



[한국ICT인재개발원] 안드로이드

4. 레이아웃 익히기

前) 광고데이터 분석 1년

前) IT강의 경력 2년 6개월

前) 머신러닝을 활용한 데이터 분석 프로젝트반 운영 1년

前) 리그오브 레전드 데이터 분석 등...

現) 국비반 강의 진행중

레이아웃은 위젯을 배치할 수 있는 틀입니다.

여러가지 레이아웃을 구현할 수 있지만

먼저 이쪽 챕터에서는 가장 핵심이 되는 “리니어 레이아웃”을 먼저 공부하고

이어서 렐러티브 레이아웃, 테이블 레이아웃, 그리드 레이아웃, 프레임 레이아웃 등을 공부해보겠습니다.

```
public abstract class ViewGroup
extends View implements ViewParent, ViewManager

java.lang.Object
└─ android.view.View
    └─ android.view.ViewGroup

  Known direct subclasses
  AbsoluteLayout, AdapterView<T extends Adapter>, FragmentBreadCrumbs, FrameLayout, GridLayout, InlineContentView, LinearLayout,
  RelativeLayout, SlidingDrawer, Toolbar, TextView
```

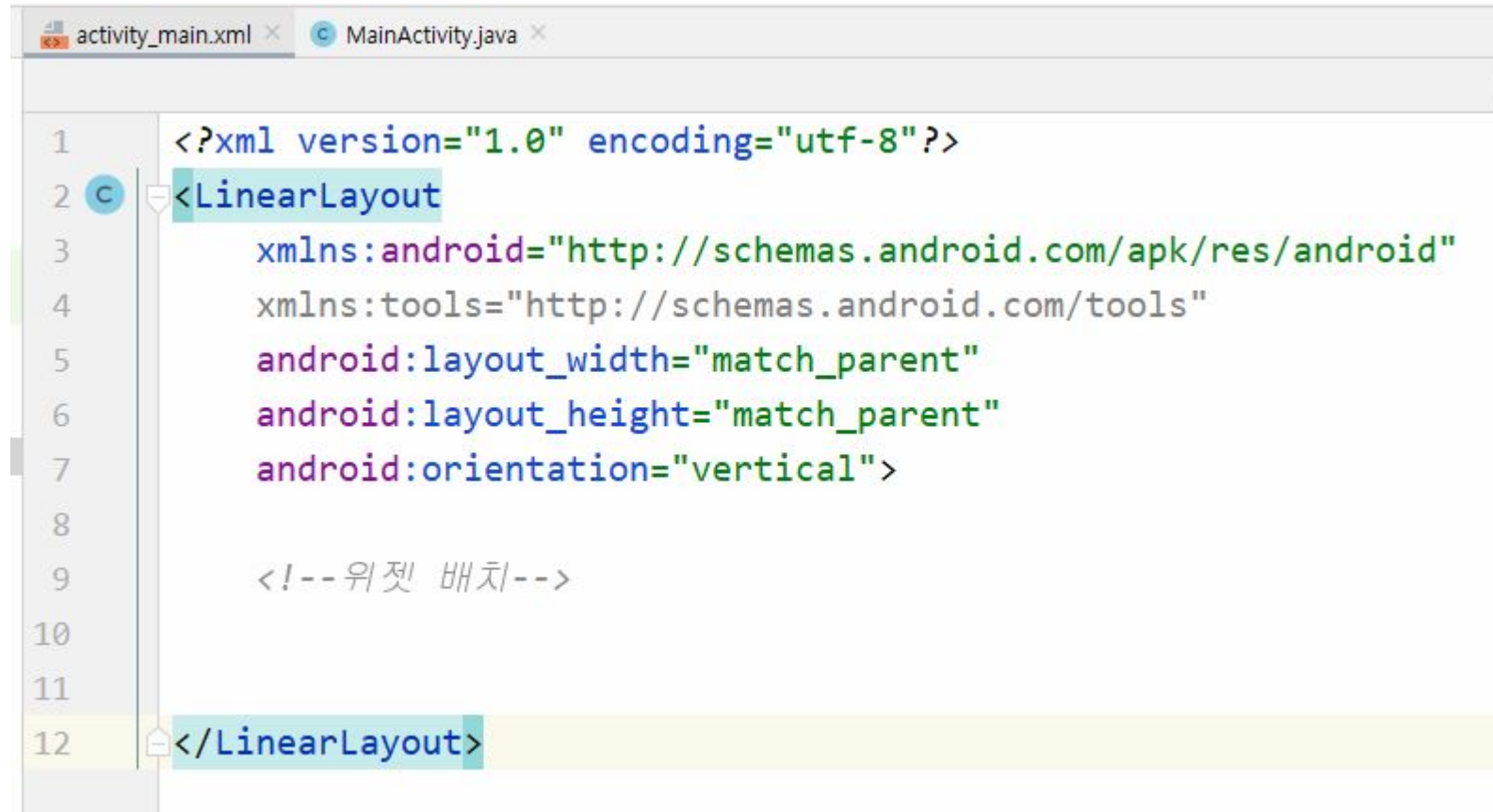
레이아웃은 기본적으로 **ViewGroup**클래스로부터 상속받도록 설계되어있습니다. 따라서 **View**에서 사용되는 모든 메서드와 속성을 사용할 수 있습니다.

```
public class LinearLayout
extends ViewGroup

java.lang.Object
└─ android.view.View
    └─ android.view.ViewGroup
        └─ android.widget.LinearLayout

▼ Known direct subclasses
ActionMenuView, NumberPicker, RadioGroup, SearchView, TabWidget, TableLayout, TableRow, ZoomControls
```

리니어레이아웃은 기본적으로 프로젝트 생성시 쓰이는 레이아웃이다.
orientation 속성에 값으로 **vertical**을 준다면 왼쪽 위부터 수직방향으로
쌓이고, **horizontal**은 수평방향으로 부품을 쌓아나간다는 뜻이다.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:orientation="vertical">
8
9     <!-- 위젯 배치 -->
10
11
12 </LinearLayout>
```

activity_main.xml 내부 코드를 위와 같이 변경하면 LinearLayout을 사용할 수 있습니다.

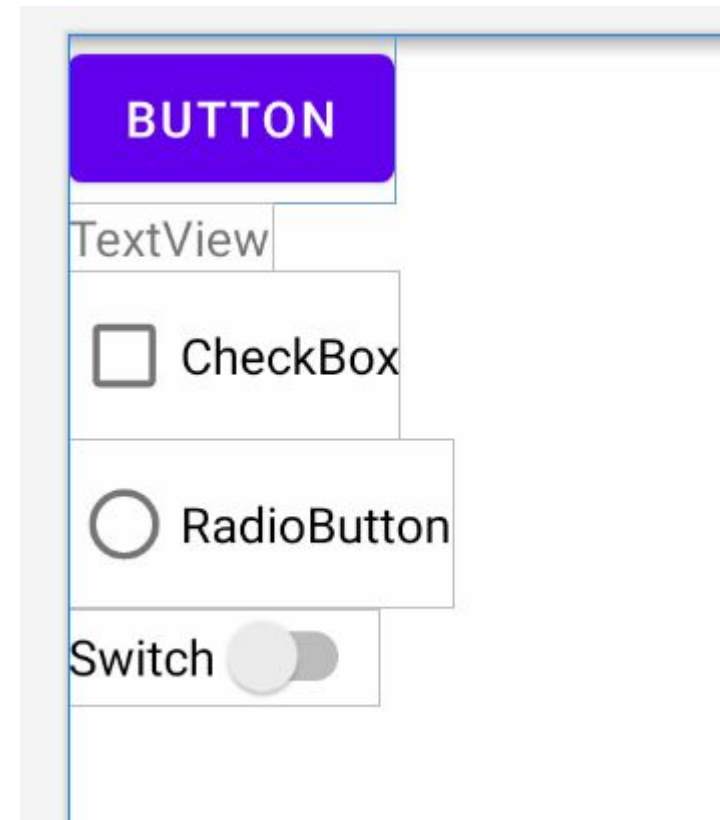
```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Button" />
```

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

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

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

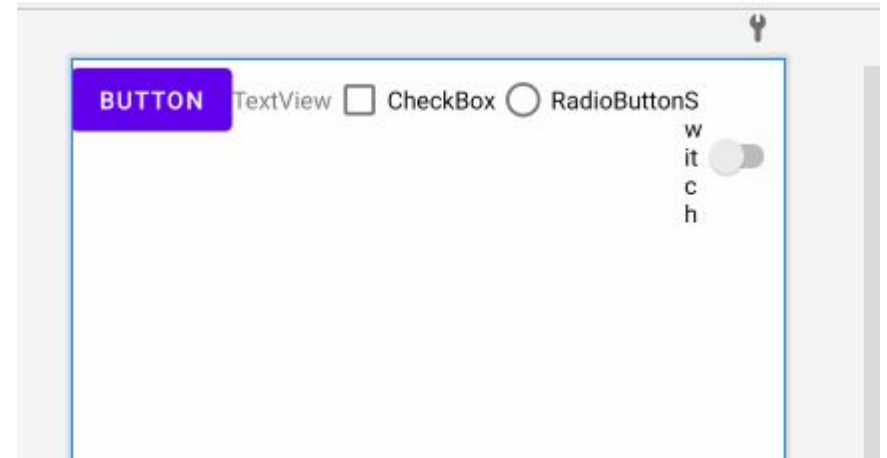
```
<Switch  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Switch" />
```



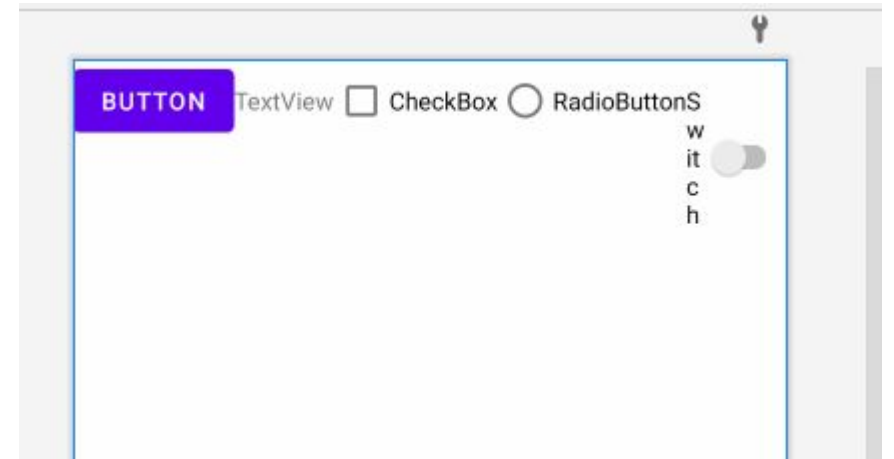
위와 같이 순서가 겹치지 않고 수직방향으로 내려오는 이유는
LinearLayout을 사용하면서, orientation에 vertical을 줬기 때문입니다.

```
<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="horizontal">

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



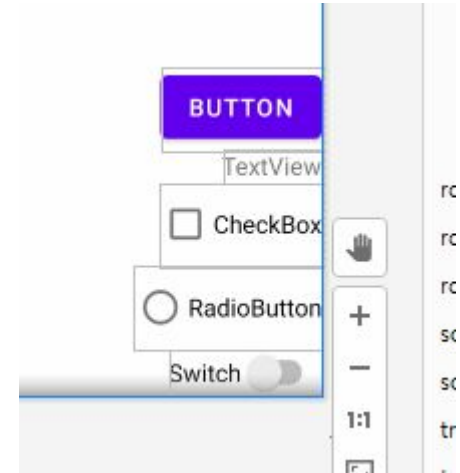
위와 같이 순서가 겹치지 않고 수직방향으로 내려오는 이유는
LinearLayout을 사용하면서, orientation에 vertical을 줬기 때문입니다.



위와 같이 순서가 겹치지 않고 수직방향으로 내려오는 이유는
LinearLayout을 사용하면서, orientation에 vertical을 줬기 때문입니다.

```
<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="horizontal">

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

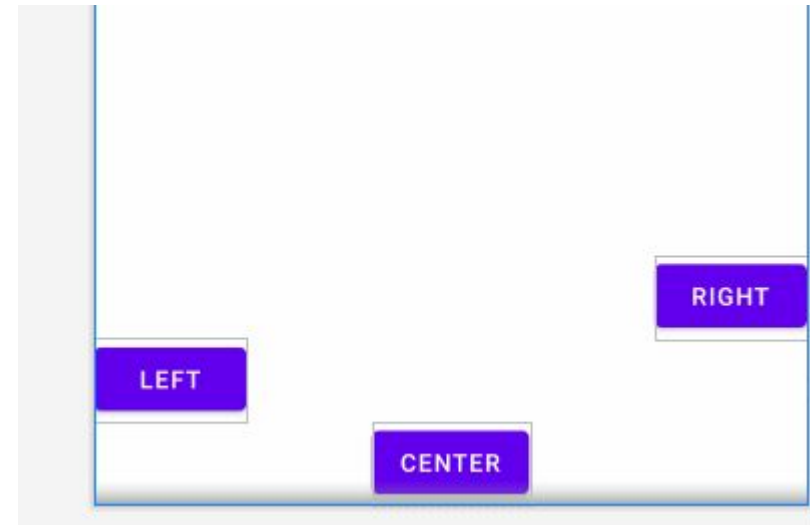


위와 같이 순서가 겹치지 않고 수직방향으로 내려오는 이유는
LinearLayout을 사용하면서, orientation에 vertical을 줬기 때문입니다.

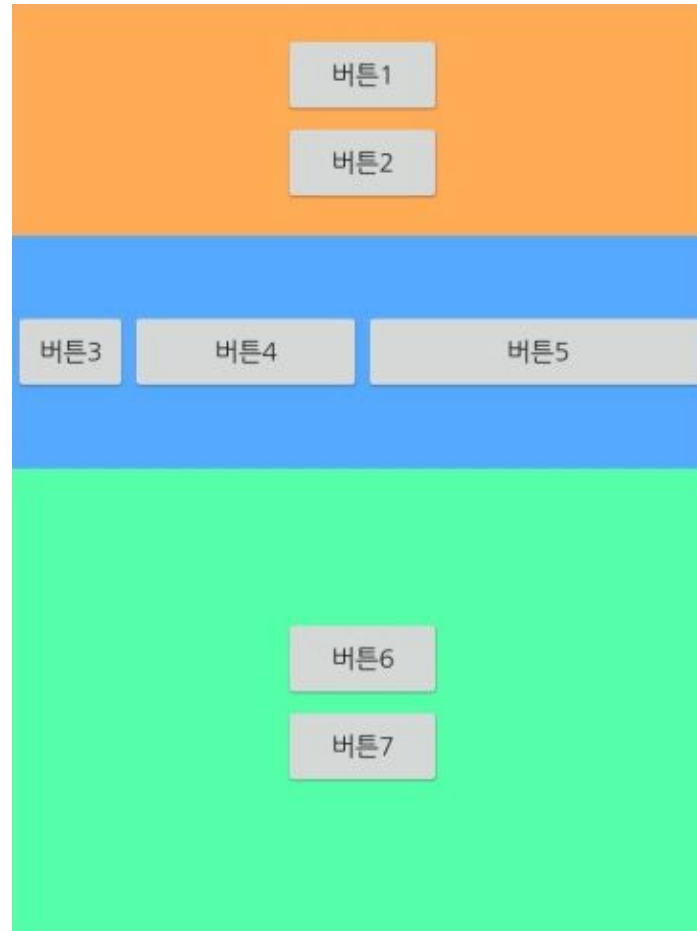
```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="right"
    android:text="right" />
```

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

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



위와 같이 순서가 겹치지 않고 수직방향으로 내려오는 이유는
LinearLayout을 사용하면서, orientation에 vertical을 줬기 때문입니다.



레이아웃은 중첩해서 작성할 수도 있습니다.
레이아웃 내부에 레이아웃을 선언하면 중첩 레이아웃이 되며
이 경우, 공간 처리가 좀 더 수월합니다.

```
<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">
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">
    <Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="버튼 1"></Button>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼 2"></Button>
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:background="#00FF00"
    android:orientation="horizontal">
    <Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="버튼 3"></Button>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼 4"></Button>
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#0000FF"
    android:gravity="center"
    android:orientation="vertical">
    <Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="버튼 5"></Button>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼 6"></Button>
</LinearLayout>
```

```
</LinearLayout>
```

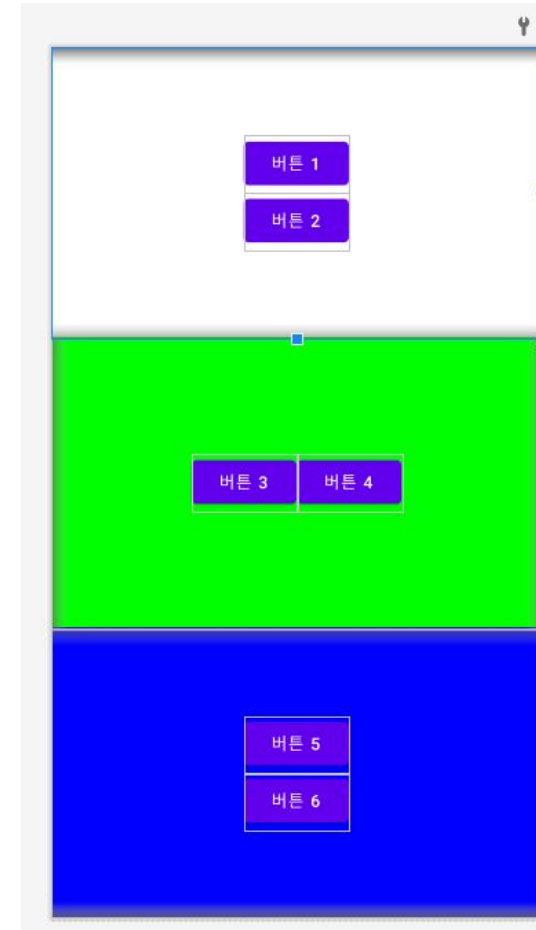
리니어레이아웃 내부에 리니어레이아웃을 추가로 작성해 3개 층을 만들어줍니다.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:background="#00FF00"
    android:orientation="horizontal">
    <Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="버튼 3"></Button>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼 4"></Button>
</LinearLayout>
```



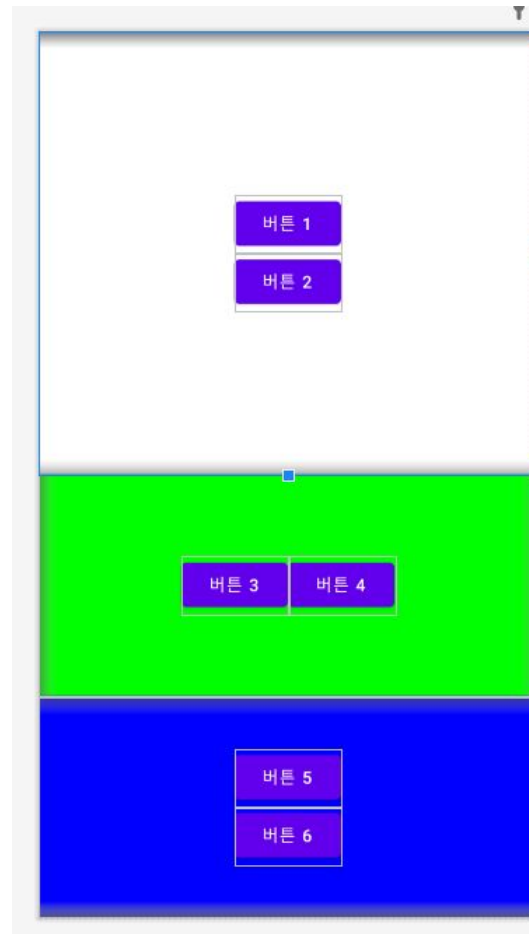
상기 코드의 내부 레이아웃의 **layout_height**을 **wrap_content**로 바꾸어 내부 버튼이 점유하는 높이만큼만 레이아웃 높이를 가지도록 변경하면 비로소 제대로 3개의 층이 나타납니다.


```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:layout_weight="1"
    android:orientation="vertical">
    <Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="버튼 1"></Button>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼 2"></Button>
</LinearLayout>
```

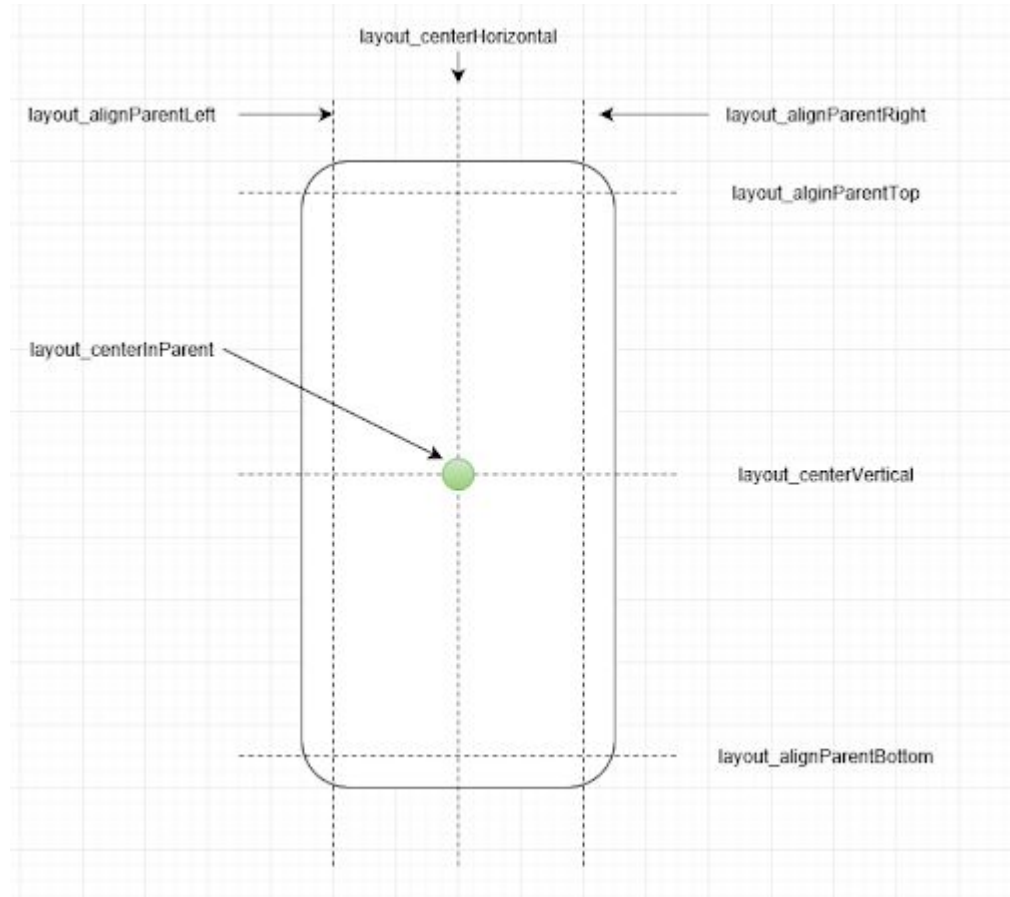


이를 방지하려면, 우선 레이아웃은 버튼 크기와는 상관없이 형성되어야 하며 1:1:1로 가져가야 합니다. 따라서 모든 레이아웃에 `android:layout_weight="1"` 을 기입해 1:1:1을 가져가도록 합니다.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:gravity="center"
    android:layout_weight="2"
    android:orientation="vertical">
    <Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="버튼 1"></Button>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼 2"></Button>
</LinearLayout>
```



이를 방지하려면, 우선 레이아웃은 버튼 크기와는 상관없이 형성되어야 하며 1:1:1로 가져가야 합니다. 따라서 모든 레이아웃에 `android:layout_weight="1"` 을 기입해 1:1:1을 가져가도록 합니다.



렐러티브 레이아웃은 하나의 중심 위젯을 두고 그 위젯을 중심으로 다른 위젯을 배치하는 방식입니다. 리니어레이아웃보다는 덜 사용하지만 간혹가다 특정 디자인을 특화시키기엔 좋기때문에 사용합니다.

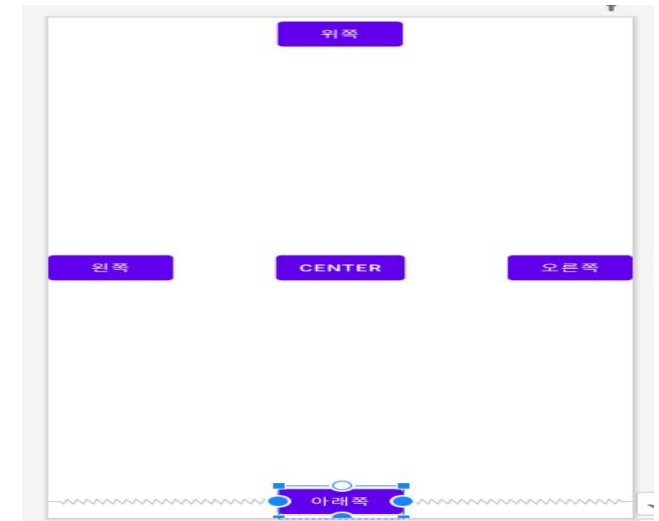
```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="center"
    android:layout_centerInParent="true">
</Button>
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="위쪽"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true">
</Button>
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="왼쪽"
    android:layout_alignParentLeft="true"
    android:layout_centerVertical="true">
</Button>
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="오른쪽"
    android:layout_alignParentRight="true"
    android:layout_centerVertical="true">
</Button>
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="아래쪽"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true">
</Button>
```



렐러티브 레이아웃은 먼저 `layout_centerInParent`를 받는 하나의 중심 위젯을 설치한 다음, 그 위젯을 바탕으로 `alignParent`(방향), `center`(방향) 두 가지 정보를 조합해서 상대적 위치를 부여합니다.

```
<Button
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:id="@+id/baseBtn"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="기준 위젯"></Button>
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/baseBtn"
    android:layout_toLeftOf="@+id/baseBtn"
    android:text="alignTop, toLeftOf"
></Button>
```



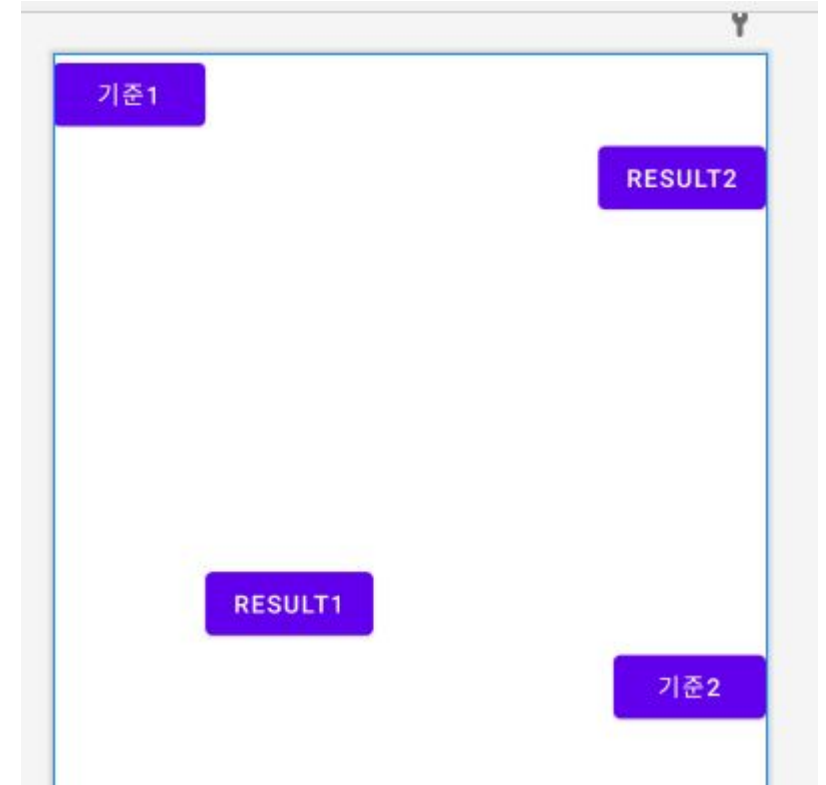
기준 위젯에 id를 부여하고, 해당 위젯을 중심으로 상대적 위치에 배열할 수도 있습니다. 이 경우는 위와 같이 align속성과 방향속성(toLeftOf, Above, Below, toRightOf)를 함께 지정해 상대적 위치를 지정할 수 있습니다.

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/baseBtn1"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:text="기준1"></Button>
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/baseBtn2"
    android:layout_alignParentRight="true"
    android:layout_centerVertical="true"
    android:text="기준2"></Button>
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/baseBtn2"
    android:layout_toRightOf="@id/baseBtn1"
    android:text="result1"
></Button>
```

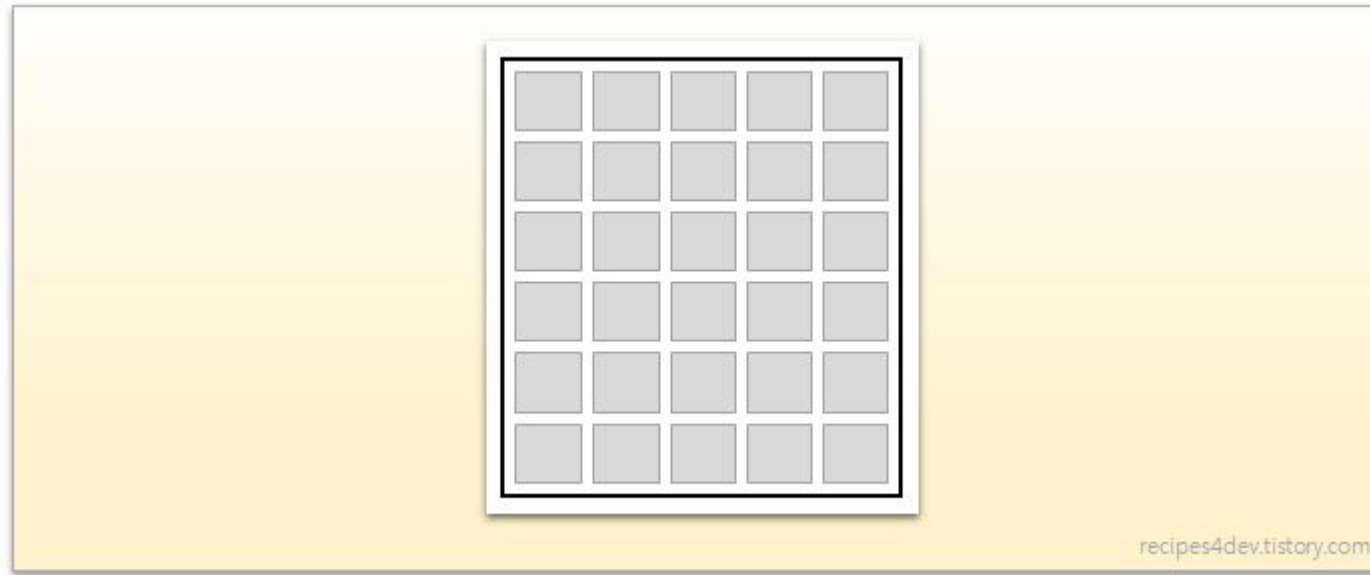
```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_below="@id/baseBtn1"
    android:text="result2"
></Button>
```



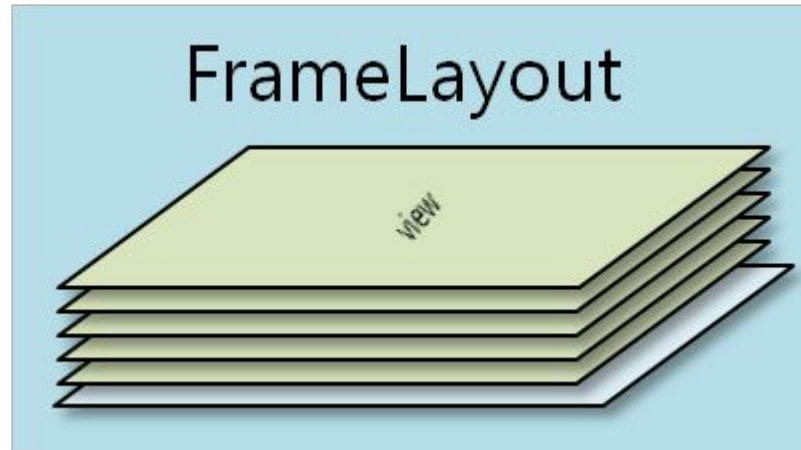
두 개 이상의 버튼을 대상으로도 상대적 위치를 각각 부여할 수 있습니다.
이 경우 가로축, 세로축 설정이 각각 참조한 버튼을 기준으로 부여됩니다.



테이블레이아웃은 마치 부트스트랩의 `div class="row"` 처럼 하나하나 row를 구성해 그 내부에 부품을 배치하는 식으로 사용합니다.



그리드 레이아웃은 격자무늬 좌표를 생성해서 그 좌표에 맞게 위젯을 배치하는 방법입니다. 좌표식으로 배치하기때문에 굉장히 표현이 편하다는 장점이 있습니다.



프레임 레이아웃은 프레임처럼 큰 레이아웃을 생성해 그 내부에 겹쳐서 출력하는것을 의미합니다.

자주 사용되지는 않기때문에 연습만 해 보고 넘어갑니다.