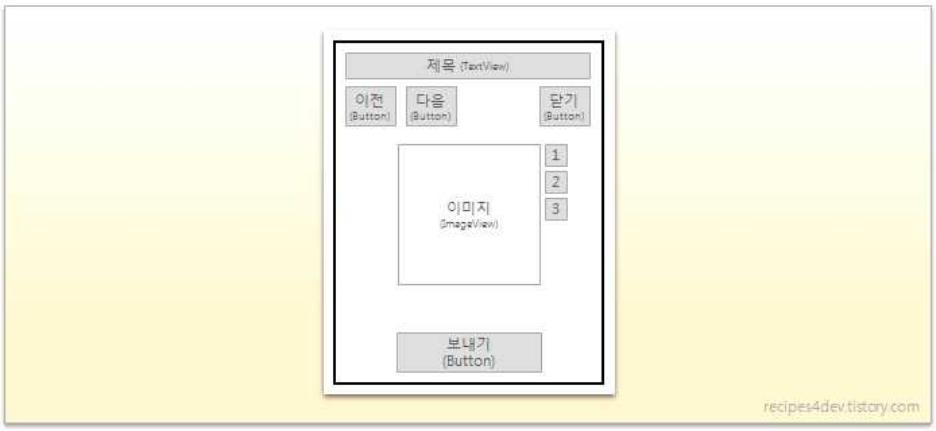
안드로이드 RelativeLayout 클래스.

[안드로이드 리니어레이아웃. (Android LinearLayout)]에서 설명한 LinearLayout을 사용하여 UI 레이아웃을 구성하다보면, 원하는 배치 구조를 만들기가 쉽지 않은 경우가 있습니다. 아래 그림의 예를 보시죠.



그림과 같은 레이아웃을 만들기 위해 LinearLayout을 사용한다고 가정하면, 어떤 식으로 구성해야 할지, 그 방법이 쉽게 떠오르지 않습니다. 그래도 한번 해볼까요?

- 루트에 vertical로 된 LinearLayout을 하나 두고, 가장 위엔 TextView(제목) 배치.
- 그리고 그 다음엔 horizontal로 정렬되는 LinearLayout을 하나 추가.
- LinearLayout 안에 LinearLayout을 두 개 추가.
- 왼쪽 LinearLayout에는 두 개의 Button(이전, 다음)을 추가
- 오른쪽 LinearLayout에는 한 개의 Button(닫기)을 추가. gravity를 end로 설정.
- 그 다음 LinearLayout을 또 하나 추가한 다음 horizontal로 정렬
- 그 안에 ImageView(이미지)와 또 하나의 LinearLayout을 추가.

음, 복잡하네요. 그리고 몇 가지 적용하기 애매한 배치도 있구요. 좋은 방법이 아닌 듯 합니다.

그럼 이제, 머리속에서, LinearLayout을 지우고 다시한번 고민해보죠. 접근 방법은, 가로 또는 세로 방향으로 뷰(View)를 정렬하는 것이 아니라, 각 **요소들 간의 상대적인 위치**를 따져보는 것입니다.

- TextView(제목)를 화면의 상단에 배치.
- TextView(제목) 아래에 Button(이전) 추가.
- Button(이전) 오른쪽에 Button(다음) 추가.
- TextView(제목) 아래, Button(닫기) 추가하고 화면의 오른쪽에 정렬
- ImageView(이미지)를 추가하고 화면의 가운데 배치.
- Button(1)을 ImageView(이미지)의 오른쪽에 추가.
- Button(2)을 Button(1)의 아래에 추가.
- Button(3)을 Button(2)의 아래에 추가.

- Button(보내기)을 추가하여 화면의 아래에 배치하고 가로 방향 중앙에 배치.

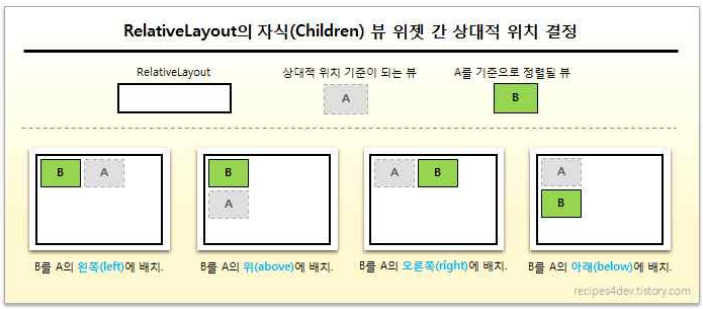
자, 어떤가요? LinearLayout만 사용해서 자식(Children) 뷰 위젯을 배치할 때와 비교해서, 화면 구성 요소들 간의 상대적 위치 관계에 따라 자식(Children) 뷰 위젯을 배치하는 방식이, 다이나믹한 화면을 구성하기에 조금 더 직관적이라는 느낌이 들지 않나요?

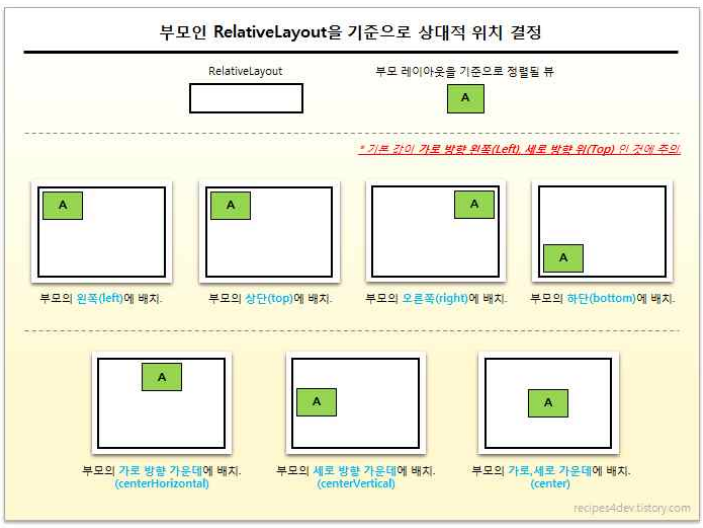
바로 이렇게, **레이아웃 내의 자식(Children) 뷰 위젯들이 서로 간의 상대적(Relative) 위치 관계에 따라 최종적으로 표시될 영역을 결정하도록 만드는 레이아웃을, RelativeLayout(렐러티브 레이아웃)** 이라고 합니다.

2. RelativeLayout 자식(Children) 뷰 위젯의 배치 기준.

RelativeLayout에 자식(Children) 뷰 위젯들이 배치될 때 서로 간의 상대적(Relative) 위치를 따진다고 했지만, 상대적 위치의 대상으로 반드시 자식(Children) 뷰 위젯만 지정할 수 있는 것은 아닙니다. 부모(Parent) 역할을 수행하는 RelativeLayout 또한 그 대상으로 사용될 수도 있습니다.

아래 그림을 보면, 자식(Children) 뷰 위젯 간 상대적(relative) 위치 설정과, 부모(Parent)인 RelativeLayout에 대한 상대적 위치 설정의 차이를 확인할 수 있습니다.



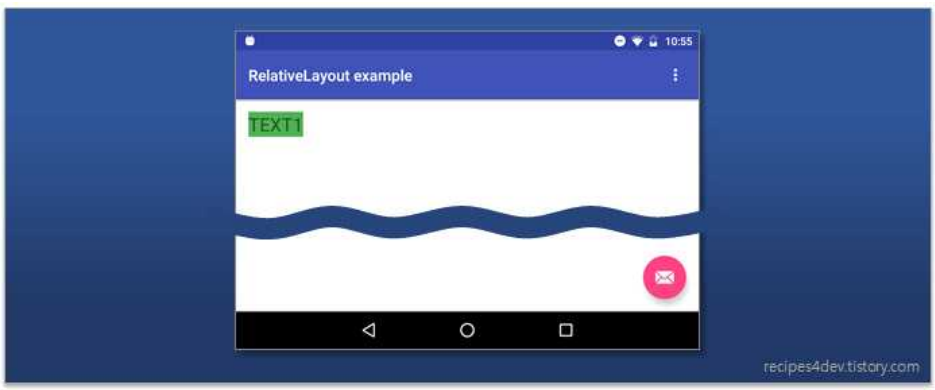


3. RelativeLayout 사용의 기본 원칙.

3.1 배치 속성 사용.

RelativeLayout에 배치 관련 속성을 지정하지 않은 뷰(View) 위젯을 추가하면, 추가된 자식(Children) 뷰(View) 위젯은 RelativeLayout의 왼쪽 위(Left, Top)에 표시됩니다.

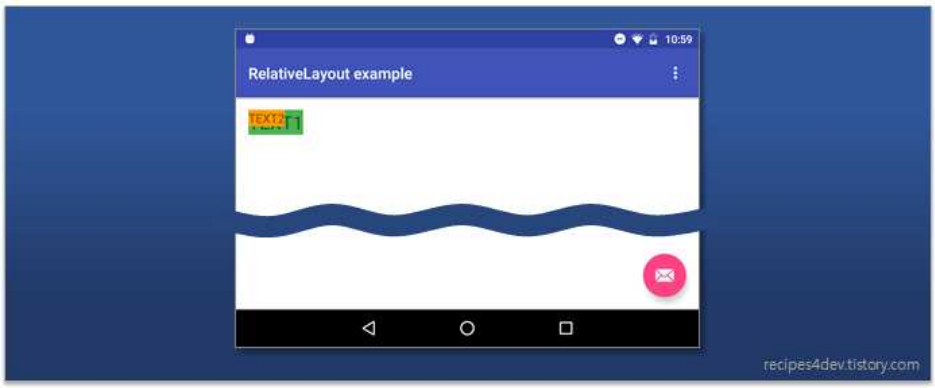
```
<RelativeLayout
    android :layout_width="match_parent "
    android :layout_height="match_parent "
    android :layout_margin="16dp ">

    <TextView
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :textSize="24sp "
        android :background="#4CAF50 "
        android :id="@+id/text1 "
        android :text="TEXT1 " />

</RelativeLayout>
```



그런데 만약 배치 관련 속성을 지정하지 않은 뷰(View) 위젯을 하나 더 추가하면, 나중에 추가된 뷰(View) 위젯이 앞서 추가한 뷰(View) 위젯 위에 오버랩(overlap)되어 표시됩니다.

```
<RelativeLayout
    android :layout_width="match_parent "
    android :layout_height="match_parent "
    android :layout_margin="16dp ">

    <TextView
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
```

```
        android :textSize="24sp "
        android :background="#4CAF50 "
        android :id="@+id/text1 "
        android :text="TEXT1 " />

    <TextView
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :textSize="16sp "
        android :background="#FF9800 "
        android :id="@+id/text2 "
        android :text="TEXT2 " />

</RelativeLayout>
```



그러므로 **RelativeLayout에 뷰(View) 위젯을 배치할 때 의도한 결과를 얻기 위해서는 반드시 자식(Children) 뷰 위젯에 배치 기준을 위한 속성을 하나 이상 사용**해야 합니다.

3.2 RelativeLayout에서 사용 가능한 속성의 종류.

RelativeLayout의 자식(Children) 뷰 위젯에 사용할 수 있는 속성은 아래와 같은 것들이 있습니다.

속성	설명
layout_toLeftOf	뷰(View)를 기준 뷰(Anchor View)의 왼쪽(Left)에 배치.
layout_above	뷰(View)를 기준 뷰(Anchor View)의 위(Above)에 배치.
layout_toRightOf	뷰(View)를 기준 뷰(Anchor View)의 오른쪽(Right)에 배치.
layout_below	뷰(View)를 기준 뷰(Anchor View)의 아래(Below)에 배치.
layout_toStartOf	뷰(View)를 기준 뷰(Anchor View)의 시작(Start)에 배치.
layout_toEndOf	뷰(View)를 기준 뷰(Anchor View)의 끝(End)에 배치.
layout_alignParentLeft	뷰(View)를 부모(Parent) 영역 내에서 왼쪽(Left)에 배치.
layout_alignParentTop	뷰(View)를 부모(Parent) 영역 내에서 위쪽(Top)에 배치.
layout_alignParentRight	뷰(View)를 부모(Parent) 영역 내에서 오른쪽(Right)에 배치.

layout_alignParentBottom	뷰(View)를 부모(Parent) 영역 내에서 아래쪽(Bottom)에 배치.
layout_centerHorizontal	뷰(View)를 부모(Parent) 영역의 가로 방향 가운데 배치.
layout_centerVertical	뷰(View)를 부모(Parent) 영역의 세로 방향 가운데 배치.
layout_centerInParent	뷰(View)를 부모(Parent) 영역의 정 중앙(center)에 배치.
layout_alignParentStart	뷰(View)를 부모(Parent) 영역의 시작 지점(Start)에 배치.
layout_alignParentEnd	뷰(View)를 부모(Parent) 영역의 끝 지점(End)에 배치.
layout_alignLeft	뷰(View)의 왼쪽(Left)을 기준 뷰(View)의 왼쪽(Left)에 맞춤.
layout_alignTop	뷰(View)의 위(Top)를 기준 뷰(View)의 위(Top)에 맞춤.
layout_alignRight	뷰(View)의 오른쪽(Right)를 기준 뷰(View)의 오른쪽(Right)에 맞춤.
layout_alignBottom	뷰(View)의 아래(Bottom)를 기준 뷰(View)의 아래(Bottom)에 맞춤.
layout_alignBaseline	뷰(View)의 폰트 기준선(Baseline)을 기준 뷰(View)의 폰트 기준선(Baseline)에 맞춤.

4. RelativeLayout 사용법.

RelativeLayout의 자식(Children) 뷰 위젯에 사용할 수 있는 배치 관련 속성은, 앞서 설명했듯이, 배치 기준을 어디로 설정하느냐에 따라, **RelativeLayout에 포함된 자식(Children) 뷰(View) 위젯 간 상대적 위치를 지정하는 속성과, 부모(Parent)인 RelativeLayout을 기준으로 상대적 위치를 지정하는 속성**으로 나뉩니다.

4.1 RelativeLayout에 포함된 자식(Children) 뷰(View) 위젯 간 상대적 위치 결정.

RelativeLayout 내에 포함된 다른 자식(Children) 뷰 위젯, 즉, 형제(Sibling)를 기준으로 상대적 위치를 지정할 때는 아래와 같은 속성을 사용합니다.

*** android:layout_toLeftOf - 기준 뷰(View)의 왼쪽(Left)에 배치.**

> 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
> 뷰(View) Right = 기준 뷰(Anchor View) Left

*** android:layout_above - 기준 뷰(View)의 위(Above)에 배치.**

> 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
> 뷰(View) Bottom = 기준 뷰(Anchor View) Top

*** android:layout_toRightOf - 기준 뷰(View)의 오른쪽(Right)에 배치.**

> 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
> 뷰(View) Left = 기준 뷰(Anchor View) Right

*** android:layout_below - 기준 뷰(View)의 아래(Below)에 배치.**

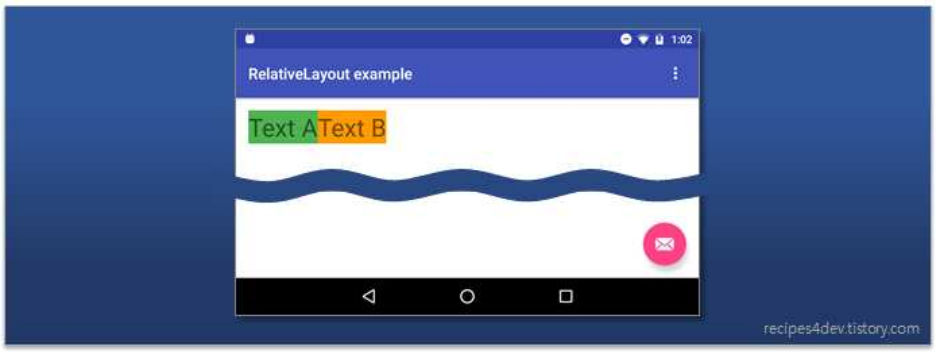
> 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
> 뷰(View) Top = 기준 뷰(Anchor View) Bottom

* android:layout_toStartOf - 기준 뷰(View)의 시작(Start)에 배치.
    > 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
    > 뷰(View) End = 기준 뷰(Anchor View) Start

* android:layout_toEndOf - 기준 뷰(View)의 끝(End)에 배치.
    > 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
    > 뷰(View) Start = 기준 뷰(Anchor View) End

아래 예제는 형제(Sibling) 뷰(View)를 기준으로 상대적 위치를 지정하는 예제입니다. "Text B"(id:textB)를 화면에 표시할 때 "Text A"(id:textB)를 기준으로 오른쪽(Right)에 배치되도록, "Text B"(id:textB)의 toRightOf 속성에 "Text A"의 ID인 "@id/textA"를 지정하였습니다.

```
<RelativeLayout
    android :layout_width="match_parent "
    android :layout_height="match_parent ">

    <TextView
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :background="#4CAF50 "
        android :textSize="32sp "
        android :id="@+id/textA "
        android :text="Text A " />

    <TextView
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :background="#FF9800 "
        android :textSize="32sp "
        android :id="@+id/textB "
        android :text="Text B "
        android :layout_toRightOf="@id/textA " />

</RelativeLayout>
```



아래 그림은 각 속성 사용에 따른 출력 결과를 정리한 것입니다.

4.2 부모(Parent)인 RelativeLayout 기준으로 상대적 위치 결정.

RelativeLayout 내에 포함된 자식(Children) 뷰 위젯이 자신의 부모(Parent), 즉, 자신이 포함된 RelativeLayout을 기준으로 상대적(relative) 위치를 지정할 때는 아래와 같은 속성을 사용합니다.

* android:layout_alignParentLeft - 부모(Parent) 영역 내에서 왼쪽(Left)에 배치.
    > 속성 값에는 true 또는 false 사용.
    > 뷰(View) Left = 부모(Parent) Left.

* android:layout_alignParentTop - 부모(Parent) 영역 내에서 위쪽(Top)에 배치.

> 속성 값에는 true 또는 false 사용.
> 뷰(View) Top = 부모(Parent) Top.

* android:layout_alignParentRight - 부모(Parent) 영역 내에서 오른쪽(Right)에 배치.
> 속성 값에는 true 또는 false 사용.
> 뷰(View) Right = 부모(Parent) Right.

* android:layout_alignParentBottom - 부모(Parent) 영역 내에서 아래쪽(Bottom)에 배치.
> 속성 값에는 true 또는 false 사용.
> 뷰(View) Bottom = 부모(Parent) Bottom.

* android:layout_centerHorizontal
- 부모(Parent) 영역의 가로(horizontal) 방향 가운데(center) 배치.
> 속성 값에는 true 또는 false 사용.

* android:layout_centerVertical - 부모(Parent) 영역의 세로(vertical) 방향 가운데(center) 배치.
> 속성 값에는 true 또는 false 사용.

* android:layout_centerInParent - 부모(Parent) 영역의 정 중앙(center)에 배치.
> 속성 값에는 true 또는 false 사용.

* android:layout_alignParentStart - 부모(Parent) 영역 내에서 시작 지점(Start)에 배치.
> 속성 값에는 true 또는 false 사용.
> 뷰(View) Start = 부모(Parent) Start.

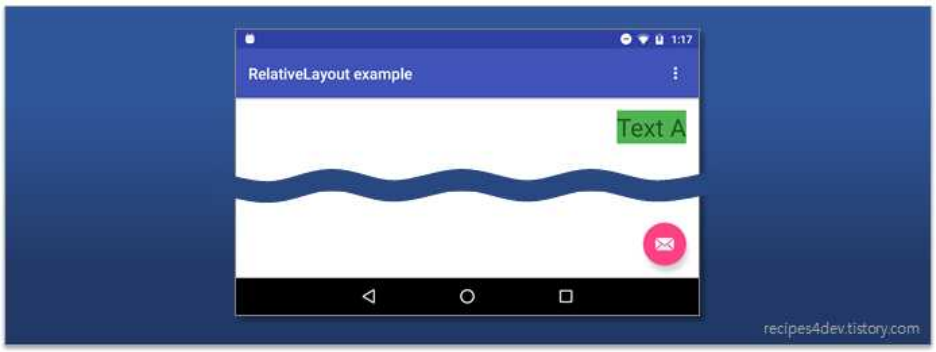
* android:layout_alignParentEnd - 부모(Parent) 영역 내에서 끝 지점(End)에 배치.
> 속성 값에는 true 또는 false 사용.
> 뷰(View) End = 부모(Parent) End.

해당 속성들은 특정 뷰(View)가 아닌 부모(Parent)를 기준으로 사용되므로, 속성 값은 true 또는 false 값이 사용됩니다.

다음 예제는 부모(Parent)를 기준으로 상대적 위치를 지정하는 예제입니다. "A"(id:textA)를 화면에 표시할 때 RelativeLayout의 오른쪽(Right) 영역에 표시하도록 만든 예제입니다. (세로 방향 기본 배치가 위(Top)인 것을 주의하세요.)

```
<RelativeLayout
    android :layout_width="match_parent "
    android :layout_height="match_parent ">

    <TextView
        android :layout_width="wrap_content "
```

```
        android :layout_height="wrap_content "
        android :background="#4CAF50 "
        android :textSize="32sp "
        android :id="@+id/textA "
        android :text="Text A "
        android :layout_alignParentRight="true " />

</RelativeLayout>
```



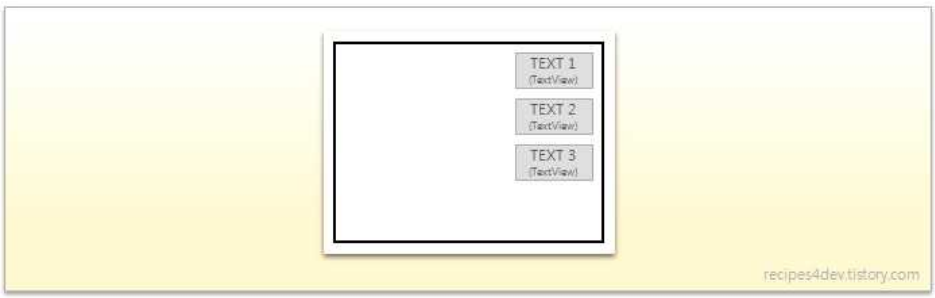
그런데 위의 속성표를 보면, 배치 방향이 가로/세로로 구분되어 있는 것을 알 수 있습니다. 그래서 부모를 기준으로 상대적 위치를 결정할 때는 가로 방향과 세로 방향을 위한 속성을 동시에 사용하는 게 좋습니다. 만약 속성을 지정하지 않으면, 가로 방향인 경우 왼쪽(Left), 세로 방향인 경우 위(Top)가 기본 값으로 적용됩니다.

아래 그림은 가로/세로 속성의 조합으로 배치할 수 있는 경우의 수를 나타낸 것입니다.

4.3 형제(Sibling) 뷰(View) 위젯과 부모(Parent) RelativeLayout 둘 다 적용하기.

RelativeLayout의 자식(Children) 뷰 위젯 배치를 위한 속성이, 자식(Children) 뷰 위젯 간 상대적 위치 결정 속성과 부모(Parent)를 기준으로 상대적 위치 결정 속성으로 나뉘어져 있지만, 무조건 한 가지 종류의 속성만 사용해야 하는 것은 아닙니다. 부모(Parent)와 형제(Sibling)를 동시에 정렬의 기준으로 사용할 수 있다는 것이죠.

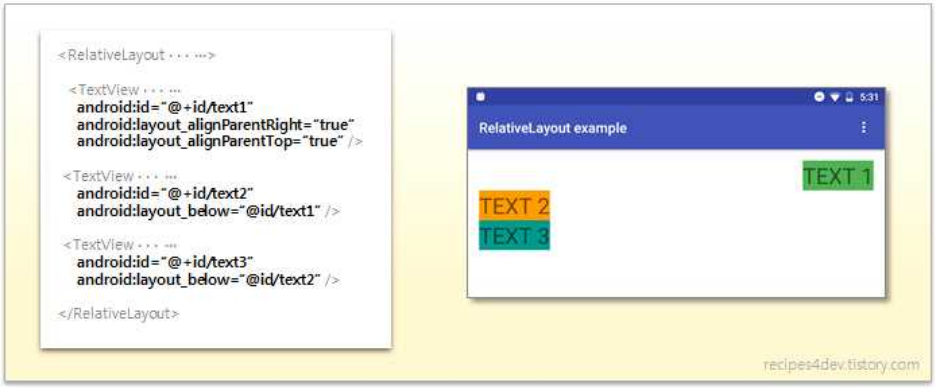
아래 그림의 예를 보겠습니다.



예제와 같은 배치를 완성하기 위한 과정을 한번 수행해 볼까요?

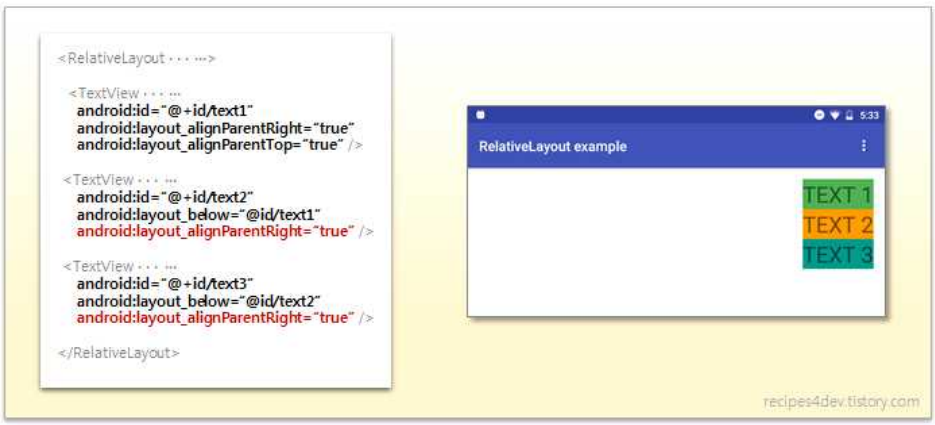
- 최초 기준이 되는 "TEXT 1"을 부모(Parent)의 오른쪽, 상단에 배치.
- "TEXT 2"를 "TEXT 1"의 아래에 배치.
- "TEXT 3"를 "TEXT 2"의 아래에 배치.

간단하죠? 레이아웃 XML 코드를 작성하고 그 결과를 한번 확인해 볼까요?

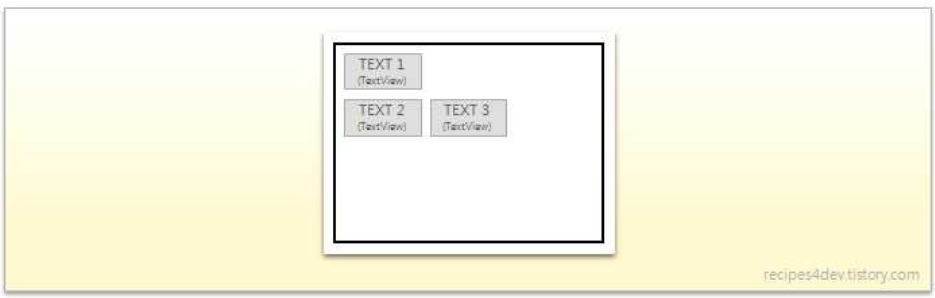


음, 만드려고 했던 결과가 나오지 않네요. 분명 "TEXT 2"를 "TEXT 1"의 아래(below)에 두긴 했지만, 가로 방향 기본 값인 "layout_alignParentLeft"이 적용되어 부모(Parent)의 왼쪽에 정렬되기 때문입니다.

이 때 "TEXT 2"와 "TEXT 3"에 "layout_alignParentRight" 속성을 같이 사용하면, 처음 만드려고 했던 화면을 구성할 수 있습니다.



그리고 또 다른 예를 하나 들어볼까요?

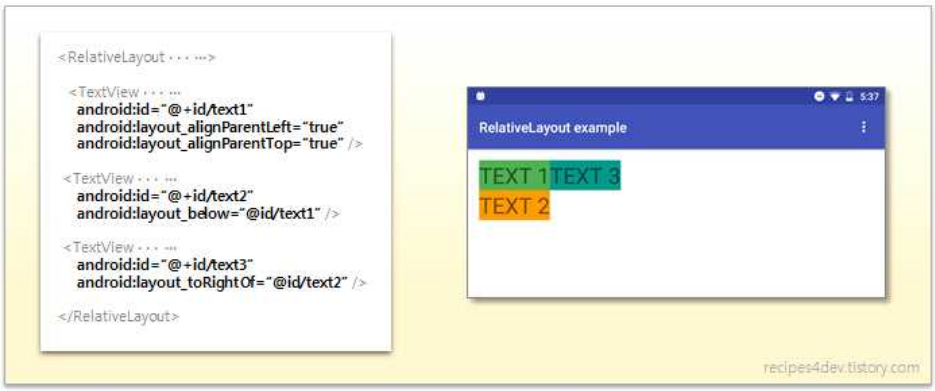


그림과 같은 구조를 만들기 위해서는 다음과 같은 과정을 수행할 것입니다.

- **최초 기준이 되는 "TEXT 1"을 부모(Parent)의 왼쪽, 상단에 배치.**

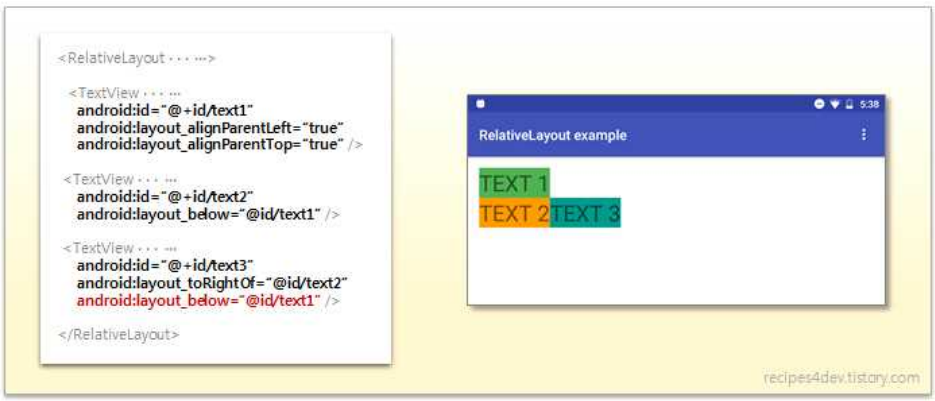
- "TEXT 2"를 "TEXT 1" 아래에 배치.
- "TEXT 3"를 "TEXT 2" 오른쪽에 배치.

자, 결과를 확인하겠습니다.



음, 이번에도 원하는 결과가 나오지 않는군요. "TEXT 3"를 "TEXT 2"의 오른쪽에 배치하는 것 까지는 좋았지만, 세로 방향의 정렬을 지정하지 않았기 때문에 "layout_alignParentTop"이 적용되어 부모 영역의 상단에 정렬되기 때문입니다.

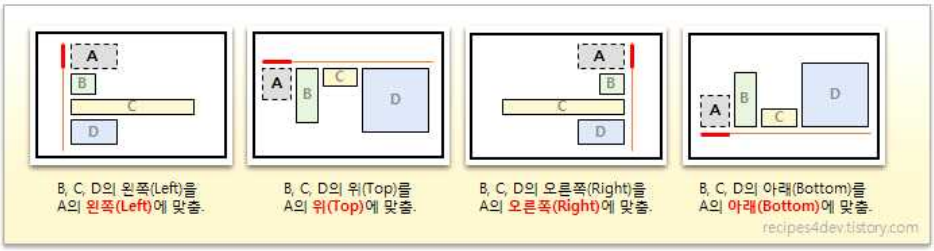
아래와 같이 "TEXT 3"에 "TEXT 2"의 오른쪽(toRightOf)과 "TEXT 1"의 아래(below) 속성을 동시에 사용하여 원하는 결과를 얻을 수 있습니다.



5. 맞춤 정렬(Alignment) 기준 적용하기.

5.1 맞춤 정렬(Alignment)를 위한 속성.

바로 앞에서 설명한 내용에서, RelativeLayout 배치와 관련된 두 개 이상의 속성을 사용하여 자식(Children) 뷰 위젯의 위치를 결정하는 것에 대해 살펴보았습니다. 그런데 예제의 결과를 보면 두 가지 이상의 속성을 같이 사용하는 이유가, 화면에 배치하는 뷰(View)를 기준 뷰(View)와 나란히 배치하고자 함이라는 것을 알 수 있습니다.



그런데 뷰(View) 위젯 사이의 맞춤 정렬(Alignment)을 위해, 굳이 두 가지 이상의 속성을 조합할 필요가 없습니다. RelativeLayout에는 뷰(View) 위젯 간 맞춤 정렬(Alignment)을 명시적으로 지정하는데 사용되는 속성이 제공되기 때문입니다.

* android:layout_alignLeft - 뷰(View)의 왼쪽(Left)을 기준 뷰(Anchor View)의 왼쪽(Left)에 맞춤.
    > 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
    > 뷰(View) Left = 기준 뷰(Anchor View) Left

* android:layout_alignTop - 뷰(View)의 위(Top)을 기준 뷰(Anchor View)의 위(Top)에 맞춤.
    > 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
    > 뷰(View) Top = 기준 뷰(Anchor View) Top

* android:layout_alignRight - 뷰(View)의 오른쪽(Right)을 기준 뷰(Anchor View)의 오른쪽(Right)에 맞춤.
    > 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
    > 뷰(View) Right = 기준 뷰(Anchor View) Right

* android:layout_alignBottom - 뷰(View)의 아래(Bottom)를 기준 뷰(Anchor View)의 아래(Bottom)에 맞춤.
    > 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
    > 뷰(View) Bottom = 기준 뷰(Anchor View) Bottom

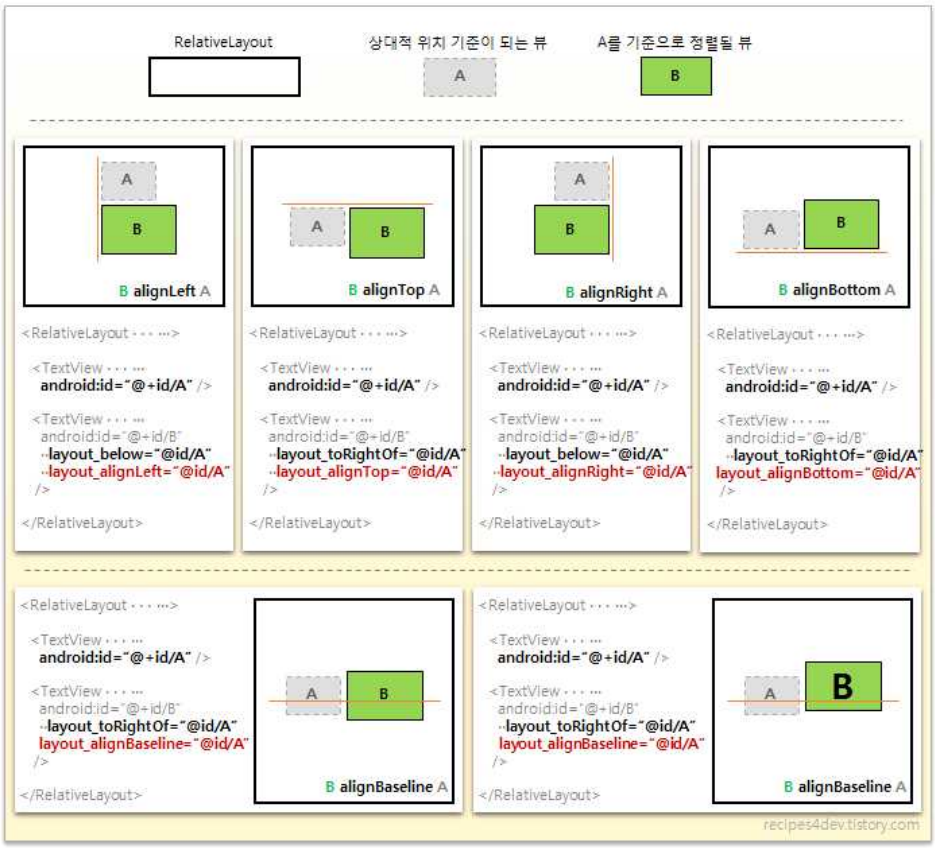
* android:layout_alignBaseline - 뷰(View)의 폰트 기준선(Baseline)을 기준 뷰(View)의 폰트 기준선(Baseline)에 맞춤.
    > 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
    > 뷰(View) Baseline = 기준 뷰(Anchor View) Baseline

* android:layout_alignStart - 뷰(View)의 시작(Start)을 기준 뷰(Anchor View)의 시작(Start)에 맞춤.
    > 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
    > 뷰(View) Start = 기준 뷰(Anchor View) Start

* android:layout_alignEnd - 뷰(View)의 끝(End)을 기준 뷰(Anchor View)의 끝(End)에 맞춤.

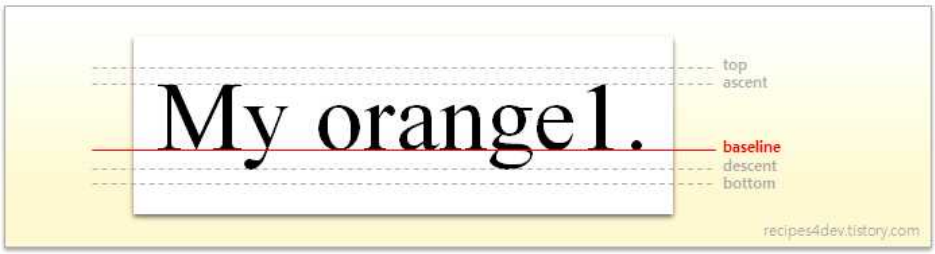
> 속성 값에는 기준 뷰(Anchor View)의 ID를 지정.
> 뷰(View) End = 기준 뷰(Anchor View) End

각 속성을 사용한 예제는 아래와 같습니다.

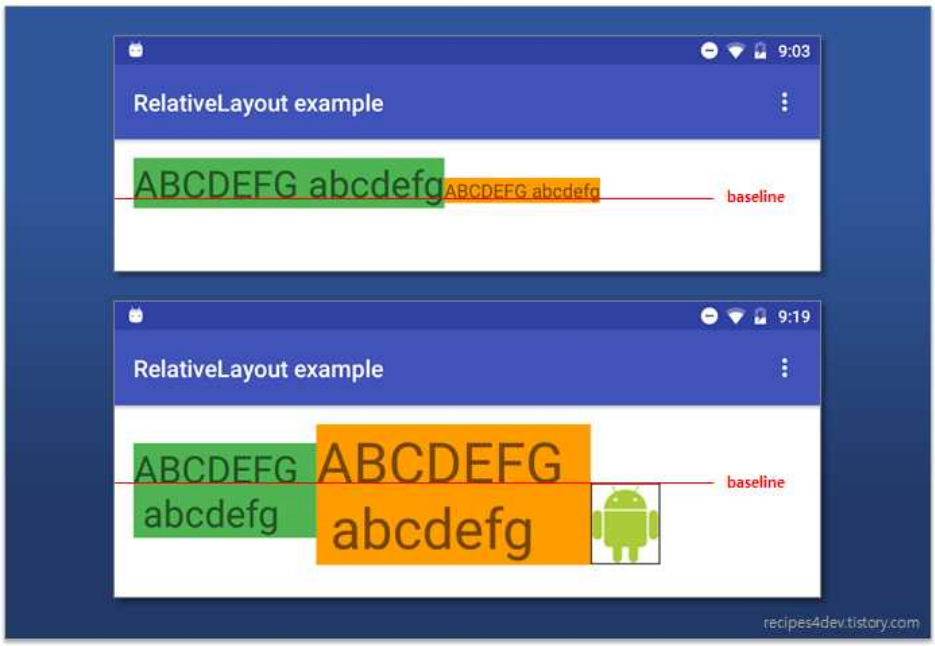


5.2 layout_alignBaseline

맞춤 정렬(Alignment) 속성 중, "layout_alignBaseline" 속성은 그 이름만으로는 표시 결과가 쉽게 떠오르지 않을텐데요, "Baseline"은 폰트 메트릭스(Font Metrics)에서 폰트를 그릴 때 사용되는 기준선(baseline)을 의미합니다.

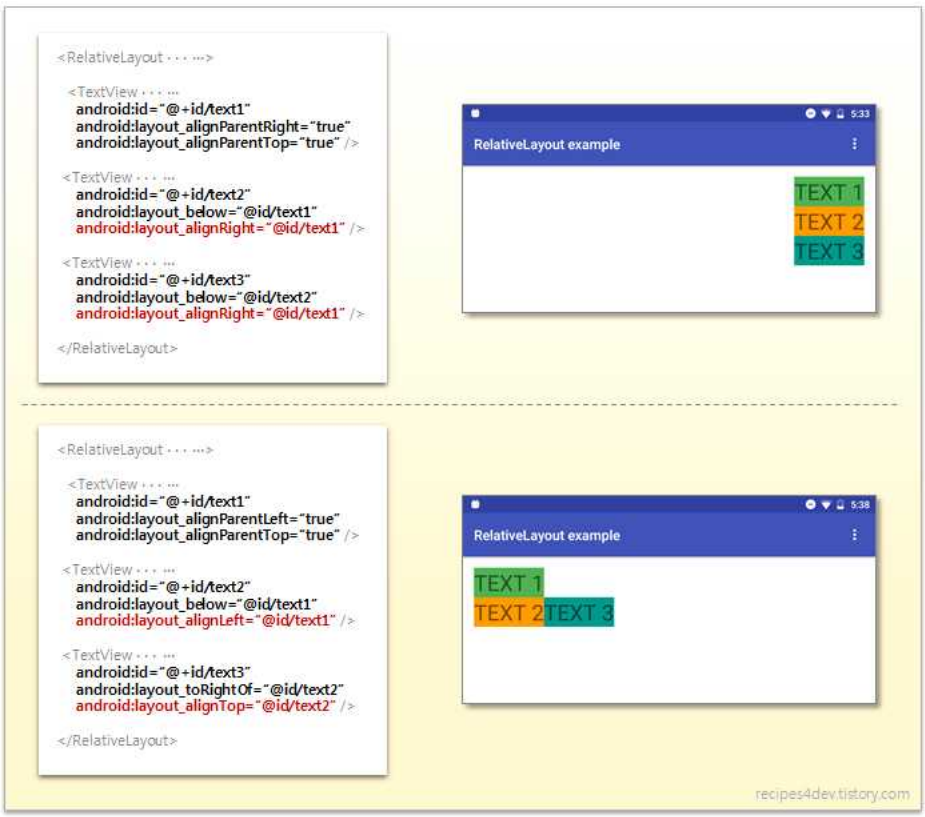


바로 이 기준선(baseline)에 맞춰 정렬(Alignment)하는 속성이 "layout_alignBaseline" 속성인 것이죠. 참고로, 텍스트가 여러 줄(line)로 구성되어도 첫 번째 줄(line)의 기준선(baseline)에 맞춰 정렬되고, 이미지뷰(ImageView)와 같이 텍스트를 포함하지 않는 경우에는 위(Top)가 기준선(baseline)으로 취급됩니다.



5.3 맞춤 정렬 사용 예제

맞춤 정렬(Alignment) 속성에 대한 좀 더 다양한 결과를 확인하기 위해, 앞서 〔4.3 형제(Sibling) 뷰(View) 위젯과 부모(Parent) RelativeLayout 둘 다 적용하기.〕에서 살펴본 예제를, 맞춤 정렬(Alignment) 속성을 사용하여 구현해 보겠습니다.



6. Start와 End

RelativeLayout에서 사용할 수 있는 속성을 보면 Start와 End가 포함된 속성을 발견할 수 있습니다.

* layout_toStartOf
* layout_toEndOf
* layout_alignParentStart
* layout_alignParentEnd
* layout_alignStart
* layout_alignEnd

일단 시스템 언어 설정이 "한국어"인 경우라면, Start는 Left와 동일하고, End는 Right와 동일합니다. 이는 "한국어"가 기본적으로 LTR(Left To Right), 즉, 왼쪽에서 오른쪽으로 쓰고 읽는 구조이기 때문에, 컨텐츠의 시작 지점인 Start가 Left와 같은 의미가 되고, 반대로 끝 지점인 End가 Right와 동일한 의미가 되는 것입니다.

하지만 아랍권의 일부에서는 반대 방향인 경우가 있습니다. 컨텐츠의 방향이 오른쪽에서 왼쪽으로 나열되는 것이죠. 이를 RTL(Right to Left) 언어라고 지칭하며, 안드로이드에서는 API Level 17(4.2)부터 시스템 언어 설정에 따른 RTL 방식을 지원하고 있습니다.

본문에서는 Left와 Right를 예로 들었지만, 일반적인 상황에서는 Start와 End를 사용하시기 바랍니다. 그래야만 RTL 언어 사용 환경에서 그에 맞는 내용을 표시할 테니까요.

7. Layout 관련 공통 사항.

7.1 LinearLayout안에 배치되는 자식(Children) View 위젯의 크기 지정.

[Layout 또는 View 위젯의 크기. (layout_width, layout_height)]의 내용을 참고하세요.

7.2 LinearLayout과 자식(Children) View 위젯 사이의 여백 지정.

[Layout 또는 자식(Children) View 위젯 요소 간 여백. (layout_margin, padding)]의 내용을 참고하세요.

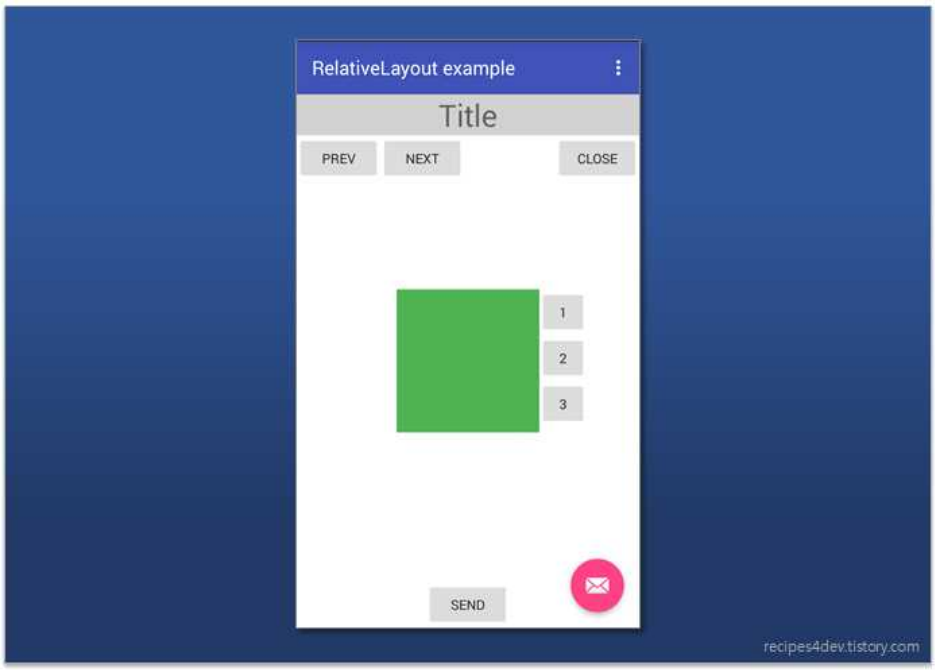
8. RelativeLayout 사용 예제

마지막으로, 이 글의 처음에 예로 든 화면을, RelativeLayout을 사용하여 작성한 코드는 아래와 같습니다.

```
<RelativeLayout
    android :layout_width="match_parent "
    android :layout_height="match_parent ">

    <TextView
        android :layout_width="match_parent "
        android :layout_height="wrap_content "
```

```
        android :background="#CCCCCC "
        android :gravity="center "
        android :textSize="32sp "
        android :id="@+id/title "
        android :text="Title " />

    <Button
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :id="@+id/prev "
        android :layout_below="@id/title "
        android :text="Prev "/>

    <Button
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :layout_below="@id/title "
        android :layout_toRightOf="@id/prev "
        android :text="Next "/>

    <Button
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :layout_below="@id/title "
        android :layout_alignParentRight="true "
        android :text="Close "/>

    <ImageView
        android :layout_width="150dp "
        android :layout_height="150dp "
        android :background="#4CAF50 "
        android :id="@+id/image "
        android :layout_centerInParent="true "/>

    <Button
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :id="@+id/no1 "
        android :layout_toRightOf="@id/image "
        android :layout_alignTop="@id/image "
        android :text="1 "/>

    <Button
```

```
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :id="@+id/no2 "
        android :layout_below="@id/no1 "
        android :layout_alignLeft="@id/no1 "
        android :text="2 "/>

    <Button
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :id="@+id/no3 "
        android :layout_below="@id/no2 "
        android :layout_alignLeft="@id/no1 "
        android :text="3 "/>

    <Button
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :id="@+id/send "
        android :layout_alignParentBottom="true "
        android :layout_centerHorizontal="true "
        android :text="Send "/>

</RelativeLayout>
```



9. 참고.

- 레이아웃에 대한 자세한 도움말.
    - [안드로이드 개발 참조문서 - 레이아웃 항목]을 참고하세요.
- RelativeLayout에 대한 자세한 도움말.
    - [안드로이드 개발 API 가이드 - RelativeLayout 항목]을 참고하세요.
    - [안드로이드 개발 참조문서 - RelativeLayout 항목]을 참고하세요.
    - [안드로이드 개발 참조문서 - RelativeLayout.LayoutParams 항목]을 참고하세요.

.END.