

소프트웨어설계및실험

2024 Spring

뷰와 위젯, 레이아웃

이론

- ◆ View
 - ◆ View의 실행화면
 - ◆ View의 종류와 배치
 - ◆ View의 속성과 크기
 - ◆ View의 영역
 - ◆ View의 다중 배치와 시각표현
- ◆ Widget
 - ◆ Widget 배치하기
 - ◆ Widget 연결하기
 - ◆ 알고리즘 연결하기
- ◆ Layout
 - ◆ Layout의 개요
 - ◆ Layout 종류와 속성
 - ◆ LinearLayout
 - ◆ RelativeLayout
 - ◆ FrameLayout
 - ◆ GridLayout
 - ◆ ComstraintLayout

View - View와 실행 화면

❖ 실행화면

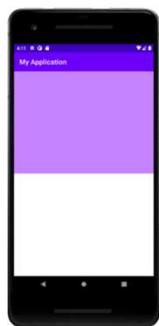
- 앱을 실행하면, 화면에 무엇이 나와야 사용자가 인식하고 사용할 수 있음
- 실행화면은 **View들을 구성하고 배치한 결과물**임

❖ View 란?

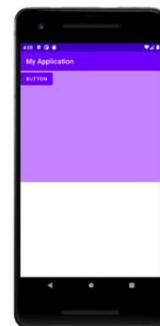
- 앱의 **실행화면**을 구성하는 요소



프로젝트 만들기



공간 만들기



배치(구성)하기

```
<LinearLayout  
    android:layout_weight="1"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:background="@color/purple_200">  
    <Button  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Button"/>  
</LinearLayout>
```



View - View의 종류와 배치

❖ View의 종류

Button



Layout

- Widget
 - View(View) 중에 **시각적인 요소**
 - ImageView, TextView, Button, ...

- ViewGroup
 - View(View) 중에 **영역적인 요소**
 - Layout, container

❖ View의 요소 배치

- View의 요소 배치는 구조를 표현하는 언어인 **XML로 수행됨**

- 모든 View 요소는 태그라는 형태로 명시됨**

- <시작 태그>~</끝 태그> 혹은 <약식 태그>/** 표현됨

- 모든 View 요소는 Layout 태그 내에 포함됨**

- 최상단에 존재하는 태그는 선언 태그와 Layout 태그임**

선언 태그

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="vertical"
    android:weightSum="2"
    tools:context=".MainActivity">
```

시작 태그

```
<LinearLayout
    android:layout_weight="1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/purple_200">
```

약식 태그

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

끝 태그

```
</LinearLayout>
```

```
</LinearLayout>
```

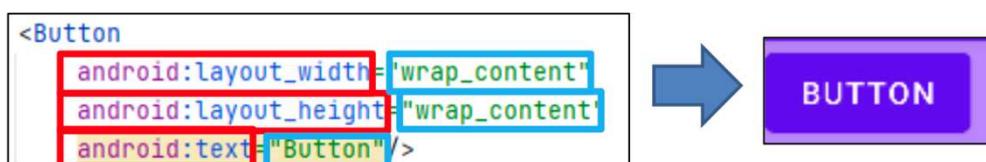
View - View의 속성과 크기

❖ View의 속성(Attribute)

- View 요소에 부여하는 **특징 및 참조 정보**
- View 요소의 속성 중 **크기 속성은 필수 정보**
 - 속성 = “속성값”

❖ View 의 크기

- View 는 화면에 배치되는 요소로서, **공간의 크기에 대한 속성은 필수적으로 명시되어야 함**
- **공간의 크기는 참조하는 방식과 값을 지정해주는 방식이 있음**
 - **match_parent**: 상위 요소(부모 요소)의 공간 범위를 참조함
 - **wrap_content**: 본인 및 하위 요소의 텍스트 문자 크기를 참조함
 - **지정 값**: View 요소의 크기를 지정함
 - “범위 값 + 범위 단위”
 - **Widget 최적 단위** : dp
 - **텍스트 최적 단위** : sp



The detailed XML code for the button is as follows, with annotations:

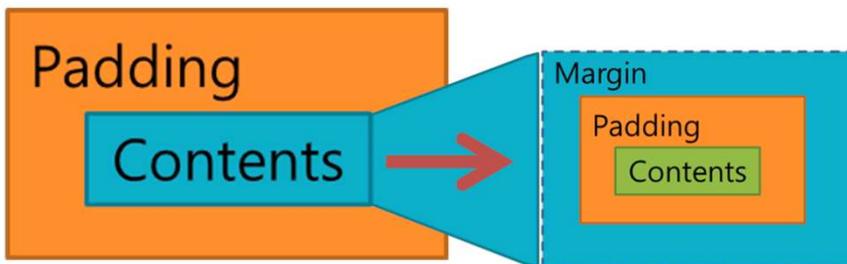
```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android=  
    "http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    android:weightSum="2"  
    tools:context=".MainActivity">  
  
<LinearLayout  
    android:layout_weight="1"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:background="@color/purple_200">  
  
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Button"/>  
  
</LinearLayout>  
</LinearLayout>
```

A red box highlights the top-level LinearLayout's width and height attributes. Another red box highlights the inner LinearLayout's layout_weight attribute. A third red box highlights the Button's layout_width and layout_height attributes. A fourth red box highlights the Button's text content. Red arrows point from the text "속성" (Attribute) to each of these highlighted sections.

View - View의 영역

❖ View의 영역

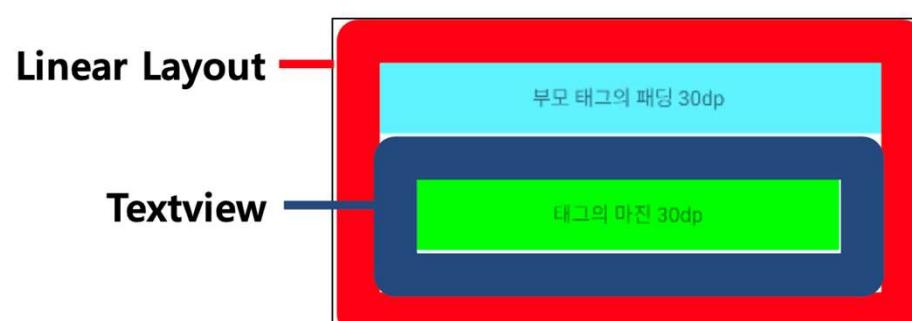
- 내용(Contents), 패딩(Padding), 마진(Margin)
- View의 간격은 margin과 padding 속성으로 설정함



❖ 패딩(Padding)과 마진(Margin)

- 패딩 속성은 Widget의 경계선으로부터 **Widget 내부의 요소가 멀어지게 여백을 설정할 수 있음**
- 마진 속성은 Widget의 경계선 밖으로 **부모 태그로부터 Widget이 멀어지게 여백을 설정할 수 있음**

```
<LinearLayout  
    ...  
    android:padding="30dp">  
    <TextView  
        ...  
        android:background="#89EEFF"  
        android:text="부모 태그의 패딩 30dp"/>  
    <TextView  
        ...  
        android:layout_margin="30dp"  
        android:background="#00FF00"  
        android:text="태그의 마진 30dp"/>  
</LinearLayout>
```



View - View의 다중 배치와 시각 표현

❖ View의 다중 배치

- 하나의 Layout 안에 여러 개의 View 요소 배치가 가능함
- 다중 배치를 위해서는 적절한 시각 표현이 필요함

❖ View의 시각 표현

- 시각 표현 관련 속성을 통해서 태그를 화면에 다양하게 표현 할 수 있다.
- 색상 및 회전을 통해서 View 요소를 강조하는 표현할 수 있다.
- View 요소를 숨기거나, 드러낼 수 있다.
- View 요소의 상호작용을 비활성화하거나, 활성화 할 수 있다.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="vertical"
    android:weightSum="2"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="30dp">
        <Button...>
        <Button...>
        <Button...>
        <Button...>
        <Button...>
        <Button...>
        <Button...>
        <Button...>
    </LinearLayout>
</LinearLayout>
```

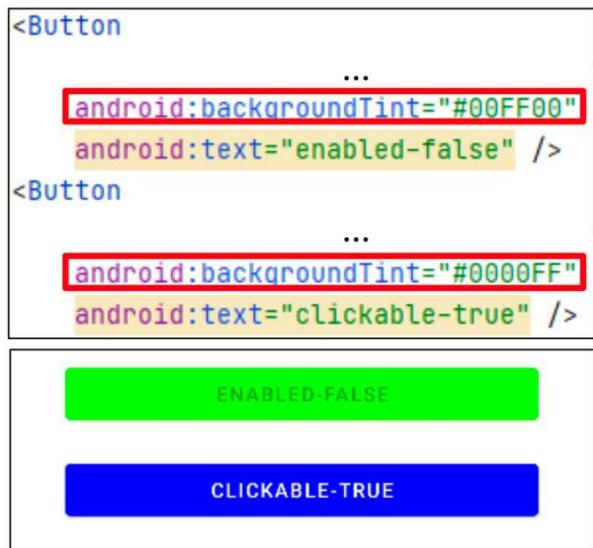


View - View의 시각 표현 – 색상과 회전

❖ View의 색상 속성

- View 요소에 색상을 결정

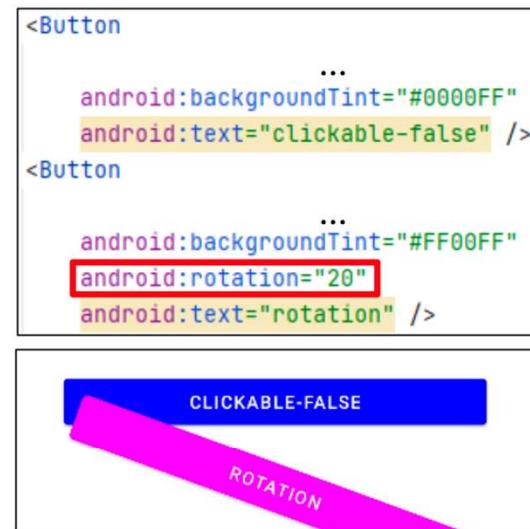
- backgroundTint="RGB 색상 코드"



❖ View의 회전 속성

- View 요소에 회전 각도를 결정

- rotation="각도"



❖ View의 색상과 회전

- View 의 색상과 회전은 제공하는 Widget을 강조하여 사용자의 이목을 끌 수 있다.
- Layout을 포함한 모든 View 요소 동일하게 적용이 가능하다.

View - View의 시각 표현 – 가시성과 활성화

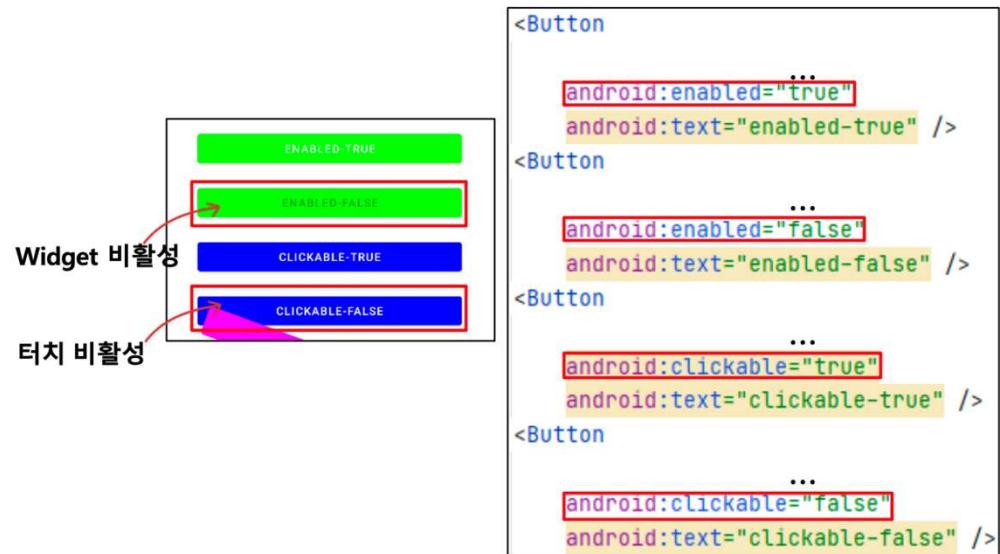
❖ View의 가시성

- View 요소가 보일 것인지 여부를 결정
 - 숨김 : "invisible"-영역 유지, "gone"-영역 비 유지



❖ View의 활성화

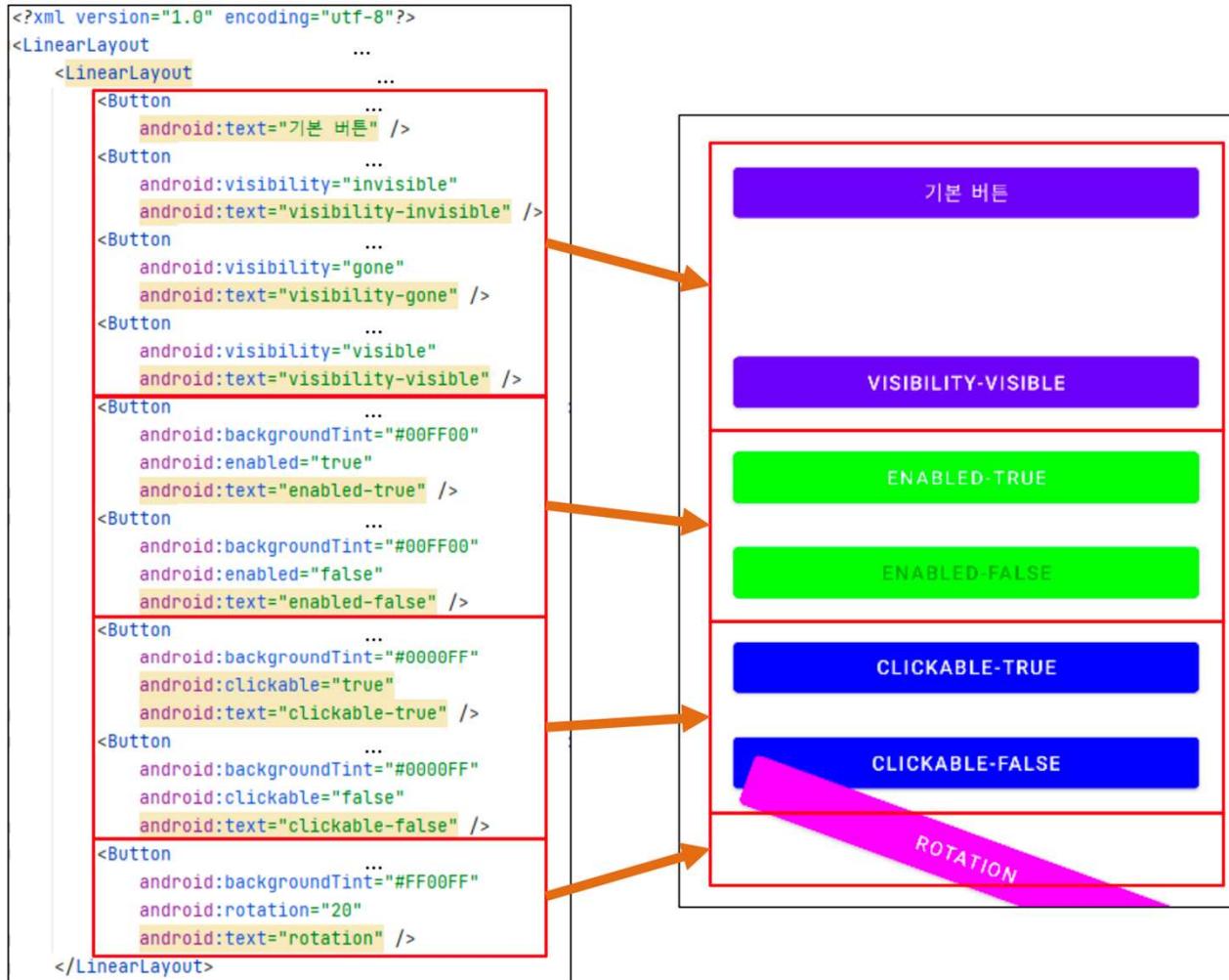
- View 요소의 활성화 여부를 결정
 - **enabled**: Widget 동작 활성화
 - **Clickable**: 상호작용(클릭, 터치) 활성화



❖ View의 가시성과 활성화

- View 요소의 가시성과 활성화는 이벤트 처리 측면에서 중요한 역할을 수행한다.
- XML 코드에서 사용되기보다, 주로 Kotlin 코드와 연계되어서 많이 사용되는 속성이다.

View - View의 시각 표현 - 소스 코드



Widget - Widget 다루기

❖ Widget의 역할

- Widget은 구성된 알고리즘을 시작하는 트리거 역할을 수행함
- 알고리즘의 연산 결과를 Widget으로 표현함

```
<Button  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="10dp"  
    android:id="@+id/BtnAdd"  
    android:text="더하기" />
```

Widget 배치(XML)

```
lateinit var btnAdd : Button; lateinit var btnSub : Button  
lateinit var btnMul : Button; lateinit var btnDiv : Button  
lateinit var textResult : TextView  
lateinit var num1 : String; lateinit var num2 : String  
var result : Int? = null  
  
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    title = "초간단 계산기"  
  
    edit1 = findViewById<EditText>(R.id.Edit1)  
    edit2 = findViewById<EditText>(R.id.Edit2)  
  
    btnAdd = findViewById<Button>(R.id.BtnAdd)
```

Widget 연결(Kotlin)

```
btnAdd.setOnTouchListener{ view, motionEvent ->  
    num1 = edit1.text.toString()  
    num2 = edit2.text.toString()  
    result = Integer.parseInt(num1) + Integer.parseInt(num2)  
    textResult.text = "계산 결과 : "+result.toString()  
    false ^setOnTouchListener
```

알고리즘 구성(Kotlin)



Widget - Widget 배치하기

❖ Widget의 연결 속성

- 개발 단계에서 자동 완성 지원
 - tools:context=".클래스명"
- 알고리즘과 태그 사이의 연결 / id 리스트 등록
 - android:id="@+id/고유식별자"



❖ Widget 배치하기

- Widget은 화면에 배치되며, 터치를 통해서 사용자와의 상호작용으로 알고리즘 수행이 가능함
- 알고리즘 수행을 위해서 구성된 알고리즘을 Widget에 연결해야 함
- 연결을 위해서, XML과 Kotlin의 각 코드에 연결을 위한 부분 추가가 필요함

A screenshot of an Android XML layout file is shown. The file contains a `LinearLayout` with various child views like `EditText`, `Button`, and `TextView`. Annotations with red arrows point to specific attributes:

- An arrow labeled "tools" points to the `tools:context=".MainActivity"` attribute.
- An arrow labeled "id" points to the `android:id="@+id/Edit1"` attribute of the first `EditText`.
- An arrow labeled "id" also points to the `android:id="@+id/BtnAdd"` attribute of the second `Button`.
- An arrow labeled "id" points to the `android:id="@+id/TextResult"` attribute of the `TextView` at the bottom.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    tools:context=".MainActivity">
    ...
    <EditText
        android:id="@+id/Edit1"
        android:hint="숫자1" />
    ...
    <EditText android:layout_width="wrap_content" />
    <Button android:layout_width="match_parent" />
    <Button android:layout_width="match_parent" />
    <Button android:layout_width="match_parent" />
    <TextView android:layout_width="wrap_content"
        android:id="@+id/TextResult"
        android:textSize="30dp"
        android:textColor="#FF0000"
        android:text="계산 결과 : "/>
</LinearLayout>
```

Widget - Widget 연결하기

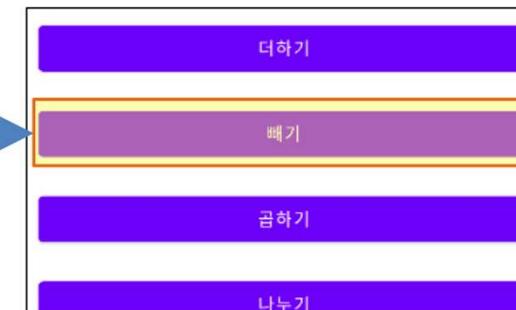
❖ Widget 객체

- 태그를 연결하기 위한 객체 생성
- 태그의 정보에 접근하기 위한 `findViewById()` 메소드 호출



❖ Widget 연결하기

- `id` 속성 값을 사용해서 XML 태그와 Kotlin 객체를 **3** **Var BtnSub** 연결



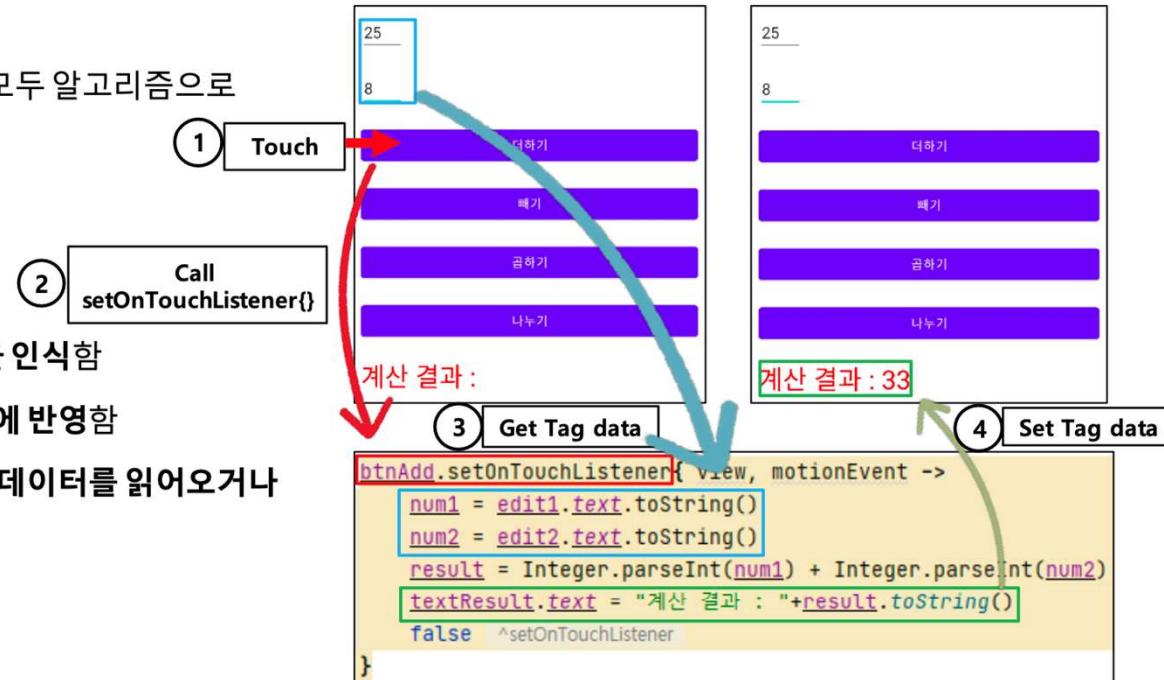
Widget - 알고리즘 실행하기

❖ 알고리즘 실행

- 애플리케이션에서 처리하고 동작하는 과정은 모두 알고리즘으로 수행됨

❖ 알고리즘

- Widget의 백그라운드 동작 역할을 수행함
- 주로, 화면 내의 Widget 위치에 대한 상호작용을 인식함
- 알고리즘의 수행 결과를 Widget을 통해서 화면에 반영함
- 태그와 연결된 객체는 태그에 접근하여 태그의 데이터를 읽어오거나 변경할 수 있음



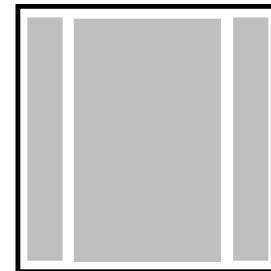
Layout - Layout의 개요

❖ Layout 이란?

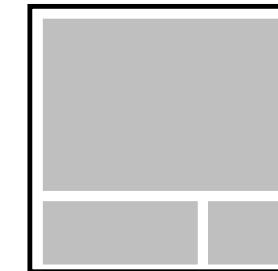
- View를 배치하는 클래스, View를 배치하는 규칙

❖ Layout의 종류

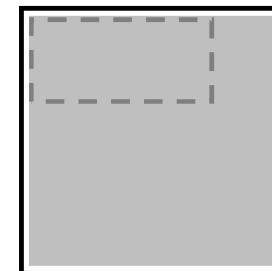
- **LinearLayout** - 선형으로 배치
- **RelativeLayout** – 상대 위치로 배치
- **FrameLayout** – 겹쳐서 배치
- **GridLayout** – 표 형태로 배치
- **ConstraintLayout** – 계층 구조로 배치



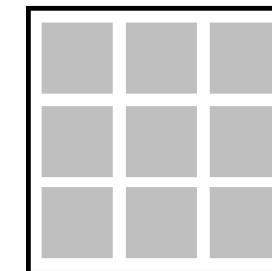
LinearLayout



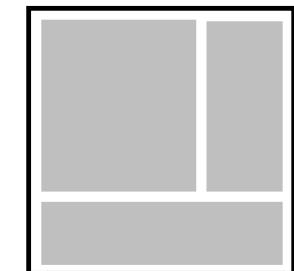
RelativeLayout



FrameLayout



GridLayout



ConstraintLayout

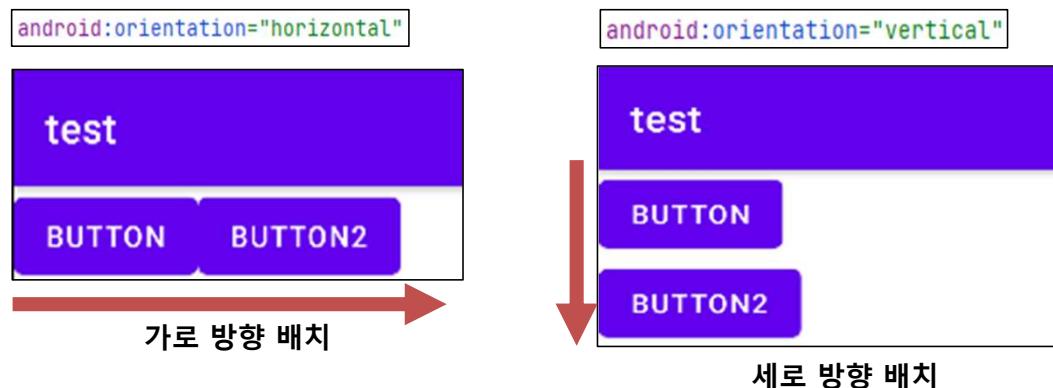
LinearLayout

❖ LinearLayout 이란?

- View 를 가로, 세로 방향으로 나열하는 Layout 클래스

❖ LinearLayout - Orientation 방향 속성

- **horizontal**: View를 가로로 나열한다.
- **vertical**: View를 세로로 나열한다.



```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="horizontal">  
    <Button  
        android:id="@+id/button"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Button"  
    />  
    <Button  
        android:id="@+id/button2"  
        android:layout_height="wrap_content"  
        android:layout_width="wrap_content"  
        android:text="Button2"  
    />  
</LinearLayout>
```

가로 배치

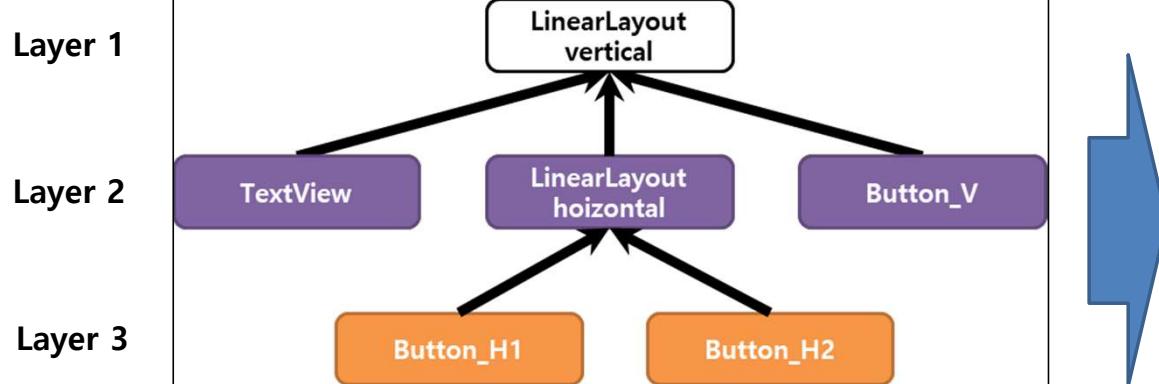

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical">  
    <Button  
        android:id="@+id/button"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Button"  
    />  
    <Button  
        android:id="@+id/button2"  
        android:layout_height="wrap_content"  
        android:layout_width="wrap_content"  
        android:text="Button2"  
    />  
</LinearLayout>
```

세로 배치

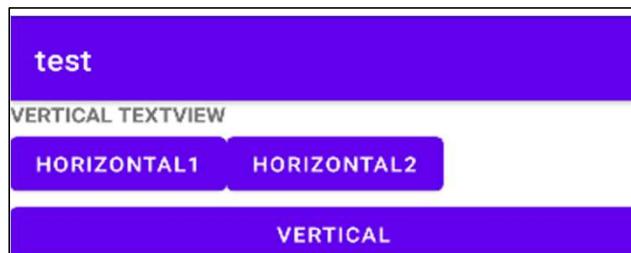
LinearLayout

❖ LinearLayout에서의 중첩 구조

- Layout 클래스도 View이므로 다른 Layout에 포함될 수 있다.



- Layout 클래스 안에 다른 Layout 클래스를 추가함으로써 중첩 Layout 구조를 구현할 수 있다.



```
<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:text="VERTICAL TEXTVIEW"  
    />  
    <LinearLayout  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:orientation="horizontal">  
        <Button  
            android:id="@+id/button"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="horizontal1" />  
        <Button  
            android:id="@+id/button2"  
            android:layout_height="wrap_content"  
            android:layout_width="wrap_content"  
            android:text="horizontal2" />  
    </LinearLayout>  
    <Button  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:text="vertical" />  
</LinearLayout>
```

The XML code is annotated with red boxes and labels:

- Layer 1:** The outermost `LinearLayout` node.
- Layer 2:** The `TextView` and the inner `LinearLayout` node.
- Layer 3:** The two `Button` nodes inside the inner `LinearLayout`.

LinearLayout

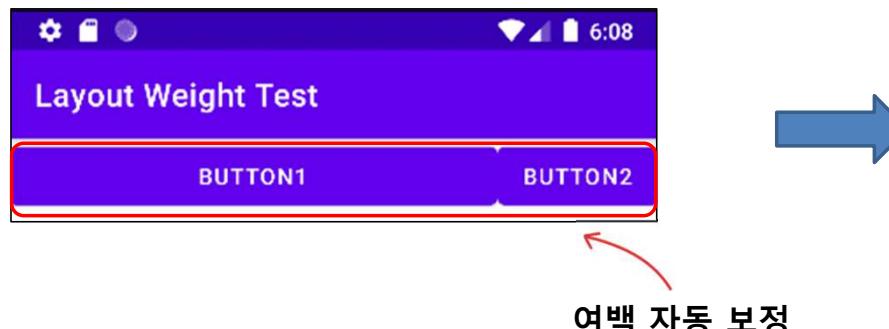
❖ Layout의 속성 – layout_weight

- View에 가중치를 주는 속성으로 각각의 컴포넌트들의 크기를 조절해준다.
- View를 배치하다 보면 가로나 세로방향으로 여백이 생길 수 있다.



```
<Button  
    android:id="@+id/button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Button1" />  
  
<Button  
    android:id="@+id/button2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="button2" />
```

- layout_weight 속성을 통해 여백을 View로 채울 수 있다.



```
<Button  
    android:id="@+id/button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="Button1" />  
  
<Button  
    android:id="@+id/button2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="button2" />
```

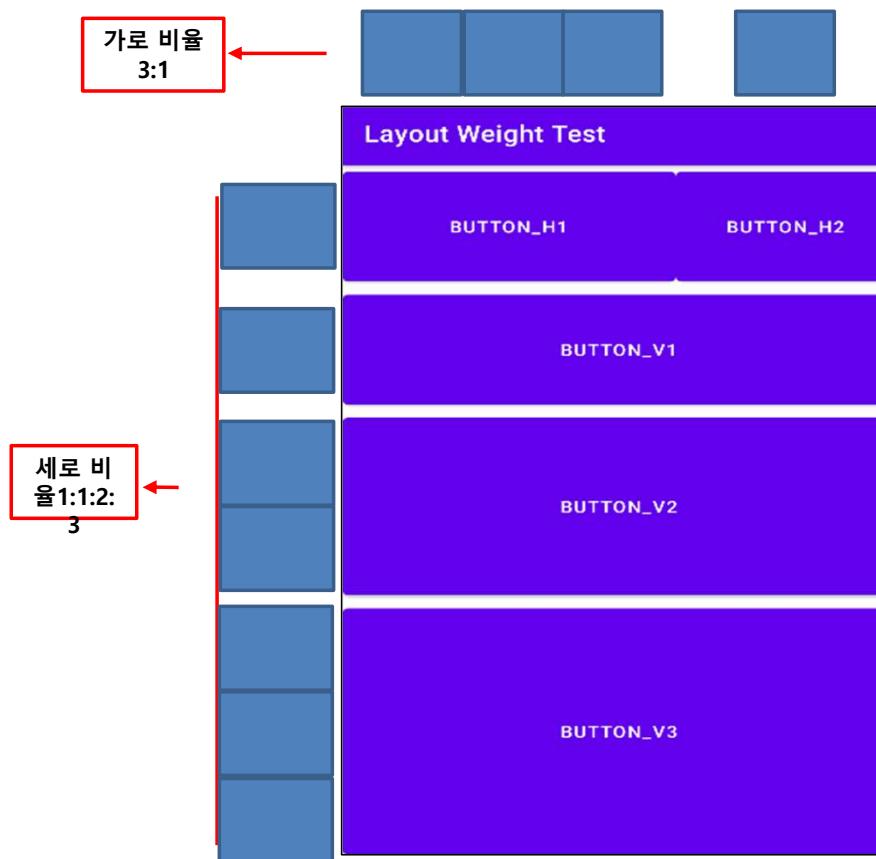
A red arrow points to the "layout_weight" attribute in the first button's XML code, with the text "weight 명시" (specify weight) next to it.

LinearLayout

❖ 중첩된 layout에서 여백 채우기

- 중첩된 layout에서는 `layout_weight`를 통해 지정된 비율로 화면에 배치할 수 있다.

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical">  
    <LinearLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:layout_weight="1"  
        android:orientation="horizontal">  
            <Button  
                android:layout_width="wrap_content"  
                android:layout_height="match_parent"  
                android:layout_weight="3"  
                android:text="Button_h1"/>  
            <Button  
                android:layout_width="wrap_content"  
                android:layout_height="match_parent"  
                android:layout_weight="1"  
                android:text="Button_h2"/>  
        </LinearLayout>  
        <Button  
            android:layout_width="match_parent"  
            android:layout_height="wrap_content"  
            android:layout_weight="1"  
            android:text="Button_V1"/>  
        <Button  
            android:layout_width="match_parent"  
            android:layout_height="wrap_content"  
            android:layout_weight="2"  
            android:text="Button_V2"/>  
        <Button  
            android:layout_width="match_parent"  
            android:layout_height="wrap_content"  
            android:layout_weight="3"  
            android:text="Button_V3"/>  
    </LinearLayout>
```



LinearLayout

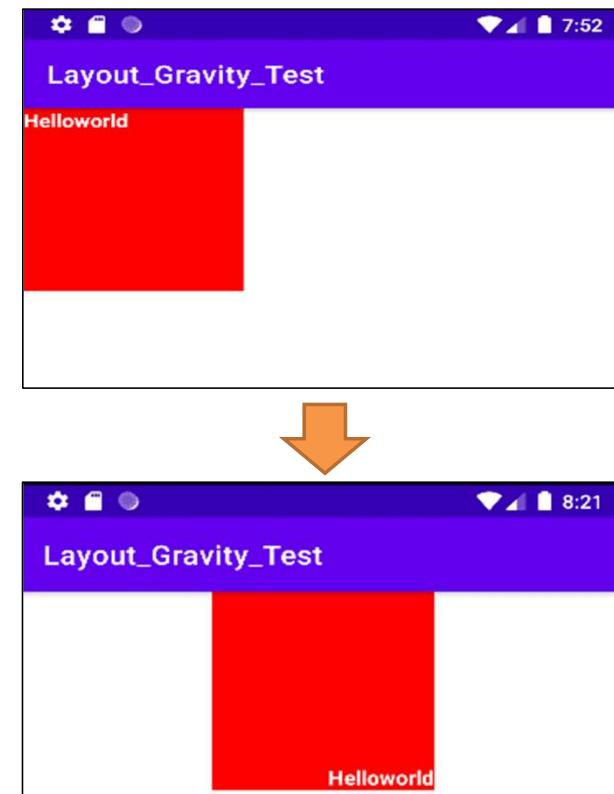
❖ Layout의 속성 – gravity, layout_gravity

- View를 정렬할 때 **gravity, layout_gravity 속성**을 사용한다.
- Gravity, layout_gravity 속성을 사용하지 않으면 **default 값은 left/top**. 즉, 왼쪽상단을 기준으로 정렬한다.
- **Layout_gravity** – 위젯 영역을 정렬한다.
- **Gravity** – View 내부의 콘텐츠(text, etc.)를 정렬한다.

```
<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="150dp"  
        android:layout_height="150dp"  
        android:background="#FF0000"  
        android:text="Helloworld"  
        android:textColor="#FFFFFF"  
        android:textSize="15dp"  
        android:textStyle="bold"  
        android:gravity="right|bottom"  
        android:layout_gravity="center_horizontal"  
    />  
</LinearLayout>
```

내부 콘텐츠 정렬

위젯 영역 정렬



RelativeLayout

❖ RelativeLayout 이란?

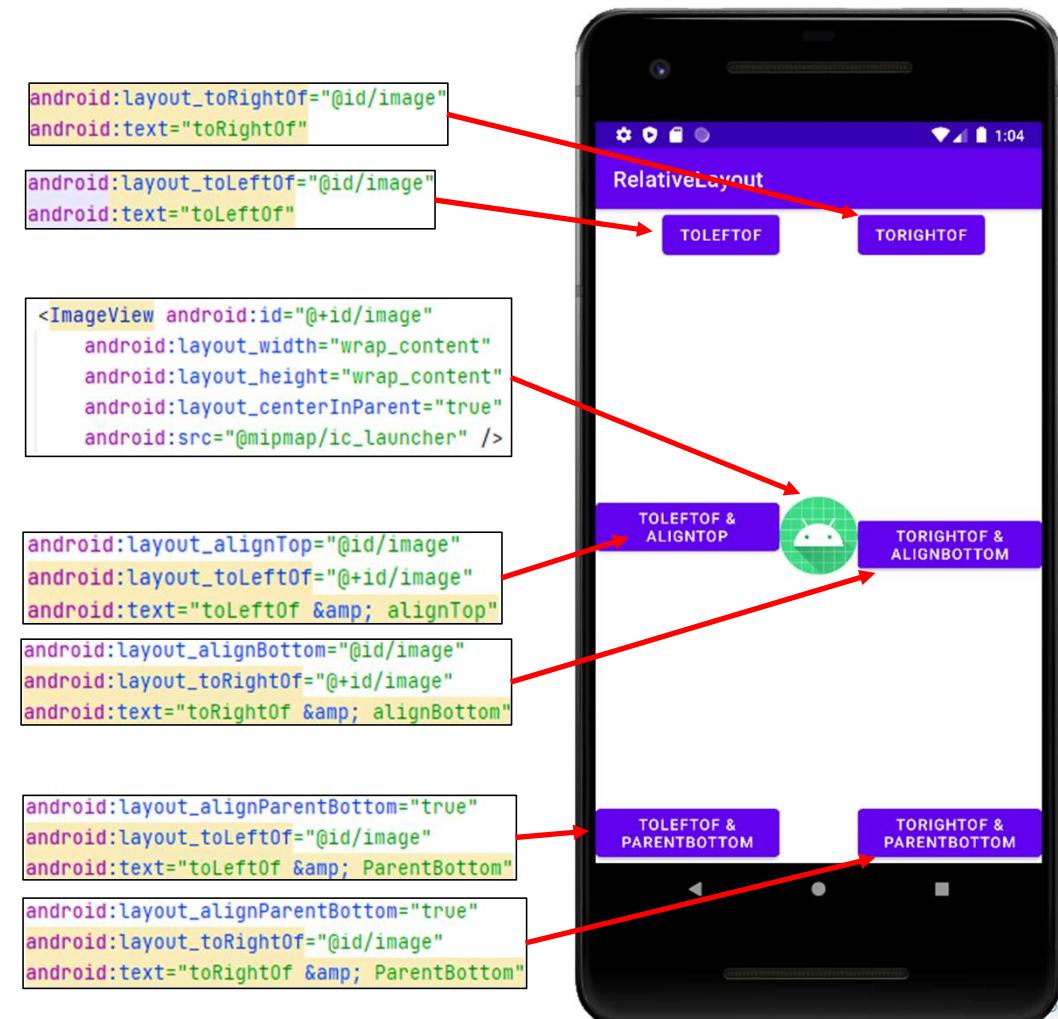
- 상대 View의 위치를 기준으로 정렬하는 Layout 클래스

❖ RelativeLayout의 속성

- 배치 속성과 정렬 속성(align)이 존재함
 - 정렬 속성에는 View를 기준으로 정렬하는 방법, 부모 Layout을 기준으로 정렬하는 방법이 있음

* 기준 View는 id 선언이 필수적이다.

속성	기능
layout_	기준 View 배치 (above, below, toLeftOf, toRightOf)
layout_align	기준 View 정렬 (Top, Bottom, Left, Right)
layout_alignParent	부모 View의 방향에 맞춤 (Top, Bottom, Left, Right)

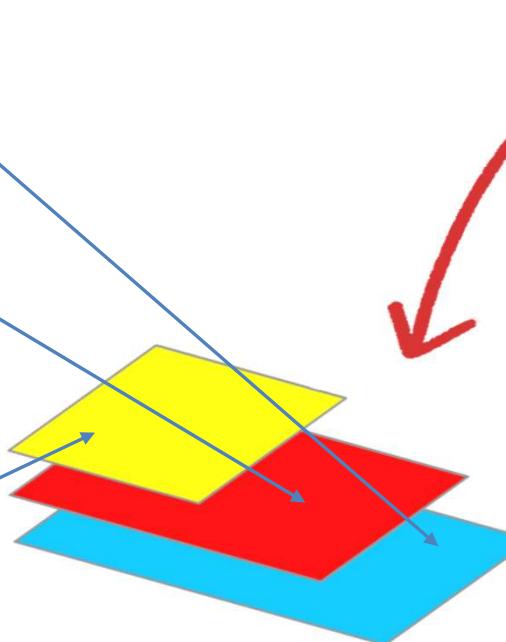


FrameLayout

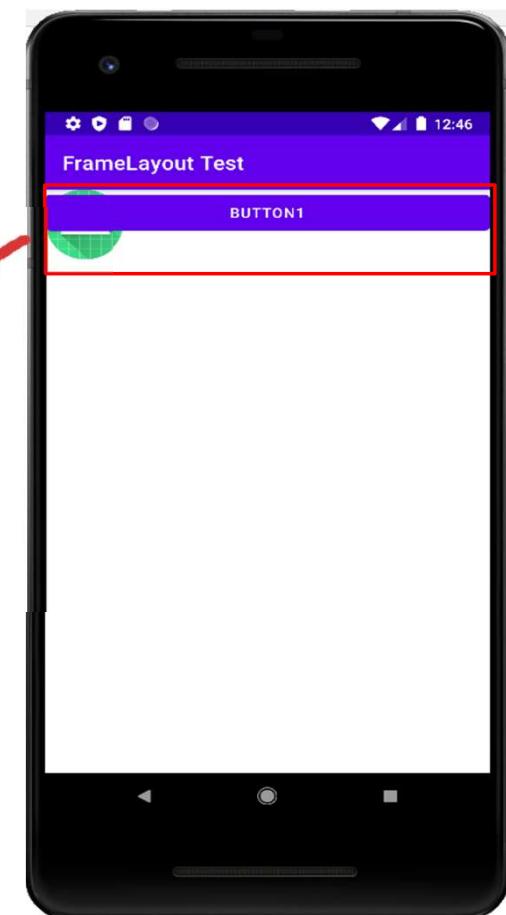
❖ FrameLayout 이란?

- View를 겹쳐서 출력하는 Layout 클래스이다.
- View를 추가한 순서대로 위에 겹쳐서 계속 출력하는 Layout이다.
- 겹쳐진 View 중에서 특정 순간 하나의 View만 출력할 때 사용한다.
- 액티비티 코드에서 원하는 순간에 View의 visibility 속성값을 바꾸어 조절한다.

```
<FrameLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".MainActivity">  
    <ImageView  
        android:id="@+id/imV"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:src="@mipmap/ic_launcher"/>  
    <Button  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:text="Button1" />  
</FrameLayout>
```



Widget
증첩



GridLayout

❖ GridLayout이란?

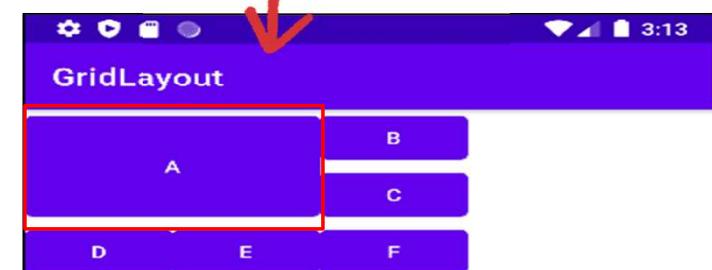
- 행과 열로 구성된 테이블 화면을 만드는 Layout

❖ GridLayout의 설명

- GridLayout의 배치 규칙
 - Orientation – View를 나열할 방향을 설정한다.
 - rowCount, columnCount – 가로·세로로 나열할 View의 개수를 지정한다.
- GridLayout의 특정 View의 위치 조정
 - layout_row, layout_column – View가 위치할 가로, 세로 방향 인덱스를 지정
- GridLayout의 특정 View의 크기 확장 및 병합
 - layout_gravity – 특정 View를 확장해서 출력한다.
 - fill, fill_horizontal, fill_vertical 등 옵션이 존재한다.
 - Layout_columnSpan, layout_rowSpan: 가로 또는 세로로 병합한다.

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="3"
    android:orientation="horizontal">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_rowSpan="2"
        android:layout_columnSpan="2"
        android:layout_gravity="fill"
        android:text="A" />
    <Button...>
    <Button...>
    <Button...>
    <Button...>
    <Button...>
</GridLayout>
```

column(2) + row(2) = 4



실습

실습

- ◆ 실습 (따라하기)
 - ◆ 예제 1 - 화면 색 변경하기
 - ◆ 예제 2 - 버튼 릴레이
 - ◆ 예제 3 - 갤러리 만들기

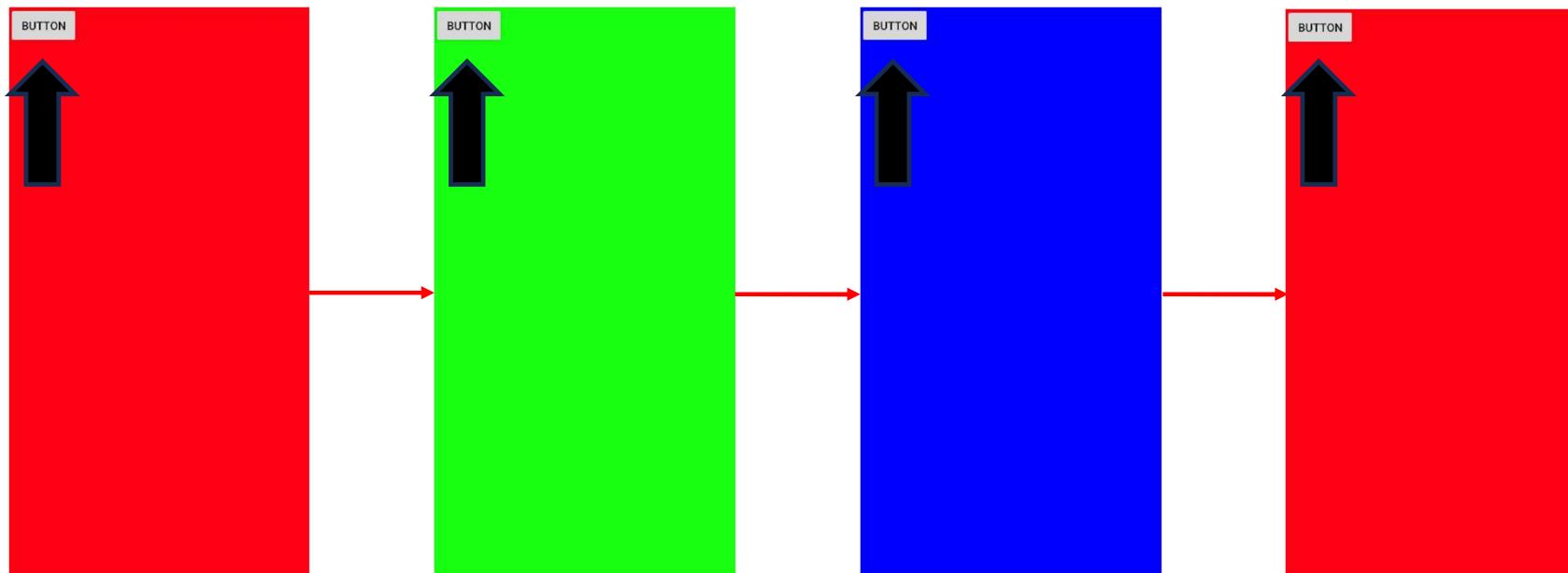
- ◆ 응용
 - ◆ 예제 4 – 계산기 만들기

예제 1 – 화면 색 변경하기

◆ 버튼 터치로 배경색 변경하기

- ◆ 버튼 터치 시 배경화면 색상 전환
 - ◆ 순서: red -> green -> blue -> red
 - ◆ #FF0000, #00FF00, #0000FF
- ◆ 터치가 두번씩 되는 경우는 무시해도 됨
- ◆ 버튼 레이아웃은 자유롭게 사용할 것

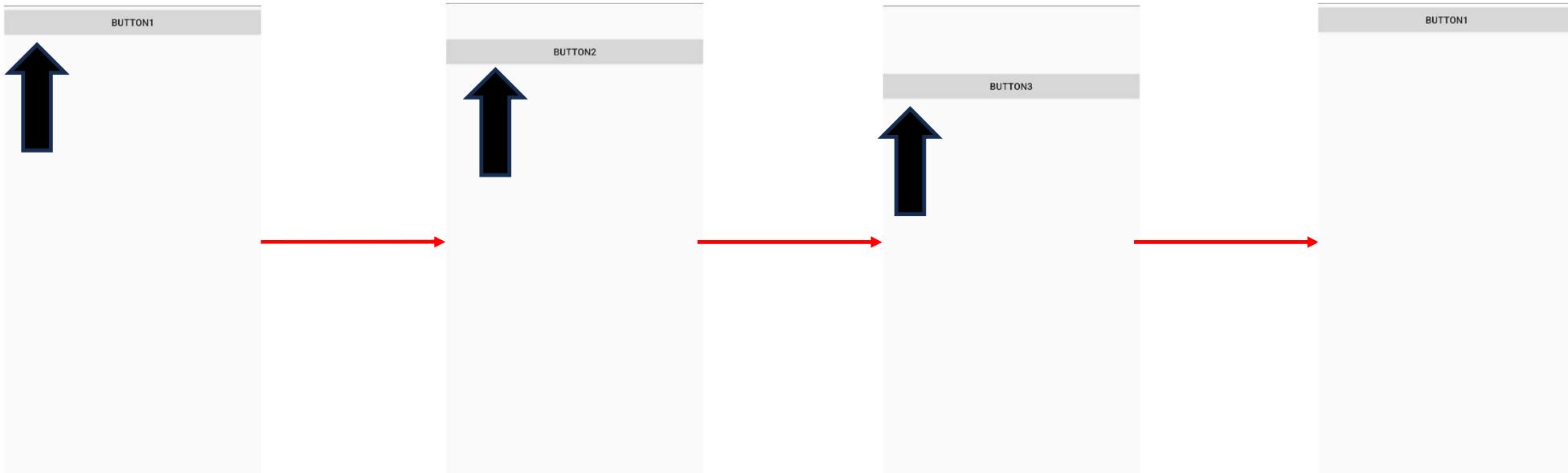
```
lateinit var linLayer : LinearLayout  
linLayer = findViewById<LinearLayout>(R.id.LinLay)  
linLayer.setBackgroundColor(Color.parseColor( colorString: "#FF0000"))
```



예제 2 – 버튼 릴레이

- ◆ 버튼을 누르면 3개의 버튼이 순차적으로 활성화됨
 - ◆ BUTTON1 -> BUTTON2 -> BUTTON3 -> BUTTON1

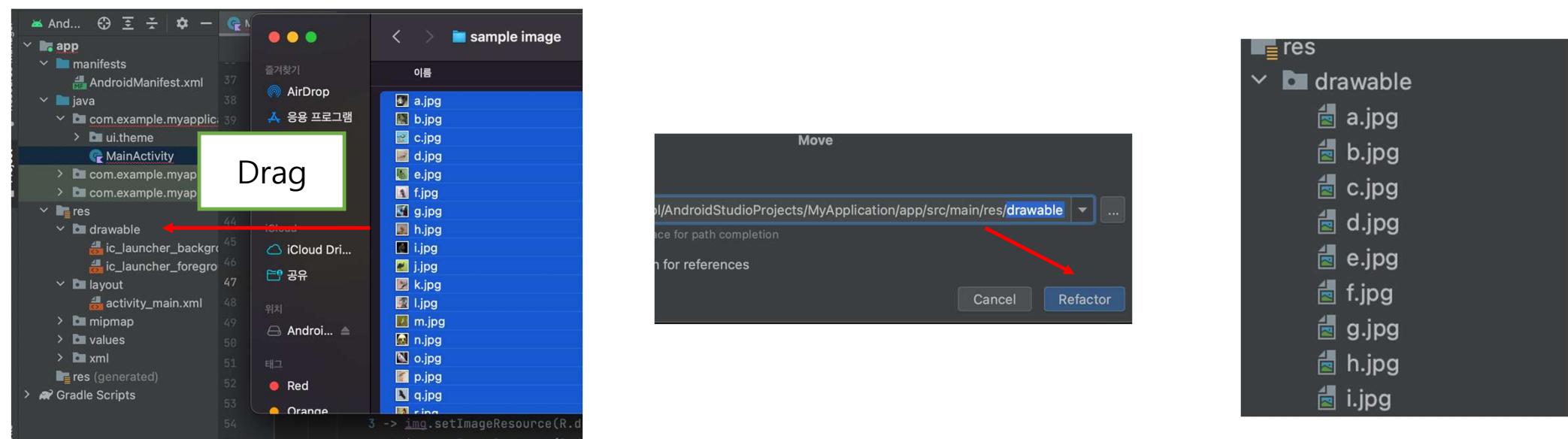
```
btn1.visibility = View.INVISIBLE  
btn2.visibility = View.VISIBLE
```



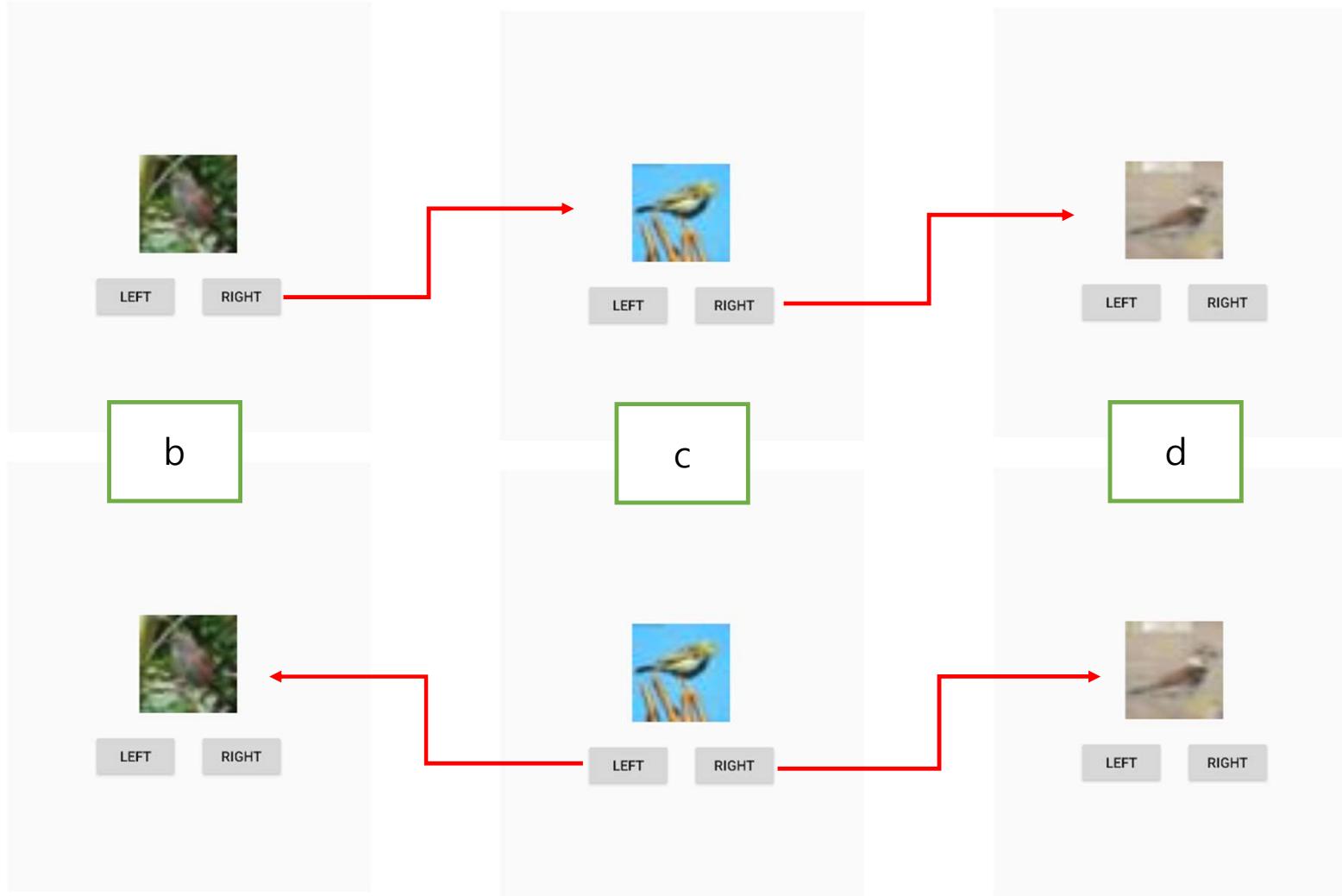
예제 3 – 갤러리 만들기

- ◆ 각 버튼으로 앞 / 뒤 이미지 화면에 출력하기

- ◆ Example > 이미지 파일 26개 넣기 (a~z)
- ◆ 이미지를 App/res/drawable 에 드래그로 넣기
- ◆ 파일이름은 a~z로 지정

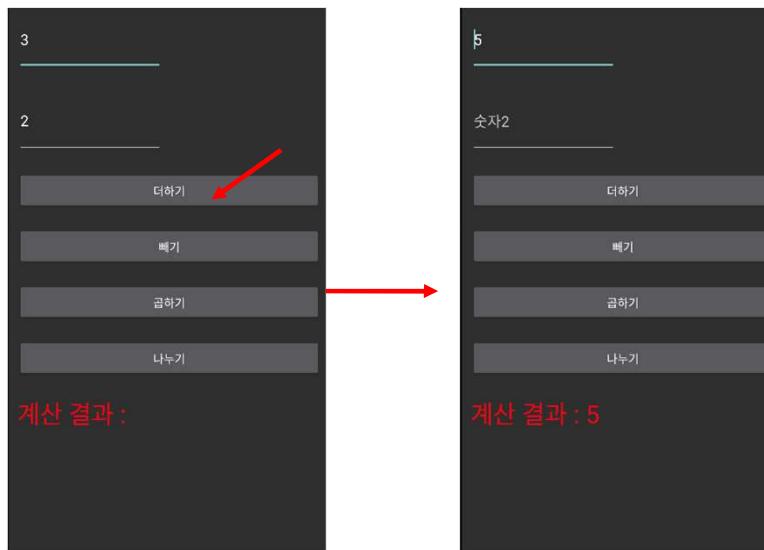


예제 3 – 갤러리 만들기

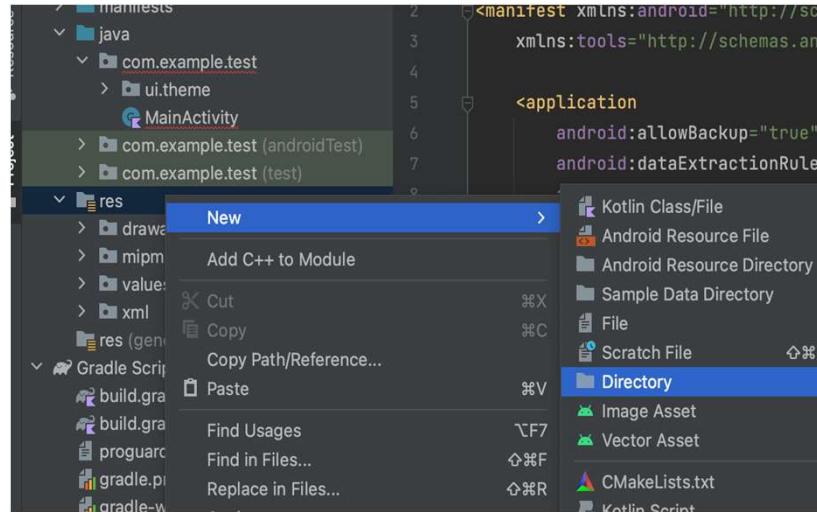
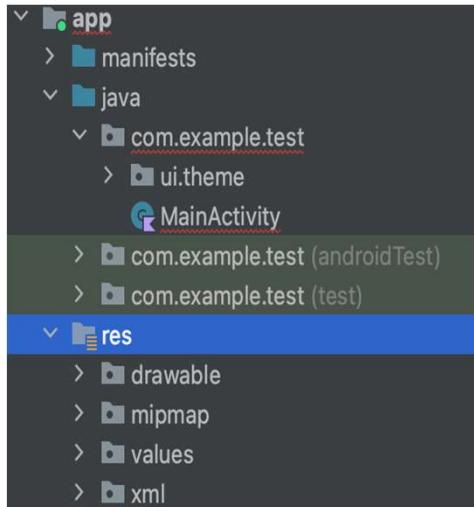


예제 4 – 계산기 만들기

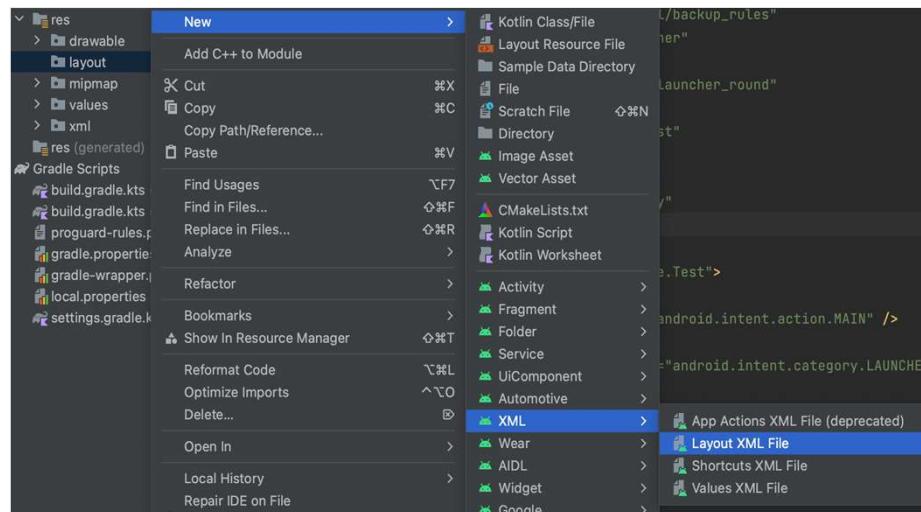
- ◆ 계산기 확장하기
 - ◆ 예외처리하기
 - ◆ 두 개의 숫자가 존재 해야만 연산 수행
 - ◆ 계산 결과를 Num1에 업데이트하고, Num2는 비우기



에러 - Res에 layout이 없을때

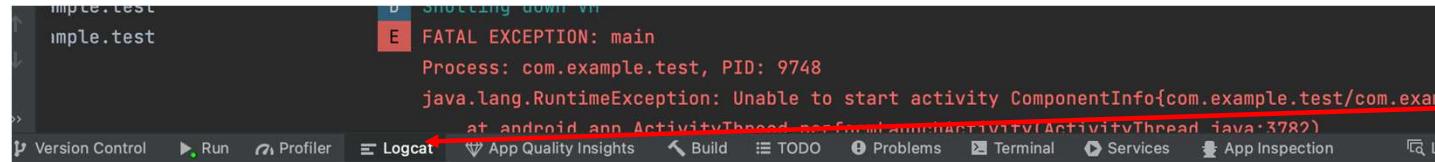


Directory name:
layout



Xml name:
Activity_main.xml

에러 – 비정상 앱 종료시 (실행시 앱 꺼짐)



The screenshot shows the Android Studio interface with the Logcat tab selected. A red arrow points from the word 'logcat' in the green box to the 'Logcat' button in the toolbar. The Logcat window displays the following error message:

```
import test
import test

D: Shutting down VM
E: FATAL EXCEPTION: main
Process: com.example.test, PID: 9748
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.example.test/com.example.test.MainActivity}
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3782)
    at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3687)
    at android.app.servertransaction.LaunchActivityItem.execute(LaunchActivityItem.java:79)
    at android.app.servertransaction.TransactionExecutor.executeCallbacks(TransactionExecutor.java:135)
    at android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.java:95)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:225)
    at android.os.Handler.dispatchMessage(Handler.java:102)
    at android.os.Looper.loop(Looper.java:154)
    at android.app.ActivityThread.main(ActivityThread.java:707)
    at java.lang.reflect.Method.invoke(Native Method)
    at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:493)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:148)
```

logcat

You need to use a Theme.AppCompat theme (or descendant) with this activity.

- ◆ Extends Activity 인지 확인
 - ◆ AppcompatActivity거나 ActionBarActivity 이면 오류 발생
- ◆ Activity를 쓸 수 없을 경우
 - ◆ `Android:theme="@style/Theme.Appcompat"` 추가