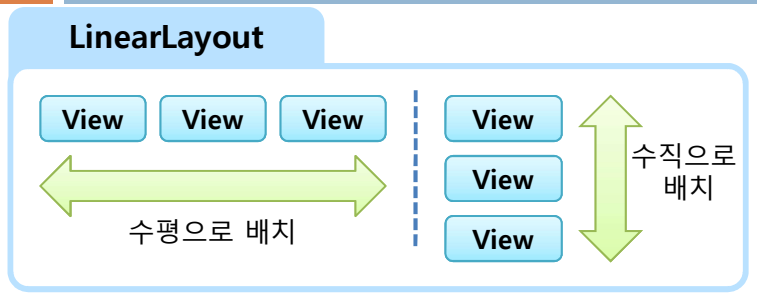
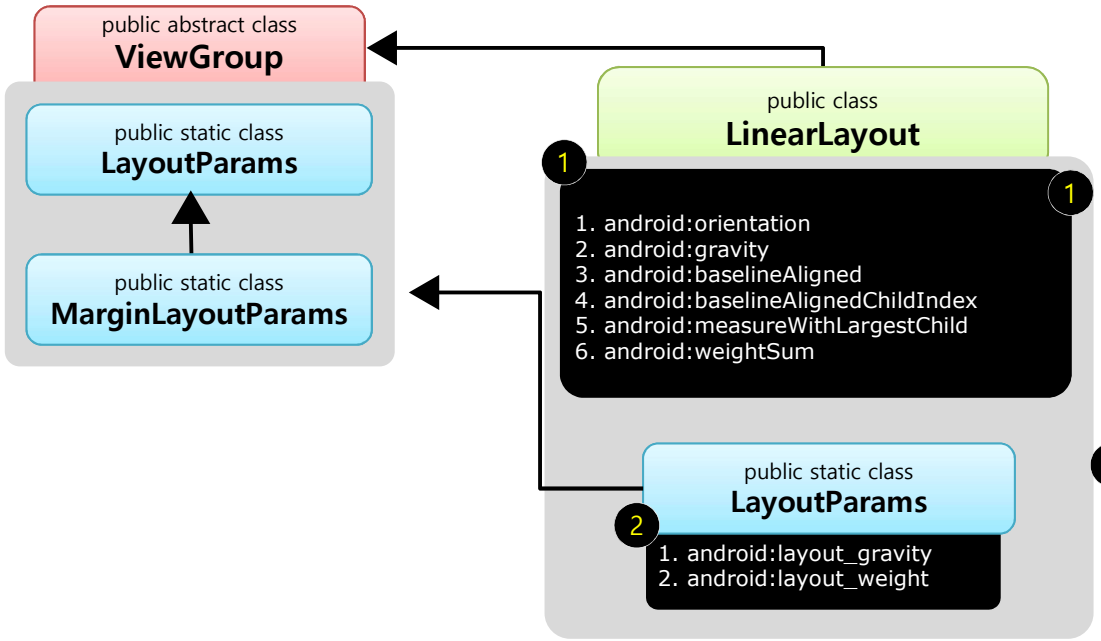
대표적인 뷰그룹

LinearLayout과 LinearLayout.LayoutParams

2





1 뷰그룹 자체의 속성

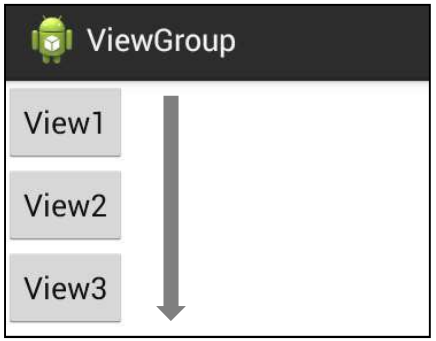
2 뷰그룹이 자식뷰를 배치하기 위해 자식뷰에게 요구하는 속성

뷰그룹은 자체 뷰그룹의 속성과 LayoutParams 속성을 함께 학습해야 정확한 원리를 알 수 있다.

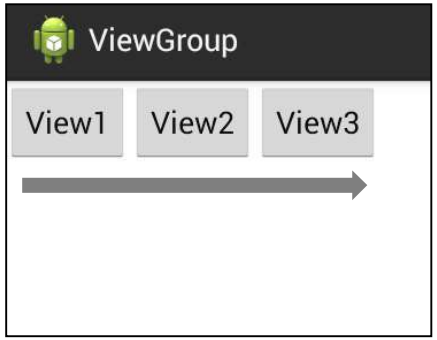
LinearLayout의 기본 속성 - android:orientation

res/layout/activity_main.xml

```
<LinearLayout xmlns:android="…"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View1" />

    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View2" />

    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View3" />

</LinearLayout>
```



orientation= "***vertical***"

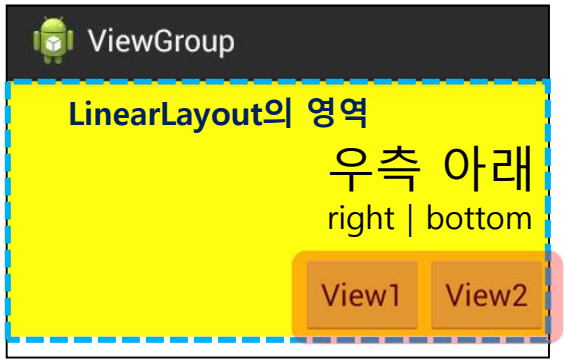


orientation= "***horizontal***"

LinearLayout의 기본 속성 - android:gravity

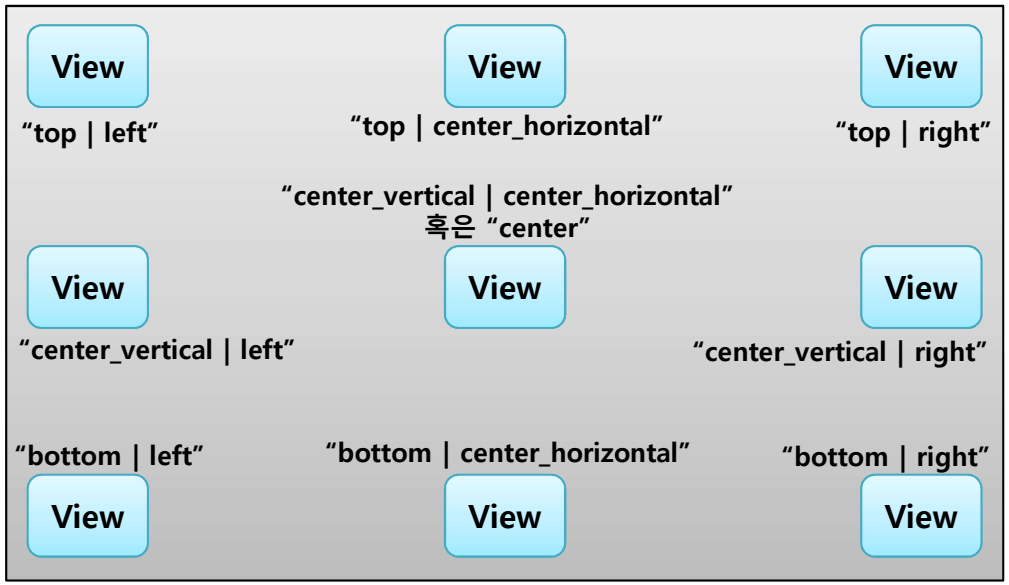
res/layout/activity_main.xml

```
<LinearLayout xmlns:android="…"
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:background="#FF0"
    android:orientation="horizontal"
    android:gravity="right|bottom">

    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="View1" />

    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="View2" />

</LinearLayout>
```



■ **gravity 속성값**

- top : 상단에 배치
- bottom : 하단에 배치
- left : 좌측에 배치
- right : 우측에 배치
- center_vertical : 수직 중앙에 배치
- center_horizontal : 수평 중앙에 배치
- center : 정 중앙에 배치

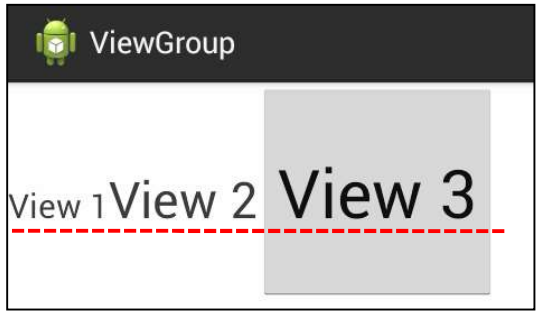


LinearLayout의 기본 속성 - android:baselineAligned

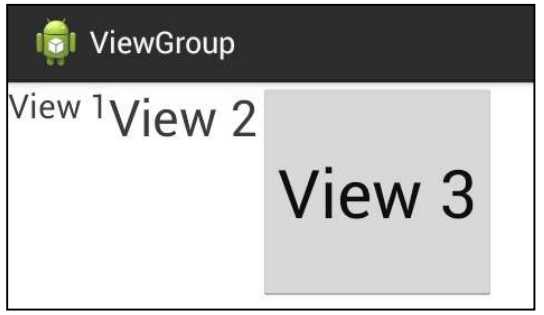
- android:baselineAligned
  - 기본값은 true
  - 이 속성값을 true로 설정하면 텍스트가 포함된 자식 뷰 중 가장 높이가 긴 뷰를 기준으로 정렬
    - 텍스트가 포함된
      - TextView를 상속받은 뷰
  - 자식 뷰들이 수평으로 배치된 경우에만 적용
    - 수직으로 배치된 경우에는 적용되지 않음

LinearLayout의 기본 속성 - android:baselineAligned


```
res/layout/activity_main.xml
<LinearLayout xmlns:android="…"
    android:layout_width="match_parent"
    android:layout_height="130dp"
    android:orientation="horizontal"
    android:baselineAligned="true">

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:text="View 1"/>

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30dp"
        android:text="View 2" />

    <Button android:layout_width="wrap_content"
        android:layout_height="130dp"
        android:textSize="40dp"
        android:text="View 3" />

</LinearLayout>
```



baselineAligned= "*true*"



baselineAligned= "*false*"

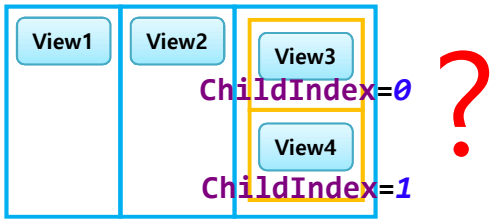
■ **왜 baselineAligned이라는 속성이 필요할까?**

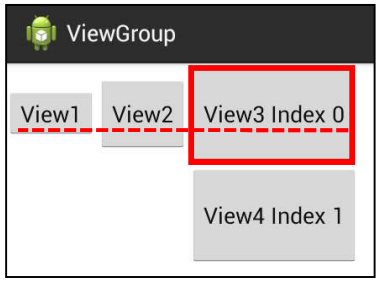
크기가 다른 텍스트 뷰들이 수평으로 배치되었을 때는 텍스트의 하단을 기준으로 정렬하는 것이 보기가 좋기 때문이다.

LinearLayout의 기본 속성 -
android:baselineAligned와 android:baselineAlignedChildIndex

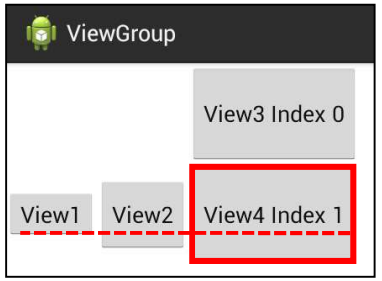
res/layout/activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="160dp"
    android:orientation="horizontal"
    android:baselineAligned="true">

    <Button android:layout_width="wrap_content"
        android:layout_height="40dp"
        android:text="View1"/>

    <Button android:layout_width="wrap_content"
        android:layout_height="60dp"
        android:text="View2" />

    <LinearLayout android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:baselineAlignedChildIndex="1">

        <Button android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="View3 Index 0" />

        <Button android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="View4 Index 1" />
    </LinearLayout>

</LinearLayout>
```





baselineAlignedChildIndex= "0"

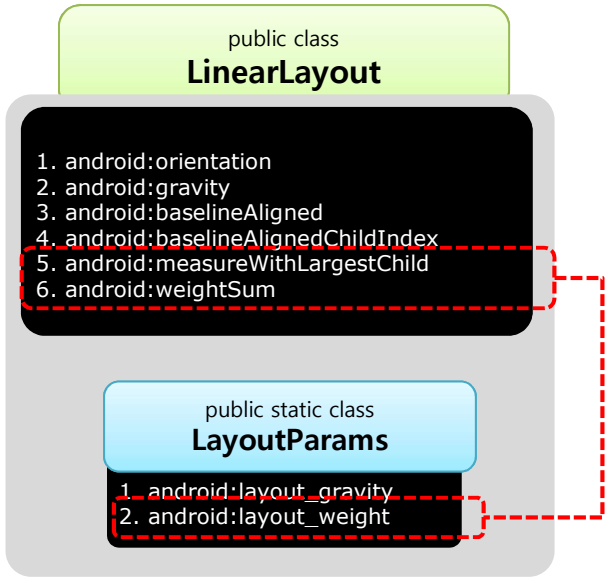


baselineAlignedChildIndex= "1"

LinearLayout의 기본 속성 - android:measureWithLargestChild와 android:weightSum

measureWithLargestChild와 weightSum 속성을 설명하기 위해서는

LayoutParams의 layout_weight 속성에 대한 사전지식이 필요하다.



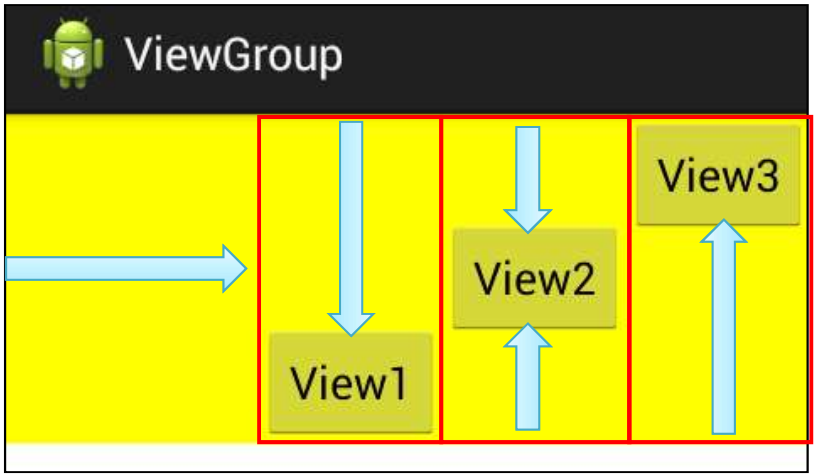
LinearLayout.LayoutParams 속성 – android:layout_gravity

res/layout/activity_main.xml

```
<LinearLayout xmlns:android="…"
    android:layout_width="match_parent"
    android:layout_height="130dp"
    android:orientation="horizontal"
    android:background="#FF0"
    android:gravity="right">

    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:text="View1"/>

    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:text="View2" />

    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:text="View3" />

</LinearLayout>
```

■ **gravity 속성값**

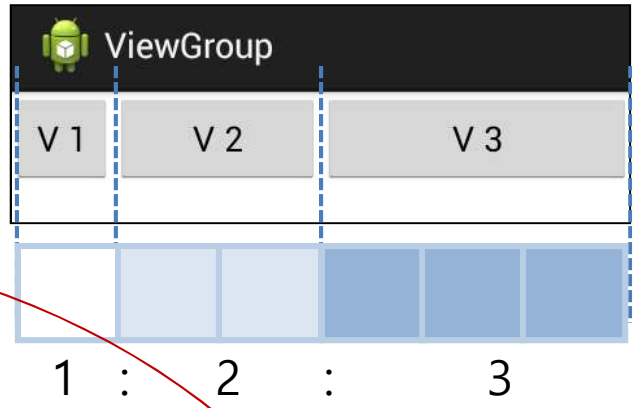
- top : 상단에 배치
- bottom : 하단에 배치
- left : 좌측에 배치
- right : 우측에 배치
- center_vertical : 수직 중앙에 배치
- center_horizontal : 수평 중앙에 배치
- center : 정 중앙에 배치



LinearLayout.LayoutParams 속성 – android:layout_weight

res/layout/activity_main.xml

```
<LinearLayout xmlns:android="…"
    android:layout_width="match_parent"
    android:layout_height="130dp"
    android:orientation="horizontal">

    <Button android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="V 1"/>

    <Button android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:text="V 2" />

    <Button android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:text="V 3" />

</LinearLayout>
```

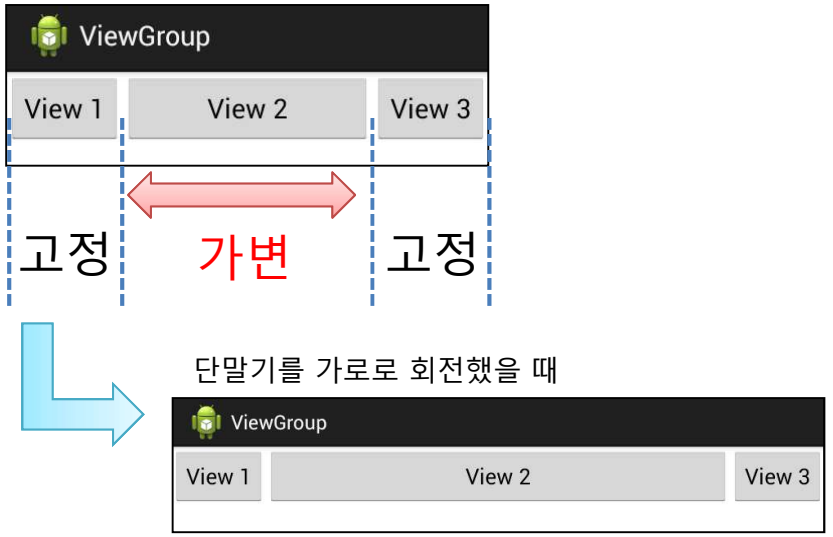


비율을 통해 크기를 결정하므로 크기는 당장 알 수 없다는 의미로 0을 설정

layout_weight 속성은 LinearLayout이 수평 배치일 때는 자식 뷰들의 너비를 비율로 조정하고 수직 배치일 때는 높이를 조정한다.

LinearLayout.LayoutParams 속성 – android:layout_weight


```
res/layout/activity_main.xml
<LinearLayout xmlns:android="…"
    android:layout_width="match_parent"
    android:layout_height="130dp"
    android:orientation="horizontal">

    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="View 1"/>

    <Button android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="View 2" />

    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="View 3" />

</LinearLayout>
```



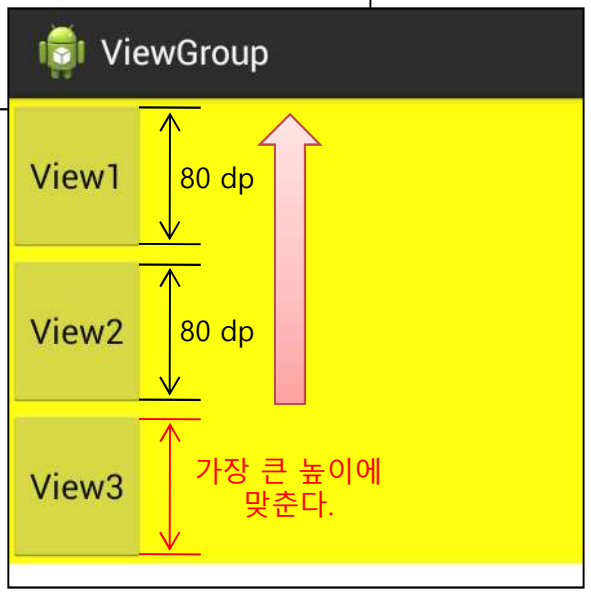
LinearLayout.LayoutParams의 layout_weight 속성은 특정 자식뷰의 크기를 가변적으로 조절할 수 있기 때문에 다양한 화면 크기의 단말에서 유연하게 레이아웃을 유지할 수 있다.

매우 빈번히 사용되니 꼭 기억해두자.

LinearLayout.LayoutParams의 android:layout_weight 속성과 LinearLayout의 android:measureWithLargestChild 속성

res/layout/activity_main.xml

```
<LinearLayout xmlns:android=""
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:background="#FF0"
  android:measureWithLargestChild="false">

    <Button android:layout_width="wrap_content"
      android:layout_height="30dp"
      android:layout_weight="1"
       android:text="View1"/>

    <Button android:layout_width="wrap_content"
      android:layout_height="50dp"
      android:layout_weight="1"
       android:text="View2" />

    <Button android:layout_width="wrap_content"
      android:layout_height="80dp"
      android:layout_weight="1"
       android:text="View3" />

</LinearLayout>
```



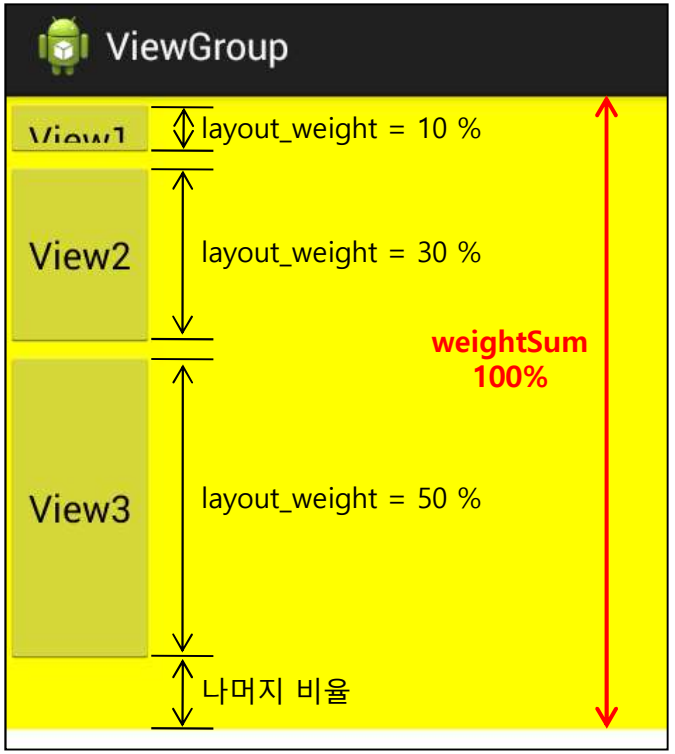
measureWithLargestChild 속성을 "true"로 설정하면 레이아웃 내 layout_weight 값이 (> 0)로 설정된 모든 자식뷰를 가장 큰 자식뷰의 크기로 조정됨

LinearLayout.LayoutParams의 android:layout_weight 속성과 LinearLayout의 android:weightSum 속성

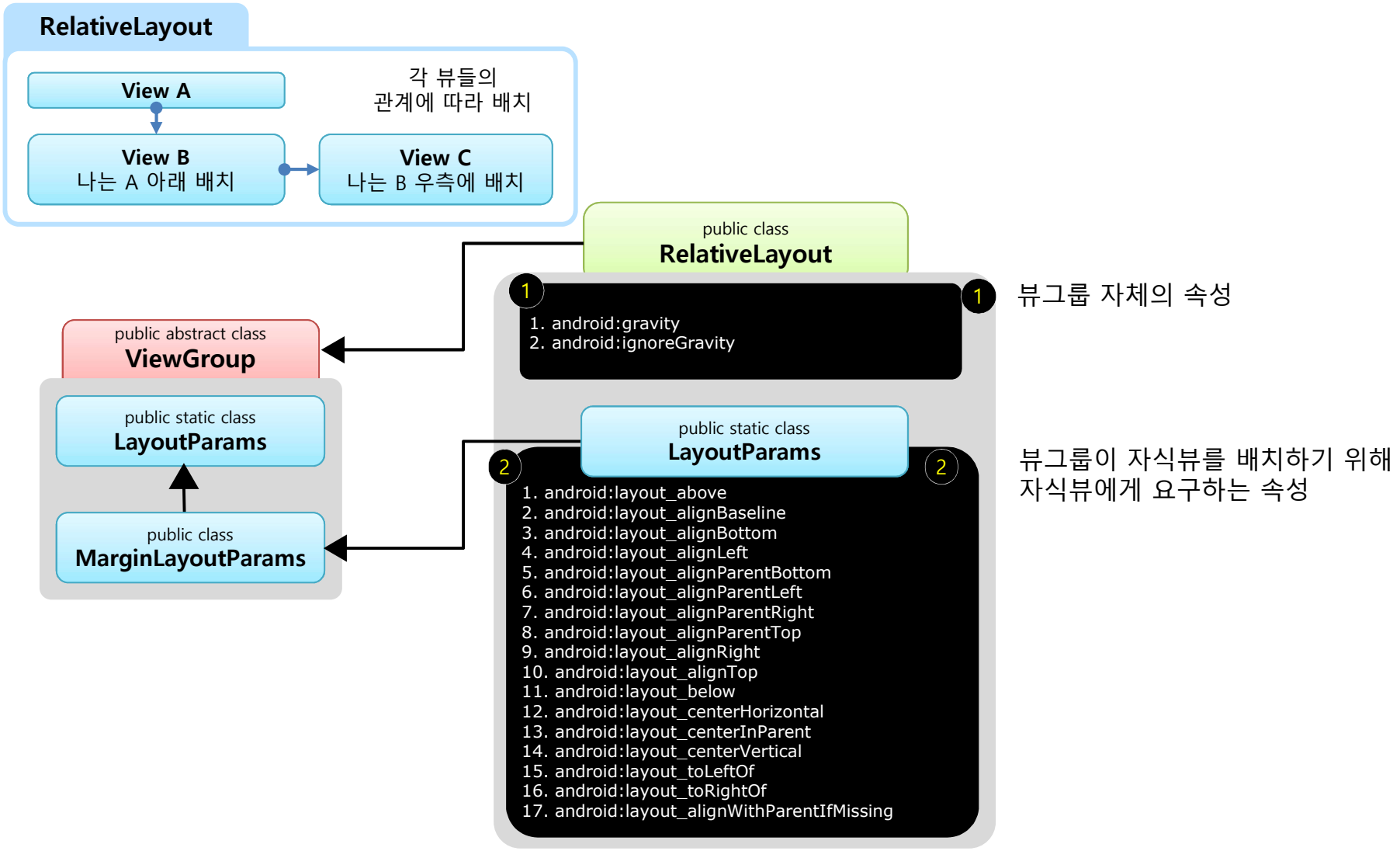
"1"이라고 설정해도 무관

res/layout/activity_main.xml

```
<LinearLayout xmlns:android="…"
    android:layout_width="match_parent"
    android:layout_height="300dp"
    android:background="#FF0"
    android:orientation="vertical"
    android:weightSum="100">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="10"
        android:text="View1"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="30"
        android:text="View2" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="50"
        android:text="View3" />

</LinearLayout>
```



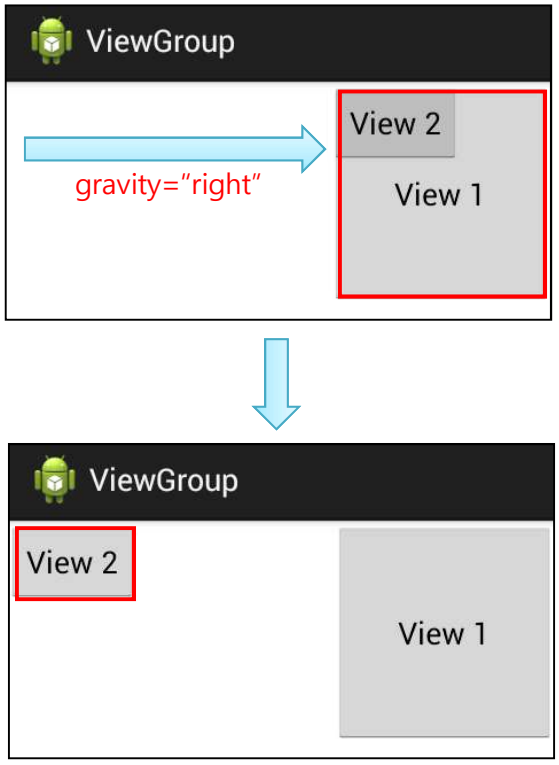
RelativeLayout과 RelativeLayout.LayoutParams



RelativeLayout의 기본 속성 – android:gravity와 android:ignoreGravity

res/layout/activity_main.xml

```
<RelativeLayout xmlns:android="…"
    android:layout_width="match_parent"
    android:layout_height="130dp"
    android:gravity="right"
    android:ignoreGravity="@id/view2">

    <Button android:layout_width="130dp"
        android:layout_height="130dp"
        android:text="View 1" />

    <Button android:id="@+id/view2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="View 2" />

</RelativeLayout>
```



android:ignoreGravity= "*@id/view2*"

- top : 상단에 배치
- bottom : 하단에 배치
- left : 좌측에 배치
- right : 우측에 배치
- center_vertical : 수직 중앙에 배치
- center_horizontal : 수평 중앙에 배치
- center : 정 중앙에 배치

RelativeLayout.LayoutParams의 속성 – 부모 뷰그룹과의 관계 배치 속성

public class
RelativeLayout

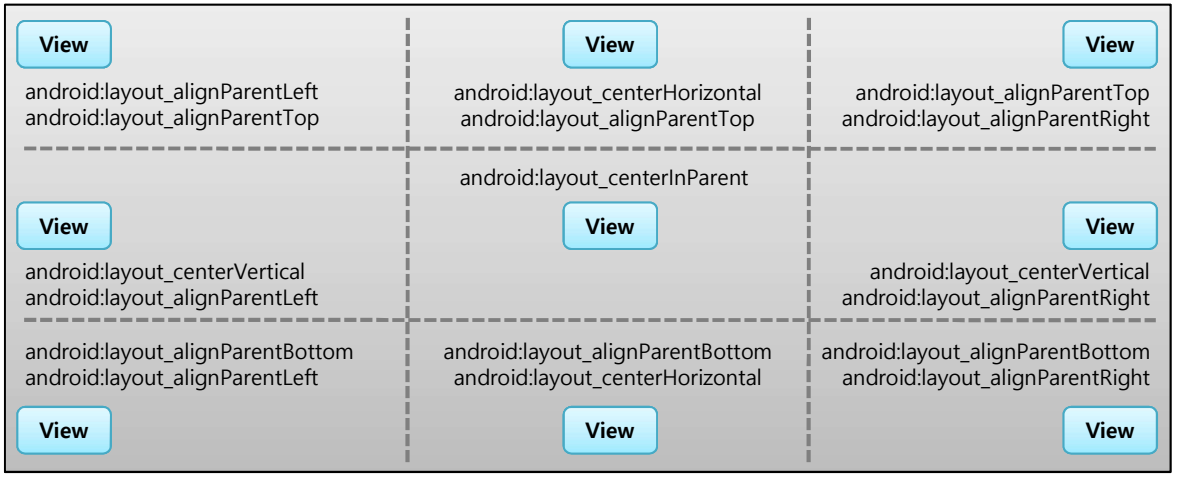
1. android:gravity
2. android:ignoreGravity

public static class
LayoutParams

1. android:layout_above
2. android:layout_alignBaseline
3. android:layout_alignBottom
4. android:layout_alignLeft
5. android:layout_alignParentBottom
6. android:layout_alignParentLeft
7. android:layout_alignParentRight
8. android:layout_alignParentTop
9. android:layout_alignRight
10. android:layout_alignTop
11. android:layout_below
12. android:layout_centerHorizontal
13. android:layout_centerInParent
14. android:layout_centerVertical
15. android:layout_toLeftOf
16. android:layout_toRightOf
17. android:layout_alignWithParentIfMissing

부모 뷰그룹과의 관계 배치 속성들은 기준 자식뷰에 대부분 설정된다.

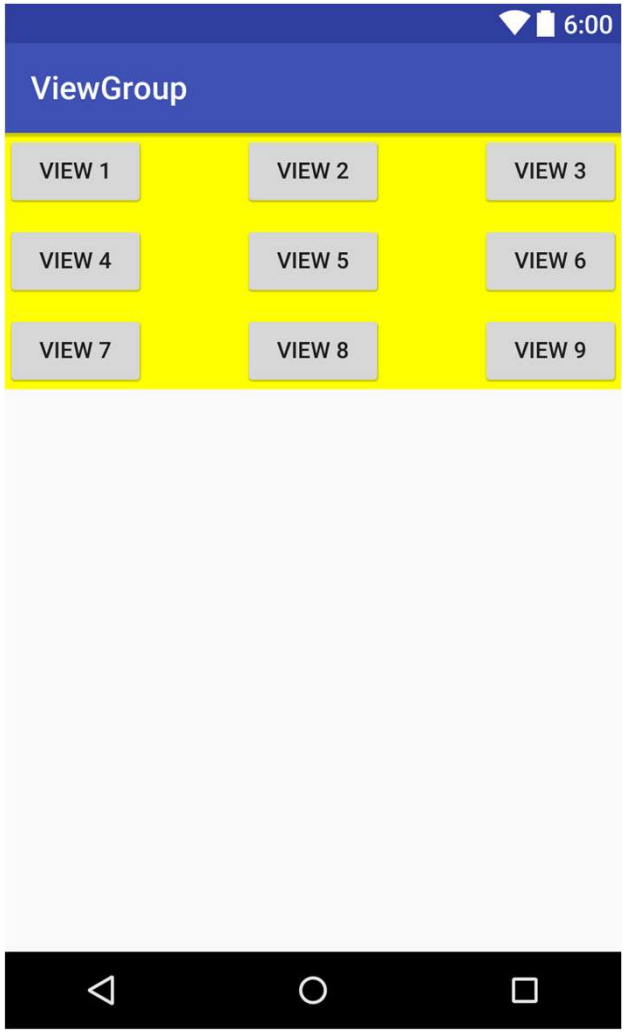
XML 속성	의미	미리 보기
android:layout_alignParentLeft	부모 영역 내 좌측에 배치	
android:layout_centerHorizontal	부모 영역 내 수평 중앙 배치	
android:layout_alignParentRight	부모 영역 내 우측에 배치	
android:layout_alignParentTop	부모 영역 내 상단 배치	
android:layout_centerVertical	부모 영역 내 수직 중앙 배치	
android:layout_alignParentBottom	부모 영역 내 하단 배치	
android:layout_centerInParent	부모 영역 내 정 중앙 배치	



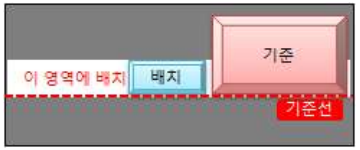
RelativeLayout.LayoutParams의 속성 – 부모 뷰그룹과의 관계 배치 속성


```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:background="#FF0"
   android:layout_height="160dp" >

   <!-- 수직 상단, 수평 좌측 -->
   <Button android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_alignParentTop="true"
      android:text="View 1" />

   <!-- 수직 상단, 수평 중앙 -->
   <Button android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_alignParentTop="true"
      android:text="View 2" />

   <!-- 수직 상단, 수평 우측 -->
   <Button android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_alignParentRight="true"
      android:text="View 3" />

   <!-- 수직 중앙, 수평 좌측 -->
   <Button android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerVertical="true"
      android:layout_alignParentLeft="true"
      android:text="View 4" />

   <!-- 수직 중앙, 수평 중앙 -->
   <Button android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerInParent="true"
      android:text="View 5" />

   <!-- 수직 중앙, 수평 우측 -->
   <Button android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerVertical="true"
      android:layout_alignParentRight="true"
      android:text="View 6" />

   <!-- 수직 하단, 수평 좌측 -->
   <Button android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"
      android:layout_alignParentLeft="true"
      android:text="View 7" />

   <!-- 수직 하단, 수평 중앙 -->
   <Button android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"
      android:layout_centerHorizontal="true"
      android:text="View 8" />

   <!-- 수직 하단, 수평 우측 -->
   <Button android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"
      android:layout_alignParentRight="true"
      android:text="View 9" />

</RelativeLayout>
```

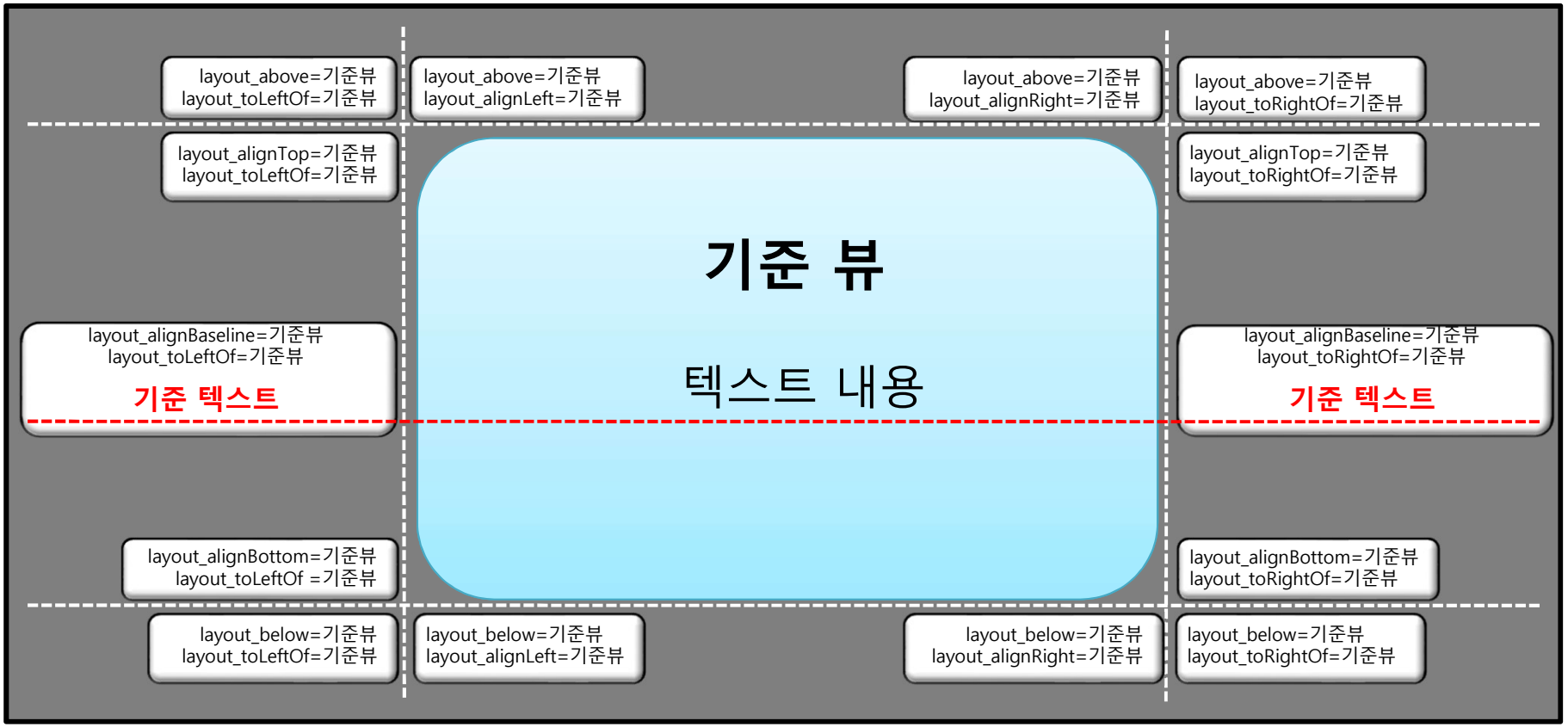
RelativeLayout.LayoutParams의 속성 – 부모 뷰그룹과의 관계 배치 속성



RelativeLayout.LayoutParams의 속성 – 자식 뷰 간의 관계 배치 속성

XML 속성	의미	미리 보기
layout_above = [기준 뷰 ID]	기준이 되는 뷰 상단 위에 배치	
layout_alignTop = [기준 뷰 ID]	기준이 되는 뷰 상단 아래쪽에 배치	
layout_alignBottom = [기준 뷰 ID]	기준이 되는 뷰 하단 위쪽에 배치	
layout_below = [기준 뷰 ID]	기준이 되는 뷰 하단 아래쪽에 배치	
layout_toLeftOf = [기준 뷰 ID]	기준이 되는 뷰 좌측 왼쪽에 배치	
layout_alignLeft = [기준 뷰 ID]	기준이 되는 뷰 좌측 오른쪽에 배치	
layout_toRightOf = [기준 뷰 ID]	기준이 되는 뷰 우측 오른쪽배치	
layout_alignRight = [기준 뷰 ID]	기준이 되는 뷰 우측 왼쪽에 배치	
layout_alignBaseline = [기준 뷰 ID]	기준이 되는 뷰에 텍스트가 존재하는 경우 배치될 뷰의 텍스트와 기준선을 맞춘다.	

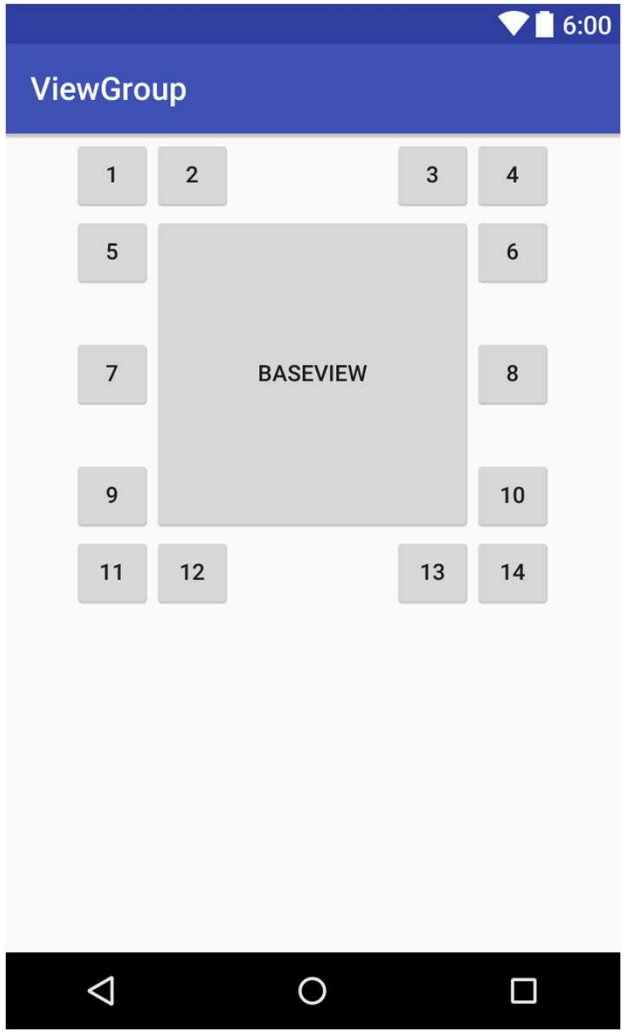
RelativeLayout.LayoutParams의 속성 – 자식 뷰 간의 관계 배치 속성



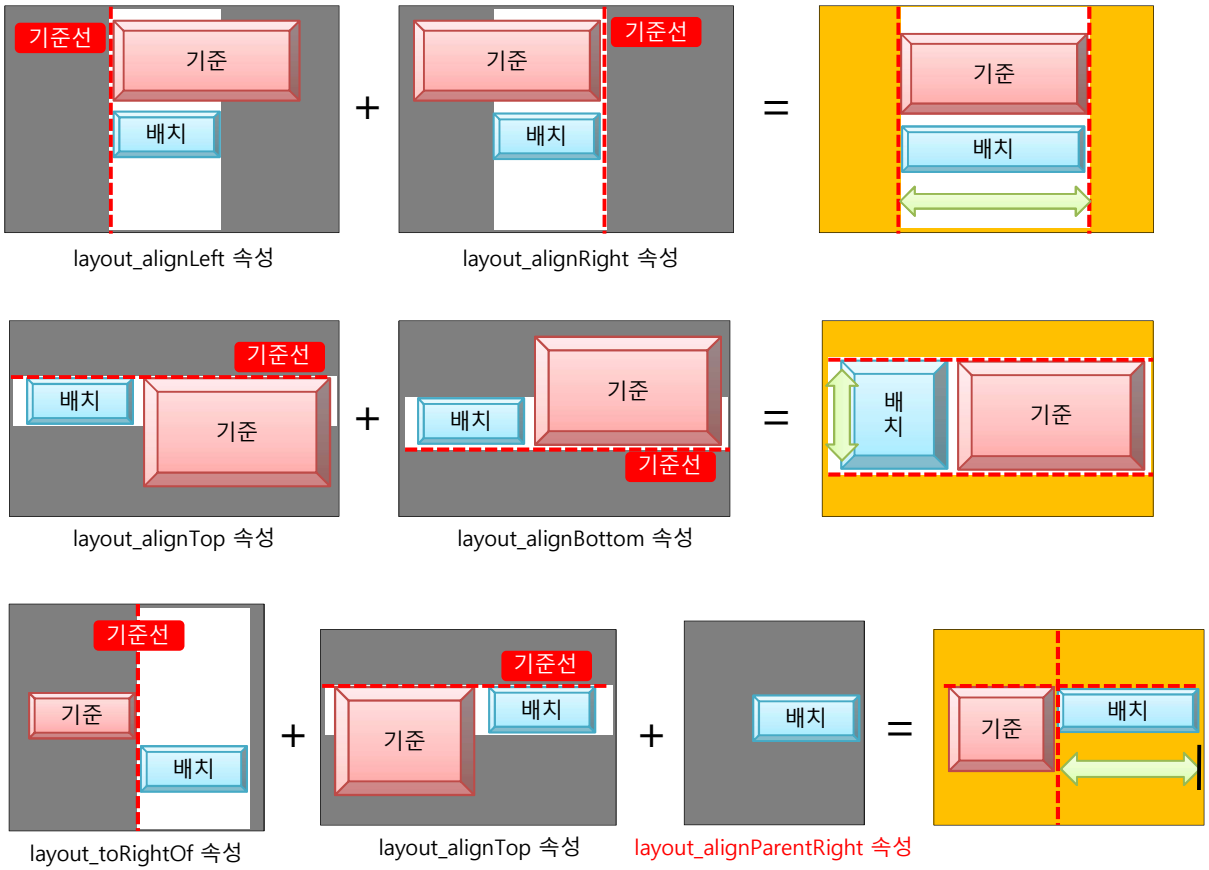
RelativeLayout.LayoutParams의 속성 – 자식 뷰 간의 관계 배치 속성


```
<?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="300dp" >

   <Button android:id="@+id/baseView"
      android:layout_width="200dp"
      android:layout_height="200dp"
      android:layout_centerInParent="true"
      android:text="BaseView" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_above="@id/baseView"
      android:layout_toLeftOf="@id/baseView"
      android:text="1" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_above="@id/baseView"
      android:layout_alignLeft="@id/baseView"
      android:text="2" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_above="@id/baseView"
      android:layout_alignRight="@id/baseView"
      android:text="3" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_above="@id/baseView"
      android:layout_toRightOf="@id/baseView"
      android:text="4" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_alignTop="@id/baseView"
      android:layout_toLeftOf="@id/baseView"
      android:text="5" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_alignTop="@id/baseView"
      android:layout_toRightOf="@id/baseView"
      android:text="6" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@id/baseView"
      android:layout_toLeftOf="@id/baseView"
      android:text="7" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@id/baseView"
      android:layout_toRightOf="@id/baseView"
      android:text="8" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_alignBottom="@id/baseView"
      android:layout_toLeftOf ="@id/baseView"
      android:text="9" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_alignBottom="@id/baseView"
      android:layout_toRightOf ="@id/baseView"
      android:text="10" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_below="@id/baseView"
      android:layout_toLeftOf="@id/baseView"
      android:text="11" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_below="@id/baseView"
      android:layout_alignLeft="@id/baseView"
      android:text="12" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_below="@id/baseView"
      android:layout_alignRight="@id/baseView"
      android:text="13" />

   <Button android:layout_width="50dp"
      android:layout_height="wrap_content"
      android:layout_below="@id/baseView"
      android:layout_toRightOf="@id/baseView"
      android:text="14" />

</RelativeLayout>
```

RelativeLayout.LayoutParams의 속성 – 자식 뷰 간의 관계 배치 속성



RelativeLayout.LayoutParams의 속성 – 레이아웃의 유연성을 지원하는 속성 조합

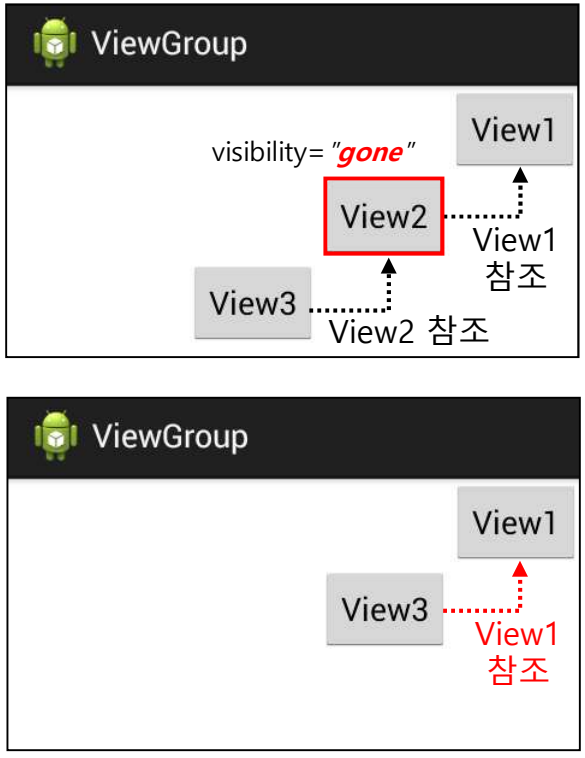


RelativeLayout.LayoutParams의 속성 – android:layout_alignWithParentIfMissing

- 연관 관계가 끊어진 경우 테스트

res/layout/activity_main.xml

```
<RelativeLayout xmlns:android="..."
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <Button android:id="@+id/view1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:text="View1"/>

    <Button android:id="@+id/view2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/view1"
        android:layout_below="@id/view1"
        android:visibility="visible"
        android:text="View2"/>

    <Button android:id="@+id/view3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/view2"
        android:layout_below="@id/view2"
        android:text="View3"/>

</RelativeLayout>
```

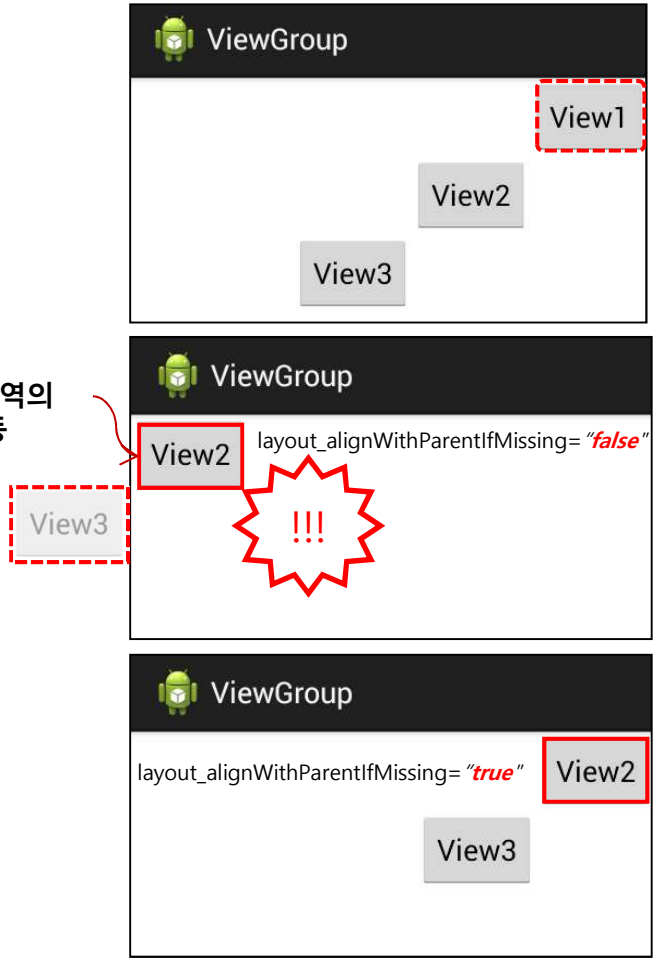


어느 하나가 사라지더라도 최대한 연결을 자동으로 유지한다.

RelativeLayout.LayoutParams의 속성 – android:layout_alignWithParentIfMissing

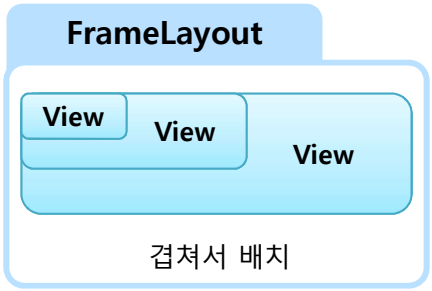

```
res/layout/activity_main.xml
<RelativeLayout xmlns:android="..."
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <Button android:id="@+id/view1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:visibility="gone"
        android:text="View1"/>

    <Button android:id="@+id/view2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/view1"
        android:layout_below="@id/view1"
        android:layout_alignWithParentIfMissing="false"
        android:text="View2"/>

    <Button android:id="@+id/view3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/view2"
        android:layout_below="@id/view2"
        android:text="View3"/>

</RelativeLayout>
```

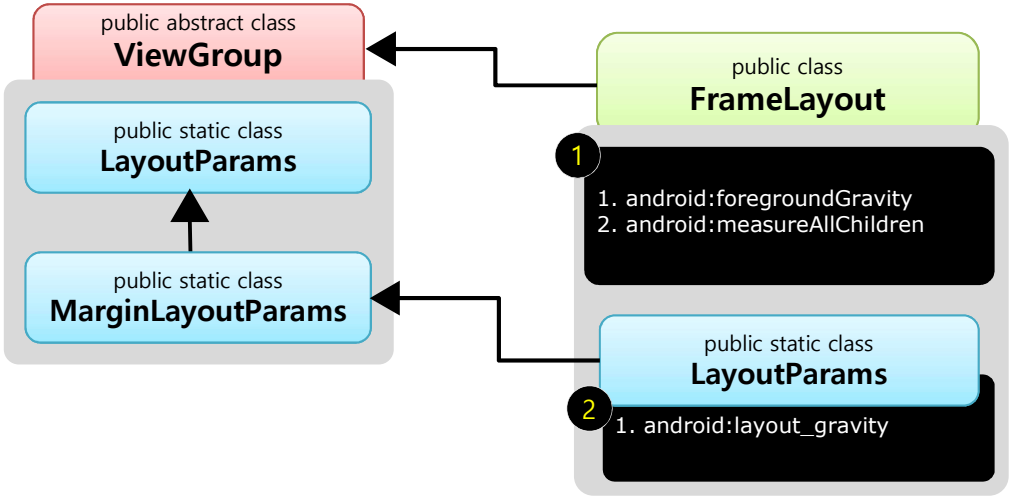
기본 위치인 RelativeLayout 영역의 좌측 상단으로 이동



참조했던 뷰의 속성을 따라간다.

FrameLayout과 FrameLayout.LayoutParams

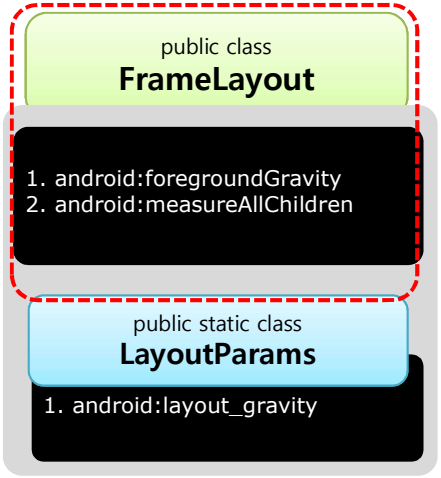




1 뷰그룹 자체의 속성

2 뷰그룹이 자식뷰를 배치하기 위해 자식뷰에게 요구하는 속성

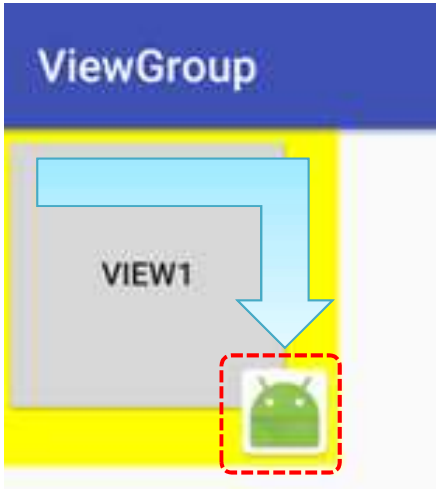
FrameLayout의 기본 속성 – android:foregroundGravity



inherited attribute from View

```
res/layout/activity_main.xml

<FrameLayout xmlns:android="..."
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:background="#FF0"
    android:foreground="@mipmap/ic_launcher"
    android:foregroundGravity="right|bottom">
    <Button
        android:layout_width="130dp"
        android:layout_height="130dp"
        android:text="View1"/>
</FrameLayout>
```

foregroundGravity 속성은
모든 자식 뷰보다 앞에 이미지를 그린다.



■ gravity 속성값

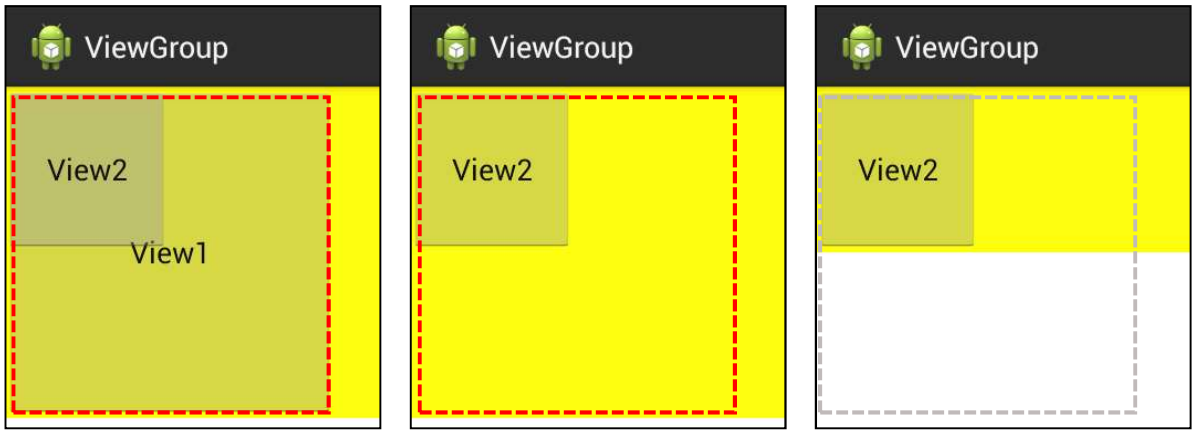
- top : 상단에 배치
- bottom : 하단에 배치
- left : 좌측에 배치
- right : 우측에 배치
- center_vertical : 수직 중앙에 배치
- center_horizontal : 수평 중앙에 배치
- center : 정 중앙에 배치

FrameLayout의 기본 속성 – android:measureAllChildren

■ **measureAllChildren 속성을 이해하기 위해서는 먼저 뷰의 기본 속성 중 하나인 visibility 속성을 알아야 한다.**

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="#FF0">

   <Button
      android:layout_width="200dp"
      android:layout_height="200dp"
      android:text="View1"
      android:visibility="visible"/>

   <Button
      android:layout_width="100dp"
      android:layout_height="100dp"
      android:text="View2"/>

</FrameLayout>
```

inherited attribute from View

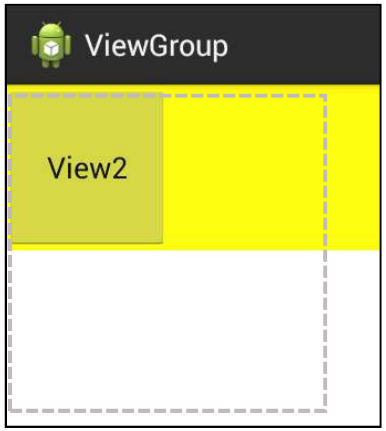


android:visibility= "***visible***" android:visibility= "***invisible***" android:visibility= "***gone***"

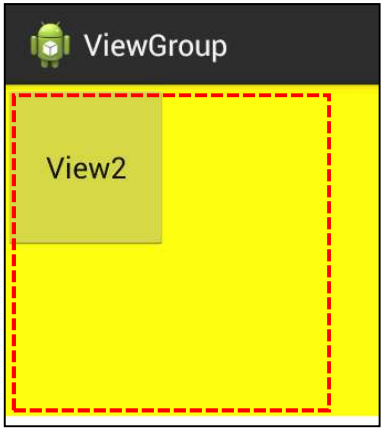
FrameLayout의 기본 속성 – android:measureAllChildren

res/layout/activity_main.xml

```
<FrameLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FF0"
    android:measureAllChildren="true">

    <Button
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:text="View1"
        android:visibility="gone"/>

    <Button
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:text="View2"/>
</FrameLayout>
```



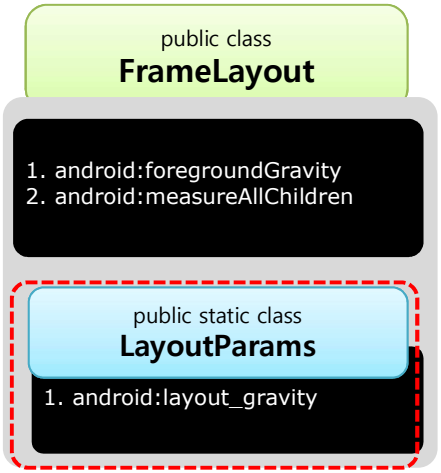
android:visibility= "***gone***"



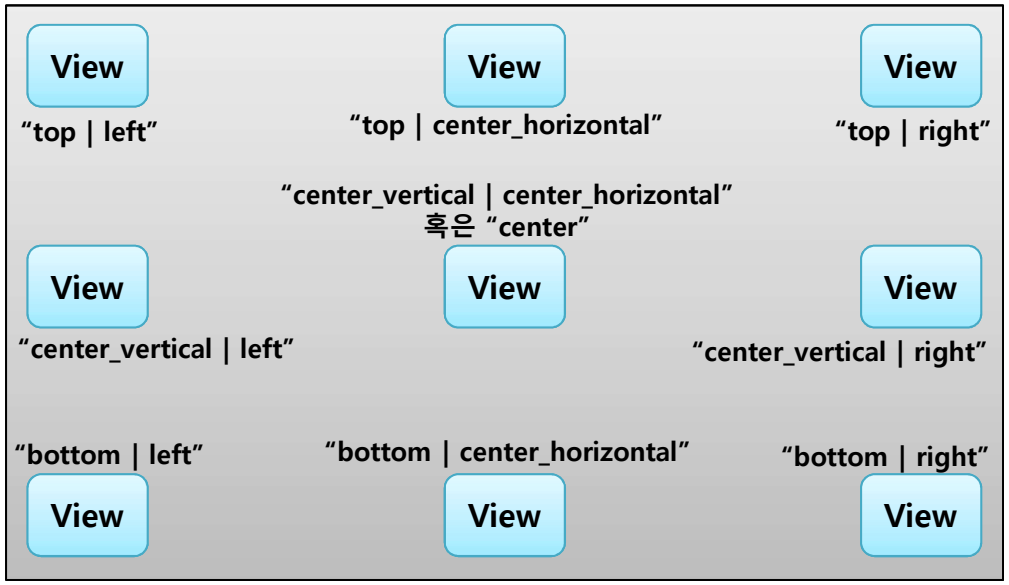
FrameLayout: android:measureAllChildren="***true***"
View1: android:visibility="***gone***"

visibility 속성이 gone으로 설정된 자식뷰라 하더라도 해당 영역을 없애지 않고 유지시킨다.

FrameLayout.LayoutParams의 속성



layout_gravity는 자식 뷰의 중력방향을 설정한다.



■ gravity 속성값

- top : 상단에 배치
- bottom : 하단에 배치
- left : 좌측에 배치
- right : 우측에 배치
- center_vertical : 수직 중앙에 배치
- center_horizontal : 수평
- center : 정 중앙에 배치

FrameLayout과 뷰의 전환 – XML 레이아웃


```
<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/button01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      android:text="Change Image"
      android:onClick="onButton1Clicked"/>
   <FrameLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent">
      <ImageView
         android:id="@+id/imageView01"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:src="@drawable/dream01"
         android:visibility="invisible"/>
      <ImageView
         android:id="@+id/imageView02"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:src="@drawable/dream02"
         android:visibility="visible"/>
   </FrameLayout>
</LinearLayout>
```

1 전환 버튼

2 화면 채우기

3 이미지 뷰 설정

4 이미지 뷰 설정

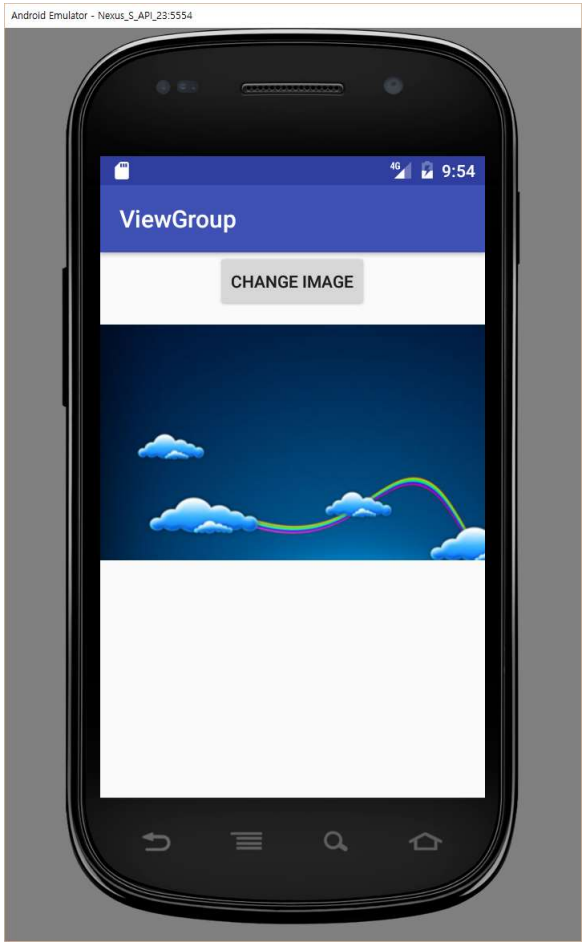
FrameLayout과 뷰의 전환 – MainActivity.java

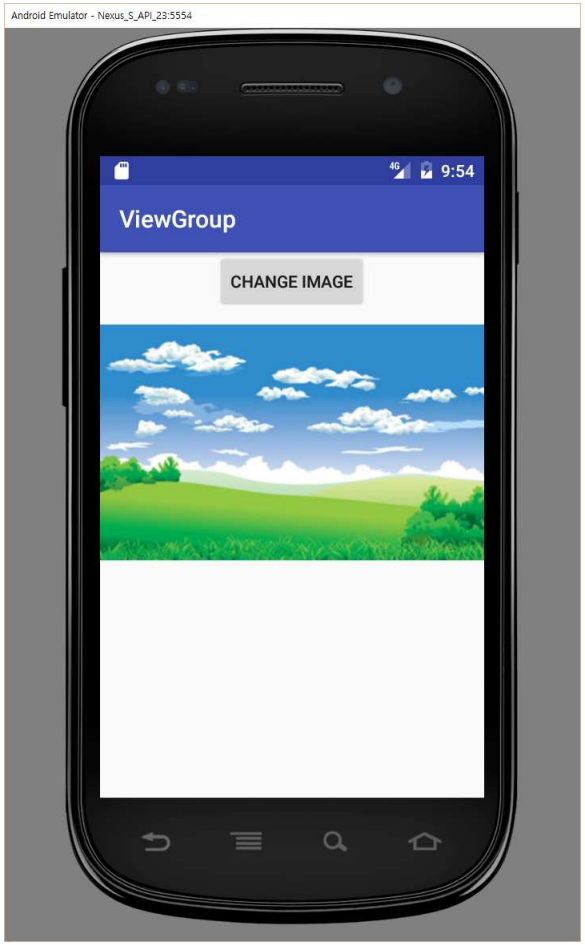
```
package kr.co.company.viewgroup;

import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import android.view.View;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

    ImageView imageView01;
    ImageView imageView02;
    int imageIndex = 0;

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

        imageView01 = (ImageView) findViewById(R.id.imageView01);

        imageView02 = (ImageView) findViewById(R.id.imageView02);

    }

    public void onButton1Clicked(View v) {
        changeImage();
    }

    private void changeImage() {
        if (imageIndex == 0) {
            imageView01.setVisibility(View.VISIBLE);
            imageView02.setVisibility(View.INVISIBLE);

            imageIndex = 1;
        } else if (imageIndex == 1) {
            imageView01.setVisibility(View.INVISIBLE);
            imageView02.setVisibility(View.VISIBLE);

            imageIndex = 0;
        }
    }
}
```

1 이미지 뷰 설정

2 이미지 뷰 설정

FrameLayout과 뷰의 전환

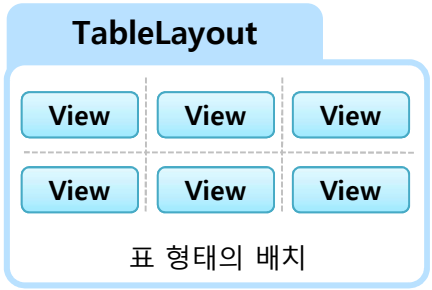


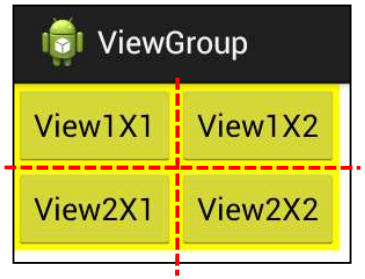


TableLayout과 TableLayout.LayoutParams, TableRow와 TableRow.LayoutParams

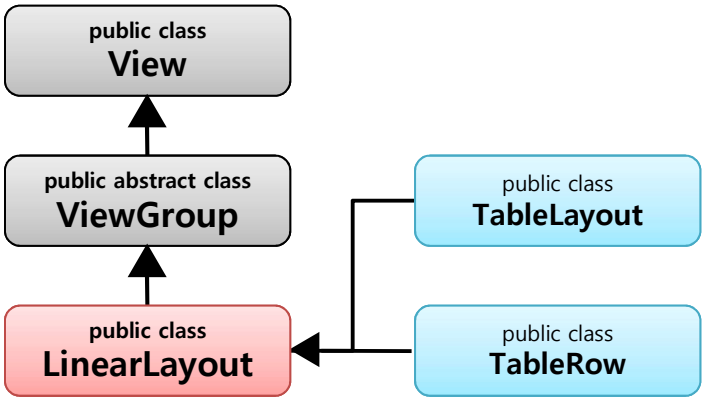
res/layout/activity_main.xml

```
<TableLayout xmlns:android="..."                    열에 해당
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#FF0">

    <TableRow>                                      행에 해당
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View1X1"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View1X2"/>
    </TableRow>

    <TableRow>                                      행에 해당
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View2X1"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View2X2"/>
    </TableRow>
</TableLayout>
```



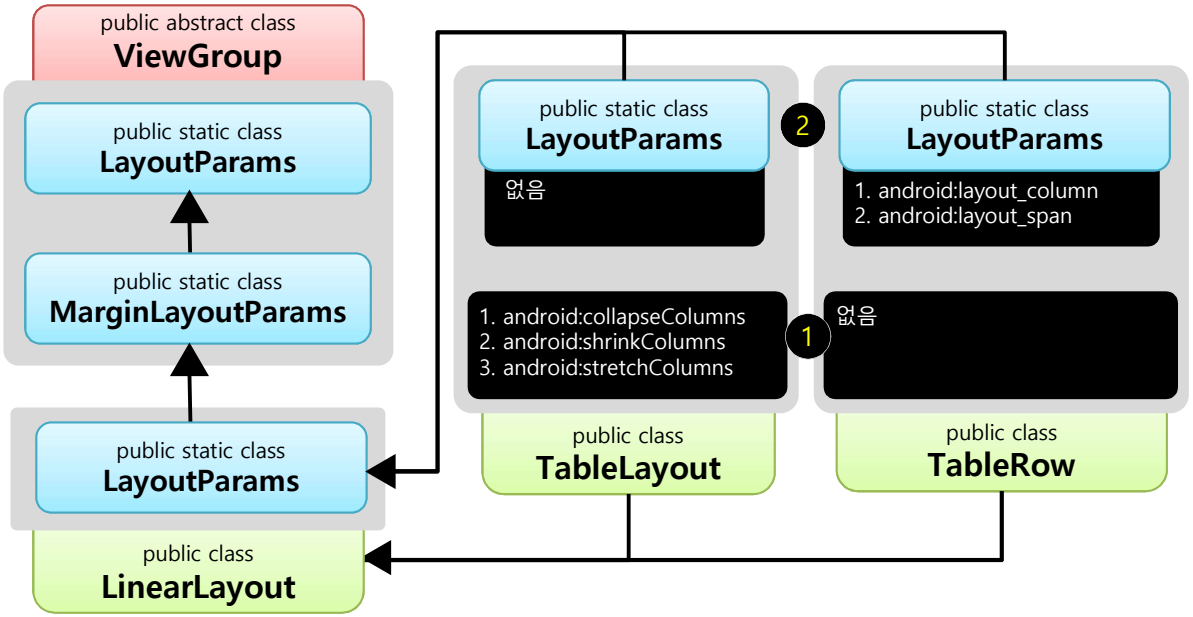


LinearLayout을 상속받은 TableLayout, TableRow



TableLayout과 TableRow는 결국 LinearLayout이다.

다만 표 형식으로 레이아웃을 구성하기 위한 편리한 속성들이 추가되었다.



1. 뷰그룹 자체의 속성
2. 뷰그룹이 자식뷰를 배치하기 위해 자식뷰에게 요구하는 속성

TableLayout의 기본 속성 – android:collapseColumns

public class
TableLayout

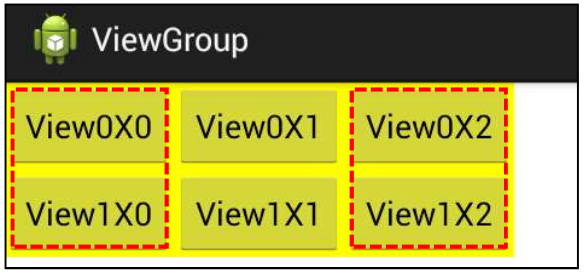
1. android:collapseColumns
2. android:shrinkColumns
3. android:stretchColumns

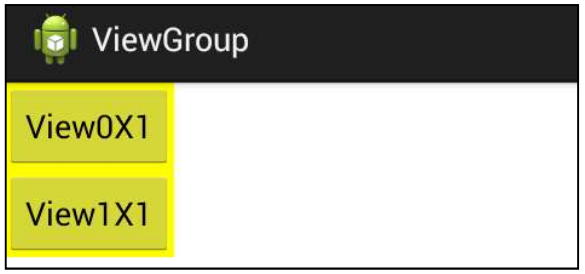
public static class
LayoutParams

없음

res/layout/activity_main.xml

```
<TableLayout xmlns:android="..."
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#FF0"
    android:collapseColumns="0,2">

    <TableRow>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View0X0"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View0X1"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View0X2"/>
    </TableRow>

    <TableRow>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View1X0"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View1X1"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View1X2"/>
    </TableRow>
</TableLayout>
```





android:collapseColumns= "0,2"

TableLayout의 기본 속성 – android:shrinkColumns

res/layout/activity_main.xml

```
<TableLayout xmlns:android="..."
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#FF0"
    android:shrinkColumns="1">

    <TableRow>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="VIEW0X0"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="VIEW0X1"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="VIEW0X2"/>
    </TableRow>

    <TableRow>

        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="VIEW1X0 ABC"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="VIEW1X1 ABC"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="VIEW1X2 ABC"/>
    </TableRow>

</TableLayout>
```

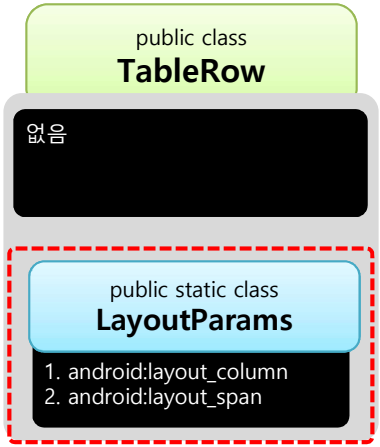
TableLayout의 기본 속성 – android:stretchColumns

res/layout/activity_main.xml

```
<TableLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FF0"
    android:stretchColumns="1">

    <TableRow>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View0X0"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View0X1"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View0X2"/>
    </TableRow>

    <TableRow>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View1X0"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View1X1"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View1X2"/>
    </TableRow>

</TableLayout>
```

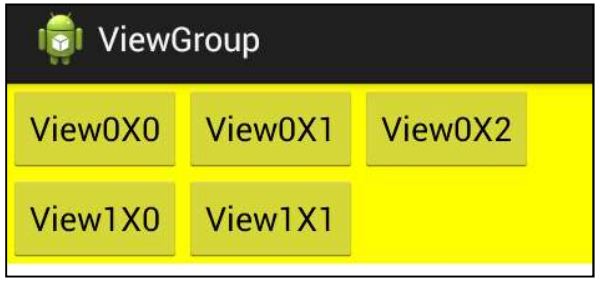
android:stretchColumns= "1"

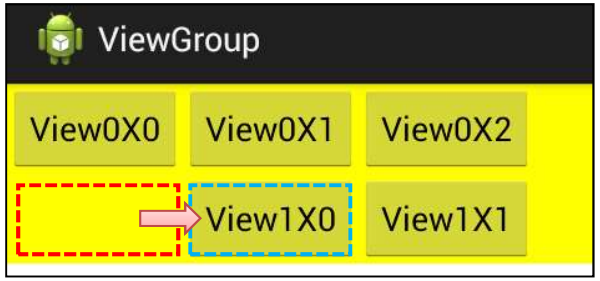
TableRow.LayoutParams의 속성 – android:layout_columns



res/layout/activity_main.xml

```
<TableLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FF0">

    <TableRow>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View0X0"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
                    android:text="View0X1"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View0X2"/>
    </TableRow>

    <TableRow>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:text="View1X0"/>

        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View1X1"/>
    </TableRow>

</TableLayout>
```



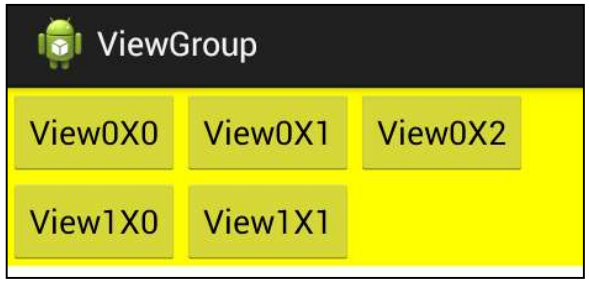


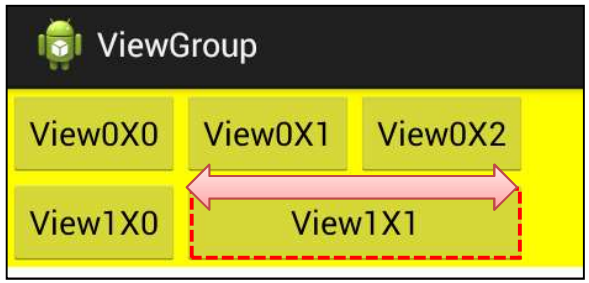
View1X0에 android:layout_column="*1*" 설정

TableRow.LayoutParams의 속성 – android:layout_span

res/layout/activity_main.xml

```
<TableLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FF0">

    <TableRow>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View0X0"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
                    android:text="View0X1"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View0X2"/>
    </TableRow>

    <TableRow>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View1X0"/>
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_span="2"
            android:text="View1X1"/>
    </TableRow>

</TableLayout>
```

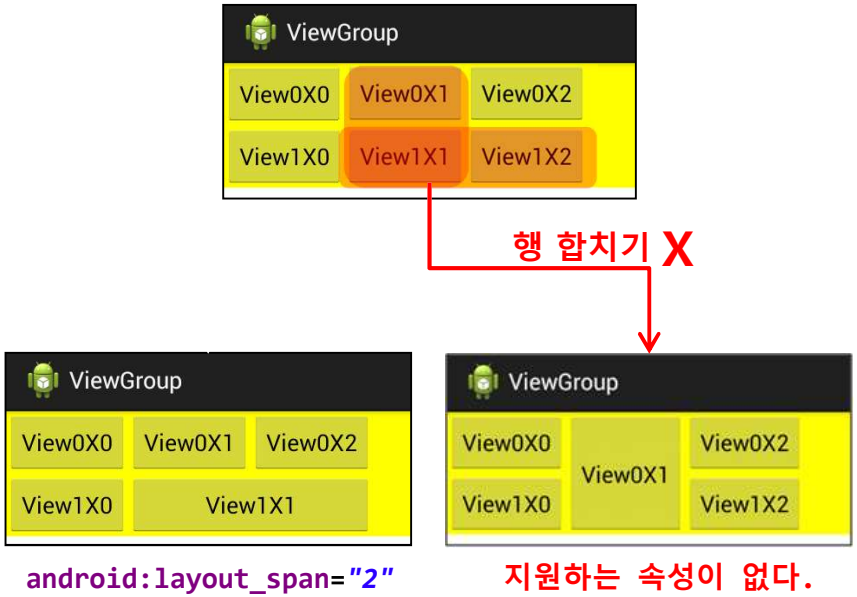




View1X1에 android:layout_span="2" 설정

TableLayout의 문제점

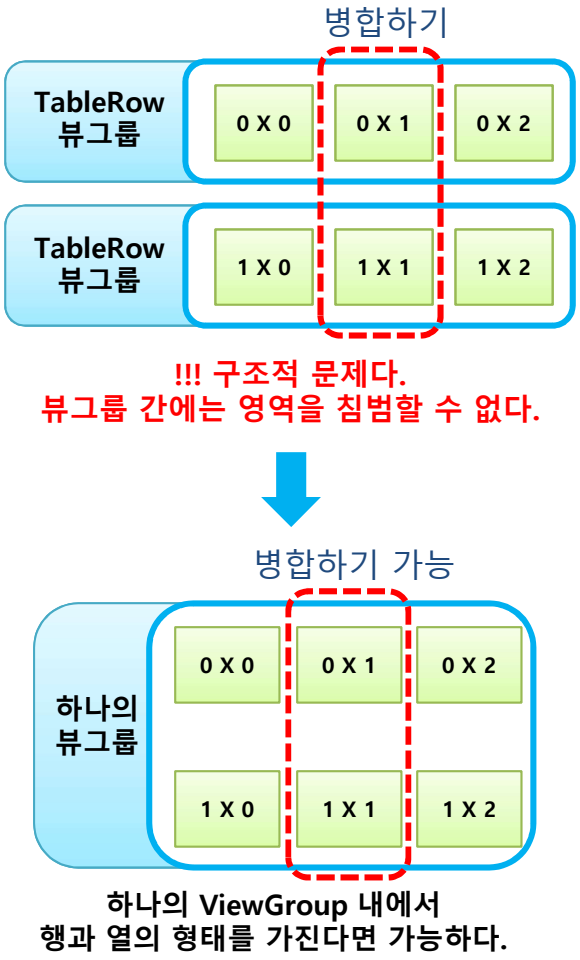
■ 행과 열의 셀 합치기



android:layout_span="2"

지원하는 속성이 없다.

■ 도대체 왜?



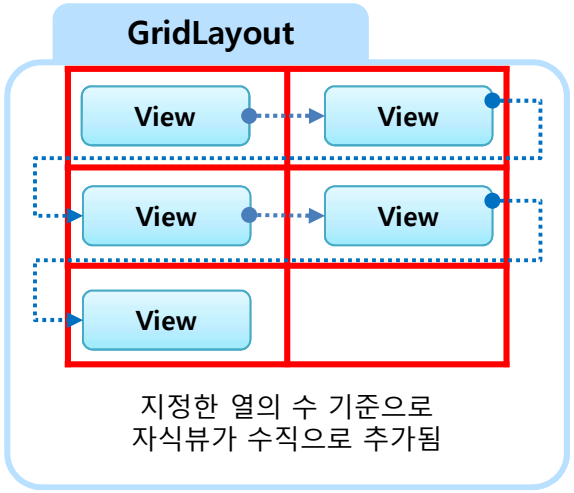
!!! 구조적 문제다.
뷰그룹 간에는 영역을 침범할 수 없다.

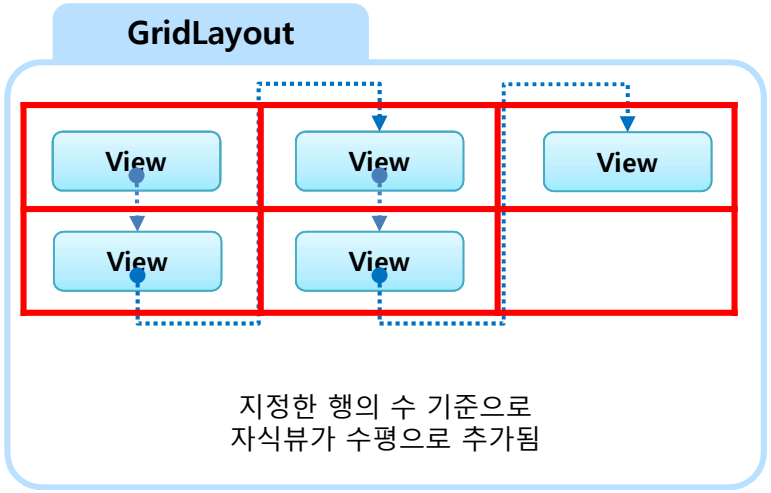
하나의 ViewGroup 내에서
행과 열의 형태를 가진다면 가능하다.

이러한 구조적 문제를 해결하기 위해 GridLayout이라는 새로운 뷰그룹이 나왔다.

GridLayout과 GridLayout.LayoutParams

GridLayout은 API 14부터 추가되었기 때문에 13 이하의 버전에서는 사용할 수 없다.
하지만 안드로이드 추가 배포 android-support-v7-gridlayout 라이브러리를 사용하면 하위버전에서도 사용할 수 있다.

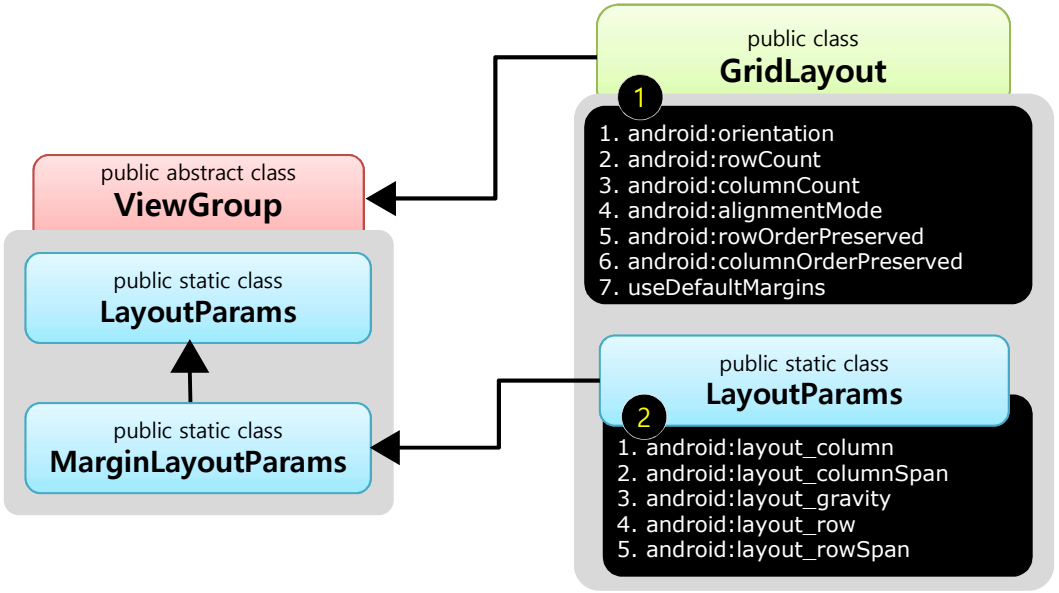




뷰가 추가되는 방향에 있어서 LinearLayout과 유사

표 형식을 취하는 것으로 보아 TableLayout과 유사

GridLayout과 GridLayout.LayoutParams



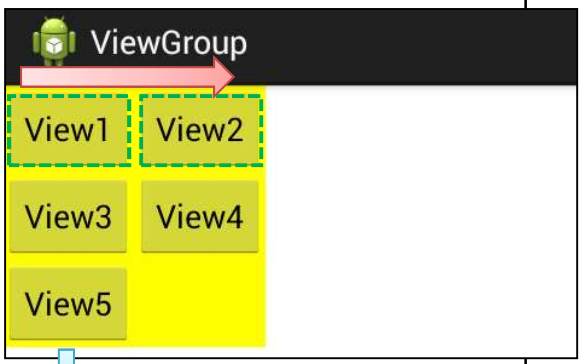
1 뷰그룹 자체의 속성

2 뷰그룹이 자식뷰를 배치하기 위해 자식뷰에게 요구하는 속성

GridLayout의 기본 속성 – android:orientation, android:rowCount, android:columnCount

res/layout/activity_main.xml

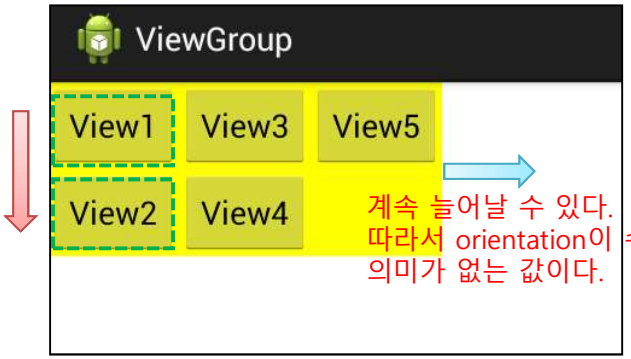
```
<GridLayout xmlns:android="..."
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#FF0"
    android:orientation="horizontal"
    android:rowCount="100"
    android:columnCount="2">

    <Button android:text="View1"/>
    <Button android:text="View2"/>
    <Button android:text="View3"/>
    <Button android:text="View4"/>
    <Button android:text="View5"/>

</GridLayout>
```



계속 늘어날 수 있다.
따라서 orientation이 수평인 경우
의미가 없는 값이다.

android:orientation= "horizontal"
android:rowCount= "100"
android:columnCount = "2"

생략하는 것이 맞다.



계속 늘어날 수 있다.
따라서 orientation이
의미가 없는 값이다.

android:orientation= "vertical"
android:rowCount= "2"
android:columnCount= "100"

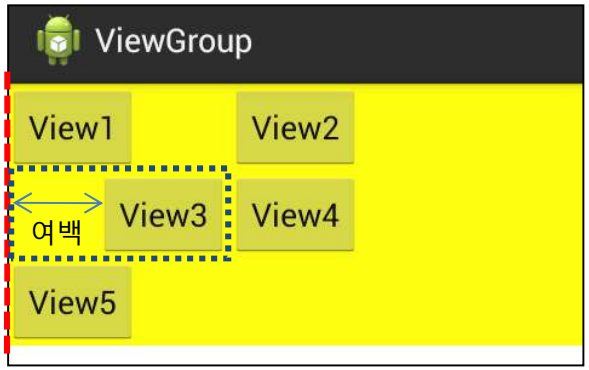
수직인 경우

GridLayout의 기본 속성 – android:alignmentMode

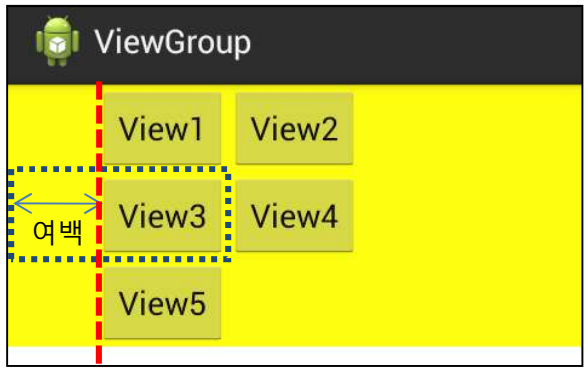
res/layout/activity_main.xml

```
<GridLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FF0"
    android:orientation="horizontal"
    android:columnCount="2"
    android:alignmentMode="alignMargins">

    <Button android:text="View1"/>
    <Button android:text="View2"/>
    <Button android:text="View3"
        android:layout_marginLeft="50dp"/>
    <Button android:text="View4"/>
    <Button android:text="View5"/>

</GridLayout>
```

너비 혹은 높이가 가장 큰 뷰의 여백(margin)을 포함해서 정렬할지 여부



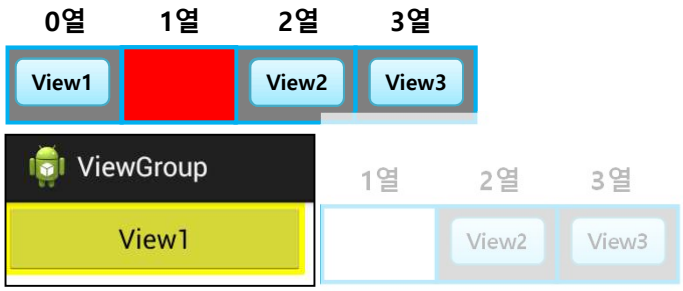
android:alignmentMode= "*alignMargins*" ← default



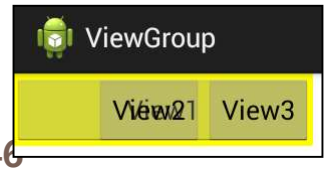
android:alignmentMode= "*alignBounds*"

GridLayout의 기본 속성 – android:columnOrderPreserved, android:rowOrderPreserved

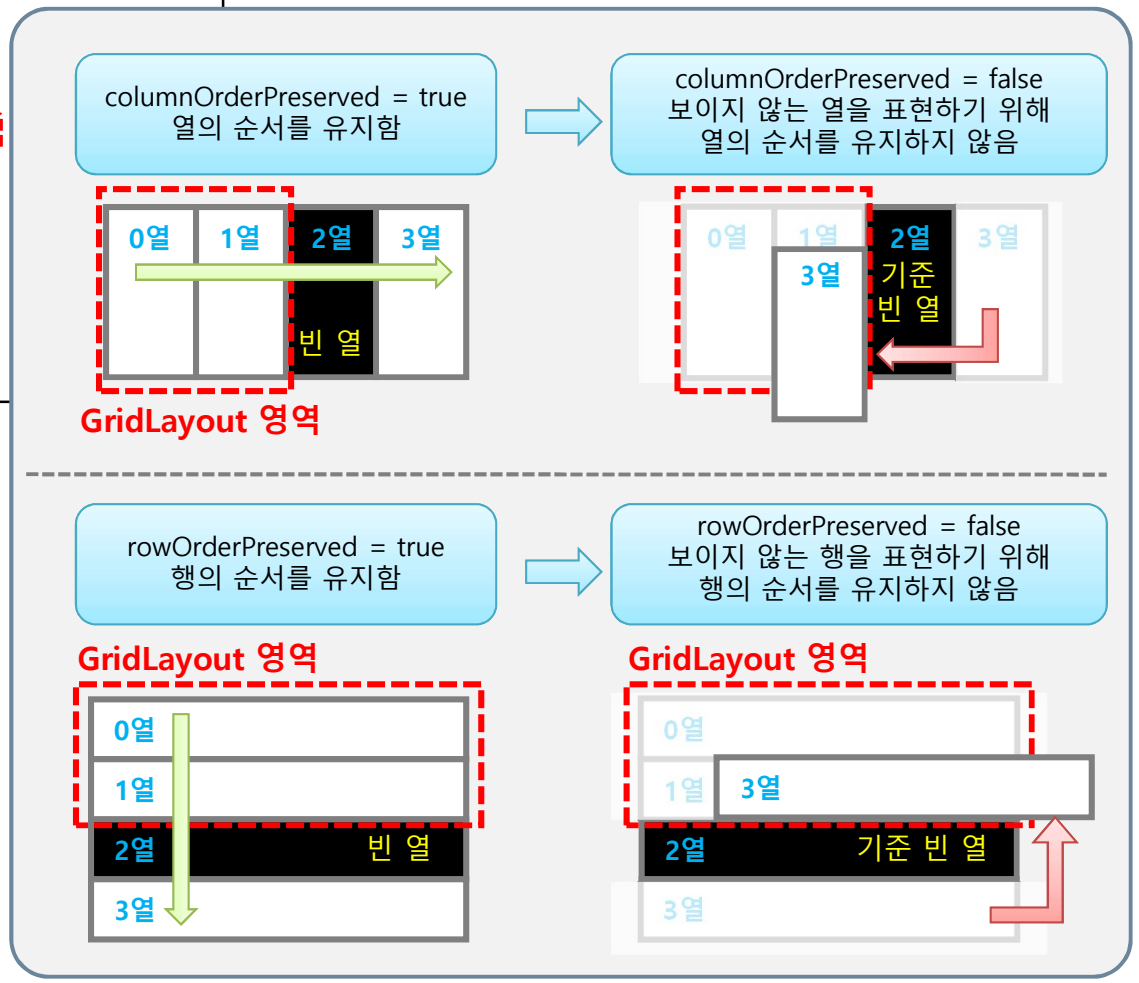
```
res/layout/activity_main.xml
<GridLayout xmlns:android="..."
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:background="#FF0"
    android:orientation="horizontal"
    android:columnCount="4"
    android:columnOrderPreserved="true">

    <Button android:text="View1"
            android:layout_width="200dp"/>
    <Button android:text="View2"
            android:layout_column="2"/>
    <Button android:text="View3"/>
</GridLayout>
```



android:columnOrderPreserved= "*true*"

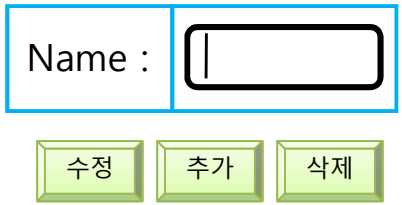


android:columnOrderPreserved= "*false*"

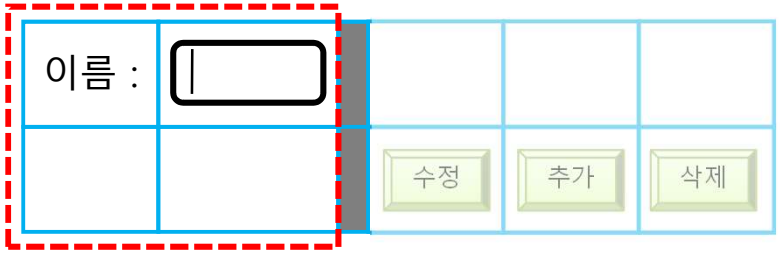


GridLayout의 기본 속성 – android:columnOrderPreserved, android:rowOrderPreserved

■ **columnOrderPreserved, rowOrderPreserved 어떻게 활용하나?**



레이아웃 구조



GridLayout 영역

GridLayout만을 사용해 다양하고 복잡한 화면을 구성할 때 필요한 속성

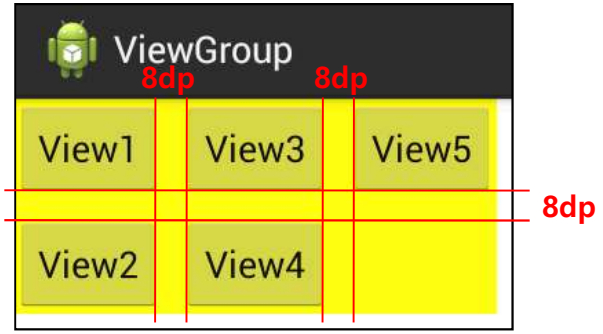
res/layout/activity_main.xml

```
<GridLayout xmlns:android="..."
    android:layout_width="280dp"
    android:layout_height="wrap_content"
    android:background="#E7E7E7"
    android:orientation="horizontal"
    android:columnCount="6"
    android:columnOrderPreserved="false">

    <TextView android:text="이름 : "
        android:layout_width="80dp"/>
    <EditText android:layout_width="200dp"/>
    <Button   android:text="수정"
              android:layout_row="1"
              android:layout_column="3"/>

    <Button   android:text="추가"/>
    <Button   android:text="삭제"/>
</GridLayout>
```

GridLayout의 기본 속성 – android:useDefaultMargins


```
res/layout/activity_main.xml
<GridLayout xmlns:android="..."
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#FF0"
    android:orientation="vertical"
    android:rowCount="2"
    android:useDefaultMargins="false">

    <Button android:text="View1"/>
    <Button android:text="View2"/>
    <Button android:text="View3"/>
    <Button android:text="View4"/>
    <Button android:text="View5"/>

</GridLayout>
```

android:useDefaultMargins= "*false*" ← default



android:useDefaultMargins= "*true*"

■ **useDefaultMargins 속성은 실용성이 없다.**

useDefaultMargins 속성으로 설정되는 여백의 크기는 사용자가 변경할 수 없다.
따라서 너무 제한적이며 유연하지 못해 실효성이 떨어진다.

GridLayout.LayoutParams의 속성 – android:layout_gravity

public class
GridLayout

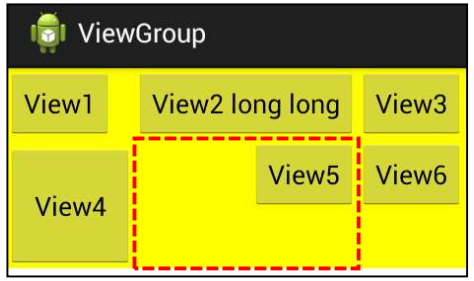
1. android:orientation
2. android:rowCount
3. android:columnCount
4. android:alignmentMode
5. android:rowOrderPreserved
6. android:columnOrderPreserved
7. android:useDefaultMargins

public static class
LayoutParams

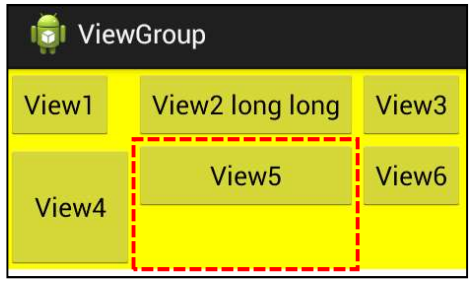
1. android:layout_column
2. android:layout_columnSpan
3. android:layout_gravity
4. android:layout_row
5. android:layout_rowSpan

res/layout/activity_main.xml

```
<GridLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FF0"
    android:orientation="horizontal"
    android:columnCount="3">

    <Button android:text="View1"/>
    <Button android:text="View2 long long"/>
    <Button android:text="View3"/>
    <Button android:text="\n View4 \n"/>
    <Button android:text="View5"
        android:layout_gravity="right"/>
    <Button android:text="View6"/>

</GridLayout>
```



layout_gravity= "***right***"

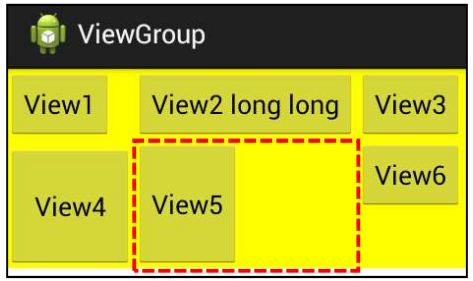


layout_gravity= "***fill_horizontal*** "

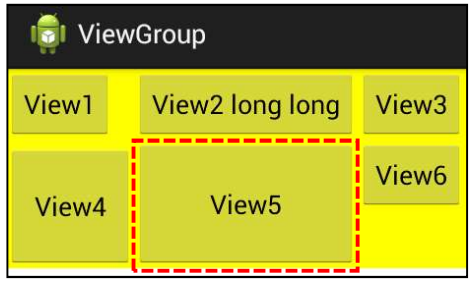
■ layout_gravity 속성값

- **top : 상단에 배치**
- **bottom : 하단에 배치**
- **left : 좌측에 배치**
- **right : 우측에 배치**
- **center_vertical : 수직 중앙에 배치**
- **center_horizontal : 수평 중앙에 배치**
- **center : 정 중앙에 배치**
- **fill_horizontal : 셀의 너비만큼 자식 뷰의 너비를 채운다. 이 값은 left|right 조합 값과 동일하다.**
- **fill_vertical : 셀의 높이만큼 자식 뷰의 높이를 채운다. 이 값은 top|bottom 조합 값과 동일하다.**
- **fill : 셀의 크기만큼 자식 뷰의 크기를 채운다. 이 값은 left|right|top|bottom 조합 값과 동일하다.**

GridLayout에서는 layout_width, layout_height 속성을 생략할 수 있음.
이 때, 생략된 해당 속성값은 "wrap_content"로 설정됨.



layout_gravity= "***fill_vertical*** "

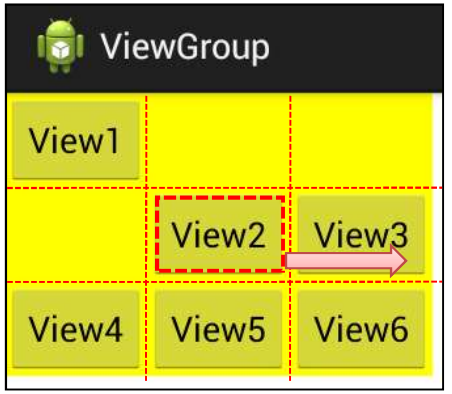


layout_gravity= "***fill***"

GridLayout.LayoutParams의 속성 – android:layout_row, android:layout_column

res/layout/activity_main.xml

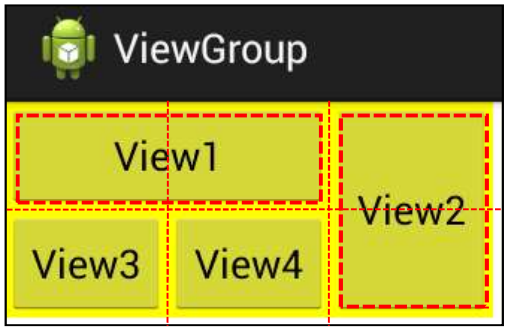
```
<GridLayout xmlns:android="..."
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#FF0"
    android:orientation="horizontal"
    android:columnCount="3">

    <Button android:text="View1"/>
    <Button android:text="View2"
       android:layout_row="1"
       android:layout_column="1"/>
    <Button android:text="View3"/>
    <Button android:text="View4"/>
    <Button android:text="View5"/>
    <Button android:text="View6"/>
</GridLayout>
```



GridLayout.LayoutParams의 속성 – android:layout_rowSpan, android:layout_columnSpan

res/layout/activity_main.xml

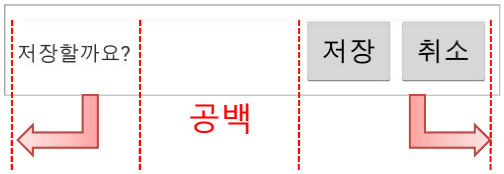
```
<GridLayout xmlns:android="..."
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#FF0"
    android:orientation="horizontal"
    android:columnCount="3">

    <Button android:text="View1"
        android:layout_columnSpan="2"
        android:layout_gravity="fill"/>

    <Button android:text="View2"
        android:layout_rowSpan="2"
        android:layout_gravity="fill"/>
    <Button android:text="View3"/>
    <Button android:text="View4"/>

</GridLayout>
```



- **GridLayout에서 자식 뷰 배치 원리**

가장 마지막에 추가된 셀의 다음 행과 열이 탐색의 시작점이 된다.

GridLayout을 위한 헬퍼 뷰 Space



GridLayout에서 각 셀에는 뷰가 존재하지 않으면 어떤 너비나 높이도 차지할 수 없다.

res/layout/activity_main.xml

```
<GridLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:columnCount="4"
    android:orientation="horizontal">

    <TextView android:text="저장할까요?"/>
    <Space android:layout_gravity="fill_horizontal"/>
    <Button android:text="저장"/>
    <Button android:text="취소"/>

</GridLayout>
```

레이아웃 구성

- 일반적 방법
  1. 전체적인 레이아웃 분리
     - LinearLayout
  2. 분리된 영역 내에 각 뷰들을 상세히 배치
     - RelativeLayout

- 전체 화면을 차지하는 여러 가지 레이아웃을 전환하여 보여줄 때 주로 사용
  - FrameLayout

- 테이블 형태의 뷰 배치
  - GridLayout과 TableLayout