 정의

```
public void setContentView(int layoutResID)

public void setContentView(View view ,[ViewGroup.LayoutParams params])
```

  - 화면 전체의 XML Layout이 아니라 일부분만을 차지하는 요소들을 XML Layout에서 불러들여 사용할 수도 있음
  - 이 경우엔 별도의 Inflation 객체를 사용해야 하고, Android에서는 이를 위해 LayoutInflater라는 클래스를 제공
  - LayoutInflater 클래스는 System Service로 제공되므로, getSystemService(Context.LAYOUT_INFLATER_SERVICE)라는 Code를 통해 LayoutInflater 객체를 참조한 후 사용할 수 있음

View 예제 1

- System Service로 제공되는 기능들은 모두 getSystemService()라는 메소드를 사용하여 객체를 참조한 후 사용해야 함. 여기서 부분 Layout File을 통하여 Layout을 보여주는 경우에는 부분 Layout File은 Inflation을 통해 ViewGroup으로, ViewGroup에서 Main Layout을 통해 Main 화면에 보여지게 됨
- 예) Main Layout이res/layout/activity_main.xml File안에 XML로 정의되어 있으면 setContentView(R.layout.activity_main)이라는 Code를 통해 화면에 나타낼 수 있음
- 그 중 일부를 분리하여 같은 folder에 button.xml이라는 File에 정의하였다면 이 File의 내용은 LayoutInflater라는 객체를 이용하여 ViewGroup 객체로 객체화(Inflation)한 후 Main Layout에 추가되는 과정을 거치게 됨

View 예제 1

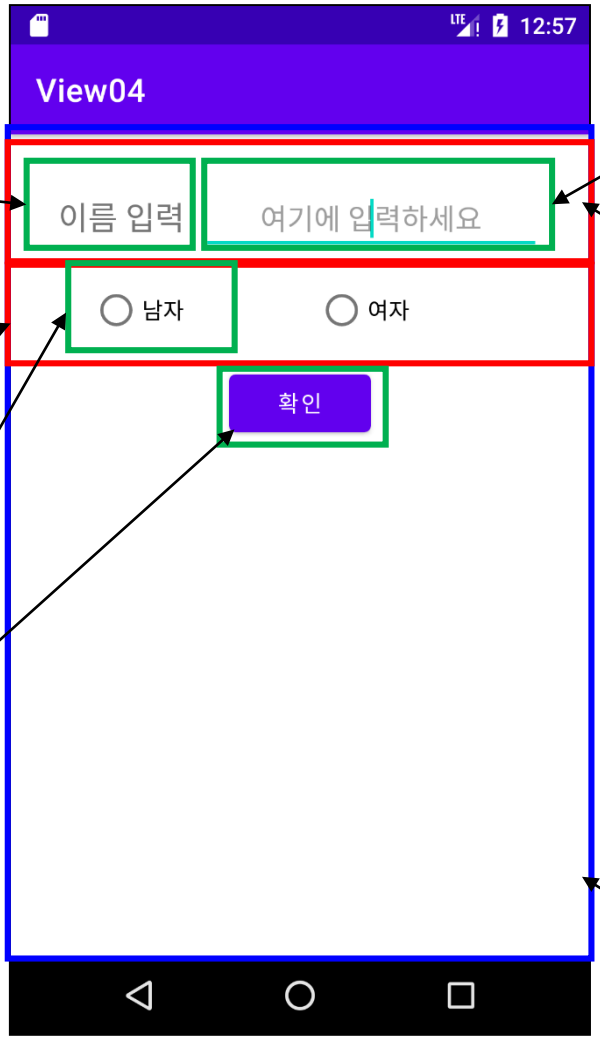
- Inflation을 사용하는 이유
  - 화면 출력 속도 향상
    - 대부분 하나의 XML Layout과 그 안에 TextView나 Button 화면만을 사용했지만, 실제 대부분의 상용 Application은 다수의 XML Layout들로 구성
    - 상식적으로 생각할 때 화면의 작은 부분에 변화가 발생하였을 때, 전체 화면 단위로 출력하는 것보다 해당 부분만을 화면에 출력하는 것이 System 성능에 도움이 됨
  - 문서 공간 절약
    - 전체 화면의 공간을 차지하는 XML 문서를 여러 개 만드는 것보다 용도별로 XML 문서를 나누어 보관하면 상대적으로 적은 공간을 차지

View 예제 1

- Inflation을 사용하는 이유
  - XML Layout 공유
    - 부품처럼 작은 단위로 나눌 수 있다는 이점은 Android 내 Application들 사이에서 XML File들을 서로 공유할 수 있다는 장점을 제공

View 예제 2

- 다음과 같은 View를 만들어보자



TextView

EditText

LinearLayout
(ViewGroup)

RadioGroup
(ViewGroup)

RadioButton

Button

LinearLayout

View 예제 2

- activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="30dp"
    tools:context=".MainActivity">
```

View 예제 2

- activity_main.xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:text="이름 입력  "/>

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:hint="여기에 입력하세요"/>
</LinearLayout>
```

View 예제 2

- activity_main.xml

```
<RadioGroup
    android:layout_marginTop="10dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:orientation="horizontal">
    <RadioButton
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="남자"/>
    <RadioButton
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="여자"/>
</RadioGroup>
```

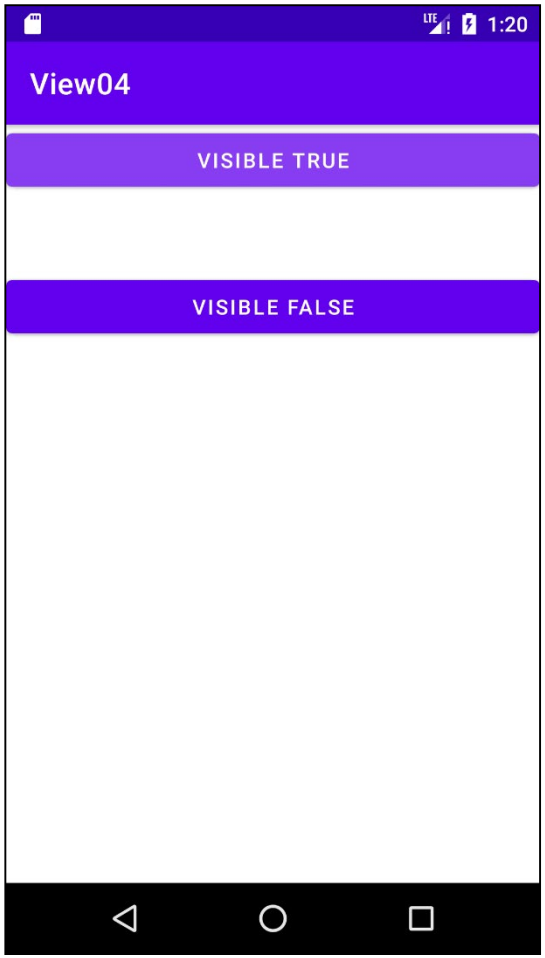
View 예제 2

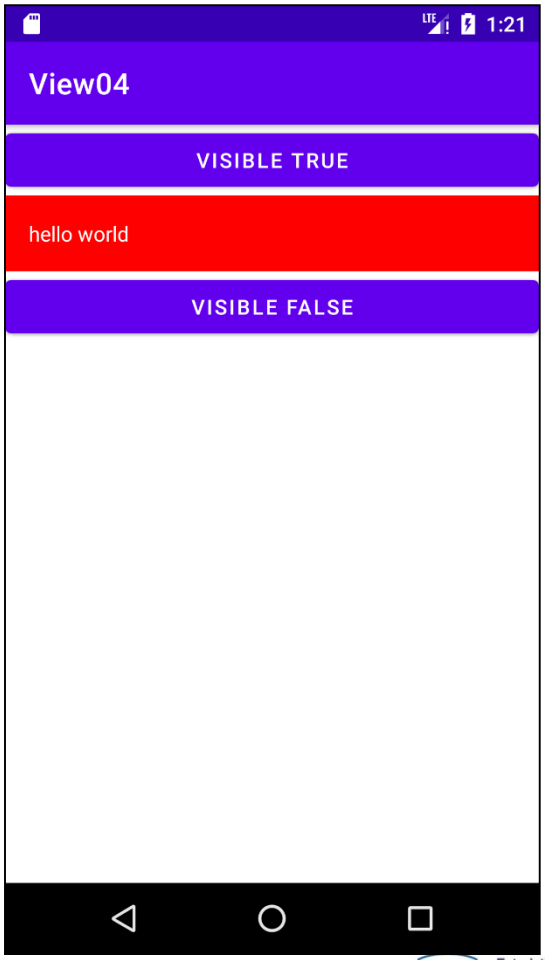
- activity_main.xml

```
    <Button
        android:layout_marginTop="10dp"
        android:layout_gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="확인"/>
</LinearLayout>
```

View 예제 3

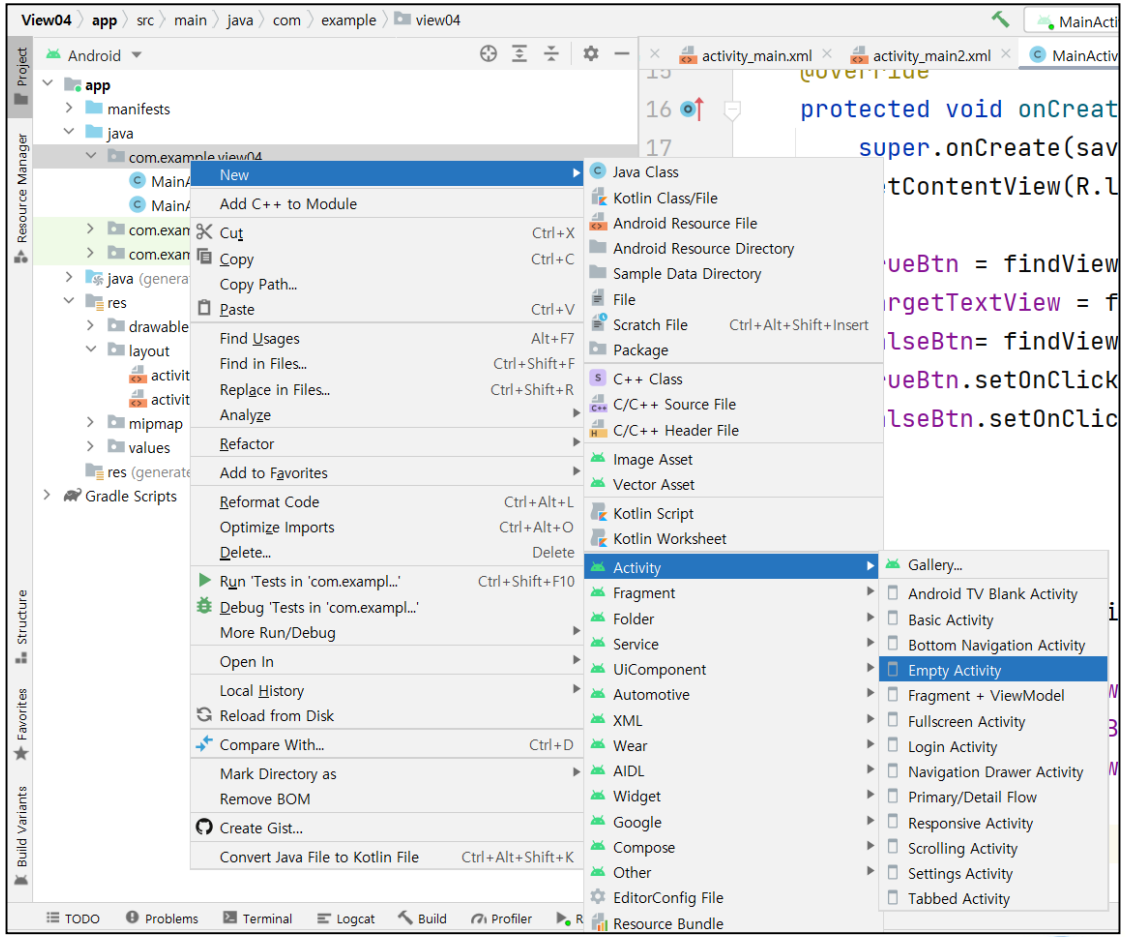
- View의 기본 속성을 활용하는 Activity를 예제 2에 추가하여 보자





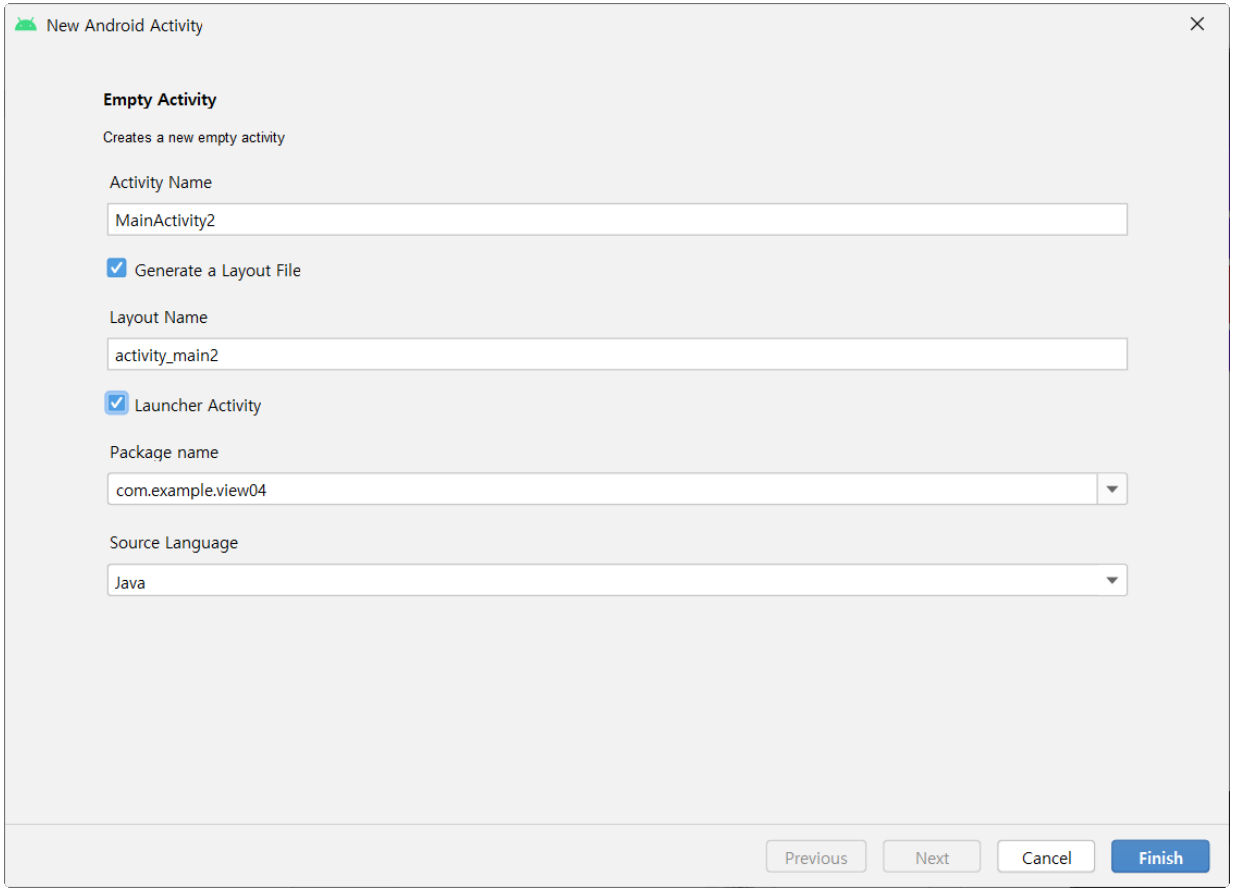
View 예제 3

- Activity 추가 방법
  - [File] – [New] – [Activity] – [Empty Activity]



View 예제 3

- Activity 이름 설정
  - MainActivity2 – Launcher Activity 체크



View 예제 3

- res/layout/activity_main2 파일을 수정

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity2">

    <Button
        android:id="@+id/visible"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="visible true" />
```

View 예제 3

- res/layout/activity_main2 파일을 수정

```
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#FF0000"
        android:padding="16dp"
        android:text="hello world"
        android:textColor="#FFFFFF"
        android:visibility="invisible" />

    <Button
        android:id="@+id/invisible"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="visible false" />
</LinearLayout>
```

View 예제 3

- MainActivity2.JAVA 작성

```
public class MainActivity2 extends AppCompatActivity
                                                    implements View.OnClickListener {
    TextView textView;
    Button button1, button2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        textView = findViewById(R.id.textView);
        button1 = findViewById(R.id.visible);
        button2 = findViewById(R.id.invisible);
        button1.setOnClickListener(this);
        button2.setOnClickListener(this);
    }
```

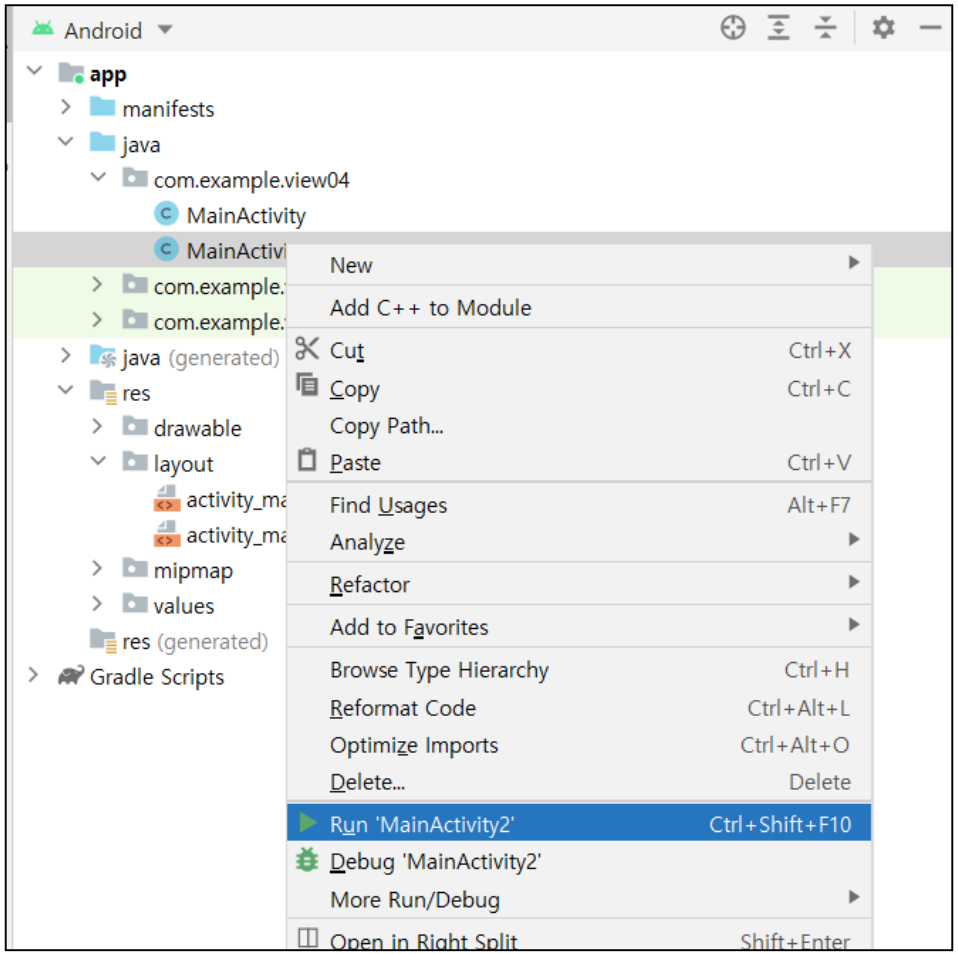
View 예제 3

- MainActivity2.JAVA 작성

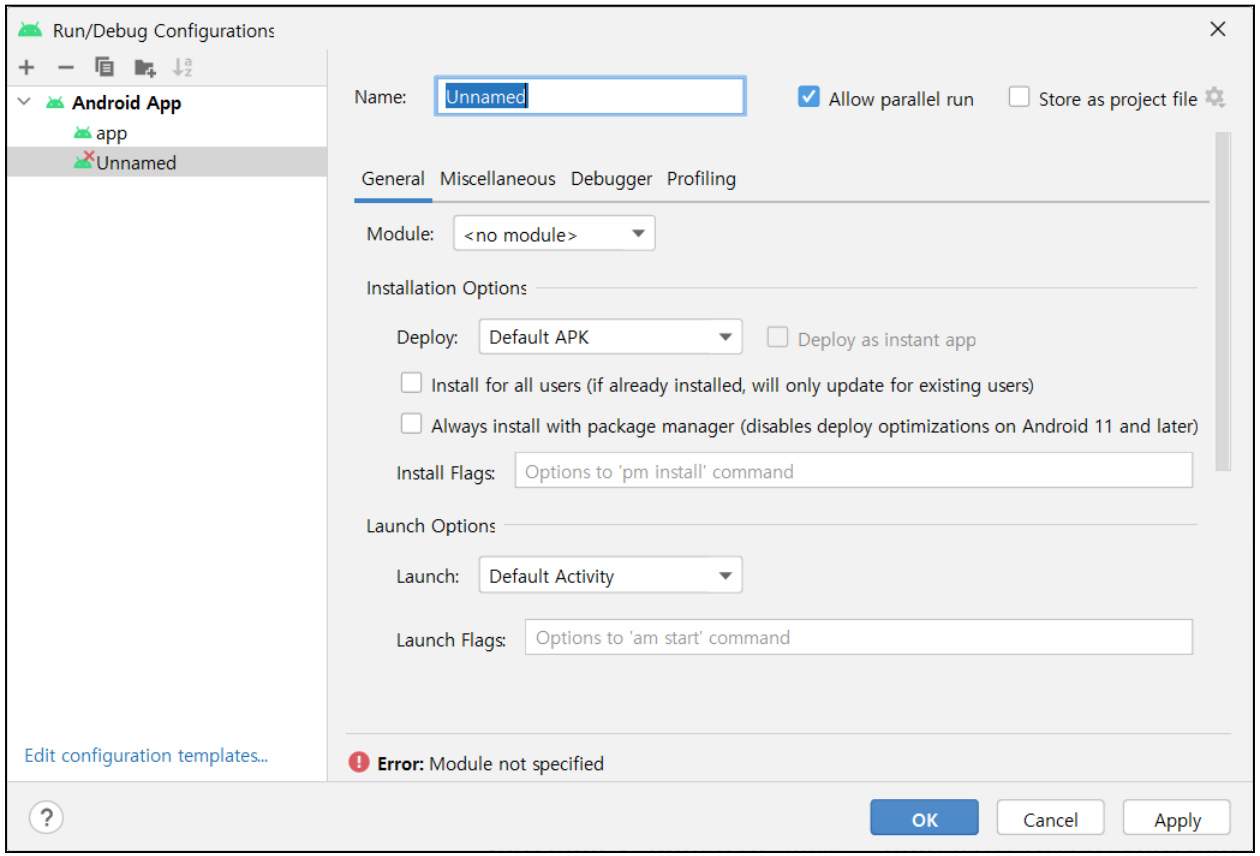
```
    @Override
    public void onClick(View v) {
        if(v == button1){
            textView.setVisibility(View.VISIBLE);
        }else if(v == button2){
            textView.setVisibility(View.INVISIBLE);
        }
    }
}
```

View 예제 3

- MainActivity2.JAVA 파일을 선택하고 마우스 오른쪽을 클릭해서 run

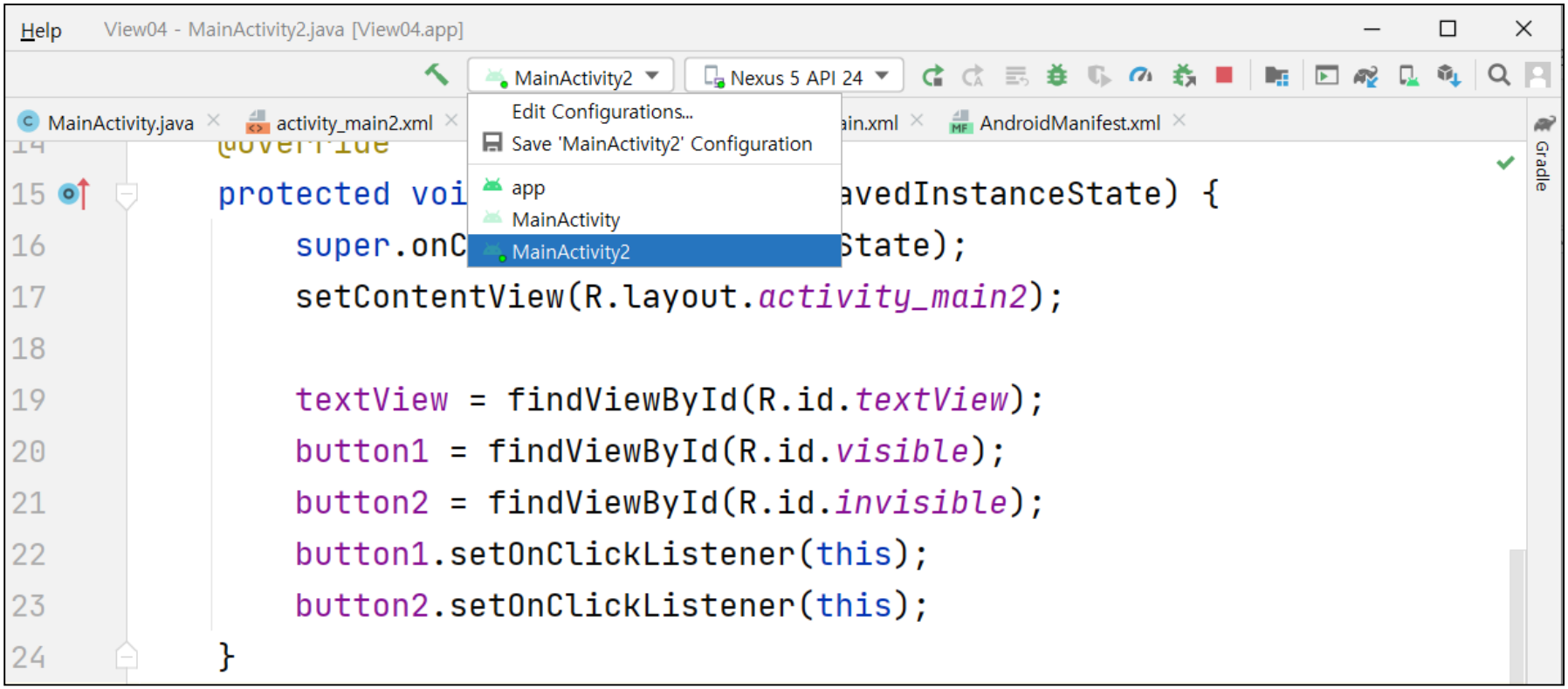


View 예제 3



View 예제 3

- MainActivity2.JAVA 파일 실행 방법



View 예제 4

View 예제 4

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity4">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#ff77ff"
        android:text="margin, padding 적용전"
        android:textColor="#000000" />
```

View 예제 4

- 사용자 인터페이스

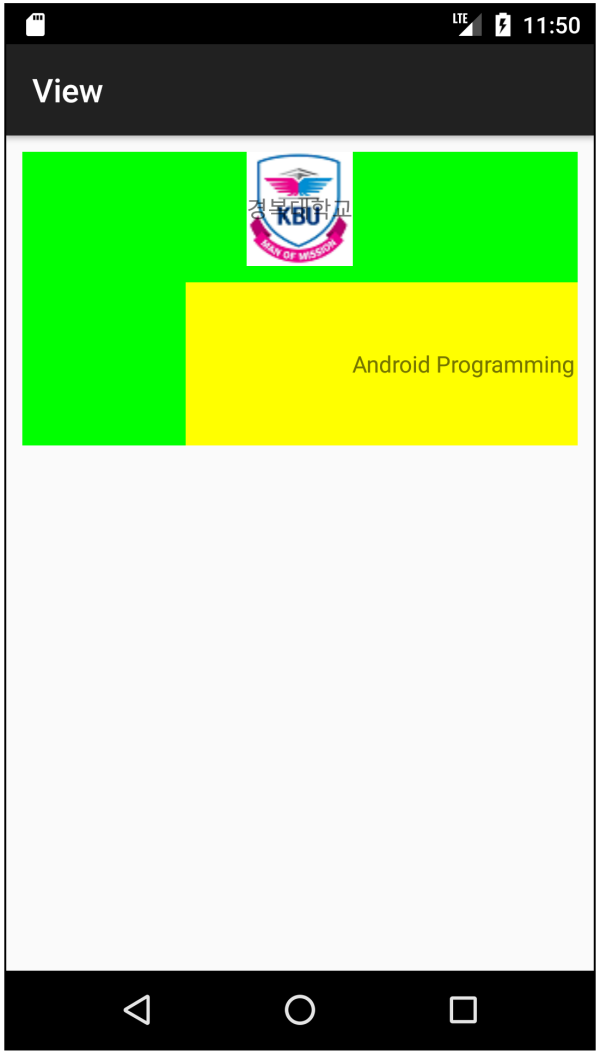
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    android:background="#ff7777"
    android:text="margin만 적용한 예"
    android:textColor="#000000" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#7777ff"
    android:padding="20dp"
    android:text="padding만 적용한 예"
    android:textColor="#000000" />
```

View 예제 4

- 사용자 인터페이스

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:background="#77ffff"
        android:padding="20dp"
        android:text="margin과 padding 모두 적용한 예"
        android:textColor="#000000" />
</LinearLayout>
```

View 예제 5



View 예제 5

- 사용자 인터페이스

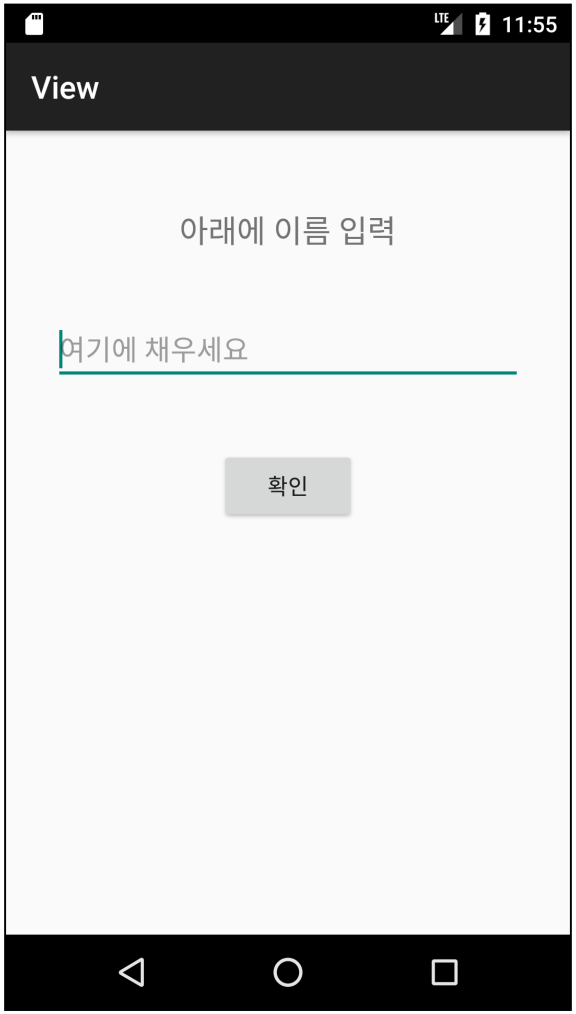
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:background="#00FF00"
    android:orientation="vertical"
    tools:context=".MainActivity5">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:background="@drawable/kyungbok"
        android:gravity="center"
        android:text="경복대학교" />
```

View 예제 5

- 사용자 인터페이스

```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="10dp"
        android:background="#FFFF00"
        android:gravity="center"
        android:paddingLeft="100dp"
        android:text="Android Programming" />
</LinearLayout>
```

View 예제 6



View 예제 6

- 사용자 인터페이스

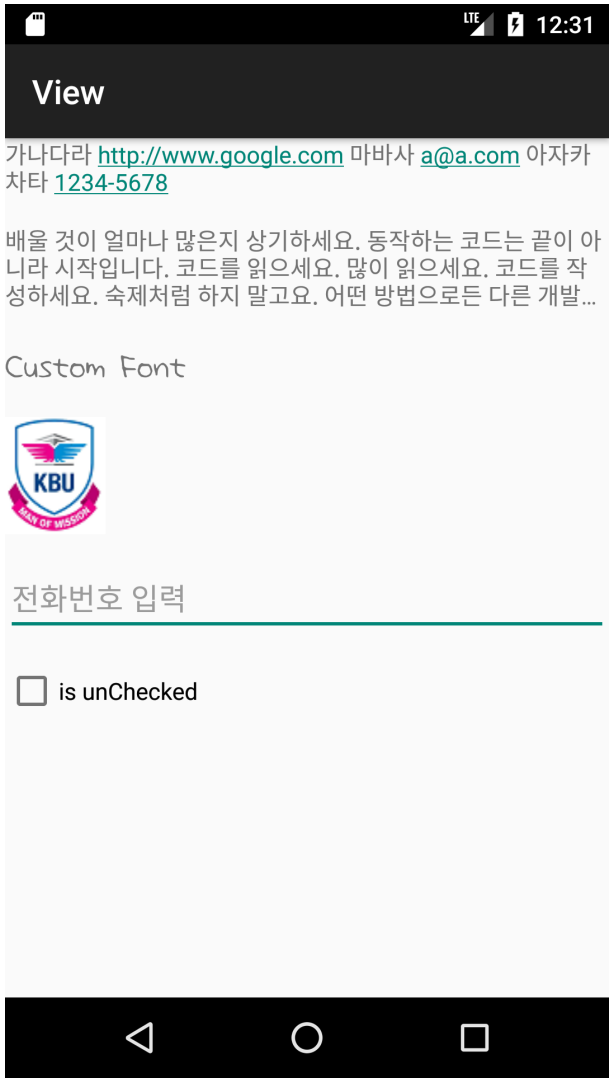
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center|top"
    android:padding="30dp"
    tools:context=".MainActivity6">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="아래에 이름 입력"
        android:textSize="20dp" />
```

View 예제 6

- 사용자 인터페이스

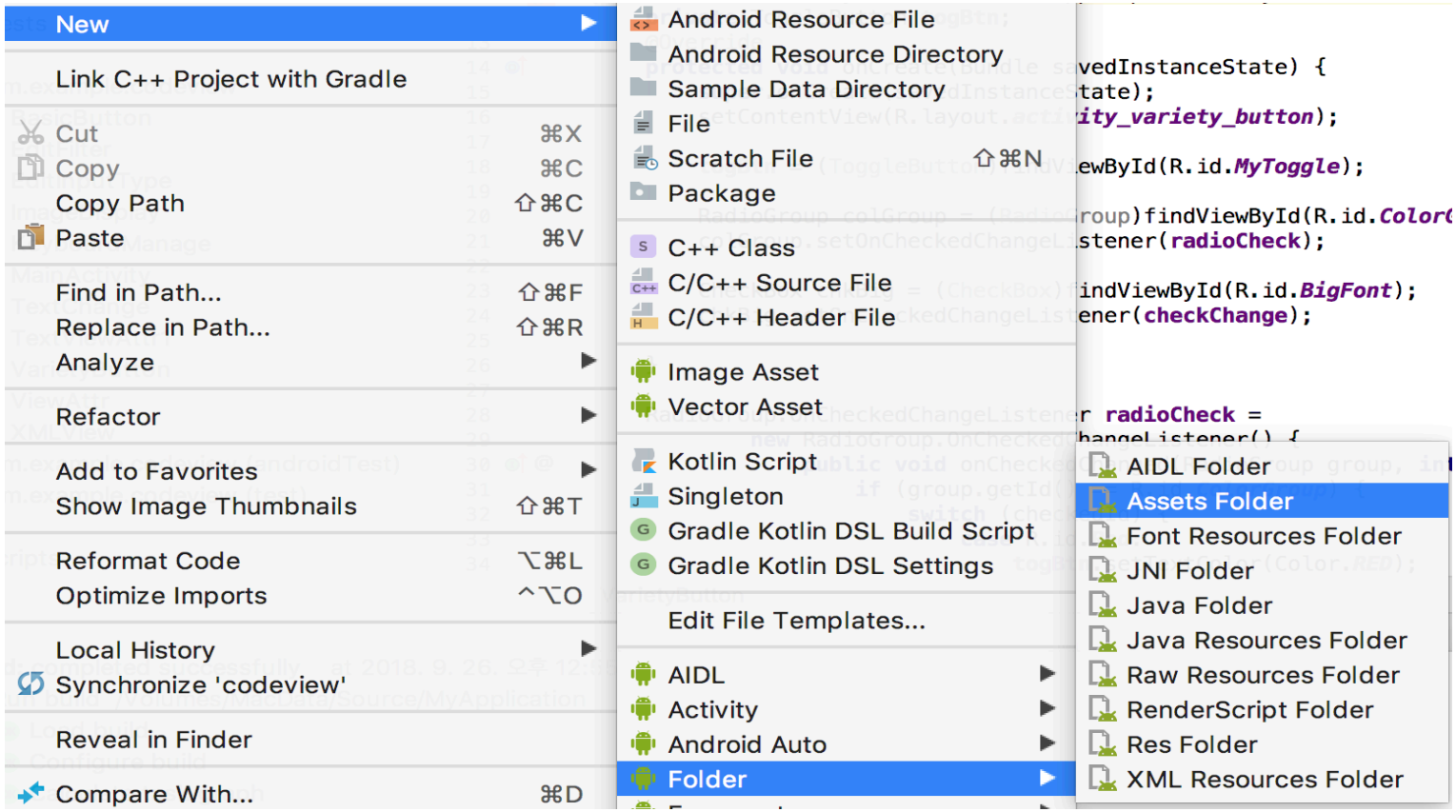
```
    <EditText
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:hint="여기에 채우세요" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="확인" />
</LinearLayout>
```

View 예제 7



View 예제 7

- Resource를 저장할 assets Directory를 추가하고 Font File을 추가



View 예제 7

- strings.xml

```
<resources>
    <string name="app_name">View</string>

    <string name="creed">
        배울 것이 얼마나 많은지 상기하세요.
            동작하는 코드는 끝이 아니라 시작입니다.
            코드를 읽으세요. 많이 읽으세요.
            코드를 작성하세요. 숙제처럼 하지 말고요.
            어떤 방법으로든 다른 개발자와 일대일로 일해보세요.
            도구가 아니라 기법을 배우세요.
            내가 알고자 하는 것은 책에 다 있습니다.
            거인의 어깨에 올라서서 더 넓은 세상을 바라보라.
            하지 안하야 할 이유를 찾지 말고 해야 할 이유를 찾자.
            學을 해야 習을 할 수 있고 習을 해야 覺을 할 수 있다.
    </string>

</resources>
```

View 예제 7

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:autoLink="web|email|phone"
        android:text="가나다라 http://www.google.com 마바사
                                    a@a.com 아자카차타 1234-5678" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:ellipsize="end"
        android:maxLines="3"
        android:text="@string/creed" />
```

View 예제 7

- 사용자 인터페이스

```
<TextView
    android:id="@+id/fontView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="Custom Font"
    android:textSize="30dp" />

<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:adjustViewBounds="true"
    android:maxWidth="100dp"
    android:maxHeight="100dp"
    android:src="@drawable/kyungbok" />
```

View 예제 7

- 사용자 인터페이스

```
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:hint="전화번호 입력"
        android:inputType="phone" />

    <CheckBox
        android:id="@+id/checkbox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="is unChecked" />
</LinearLayout>
```

View 예제 7

- JAVA Program

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main7);

    TextView textView = findViewById(R.id.fontView);
    Typeface typeface = Typeface.createFromAsset(getAssets(),
                                                 "나눔손글씨 무궁화.ttf");
    textView.setTypeface(typeface);
```

View 예제 7

- JAVA Program

```
    CheckBox checkBox = findViewById(R.id.checkbox);
    checkBox.setOnCheckedChangeListener(
                            new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView,
                                                        boolean isChecked) {
            if (isChecked) {
                checkBox.setText("is Checked");
            } else {
                checkBox.setText("is unChecked");
            }
        }
    });
}
```