출처: maxpixel.net

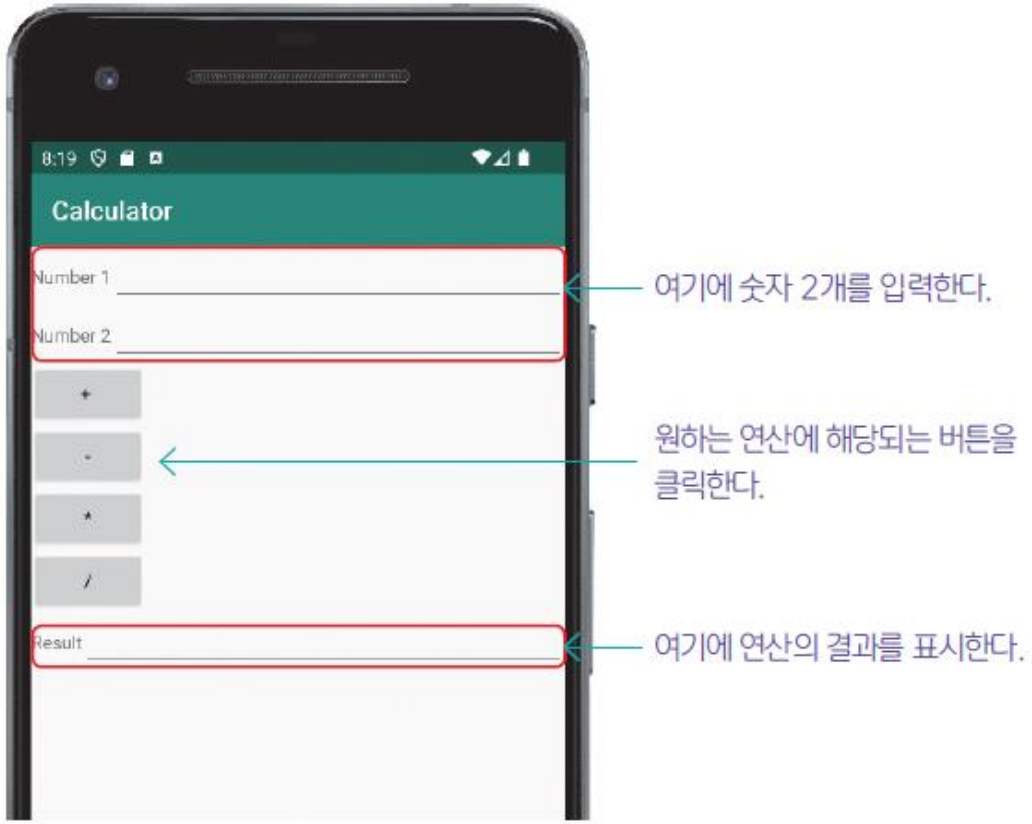
CHAP 4. 레이아웃

레이아웃


안드로이드의 레이아웃에는 어떤 것이 있나요?
레이아웃에는 여러 가지 종류가 있지만, 우리가 사용할 수 있는 것은 선형 레이아웃과 상대적 레이아웃 등이 있단다.

4주 목표

- 다음과 같은 계산기 앱을 작성해보자.



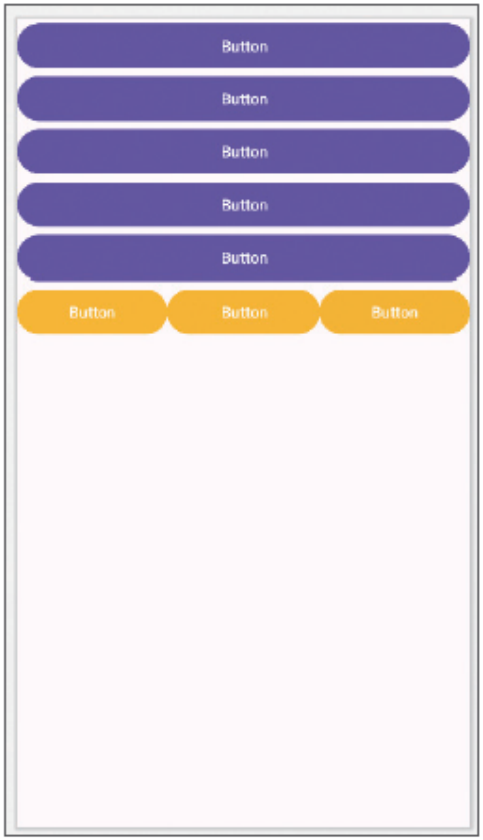
레이아웃

- 뷰들을 화면에 배치하는 방법
- 화면의 크기가 통일되어 있지 않기 때문에 (300, 200)과 같이 절대 위치를 사용하여 위젯을 배치하는 것은 좋은 생각이 아니다.

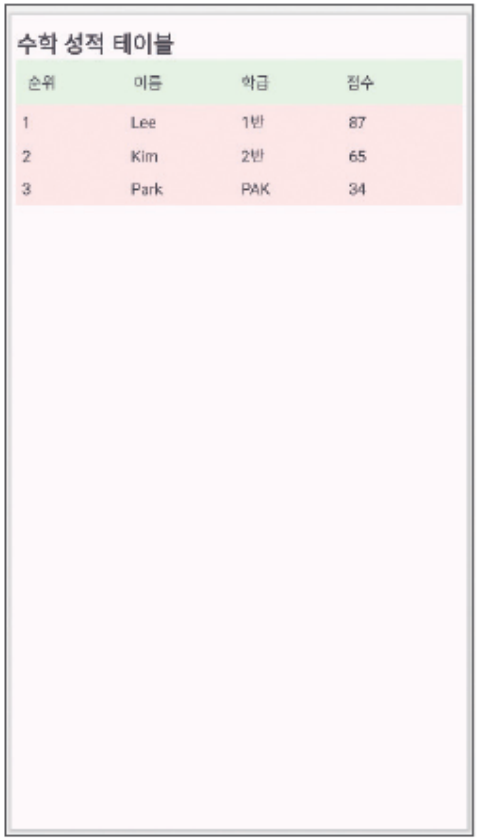
레이아웃의 종류

레이아웃 클래스	설명
LinearLayout	선형 레이아웃으로 수평 또는 수직 방향으로 위젯을 배치할 수 있다. 주로 단순한 레이아웃을 만들 때 사용되며, 단순한 배치 구조를 가지고 있다.
TableLayout	그리드 또는 테이블 형식의 레이아웃을 만들 때 사용된다. 행과 열로 구성된 테이블 레이아웃을 사용하여 데이터를 정렬할 수 있다.
GridLayout	그리드 형식의 레이아웃을 만들 때 사용되며, TableLayout과 유사하지만 보다 유연하게 그리드를 설정할 수 있다.
RelativeLayout	위젯을 상대적인 위치에 배치하는 레이아웃이다. 다른 위젯과의 상대적인 위치를 설정하여 유연한 UI 디자인을 가능하게 한다.
ConstraintLayout	복잡한 UI 디자인을 만들 때 사용되며, 위젯 간의 제약 조건을 사용하여 배치한다. 유연하고 복잡한 레이아웃을 만들 수 있으며, Android Studio의 디자인 에디터에서 시각적으로 편집할 수 있다.
TabLayout	탭 형식의 인터페이스를 생성할 때 사용된다. 여러 화면을 탭으로 전환할 수 있다.
AbsoluteLayout	위젯을 화면의 절대적인 좌표 (x, y)를 사용하여 배치한다. 이는 다양한 화면 크기 및 해상도에 대응하기 어렵게 만든다. 현재의 안드로이드 버전에서는 공식적으로 더 이상 권장되지 않고, 사용하지 않는 것이 좋다.
ScrollView	화면이 스크롤이 가능하도록 하는 컨테이너 레이아웃이다. 넘치는 컨텐츠를 스크롤하여 볼 수 있게 한다.
FrameLayout	하나의 위젯 또는 뷰만을 표시할 수 있는 가장 간단한 레이아웃이다. 주로 한 번에 하나의 위젯 또는 뷰를 표시하는 데 사용되며, 레이아웃이 중첩될 때 유용하다.

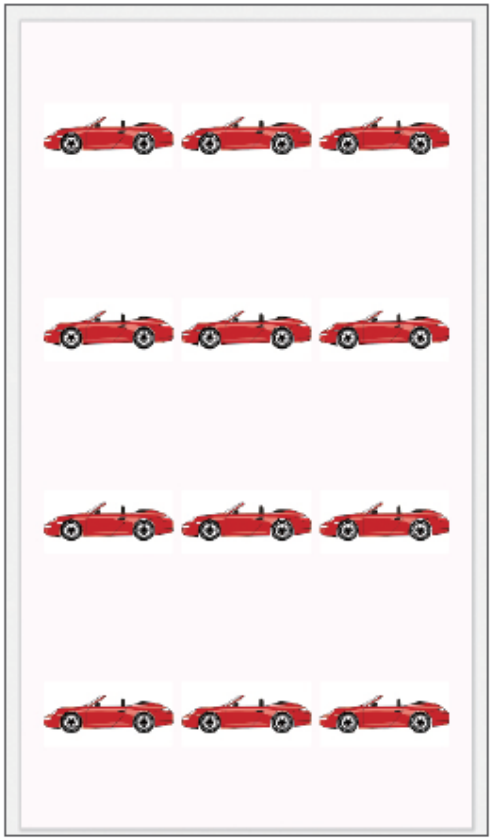
레이아웃의 종류



LinearLayout



TableLayout



GridLayout

선형 레이아웃

- 선형 레이아웃(linear layout)은 가장 기본적인 배치 관리자이다. 선형 레이아웃은 자식 뷰들을 수직 또는 수평으로 배치한다.

(a) 수평 배치 (b) 수직 배치

선형 레이아웃 클래스의 속성

속성	관련 메소드	설명
orientation	setOrientation(int)	"horizontal"은 수평으로, "vertical"은 수직으로 배치한다.
gravity	setGravity(int)	x축과 y축 상에 자식을 어떻게 배치할 것인지를 지정한다.
baselineAligned	setBaselineAligned (boolean)	false로 설정하면 자식뷰들의 기준선을 정렬하지 않는다.

예제: 수평 선형 레이아웃

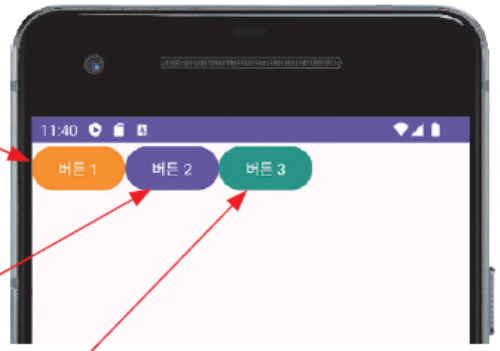
- 뷰들을 수직으로 배치하는 예제는 앞장에서 충분히 다루어 보았으므로 이번에는 수평으로 뷰들을 배치하는 예제를 살펴보자.

예제: 수평 선형 레이아웃

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼 1"/>
    <Button
        android:id="@+id/button02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼 2"/>
    <Button
        android:id="@+id/button03"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼 3"/>
</LinearLayout>
```

자식을 수평으로 배치



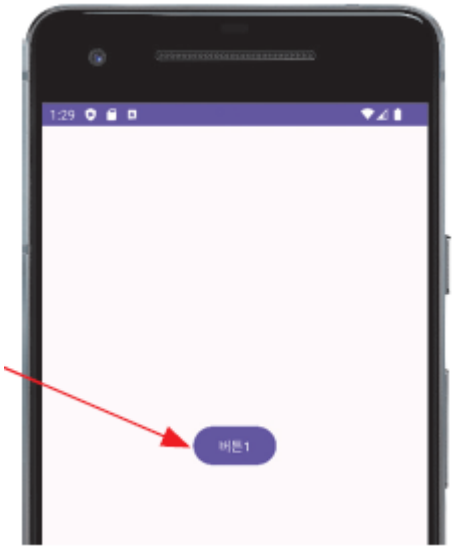
Gravity 속성 값

- gravity 속성을 이용하여서 자식 뷰를 화면의 중앙에 배치할 수도 있고 상단이나 하단에 배치할 수도 있다.

상수	값	설명
top	0x30	객체를 컨테이너의 상단에 배치, 크기를 변경하지 않음
bottom	0x50	객체를 컨테이너의 하단에 배치, 크기를 변경하지 않음
left	0x03	객체를 컨테이너의 좌측에 배치, 크기를 변경하지 않음
right	0x05	객체를 컨테이너의 우측에 배치, 크기를 변경하지 않음
center_vertical	0x10	객체를 컨테이너의 수직의 중앙에 배치, 크기를 변경하지 않음
fill_vertical	0x70	객체를 컨테이너의 수직을 채우도록 배치
center_horizontal	0x01	객체를 컨테이너의 수평의 중앙에 배치, 크기를 변경하지 않음
fill_horizontal	0x07	객체를 컨테이너의 수평을 채우도록 배치
center	0x11	객체를 컨테이너의 수평, 수직의 중앙에 배치
fill	0x77	객체가 컨테이너를 가득 채우도록 배치

예제: Gravity 속성을 이용한 배치 예제

- 선형 레이아웃의 gravity 값을 “center”로 하는 애플리케이션을 작성하여 결과를 살펴보자.

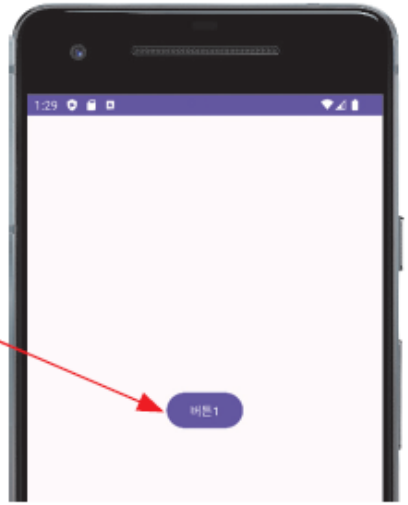


예제: Gravity 속성을 이용한 배치 예제

activity_main.xml

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

    <Button
        android:id="@+id/button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼1"
    />
</LinearLayout>
```

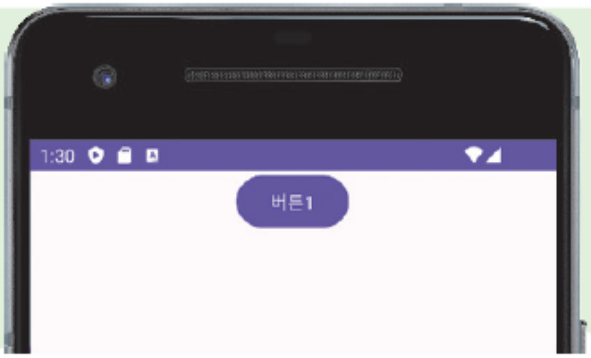
"자식 뷰를 중앙에 배치할 것!" 이라는 의미이다.



gravity 속성 vs layout_gravity 속성

- gravity 속성이 컨테이너에 자식 위젯들을 어떻게 배치하느냐를 나타낸다면, layout_gravity는 자신의 위치를 부모 레이아웃의 어디에 위치시킬 것인지를 지정한다.
- 차이가 생기는 경우도 있다.

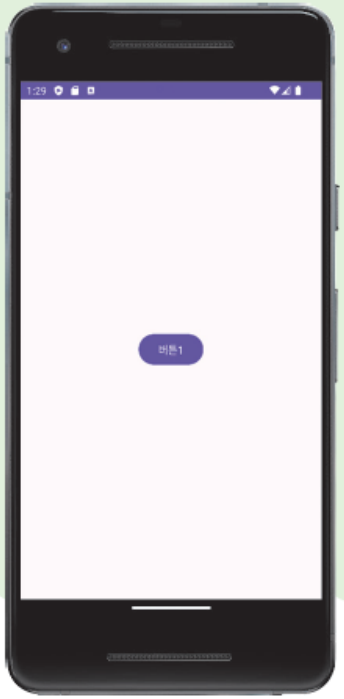
```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:text="버튼1" />
```



gravity 속성 vs layout_gravity 속성

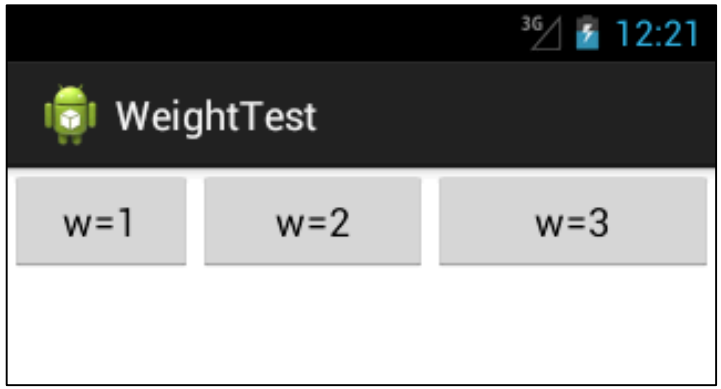
- android:gravity="center"(레이아웃에 설정): 이 설정은 레이아웃 안의 모든 자식 뷰(버튼 포함)에게 영향을 미친다.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼1" />

    <!-- 다른 뷰나 버튼들 -->
</LinearLayout>
```



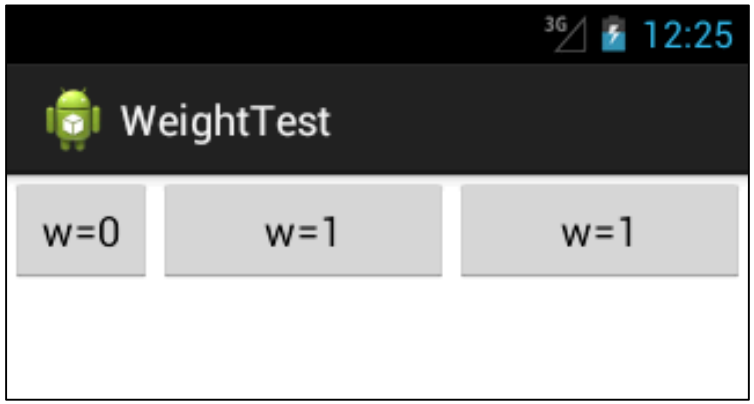
가중치(weight)

- 선형 레이아웃의 자식 뷰들의 가중치가 각각 1, 2, 3이면, 남아있는 공간의 1/6, 2/6, 3/6을 각각 할당받는다.



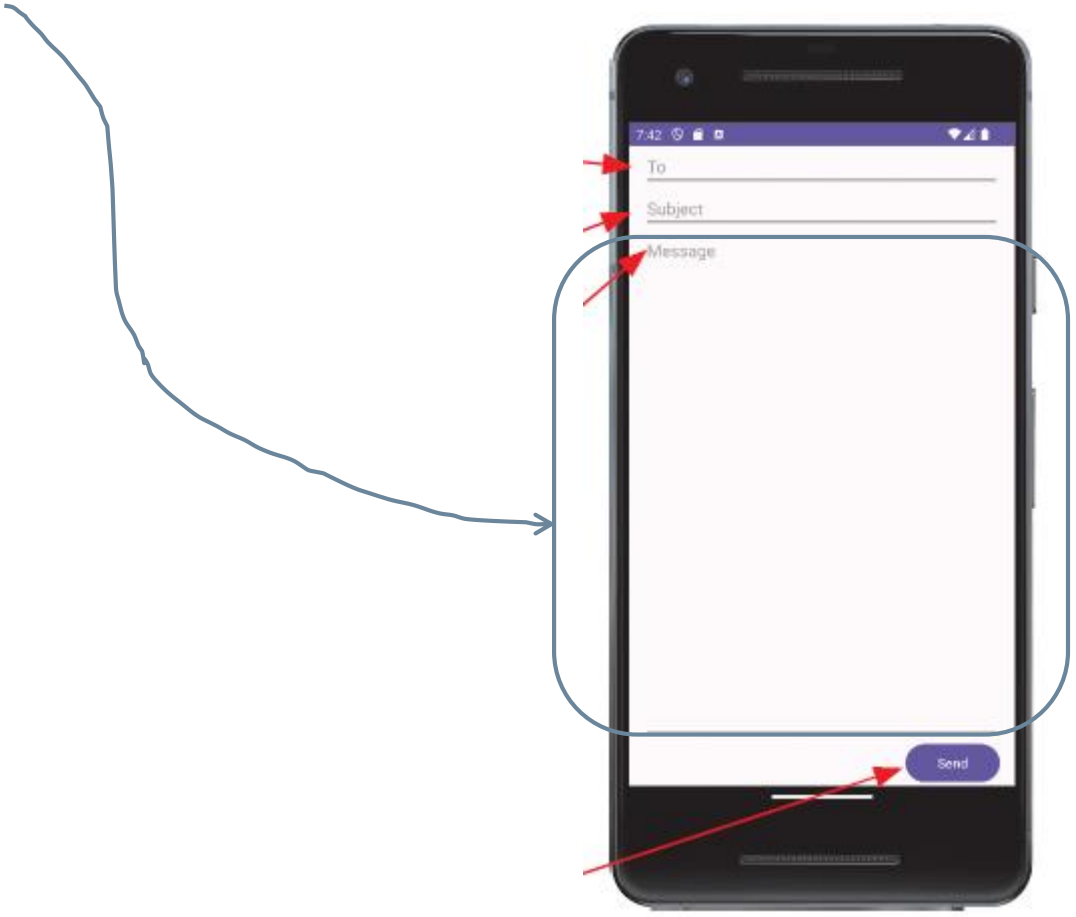
가중치(weight)

- 가중치를 1로 선언한 2개의 텍스트 뷰들은 남아있는 공간을 동일하게 차지할 것이다.



예제: 가중치

- 에디트 텍스트만 가중치가 1이고 나머지는 전부 0



activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingLeft="16dp"
  android:paddingRight="16dp"
  android:orientation="vertical" >

  <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="To" />
  <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Subject" />
  <EditText
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_weight="1"
      android:gravity="top"
      android:hint="Message" />
  <Button
      android:layout_width="100dp"
      android:layout_height="wrap_content"
      android:layout_gravity="right"
      android:text="Send" />
</LinearLayout>
```

가중치가 디폴트로 0이 된다.

가중치를 1로 설정

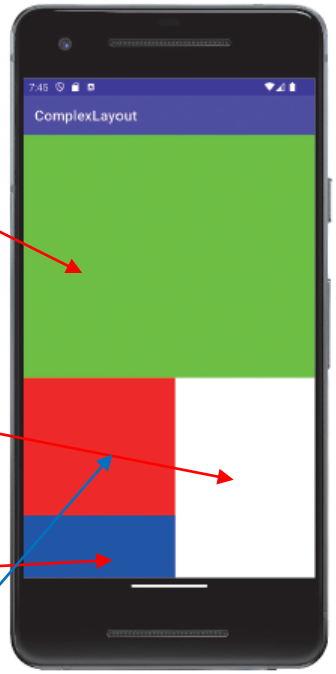
중첩 선형 레이아웃

- 다음과 같은 앱을 작성하려면 선형 레이아웃을 어떻게 사용해야 할까?

중첩 선형

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFFFFF"
    android:orientation="vertical"

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="344dp"
        android:background="#00FF00"
        android:orientation="horizontal"></LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="385dp"
        android:orientation="horizontal">

        <LinearLayout
            android:layout_width="214dp"
            android:layout_height="match_parent"
            android:background="#0000FF"
            android:orientation="vertical">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="195dp"
                android:background="#FF0000"
```

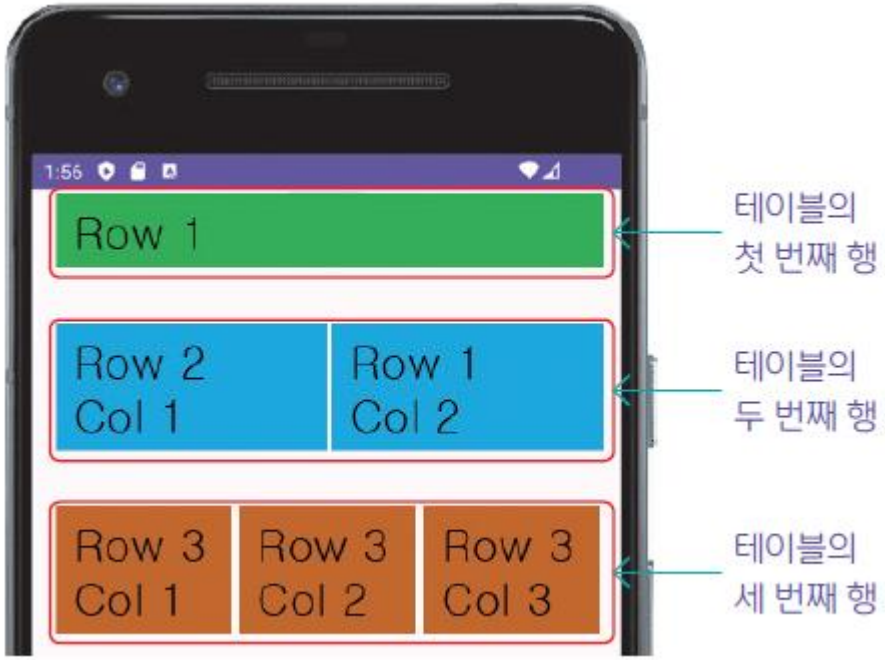


이것이 선형 레이아웃의 문제점이기도 하다. 중첩 레이아웃이 발생하면 안드로이드가 화면을 그리는 데 시간이 더 소요된다.

ContraintLayout을 사용하면 중첩 레이아웃이 발생하지 않아서 속도가 빨라진다.

테이블 레이아웃

- 테이블 레이아웃은 자식 뷰들을 테이블 형태로 배치한다.
- 하나의 테이블은 여러 개의 TableRow 객체로 이루어지고 하나의 TableRow 안에는 여러 개의 셀(cell)들이 들어간다.



예제: 테이블 레이아웃

- 예를 들어서 3행×2열의 테이블을 작성하여 보자.

예제: 테이블 레이아웃

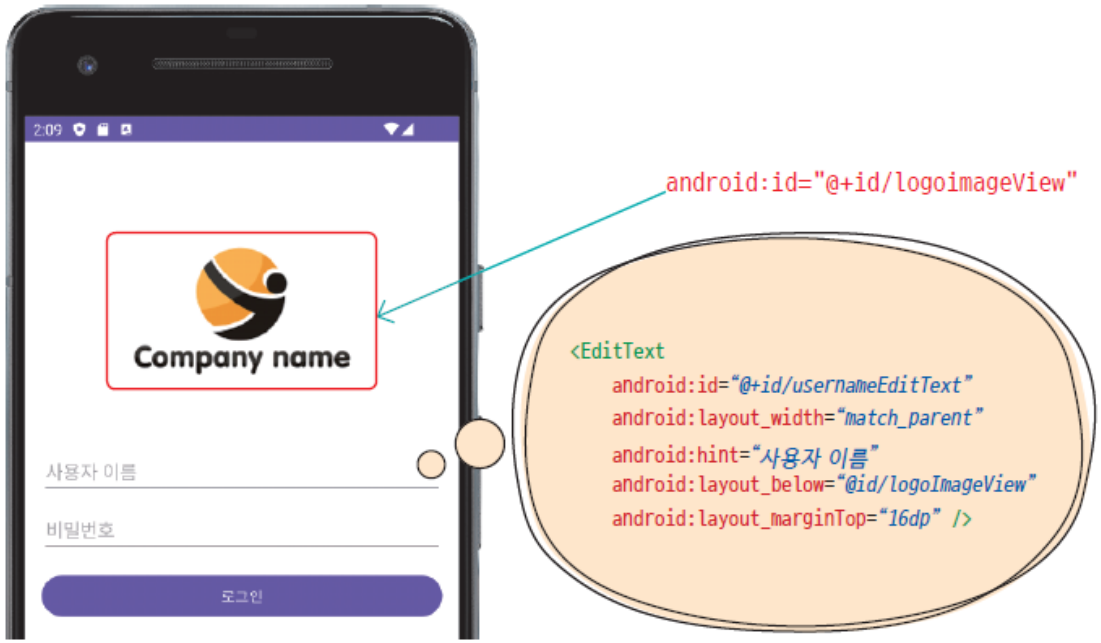
activity_main.xml

```
<TableLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

    <TableRow>
        <TextView android:text="주소"/>
        <EditText android:text="서울시 종로구 120"/>
    </TableRow>

    <TableRow>
        <TextView android:text="이름"/>
        <EditText android:text="홍길동"/>
    </TableRow>

    <TableRow>
        <Button android:text="저장"/>
        <Button android:text="취소"/>
    </TableRow>
</TableLayout>
```

테이블의 하나의 행

상대적 레이아웃

- 상대적 레이아웃은 자식 뷰의 위치를 부모 뷰나 다른 자식 뷰들에 상대적으로 지정하는 방법이다.

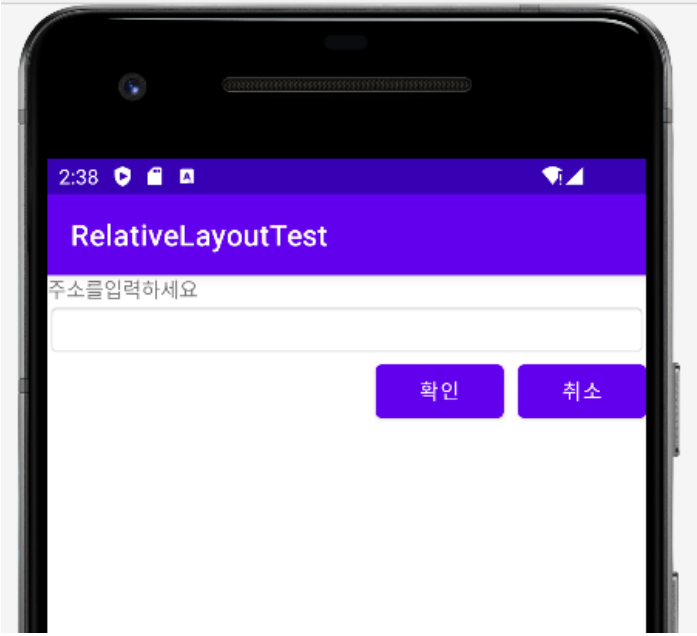


상대적 레이아웃 속성

속성	설명
layout_above	만약 true이면 현재 뷰의 하단을 기준 뷰의 위에 일치시킨다.
layout_below	현재 뷰의 상단을 기준 뷰의 하단에 위치시킨다.
layout_centerHorizontal	수평으로 현재 뷰의 중심을 부모와 일치시킨다.
layout_centerInParent	부모의 중심점에 현재 뷰를 위치시킨다.
layout_centerVertical	수직으로 현재 뷰의 중심을 부모와 일치시킨다.
layout_toLeftOf	현재 뷰의 우측단을 기준 뷰의 좌측단에 위치시킨다.
layout_toRightOf	현재 뷰의 좌측단을 기준 뷰의 우측단에 위치시킨다.

예제: 상대적 레이아웃

- 다음의 예는 XML 파일과 실행 결과 화면을 보여준다. 기준이 되는 뷰는 @id/address와 같은 형식을 이용해 참조하는 것에 유의하라.



UI

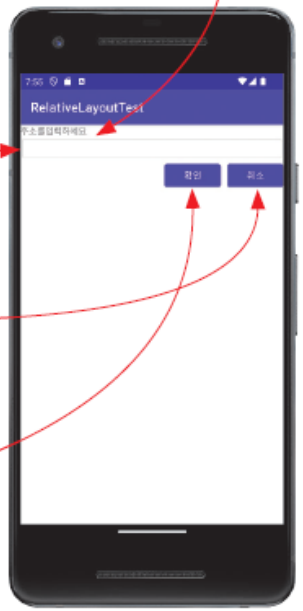
activity_main.xml

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

    <TextView
        android:id="@+id/address"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:text="주소를 입력하세요" />
    <EditText
        android:id="@+id/input"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/address" />
    <Button
        android:id="@+id/cancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/input"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip"
        android:text="취소" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/cancel"
        android:layout_alignTop="@id/cancel"
        android:text="확인" />
</RelativeLayout>
```

address 아래에 배치

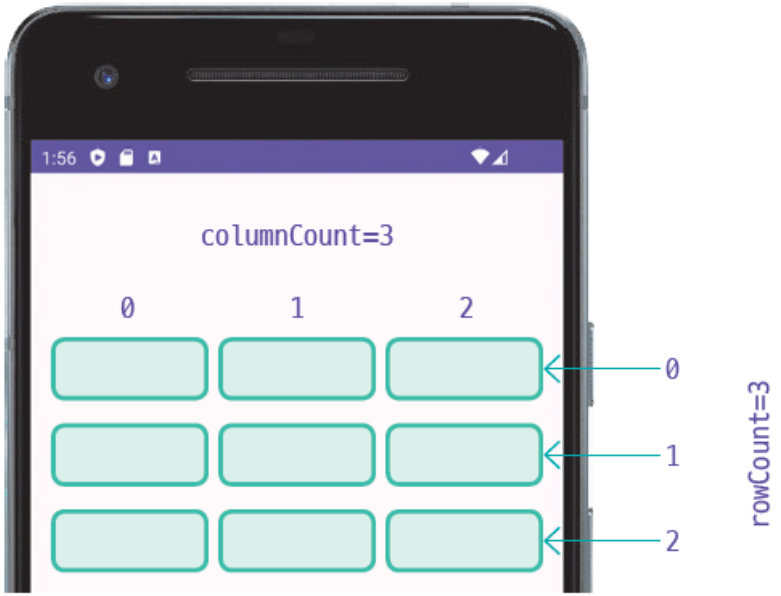
input 아래에 배치

cancel의 왼쪽에 배치



그리드 레이아웃(GridLayout)

- 그리드 형식의 행과 열로 위젯을 정렬하고 배치하는 데 사용된다. 이 레이아웃은 복잡한 UI 디자인을 구성할 때 특히 유용하다.



그리드 레이아웃(GridLayout)

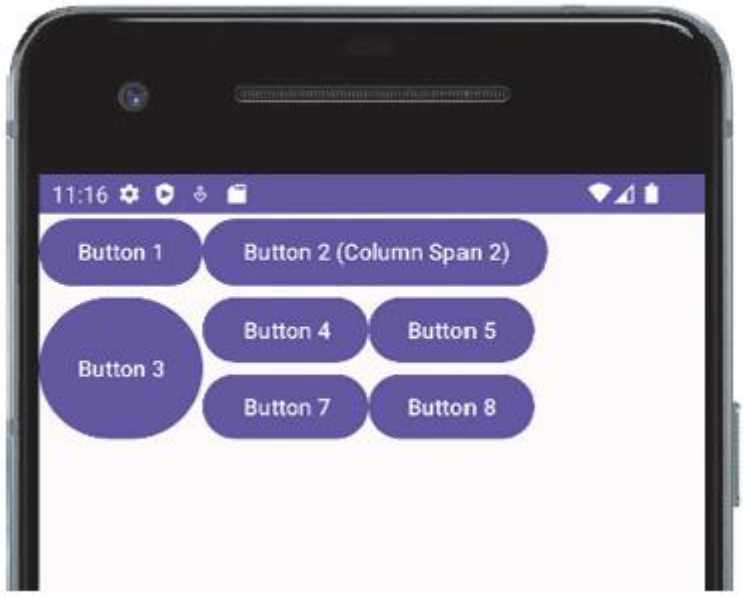
- android:rowCount 및 android:columnCount: android:rowCount는 그리드 레이아웃의 행 수를 설정하며, android:columnCount는 열 수를 설정한다.

- android:layout_row 및 android:layout_column: 각 위젯의 android:layout_row 속성은 해당 뷰가 그리드 레이아웃의 몇 번째 행에 위치할지를 지정한다. android:layout_column속성은 열 위치를 지정한다.

- android:layout_rowSpan 및 android:layout_columnSpan: android:layout_rowSpan 속성은 뷰가 여러 행을 차지할 때 사용된다.

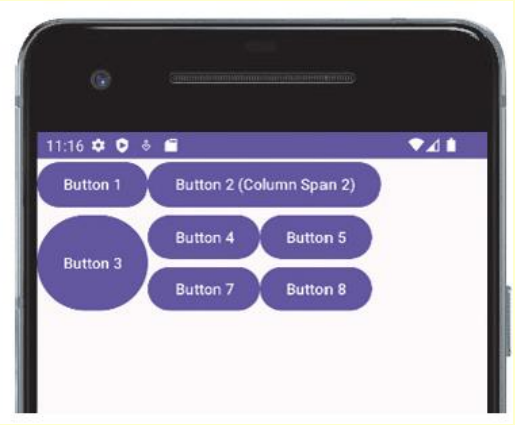
예제: 그리드 레이아웃 예제

- 다음 XML 코드는 3×3 그리드 레이아웃을 생성하고 그 안에 8개의 버튼을 배치한다.



예제: 그리드 레이아웃 예제

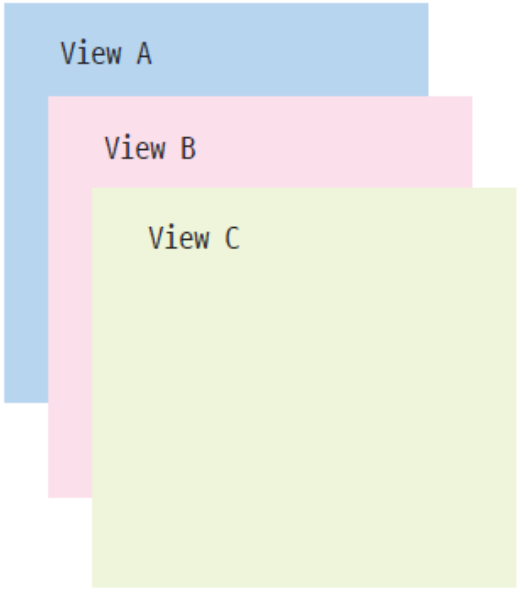
```
<GridLayout xmlns:android=http://schemas.android.com/apk/res/android>

   <Button       android:text="Button 1" />

   <Button
      android:layout_columnSpan="2"
      android:text="Button 2 (Column Span 2)" />

   <Button
      android:layout_rowSpan="2"
      android:layout_gravity = "fill_vertical"
      android:text="Button 3" />

   <Button       android:text="Button 4" />

   <Button       android:text="Button 5" />

   <Button       android:text="Button 7" />

   <Button       android:text="Button 8" />

</GridLayout>
```



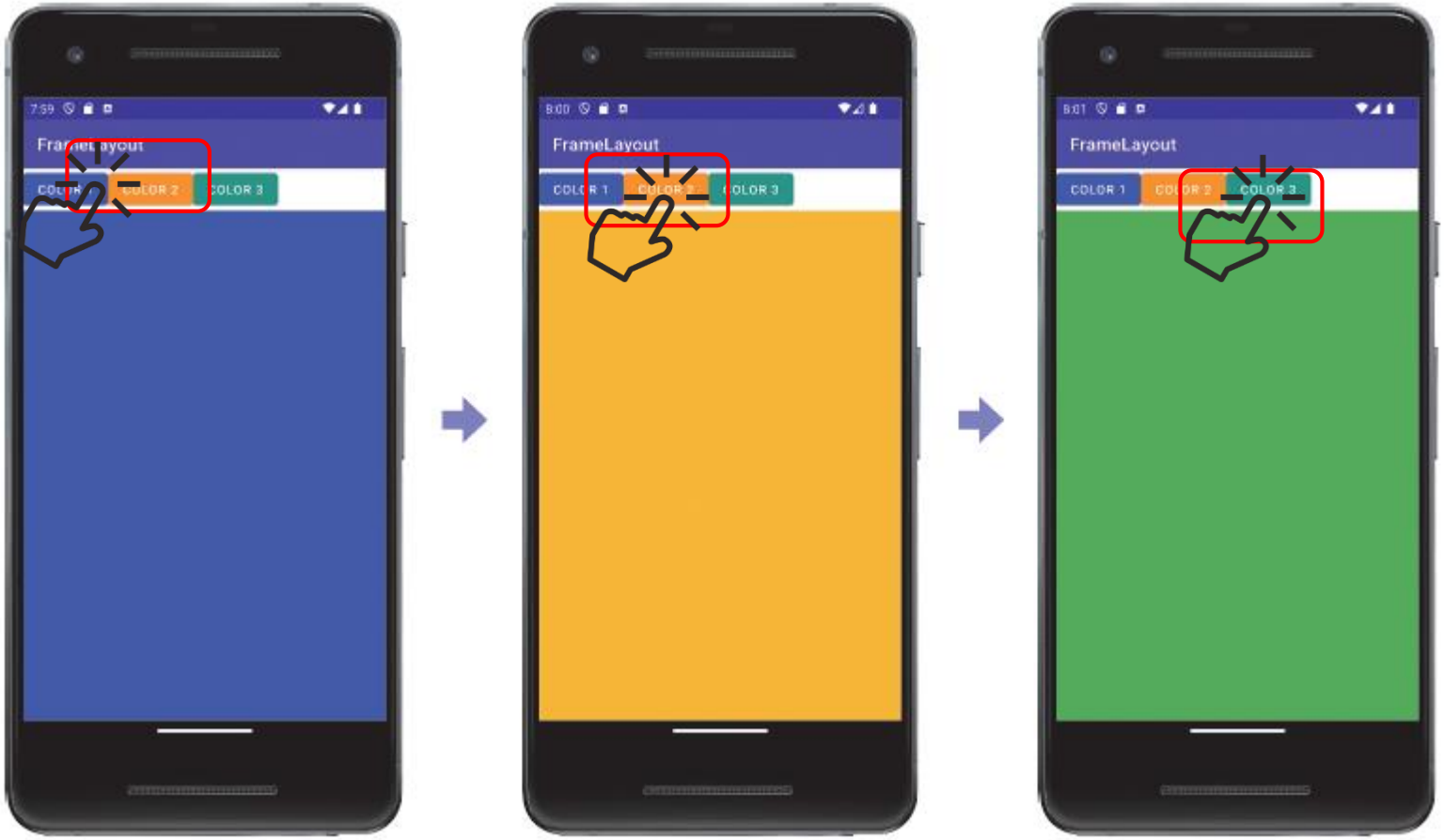
프레임 레이아웃

- 프레임 레이아웃 안에서 자식 뷰들은 등장하는 순서대로 화면에 표시된다.
- 만약 자식 뷰가 여러 개이면 이전에 추가된 자식 위에 새로운 자식이 중첩되어 그려진다.

```
</FrameLayout>
  <TextView id=A...>
  <TextView id=B...>
  <TextView id=C...>
</FrameLayout>
```



예제: 프레임 레이아웃


FrameLayout
COLOR 1
COLOR 2
COLOR 3
FrameLayout
COLOR 1
COLOR 2
COLOR 3
FrameLayout
COLOR 1
COLOR 2
COLOR 3

예제: 프레임 레이아웃

activity_main.xml

```
<LinearLayout >
    <LinearLayout   android:orientation="horizontal">
        <Button
            android:id="@+id/button1"
            android:onClick="onClick"
            android:text="COLOR 1" />

        <Button
            android:id="@+id/button2"
            android:onClick="onClick"
            android:text="COLOR 2" />

        <Button
            android:id="@+id/button3"
            android:onClick="onClick"
```

예제: 프레임 레이아웃

```
            android:text="COLOR 3" />
    </LinearLayout>

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

        <TextView
            android:id="@+id/view1"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="#3F51B5"></TextView>
        <TextView
            android:id="@+id/view2"> </TextView>
        <TextView
            android:id="@+id/view3"></TextView>
    </FrameLayout>
</LinearLayout>
```

프레임 레이아웃은 자식 뷰들을 중첩하여 배치한다.

예제: 프레임 레이아웃

MainActivity.java

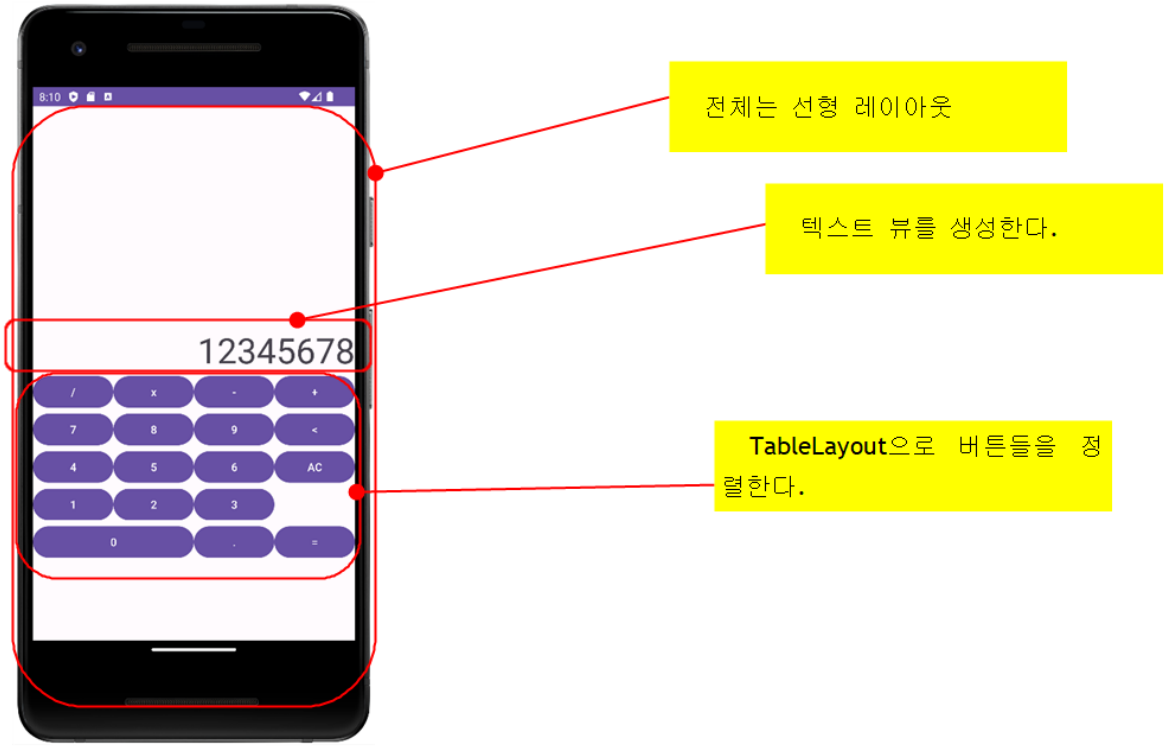
```
public class MainActivity extends AppCompatActivity {
    TextView tv1, tv2, tv3;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv1 = (TextView) findViewById(R.id.view1);
        tv2 = (TextView) findViewById(R.id.view2);
        tv3 = (TextView) findViewById(R.id.view3);
    }
```

예제: 프레임 레이아웃

```
    public void onClick(View view) {
        tv1.setVisibility(View.INVISIBLE);
        tv2.setVisibility(View.INVISIBLE);
        tv3.setVisibility(View.INVISIBLE);
        switch (view.getId()) {
            case R.id.button1:
                tv1.setVisibility(View.VISIBLE);
                break;
            case R.id.button2:
                tv2.setVisibility(View.VISIBLE);
                break;

            case R.id.button3:
                tv3.setVisibility(View.VISIBLE);
                break;
        }
    }
}
```

Lab: 계산기 앱 #2 작성

- 이번 실습에서는 다음과 같은 화면을 가지는 계산기 앱을 작성하여 보자. 전체적으로는 선형 레이아웃을 사용하고 버튼 부분은 테이블 레이아웃을 사용한다.



Lab: 계산기 앱 #2 작성

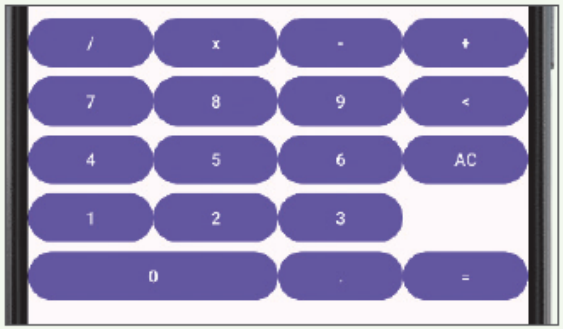
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView android:layout_width="match_parent"
        android:layout_height="match_parent" android:textSize="45dip"
        android:gravity="right|bottom" android:text="12345678"
        android:layout_weight="1"/>
```

Lab: 계산기 앱 #2 작성

activity_main.xml

```
<TableLayout android:layout_width="match_parent" android:layout_weight="1"
    android:layout_height="match_parent" android:stretchColumns="0,1,2,3"
    >
    <TableRow>
        <Button android:text="/"  />
        <Button android:text="x"  />
        <Button android:text="-"  />
        <Button android:text="+"  />
    </TableRow>
    <TableRow>
        <Button android:text="7"  />
        <Button android:text="8"  />
        <Button android:text="9"  />
        <Button android:text="&lt;"  />
    </TableRow>
    <TableRow>
        <Button android:text="4"  />
        <Button android:text="5"  />
        <Button android:text="6"  />
        <Button android:text="AC"  />
    </TableRow>
```

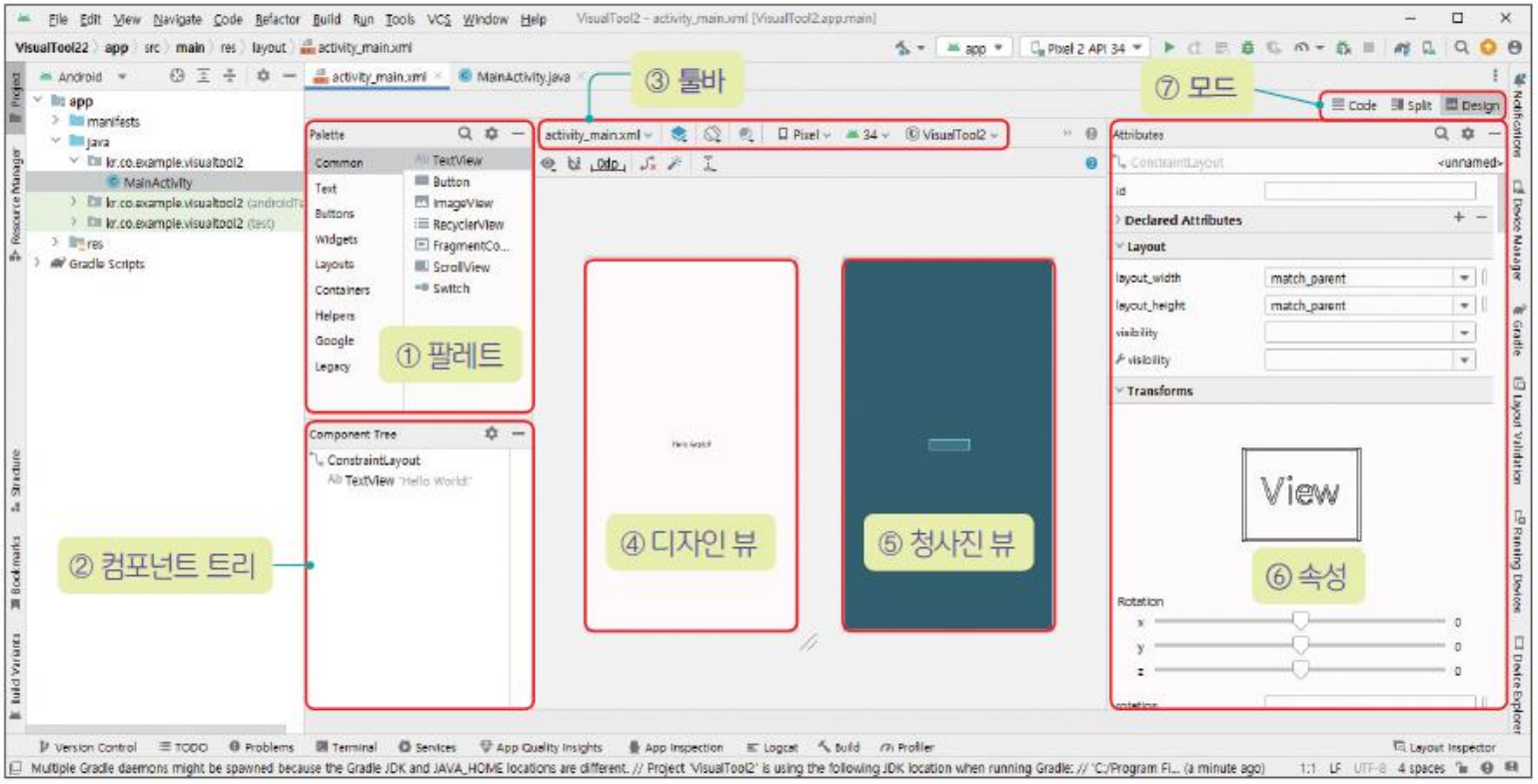


Lab: 계산기 앱 #2 작성

```
        <TableRow>
            <Button android:text="1"  />
            <Button android:text="2"  />
            <Button android:text="3"  />

        </TableRow>
        <TableRow>
            <Button android:text="0" android:layout_span="2" />
            <Button android:text="."  />
            <Button android:text="="  />
        </TableRow>
    </TableLayout>
</LinearLayout>
```

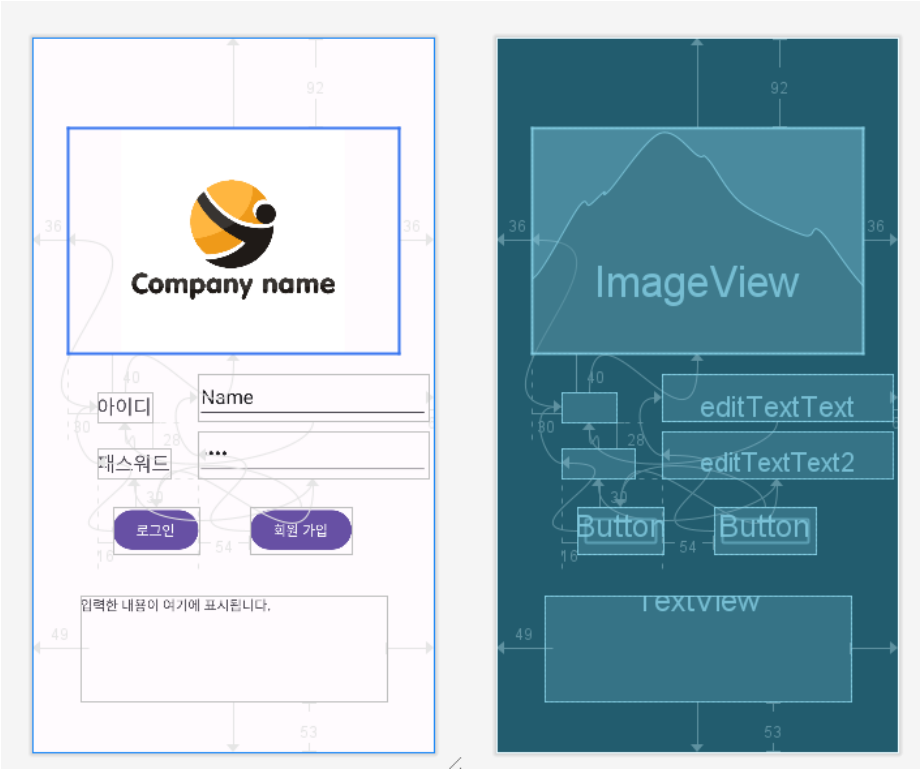
레이아웃 편집기

- 레이아웃 편집기에서는 모든 레이아웃과 위젯을 시각적으로 보면서 화면에 배치할 수 있다.



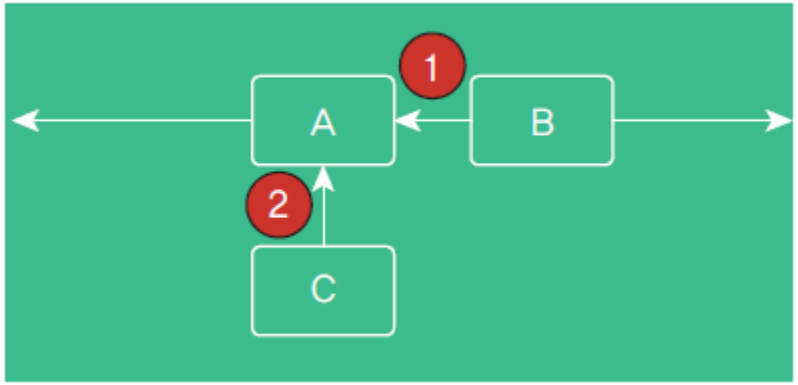
제약 레이아웃

- 제약 레이아웃은 위젯 사이의 제약 조건(constraint)을 사용하여 위젯을 배치하고 정렬한다.
- 제약 조건이란 하나의 위젯을 다른 위젯이나 컨테이너의 경계선에 붙이는 것이다.



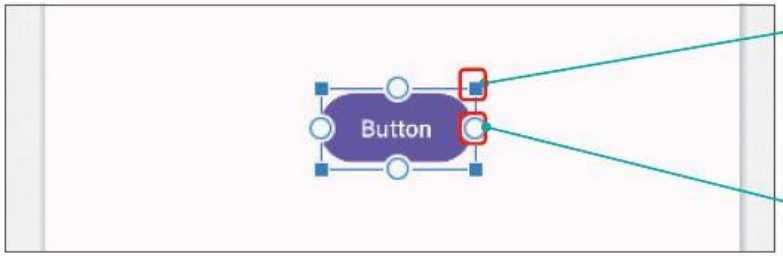
제약 조건이란?

- 제약(contraint)이란 두 위젯 사이의 연결이나 정렬을 의미한다. 즉 하나의 위젯을 다른 위젯에 연결하거나 정렬한다.
- 제약 레이아웃에서는 최소한 하나의 수평 및 하나의 수직 제약 조건을 정의하여 위젯을 배치한다



위젯의 핸들 설명

- 레이아웃 편집기에서 레이아웃 안의 위젯을 클릭하면 위젯 테두리에 나타나는 작은 점(핸들)이 있다.

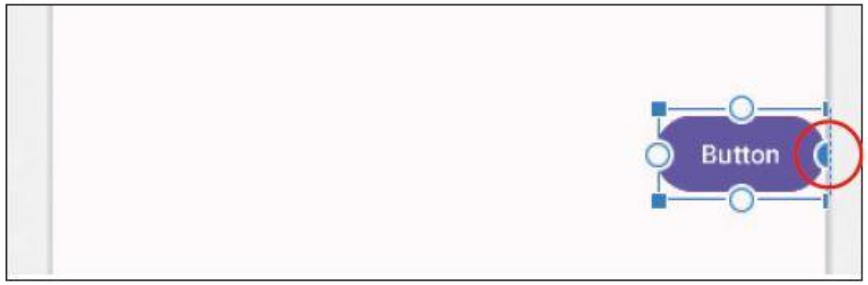


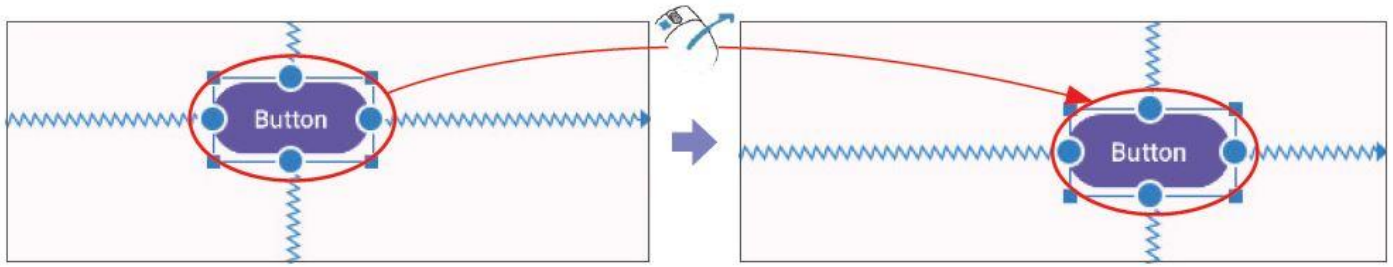
크기 조절 핸들: 마우스로 드래그하면 위젯의 크기를 조정할 수 있다.

측면 핸들: 위젯의 상하좌우에 제약 조건을 붙이는 데 사용한다. 마우스로 드래그하여 다른 위젯이나 경계선에 붙일 수 있다.

위젯의 위치 설정하기

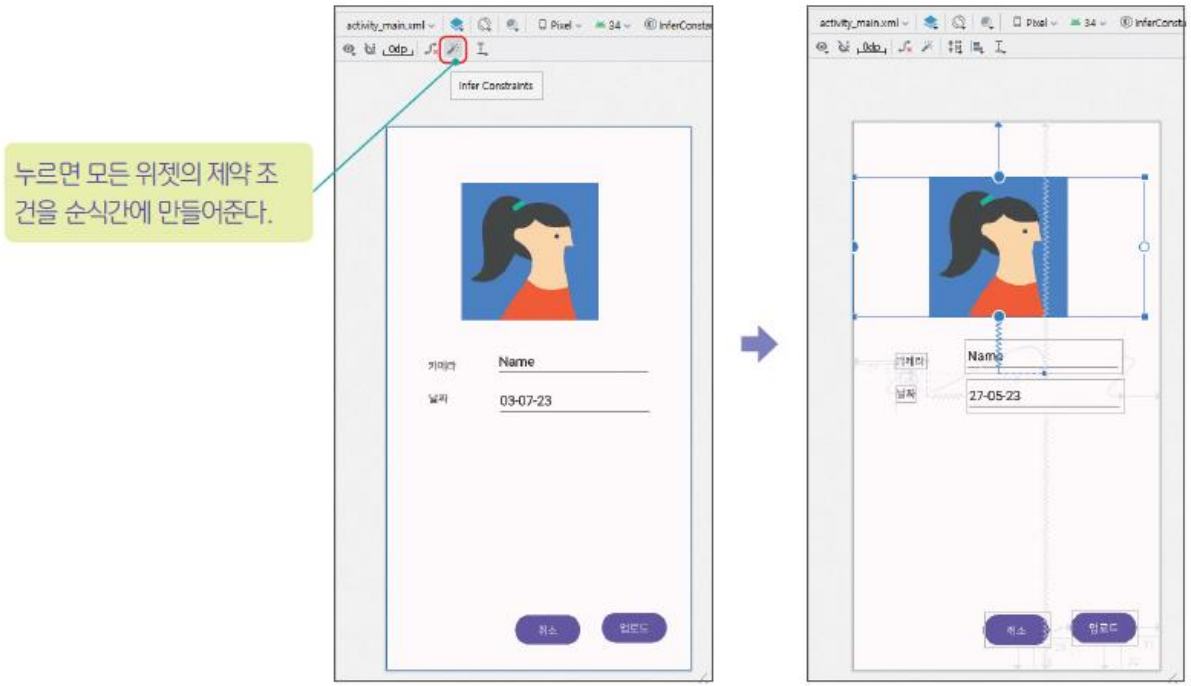
- 위젯의 위치를 설정하기 위해서는 레이아웃 편집기에서 측면 핸들을 마우스로 드래그하여서 원하는 곳에 붙이면 된다.





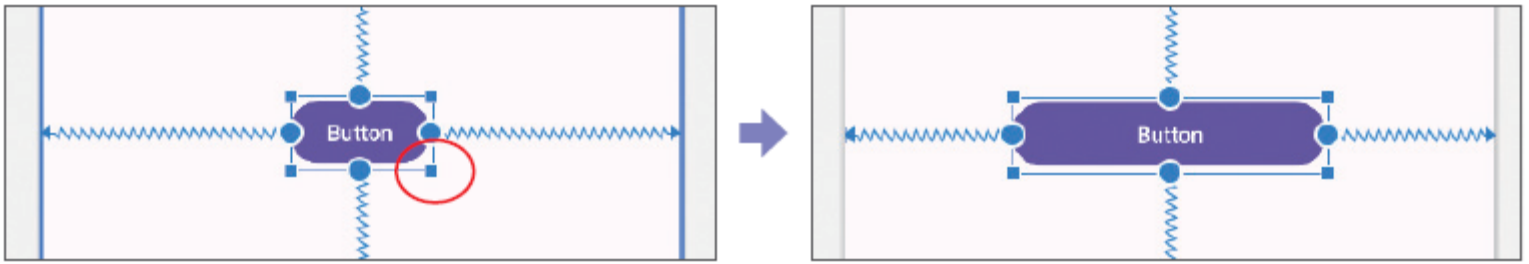
[Infer Contraints] 아이콘 사용하기

- 이 버튼을 클릭하면 사용자가 화면에서 위젯을 배치한 상태로 제약 조건을 만들어준다.

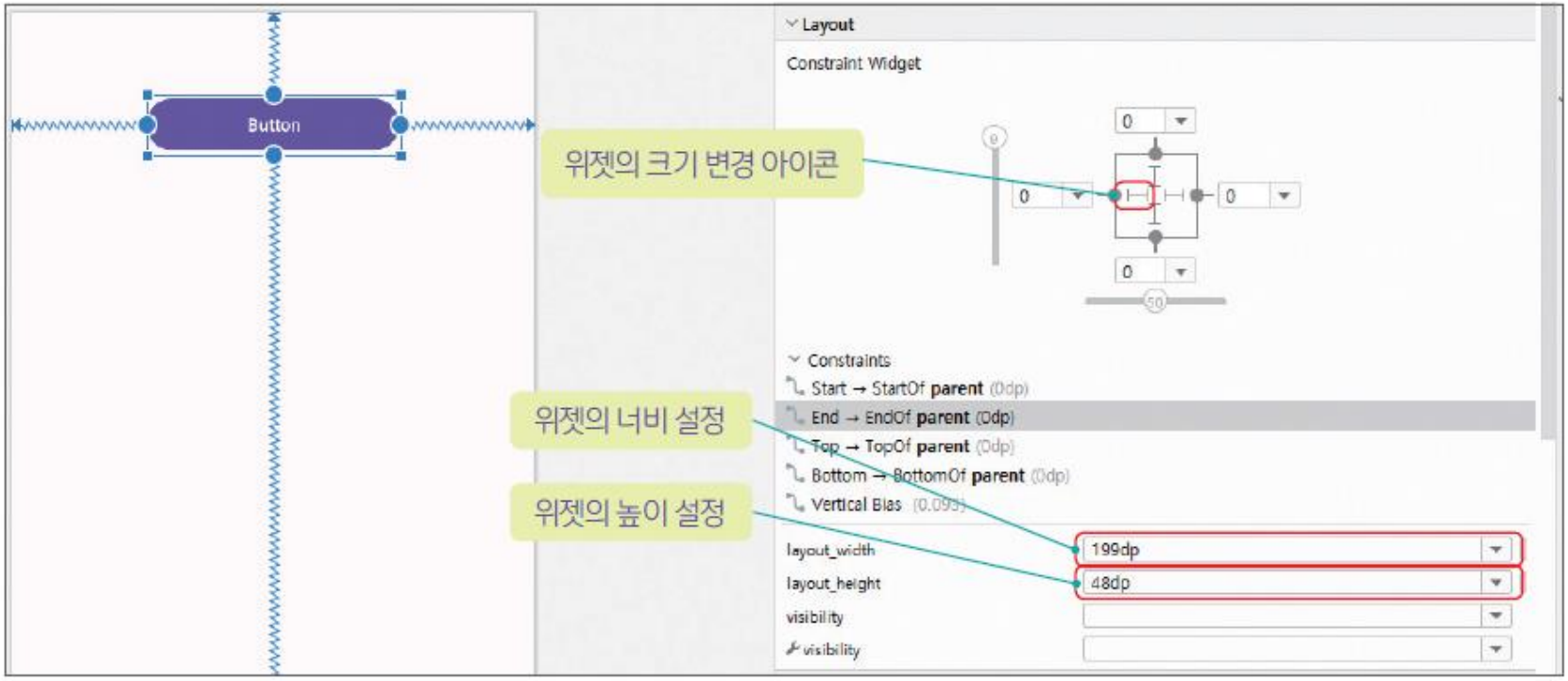


위젯의 크기 설정하기

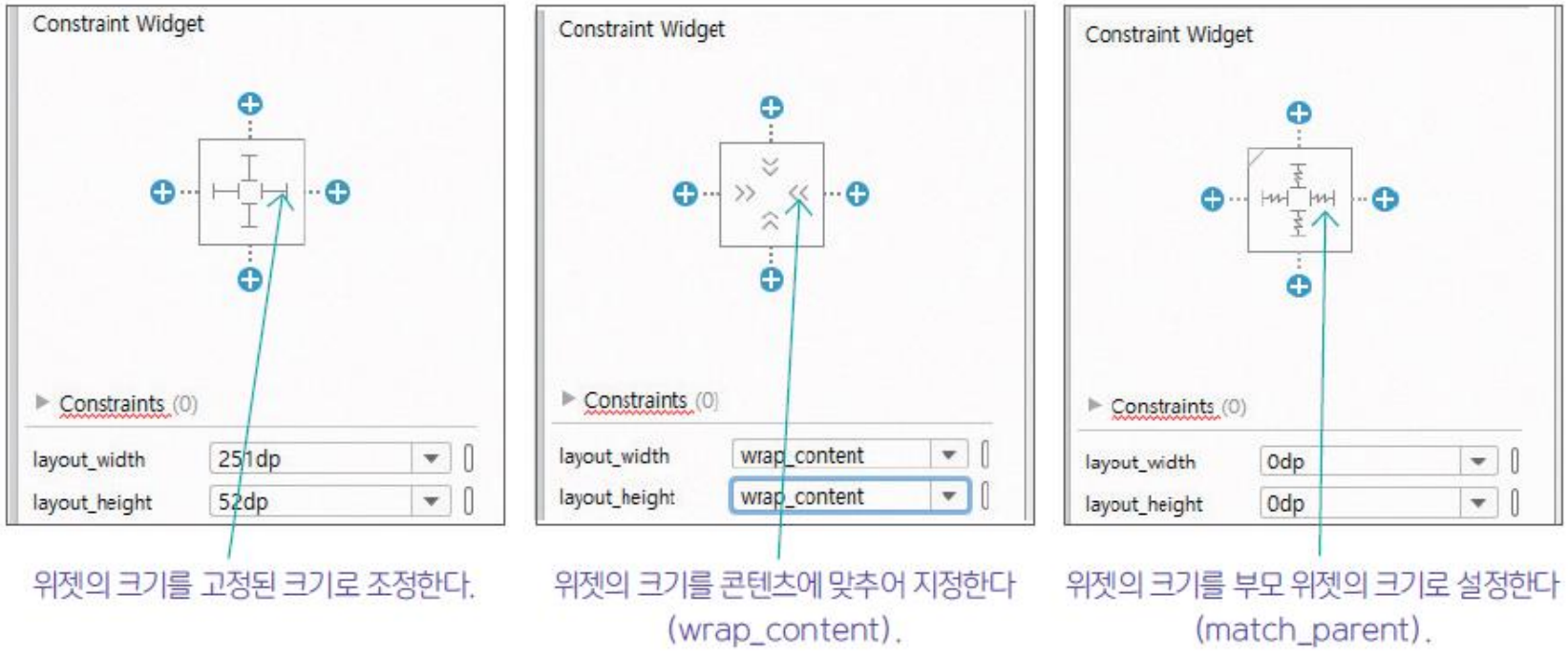
- 위젯의 크기는 기본적으로 위젯의 사각형 핸들을 마우스로 잡아서 늘리면 위젯의 크기는 고정된 크기로 설정된다.



위젯의 크기 설정하기


Button
위젯의 크기 변경 아이콘
위젯의 너비 설정
위젯의 높이 설정
Layout
Constraint Widget
Constraints
Start → StartOf parent (0dp)
End → EndOf parent (0dp)
Top → TopOf parent (0dp)
Bottom → BottomOf parent (0dp)
Vertical Bias
layout_width
199dp
layout_height
48dp
visibility
visibility

위젯의 크기 설정하기

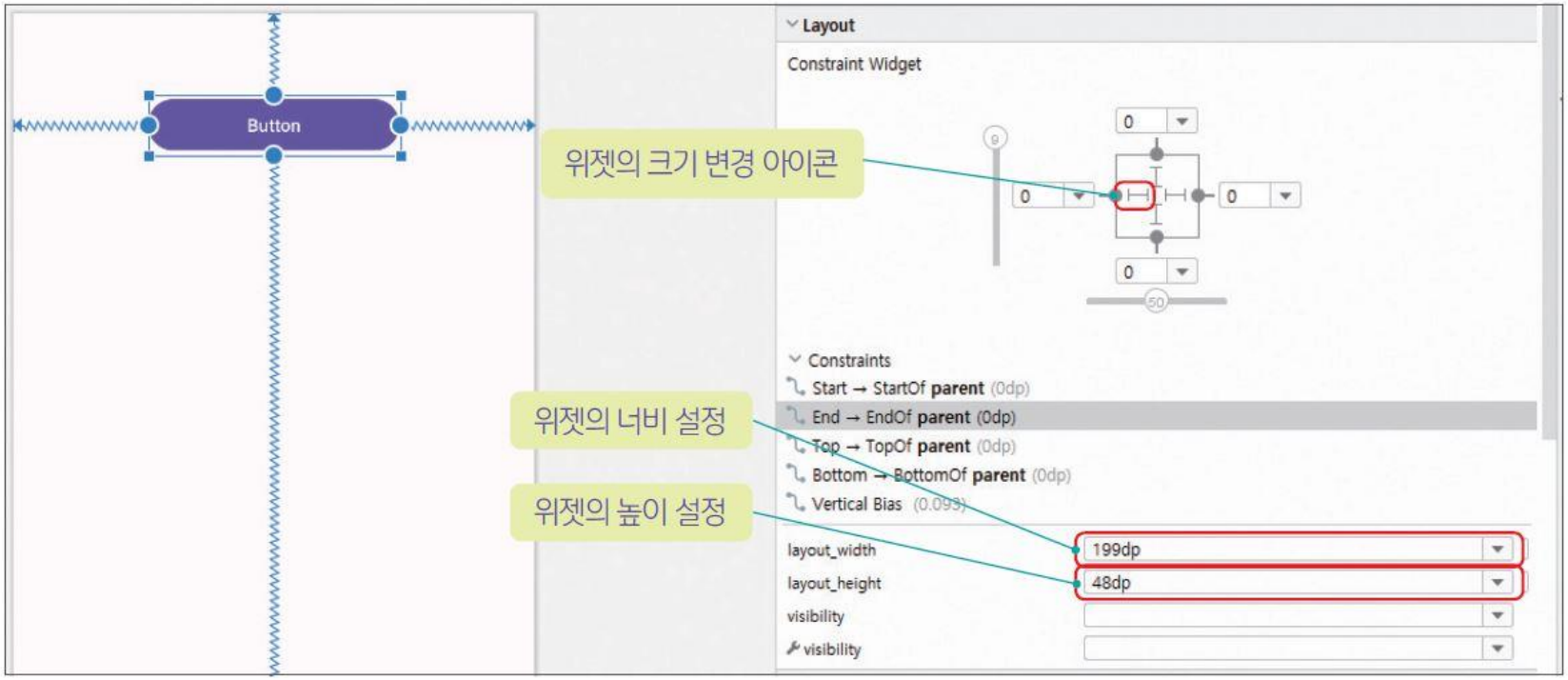

Constraint Widget
Constraints (0)
layout_width
251dp
layout_height
52dp
위젯의 크기를 고정된 크기로 조정한다.
Constraint Widget
Constraints (0)
layout_width
wrap_content
layout_height
wrap_content
위젯의 크기를 콘텐츠에 맞추어 지정한다 (wrap_content).
Constraint Widget
Constraints (0)
layout_width
0dp
layout_height
0dp
위젯의 크기를 부모 위젯의 크기로 설정한다 (match_parent).

위젯의 크기 설정하기

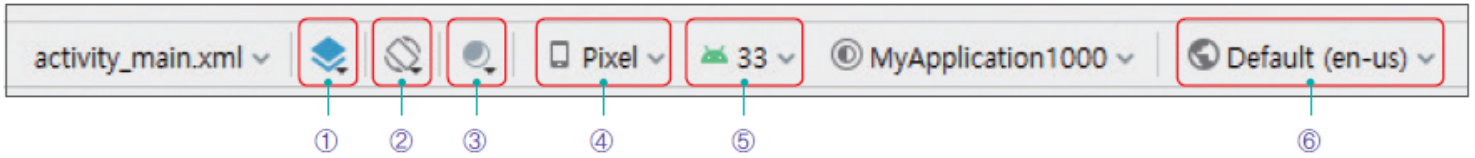
- 아이콘의 의미

기호	설명
	위젯의 크기는 고정된 크기가 된다. (fixed)
	위젯의 크기는 콘텐츠의 크기에 맞게 조정된다. (wrap_content)
	위젯은 부모 위젯의 크기를 전부 차지한다(최대 크기). (match_parent)

마진 조정


Button
위젯의 크기 변경 아이콘
위젯의 너비 설정
위젯의 높이 설정
Layout
Constraint Widget
0
0
0
0
Constraints
Start → StartOf parent (0dp)
End → EndOf parent (0dp)
Top → TopOf parent (0dp)
Bottom → BottomOf parent (0dp)
Vertical Bias (0.093)
layout_width
199dp
layout_height
48dp
visibility
visibility

툴바 설명



① 레이아웃 편집기에서 레이아웃을 표시할 방법을 선택한다. 레이아웃의 렌더링된 미리보기를 표시하려면 [Design]을 선택한다. 각 뷰의 윤곽선만 표시하려면 Blueprint를 선택한다.

② 화면 가로 모드 방향과 세로 모드 방향 중에서 선택한다.

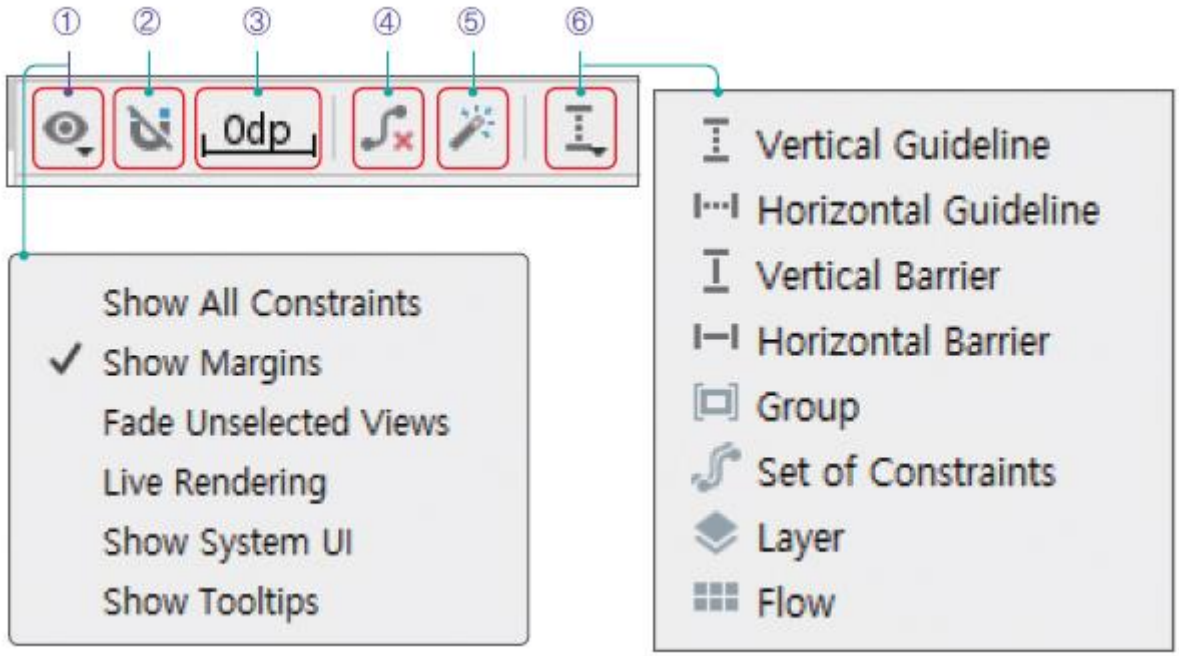
③ 기기 유형(스마트폰/태블릿, Android TV 또는 Wear OS) 및 화면 구성(크기 및 밀도)을 선택한다.

④ 레이아웃으로 미리 볼 Android의 버전을 선택한다.

⑤ 미리보기에 적용할 UI 테마를 선택한다.

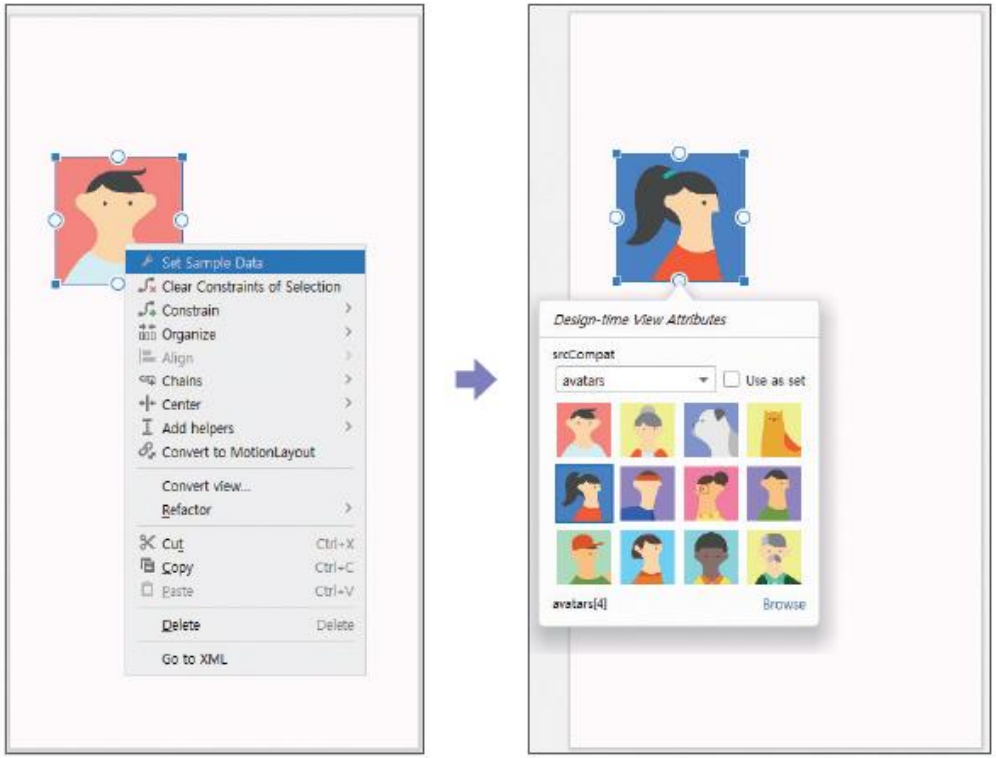
⑥ 미리보기에 사용할 언어와 로케일을 선택한다. 이 목록에는 문자열 리소스에서 사용할 수 있는 언어만 표시된다.

툴바 설명


①
②
③
④
⑤
⑥
0dp
Show All Constraints
Show Margins
Fade Unselected Views
Live Rendering
Show System UI
Show Tooltips
Vertical Guideline
Horizontal Guideline
Vertical Barrier
Horizontal Barrier
Group
Set of Constraints
Layer
Flow

위젯에 샘플 데이터 추가

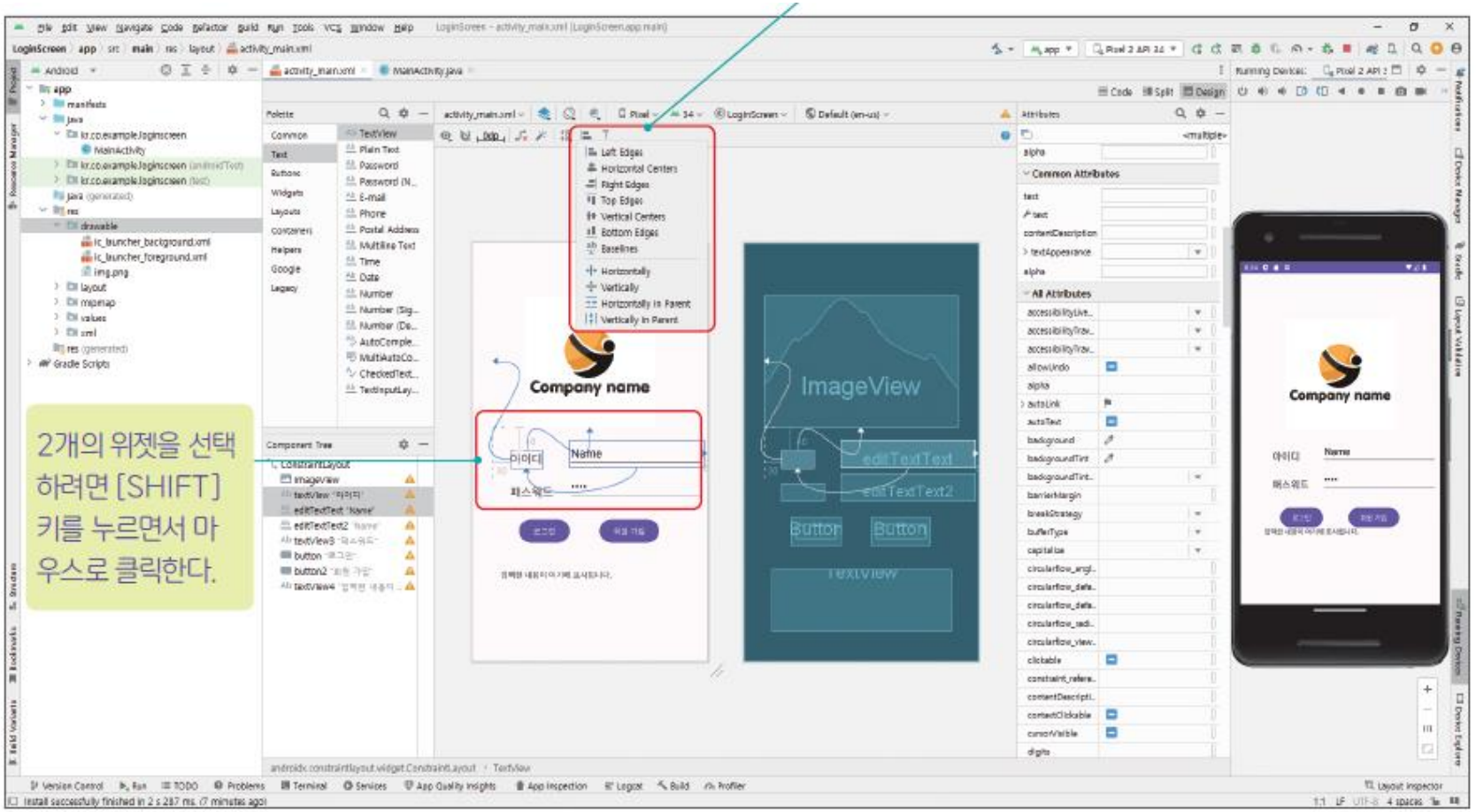
- 안드로이드 스튜디오 3.2 이상에서는 레이아웃 에디터 내에서 샘플 미리보기 데이터를 텍스트 뷰나 이미지 뷰에 추가할 수 있다



예제: 로그인 화면 만들기

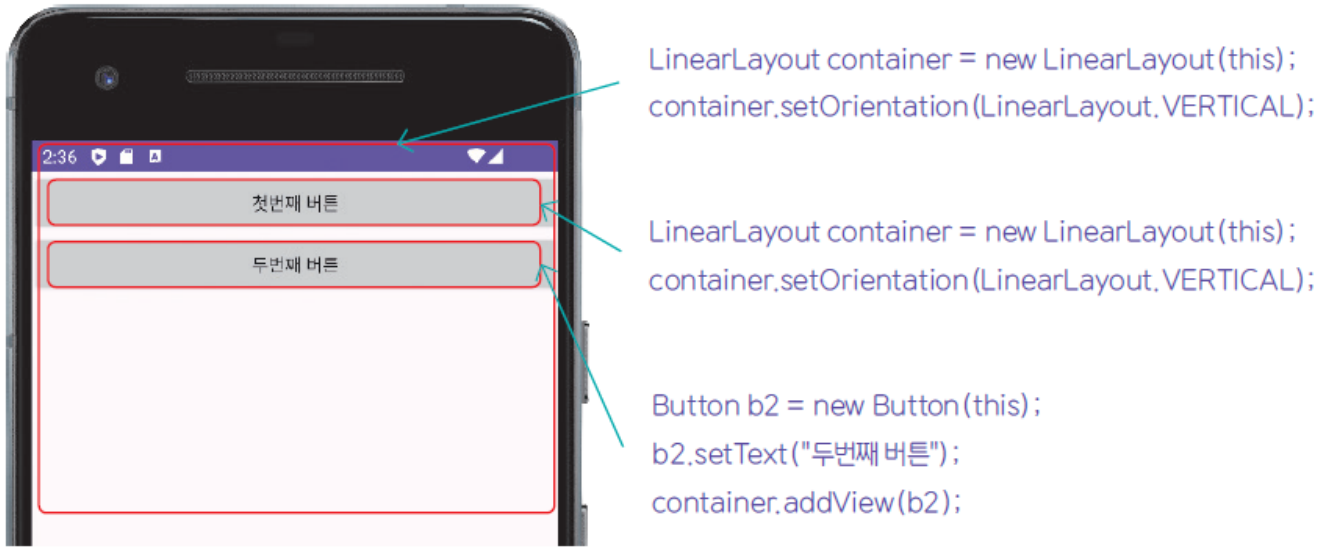
- 다음과 같은 로그인 화면을 작성해보자. 레이아웃 편집기와 제약 레이아웃을 사용한다.

예제: 로그인 화면 만들기


2개의 위젯을 선택하려면 [SHIFT] 키를 누르면서 마우스로 클릭한다.
Company name
Name
ImageView
editTextText
editTextText2
Button
Button
TextView

코드로 레이아웃 만들기

- 우리는 코드를 이용하여 레이아웃과 뷰들을 생성할 수 있다. 뷰의 속성도 코드로 설정한다.
- 예를 들어서 2개의 버튼을 만들고 버튼에 표시되는 텍스트를 설정하는 코드를 작성해보자.



예제: 코드로 레이아웃 만들기

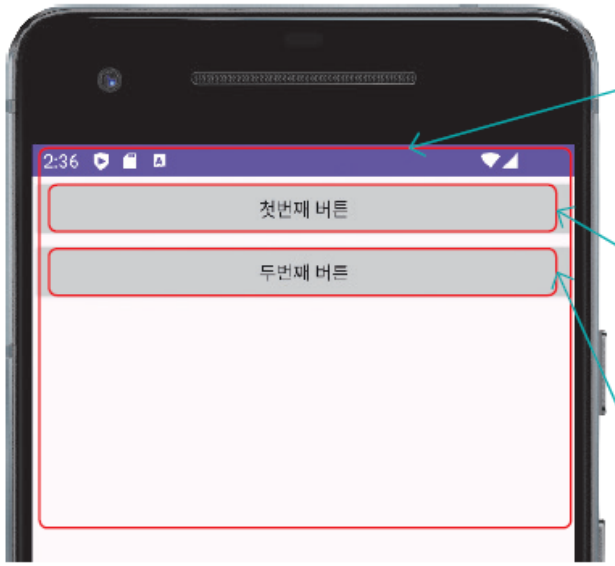
MainActivity.java

```
package.kr.co.company.userinterface2;

public class MainActivity extends AppCompatActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      LinearLayout container = new LinearLayout(this);
      container.setOrientation(LinearLayout.VERTICAL);

      Button b1 = new Button(this);
      b1.setText("첫번째 버튼");
      container.addView(b1);

      Button b2 = new Button(this);
      b2.setText("두번째 버튼");
      container.addView(b2);

      setContentView(container);
  }
}
```

선형 레이아웃을 생성한다.

버튼을 선형 레이아웃에 추가한다.

실행결과

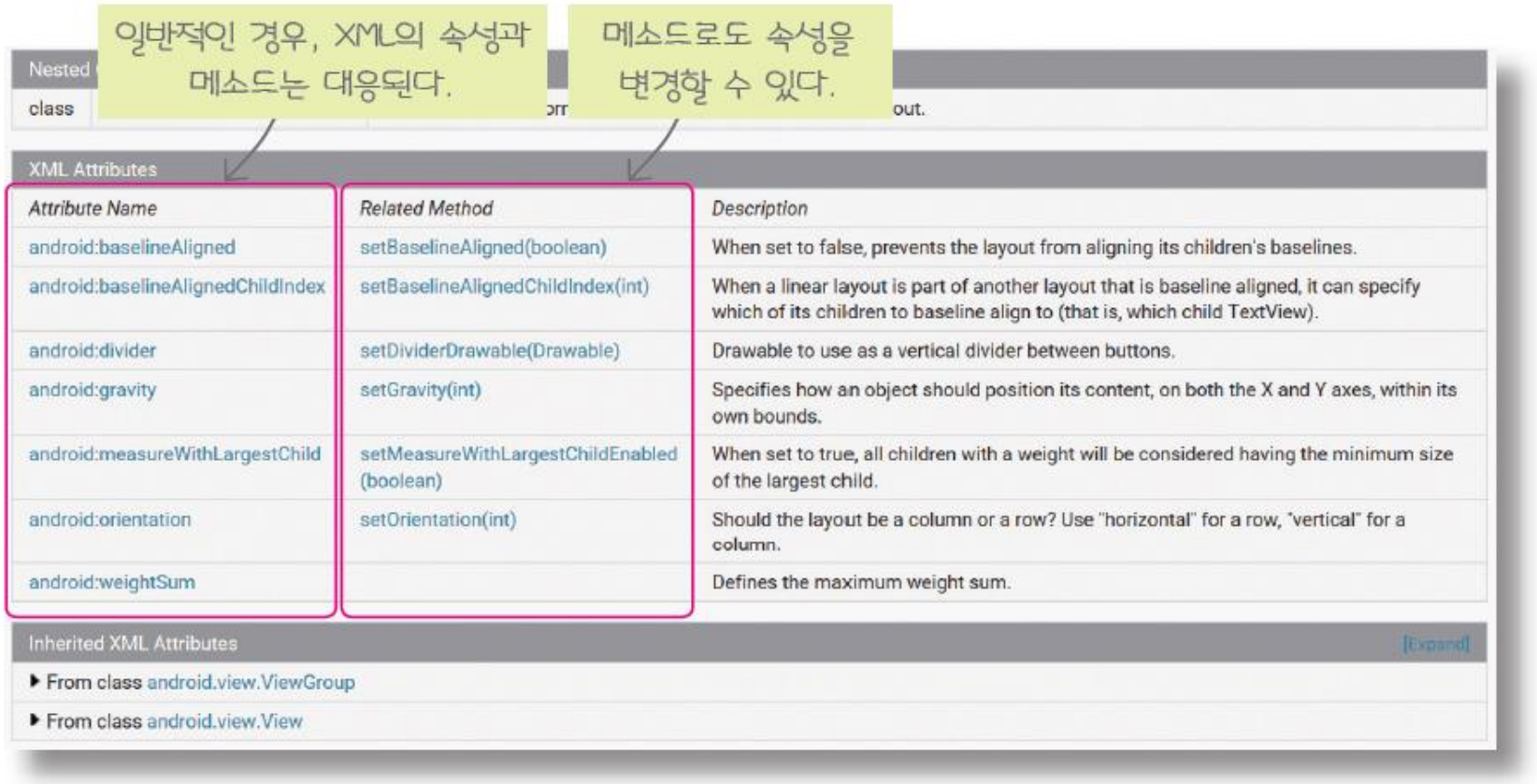



```
LinearLayout container = new LinearLayout(this);
container.setOrientation(LinearLayout.VERTICAL);
```

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

```
Button b2 = new Button(this);
b2.setText("두번째 버튼");
container.addView(b2);
```

코드로 속성 변경하기


일반적인 경우, XML의 속성과 메소드는 대응된다.
메소드로도 속성을 변경할 수 있다.
Nested
class
out.
XML Attributes
Attribute Name
Related Method
Description
android:baselineAligned
setBaselineAligned(boolean)
When set to false, prevents the layout from aligning its children's baselines.
android:baselineAlignedChildIndex
setBaselineAlignedChildIndex(int)
When a linear layout is part of another layout that is baseline aligned, it can specify which of its children to baseline align to (that is, which child TextView).
android:divider
setDividerDrawable(Drawable)
Drawable to use as a vertical divider between buttons.
android:gravity
setGravity(int)
Specifies how an object should position its content, on both the X and Y axes, within its own bounds.
android:measureWithLargestChild
setMeasureWithLargestChildEnabled (boolean)
When set to true, all children with a weight will be considered having the minimum size of the largest child.
android:orientation
setOrientation(int)
Should the layout be a column or a row? Use "horizontal" for a row, "vertical" for a column.
android:weightSum
Defines the maximum weight sum.
Inherited XML Attributes
[Expand]
From class android.view.ViewGroup
From class android.view.View

예제: 코드로 속성 변경

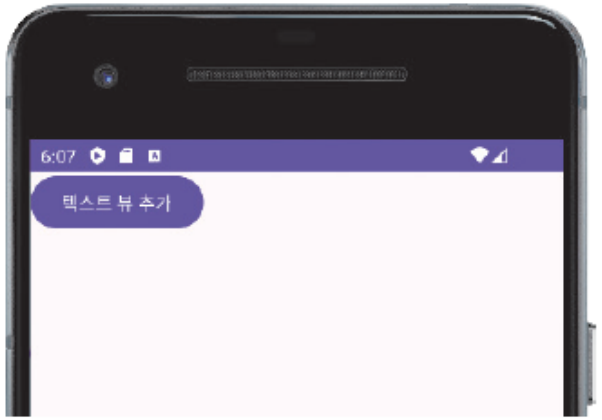
- 버튼을 클릭하면 동적으로 텍스트 뷰를 추가하는 경우이다.

예제: 코드로 속성 변경

activity_main.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mainLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/addButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="텍스트 뷰 추가" />
```

id를 부여한다.



예제: 코드로 속성 변경

MainActivity.java

```
...
public class MainActivity extends AppCompatActivity {

    private LinearLayout mainLayout;
    private Button addButton;

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

        mainLayout = findViewById(R.id.mainLayout);
        addButton = findViewById(R.id.addButton);
```

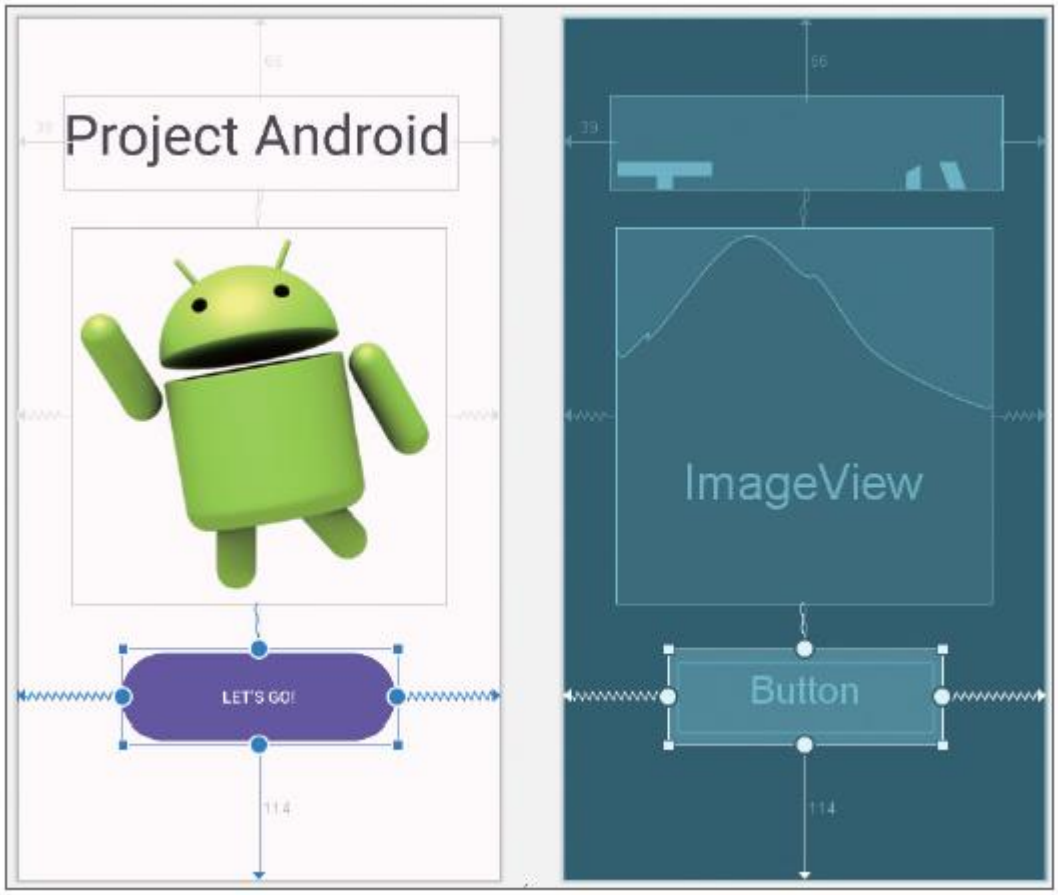
예제: 코드로 속성 변경

```
        addButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 텍스트 뷰를 동적으로 생성하고 설정
                TextView textView = new TextView(MainActivity.this);
                textView.setText("동적으로 추가된 텍스트 뷰");

                // 레이아웃 파라미터를 설정하여 레이아웃에 추가
                LinearLayout.LayoutParams layoutParams =
                    new LinearLayout.LayoutParams(
                        LinearLayout.LayoutParams.WRAP_CONTENT,
                        LinearLayout.LayoutParams.WRAP_CONTENT );
                textView.setLayoutParams(layoutParams);

                mainLayout.addView(textView);
            }
        });
    }
}
```

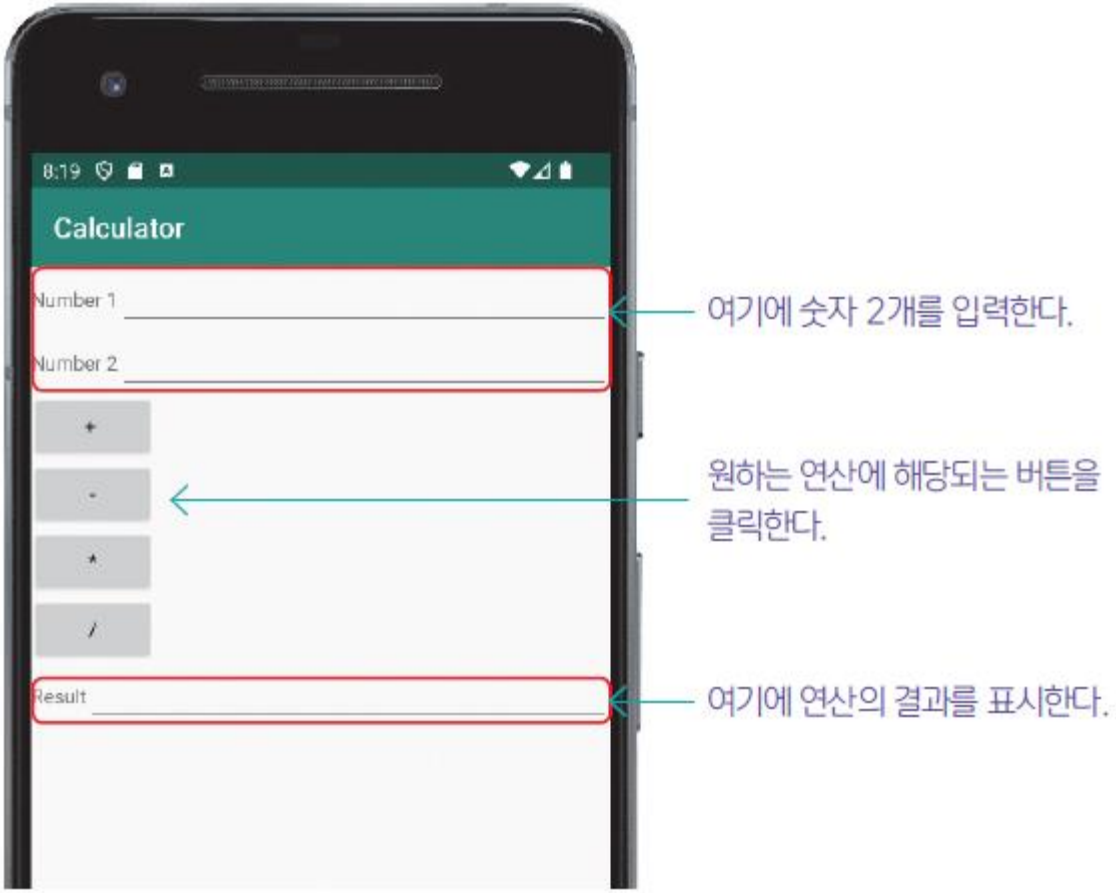
예제: 코드로 속성 변경

Coding Challenge:
레이아웃 편집기 사용하기

레이아웃 편집기의 연습을 위하여 오직 레이아웃 편집기만을 사용해보자.



Coding Challenge: 계산기 앱 #3 작성



Q & A