안드로이드 앱 프로그래밍

Chapter 03

레이아웃 익히기

이번 장에서는 무엇을 다룰까요?

화면을 잘 만들어보고 싶어요.

- 레이아웃에는 어떤 것들이 있는지 알아봐요.
- 대표적인 레이아웃들을 사용해봐요.

이번 장에서는 무엇을 다룰까요?

레아아웃에는 어떤 것들이 있는지 알아볼까요?

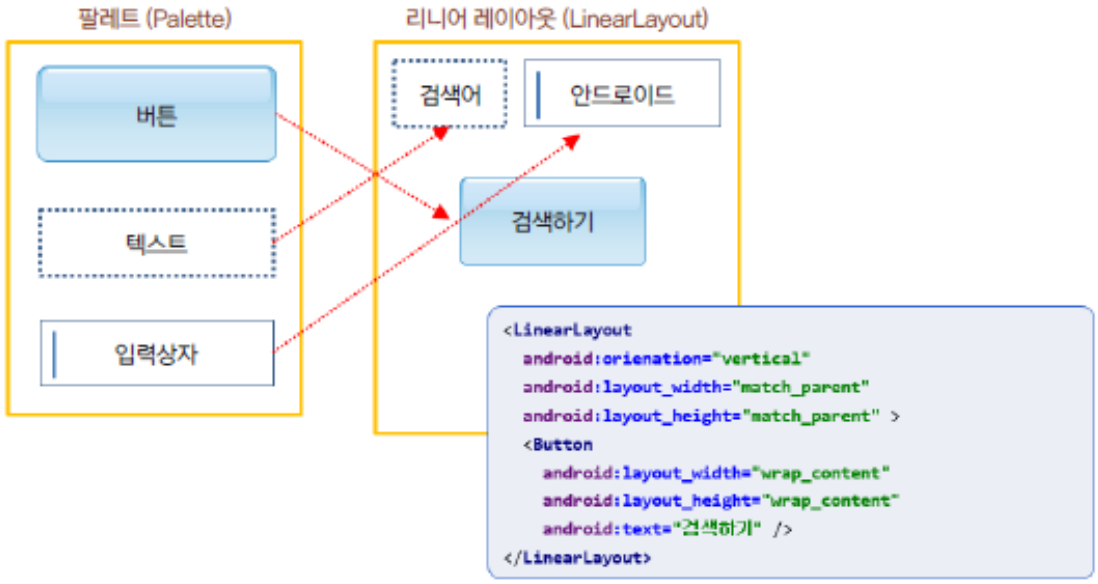
- 대표적인 레이아웃 살펴보기

대표적인 레이아웃을 사용해 볼까요?

- 리니어 레이아웃 사용하기
- 상대 레이아웃 사용하기
- 테이블 레이아웃 사용하기
- 프레임 레이아웃 사용하기

스크롤을 만들어 볼까요?

- 스크롤 사용하기



강의 주제 및 목차

강의 주제

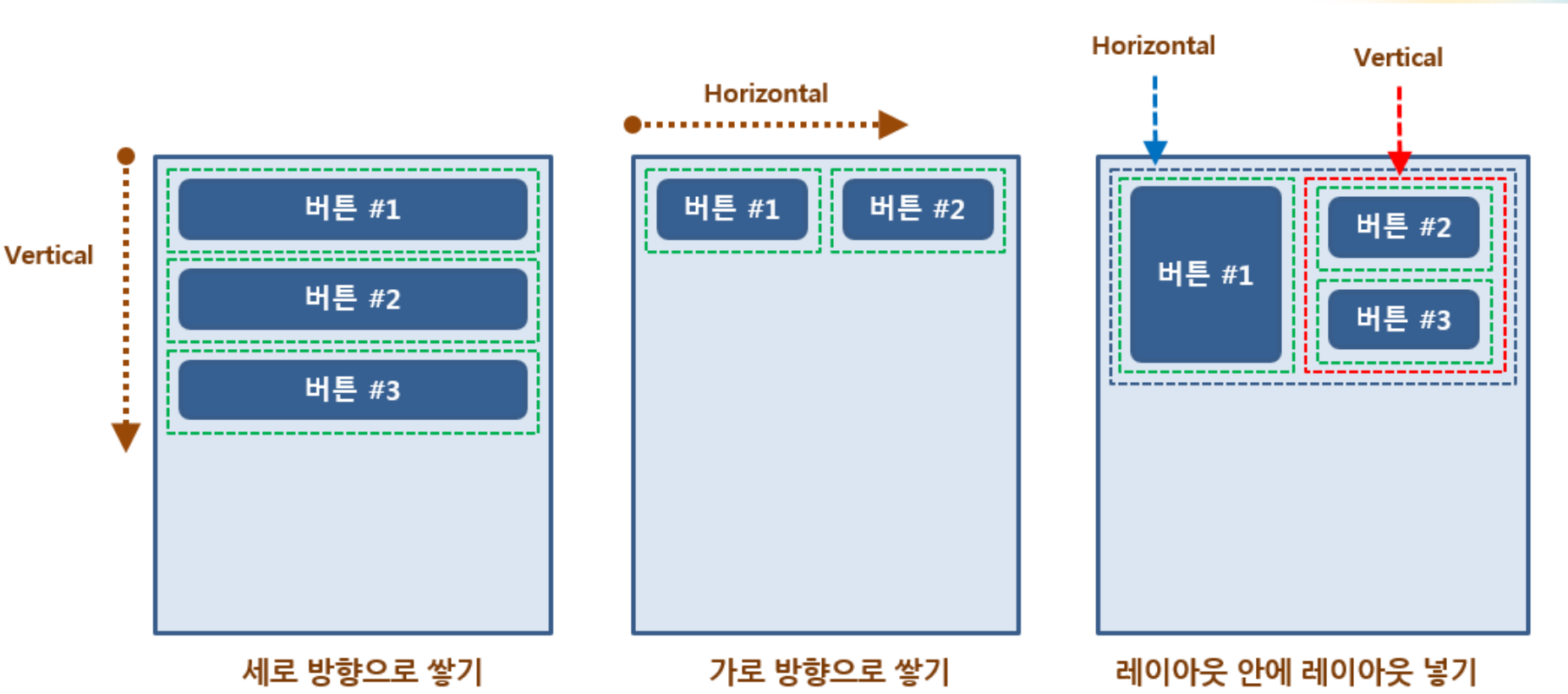
대표적인 레이아웃 이해하기

1. 대표적인 레이아웃 살펴보기

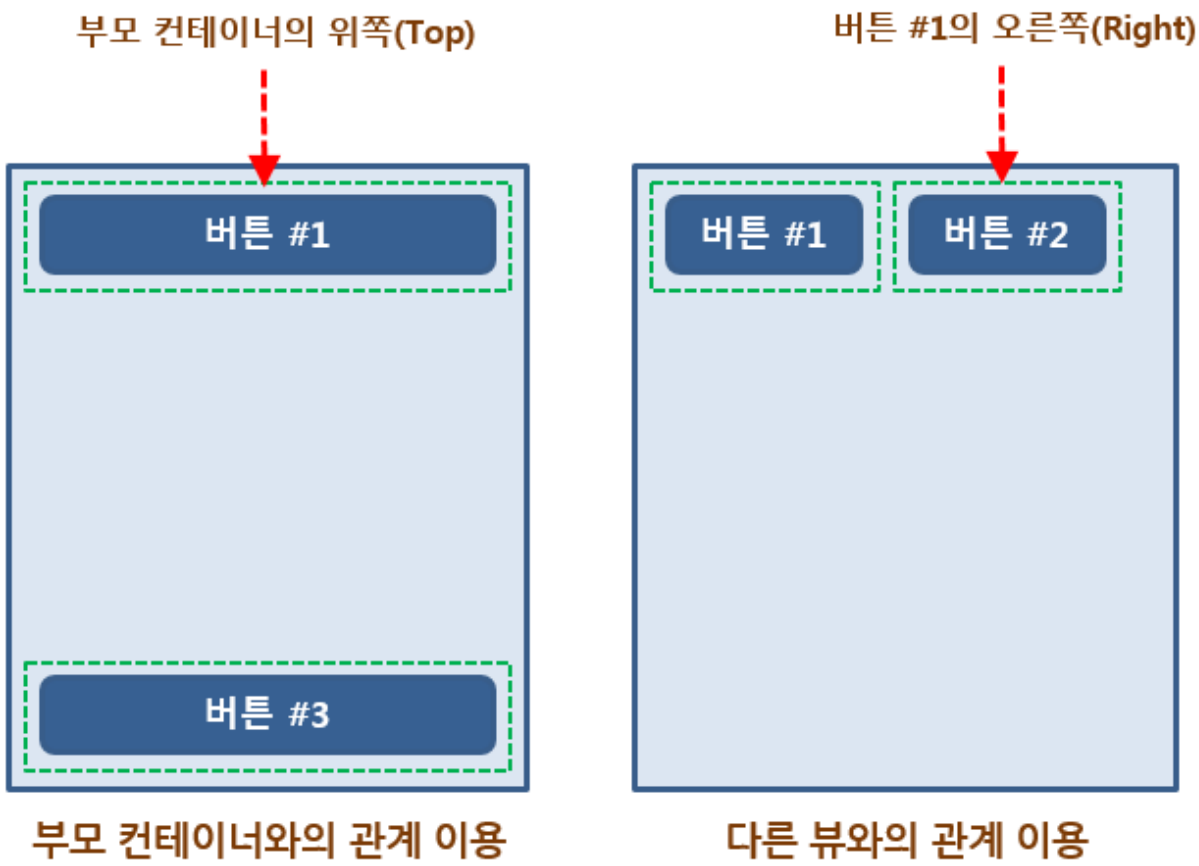
대표적인 레이아웃

레이아웃 이름	설 명
제약 레이아웃 (ConstraintLayout)	제약 조건(Constraint) 기반 모델 제약 조건을 사용해 화면을 구성하는 방법 안드로이드 스튜디오에서 자동으로 설정하는 디폴트 레이아웃
리니어 레이아웃 (LinearLayout)	박스(Box) 모델 한 쪽 방향으로 차례대로 뷰를 추가하며 화면을 구성하는 방법 뷰가 차지할 수 있는 사각형 영역을 할당
상대 레이아웃 (RelativeLayout)	규칙(Rule) 기반 모델 부모 컨테이너나 다른 뷰와의 상대적 위치로 화면을 구성하는 방법
프레임 레이아웃 (FrameLayout)	싱글(Single) 모델 가장 상위에 있는 하나의 뷰 또는 뷰그룹만 보여주는 방법 여러 개의 뷰가 들어가면 중첩하여 쌓게 됨. 가장 단순하지만 여러 개의 뷰를 중첩한 후 각 뷰를 전환하여 보여주는 방식으로 자주 사용함
테이블 레이아웃 (TableLayout)	격자(Grid) 모델 격자 모양의 배열을 사용하여 화면을 구성하는 방법 HTML에서 많이 사용하는 정렬 방식과 유사하지만 많이 사용하지는 않음

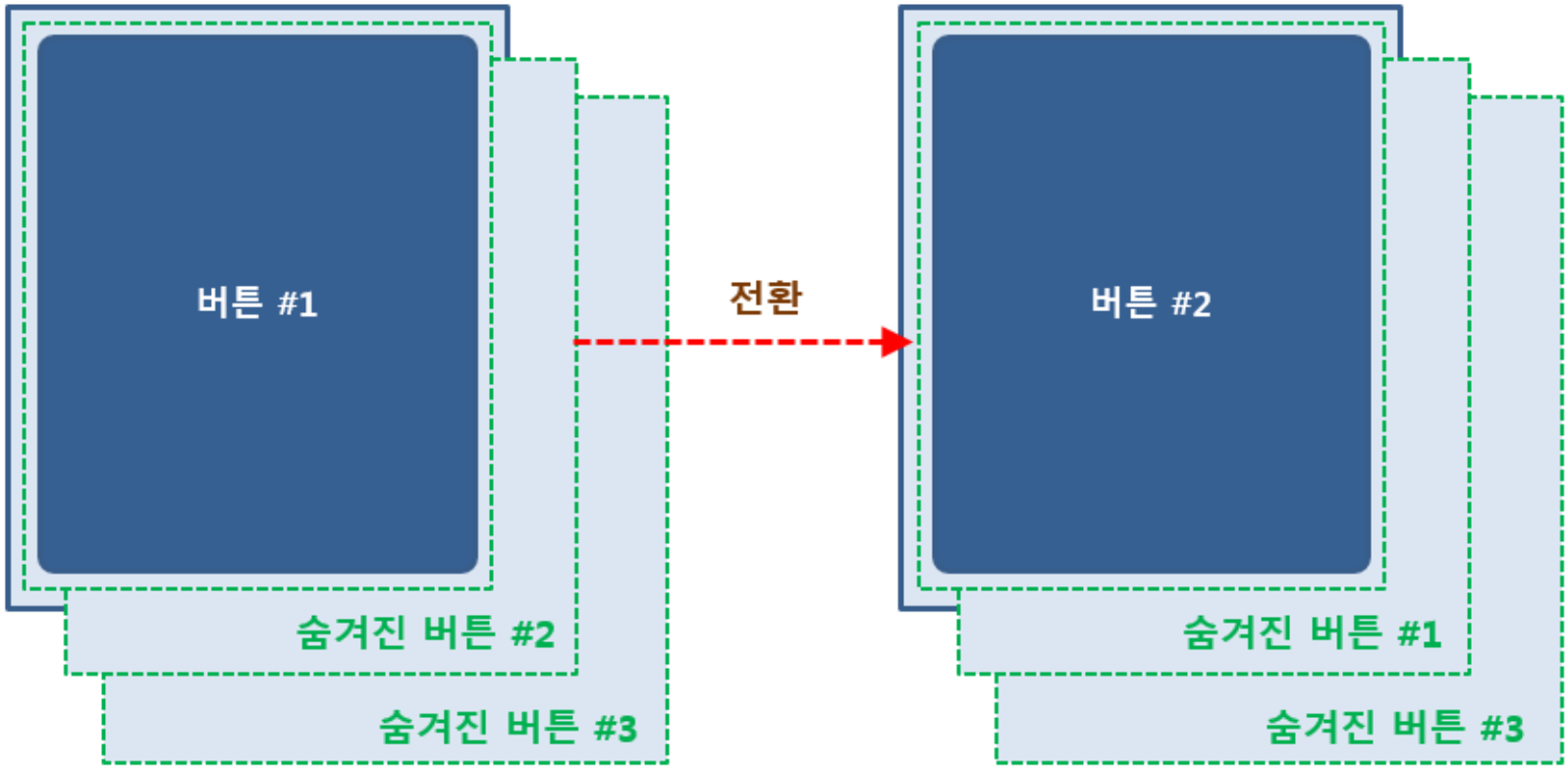
리니어 레이아웃 사용방식



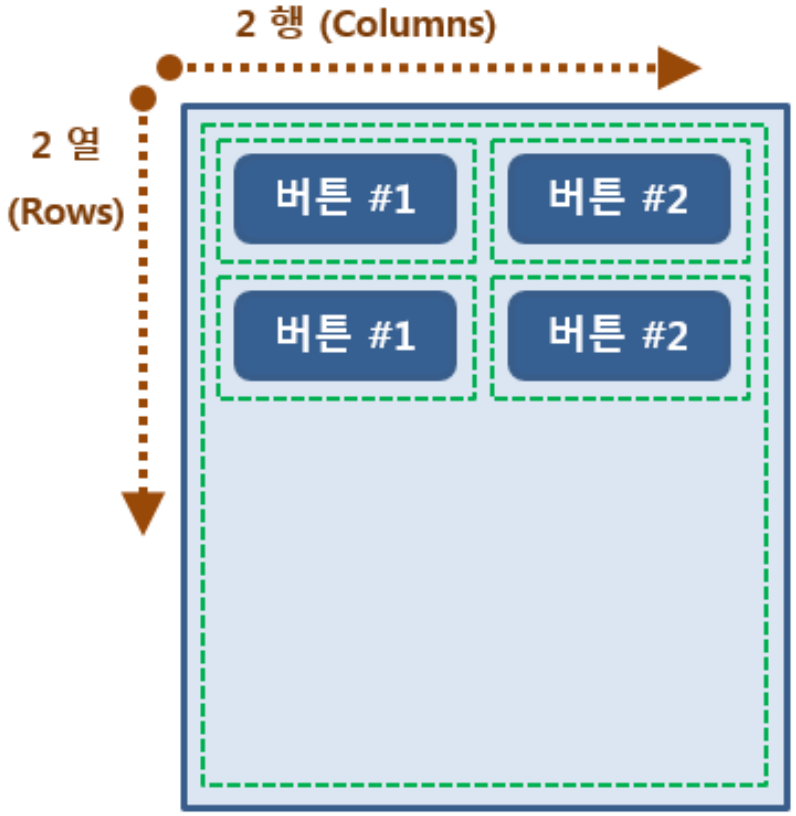
상대 레이아웃 사용방식


부모 컨테이너의 위쪽(Top)
버튼 #1
버튼 #3
부모 컨테이너와의 관계 이용
버튼 #1의 오른쪽(Right)
버튼 #1
버튼 #2
다른 뷰와의 관계 이용

프레임 레이아웃 사용방식



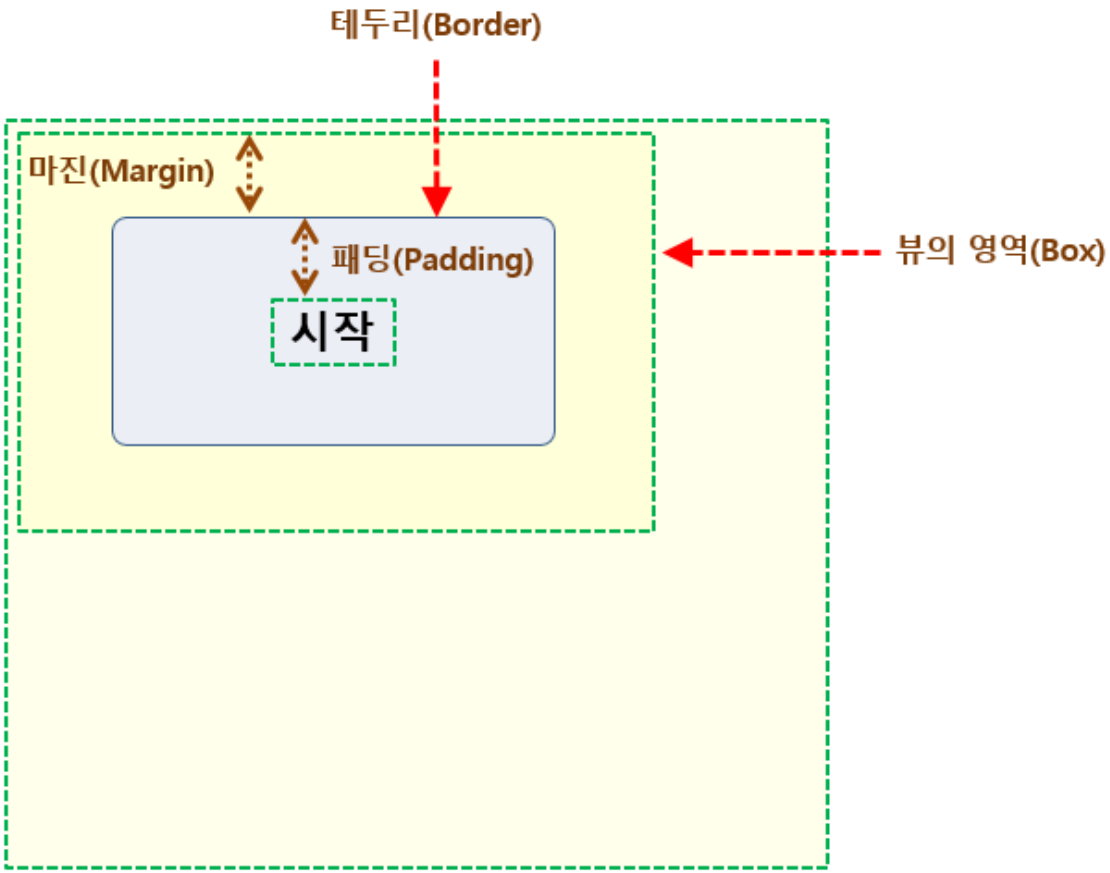
테이블 레이아웃 사용방식



격자 형태로 쌓기

뷰의 영역

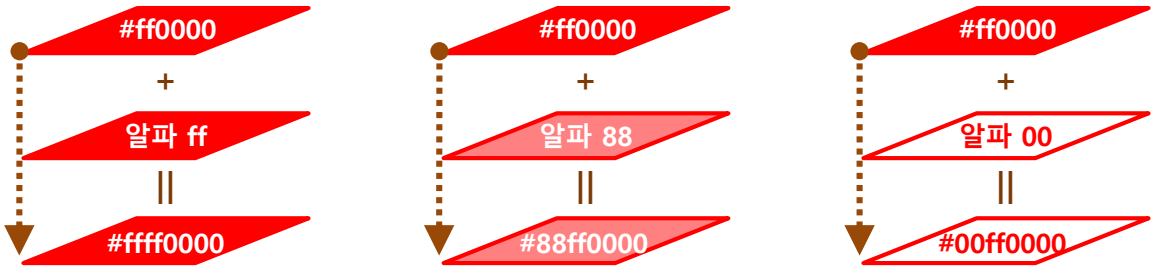
- 테두리를 기준으로 바깥쪽이 마진, 안쪽이 패딩임
- 뷰의 영역은 마진까지를 포함함



뷰의 영역 구분

뷰의 배경색

- background 속성으로 배경색 설정
- 배경 이미지를 설정할 수도 있음
- 배경색은 ARGB를 기준으로 16진수 두 자리씩 할당하며 # 뒤에 코드를 붙임

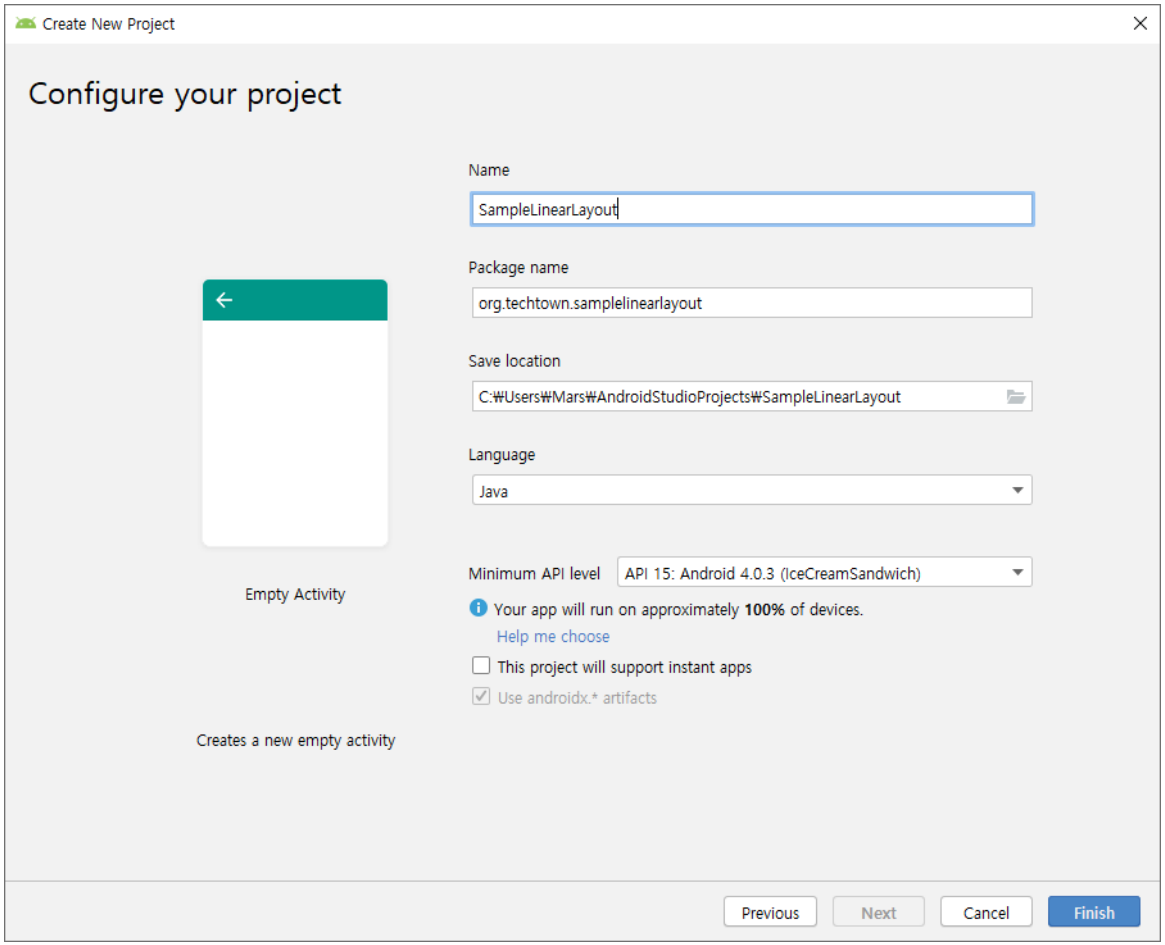



```
android:background="@drawable/house"
```

2. 리니어 레이아웃 사용하기

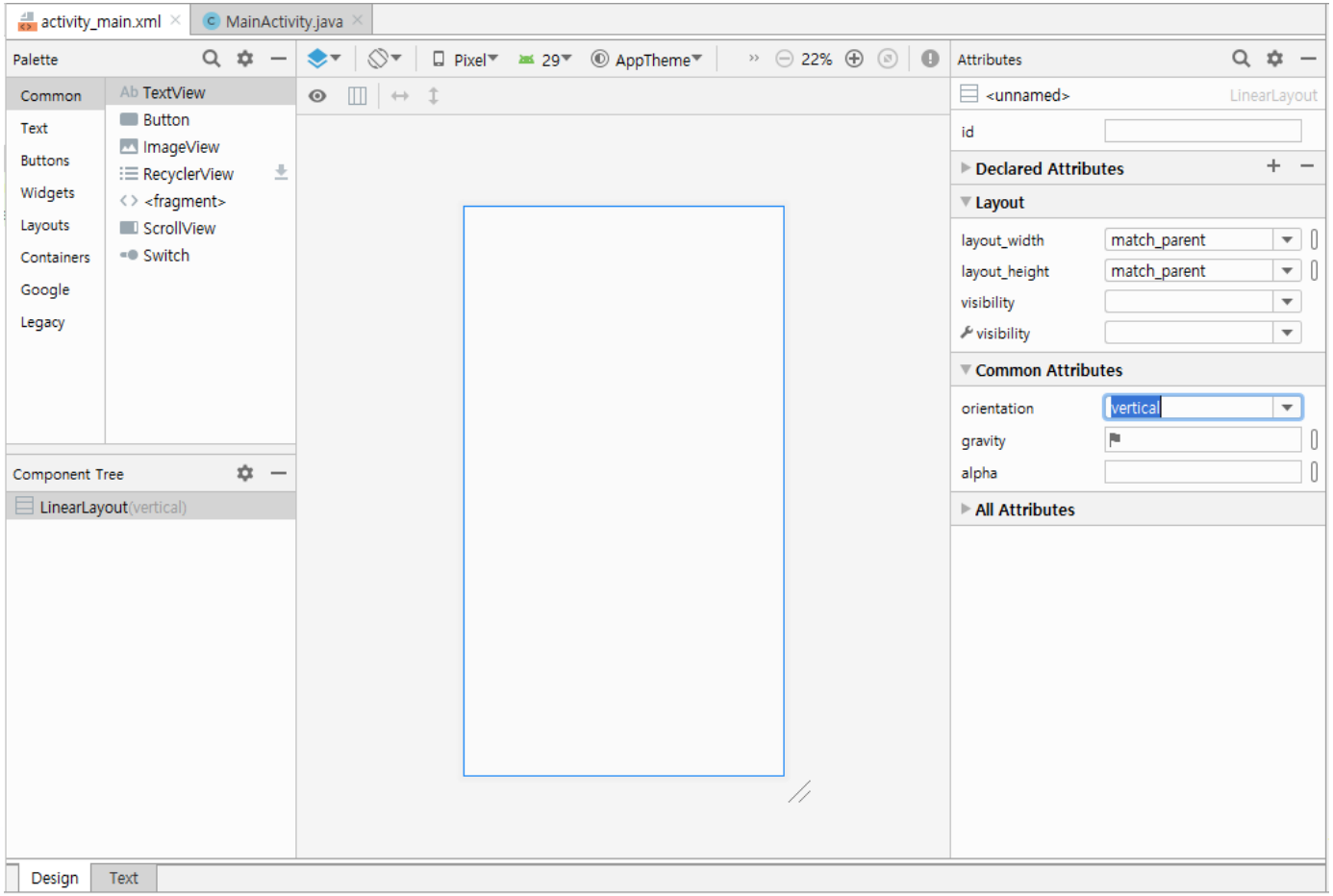
새로운 프로젝트 생성

- SampleLinearLayout 이라는 이름으로 새로운 프로젝트 생성

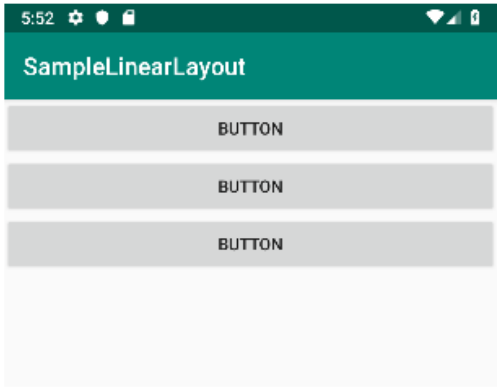


리니어 레이아웃 – 방향 설정하기

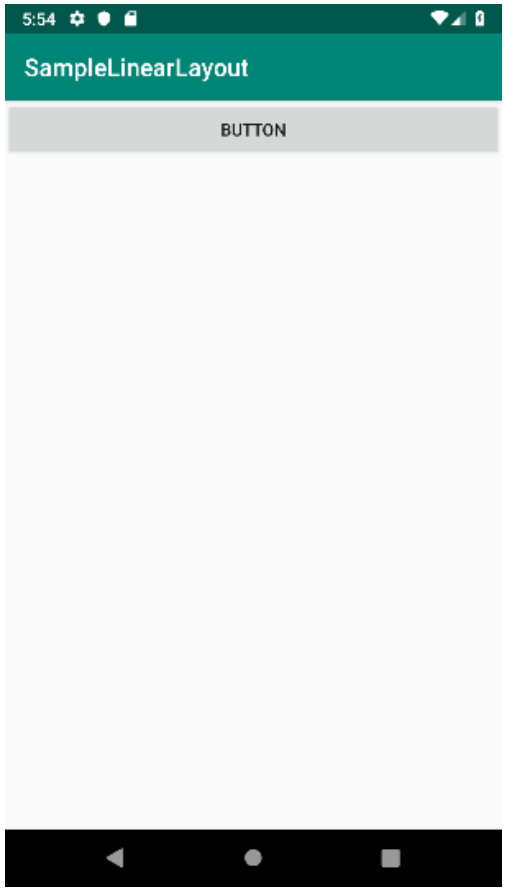
- 방향 속성은 리니어 레이아웃의 필수 속성임
- 가로(horizontal), 세로(vertical)



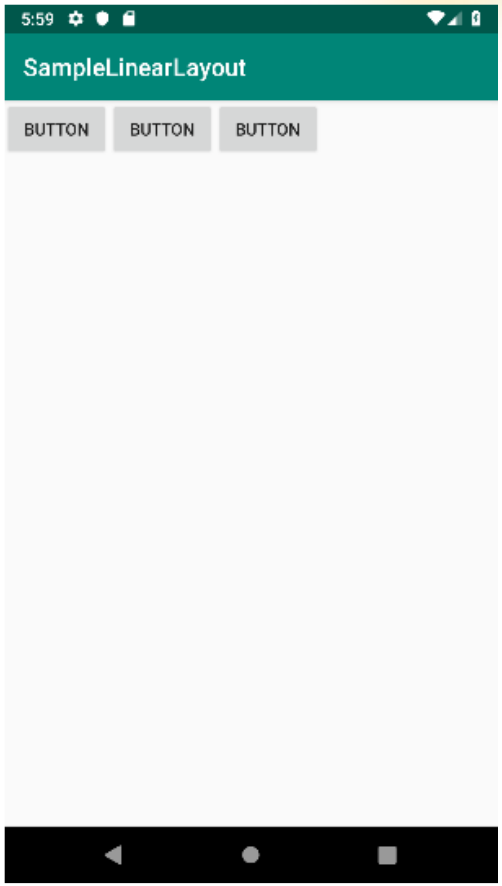
리니어 레이아웃 – 방향 설정하기 (계속)



[세로 방향으로 설정한 경우]



[가로 방향으로 설정을 바꾼 경우]



[버튼의 layout_width 속성을 wrap_content로 바꾼 경우]

리니어 레이아웃 – 자바 코드에서 구성하기

- 자바 코드에서 직접 레이아웃 객체를 만들고 파라미터 설정하는 방법

```
LinearLayout mainLayout = new LinearLayout(this);
mainLayout.setOrientation(LinearLayout.VERTICAL);

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
        LinearLayout.LayoutParams.MATCH_PARENT,
        LinearLayout.LayoutParams.WRAP_CONTENT);

Button button1 = new Button(this);
button1.setText("Button 01");
button1.setLayoutParams(params);
mainLayout.addView(button1);

setContentView(mainLayout);
```

1 레이아웃 객체 생성

2 파라미터 설정

3 버튼 객체 생성하여 추가

4 화면 설정

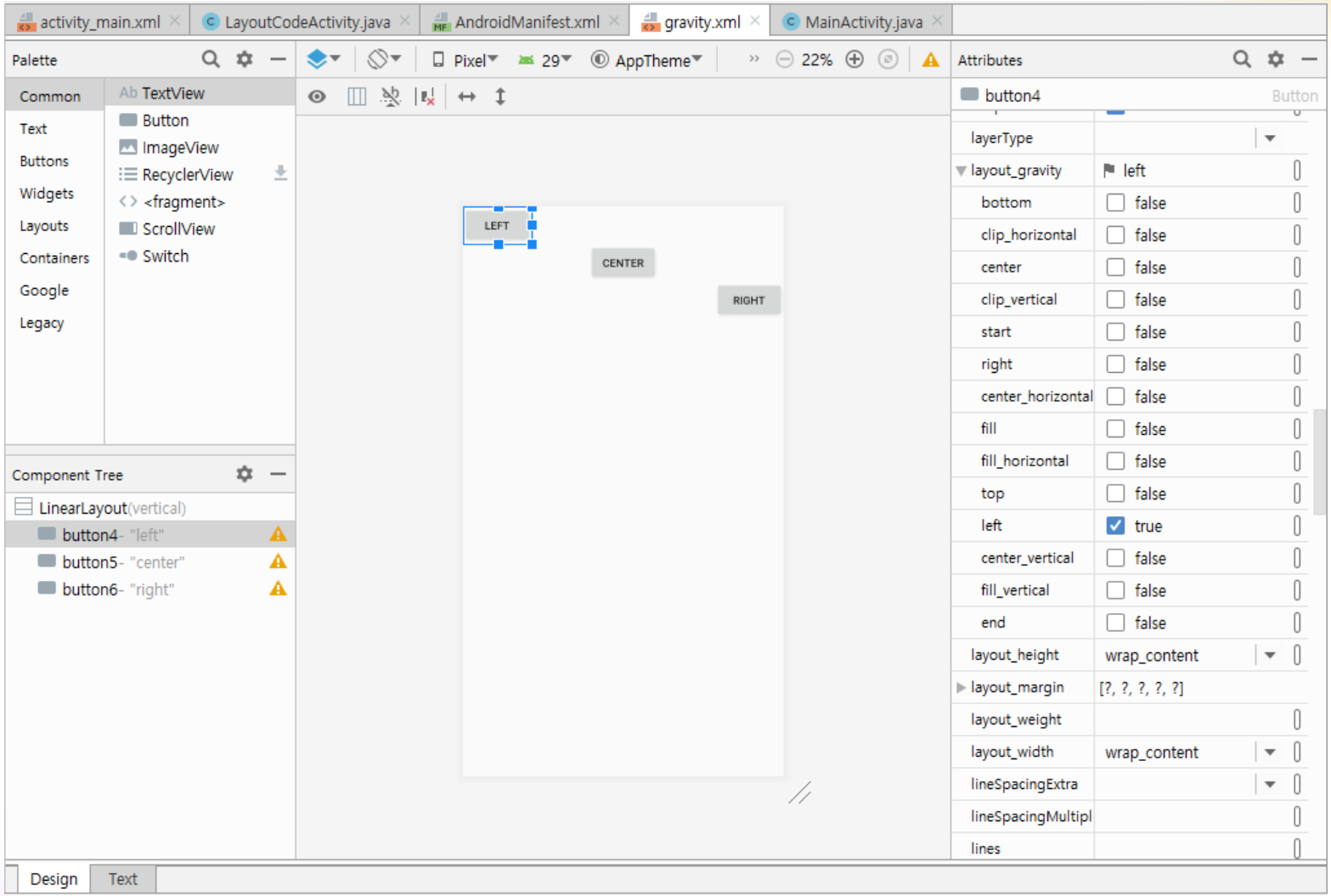
리니어 레이아웃 – 뷰 정렬하기

[두 가지 정렬 속성]

정렬 속성	설 명
layout_gravity	- [외부] 부모 컨테이너의 여유 공간에 뷰가 모두 채워지지 않아 여유 공간 안에서 뷰를 정렬할 때
gravity	- [내부] 뷰에서 화면에 표시하는 내용물을 정렬할 때 (텍스트뷰의 경우, 내용물은 글자가 되고 이미지뷰의 경우 내용물은 이미지가 됨)

- **layout_gravity**
  - 뷰의 layout_width나 layout_height 속성이 match_parent가 아닐 경우에 같이 사용할 수 있음

리니어 레이아웃 – 뷰 정렬하기 (계속)



리니어 레이아웃 – 정렬을 위해 사용할 수 있는 값

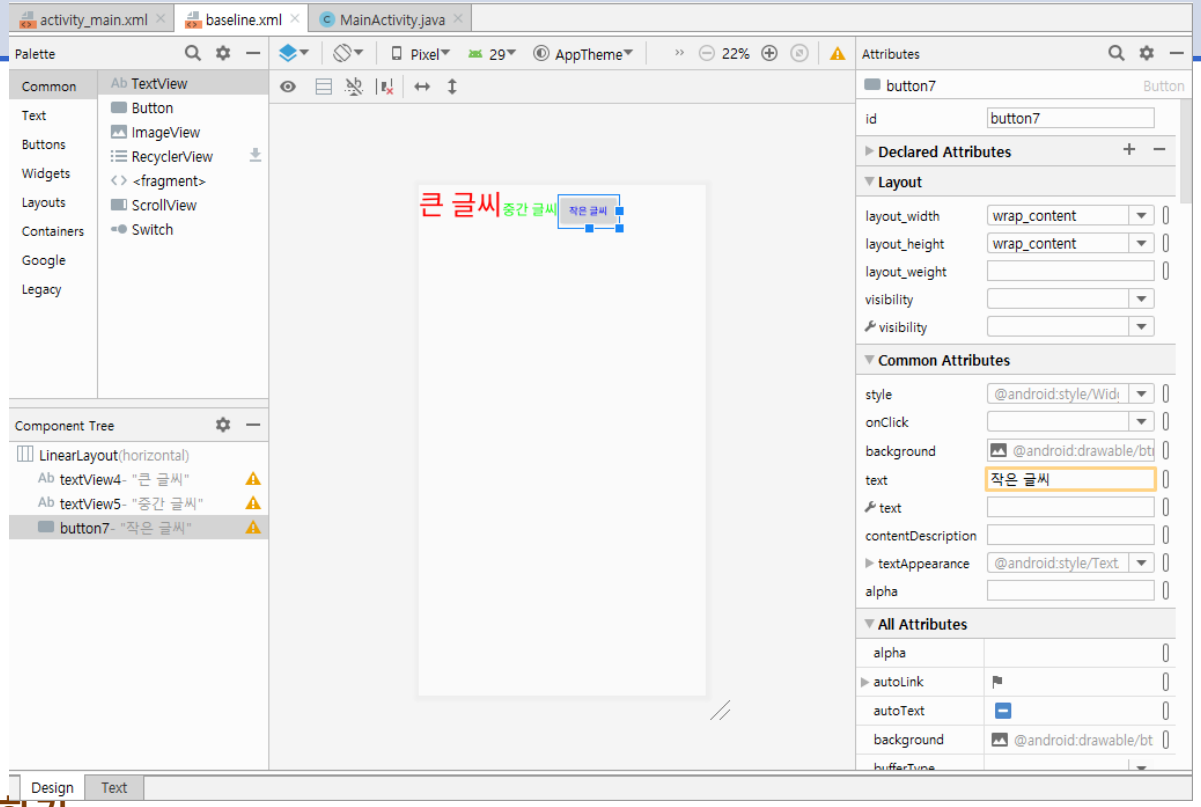
[정렬을 위해 gravity 속성에 지정할 수 있도록 정의된 값]

정렬 속성값	설 명
top	- 대상 객체를 위쪽 끝에 배치하기
bottom	- 대상 객체를 아래쪽 끝에 배치하기
left	- 대상 객체를 왼쪽 끝에 배치하기
right	- 대상 객체를 오른쪽 끝에 배치하기
center_vertical	- 대상 객체를 수직 방향의 중앙에 배치하기
center_horizontal	- 대상 객체를 수평 방향의 중앙에 배치하기
fill_vertical	- 대상 객체를 수직 방향으로 여유 공간만큼 확대하여 채우기
fill_horizontal	- 대상 객체를 수평 방향으로 여유 공간만큼 확대하여 채우기
center	- 대상 객체를 수직 방향과 수평 방향의 중앙에 배치하기
fill	- 대상 객체를 수직 방향과 수평 방향으로 여유 공간만큼 확대하여 채우기
clip_vertical	- 대상 객체의 상하 길이가 여유 공간보다 클 경우에 남는 부분을 잘라내기 - top|clip_vertical 로 설정한 경우 아래쪽에 남는 부분 잘라내기 - bottom|clip_vertical 로 설정한 경우 위쪽에 남는 부분 잘라내기 - center_vertical|clip_vertical 로 설정한 경우 위쪽과 아래쪽에 남는 부분 잘라내기
clip_horizontal	- 대상 객체의 좌우 길이가 여유 공간보다 클 경우에 남는 부분을 잘라내기 - right|clip_horizontal 로 설정한 경우 왼쪽에 남는 부분 잘라내기 - left|clip_horizontal 로 설정한 경우 오른쪽에 남는 부분 잘라내기 - center_horizontal|clip_horizontal 로 설정한 경우 왼쪽과 오른쪽에 남는 부분 잘라내기

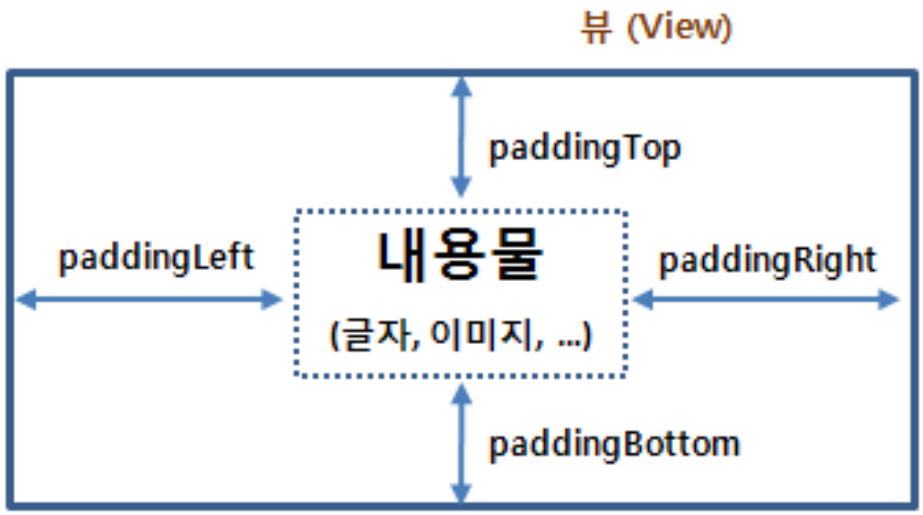
리니어 레이아웃 – 글자 아랫줄 정렬

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="horizontal"
  android:layout_width=" match_parent"
  android:layout_height="match_parent"
  android:baselineAligned="true"
  >
```

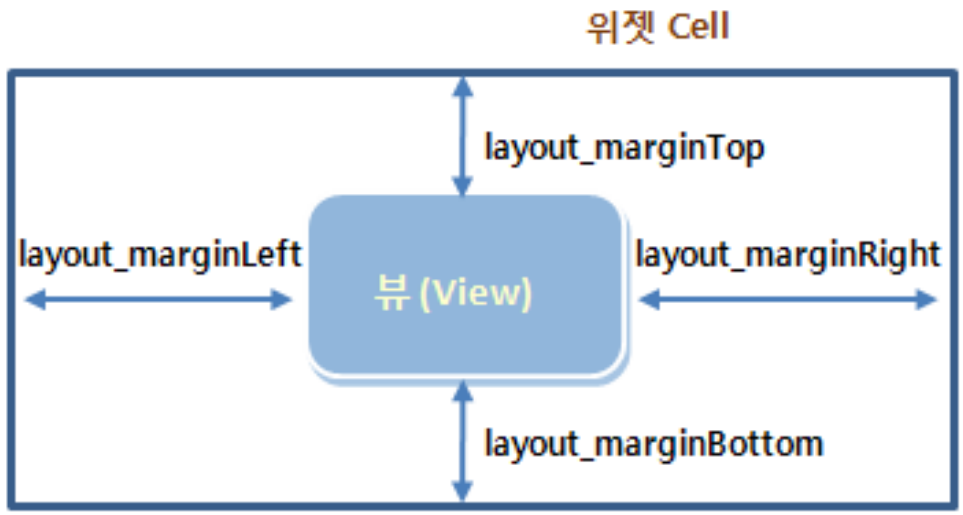
1 글자의 아랫줄 맞추기



리니어 레이아웃 – 마진과 패딩 설정하기



[padding을 이용한 뷰 내부의 여백 주기]



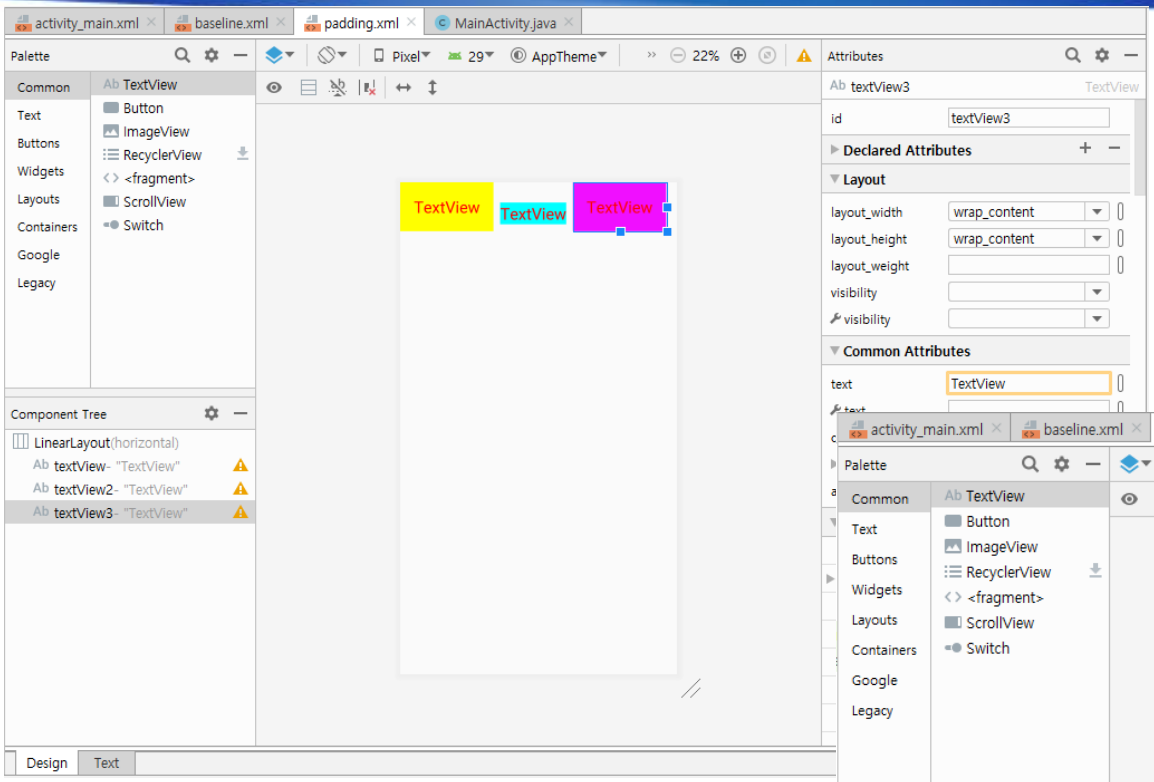
[layout_margin을 이용한 부모 여유공간과의 여백 주기]

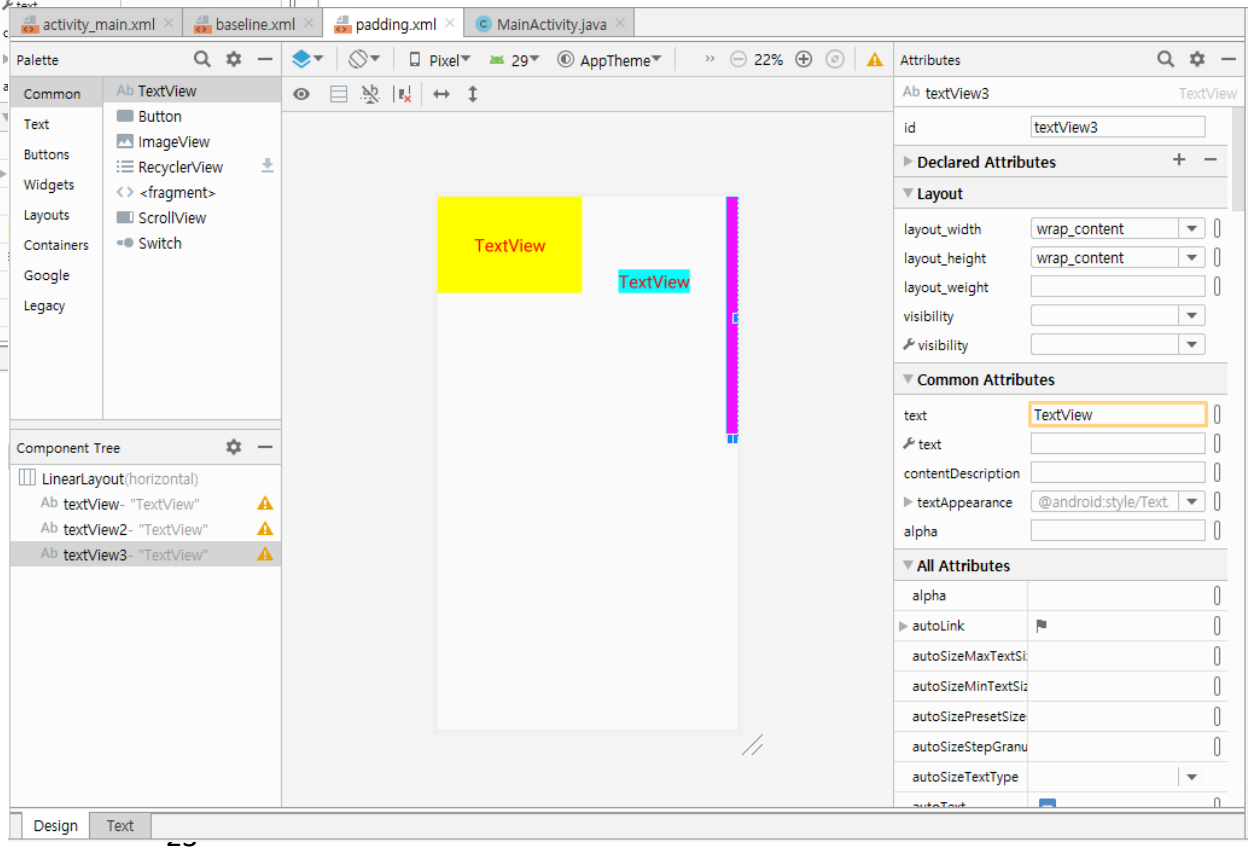
- **padding 속성**
  - 뷰 안의 내용물인 텍스트나 이미지와 뷰 안의 영역 사이의 여백을 줄 수 있는 방법

- **layout_margin 속성**
  - 부모 컨테이너의 여유 공간과 뷰 사이의 여백을 줄 수 있는 방법

- **위젯 셀**
  - 위젯이나 뷰들은 부모 컨테이너로부터 할당된 공간을 차지하게 되며 이를 '위젯 셀(cell)'이라고 부름

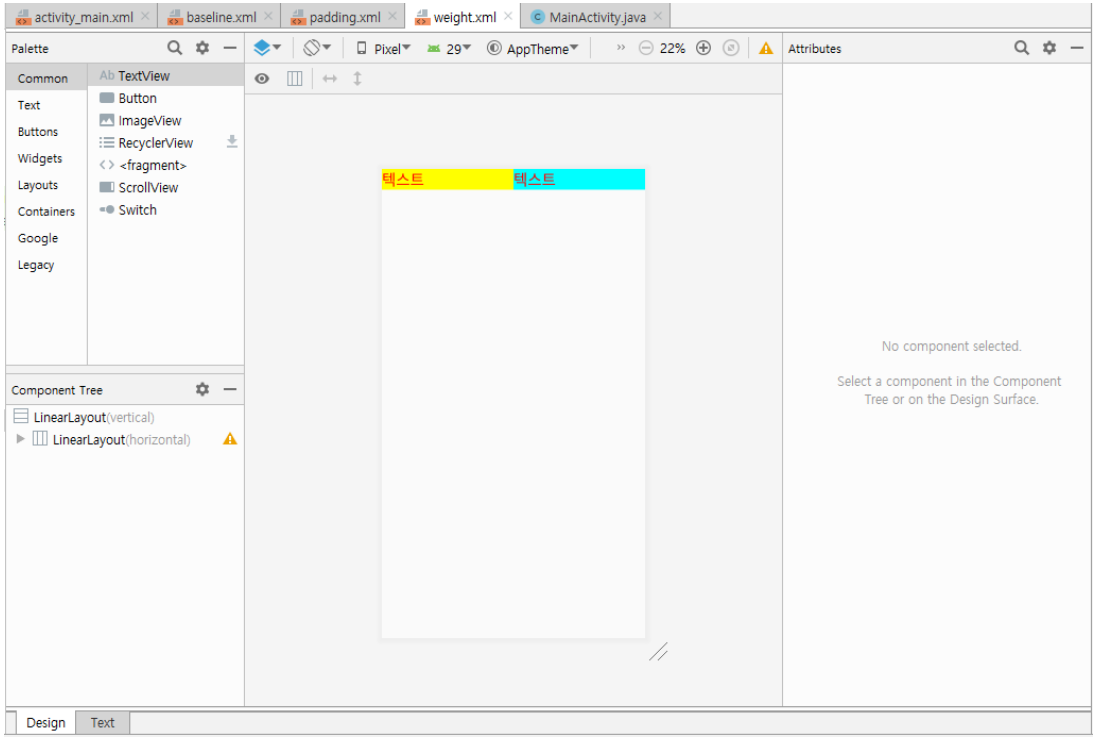
리니어 레이아웃 – 마진과 패딩 설정하기 (계속)





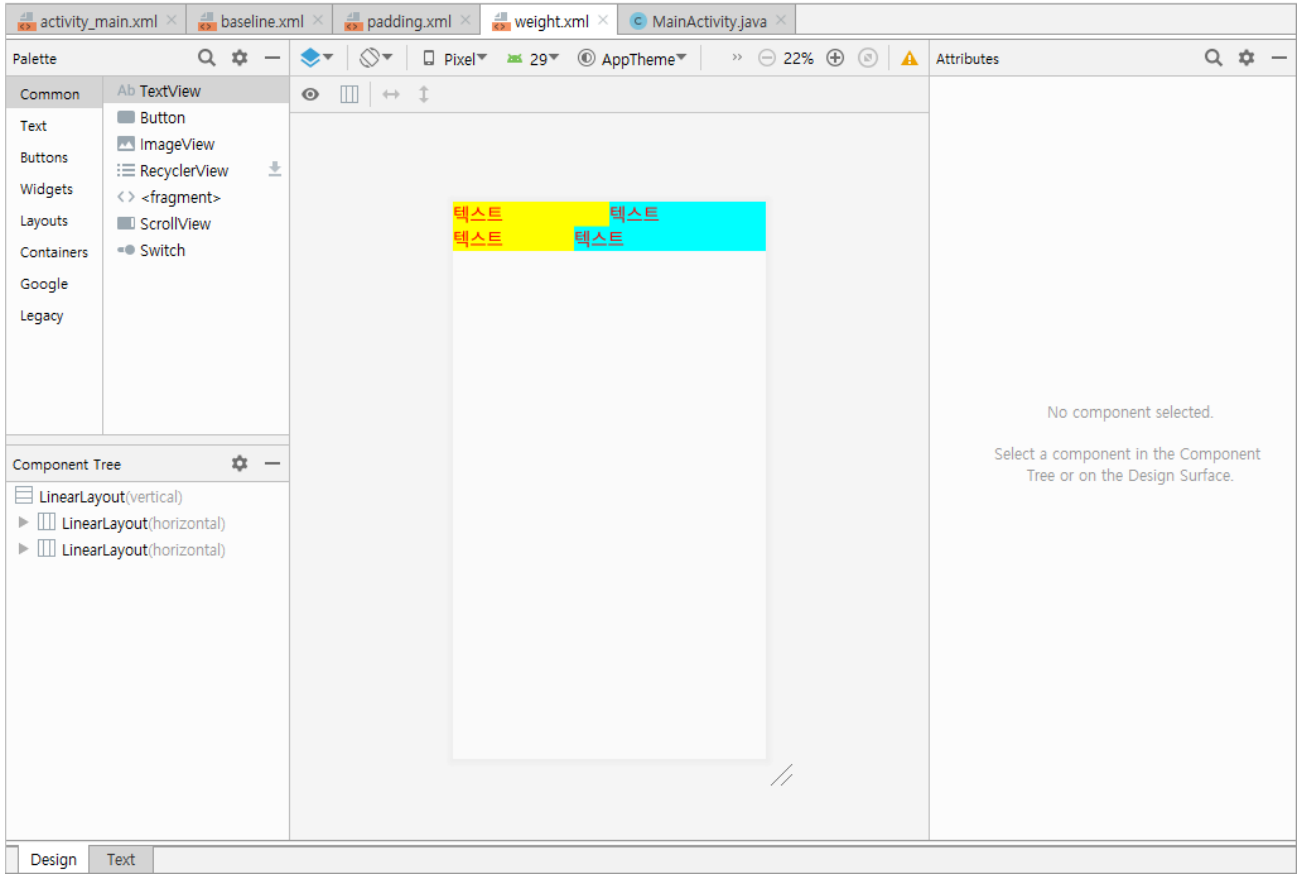
리니어 레이아웃 – 여유공간 분할하기

- layout_weight 속성은 같은 남아있는 여유공간을 얼마나 차지할 수 있는지를 비율로 지정하는 것
- android:layout_weight 속성 사용
- 두 개의 뷰에 모두 1 값을 설정한 경우



리니어 레이아웃 – 여유공간 분할하기

- 분할하려는 방향의 크기를 0dp로 하는 경우 공간을 분할하는 효과가 생김
- layout_width 속성값을 0dp로 설정한 후 하나는 1, 다른 하나는 2로 설정



리니어 레이아웃 – 레이아웃 안에 레이아웃 넣기

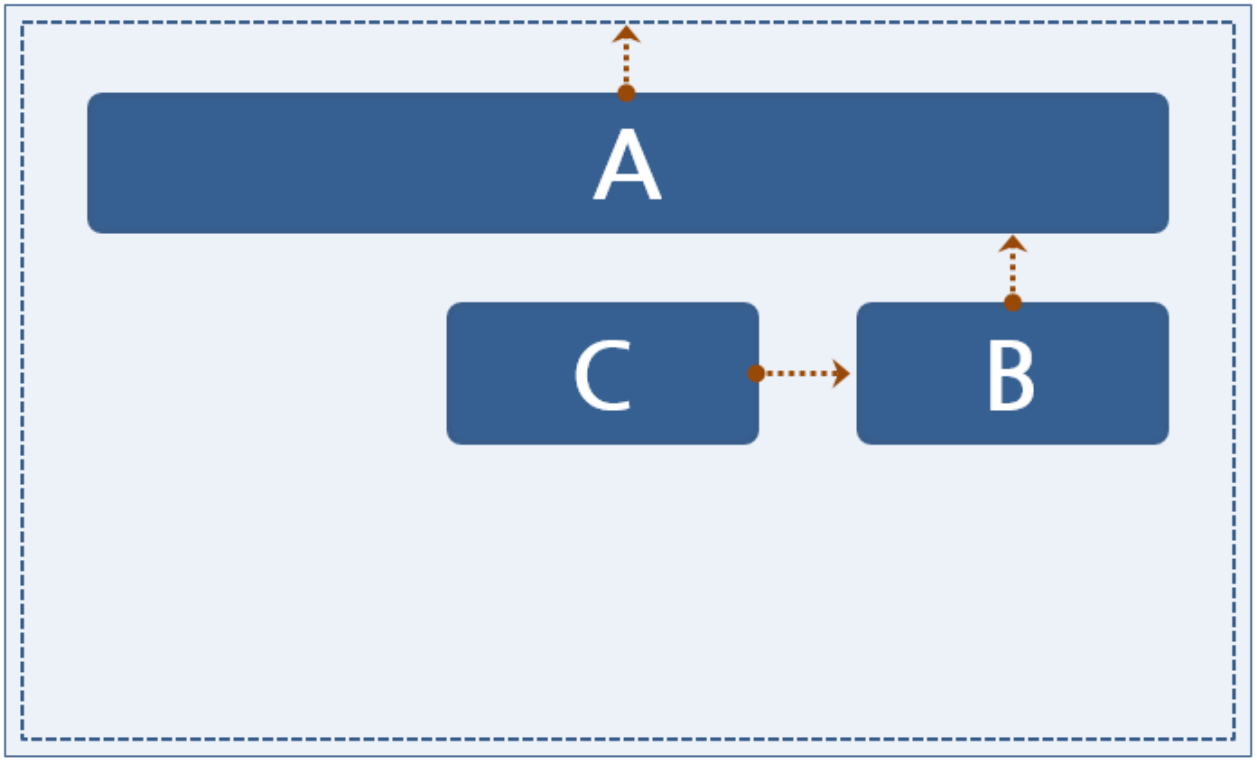
- 세로 방향의 리니어 레이아웃 안에 가로 방향의 리니어 레이아웃을 넣을 수 있음
- 레이아웃 안에 레이아웃을 계속 넣을 수 있으나 레벨이 너무 깊으면 시스템 부하 증가

```
<LinearLayout android:orientation="horizontal" >
    <ImageView />
    <LinearLayout android:orientation="vertical" >
        <TextView />
        <TextView />
    </LinearLayout>
</LinearLayout>
```

3. 상대 레이아웃 사용하기

상대 레이아웃

- 상대 레이아웃은 다른 뷰나 부모 뷰와의 상대적인 위치를 이용해 뷰를 배치하는 방법



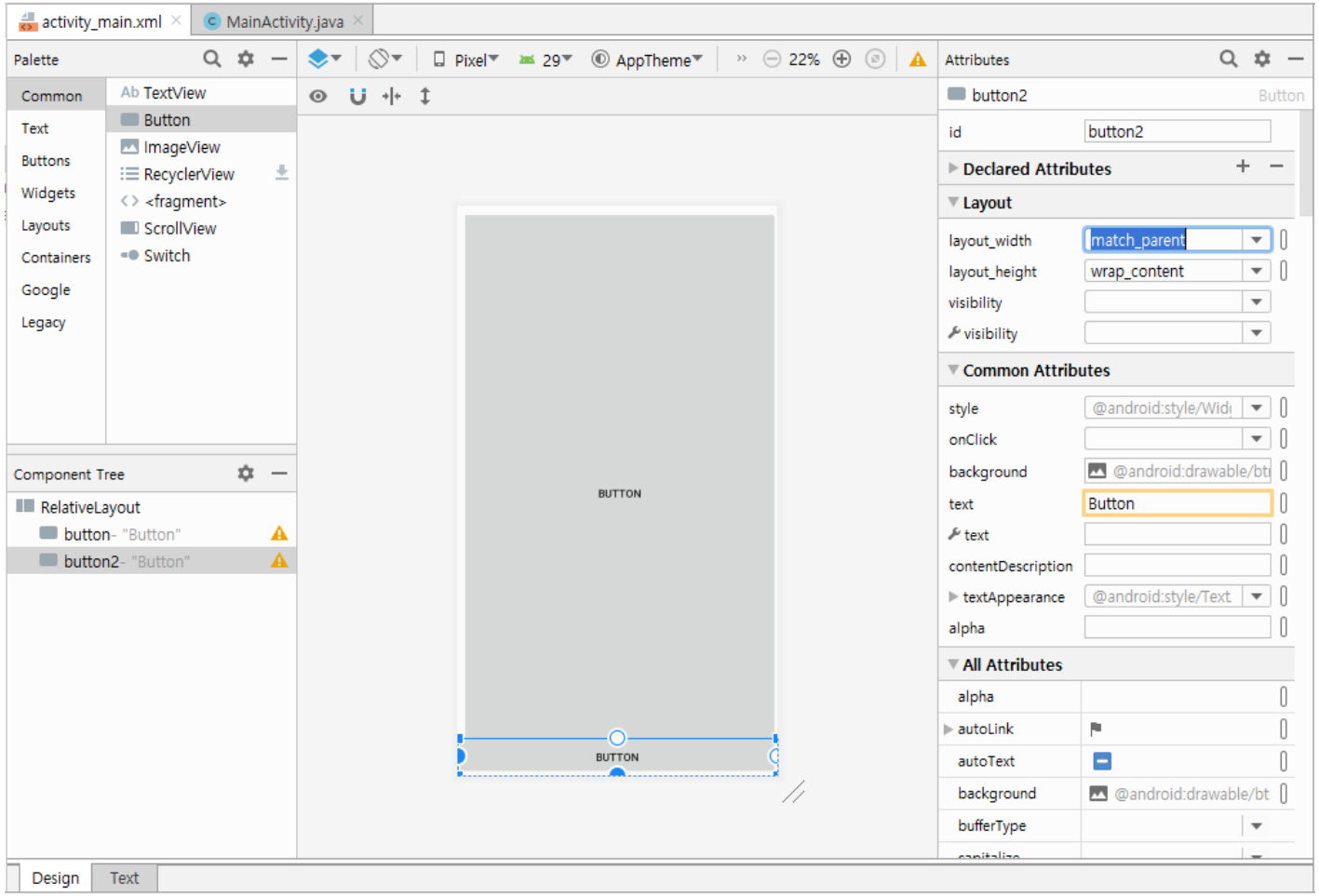
A : 부모 레이아웃의 위쪽

B : 뷰 A의 아래쪽

C : 뷰 B의 왼쪽

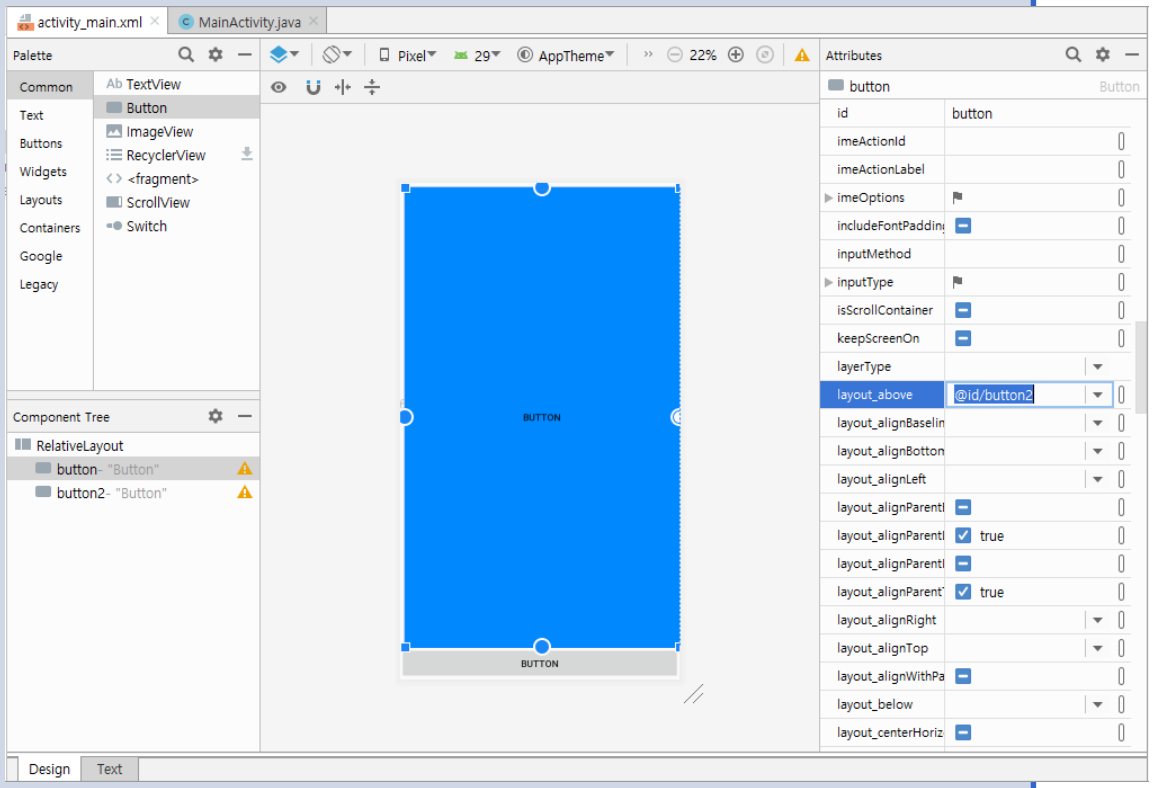
상대 레이아웃 화면 배치

- XML 레이아웃 파일에서 가운데 하나, 아래쪽에 하나 배치



상대 레이아웃의 속성 사용

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

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true“
        android:layout_above="@+id/button2"
        android:text="Button"
        android:background="#ff0088ff"/>
```



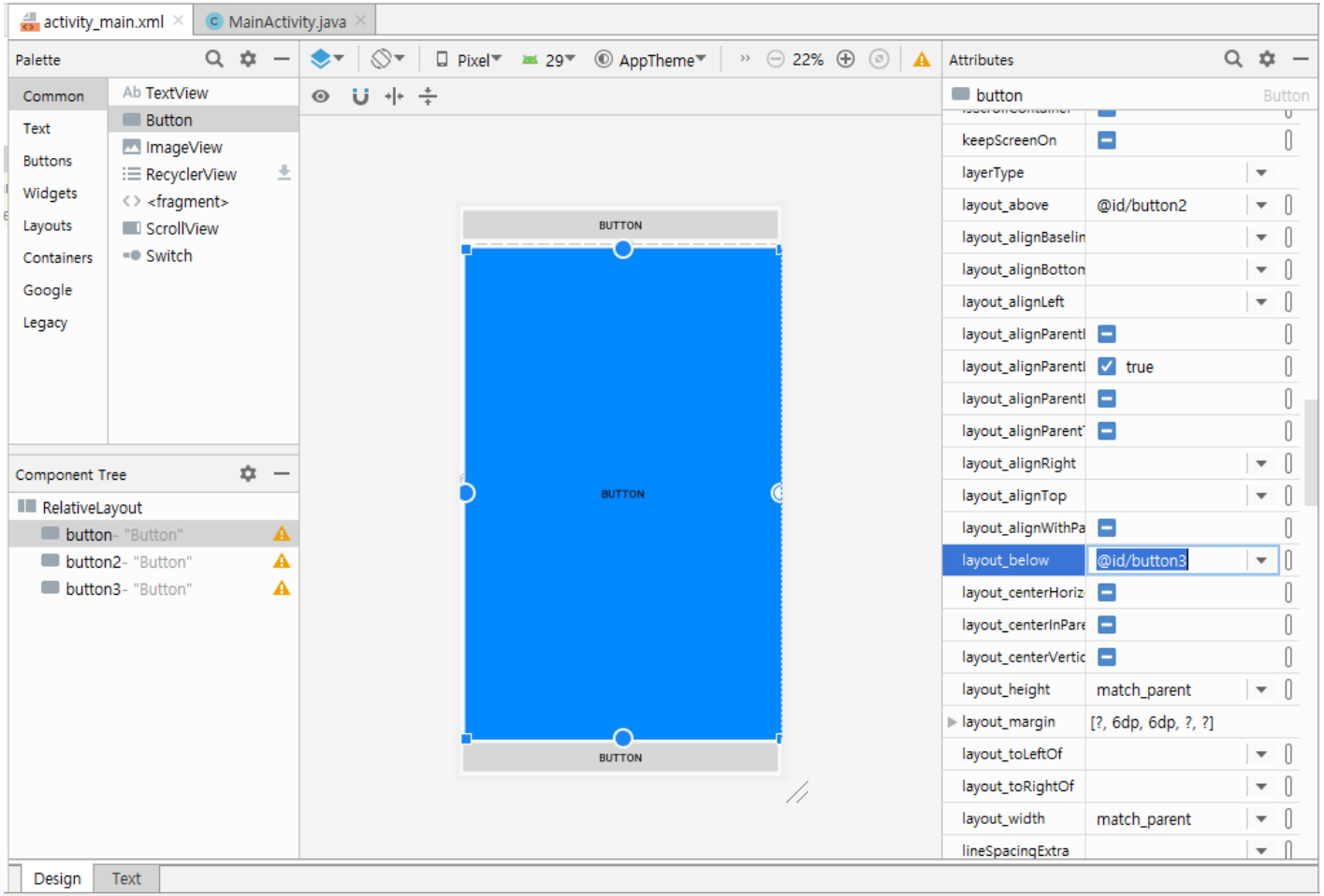
Continued..

상대 레이아웃의 속성 사용 (계속)

```
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

상대 레이아웃 화면 배치

- XML 레이아웃 파일에서 가운데 하나, 위와 아래쪽에 하나씩 배치



상대 레이아웃에서 사용할 수 있는 속성들

[상대 레이아웃에서 부모 컨테이너와의 상대적 위치를 이용하는 속성]

속성	설 명
layout_alignParentTop	- 부모 컨테이너의 위쪽과 뷰의 위쪽을 맞춤
layout_alignParentBottom	- 부모 컨테이너의 아래쪽과 뷰의 아래쪽을 맞춤
layout_alignParentLeft	- 부모 컨테이너의 왼쪽 끝과 뷰의 왼쪽 끝을 맞춤
layout_alignParentRight	- 부모 컨테이너의 오른쪽 끝과 뷰의 오른쪽 끝을 맞춤
layout_centerHorizontal	- 부모 컨테이너의 수평 방향 중앙에 배치함
layout_centerVertical	- 부모 컨테이너의 수직 방향 중앙에 배치함
layout_centerInParent	- 부모 컨테이너의 수평과 수직 방향 중앙에 배치함

상대 레이아웃에서 사용할 수 있는 속성들

[상대 레이아웃에서 다른 뷰와의 상대적 위치를 이용하는 속성]

속성	설 명
layout_above	- 지정한 뷰의 위쪽에 배치함
layout_below	- 지정한 뷰의 아래쪽에 배치함
layout_toLeftOf	- 지정한 뷰의 왼쪽에 배치함
layout_toRightOf	- 지정한 뷰의 오른쪽에 배치함
layout_alignTop	- 지정한 뷰의 위쪽과 맞춤
layout_alignBottom	- 지정한 뷰의 아래쪽과 맞춤
layout_alignLeft	- 지정한 뷰의 왼쪽과 맞춤
layout_alignRight	- 지정한 뷰의 오른쪽과 맞춤
layout_alignBaseline	- 지정한 뷰와 내용물의 아래쪽 기준선(baseline)을 맞춤

4. 테이블 레이아웃 사용하기

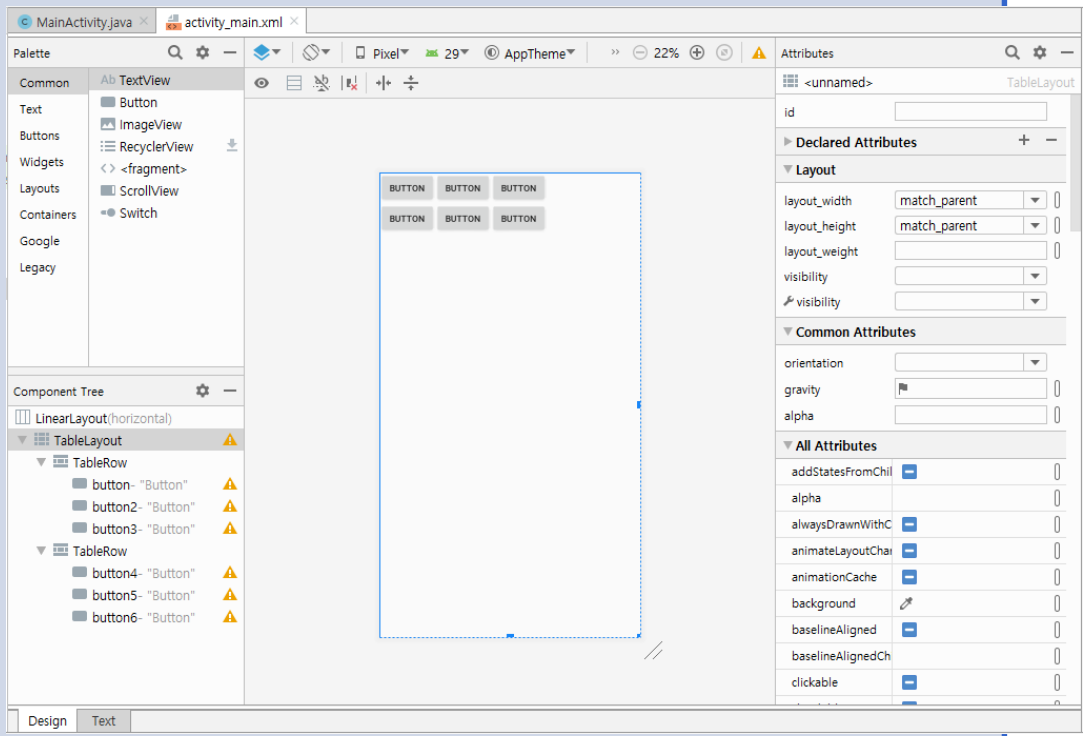
테이블 레이아웃

- 테이블 레이아웃은 격자 모양으로 뷰를 배치하는 방법

[테이블 레이아웃을 이용한 뷰의 배치 방법]

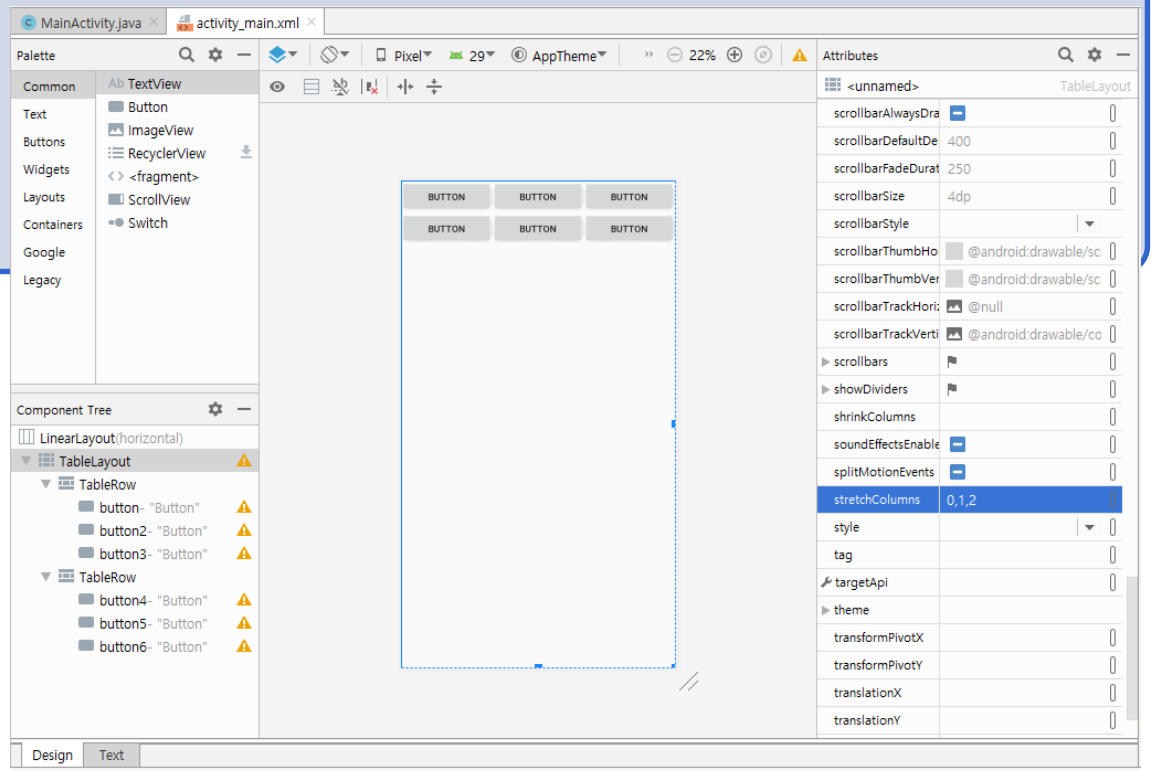
테이블 레이아웃

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

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

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



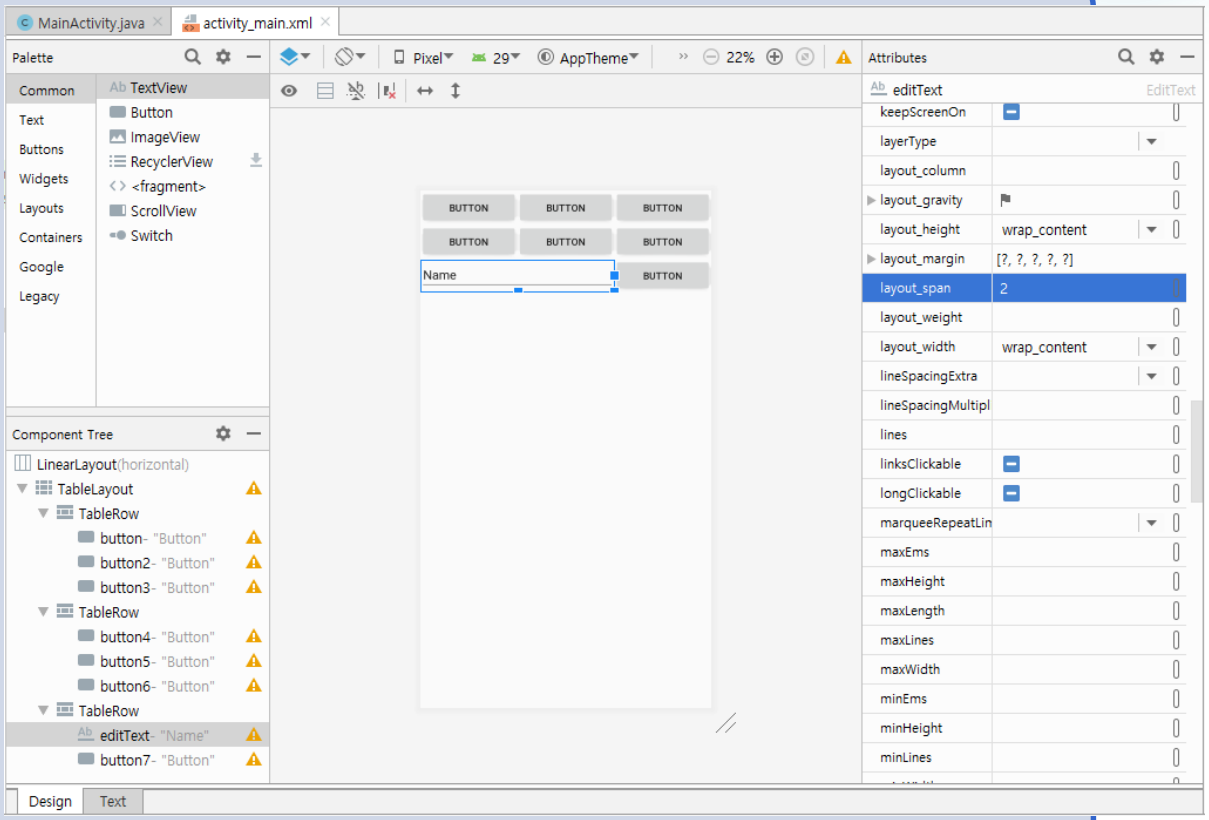
stretchColumns 속성

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent“
     android:stretchColumns="0,1,2" >

. . .
```



layout_span과 layout_column 속성의 사용

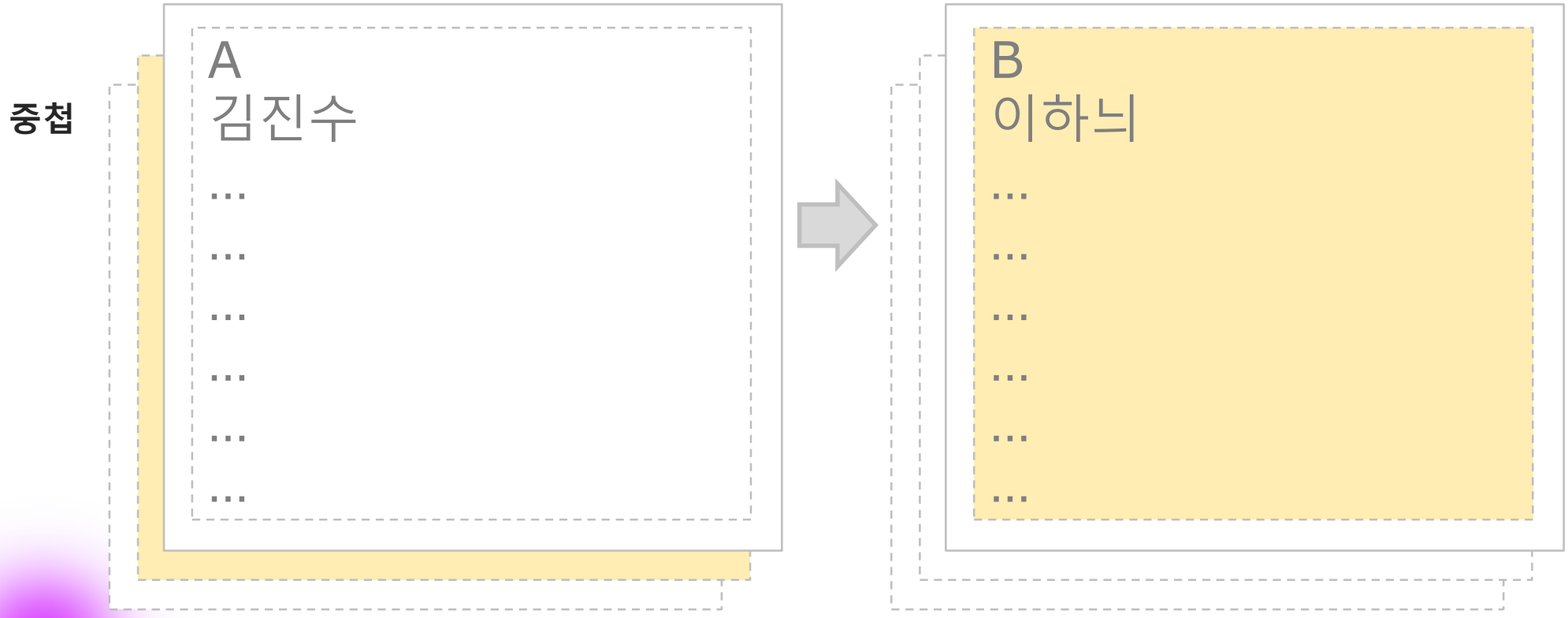
```
<EditText
android:id="@+id/editText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_span="3" />

. . .

<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_column="2"
android:text="아니오" />

. . .
```



5. 프레임 레이아웃과 뷰의 전환

프레임 레이아웃과 뷰의 전환

- 한 번에 하나의 뷰만 보여주며, 다른 뷰들은 그 아래에 중첩되어 쌓임
- 중첩되는 효과와 함께 뷰의 가시성(Visibility) 속성을 이용해 다양한 화면 구성이 가능함



[프레임 레이아웃과 가시성 속성의 사용]

뷰 A에서 뷰 B로 전환

가시성 속성 사용하기

- 사용 예 – XML 레이아웃

```
<LinearLayout
    android:id="@+id/layout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:visibility="gone"
    >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        />
</ScrollView>
```

- 사용 예 – 소스 코드

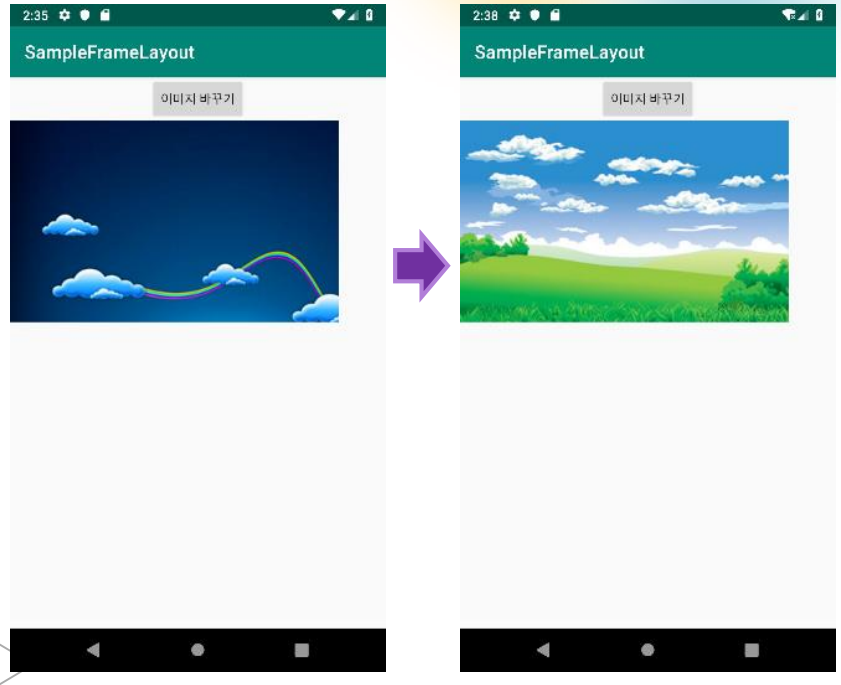
```
layout1.setVisibility(View.GONE);
layout1.setVisibility(View.VISIBLE);
layout1.setVisibility(View.INVISIBLE);
```

프레임 레이아웃과 뷰의 전환

뷰 전환 예제

-프레임 레이아웃을 이용해 뷰를 중첩하여 만들기

-버튼을 누르면 다른 이미지로 전환하기



XML 레이아웃

-레이아웃 코드 작성

메인 액티비티 코드

-메인 액티비티 코드 작성

프레임 레이아웃과 뷰의 전환 – XML 레이아웃

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

1 전환 버튼

2 화면 채우기

Continued..

프레임 레이아웃과 뷰의 전환 – XML 레이아웃

```
  <ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/dream01"
    android:visibility= "invisible"
  />
  <ImageView
    android:id="@+id/imageView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/dream02"
    android:visibility= "visible"
  />
 </FrameLayout>
</LinearLayout>
```

3 이미지 뷰 설정

4 이미지 뷰 설정

프레임 레이아웃과 뷰의 전환 – 메인 액티비티 코드

```
private void changeImage() {
    if (imageIndex == 0) {
        imageView1.setVisibility(View.VISIBLE);
        imageView2.setVisibility(View.INVISIBLE);
        imageIndex = 1;
    } else if (imageIndex == 1) {
        imageView1.setVisibility(View.INVISIBLE);
        imageView2.setVisibility(View.VISIBLE);
        imageIndex = 0;
    }
  }
}
```

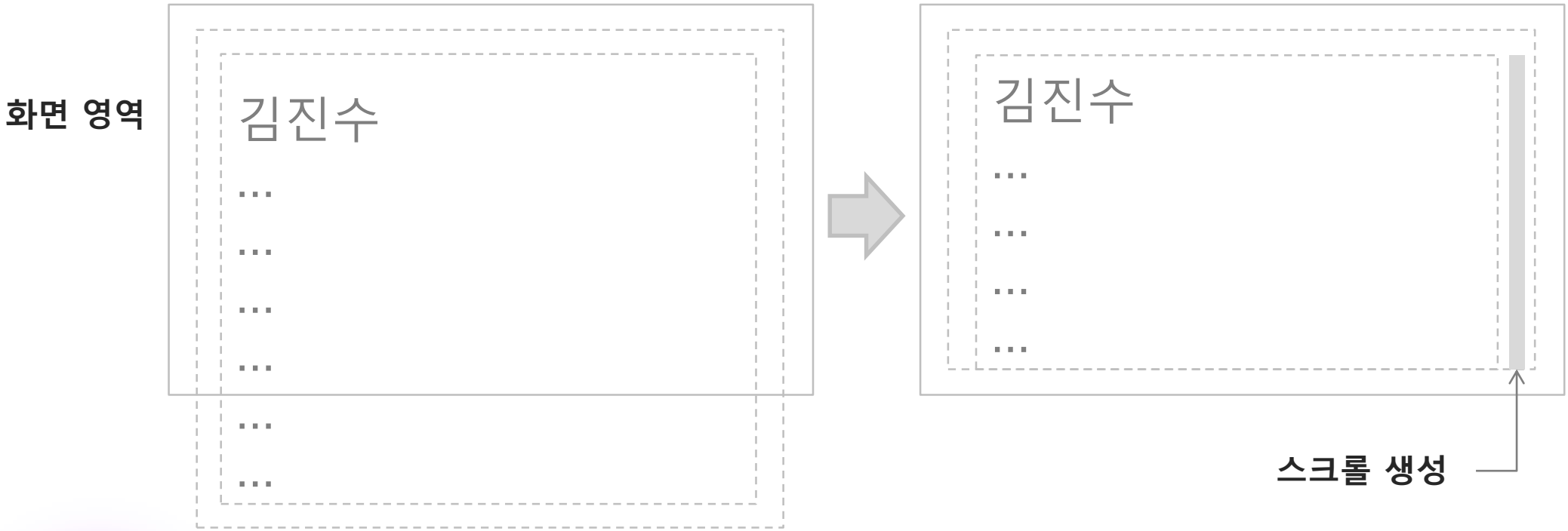
1 이미지 뷰 설정

2 이미지 뷰 설정

6. 스크롤뷰 사용하기

스크롤뷰 사용하기

- 위젯의 내용이 화면 영역을 벗어나면 스크롤 표시가 자동으로 보임
- 스크롤뷰를 이용해 다른 뷰를 감싸주기만 하면 됨



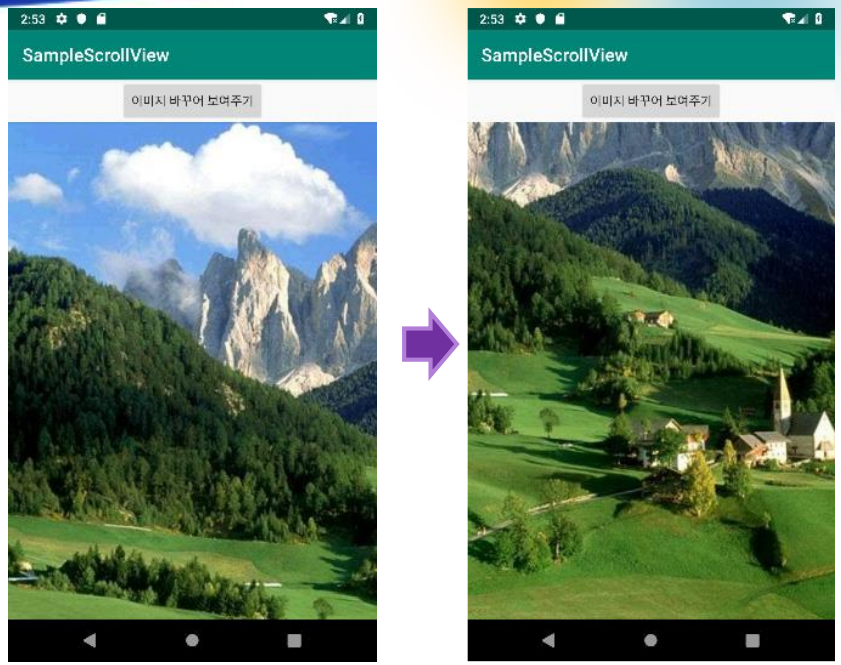
[스크롤뷰의 사용]

스크롤뷰 사용하기

- 사용 예 – 텍스트뷰를 스크롤뷰로 감싸주는 경우

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        />
</ScrollView>
```

스크롤뷰 사용하기



스크롤뷰 예제

-주어진 영역보다 큰 이미지에 스크롤이 생기도록 만들기

XML 레이아웃

-레이아웃 코드 작성

메인 액티비티 코드

-메인 액티비티 코드 작성

스크롤뷰의 XML 레이아웃

```
android:onClick="onButton1Clicked"
/>

<HorizontalScrollView
    android:id="@+id/horScrollView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <ScrollView
        android:id="@+id/scrollView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <ImageView
            android:id="@+id/imageView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            />
```

Continued..

스크롤뷰의 메인 액티비티 코드

```
...
scrollView = (ScrollView) findViewById(R.id.scrollView);
imageView = (ImageView) findViewById(R.id.imageView);
...
scrollView.setHorizontalScrollBarEnabled(true);
Resources res = getResources();
BitmapDrawable bitmap = (BitmapDrawable)
                        res.getDrawable(R.drawable.system_architecture);
int bitmapWidth = bitmap.getIntrinsicWidth();
int bitmapHeight = bitmap.getIntrinsicHeight();
imageView.setImageDrawable(bitmap);
imageView.getLayoutParams().width = bitmapWidth;
imageView.getLayoutParams().height = bitmapHeight;
...
```

1 객체 참조

2 기능 설정

3 이미지 참조

4 크기 설정

Continued..