Layout : Part II

Mobile Software

2021 Fall

What to do next?

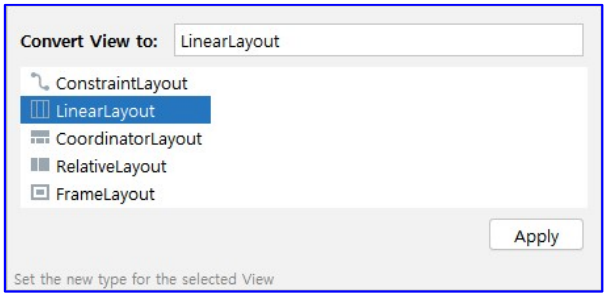
- **Layout**
  - **LinearLayout**
  - RelativeLayout
  - FrameLayout
  - TableLayout

- 실습: UI를 구현하는 3가지 coding style

- **강의 노트에 포함된 코드: 4장-소스코드.hwp**

프로젝트 생성 + 화면 레이아웃

- 새 프로젝트 만들기
  - **Project name** : **Layout Example**
  - **Package name** : **edu.ourincheon.layoutexample**
  - **Activity** : **Empty Activity**
  - **Activity name** : **MainActivity.kt**
  - **Layout name** : **activity_main.xml**
- **자동 생성된 XML 파일**의 root view는 **ConstraintLayout**
  - 이를 **LinearLayout**으로 변경
    - **Component Tree 창 > ConstraintLayout** 클릭 > 오른쪽 버튼
    - **Convert View > LinearLayout > Apply**
  - TextView 도 삭제

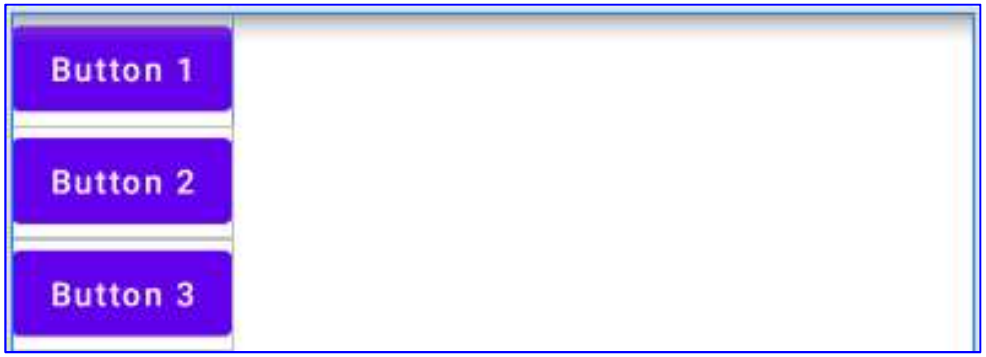


LinearLayout

- 속성
  - **orientation: 수직 또는 수평 방향으로 view를 배치**
  - **layout_gravity 와 gravity**
  - **layout_weight**
  - **baselineAligned**

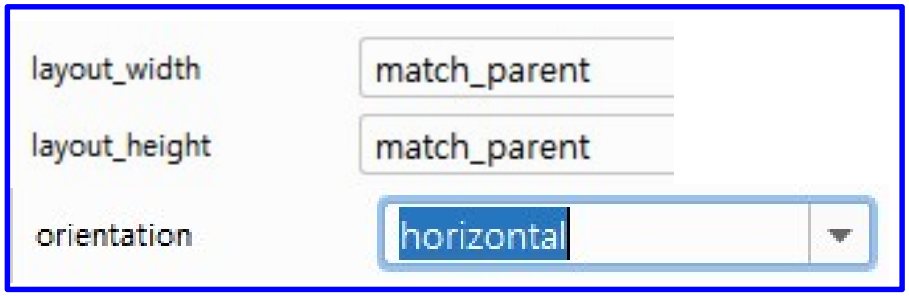
orientation="**horizontal**"

orientation="**vertical**"

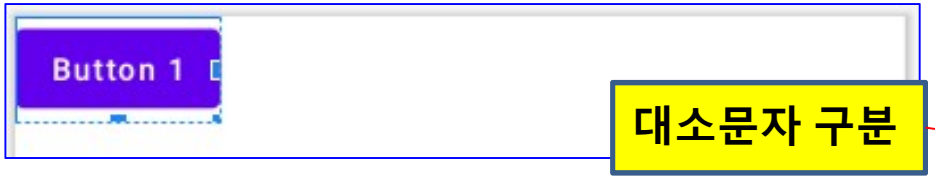


버튼 3개를 수평 방향으로 배치

1. LinearLayout 속성 : **orientation**



2. Button 1 Drag & Drop ➔ 속성 설정



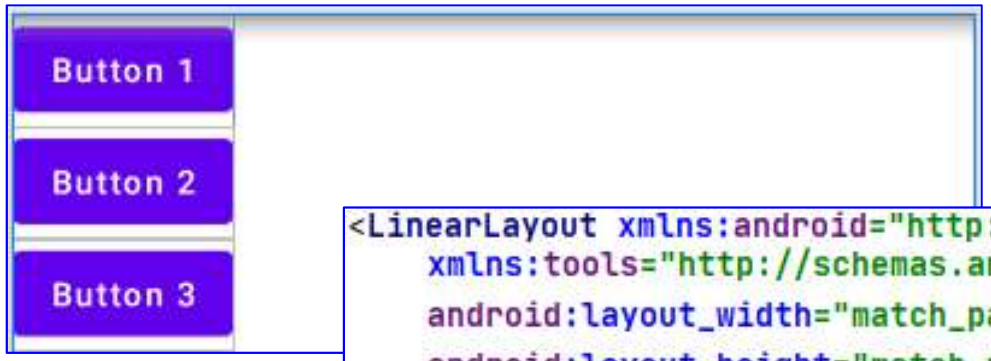
3. Button 2 Drag & Drop ➔ 속성 설정

버튼 3개를 수직 방향으로 배치

4. Button 3 Drag & Drop > 속성 설정

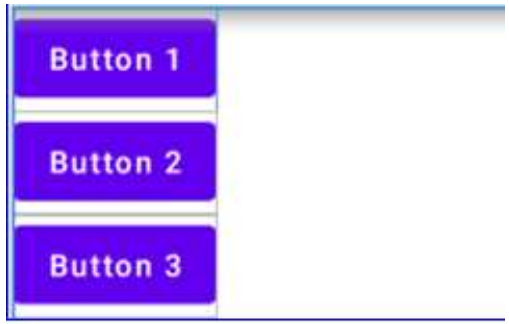

```
<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button 3"
    android:textAllCaps="false" />
```

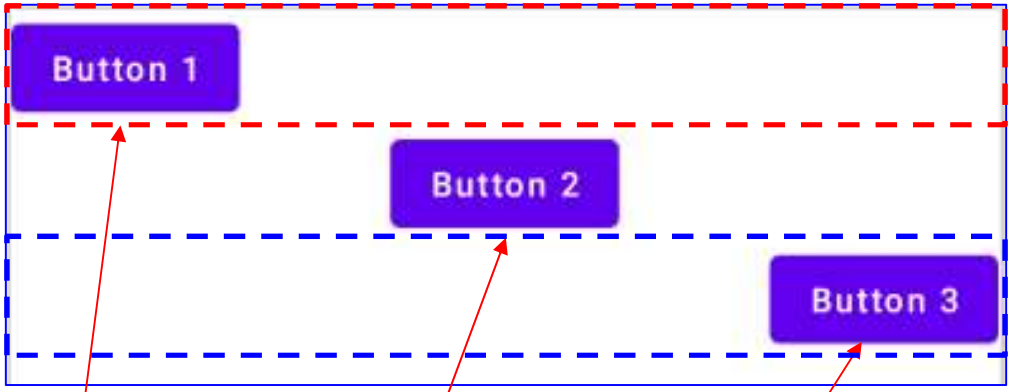
5. LinearLayout의 **orientation** 속성 변경
horizontal ➔ **vertical**




```
<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=".MainActivity">
```

View를 배치할 위치: layout_gravity





layout_gravity	left
bottom	false
clip_horizontal	false
center	false
clip_vertical	false
start	false
right	false
center_horizontal	false
fill	false
fill_horizontal	false
top	false
left	true
center_vertical	false
fill_vertical	false
end	false

```
android:layout_gravity="left"
```

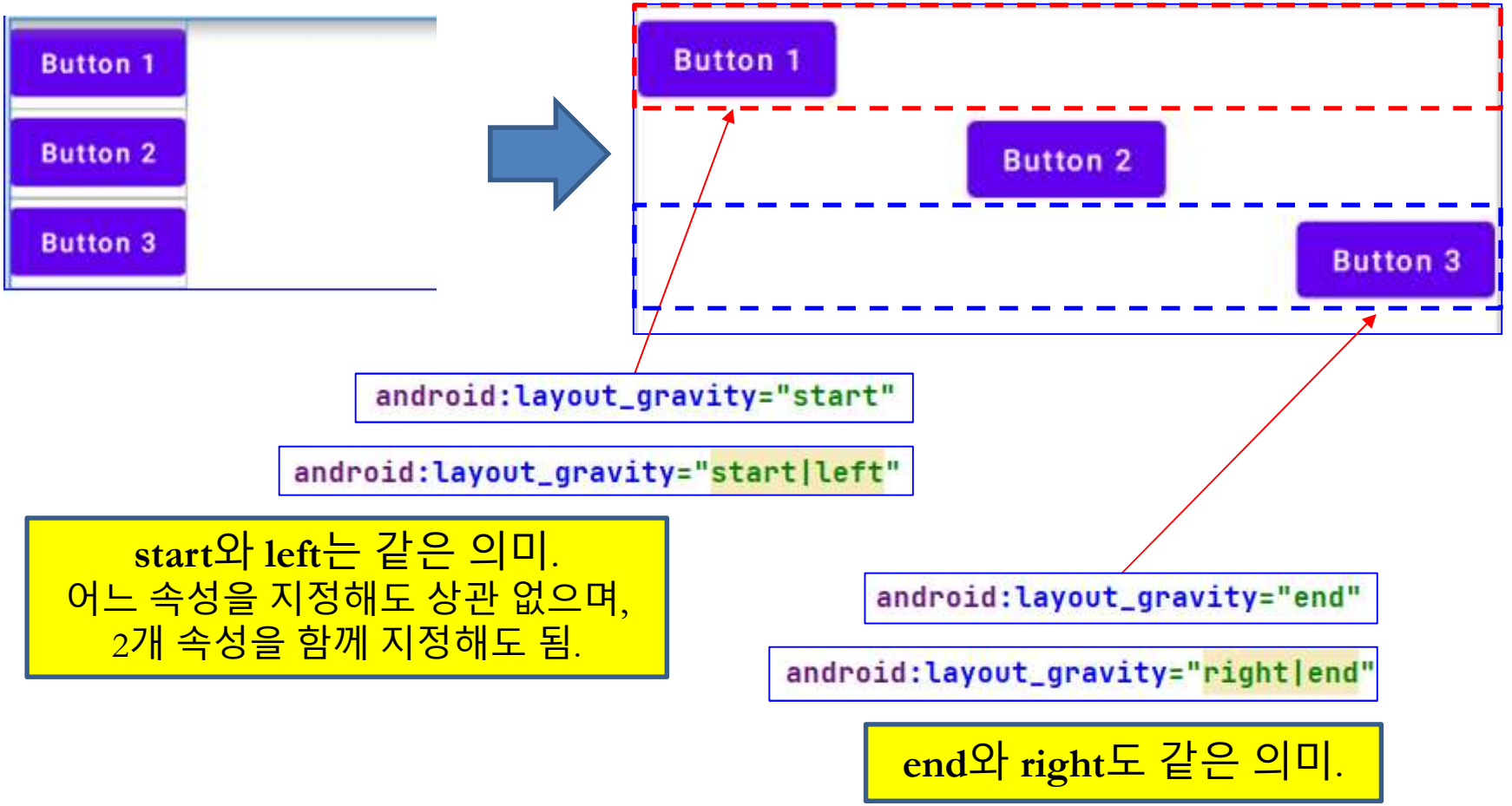
```
android:layout_gravity="center_horizontal"
```

```
android:layout_gravity="right"
```

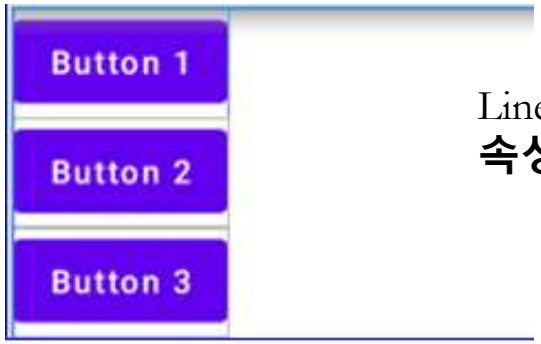
체크박스를
체크하면
해당 속성이
true로 바뀜.

view를 배치할 공간에 여유가 있을 때
layout_gravity 속성을 설정.

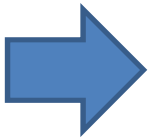
View를 배치할 위치: layout_gravity

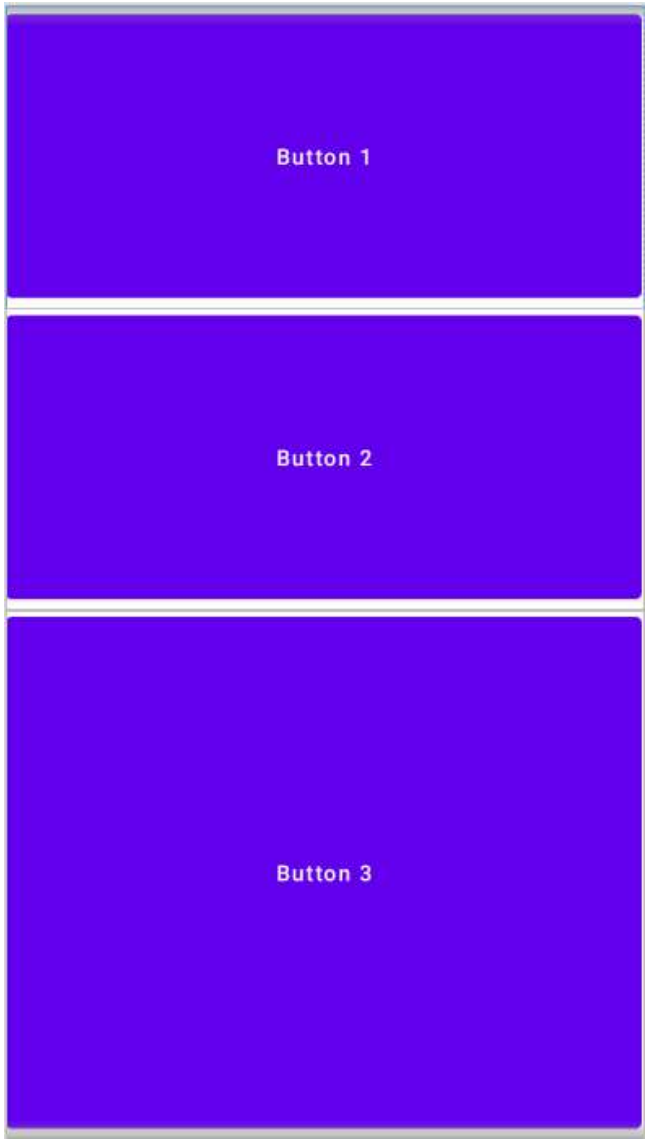


View가 차지하는 공간을 달리 함: layout_weight



LinearLayout의 **orientation 속성은 vertical.**





1. 모든 Button의 **width** 속성을 아래와 같이 변경

2. **Button 1과 Button 3**의 layout_weight 속성을 1로 설정

```
android:layout_weight="1"
```

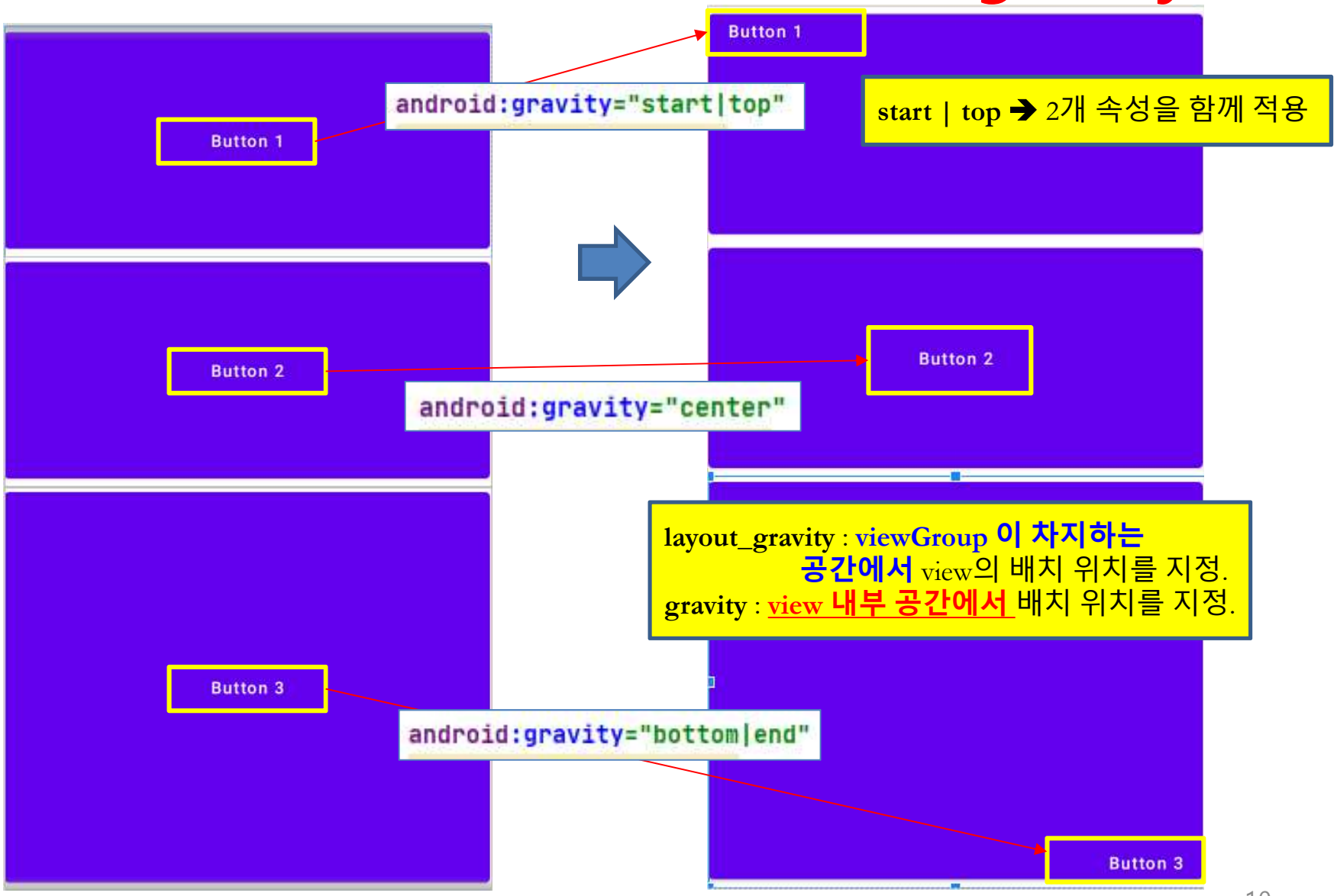
3. **Button 2**의 layout_weight 속성은 2로 설정

```
android:layout_weight="2"
```

4. 전체 높이를 기준으로 아래와 같이 배분

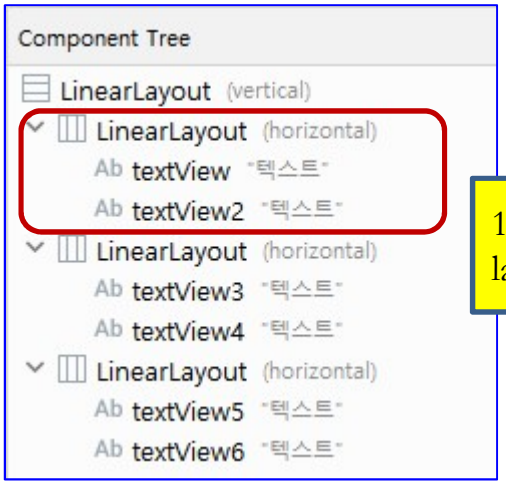
Button 1 : Button 2 : Button 3 = 1 : 1 : 2 = 0.25 : 0.25 : 0.5

View 내부에서 문자열 배치 위치를 결정: gravity



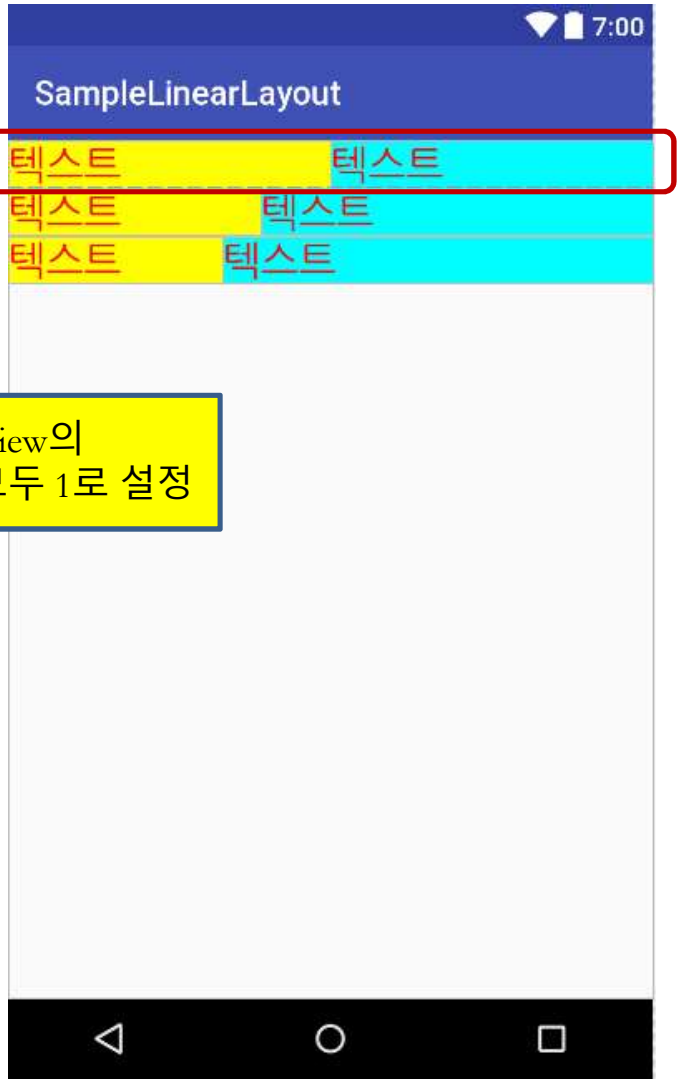
view 크기와 layout_weight 속성 관계 (1/3)

부모 LinearLayout (vertical) 안에
3개의 LinearLayout (horizontal)이 있음



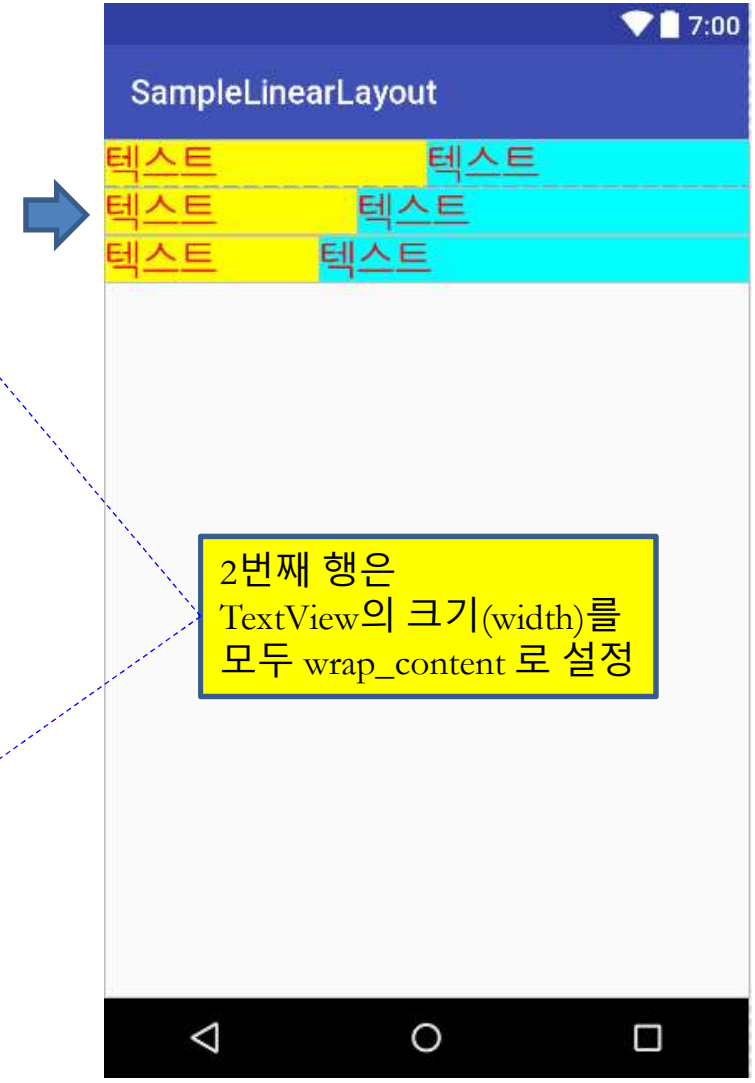
1번째 행은 TextView의
layout_weight를 모두 1로 설정

자식 LinearLayout (horizontal)은
각각 2개의 TextView를 갖고 있음.



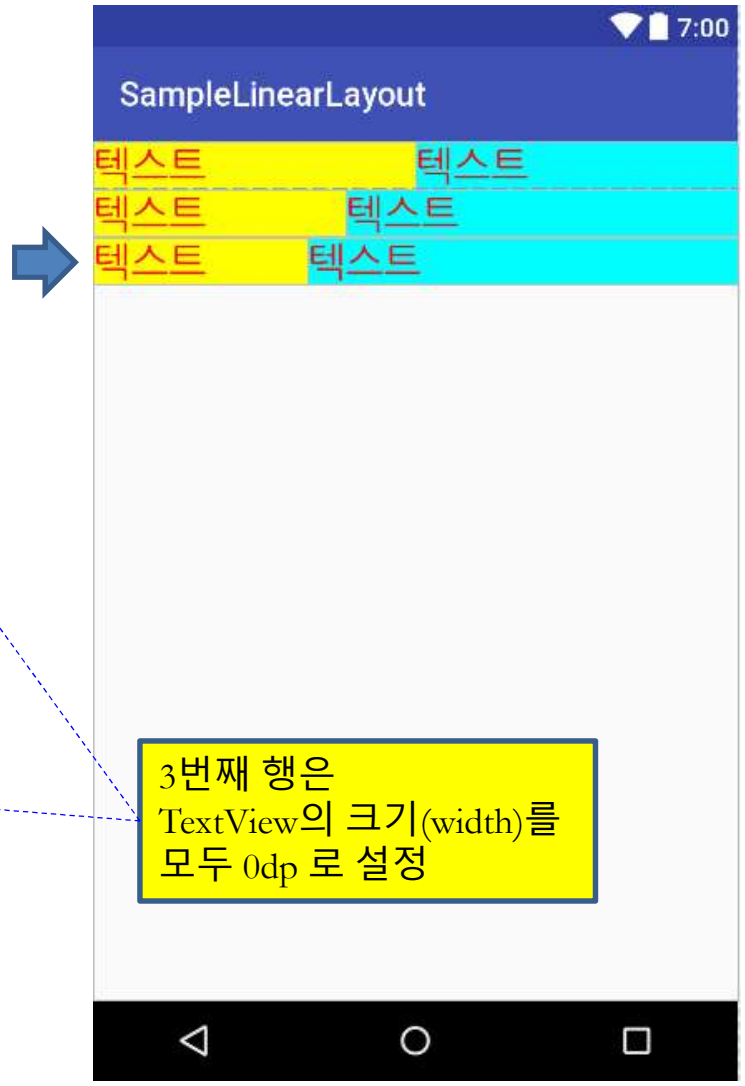
view 크기와 layout_weight 속성 관계 (2/3)

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#ffffff00"
        android:text="텍스트"
        android:textColor="#ffff0000"
        android:textSize="24dp"
        android:layout_weight="1" />
    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#ff00ffff"
        android:text="텍스트"
        android:textColor="#ffff0000"
        android:textSize="24dp"
        android:layout_weight="2" />
</LinearLayout>
```



view 크기와 layout_weight 속성 관계 (3/3)

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:id="@+id/textView5"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="#ffffff00"
        android:text="텍스트"
        android:textColor="#ffff0000"
        android:textSize="24dp"
        android:layout_weight="1" />
    <TextView
        android:id="@+id/textView6"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="#ff00ffff"
        android:text="텍스트"
        android:textColor="#ffff0000"
        android:textSize="24dp"
        android:layout_weight="2" />
</LinearLayout>
```



Colors

- 투명도와 빛의 3원색인 RGB값을 지정
  - "#**RRGGBB**"
  - "#AARRGGBB"
    - 16진수 사용
      - 00(0)~FF(255)
    - R-빨간색, G-녹색, B-파란색
    - A(alpha)-투명도
      - 00: 투명, FF: 불투명

문자열의 baseline 일치: baselineAligned (1/2)

- **TextView 3개의 textSize를 달리해서 수평 배치**
  - **LinearLayout** 의 orientation="**horizontal**"
  - Palette 창 > **Text** > **TextView**
    - 3개의 **TextView** 를 순서대로 삽입
      - width, height 모두 "**wrap_content**"로 지정

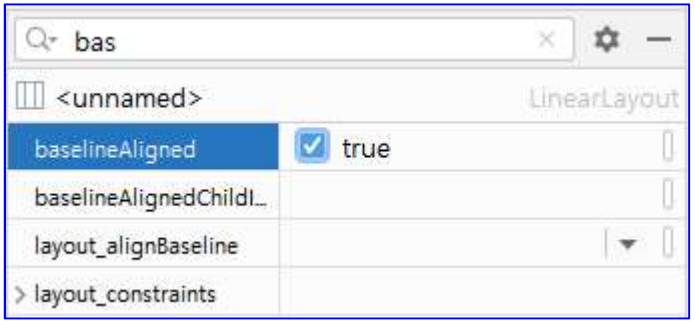
layout_weight 가 1로 자동 설정되어 있으면 지우거나 0을 입력.

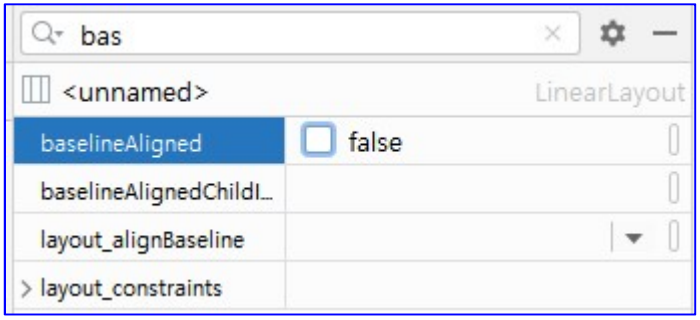
  - Component tree > **TextView** 를 순서대로 클릭
    - Properties window의 검색 창에서 "text" 입력
    - text="**The**", textAllCaps="**false**", textSize="**20sp**"
    - text="**World**", textAllCaps="**false**", textSize="**30sp**"
    - text = "**of Android**", textAllCaps="**false**", textSize="**40sp**"

문자열의 크기 단위는 dp가 아닌 sp를 사용

  - Component tree > **LinearLayout** > Properties window
    - **baselineAligned** > check-box 클릭 > 체크 표시

문자열의 baseline 일치: baselineAligned (2/2)






```
<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:baselineAligned="false"
    android:orientation="horizontal"
    tools:context=".MainActivity">
```

gray 폰트로 표시된 것은 **xmlns:app**에서 정의한 속성을 사용하지 않았기 때문

layout_margin과 padding (1/2)

- **layout_margin**
  - 부모 container와 자식 view 사이 여백
- **padding**
  - View의 (바깥) 테두리와 view의 실제 내용 사이의 간격

- Padding
  - **padding**Left
  - paddingRight
  - paddingTop
  - **padding**Bottom
- Margin
  - **layout_margin**Left
  - **layout**_marginRight
  - **layout**_marginTop
  - **layout_margin**Bottom

layout_margin과 padding (2/2)

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="2dp"
    android:layout_margin="2dp"
    android:text="Padding=2, Margin=2" />
```

```
<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="16dp"
    android:layout_margin="16dp"
    android:text="Padding=16, Margin=16" />
```

```
<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="32dp"
    android:layout_margin="32dp"
    android:text="Padding=32, Margin=32" />
```

Padding이 커질수록
Button 크기도 점점 커짐.

What to do next?

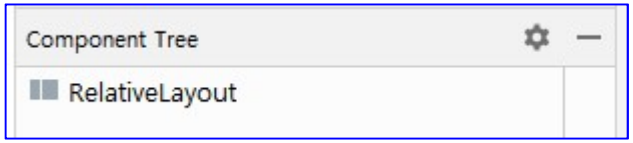
- **Layout**
  - LinearLayout
  - **RelativeLayout**
  - FrameLayout
  - TableLayout
- 실습: UI를 구현하는 3가지 coding style

RelativeLayout

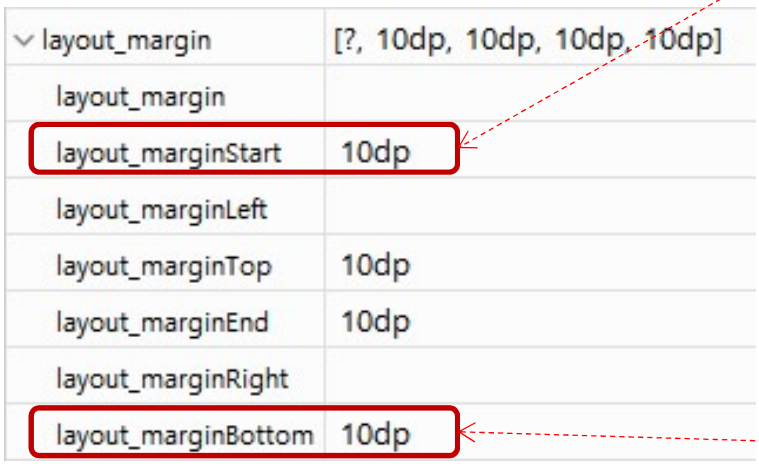
- 부모 container나 다른 view의 위치를 기준으로 view의 배치 위치를 결정
  - 부모 container나 다른 view의 위치는 어떻게 알 수 있을까?
    - RelativeLayout에서는 **모든 view에 대해 id를 정의**
      - "@+id/***identifier***"
    - 다른 view의 id를 참조 ➔ "@+id/***다른 view의 id***"
  - 상대 위치를 지정하는 속성
    - layout_alignParent**Top**, layout_alignParent**Bottom**
    - layout_alignParent**Left**, layout_alignParent**Right**
    - layout_**above**, layout_**below**
    - layout_**toLeftOf**, layout_**toRightOf**
    - layout_align**Top**, layout_align**Bottom**, …
    - layout_align**Baseline**

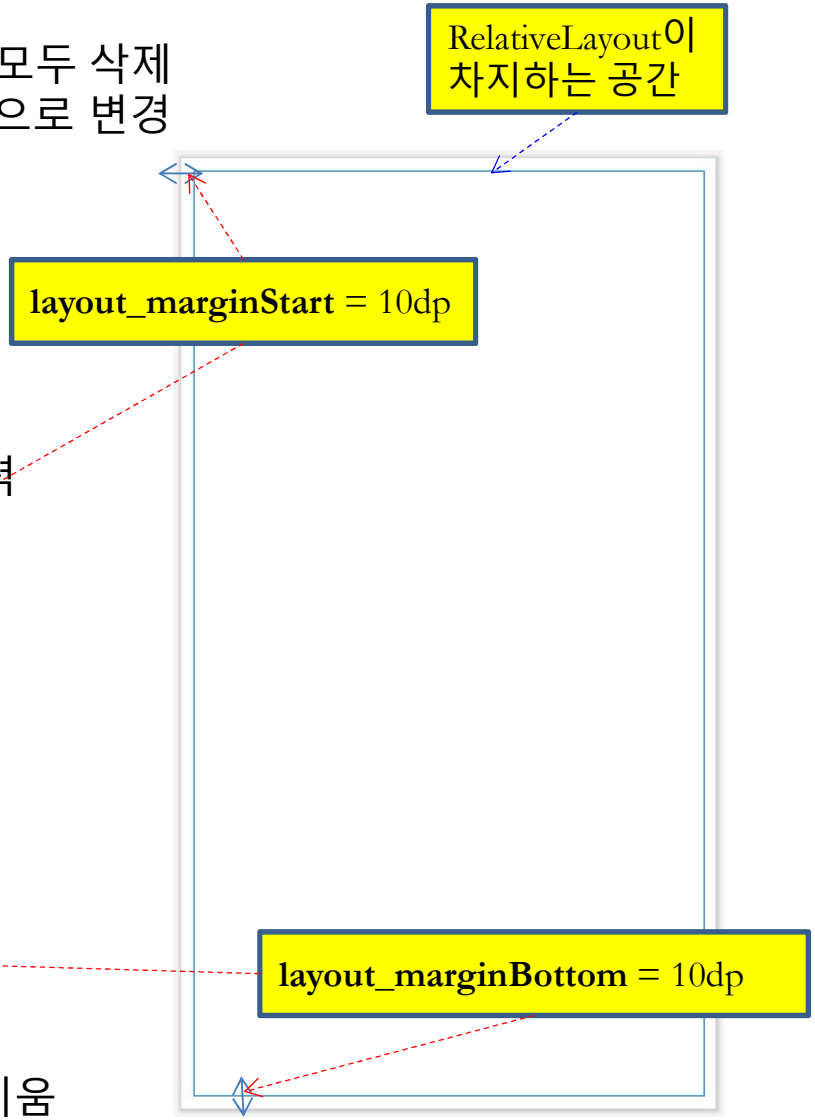
RelativeLayout : layout_margin

1. 이전에 만들었던 view는 root view만 남겨놓고 모두 삭제
   ➔ **Convert view**: LinearLayout 을 RelativeLayout으로 변경



2. **layout_margin 설정 연습** : RelativeLayout 클릭
   Properties window의 검색 창에서 "**margin**" 입력

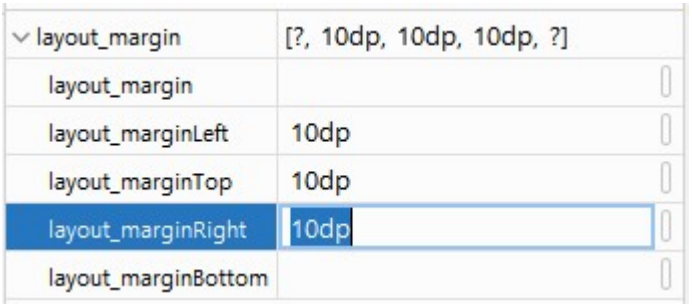




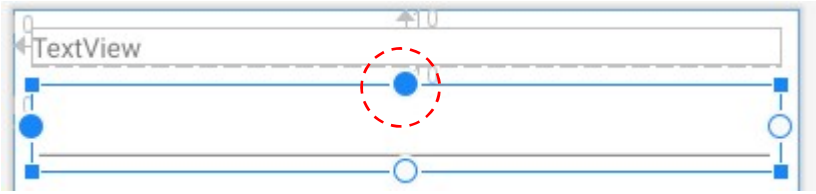
3. **layout_margin 원래대로 복원** : 입력을 모두 지움

주소 입력 화면 (1/3)

1. TextView drag & drop (**id=textView**)
   ➔ width는 match_parent, height는 wrap_content로 지정
   ➔ layout_margin은 layout_marginBottom만 제외하고 모두 10dp 로 지정

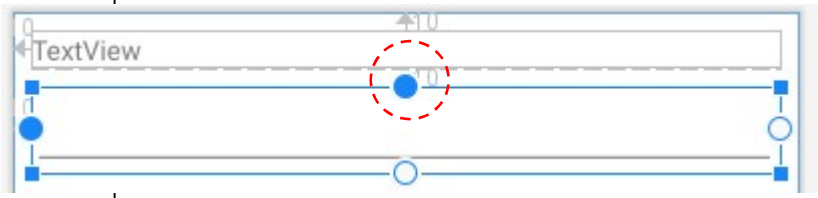


2. Palette 창 > Text > Postal Address (EditText) drag & drop
   ➔ **id = addressEditText**
   ➔ width는 match_parent, height는 wrap_content로 지정
   ➔ EditText의 top 핸들을 TextView의 Bottom 핸들에 연결 ➔ **RelativeLayout**
   ➔ layout_margin은 layout_marginBottom만 제외하고 모두 10dp 로 지정



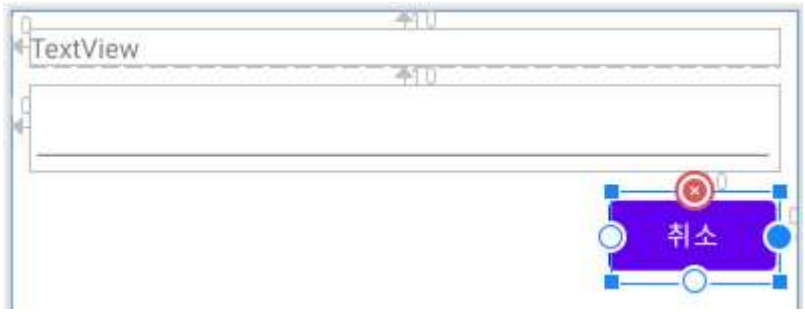
RelativeLayout : 지금까지 만들어진 XML

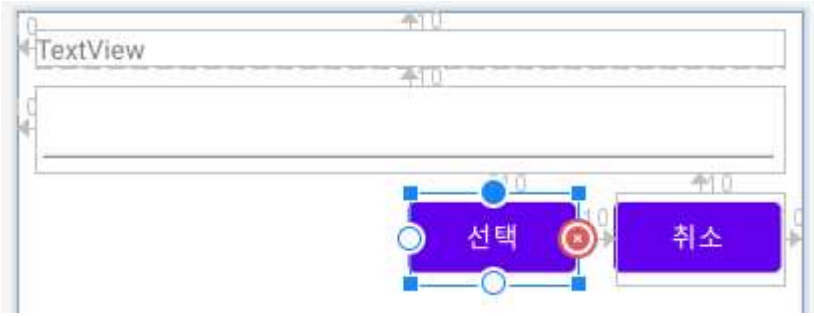
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="10dp"
        android:text="TextView" />

    <EditText
        android:id="@+id/addressEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView"
        android:layout_marginStart="10dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="10dp"
        android:ems="10"
        android:inputType="textPostalAddress" />
</RelativeLayout>
```



주소 입력 화면 (2/3)

1. Palette 창 > Buttons > Button drag & drop
   - ➔ **id = cancelButton, text = "취소"**
   - ➔ cancelButton의 top 핸들을 **addressEditText** 의 Bottom 핸들에 연결
   - ➔ **layout_alignParentRight = true**
   - ➔ layout_margin은 layout_marginBottom만 제외하고 모두 10dp 로 지정
   - ➔ width, height 모두 wrap_content로 지정

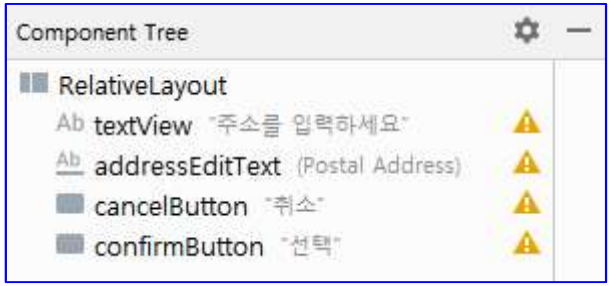




2. Palette 창 > Buttons > Button (EditText) drag & drop
   - ➔ **id = confirmButton**
   - ➔ **text = "선택"**
   - ➔ width, height 모두 wrap_content로 지정
   - ➔ confirmButton의 right 핸들을 **cancelButton** 의 Left 핸들에 연결
   - ➔ confirmButton의 top 핸들을 **addressEditText** 의 Bottom 핸들에 연결
   - ➔ layout_margin은 layout_marginTop, layout_marginRight만 10dp 로 지정

완성된 주소 입력 화면 (3/3)

TextView 의 text 속성에
"**주소를 입력하세요**" 입력



start와 left는 같은 의미.
end와 right는 같은 의미.

```
<Button
    android:id="@+id/cancelButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/addressEditText"
    android:layout_alignParentRight="true"
    android:layout_marginTop="10dp"
    android:layout_marginRight="10dp"
    android:text="취소" />

<Button
    android:id="@+id/confirmButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/addressEditText"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="10dp"
    android:layout_toStartOf="@+id/cancelButton"
    android:text="선택" />
```

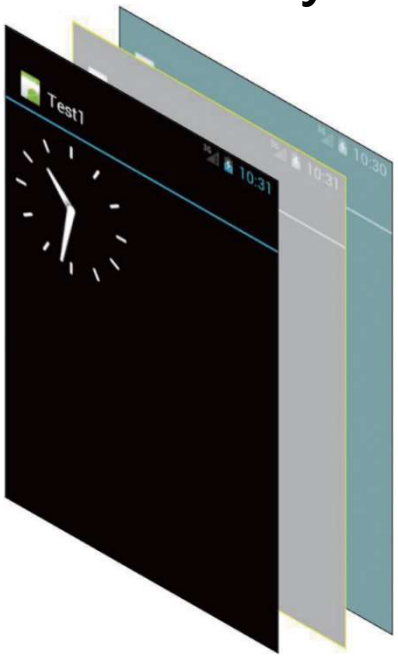
What to do next?

- **Layout**
  - LinearLayout
  - RelativeLayout
  - **FrameLayout**
  - TableLayout

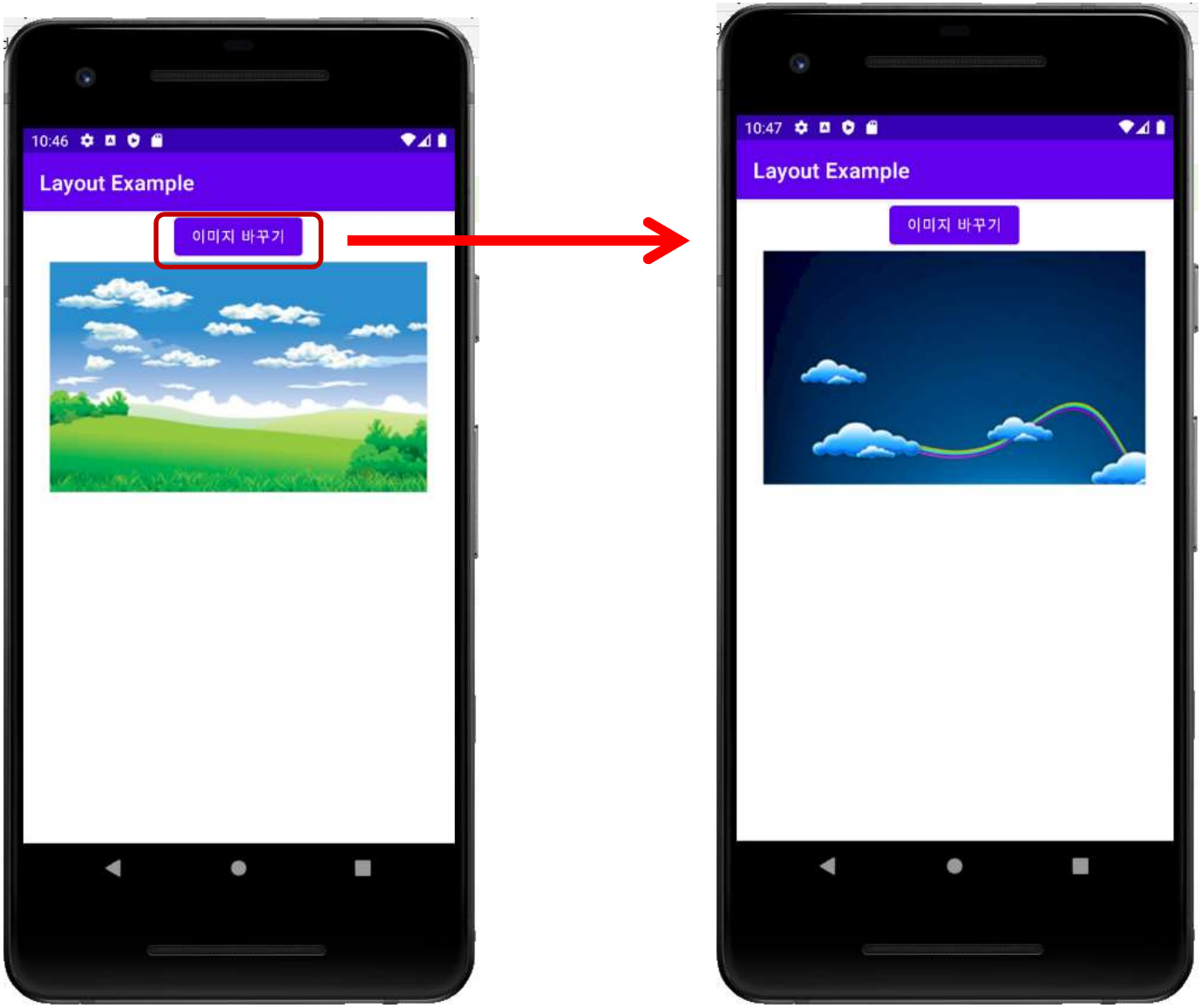
- 실습 : UI를 구현하는 3가지 coding style

FrameLayout

- 여러 개의 view를 겹쳐서 배치
  - 배치 기준은 맨 위 상단 (*upper left corner*)
  - 선언한 순서대로 배치
  - Child view의 가시성(visibility)은 속성 값을 설정
    - **레이아웃 >>** android: **visibility** = "**visible**"
    - **코드 >>** imageView. **visibility** = View. **VISIBLE**

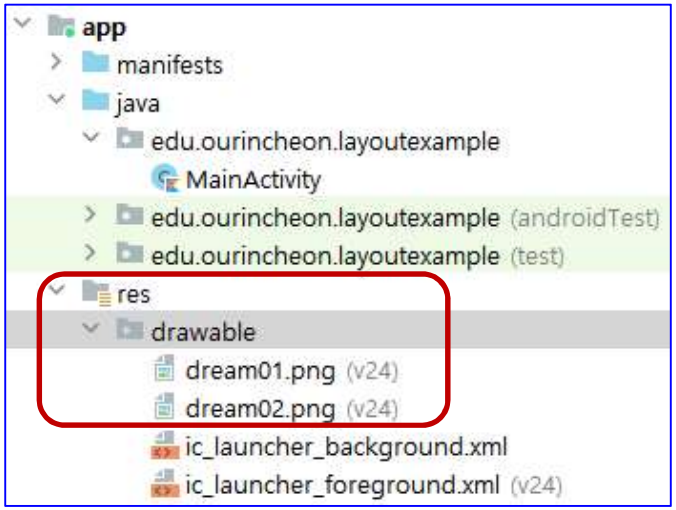


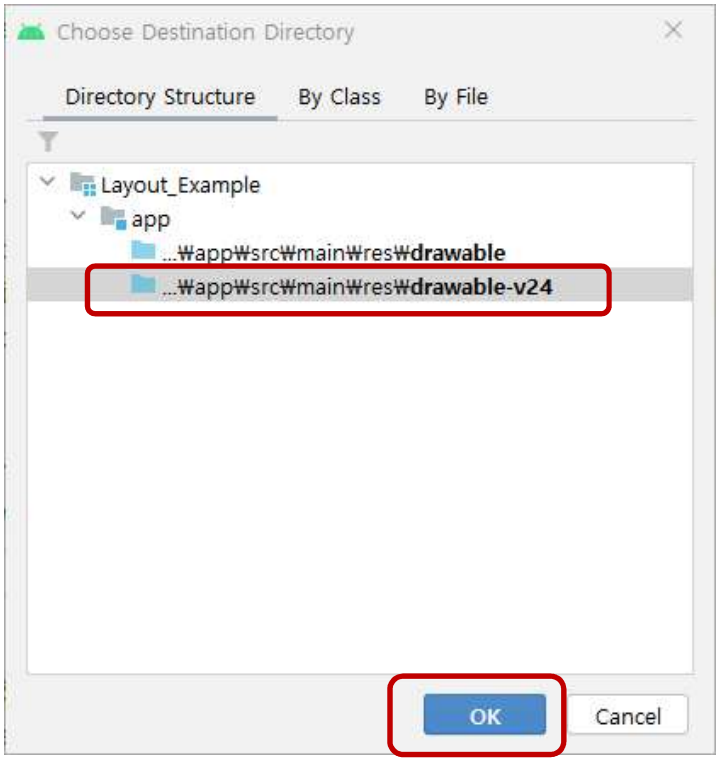
FrameLayout 예



이미지 리소스 가져오기

1. 이미지 파일을 **res/drawable** 폴더에 drag & drop





drawable-v24: 최근 출시된 android API 버전을 사용하는 단말과 호환성(density, 화면 크기 등)을 유지하기 위한 folder.
숫자 24는 API 버전을 가리킴.
drawable : 이전 버전의 안드로이드 API를 사용하는 단말과 호환성을 유지하기 위한 folder.

화면 구성(1/2): LinearLayout 안에 FrameLayout

1. LinearLayout의 **orientation** 속성 (vertical)

2. Button 속성
   - ➔ layout_width, layout_height 모두 "wrap_content"
   - ➔ text ="이미지 바꾸기"
   - ➔layout_gravity="center"
   - ➔onClick="onButtonClicked"

클릭 이벤트 핸들러 코드에서 구현

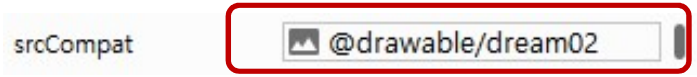
3. Palette > Layouts > FrameLayout 추가
   - ➔ layout_width="match_parent"
   - ➔ layout_height="wrap_content"

화면 구성(2/2): FrameLayout에 2개 이미지 추가

4. FrameLayout 아래에 ImageView 추가
   - ➔ id = "**imageView**"
   - ➔ layout_width="wrap_content"
   - ➔ layout_height="wrap_content"
   - ➔ srcCompat 속성 (dream01.png) 확인

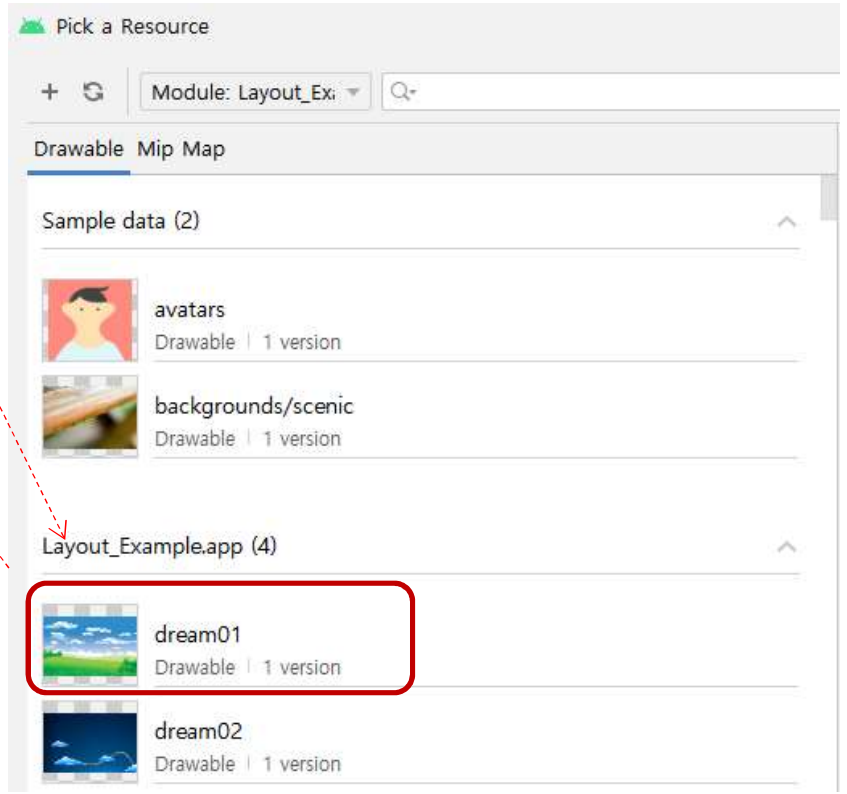
   - ➔ visibility = "visible"
   - ➔ layout_gravity="center"

5. imageView 아래에 imageView2 추가
   - ➔ id = "**imageView2**"
   - ➔ layout_width="wrap_content"
   - ➔ layout_height="wrap_content"
   - ➔ srcCompat 속성 (dream02.png) 확인



   - ➔ visibility = "invisible"
   - ➔ layout_gravity="center"

2개 이미지가
겹쳐 있기 때문에
이미지 1개는 invisible



클릭 이벤트 : MainActivity.kt

```
class MainActivity : AppCompatActivity() {
    private var toggleImage = false

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

    fun onButtonClicked(view: View) {
        var imageView:ImageView = findViewById(R.id.imageView)
        var imageView2:ImageView = findViewById(R.id.imageView2)

        if (toggleImage) {
            imageView.visibility = View.VISIBLE
            imageView2.visibility = View.INVISIBLE
        } else {
            imageView.visibility = View.INVISIBLE
            imageView2.visibility = View.VISIBLE
        }
        toggleImage = !toggleImage
    }
}
```

App.이 처음 실행되면 imageView가 visible이므로, 버튼 클릭했을 때 이미지를 바꾸려면 **toggleImage = false**

imageView2가 visible 하게 됨.

toggleImage 의 상태가 true 였으면 false로, False 였으면 true로 바꿈.

What to do next?

- **Layout**
  - LinearLayout
  - RelativeLayout
  - FrameLayout
  - **TableLayout**

- 실습 : UI를 구현하는 3가지 coding style

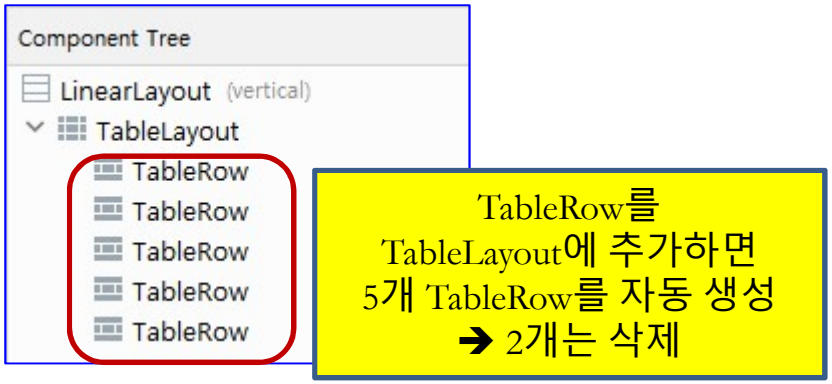
TableLayout : 가장 간단한 레이아웃

- TableLayout은 root view로 단독으로 사용하기 보다는
  - LinearLayout에 view로 포함되는 형태로 주로 사용.

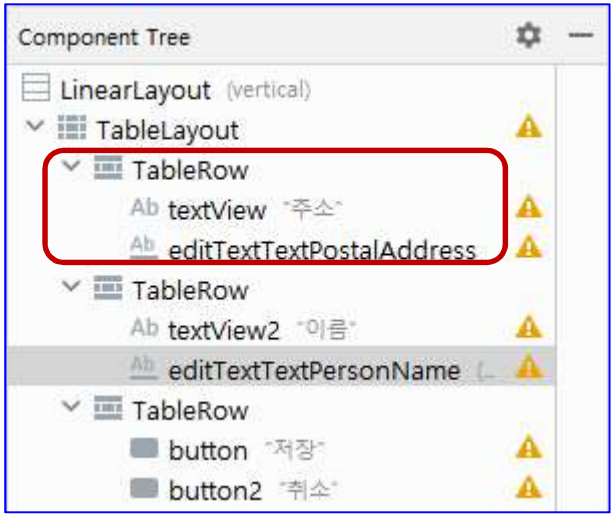
- TableLayout은 1개 이상의 **TableRow**를 포함
  - **TableRow** = 표에서 한 개의 행(row)에 해당

- 주요 속성
  - **stretchColumns, shrinkColumns**
  - **layout_column, layout_span**

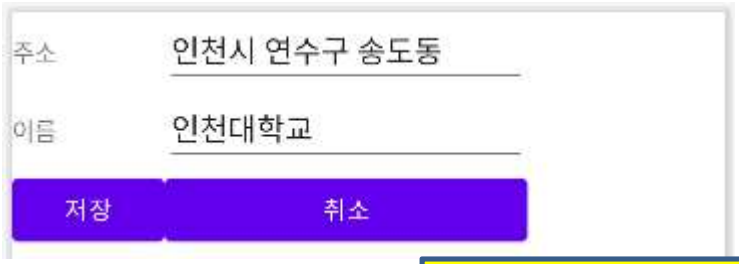
TableLayout 예



TableRow를
TableLayout에 추가하면
5개 TableRow를 자동 생성
➔ 2개는 삭제



TableRow가 3개 ➔ 3개의 행(row)



LinearLayout 과
'저장' 버튼에
layout_margin 설정

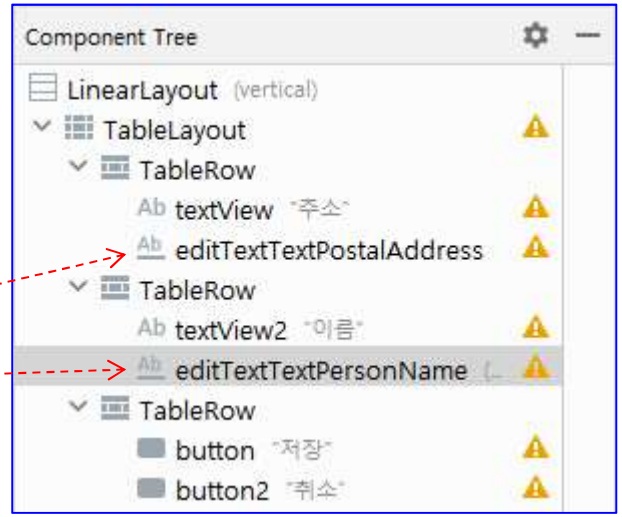
TableLayout 수정 (1/2)

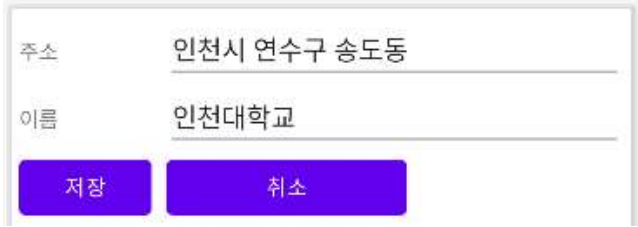
```
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1,2,3">
```

2개 컬럼에서
4개 컬럼으로 확장

```
<EditText
    android:id="@+id/editTextTextPostalAddress"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_span="3"
    android:ems="10"
    android:inputType="textPostalAddress"
    android:text="인천시 연수구 송도동" />
```

4번째 컬럼까지
확장



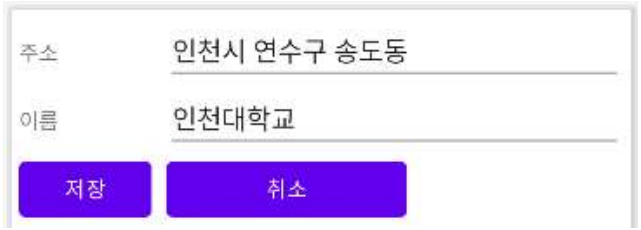


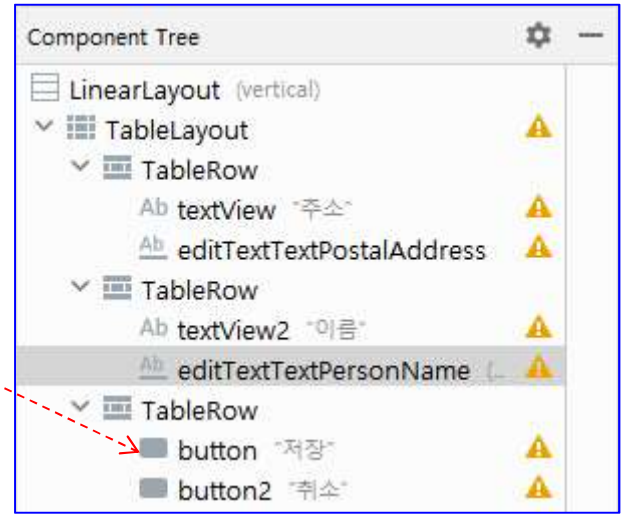
TableLayout 수정 (2/2)

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="2"
    android:layout_marginRight="10dp"
    android:text="저장" />
```

위치를
3번째 컬럼으로
지정

'**취소**' 버튼의 위치는
자연스럽게 4번째 컬럼이 됨.





What to do next?

- Layout
  - LinearLayout
  - RelativeLayout
  - FrameLayout
  - TableLayout

- **실습 : UI를 구현하는 3가지 coding style**

실습 프로젝트 생성

- 새 프로젝트 생성
  - **Project name** : **Layout Example Practice**
  - **Package name** : **edu.ourincheon.layoutexamplepractice**
  - **Activity** : **Empty Activity**
  - **Activity name** : **MainActivity.kt**
  - **Layout name** : **activity_main.xml**
- **자동 생성된 XML 파일**의 root view는 **ConstraintLayout**
  - 이를 **LinearLayout**으로 변경
    - **Component Tree 창 > ConstraintLayout** 클릭 > 오른쪽 버튼
    - **Convert View > LinearLayout > Apply**
    - **LinearLayout의 orientation 속성 : vertical**
  - **TextView** 도 삭제

UI를 구현하는 3가지 coding style

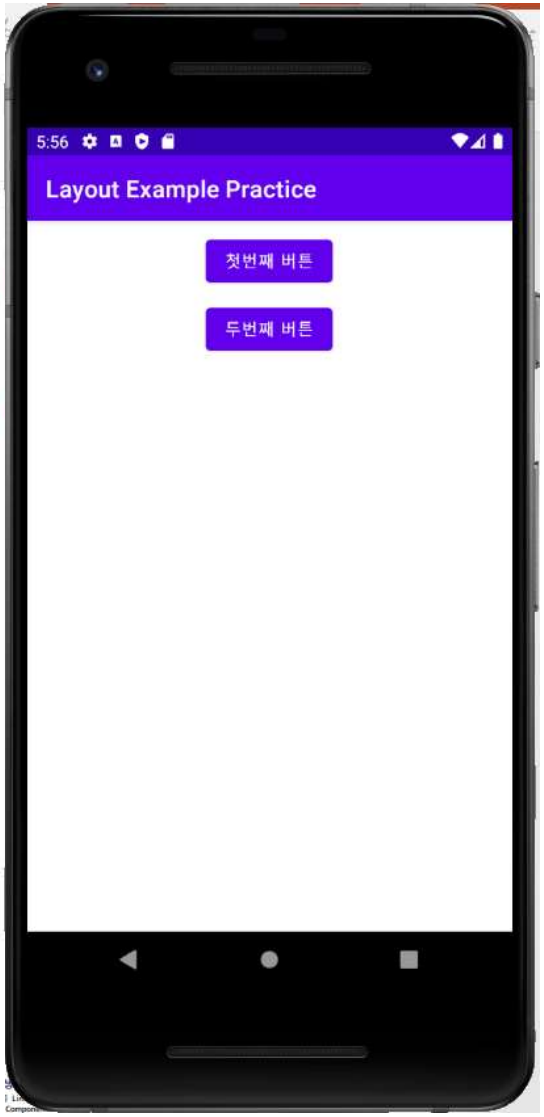
1번째 방법 : XML layout

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:text="@string/first_button" />
...
```

2번째 방법 : kotlin 코드

```
...
val b1 = Button(this)
b1.text = "첫 번째 버튼"
linearLayout.addView(b1)
...
```

3번째 방법 : XML layout + kotlin 코드

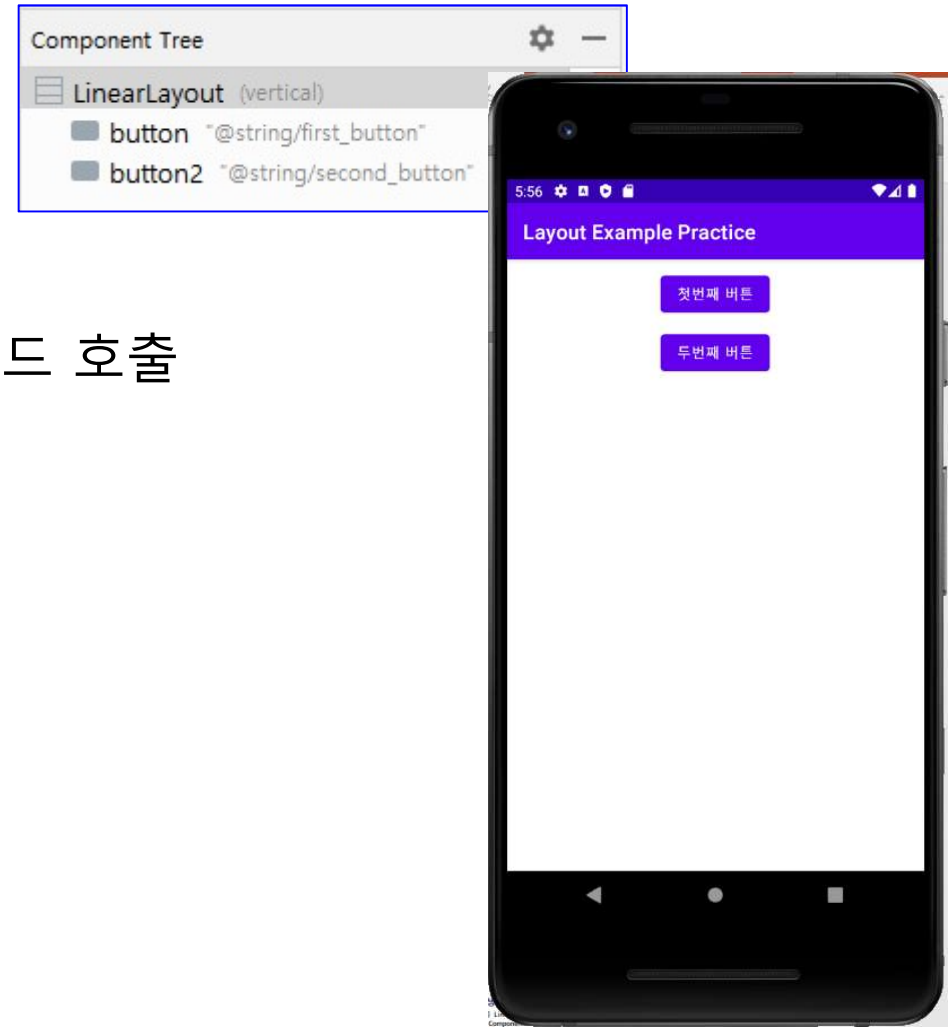


XML layout .vs. kotlin code

- **XML layout**
  - Android studio layout editor 사용
    - XML 코드 자동 생성
  - UI를 변경할 경우 XML 파일만 수정
    - Java 코드를 recompile할 필요가 없음
      - preview 기능 : UI를 수정할 때마다 즉시 확인 가능
  - **정적 UI** 구현에 유리
- **kotlin code**
  - UI를 확인하려면 ➔ gradle build > run
  - **동적 UI** 구현
    - **Activity 실행 중에 UI를 바꿀 수 있음**.

첫 번째 방법: XML Layout

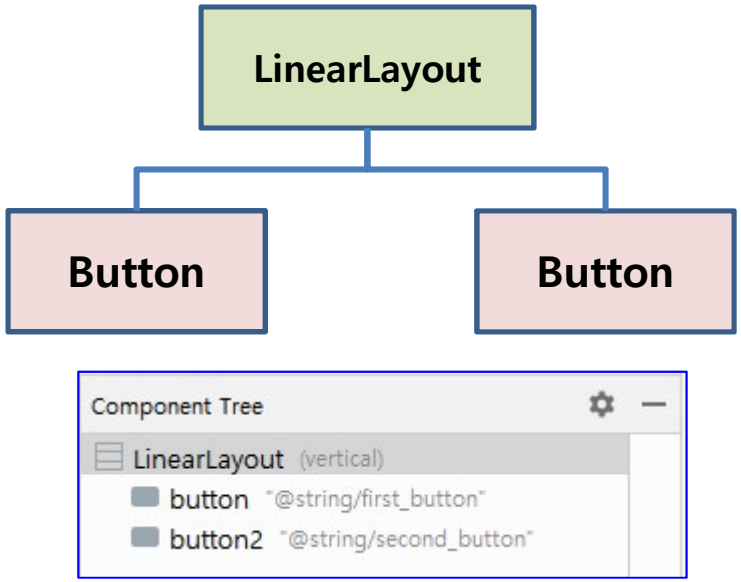
1. **XML Layout**
   - ViewGroup 생성
   - View를 ViewGroup에 추가
2. **Kotlin code**
   - Activity에 나타나도록 메소드 호출
     - **setContentView**



첫 번째 방법: XML Layout (1/2)

res/layout/activity_main.xml

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:text="@string/first_button" />

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:text="@string/second_button" />
```



res/val/strings.xml

```
<resources>
    <string name="app_name">Layout Example Practice</string>
    <string name="first_button">첫번째 버튼</string>
    <string name="second_button">두번째 버튼</string>
</resources>
```

첫 번째 방법: XML Layout (2/2)

MainActivity.kt

```
package edu.ourincheon.layoutexamplepractice

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity

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

두 번째 방법 : 코드에서 UI 구현 (1/2)

MainActivity.kt

```
const val TAG = "CODE >>> "

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

        val linearLayout = LinearLayout(this)
        linearLayout.orientation = LinearLayout.VERTICAL

        val param =
            LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.MATCH_PARENT
            )
        linearLayout.layoutParams = param

        b1 : Button ("첫번째 버튼) 추가

        b2 : Button ("두번째 버튼) 추가

        linearLayout.addView(b1)
        linearLayout.addView(b2)
        setContentView(linearLayout)
    }
}
```

XML Layout 파일을 가져와 화면에 출력하는 부분을 주석 처리

LinearLayout 객체 속성 정의

2개 버튼을 linearLayout의 Child view로 추가

Root view인 linearLayout 객체를 화면에 출력

두 번째 방법 : 코드에서 UI 구현 (2/2)

MainActivity.kt

```
val b1 = Button(this)
b1.text = resources.getString(R.string.first_button)
b1.id = generateViewId()
Log.d(TAG, "id of ${b1.text} = ${b1.id}")

val b2 = Button(this)
b2.text = resources.getString(R.string.second_button)
b2.id = generateViewId()
Log.d(TAG, "id of ${b2.text} = ${b2.id}")

val buttonParam =
    LinearLayout.LayoutParams(
        LinearLayout.LayoutParams.WRAP_CONTENT,
        LinearLayout.LayoutParams.WRAP_CONTENT
    )
buttonParam.setMargins(10, 0, 0, 10)
buttonParam.gravity = Gravity.CENTER

b1.layoutParams = buttonParam
b2.layoutParams = buttonParam
```

Button 객체에 id 할당

Button 객체에 어떤 id가 할당되었는지 LogCat 창에서 확인

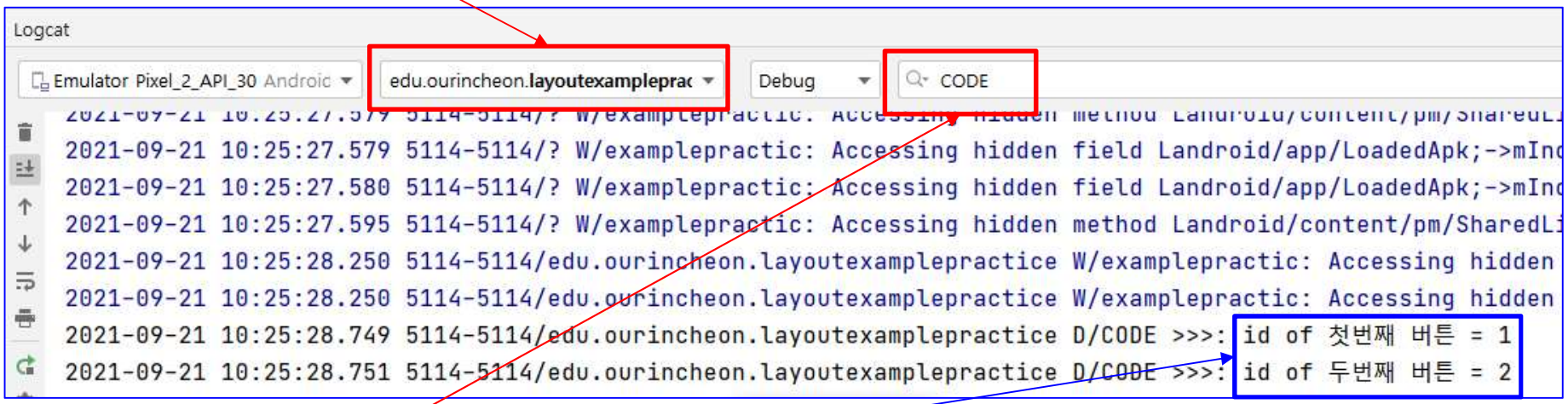
Button 객체의 공통 속성 정의

Button 객체에 속성 할당

LogCat : 출력 메시지

LogCat 창은 주로 debug를 위해 변수나 객체 또는 메시지를 출력

```
package edu.ourincheon.layoutexamplepractice
```



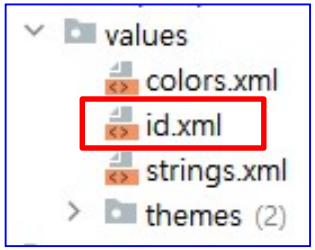

```
Log.d(TAG, "id of ${b1.text} = ${b1.id}")
```

```
const val TAG = "CODE >>> "
```

두 번째 방법 – id 리소스 참조

앞의 방법에서 id 속성을 자동으로 생성하는 코드가 복잡
➔ id 속성을 리소스 폴더에 생성하면 코드 복잡도를 **조금** 줄일 수 있음.

app > res 폴더 ➔ 마우스 오른쪽 버튼
➔ New > Values Resource file ➔ "id.xml" 입력 ➔ Enter



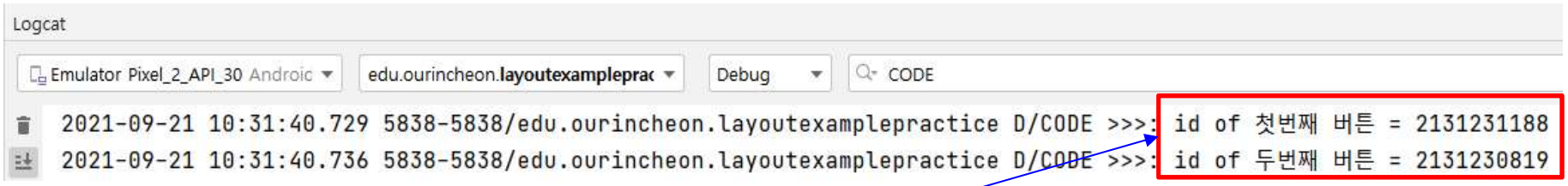

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="button1" type="id" />
    <item name="button2" type="id" />
</resources>
```

```
val b1 = Button(this)
b1.text = resources.getString(R.string.first_button)  // "첫 번째 버튼"
b1.id = R.id.button1
Log.d("CODE >>>", "id of ${b1.text} = ${b1.id}")

val b2 = Button(this)
b2.text = resources.getString(R.string.second_button) // "두 번째 버튼"
b2.id = R.id.button2
Log.d("CODE >>>", "id of ${b2.text} = ${b2.id}")
```

리소스 참조로
코드가 바뀐 부분

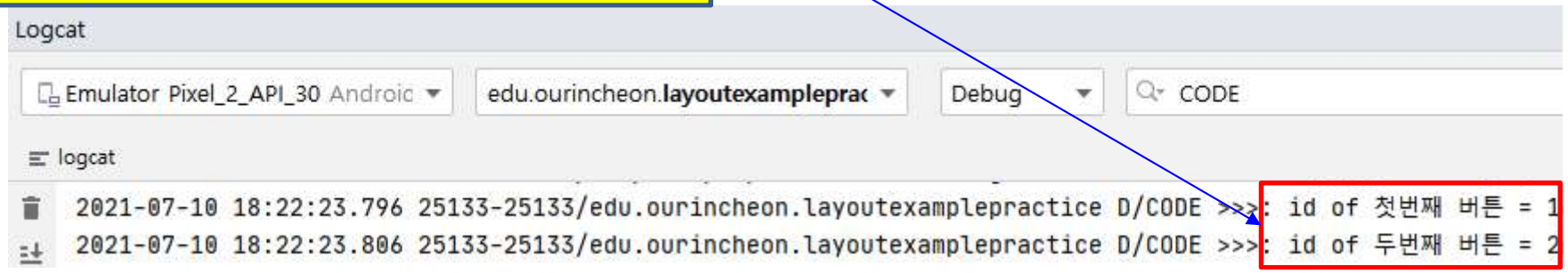
LogCat : 출력 메시지

리소스 참조로 id 할당




```
Log.d("CODE >>>", "id of ${b1.text} = ${b1.id}")
```

자동 생성 함수로 id 할당



세 번째 방법 : XML layout +코드 = UI

MainActivity.kt

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

        var bt1:Button = findViewById(R.id.button)
        var bt2:Button = findViewById(R.id.button2)
        bt1.text = "코드에서도 변경 가능"
        bt2.isEnabled = false
    }
}
```

앞(슬라이드 48)에서 만든 id.xml은 삭제하고 실행 ➔ id가 겹침

button, button2는 XML Layout 에서 정의한 해당 view의 id