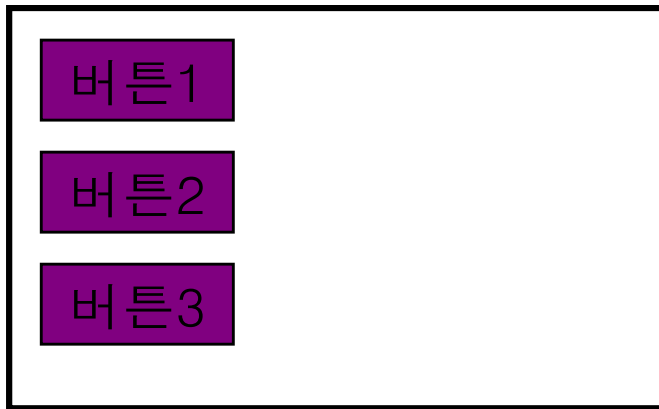


# 레이아웃

# 리니어 레이아웃

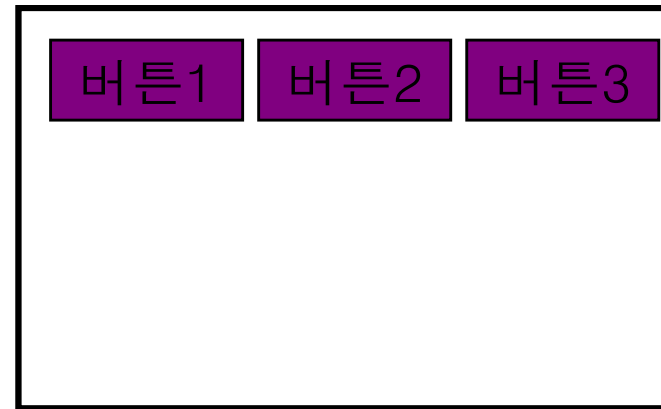
## ■ 방향

- 차일드를 일렬로 배치한다. 등장 순서대로 차곡 차곡 배치하며 공백은 없다.
- 단순해 보이지만 직관적이고 사용 빈도가 높다.
- orientation 속성 : 차일드의 배치 방향을 지정한다.



vertical

가



horizontal

가

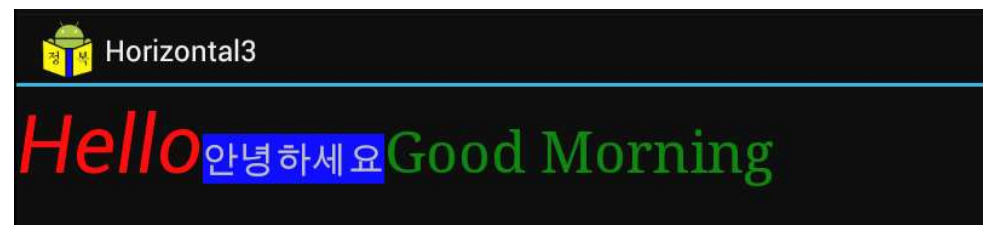
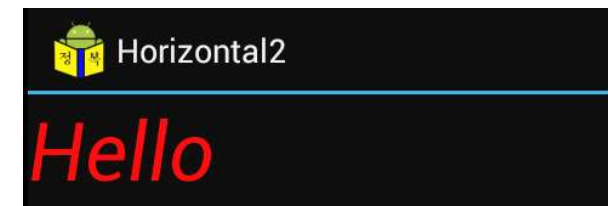
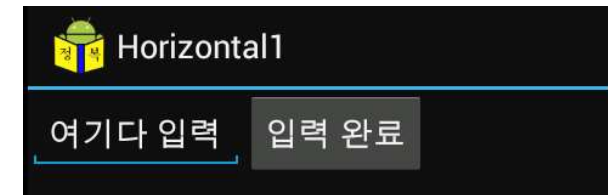
# 리니어 레이아웃

## ■ 방향

- 속성끼리 상호 영향을 미치므로 조합을 잘 적용해야 한다.

## ■ 예제

- [horizontal1, layout](#)
- [horizontal2, layout](#)
- [horizontal3, layout](#)
- [strings](#)



# 리니어 레이아웃

## ■ 정렬

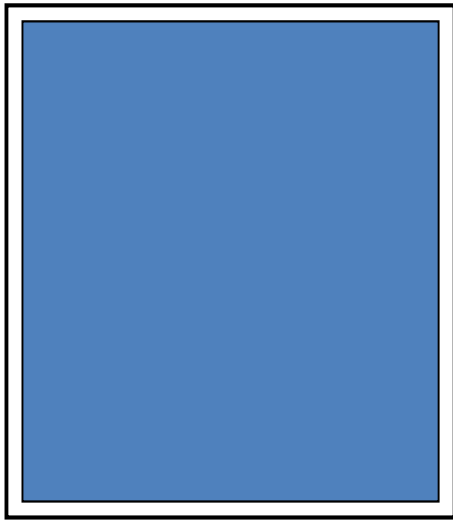
- gravity 속성 : 뷰의 내용물을 어디에 배치할 것인가를 지정
- 수직, 수평 각 방향에 대해 다르게 정렬할 수 있다.

상수	값	설명
center_horizontal	0x01	수평으로 중앙에 배치한다.
left	0x03	컨테이너의 왼쪽에 배치한다. 크기는 바뀌지 않는다.
right	0x05	컨테이너의 오른쪽에 배치한다.
fill_horizontal	0x07	수평 방향으로 가득 채운다.
center_vertical	0x10	수직으로 중앙에 배치한다.
top	0x30	컨테이너의 상단에 배치한다. 크기는 바뀌지 않는다.
bottom	0x50	컨테이너의 하단에 배치한다.
...		
fill_vertical	0x70	수직 방향으로 가득 채운다.
center	0x11	수평으로나 수직으로 중앙에 배치한다.
fill	0x77	컨테이너를 가득 채우도록 수직, 수평 크기를 확장한다.

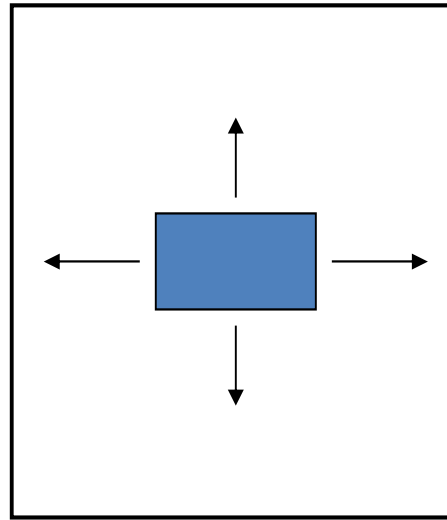
# 리니어 레이아웃

## ■ 차일드 정렬

- `layout_gravity` : 뷰를 부모의 어디에 배치할 것인가를 지정한다.
- 뷰와 부모 사이에 여백이 있어야만 효과가 있다.



정렬이 의미가 없는 경우



정렬 가능한 경우

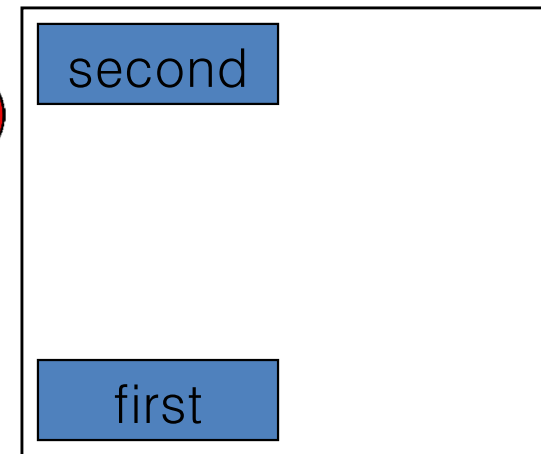
# 리니어 레이아웃

## ■ 정렬과 레이아웃

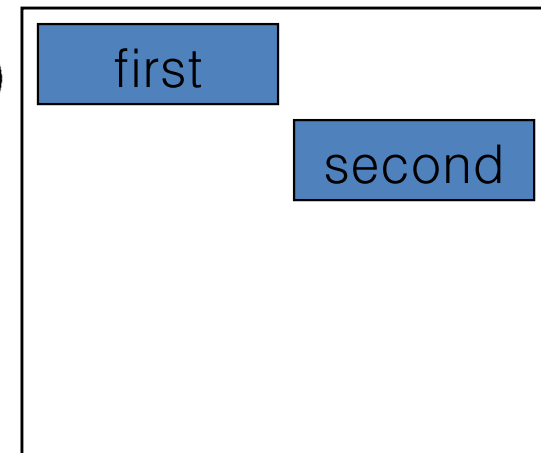
- 정렬보다는 레이아웃의 고유 규칙이 우선이다. 리니어는 순서대로 배치하므로 같은 방향의 뷰 정렬을 무시한다.

`layout_gravity`  
`orientation,` `horizontal`

```
<LinearLayout orientation="vertical"
  <TextView text="first" layout_gravity="bottom"
  <TextView text="second" layout_gravity="top"
```



```
<LinearLayout orientation="vertical"
  <TextView text="first" layout_gravity="left"
  <TextView text="second" layout_gravity="right"
```

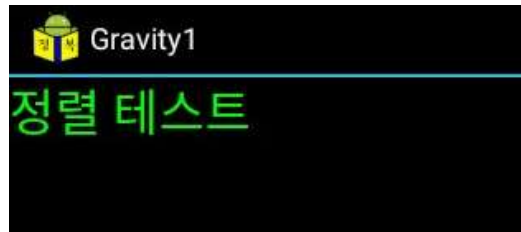


# 리니어 레이아웃

## ■ 정렬

- gravity는 뷰 안의 내용물, layout\_gravity는 뷰 자체의 정렬을 지정한다.
- 레이아웃은 보기보다 어렵고 경험을 요하는 기술이다.
- 예제 : [gravity1](#), [layout](#)

가



- 예제 : [gravity2](#), [layout](#)



- 예제 : [gravity3](#), [layout](#)

# 리니어 레이아웃

## ■ 정렬

- 예제 : [gravity4](#), [layout](#)



- 예제 : [gravity5](#), [layout](#)





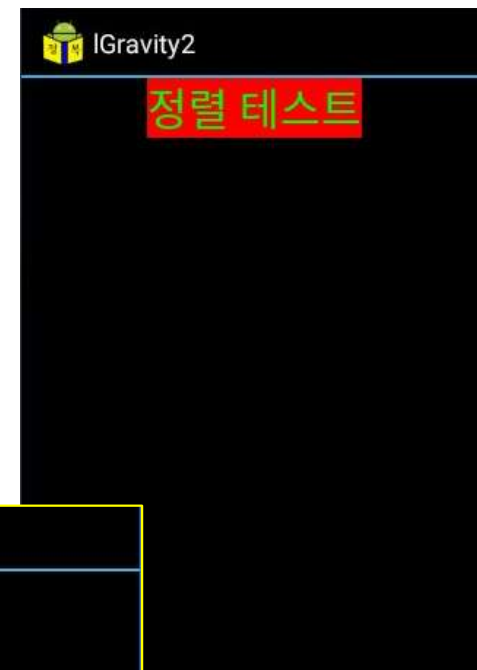
# 리니어 레이아웃

## ■ 정렬

● 예제 : [IGravity1](#), [layout](#)

● 예제 : [IGravity2](#), [layout](#)

● 예제 : [IGravity3](#), [layout](#)



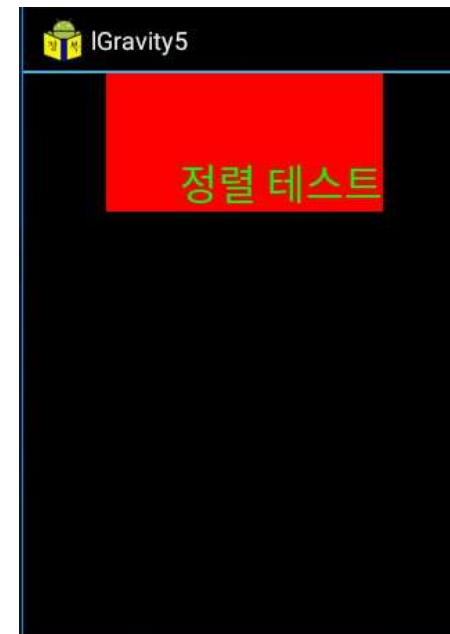
# 리니어 레이아웃

## ■ 정렬

- 예제 : [IGravity4](#), [layout](#)



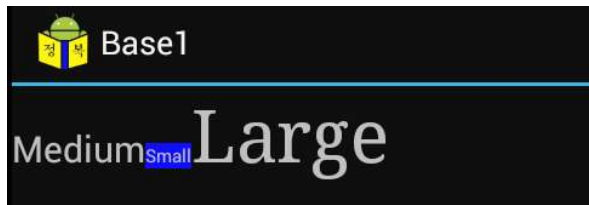
- 예제 : [IGravity5](#), [layout](#)



# 리니어 레이아웃

## ■ 베이스 정렬

- `baselineAligned` : 높이가 다른 차일드를 수평 배치할 때 밑면을 정렬할 것인가를 지정한다. 디폴트는 `true`이다.
- 예제: [Base1, layout](#), [Base2, layout](#)



엄마와 나

`baselineAligned = true`



엄마와 나

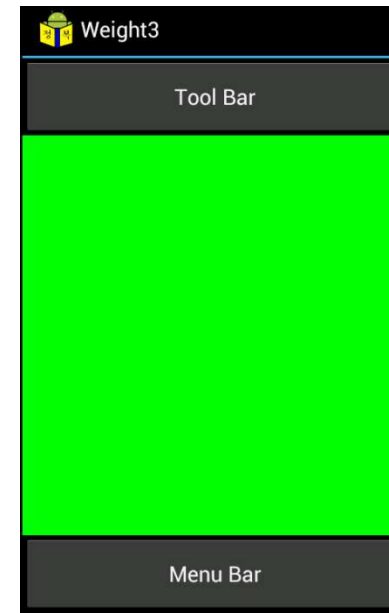
`baselineAligned = false`

# 리니어 레이아웃

## ■ 영역 분할

- layout\_weight : 부모 레이아웃의 남은 영역을 차지할 비율.
- 0이면 고유의 크기만큼 차지하며 1이상이면 형제 뷰와 비율에 따라 균등 배분된다.
- 같은 방향의 크기는 0으로 지정한다.
- 비율일 뿐이므로 절대값은 중요하지 않다.

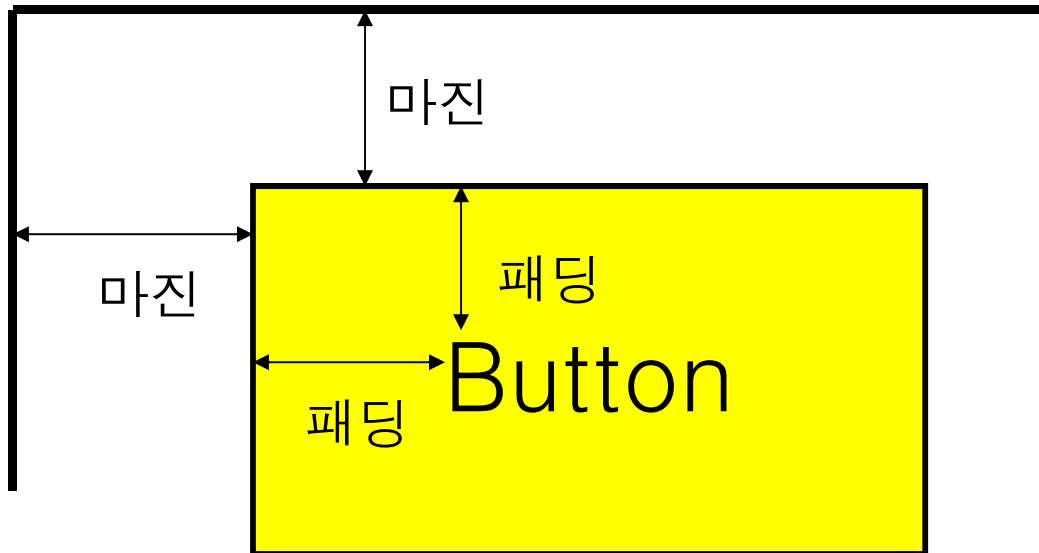
## ■ 예제 : [weight1 layout](#), [weight2 layout](#), [weight3 layout](#)



# 리니어 레이아웃

## ■ 마진과 패딩

- padding : 뷰와 내용물 사이의 간격, 안쪽 여백
- layout\_margin : 뷰와 부모와의 간격, 바깥 여백
- 상하좌우 각각에 여백을 따로 줄 수도 있고 한꺼번에 여백을 지정할 수도 있다.



# 렐러티브 레이아웃

## ■ RelativeLayout 소개

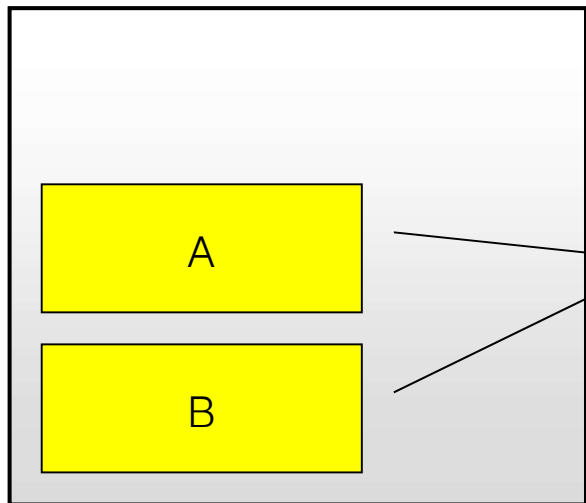
- 뷰와 부모와의 관계 또는 뷰끼리의 관계를 지정함으로써 차일드를 배치한다.
- 관계 지정을 위해 id가 반드시 필요하다.

속성	설명
layout_above	~의 위에 배치한다.
layout_below	~의 아래에 배치한다.
layout_toLeftOf	~의 왼쪽에 배치한다.
layout_toRightOf	~의 오른쪽에 배치한다.
layout_alignLeft	~와 왼쪽 변을 맞춘다.
layout_alignTop	~와 위쪽 변을 맞춘다.
layout_alignRight	~와 오른쪽 변을 맞춘다.
layout_alignBottom	~와 아래쪽 변을 맞춘다.
layout_alignParentLeft	true이면 부모와 왼쪽 변을 맞춘다.
layout_alignParentTop	true이면 부모와 위쪽 변을 맞춘다.
layout_alignParentRight	true이면 부모와 오른쪽 변을 맞춘다.
layout_alignParentBottom	true이면 부모와 아래쪽 변을 맞춘다.
layout_alignBaseline	~와 베이스라인을 맞춘다.
layout_alignWithParentIfMissing	layout_toLeftOf 등의 속성에 대해 앵커가 발견되지 않으면 부모를 앵커로 사용한다.
layout_centerHorizontal	true이면 부모의 수평 중앙에 배치한다.
layout_centerVertical	true이면 부모의 수직 중앙에 배치한다.
layout_centerInParent	true이면 부모의 수평, 수직 중앙에 배치한다.

# 렐러티브 레이아웃

## ■ 레이아웃 전체로 위젯 위치 파악

- 화면상의 순서와 레이아웃 상의 순서가 달라질 수도 있다.



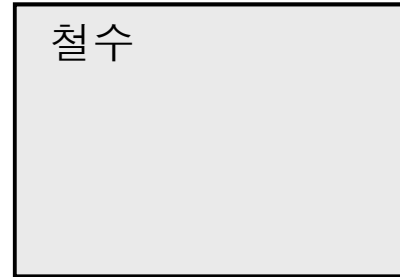
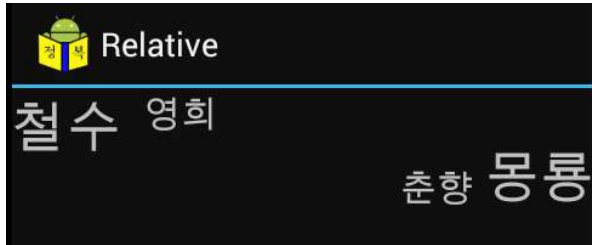
원하는 배치

```
<RelativeLayout>
<B
  android:id="@+id/b"
/>
<A
  layout_above="@id/b"
/>
</RelativeLayout>
```

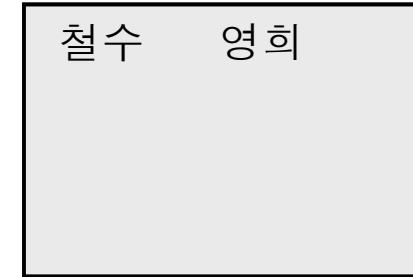
XML 문서

# 렐러티브 레이아웃

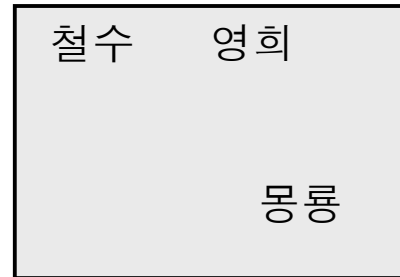
## ■ 예



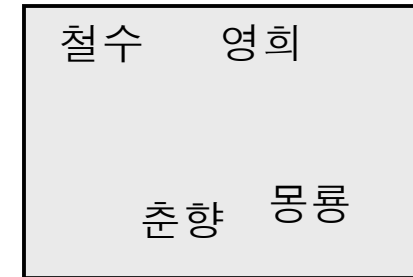
좌상단에



철수 오른쪽에



영희 밑에  
부모의 오른쪽에



몽룡 왼쪽에  
몽룡의 아래쪽 정렬



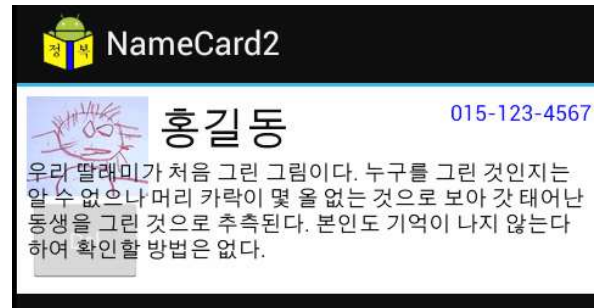
# 렐러티브 레이아웃

## ■ 예제

- [namecard layout](#)



- [namecard2 layout](#)



description

android:layout\_alignLeft="@id/name"

# 기타 레이아웃

---

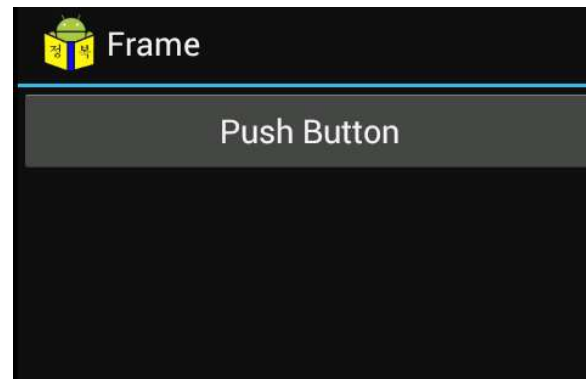
## ■ **AbsoluteLayout**

- layout\_x, layout\_y 속성으로 차일드의 절대 좌표를 지정한다.
- 가장 쉽지만 호환성이 좋지 않으며 관리하기 어렵다.
- 비권장 레이아웃이며 현재는 경고 처리된다.
- 절대 좌표 대신 마진을 사용해도 동일한 효과를 낼 수 있다.

# 기타 레이아웃

## ■ FrameLayout

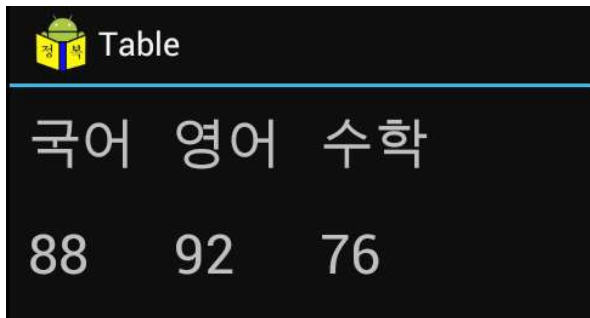
- 차일드를 좌상단에 겹쳐서 배치한다.
- 여러 개의 차일드를 가질 수 있다.
- 겹쳐 놓고 선택적으로 차일드를 보일 수 있다.
- foreground : 위쪽에 얹히는 이미지 지정



# 기타 레이아웃

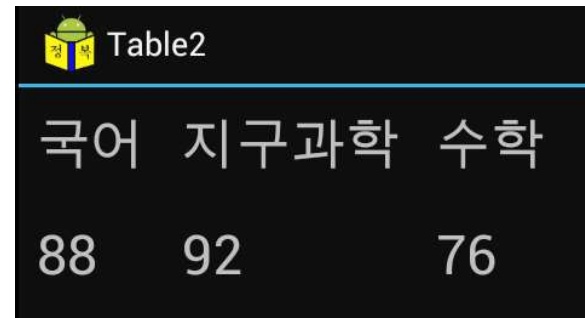
## ■ TableLayout

- 표 형식으로 차일드를 배치한다.
- TableRow가 행이며 행안에 셀이 있고 셀 안에 차일드를 배치한다.
- 셀내의 차일드는 항상 셀을 가득 채우므로 차일드의 크기는 지정하지 않아도 상관없다.
- 셀이 늘어나면 뒤쪽 셀의 위치가 자동 조정된다.
- 셀 병합 등의 고급 기능이 있으나 사용 빈도는 낮다.



A screenshot of an Android application window titled 'Table' with an Android icon. It displays a table with two rows and three columns. The first row contains the subjects '국어', '영어', and '수학'. The second row contains the scores '88', '92', and '76'.

국어	영어	수학
88	92	76



A screenshot of an Android application window titled 'Table2' with an Android icon. It displays a table with two rows and three columns. The first row contains the subjects '국어', '지구과학', and '수학'. The second row contains the scores '88', '92', and '76'.

국어	지구과학	수학
88	92	76