02. 레이아웃과 기본 위젯 사용하기

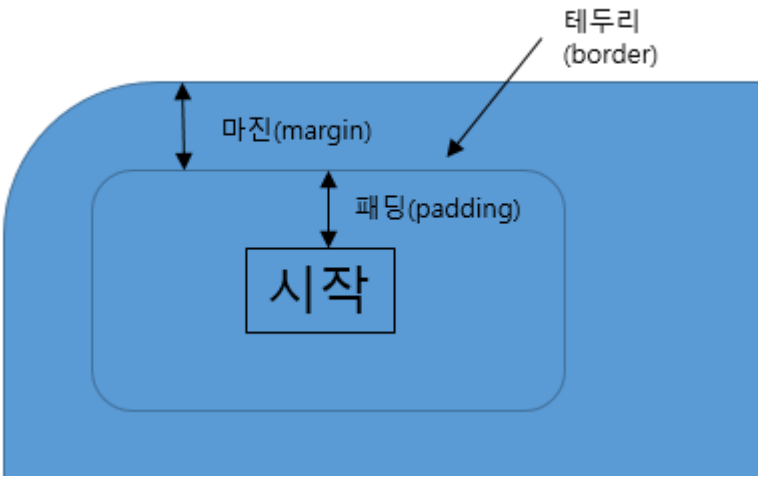
02-1. 대표적인 레이아웃 살펴보기

- **대표적인 레이아웃 다섯 가지**

레이아웃 이름	설 명
제약 레이아웃 (ConstraintLayout)	제약 조건(Constraint) 기반 모델 제약 조건을 사용해 화면을 구성하는 방법 안드로이드 스튜디오에서 자동으로 설정하는 디폴트 레이아웃
리니어 레이아웃 (LinearLayout)	박스(Box) 모델 한 쪽 방향으로 차례대로 뷰를 추가하며 화면을 구성하는 방법 뷰가 차지할 수 있는 사각형 영역을 할당
상대 레이아웃 (RelativeLayout)	규칙(Rule) 기반 모델 부모 컨테이너나 다른 뷰와의 상대적 위치로 화면을 구성하는 방법
프레임 레이아웃 (FrameLayout)	싱글(Single) 모델 가장 상위에 있는 하나의 뷰 또는 뷰그룹만 보여주는 방법 여러 개의 뷰가 들어가면 중첩하여 쌓게 됨. 가장 단순하지만 여러 개의 뷰를 중첩한 후 각 뷰를 전환하여 보여주는 방식으로 자주 사용함
테이블 레이아웃 (TableLayout)	격자(Grid) 모델 격자 모양의 배열을 사용하여 화면을 구성하는 방법 HTML에서 많이 사용하는 정렬 방식과 유사하지만 많이 사용하지는 않음

> 리니어 레이아웃
>
> - 가로 방향 : **Horizontal**
> - 세로 방향 : **Vertical**

뷰의 영역



- **뷰의 테두리(Border)**
- **테두리 바깥쪽 공간(Margin)**
- **테두리 안쪽 공간(Padding)** : 뷰의 내용물(Contents)까지의 공간

뷰의 margin / padding	속성
margin	layout_margin layout_marginTop layout_marginBottom layout_marginLeft layout_marginRight
padding	padding paddingTop paddingBottom paddingLeft paddingRight

뷰의 배경색

- **XML 레이아웃에서 색상을 지정할 때는 # 기호를 앞에 붙인 후 ARGB(A:Alpha, R:Red, G:Green, B:Blue)의 순서대로 색상의 값을 기록한다.**

  예시)

  ```
  16진수로 나타낸다.
  #ff0000     // 빨간색
  #00ff00     // 초록색
  #88ff00     // 반투명 빨간색
  ```

- **이미지를 배경으로 지정할 수도 있다.**

: background 속성에 이미지 리소스의 위치를 값으로 설정하면 됩니다. 안드로이드의 미지 리소스는 /res/drawable 폴더에 들어가며 만약 house.png라는 이미지 파일이 있다면 파일 탐색기에서 프로젝트 폴더 밑의 /res/drawable 폴더를 찾는다. 그런 다음 해당 폴더에 이미지 파일을 복사한 후 XML 레이아웃 파일에 들어있는 뷰의 background 속성을 다음과 같이 설정하면 된다.

```
andorid:background="@drawable/house"
```

02-2. 리니어(Linear) 레이아웃 사용하기

방향 설정하기

- 한 방향으로 뷰를 쌓는 방법(속성:**orientation**)
  - 가로 : **horizontal**
  - 세로 : **vertical**

실습

레이아웃 변경 및 vertical로 방향 설정

activity_main.xml

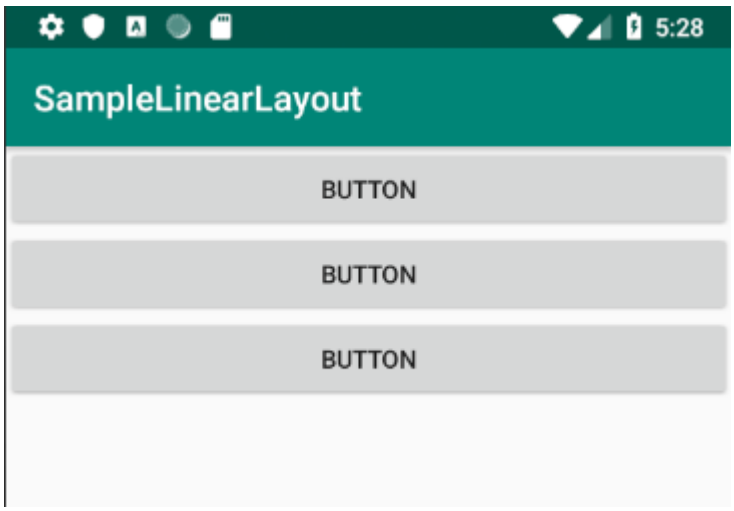
```
// LinearLayout 으로 변경
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"  // vertical(세로)로 방향 설정
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</LinearLayout>
```

버튼 세개 추가

activity_main.xml

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

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />
</LinearLayout>
```

> 세로 방향으로 스택 처럼 위에서부터 버튼이 하나씩 추가된다.



manifests에서 처음 보이는 액티비티 변환

AndroidManifest.xml 파일을 열고 액티비티를 위해 들어 있는 <activity> 태그의 android:name 속성 값을 바꾸면 지정한 액티비티 화면을 볼 수 있다.

- **예시**

```
// MainActivity2.java 파일에서 지정한 화면 보기
<activity android:name=".MainActivity2"></activity>
```

hrizontal로 레이아웃 방향 설정

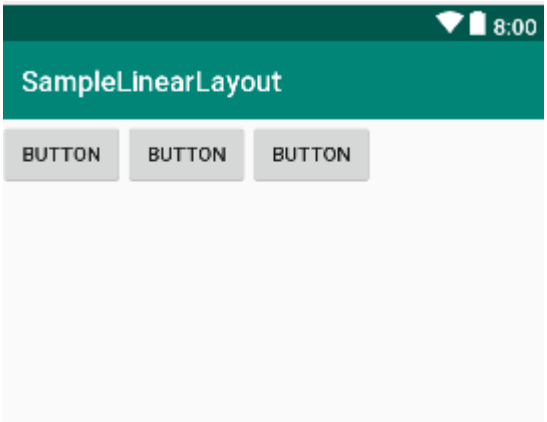
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="horizontal"    // horizontal(가로)
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    ...
```

- **버튼들의 layout_width 속성을 wrap_content로 변환**

  activity_main.xml

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

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
```



자바 코드에서 화면 구성하기

- **setContentView() 메소드**

  MainActivity.java

```
package com.example.lenovo.samplelinearlayout;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);     // 이 부분이 activity_main.xml
파일을 파라미터로 전달하면 이 레이아웃 파일이 액티비티에 설정된다.
    }
}
```

- **XML로 만든 화면을 자바 코드에서 직접 만들기**

  1. LayoutCodeActivity 엑티비티를 만든 후 수정한다.

     /java/LayoutCodeActivity.java

```
package com.example.lenovo.samplelinearlayout;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.LinearLayout;

public class LayoutCodeActivity extends AppCompatActivity {
```

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

        // new 연산자로 리니어 레이아웃을 만들고 방향 설정
        LinearLayout mainLayout = new LinearLayout(this);
        mainLayout.setOrientation(LinearLayout.VERTICAL);

        // new 연산자로 레이아웃 안에 추가될 뷰들에 설정할 파라미터 생성
        LinearLayout.LayoutParams params =
                new LinearLayout.LayoutParams(
                        LinearLayout.LayoutParams.MATCH_PARENT,
                        LinearLayout.LayoutParams.WRAP_CONTENT
                );


        // 버튼에 파라미터 설정하고 레이아웃 추가
        Button button01 = new Button(this);
        button01.setText("Button 01");
        button01.setLayoutParams(params);
        mainLayout.addView(button01);

        // 새로 만든 레이아웃을 화면에 설정
        setContentView(mainLayout);
    }
}
```

new LinearLayout()을 통해 만들어진 리니어 레이아웃 객체에는 방향을 지정하는 **setOrientation()** 메소드를 사용할 수 있으며, **setOrientation(LinearLayout.VERTICAL)**과 같이 방향 속성을 정의한 상수를 파라미터로 전달하면 세로 방향 또는 가로 방향으로 뷰를 추가할 수 있다.

뷰 객체를 코드에서 만들 때 **뷰의 생성자에는 항상 Context 객체가 전달**되어야 한다. AppCompatActivity 클래스는 Context를 상속하므로 이 클래스 안에서는 **this를 Context 객체로 사용**할 수 있다.

- **Context 객체** : UI 구성 요소인 뷰에 대한 정보를 손쉽게 확인하거나 설정할 수 있도록 뷰의 생성자에 Context 객체를 전달하도록 되어 있다.

뷰를 만들 때 뷰의 배치를 위한 속성을 설정할 수 있는 **LayoutParams 객체**를 사용한다. LayoutParams 객체를 새로 만들 때 **반드시 뷰의 가로와 세로 속성을 지정**해야하며, **LayoutParams.MATCH_PARENT와 LayoutParams.WRAP_CONTENT** 중 하나를 사용할 수 있다. 필요한 경우에는 이 두 가지 상수가 아닌 가로와 세로의 크기 값을 직접 설정 가능.

소스 코드에서 레이아웃에 **뷰를 추가하고 싶으면 addView() 메소드**를 사용한다. addView() 메소드에는 **추가할 뷰를 파라미터로 전달**할 수 있으며, **LayoutParams 객체를 같이 전달**할 수도 있다.

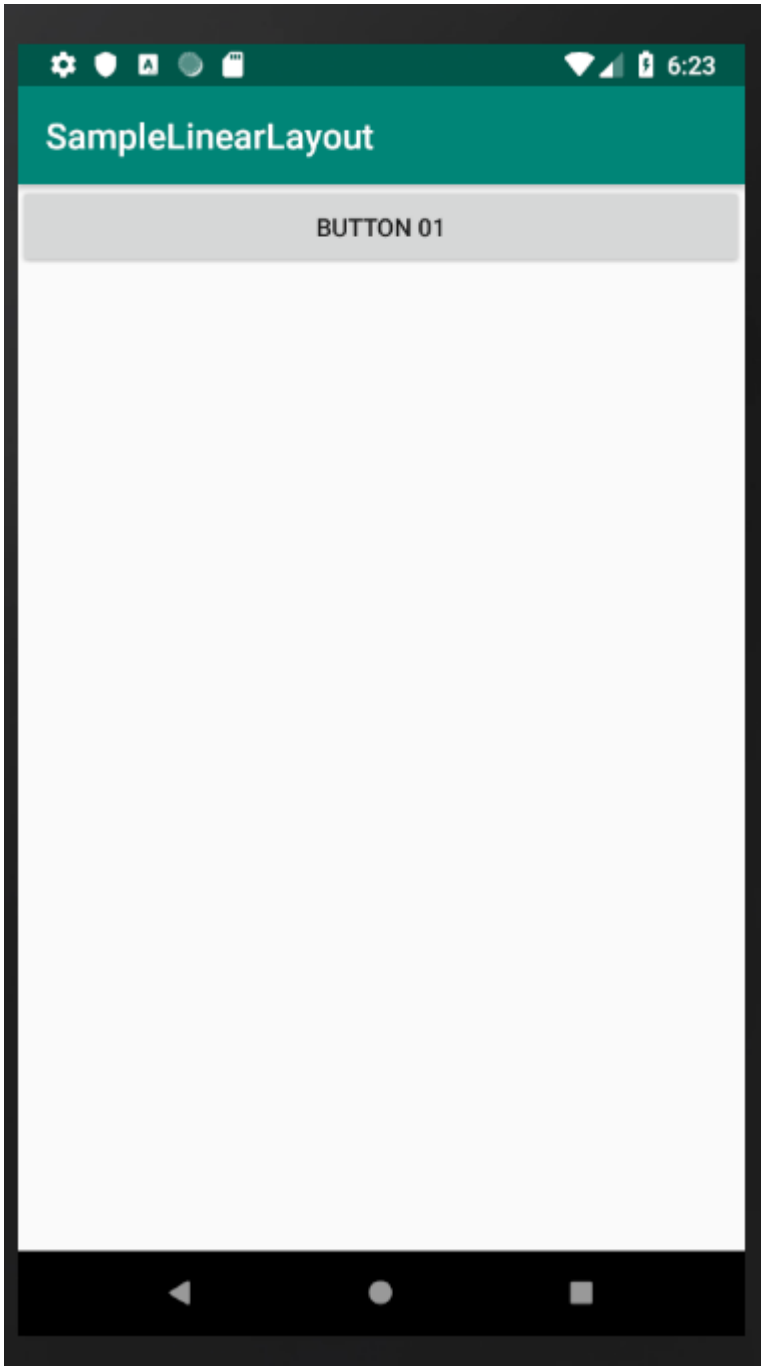
> 여기서는 버튼 객체의 setLayoutParams() 메소드를 이용해 레이아웃 파라미터를 버튼 객체에 먼저 설정했다.

2. mainfest에서 메인 화면으로 변경

AndroidMamifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.lenovo.samplelinearlayout">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        // LayoutCodeActivity로 변경
        <activity android:name=".LayoutCodeActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

3. **실행화면**



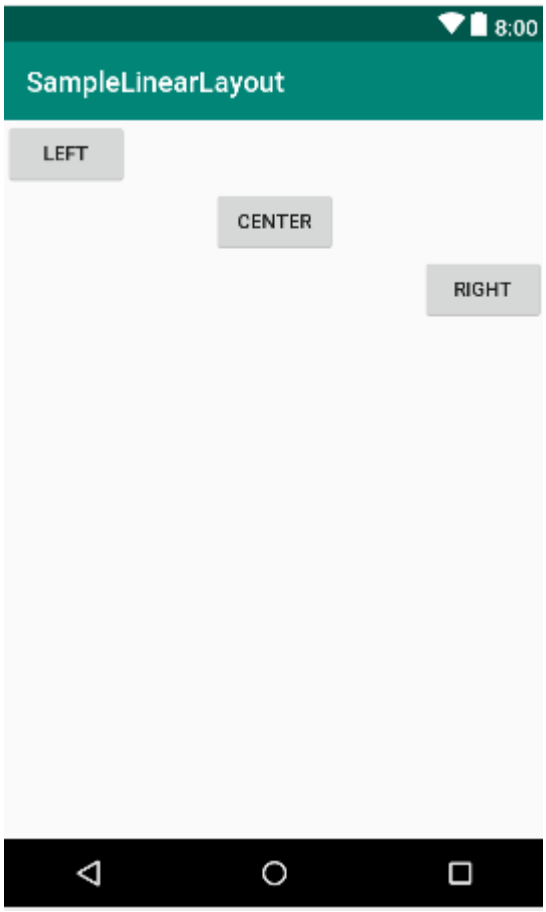
뷰 정렬하기

- **정렬 속성** : `gravity`, 어느 쪽에 무게 중심을 놓을 것인가의 의미
  - **layout_gravity** : 부모 컨테이너의 여유 공간에 뷰가 모두 채워지지 않아 여유 공간이 생겼을 때 여유 공간 안에서 뷰를 정렬
  - **gravity** : 뷰 안에 표시하는 내용물을 정렬할 때(텍스트뷰의 경우, 내용물은 글자가 되고 이미지뷰의 경우 내용물은 이미지가 됨)

코드로 레이아웃 설정 및 버튼에 gravity 설정

/res/layout/gravity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout   // 리니어 레이아웃 설정
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"  // vertical(세로)로 설정
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >

        // 첫 번째 버튼 왼쪽으로 정렬
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left"
        android:text="left"
        />

         // 두 번째 버튼 가운데로 정렬
    <Button
        android:id="@+id/button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="center"
        />

        // 세 번째 버튼 오른쪽으로 정렬
    <Button
        android:id="@+id/button06"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="right"
        />

</LinearLayout>
```

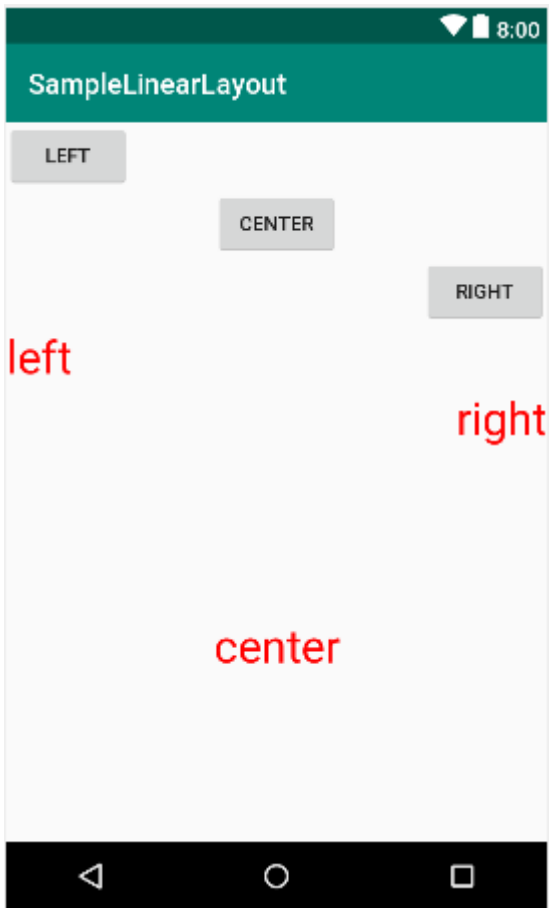


텍스트 색깔 및 gravity, 가로, 세로 설정

/res/layout/gravity.xml

```
<TextView   // 텍스트뷰의 안의 글자를 왼쪽으로 정렬
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="left"
        android:textColor="#ffff0000"   // 빨간색
        android:textSize="32dp"
        android:text="left"
        />

    <TextView   // 텍스트뷰 안의 글자를 오른쪽으로 정렬
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="right"
        android:textColor="#ffff0000"
        android:textSize="32dp"
        android:text="right"
        />
```

```
    <TextView   // 텍스트뷰 안의 글자를 가로와 세로의 가운데로 정렬
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity= "center_horizontal |center_vertical"
        android:textColor="#ffff0000"
        android:textSize="32dp"
        android:text="center"
        />
```



gravity 속성 값

정렬 속성 값	설 명
top	위쪽 끝에 배치하기
bottom	아래 ~
left	왼쪽 ~
right	오른쪽 ~
center_vertical	수직 방향의 중앙에 배치
center_horizontal	수평 ~
fill_vertical	수직 방향으로 여유 공간만큼 확대하여 채우기
fii_horizontal	수평 ~
center	중앙에 배치
fill	여유 공간만큼 확대하여 채우기
clip_vertical	상하 길이가 여유 공간보다 클 경우에 자르기
clip_horizontal	좌우 ~

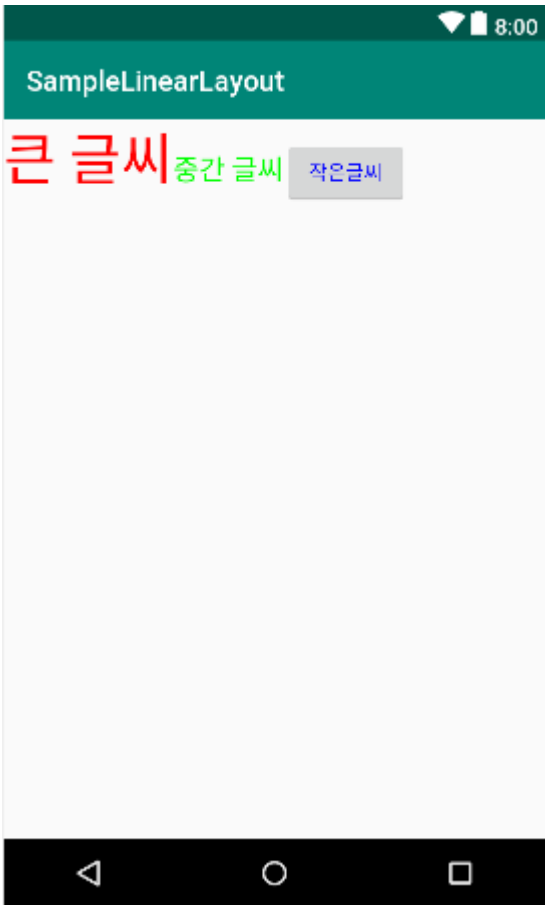
baselineAligned 속성

: 텍스트가 옆의 텍스트뷰나 버튼에 들어 있는 텍스트와 높이가 맞지 않는 경우가 있으므로 baselineAligned 속성을 이용해 정렬을 한다.

/res/layout/baseline.xml(baseline 속성 사용 전)

```
<?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">

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="큰 글씨"
        android:textColor="#ffff0000"
        android:textSize="40dp"
        />
```

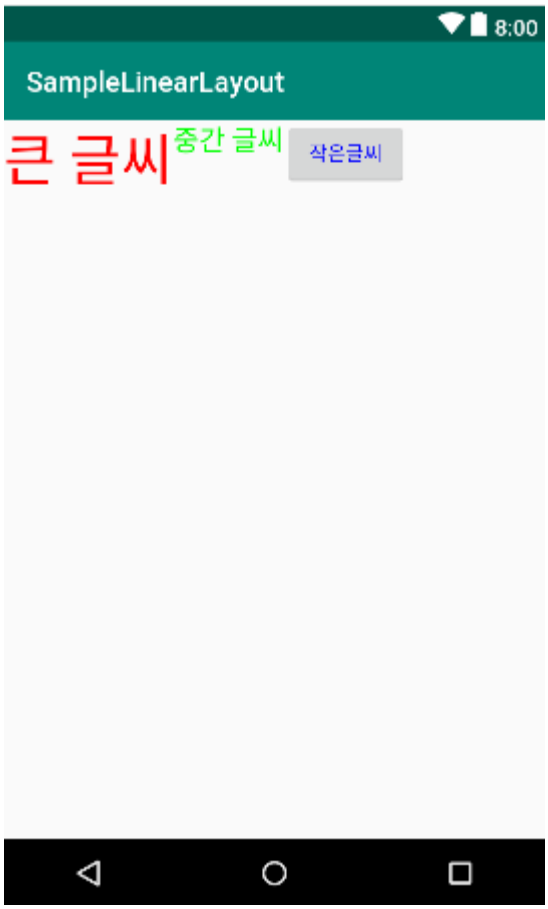
```
    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="중간 글씨"
        android:textColor="#ff00ff00"
        android:textSize="20dp"
        />

    <Button
        android:id="@+id/button7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="작은글씨"
        android:textColor="#ff0000ff"
        android:textSize="14dp"
        />

</LinearLayout>
```



/res/layout/baseline.xml(baseline 속성 사용 후)

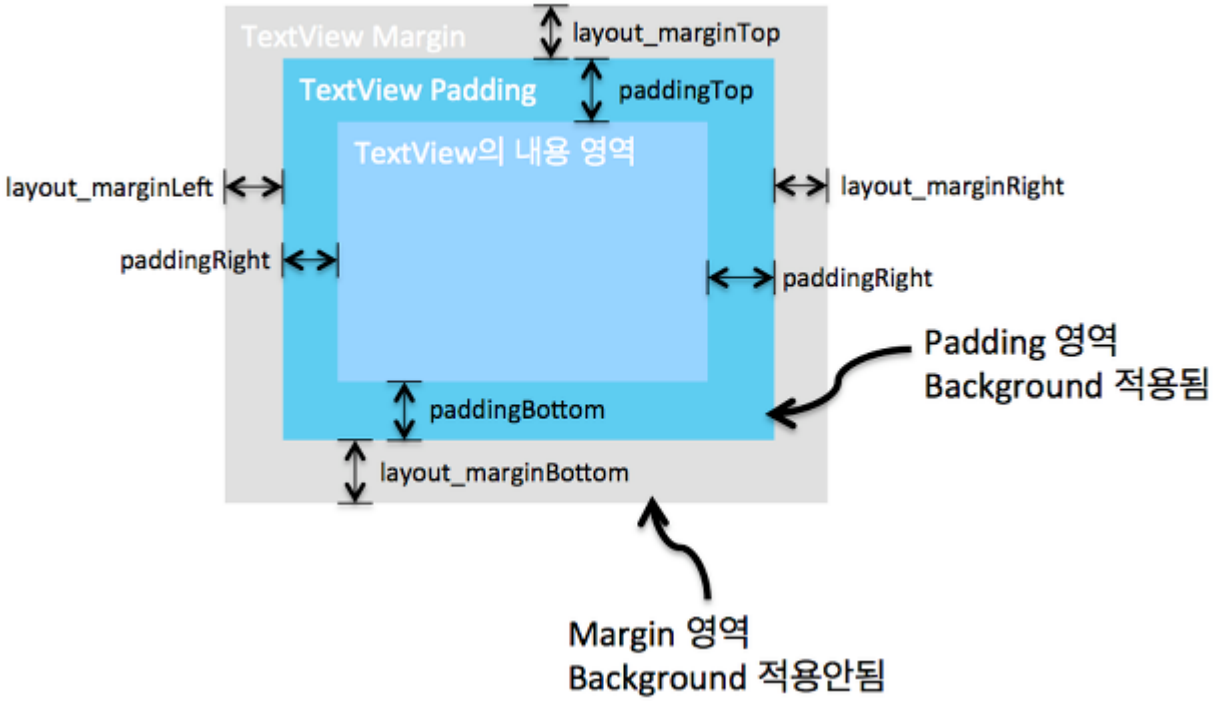
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:baselineAligned="false" // baselineAligend를 false로 지정
    >

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="큰 글씨"
        android:textColor="#ffff0000"
        android:textSize="40dp"
        />

    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="중간 글씨"
        android:textColor="#ff00ff00"
        android:textSize="20dp"
        />

    <Button
        android:id="@+id/button7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="작은글씨"
        android:textColor="#ff0000ff"
        android:textSize="14dp"
        />

</LinearLayout>
```



> 텍스트들이 위쪽으로 정렬된 것을 확인할 수 있다.

뷰의 마진과 패딩 설정하기

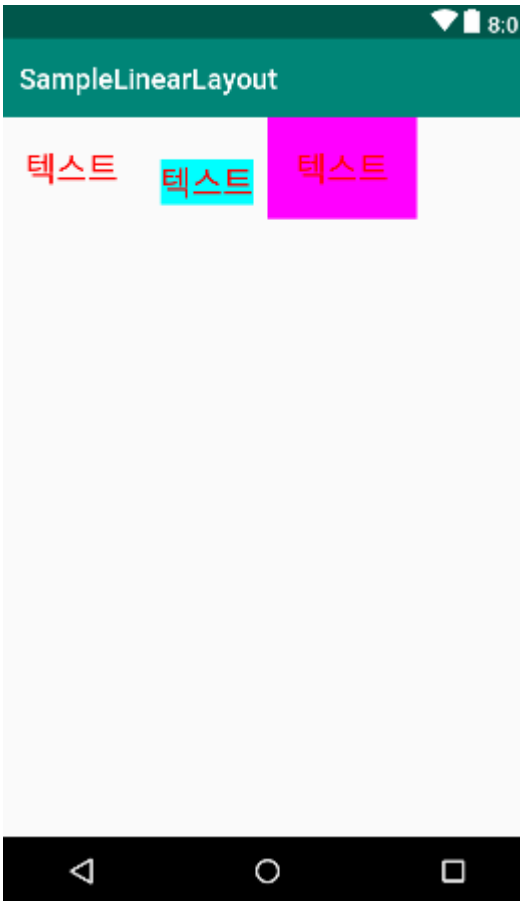


뷰의 영역은 테두리선으로 표시할 수 있다. 테두리선을 기준으로 바깥 공간과 안쪽 공간이 존재한다. 이 공간을 모두 포함하여 **뷰가 가지는 공간을 셀(Cell)**이라고 부른다.(버튼이나 텍스트는 위젯 셀이라고 부른다.) 테두리선을 기준으로 **테두리선 바깥의 공간을 마진(Margin)**이라 하며 **안쪽의 공간을 패딩(Padding)**이라고 한다.

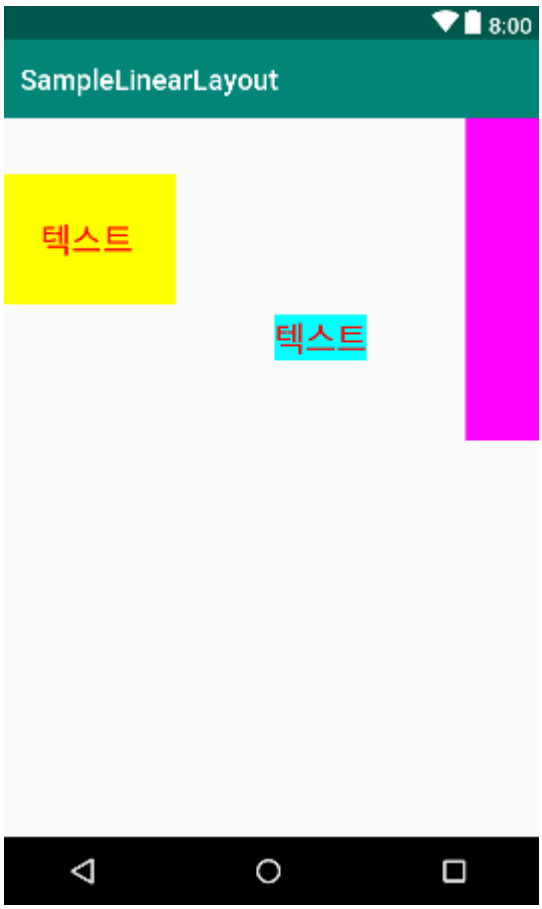
/res/layout/padding.xml(예제)

```
<?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"
    >

    <TextView   // 텍스트뷰 위젯 내부의 여백을 20 dp로 설정
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="텍스트"              // 글씨
        android:textColor="#ffff0000"     // 글씨 색
        android:textSize="24dp"           // 글씨 크기
        android:background="ffffff00"     // 배경색
        android:padding="20dp"            // 여백
        />

    <TextView   // 부모 여유 공간 사이의 여백을 10dp로 설정
        android:id="@+id/textView7"
        android:text="텍스트"
```

```
        android:textColor="#ffff0000"
        android:textSize="24dp"
        android:background="#ff00ffff"
        android:layout_margin="10dp"    // 부모 여유 공간 10 dp
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
            />

    <TextView   // 버튼 위젯 내부의 여백 20dp로 설정
        android:id="@+id/textView8"
        android:text="텍스트"
        android:textColor="#ffff0000"
        android:textSize="24dp"
        android:background="#ffff00ff"
        android:padding="20dp"  // 내부의 여백 20 dp 설정
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```



- **padding을 더 늘린 뒤**



여유 공간 분할하기

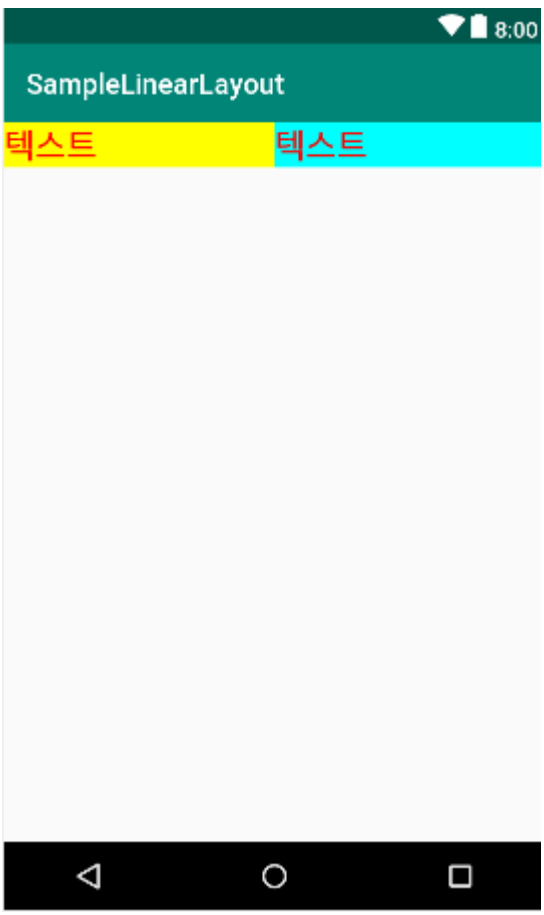
- **layout_weight에서 할당하는 크기는 원래의 뷰 크기에 추가되는 크기이다.**

  : layout_weight 속성으로 각 뷰에 할당하는 크기는 여유 공간을 분할하거나 사용하기 위해 쓰인다.

/res/layout/weight.xml(layout_weight 특징 알기)

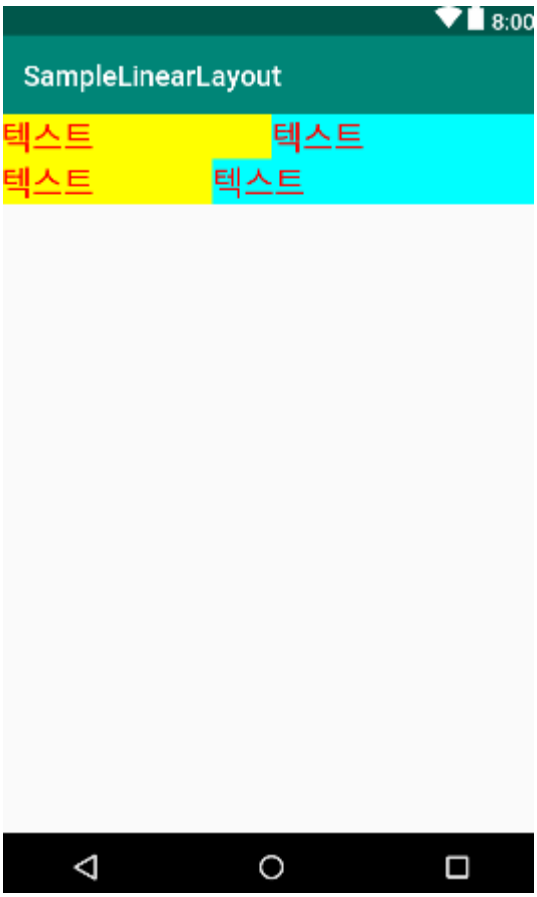
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout   // 리니어 레이아웃을 수직 방향으로 설정
    xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_height="match_parent">

    <LinearLayout   // 리니어 레이아웃을 수평 방향으로 설정
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
```

```
        <TextView   // layout_weight를 1로 설정
            android:id="@+id/textView9"
            android:background="#ffffff00"
            android:text="텍스트"
            android:textSize="24dp"
            android:textColor="#ffff0000"
            android:layout_weight="1"   // 여유 공간 = 1
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <TextView   // layout_weight를 1로 설정
            android:id="@+id/textView10"
            android:background="#ff00ffff"
            android:text="텍스트"
            android:textSize="24dp"
            android:textColor="#ffff0000"
            android:layout_weight="1"   // 여유 공간 = 1
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>

</LinearLayout>
```



/res/layout/weight.xml(layout_weight 1과 2로 설정)

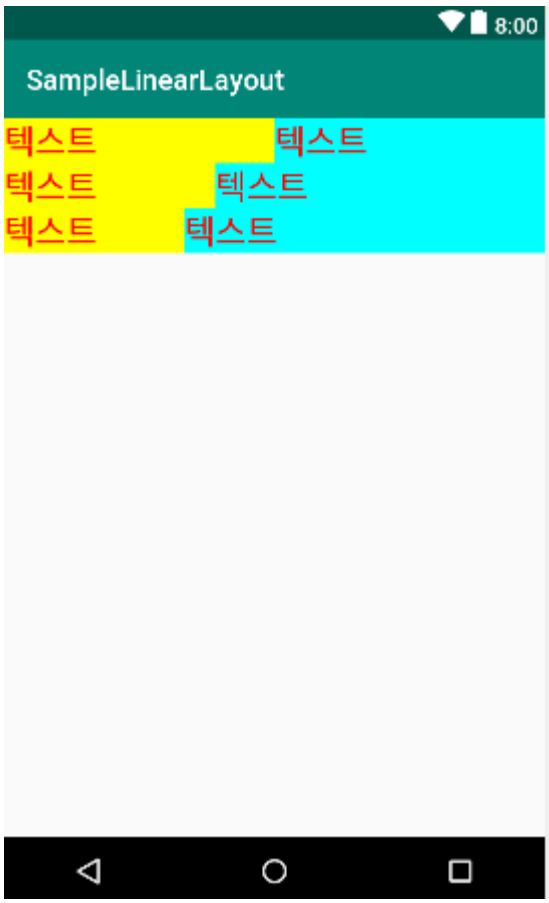
```
...
<LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >

        <TextView
            android:id="@+id/textView11"
            android:background="#ffffff00"
            android:text="텍스트"
            android:textColor="#ffff0000"
            android:textSize="24dp"
            android:layout_weight="1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <TextView
            android:id="@+id/textView12"
            android:background="#ff00ffff"
            android:text="텍스트"
            android:textSize="24dp"
            android:textColor="#ffff0000"
            android:layout_weight="2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

    </LinearLayout>
...
```



/res/layout/weight.xml(layout_width = 0dp로 지정하고 weight을 1과 2로 설정)

```
<LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView   // layout_width 0dp로 한 후 weight를 1로 설정
            android:id="@+id/textView13"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:background="#ffffff00"
            android:text="텍스트"
            android:textColor="#ffff0000"
            android:textSize="24dp"
            android:layout_weight="1"
            />

        <TextView   // layout_width 0dp로 한 후 weight를 2로 설정
            android:id="@+id/textView14"
            android:background="#ff00ffff"
            android:text="텍스트"
            android:textSize="24dp"
            android:textColor="#ffff0000"
```

```
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="wrap_content" />
    </LinearLayout>
```

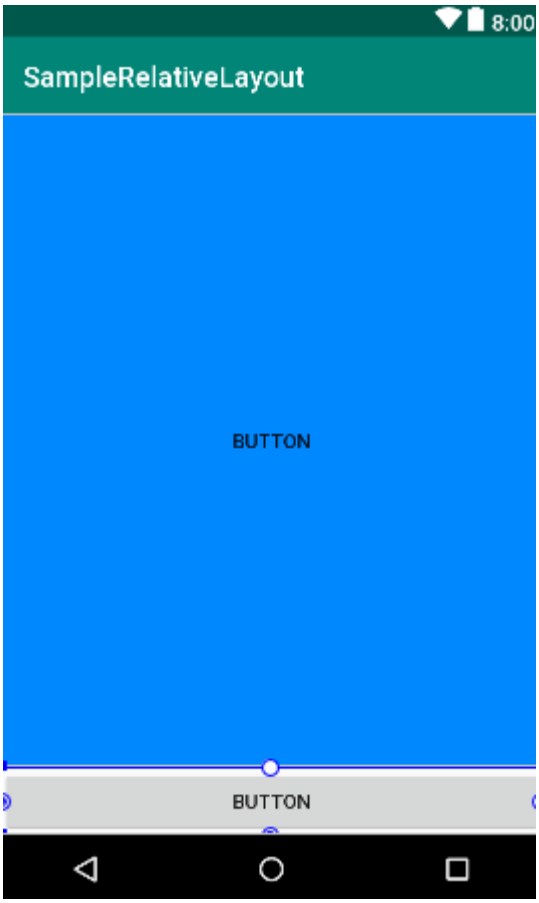


02-3. 상대 레이아웃 사용하기

/res/layout/activity_main.xml(상대 레이아웃을 이용해 효율적인 공간 차지)

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

    // 버튼을 왼쪽, 오른쪽, 위로 붙인다.(alignParentLeft, Start,Top)
    // layout_above 속성을 추가해서 두번째 버튼의 윗부분까지만 공간을 차지한다.
    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentLeft="true"
```

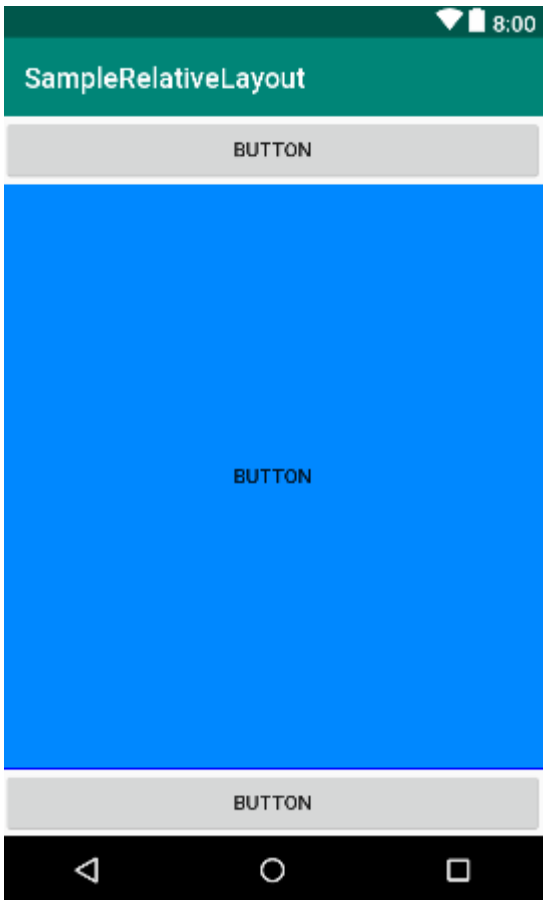
```
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:background="#ff0088ff"
        android:layout_above="@+id/button2"
        android:text="Button"
         />

    // 버튼을 왼쪽, 오른쪽, 아래로 붙인다.(alignParentLeft, Start,Top)
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:text="Button" />
</RelativeLayout>
```



/res/layout/activity_main.xml(상대 레이아웃을 이용해 효율적인 공간 차지)

```
<Button // 위쪽과 연결을 끊고 세 번째 버튼의 아래에 둔다.
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
        android:layout_alignParentLeft="true"   // 왼쪽과 오른쪽만 연결
        android:layout_alignParentRight="true"
        android:background="#ff0088ff"
        android:layout_above="@+id/button2"
        android:text="Button"
        android:layout_below="@+id/button3"     // 세 번째 버튼 아래에 둔다
         />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:text="Button" />

    <Button // 왼쪽 오른쪽 위에만 연결한다.
        android:id="@+id/button3"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:text="Button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
```



- 상대 레이아웃에서 부모 컨테이너와의 상대적 위치를 이용하는 속성

속성	설 명
layout_alignParentTop	부모의 위쪽과 뷰의 위쪽 맞춤
layout_alignParentBottom	부모의 아래쪽과 뷰의 아래쪽 맞춤
layout_alignParentLeft	왼쪽 맞춤
layout_alignParentRight	오른쪽 맞춤
layout_centerHorizontal	수평 방향 중앙에 배치
layout_centerVertical	수직 ~
layout_centerlnParent	수평과 수직 ~

- 상대 레이아웃에서 다른 뷰와의 상대적 위치를 이용하는 속성

속성	설 명
layout_above	지정한 뷰의 위쪽 배치
layout_below	지정한 뷰의 아래쪽에 배치
layout_toLeftOf	지정한 뷰의 왼쪽에 배치
layout_toRightOf	오른쪽에 배치
layout_alignTop	지정한 뷰의 위쪽과 맞춤
layout_alignBottom	아래쪽 ~
layout_alignLeft	왼쪽 ~
layout_alignRight	오른쪽 ~
layout_alignBaseline	지정한 뷰와 내용물의 아래쪽 기준선을 맞춤

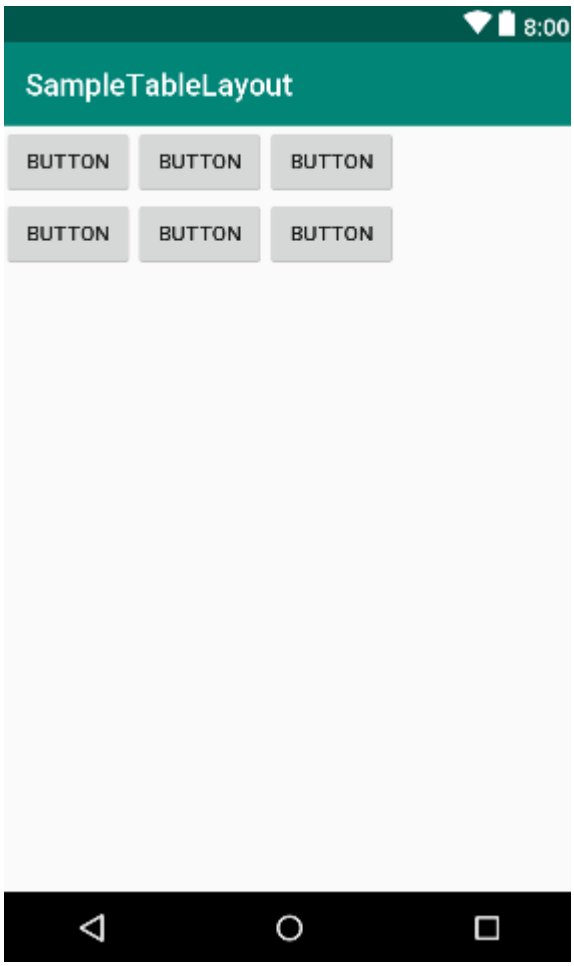
02-4. 테이블 레이아웃

/res/layout/activity_main.xml(테이블 레이아웃을 이용한 버튼 추가)

```
<?xml version="1.0" encoding="utf-8"?>
// TableLayout으로 Layout 전환
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TableRow   // Table 첫 번째 행
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        >

        <Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button"
            />

        <Button
            android:id="@+id/button2"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="Button"
            />

        <Button
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button"
            />
    </TableRow>

    <TableRow   // Table 두 번째 행
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        >

        <Button
            android:id="@+id/button6"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="Button"
            />

        <Button
            android:id="@+id/button5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button"
            />
        <Button
            android:id="@+id/button4"
```

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

    </TableRow>

</TableLayout>
```



/res/layout/activity_main.xml(버튼 가로 공간을 꽉 채우도록 한다)

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="0,1,2"  // 컬럼 3개로 가로 공간을 채운다.
    tools:context=".MainActivity">
...
```

/res/layout/activity_input.xml(열을 자동 축소하거나 확장하기)

- 테이블 레이아웃 설정 속성
  - **shrinkColumns** : 부모 컨테이너의 폭에 맞추도록 각 열의 폭을 강제로 축소
  - **stretchColumns** : 부모 컨테이너의 여유 공간을 모두 채우기 위해 각 열의 폭을 강제로 늘린다.

/res/layout/activity_input.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout    // 테이블 레이아웃으로 설정
    xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
    android:background="#ff3366cc"      // 배경 색깔 설정
    android:stretchColumns="0, 1, 2"    // 자동 확장 테이블
    android:layout_height="match_parent">

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        >
        <TextView
            android:id="@+id/textView"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="이름: "
            />

        <EditText
            android:id="@+id/editText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_span="3" // 3 개의 칼럼을 차지하게 한다.
            />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="10dp" // 위의 테이블에 10dp를 띄운다.
        >
        <Button
            android:id="@+id/button8"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="2"   // 2번째 컬럼 위치
            android:text="아니오"
            />
        <Button
            android:id="@+id/button7"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
```

```
                                        android:text="예"



            />
    </TableRow>

</TableLayout>
```

02-5. 스크롤뷰 사용하기

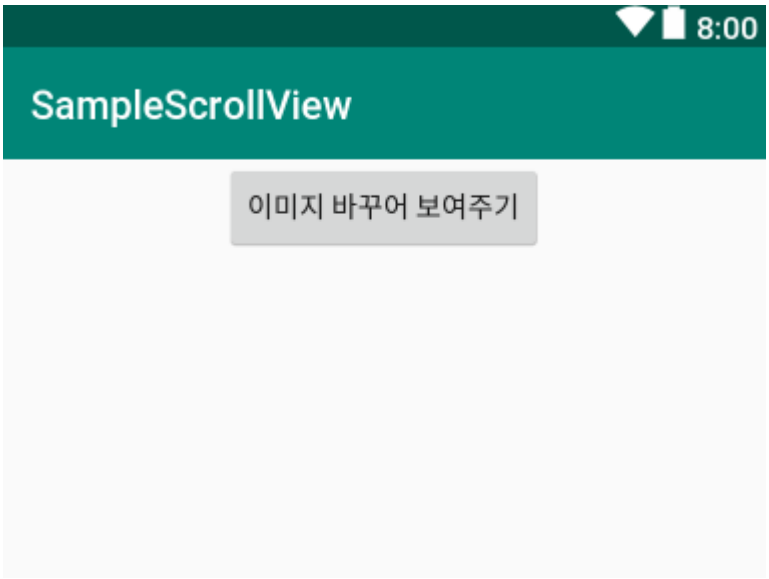
: ScrollView 태그를 사용하며 그 안에는 하나의 뷰가 들어갈 수 있다.

/res/layout/activity_main.xml(스크롤 뷰 사용하기)

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

    <Button // 이미지 변경을 위한 버튼
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="이미지 바꾸어 보여주기"
        android:onClick="onButton1Clicked"
        />

    // 수평 스크롤 안에 스크롤뷰 추가 후 그 안에 이미지뷰를 추가
    <HorizontalScrollView   // 수평 스크롤을 위한 스크롤뷰
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        >

        <ScrollView // 수직 스크롤을 위한 스크롤뷰
            android:id="@+id/scrollView"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            >
            <ImageView  // 이미지를 보여주는 스크롤뷰
                android:id="@+id/imageView"
                android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content" />
        </ScrollView>
    </HorizontalScrollView>

</LinearLayout>
```

> 스크롤뷰는 기본적으로 수직 방향의 스크롤을 지원한다. 만약 수평 방향의 스크롤을 사용하려면 HorizontalScrollView를 이용하면 된다.



02-6. 프레임 레이아웃과 뷰의 전환

: 프레임 레이아웃은 뷰를 하나 이상 추가할 경우 추가된 순서로 차곡차곡 쌓는다.(가시성 속성: 보이거나 보이지 않게 하는 속성)

/res/layout/activity_main.xml

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

    // 이미지 전환 버튼
    <Button
        android:id="@+id/button"
        android:layout_gravity="center"
```

```
        android:text="이미지 바꾸기"
        android:onClick="onButton1Clicked"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    // 프레임 레이아웃으로 나머지 화면 채우기
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        // 첫 번째 이미지뷰를 안보이도록 설정
        <ImageView
            android:id="@+id/imageView"
            android:src="@drawable/dream01"
            android:visibility="invisible"  // 안보이도록
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        // 두 번째 이미지뷰를 보이도록 설정
        <ImageView
            android:id="@+id/imageView2"
            android:src="@drawable/dream02"
            android:visibility="visible"    // 보이도록
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </FrameLayout>

</LinearLayout>
```

02-7. 기본 위젯들

텍스트뷰

: 텍스트를 화면에 보여주는 역할

text

: 텍스트뷰에 보이는 문자열을 설정할 수 있다. /res/values 폴더 안에 들어 있는 string.xml 파일 안에 들어 있는 문자열을 지정할 수도 있다.

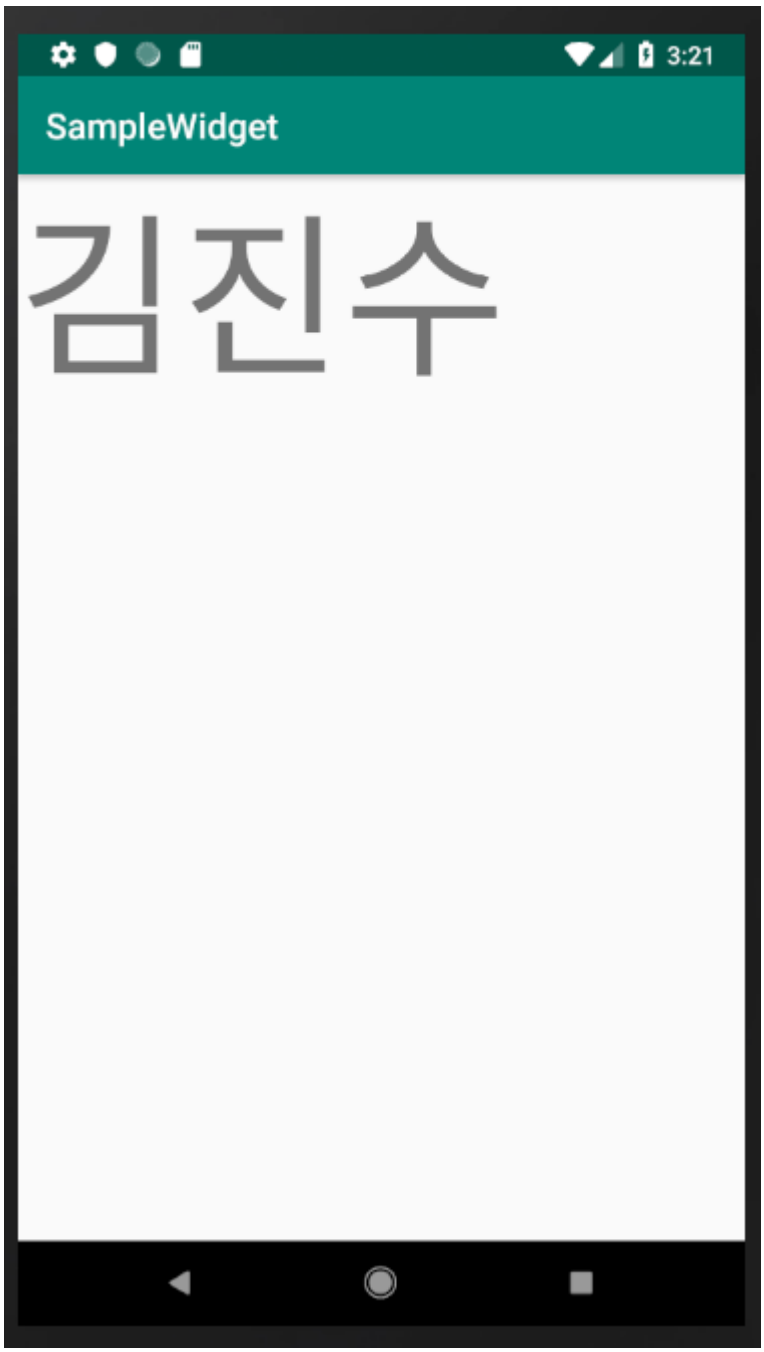
- 참조파일: **SampleWidget>/res/values/strings.xml**

```
<resources>
    <string name="app_name">SampleWidget</string>
    <string name="person_name">김진수</string> // 김진수라는 사람 이름 추가
</resources>
```

/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="100dp"
        android:text="@string/person_name"  // 태그로 텍스트 저장
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</LinearLayout>
```



textColor

: 문자열의 색상을 설정한다. 색상 설정은 "#AARRGGBB" , 각각 Alpha, Red, Green, Blue를 의미한다. **ex) Alpha값: FF(불투명), 00(투명), 88(반투명)**

textStyle

: 문자열의 스타일 속성. "normal", "bold", "italic" 등의 값 지정 가능

typeFace

: 문자열의 폰트 설정

maxLines

: 텍스트뷰에서 표시하는 문자열의 최대 줄 수 설정

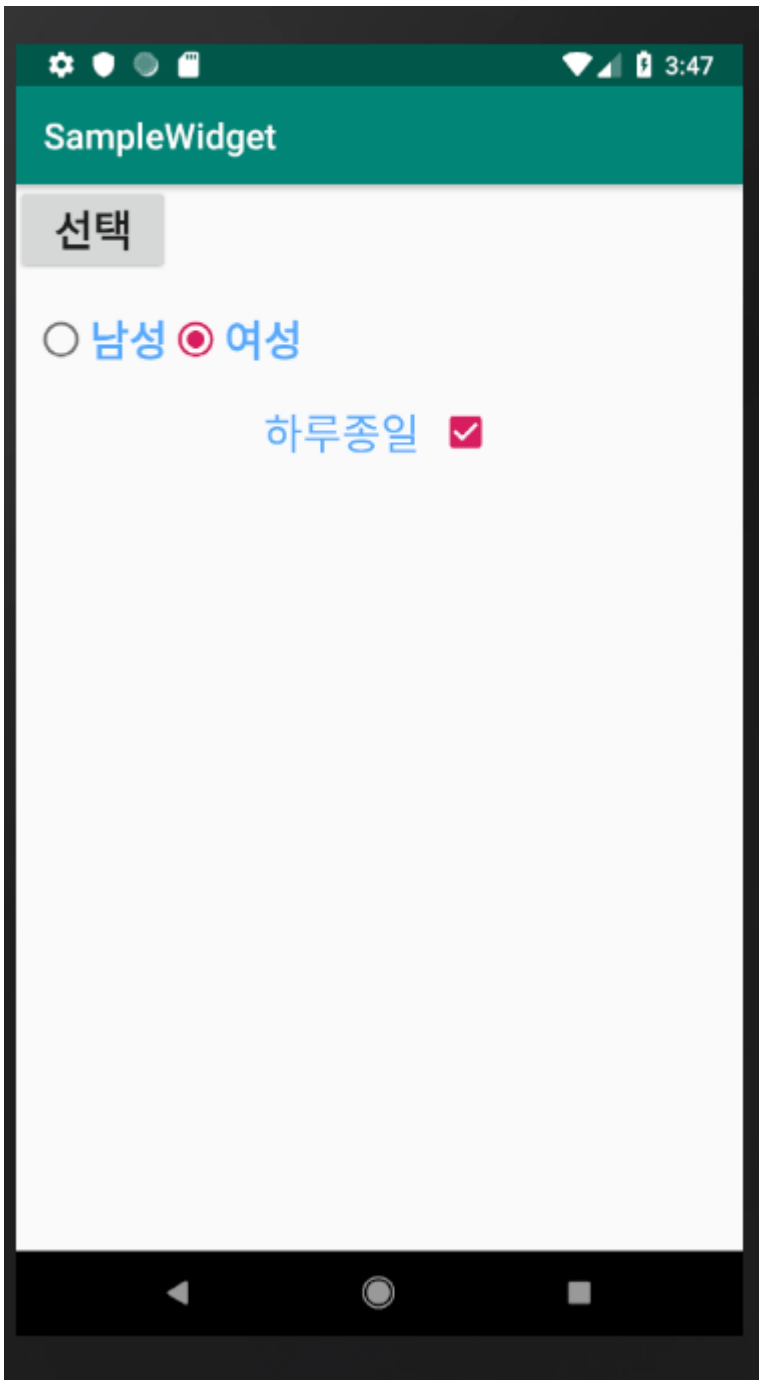
버튼

체크 박스

- 참조파일 : **SampleWidget>/res/layout/button.xml**

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

    <Button // 기본 버튼
        android:id="@+id/btnExit"
        android:text=" 선택 "
        android:textSize="24dp"
        android:textStyle="bold"
        android:gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <RadioGroup // 라디오 버튼 두 개를 묶어놓은 라디오 그룹
        android:id="@+id/radioGroup01"
        android:orientation="horizontal"
        android:layout_marginTop="20dp"
        android:paddingLeft="10dp"
        android:paddingRight="10dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <RadioButton    // 첫 번째 라디오 버튼
            android:id="@+id/radio01"
            android:text="남성"
            android:textStyle="bold"
            android:textColor="#ff55aaff"
            android:textSize="24dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
```

```
        <RadioButton    // 두 번째 라디오 버튼
            android:id="@+id/radio02"
            android:textSize="24dp"
            android:textColor="#ff55aaff"
            android:textStyle="bold"
            android:text="여성"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

    </RadioGroup>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical|center_horizontal"
        android:paddingTop="20dp"
        >
        <TextView
            android:text="하루종일"
            android:textColor="#ff55aaff"
            android:textSize="24dp"
            android:paddingRight="10dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <CheckBox   // 체크 박스
            android:id="@+id/allDay"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>

</LinearLayout>
```



입력상자

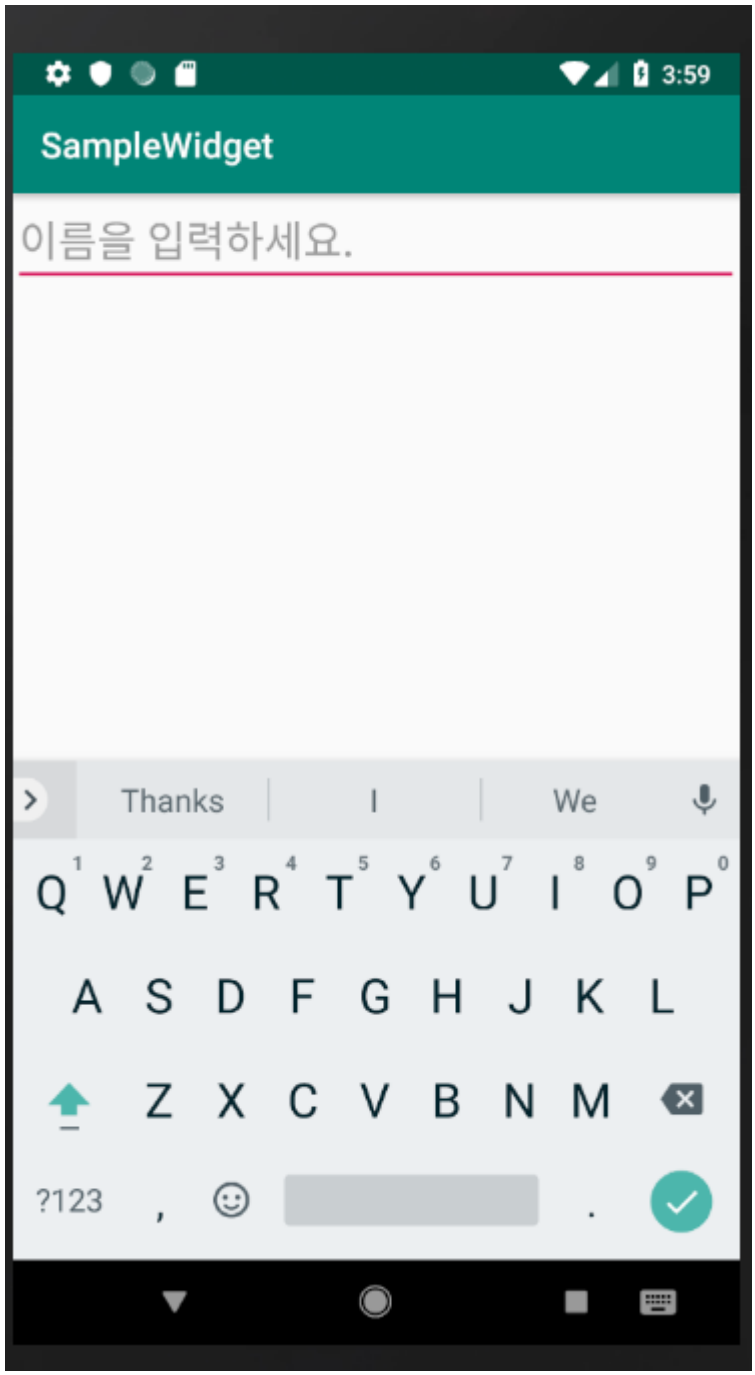
- 참조파일 : **SampleWidget>/res/layout/edittext.xml**

```
<?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:orientation="vertical">
```

```
    <EditText
        android:id="@+id/usernameInput"
        android:textSize="24dp"
        android:inputType="textCapWords"    // 입력되는 글자의 유형 정의
        android:hint="이름을 입력하세요."      // 기본 안내문 표시
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

> **hint 속성** : 글자를 입력하기 전에 간단한 안내글을 입력상자에 표시
>
> **inputType 속성** : 글자의 유형을 정할 수 있으며 글자를 입력할 때 보이는 키패드도 그 유형에 맞춰 보인다.

S



이미지뷰

: 이미지를 화면에 표시하려고 제공되는 가장 간단한 위젯

```
@drawable/이미지명
```

> 이미지명은 이미지의 확장자를 제외한 이미지 파일의 이름

src

: 원본 이미지 설정.

maxWidth, maxHeight

: 이미지가 보일 최대 크기 설정

tint

: 이미지뷰에 보이는 이미지 위에 색상을 적용하고 싶을 때 설정

scaleType

: 원본 이미지의 크기와 다르게 화면에 보이는 경우 확대/축소를 어떤 방식으로 적용할 것인지 설정.

drawble 폴더

: drawable 폴더에는 해상도별 폴더가 있으므로 각 해상도에 맞는 크기의 이미지를 넣어야한다.

- 참조파일: **SampleWidget>/res/layout/image.xml**

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

    <ImageButton    // 이미지 버튼을 사용해 이미지 추가
        android:id="@+id/imageButton"
        android:layout_marginTop="40dp"
```

```
            android:layout_marginLeft="40dp"
            android:background="@drawable/ok_btn"
            android:contentDescription="ok button"
            android:layout_width="50dp"
            android:layout_height="50dp" />

        <ImageView      // 이미지뷰를 사용해 이미지 추가
            android:id="@+id/imageView"
            android:layout_marginLeft="160dp"
            android:layout_marginTop="160dp"
            android:background="@drawable/person"
            android:contentDescription="person button"
            android:layout_width="50dp"
            android:layout_height="50dp" />

    </LinearLayout>
```

연습 문제

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

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

        <HorizontalScrollView
            android:layout_marginTop="30dp"
            android:layout_gravity="center_vertical||center_horizontal"
            android:layout_width="300dp"
            android:layout_height="200dp">

            <ScrollView
                android:id="@+id/scrollView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content">

                <ImageView
```

```
                    android:id="@+id/imageView1"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent" />

            </ScrollView>

        </HorizontalScrollView>

    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

MainActivity.java

```
package com.example.lenovo.problem;

import android.content.res.Resources;
import android.graphics.drawable.BitmapDrawable;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.ScrollView;

public class MainActivity extends AppCompatActivity {
    ScrollView scrollView;
    ImageView imageView;
    BitmapDrawable bitmap;

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

        scrollView = (ScrollView)findViewById(R.id.scrollView);
        imageView = (ImageView)findViewById(R.id.imageView1);
        scrollView.setHorizontalScrollBarEnabled(true);

        Resources res = getResources();
        bitmap = (BitmapDrawable)res.getDrawable(R.drawable.image);
        int bitmapWidth = bitmap.getIntrinsicWidth();
        int bitmapHeight = bitmap.getIntrinsicHeight();
    }


}
```