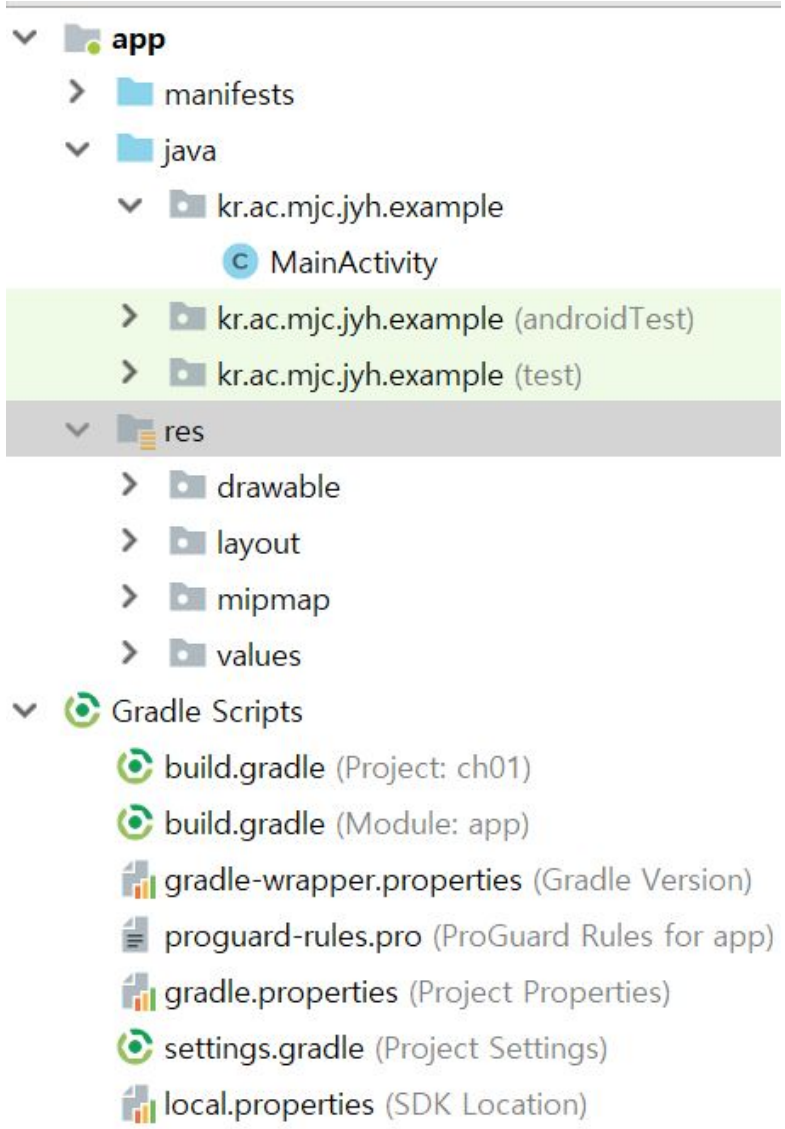
1. 안드로이드 프로젝트 구성

뷰와 뷰그룹의 정의



Manifests

- 안드로이드 앱의 설정파일 권한 Activity, Service, Broadcast 등 앱구성 등록

Java

- 안드로이드 프로젝트의 java 소스파일 저장경로

res

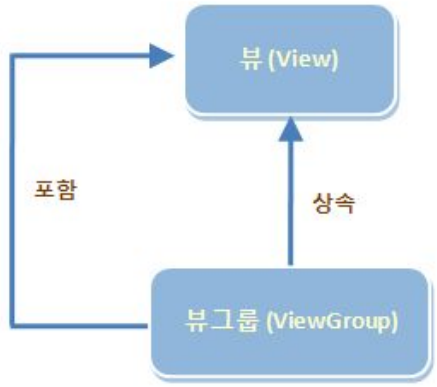
- 안드로이드의 리소스(이미지,레이아웃,아이콘등) 저장경로

Gradle Scripts

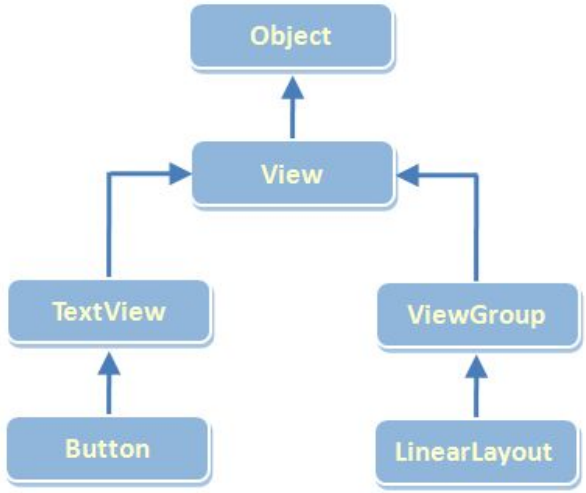
- build.gradle등 빌드시 필요한 설정 저장경로

2. 뷰와 뷰그룹

뷰와 뷰그룹의 정의



[뷰와 뷰 그룹의 관계]



[버튼과 리니어 레이아웃의 계층도]

• 뷰(View)

- 화면에 보이는 각각의 것들 (버튼, 텍스트 등등)
- 흔히 콘트롤(Control)이나 위젯(Widget)이라 불리는 UI 구성 요소

• 뷰 그룹(View Group)

- 뷰들을 여러 개 포함하고 있는 것
- 뷰 그룹도 뷰에서 상속하여 뷰가 됨. 즉, 위의 뷰는 버튼, 텍스트 뿐만 아니라 이것들을 포함하는 눈에 보이지 않는 영역을 포함함

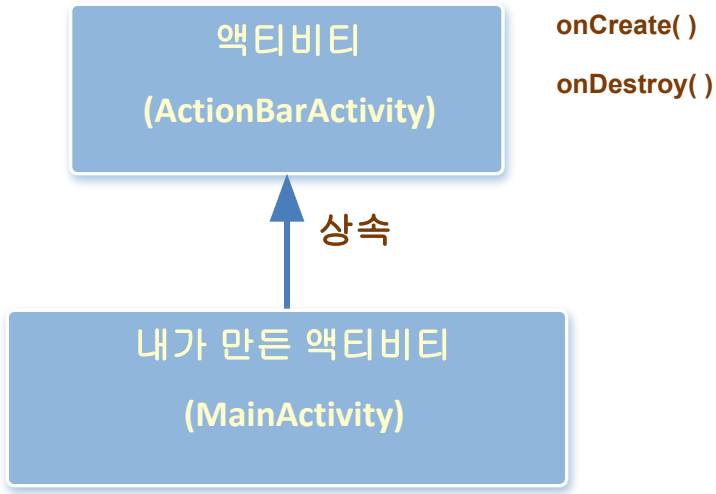
• 위젯(Widget)

- 뷰 중에서 일반적인 컨트롤의 역할을 하고 있는 것
- 버튼, 텍스트 등등

• 레이아웃(Layout)

- 뷰 그룹 중에서 내부에 뷰들을 포함하고 있으면서 그것들을 배치하는 역할을 하는 것

상속에 대해 잘 몰라요!



• 상속

- 객체지향의 가장 기본적인 개념 중 하나
- 부모의 특성을 그대로 물려받는 것으로 변수나 메소드 재사용 가능

• 액티비티의 상속

- 처음 만들어 본 액티비티에서 extends 키워드 사용

```
public class MainActivity extends ActionBarActivity
```

• 부모 클래스의 메소드를 재정의

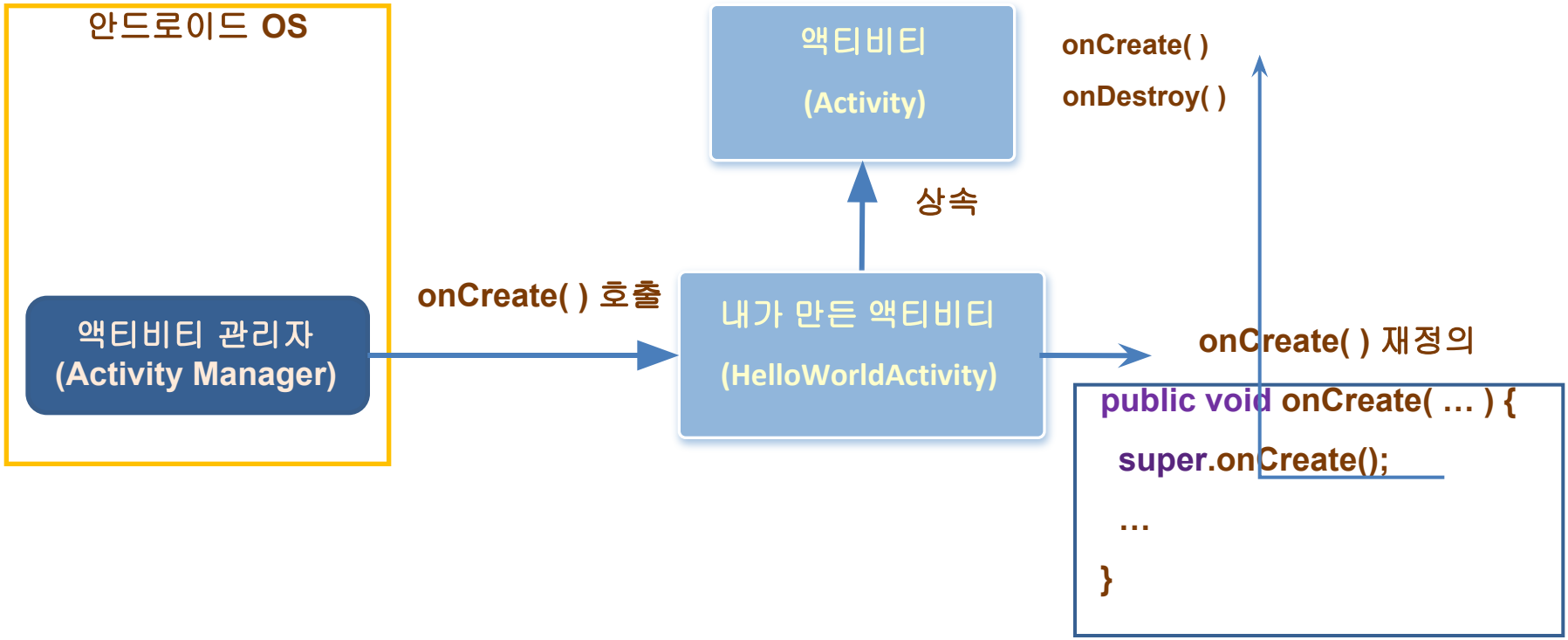
- onCreate() 메소드는 이미 부모 클래스에 정의되어 있음
- 기능을 추가하고 싶을 때 재정의(Override)

• this와 super

- 나 자신은 this, 부모는 super 를 사용하여 변수나 메소드 참조

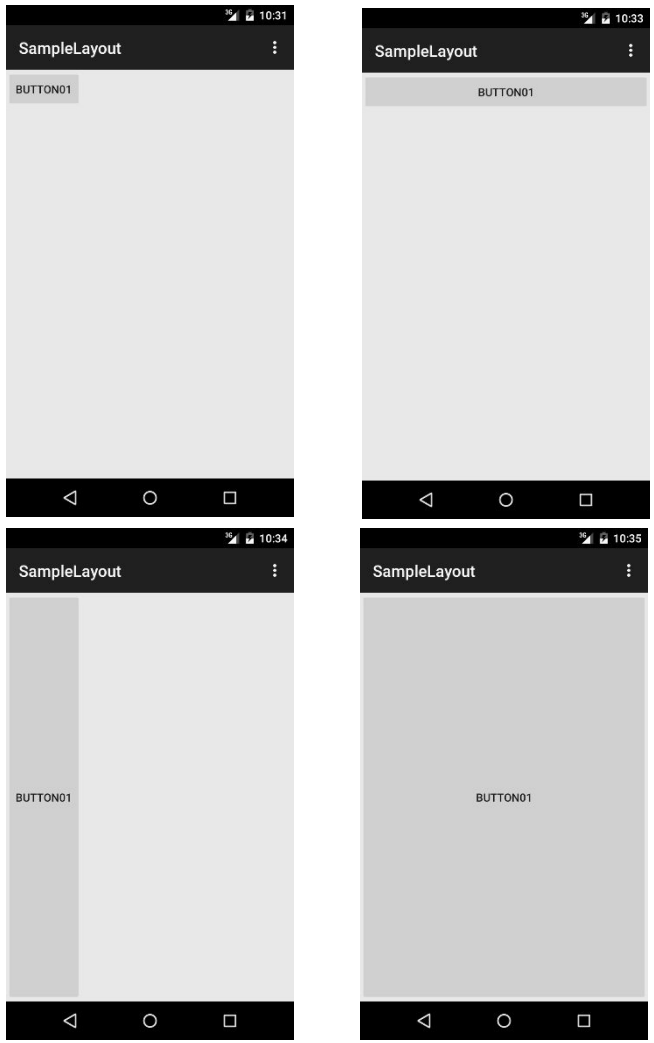
```
super.onCreate( … );
```

재정의한 메소드를 시스템에서 호출하는 방법



- **액티비티가 만들어질 때 onCreate() 호출**
  - 내가 만든 액티비티의 onCreate() 메소드를 호출
  - super.onCreate() 를 호출하여 Activity 클래스에 정의된 onCreate() 메소드의 기능을 사용
  - 그 아래에 추가적으로 필요한 기능을 코드로 추가

뷰의 대표적인 속성



[match_parent와 wrap_content 값을 폭과 넓이에 적용한 예]

- **[필수] layout_width, layout_height**
  - 가장 기본적이면서 필수 속성으로 뷰의 폭과 높이를 설정함

  (1) match_parent

  무조건 남아 있는 여유 공간을 채움

  (2) wrap_content

  뷰에 들어 있는 내용물의 크기에 따라 뷰의 크기가 결정됨

  (3) 크기 값 지정

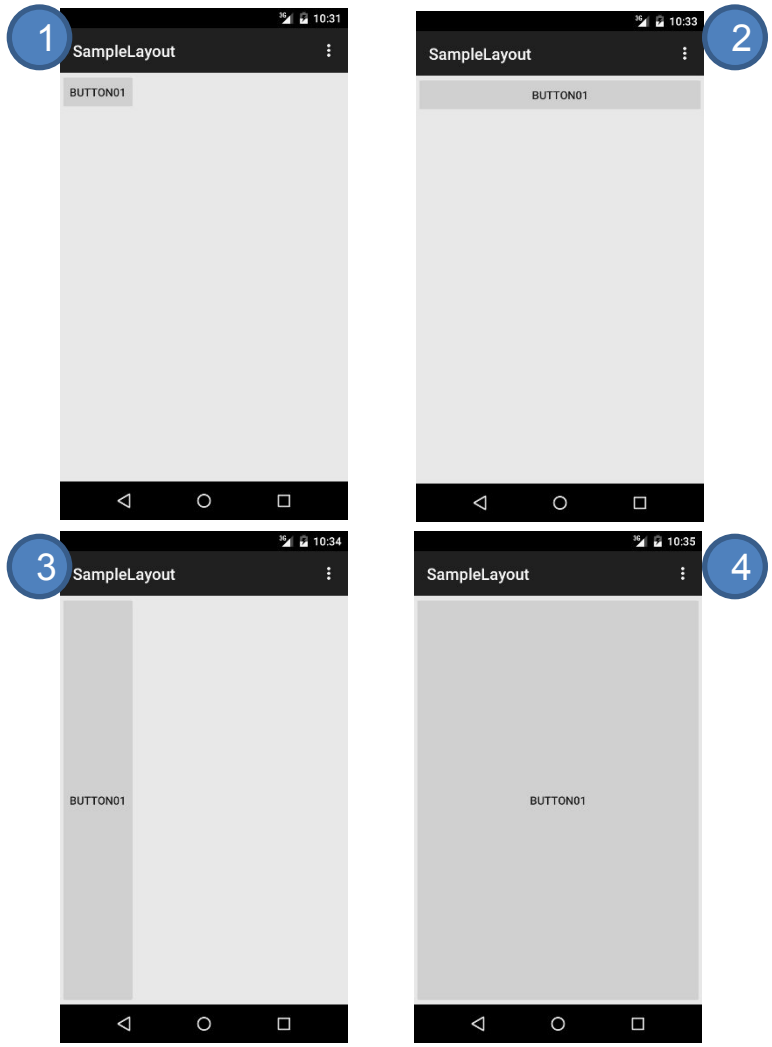
  크기를 고정된 값으로 직접 지정하고 싶을 때 사용함

  ex) “100px”, “200dp”

- **id**
  - 뷰의 ID를 지정함
  - XML 레이아웃에 정의한 뷰를 자바 소스에서 참조하는 데 사용
  - XML 레이아웃 안에서 다른 뷰를 참조하는 데 사용

- **background**
  - 뷰의 배경을 설정함 (배경색, 배경 이미지 등)

XML 레이아웃으로 구성하기



[match_parent와 wrap_content 값을 폭과 넓이에 적용한 예]

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

1

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

2

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

3

```
<Button
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:text="Button"
  />
```

4

뷰의 크기 지정에 사용되는 단위

단위	단위 표현	설 명
px	픽셀	화면 픽셀
dp 또는 dip	밀도 독립적 픽셀 (density independent pixel)	160dpi 화면을 기준으로 한 픽셀 예) 1인치 당 160개의 점이 있는 디스플레이 화면에서 1dp는 1px와 같음. 1인치 당 320개의 점이 있는 디스플레이 화면에서 1dp는 2px와 같음.
sp 또는 sip	축척 독립적 픽셀 (scale independent pixel)	가변 글꼴을 기준으로 한 픽셀로 dp와 유사하나 글꼴의 설정에 따라 달라짐
in	인치	1인치로 된 물리적 길이
mm	밀리미터	1밀리미터로 된 물리적 길이
em	텍스트 크기	글꼴과 상관없이 동일한 텍스트 크기 표시

뷰의 ID 속성

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

[버튼의 id 추가]

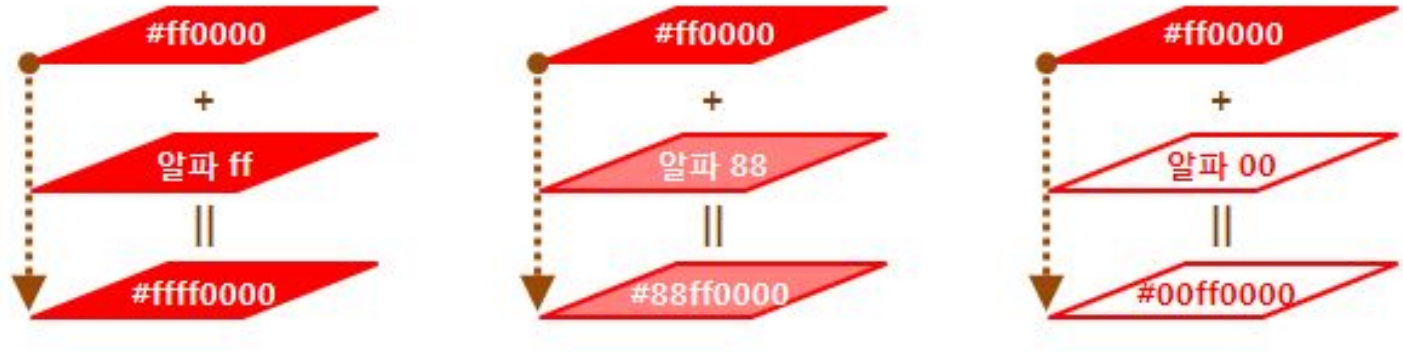
```
public class MainActivity extends ActionBarActivity {
 public void onCreate(Bundle savedInstanceState)
 {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  Button button = (Button)
                  findViewById(R.id.button );
 }
}
```

[버튼의 id 참조]

- **인플레이션(Inflation)**
  - XML 레이아웃에 정의된 정보를 메모리 상에서 객체로 만드는 객체화 과정
  - 애플리케이션이 시작될 때 이 과정을 거쳐 메모리 상에 만들어진 객체들을 참조하기 위해 ID를 지정함
- **id 속성은 자바 코드 상에서 R.id.[ID]와 같은 형태로 참조함**("@+id/...")

뷰의 background 속성

[Format]
#RGB
#ARGB
#RRGGBB
#AARRGGBB



[배경색에 알파값을 적용하여 투명도를 조절하는 경우]

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

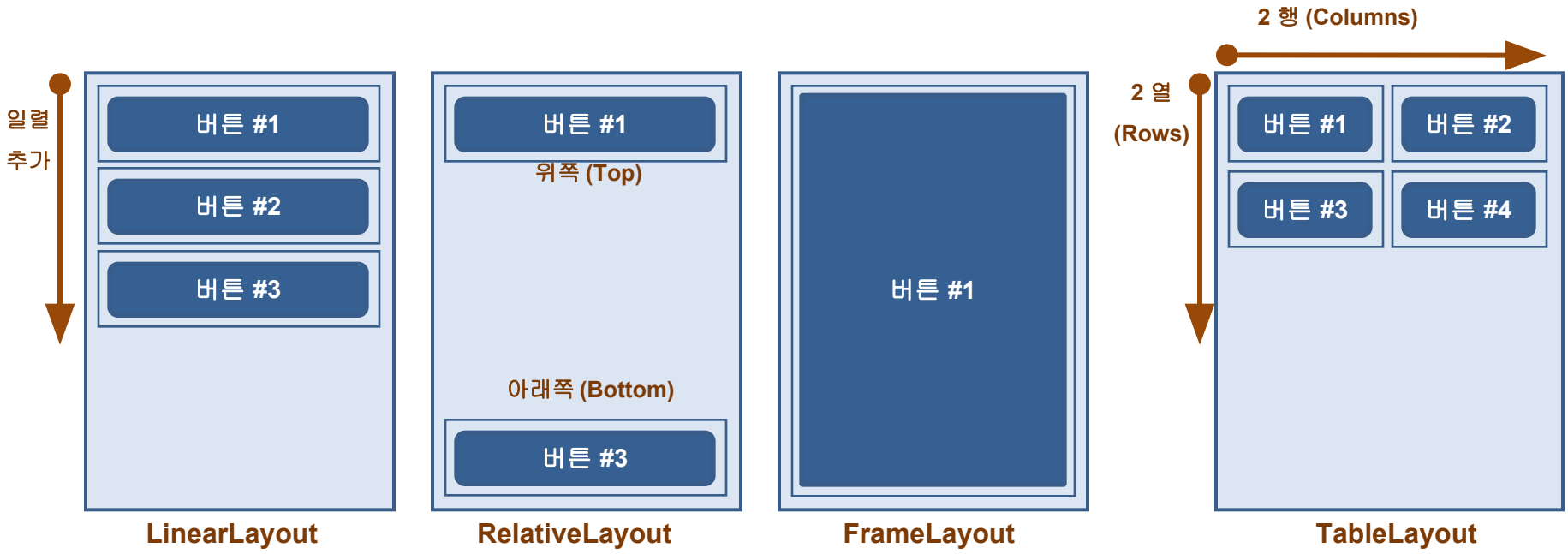
- **16**진수 값을 지정할 때는 여러 가지 포맷을 사용할 수 있음

3. 레이아웃

대표적인 레이아웃

레이아웃 이름	설 명
LinearLayout	- 박스(Box) 모델 - 사각형 영역들을 이용해 화면을 구성하는 방법 - 표준 자바의 BoxLayout과 유사
RelativeLayout	- 규칙(Rule) 기반 모델 - 부모 컨테이너나 다른 뷰와의 상대적 위치를 이용해 화면을 구성하는 방법
FrameLayout	- 기본 단위 모델 - 하나의 뷰만 보여주는 방법 - 가장 단순하지만 여러 개의 뷰를 추가하는 경우 중첩시킬 수 있으므로 뷰를 중첩한 후 각 뷰를 전환하여 보여주는 방식으로 사용할 때 유용함
TableLayout	- 격자(Grid) 모델 - 격자 모양의 배열을 이용하여 화면을 구성하는 방법 - HTML에서 많이 사용하는 정렬 방식과 유사하여 실용적임
ConstraintLayout	- LinearLayout 과 RelativeLayout FrameLayout 의 장점을 합친 레이아웃 -개념이 여러가지 혼재되어있어 복잡하지만 레이아웃을 그리기위해 소모되는 리소스가 적고 레이아웃 자유도가 높음

레이아웃에 따라 뷰를 추가하는 방식



리니어 레이아웃의 기본 속성

- **채우기 : fill model**
  - 뷰를 부모 뷰의 여유 공간에 어떻게 채울 것인지를 설정
    - 가로 크기 : **layout_width**
    - 세로 크기 : **layout_height**

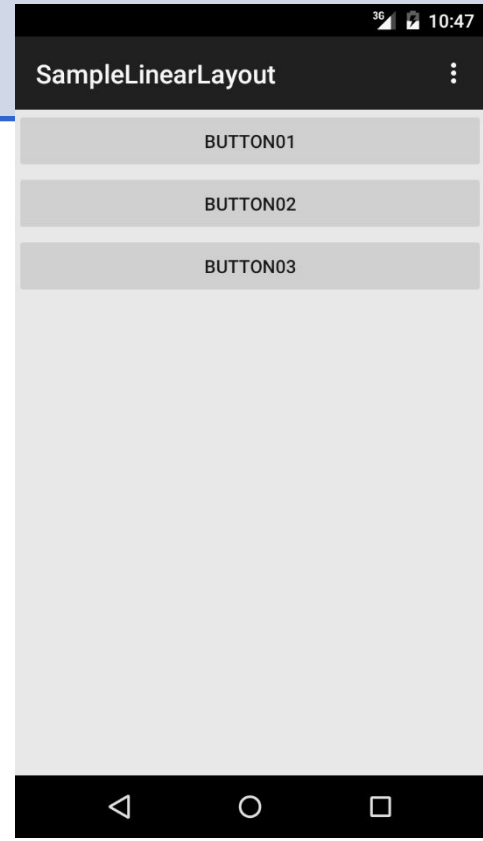
- **방향 : orientation**
  - 뷰를 추가하는 방향을 설정
- **정렬 (외부/내부) : layout_gravity, gravity**
  - 뷰의 정렬을 어떻게 할 것인지 설정
- **여유 공간 (외부/내부) : layout_margin, padding**
  - 뷰의 여유 공간을 어떻게 할 것인지 설정
- **공간가중치 (분할) : layout_weight**
  - 뷰가 차지하는 공간의 가중치 값을 설정

4. LinearLayout

리니어 레이아웃 – 방향 설정하기

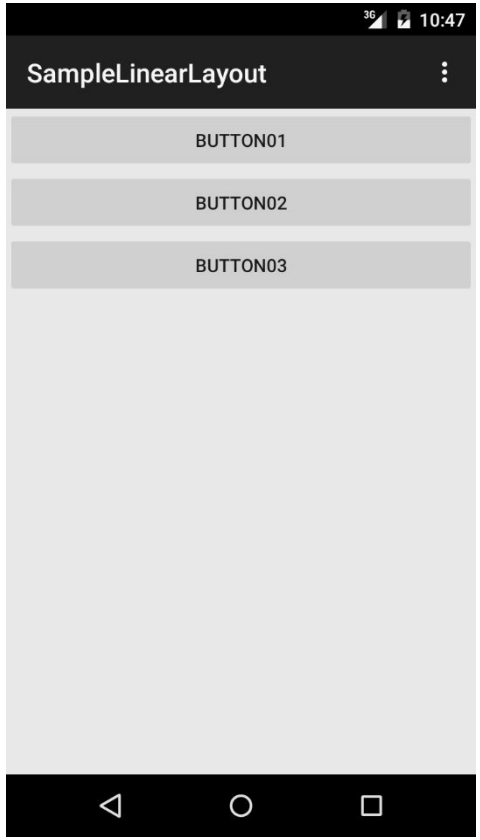
```
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation= "vertical"
 android:layout_width=“match_parent"
 android:layout_height=“match_parent"
 >
```

1 리니어 레이아웃 방향 설정

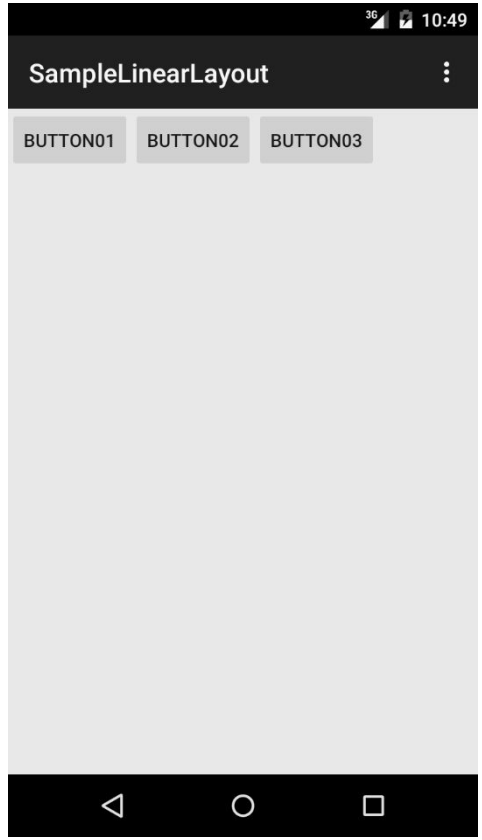


- 프로젝트를 처음 만들었을 때 만들어지는 리니어 레이아웃은 세로 방향으로 되어 있음
- 리니어 레이아웃의 필수 속성임

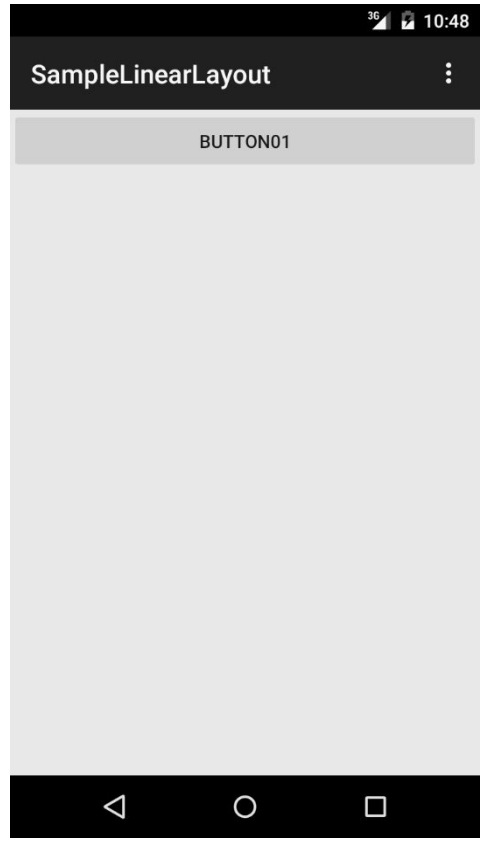
리니어 레이아웃 – 방향 설정하기 (계속)



[세로 방향으로 설정한 경우]



[가로 방향으로 설정을 바꾼 경우]



[버튼의 layout_width 속성을 match_parent로 바꾼 경우]

리니어 레이아웃 – 자바 코드에서 구성하기

- 자바 코드에서 직접 레이아웃 객체를 만들고 파라미터 설정하는 방법

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

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
   LinearLayout.LayoutParams.MATCH_PARENT,
   LinearLayout.LayoutParams.WRAP_CONTENT);

Button button1 = new Button(this);
button1.setText("Button1");
button1.setLayoutParams(params);
mainLayout.addView(button1);

setContentView(mainLayout);
```

1 레이아웃 객체 생성

2 파라미터 설정

3 버튼 객체 생성하여 추가

4 화면 설정

리니어 레이아웃 – 정렬 설정하기

[두 가지 정렬 속성]

정렬 속성	설 명
layout_gravity	- [외부] 부모 컨테이너의 여유 공간에 뷰가 모두 채워지지 않아 여유 공간 안에서 뷰를 정렬할 때
gravity	- [내부] 뷰에서 화면에 표시하는 내용물을 정렬할 때 (텍스트뷰의 경우, 내용물은 글자가 되고 이미지뷰의 경우 내용물은 이미지가 됨)

- **layout_gravity**

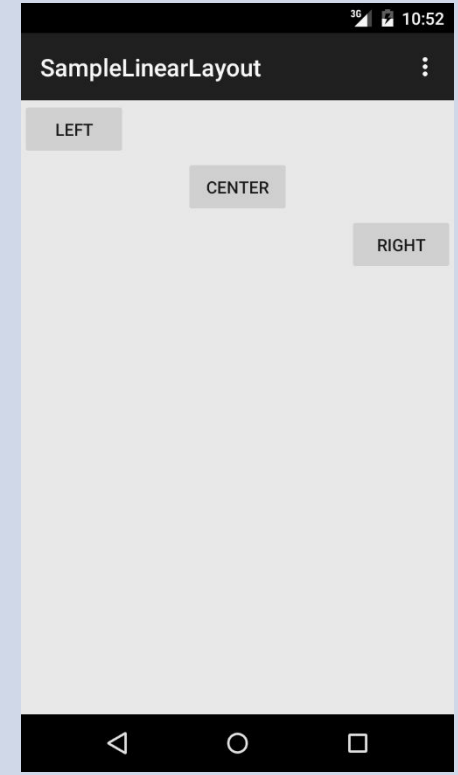
  - 뷰의 layout_width나 layout_height 속성이 match_parent가 아닐 경우에 같이 사용할 수 있음

리니어 레이아웃 – 정렬 설정하기 (계속)

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation= "vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  >
  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity= "left"
    android:text="left"
    />
```

1 리니어 레이아웃 방향 설정

2 첫 번째 버튼 정렬



Continued..

리니어 레이아웃 – 정렬을 위해 사용할 수 있는 값

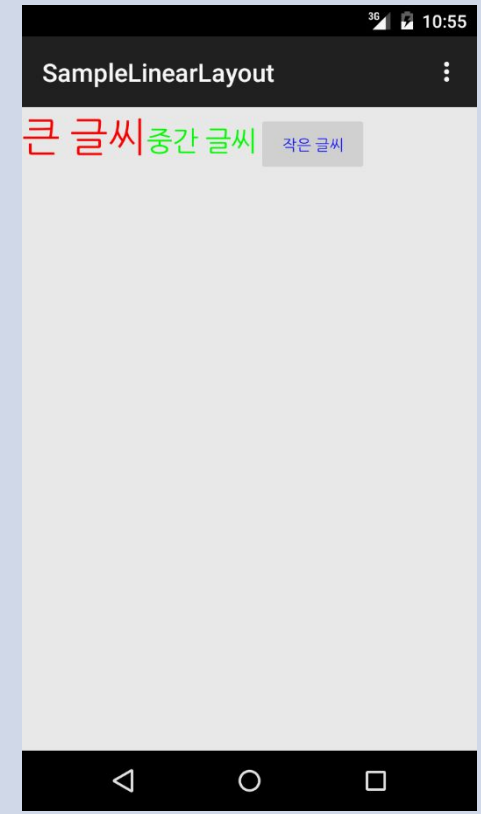
[정렬을 위해 gravity 속성에 지정할 수 있도록 정의된 값]

정렬 속성값	설 명
top	- 대상 객체를 위쪽 끝에 배치하기
bottom	- 대상 객체를 아래쪽 끝에 배치하기
left	- 대상 객체를 왼쪽 끝에 배치하기
right	- 대상 객체를 오른쪽 끝에 배치하기
center_vertical	- 대상 객체를 수직 방향의 중앙에 배치하기
center_horizontal	- 대상 객체를 수평 방향의 중앙에 배치하기
fill_vertical	- 대상 객체를 수직 방향으로 여유 공간만큼 확대하여 채우기
fill_horizontal	- 대상 객체를 수평 방향으로 여유 공간만큼 확대하여 채우기
center	- 대상 객체를 수직 방향과 수평 방향의 중앙에 배치하기
fill	- 대상 객체를 수직 방향과 수평 방향으로 여유 공간만큼 확대하여 채우기
clip_vertical	- 대상 객체의 상하 길이가 여유 공간보다 클 경우에 남는 부분을 잘라내기 - top|clip_vertical 로 설정한 경우 아래쪽에 남는 부분 잘라내기 - bottom|clip_vertical 로 설정한 경우 위쪽에 남는 부분 잘라내기 - center_vertical|clip_vertical 로 설정한 경우 위쪽과 아래쪽에 남는 부분 잘라내기
clip_horizontal	- 대상 객체의 좌우 길이가 여유 공간보다 클 경우에 남는 부분을 잘라내기 - right|clip_horizontal 로 설정한 경우 왼쪽에 남는 부분 잘라내기 - left|clip_horizontal 로 설정한 경우 오른쪽에 남는 부분 잘라내기 - center_horizontal|clip_horizontal 로 설정한 경우 왼쪽과 오른쪽에 남는 부분 잘라내기

리니어 레이아웃 – 글자 아랫줄 정렬

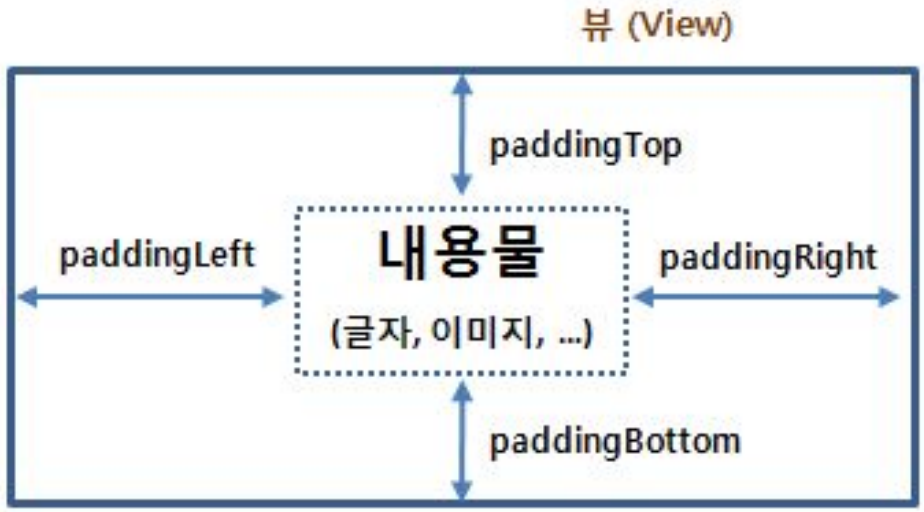
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="horizontal"
  android:layout_width= " match_parent"
  android:layout_height="match_parent"
  android:baselineAligned= "true"
  >
```

1 글자의 아랫줄 맞추기

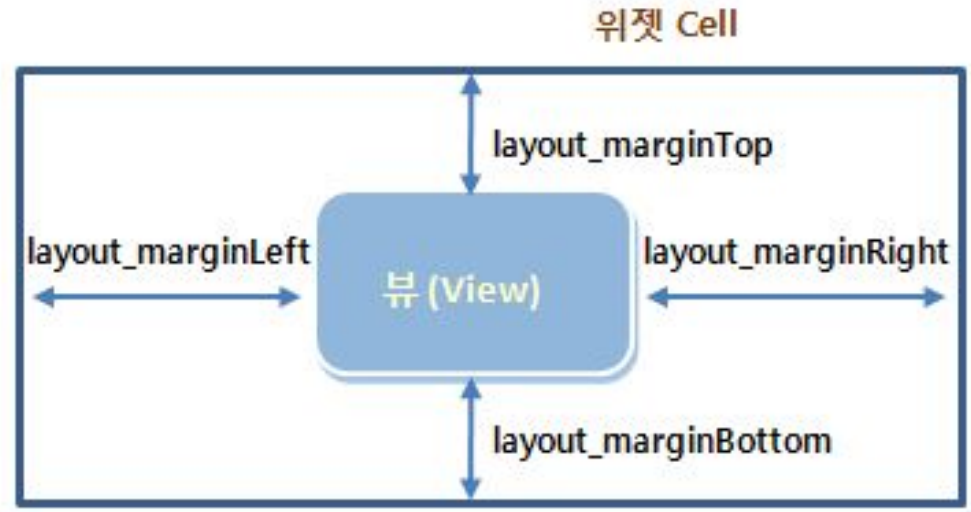


Continued..

리니어 레이아웃 – 여유공간 설정하기



[padding을 이용한 뷰 내부의 여백 주기]



[layout_margin을 이용한 부모 여유공간과의 여백 주기]

- **padding 속성**
  - 뷰 안의 내용물인 텍스트나 이미지와 뷰 안의 영역 사이의 여백을 줄 수 있는 방법

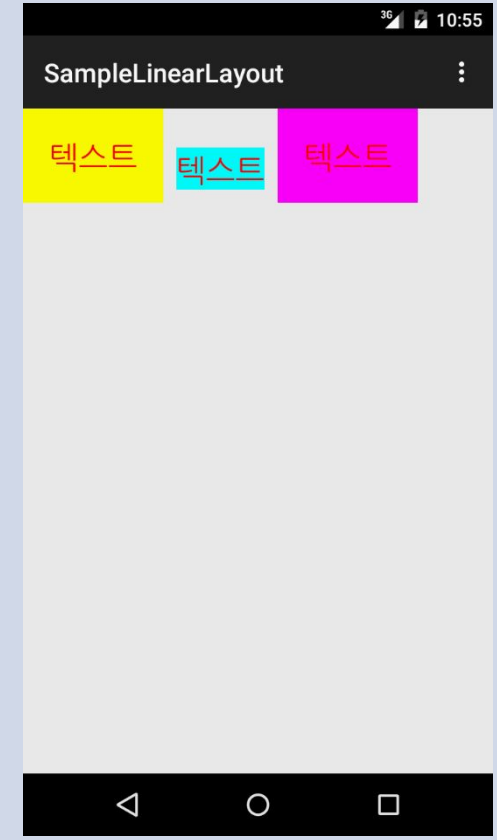
- **layout_margin 속성**
  - 부모 컨테이너의 여유 공간과 뷰 사이의 여백을 줄 수 있는 방법

- **위젯 셀**
  - 위젯이나 뷰들은 부모 컨테이너로부터 할당된 공간을 차지하게 되며 이를 '위젯 셀(cell)'이라고 부름

리니어 레이아웃 – 여유공간 설정하기 (계속)

```
<TextView
  android:id="@+id/button01"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:background="#ffffff00"
  android:text="텍스트"
  android:textColor="#ffff0000"
  android:textSize="24dp"
  android:padding= "20dp"
  />
```

위젯 내부 여백 설정



Continued..

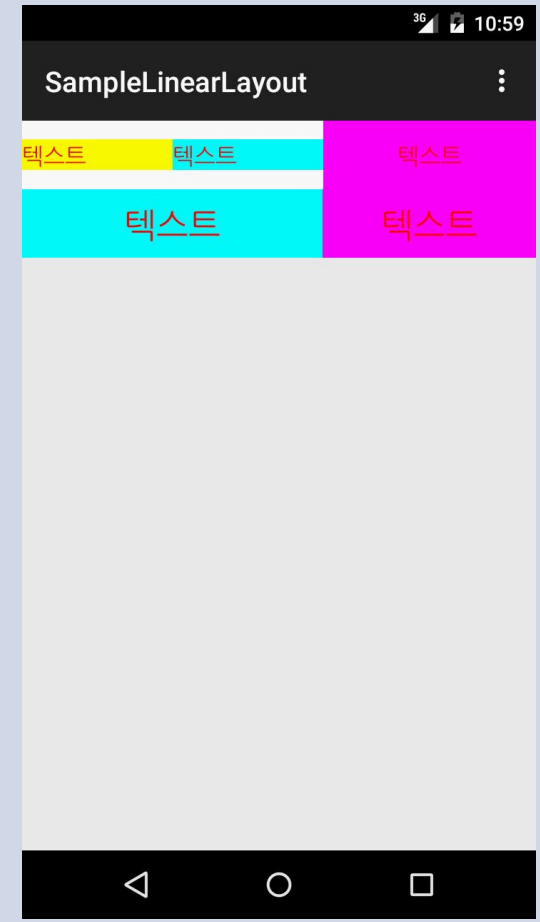
리니어 레이아웃 – 공간가중치 설정하기

- 공간가중치는 같은 부모 뷰에 포함되어 있는 뷰들이 여유공간을 얼마나 차지할 수 있는지를 비율로 지정하는 것
- **android:layout_weight** 속성 사용

리니어 레이아웃 – 공간가중치 설정하기

```
<TextView
    android:id="@+id/button01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#ffffff00"
    android:text="텍스트"
    android:textColor="#ffff0000"
    android:textSize="16dp"
    android:layout_weight= "1"
    />
```

공간 가중치 설정



Continued..

5. RelativeLayout

상대 레이아웃

- 상대 레이아웃은 다른 뷰나 부모 뷰와의 상대적인 위치를 이용해 뷰를 배치하는 방법

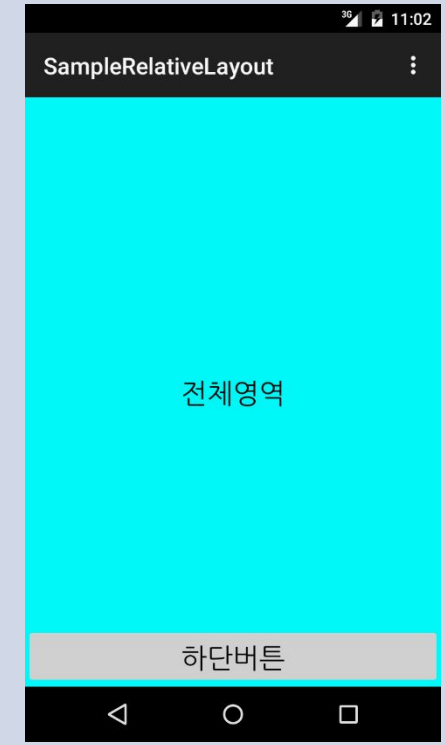
[상대 레이아웃을 이용한 뷰의 배치 방법]

A : 부모 뷰의 위쪽
C : 뷰 **A**의 오른쪽 아래
B : 뷰 **A**의 아래, 뷰 **C**의 왼쪽

상대 레이아웃의 속성 사용

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  >
  <Button
    android:text="전체버튼"
    android:textColor="#ff000000"
    android:textSize="24sp"
    android:background="#ff00ffff"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_centerInParent= "true"
    />
```

1 한 가운데 배치



Continued..

상대 레이아웃의 속성 사용 (계속)

```
<Button
  android:text="하단버튼"
  android:textColor="#ff000000"
  android:textSize="24sp"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_alignParentBottom= "true"
  />
</RelativeLayout>
```

2 아래쪽 배치

상대 레이아웃에서 사용할 수 있는 속성들

[상대 레이아웃에서 부모 컨테이너와의 상대적 위치를 이용하는 속성]

속성	설 명
layout_alignParentTop	- 부모 컨테이너의 위쪽과 뷰의 위쪽을 맞춤
layout_alignParentBottom	- 부모 컨테이너의 아래쪽과 뷰의 아래쪽을 맞춤
layout_alignParentLeft	- 부모 컨테이너의 왼쪽 끝과 뷰의 왼쪽 끝을 맞춤
layout_alignParentRight	- 부모 컨테이너의 오른쪽 끝과 뷰의 오른쪽 끝을 맞춤
layout_centerHorizontal	- 부모 컨테이너의 수평 방향 중앙에 배치함
layout_centerVertical	- 부모 컨테이너의 수직 방향 중앙에 배치함
layout_centerInParent	- 부모 컨테이너의 수평과 수직 방향 중앙에 배치함

상대 레이아웃에서 사용할 수 있는 속성들

[상대 레이아웃에서 다른 뷰와의 상대적 위치를 이용하는 속성]

속성	설 명
layout_above	- 지정한 뷰의 위쪽에 배치함
layout_below	- 지정한 뷰의 아래쪽에 배치함
layout_toLeftOf	- 지정한 뷰의 왼쪽에 배치함
layout_toRightOf	- 지정한 뷰의 오른쪽에 배치함
layout_alignTop	- 지정한 뷰의 위쪽과 맞춤
layout_alignBottom	- 지정한 뷰의 아래쪽과 맞춤
layout_alignLeft	- 지정한 뷰의 왼쪽과 맞춤
layout_alignRight	- 지정한 뷰의 오른쪽과 맞춤
layout_alignBaseline	- 지정한 뷰와 내용물의 아래쪽 기준선(baseline)을 맞춤

6. TableLayout

테이블 레이아웃

- 테이블 레이아웃은 격자 모양으로 뷰를 배치하는 방법

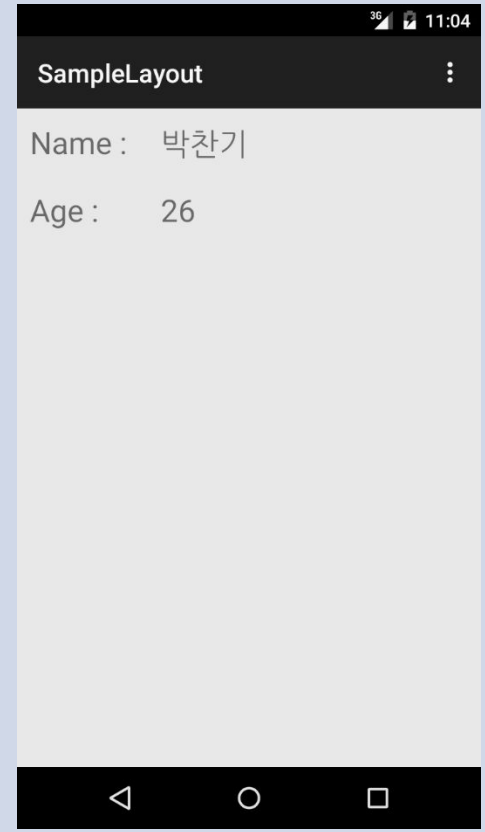
[테이블 레이아웃을 이용한 뷰의 배치 방법]

테이블 레이아웃

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:stretchColumns= "1"
>
<TableRow>
<TextView
  android:text="Name : "
  android:textSize="24dp"
  android:padding="10dp" />
<TextView
  android:text="박찬기"
  android:textSize="24dp"
  android:gravity="left"
  android:padding="10dp" />
</TableRow>
```

1 폭에 맞추어 열을 자동 확장

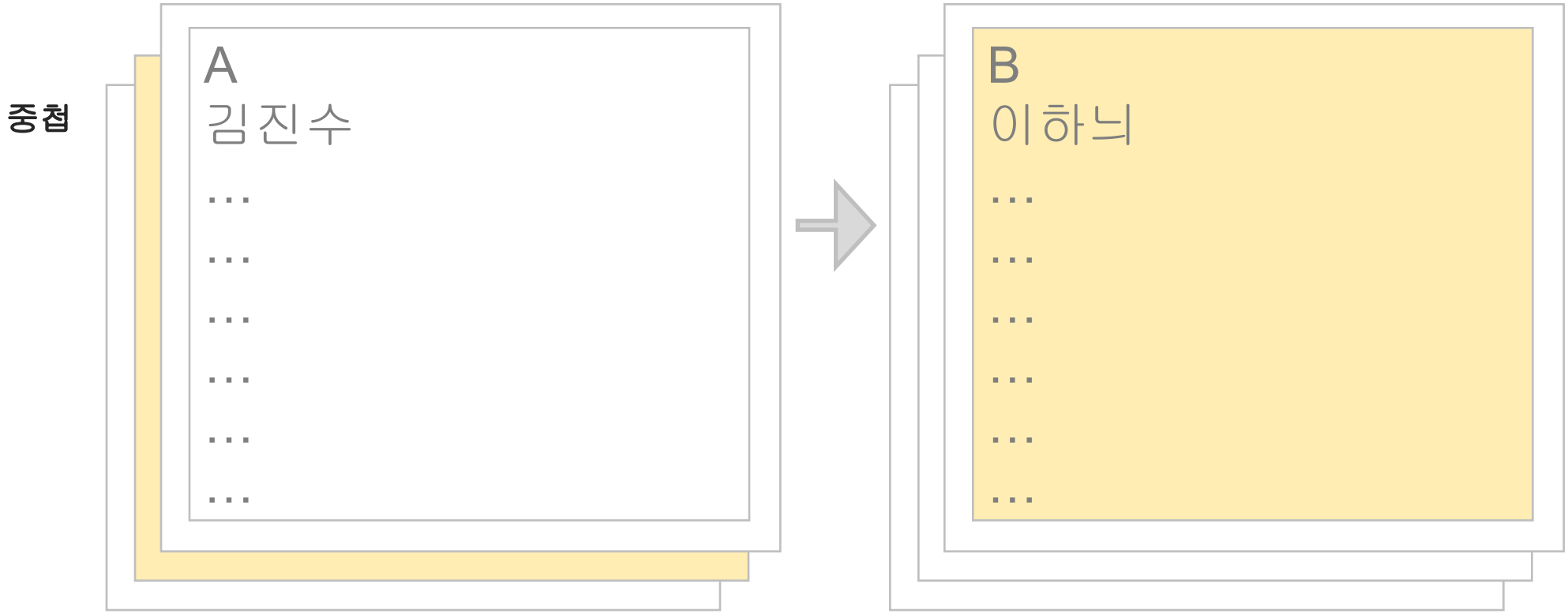
2 테이블의 첫 번째 행



7. FrameLayout

프레임 레이아웃과 뷰의 전환

- 한 번에 하나의 뷰만 보여주며, 다른 뷰들은 그 아래에 중첩되어 쌓임
- 중첩되는 효과와 함께 뷰의 가시성(Visibility) 속성을 이용해 다양한 화면 구성이 가능함



[프레임 레이아웃과 가시성 속성의 사용]

뷰 A에서 뷰 B로 전환

가시성 속성 사용하기

- 사용 예 – **XML** 레이아웃

```
<LinearLayout
   android:id="@+id/layout1"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:visibility="gone"
   >
   <TextView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      />
</ScrollView>
```

- 사용 예 – 소스 코드

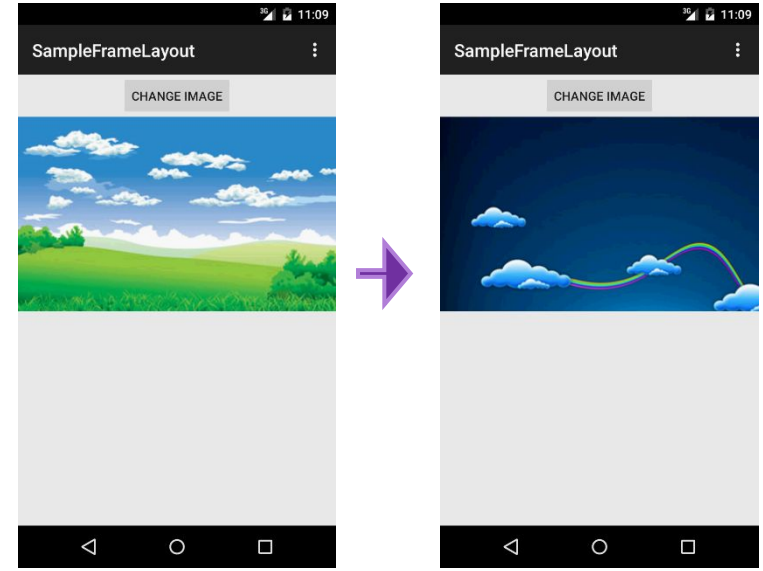
```
layout1.setVisibility(View.GONE);
layout1.setVisibility(View.VISIBLE);
layout1.setVisibility(View.INVISIBLE);
```

프레임 레이아웃과 뷰의 전환

뷰 전환 예제

-프레임 레이아웃을 이용해 뷰를 중첩하여 만들기

-버튼을 누르면 다른 이미지로 전환하기



XML 레이아웃

-레이아웃 코드 작성

메인 액티비티 코드

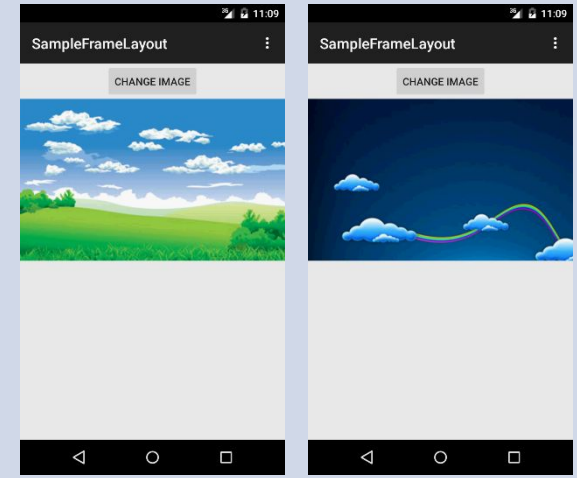
-메인 액티비티 코드 작성

프레임 레이아웃과 뷰의 전환 – XML 레이아웃

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
>
<Button
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:text="Change Image"
/>
<FrameLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
>
```

1 전환 버튼

2 화면 채우기



Continued..

프레임 레이아웃과 뷰의 전환 – XML 레이아웃

```
<ImageView
android:id="@+id/imageView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/dream01"
android:visibility= "invisible"
/>
<ImageView
android:id="@+id/imageView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/dream02"
android:visibility= "visible"
/>
</FrameLayout>
</LinearLayout>
```

3 이미지 뷰 설정

4 이미지 뷰 설정

프레임 레이아웃과 뷰의 전환 – 메인 액티비티 코드

```
private void changeImage() {
if (imageIndex == 0) {
  imageView1.setVisibility(View.VISIBLE);
  imageView2.setVisibility(View.INVISIBLE);
  imageIndex = 1;
   } else if (imageIndex == 1) {
  imageView1.setVisibility(View.INVISIBLE);
  imageView2.setVisibility(View.VISIBLE);
  imageIndex = 0;
 }
}
}
```

1 이미지 뷰 설정

2 이미지 뷰 설정

8. ConstraintLayout

ConstraintLayout

향상된 RelativeLayout + @

RelativeLayout보다 더 유연한 위치 속성

- ●RelativeLayout 뷰 위치 속성
  - ○layout_toRightOf
  - ○layout_toLeftOf
  - ○layout_toTopOf
  - ○layout_toBottomOf
- ●ConstraintLayout 뷰 위치 속성
  - ○layout_constraintTop_toTopOf
  - ○layout_constraintTop_toBottomOf
  - ○layout_constraintBottom_toTopOf
  - ○layout_constraintBottom_toBottomOf
  - ○layout_constraintLeft_toTopOf
  - ○layout_constraintLeft_toBottomOf
  - ○layout_constraintLeft_toLeftOf
  - ○layout_constraintLeft_toRightOf
  - ○layout_constraintRight_toTopOf
  - ○layout_constraintRight_toBottomOf
  - ○layout_constraintRight_toLeftOf
  - ○layout_constraintRight_toRightOf
  - ○left, right 정렬에 대해 start, end속성 지원

9. 기본 위젯들

기본 위젯 – 텍스트뷰의 속성

- **텍스트 뷰**

- **text** : 텍스트 뷰에 보이는 문자열을 설정할 수 있음
- **textColor** : 텍스트뷰에서 표시하는 문자열의 색상을 설정함
  : 색상 설정은 "#AARRGGBB" 포맷을 일반적으로 사용(Alpha, Red, Green, Blue)
  : 투명도를 나타내는 Alpha(색상만 표현할 때 - "FF", 투명 - “00“, 반투명 - "88”)
- **textSize** : 텍스트뷰에서 표시하는 문자열의 크기를 설정함
  ("dp"나 “sp" 또는 "px" 등의 단위 값을 사용함)
- **textStyle** : 텍스트뷰에서 표시하는 문자열의 스타일 속성을 설정함
  ("normal", "bold", "italic" 등의 값을 지정할 수 있음)
- **typeFace** : 텍스트뷰에서 표시하는 문자열의 폰트를 설정함
  ("normal", "sans", "serif", "monospace")
- **singleLine** : 텍스트뷰에서 표시하는 문자열이 한 줄로만 표시되도록 설정함

기본 위젯 – 텍스트뷰의 속성 사용

```
<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:id="@+id/TextView01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background= "#ff000055"
android:padding="3px"
android:text="여기에 사용자 이름을 입력하세요."
android:textSize= "22sp"
android:textStyle= "bold"
android:textColor= "#88ff8888"
android:singleLine= "true"
android:gravity="center"
/>
</LinearLayout>
```

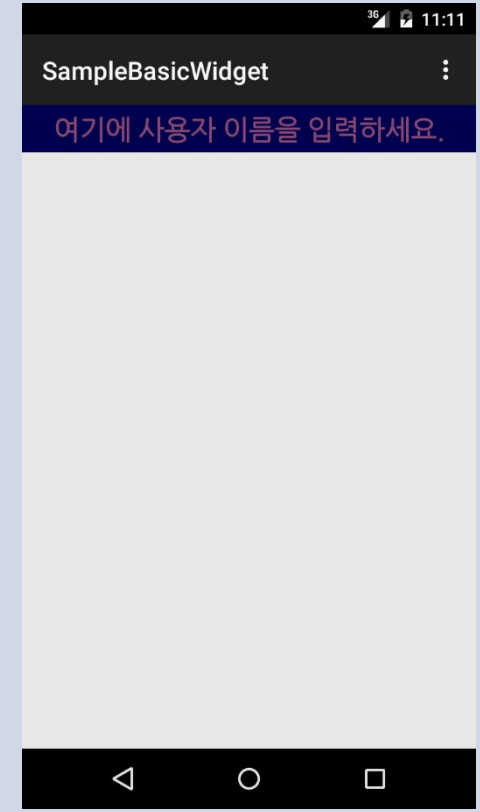
1 배경색 설정

2 크기 설정

3 스타일 설정

4 색상 설정

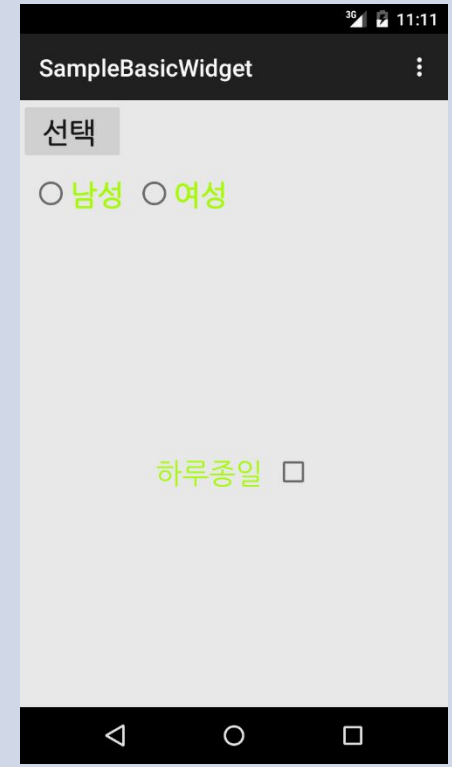
5 한 줄 설정



기본 위젯 – 버튼의 속성 사용

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
>
<Button
android:id="@+id/btnExit"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text=" 선택 "
android:textSize="24dp"
android:textStyle="bold"
android:gravity="center"
/>
```

1 기본 버튼



Continued..

기본 위젯 – 버튼의 속성 사용 (계속)

```
<RadioGroup
android:id="@+id/radioGroup01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="horizontal"
android:paddingLeft="5dp"
android:paddingRight="5dp"
>
<RadioButton
android:id="@+id/radio01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="남성"
android:textColor="#ffaaff10"
android:textStyle="bold"
android:textSize="24dp"
/>
```

2 라디오 그룹

3 첫 번째 버튼

Continued..

기본 위젯 – 버튼의 속성 사용 (계속)

```
<RadioButton
android:id="@+id/radio02"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="여성"
android:textColor="#ffaaff10"
android:textStyle="bold"
android:textSize="24dp"
/>
</RadioGroup>
<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:gravity="center_vertical|center_horizontal"
android:paddingTop="10dp"
>
```

4 두 번째 버튼

Continued..

기본 위젯 – 버튼의 속성 사용 (계속)

```
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="하루종일"
android:textSize="24dp"
android:paddingRight="10dp"
android:textColor="#ffaaff10"
/>
<CheckBox
android:id="@+id/allDay"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
</LinearLayout>
```

5 체크 박스

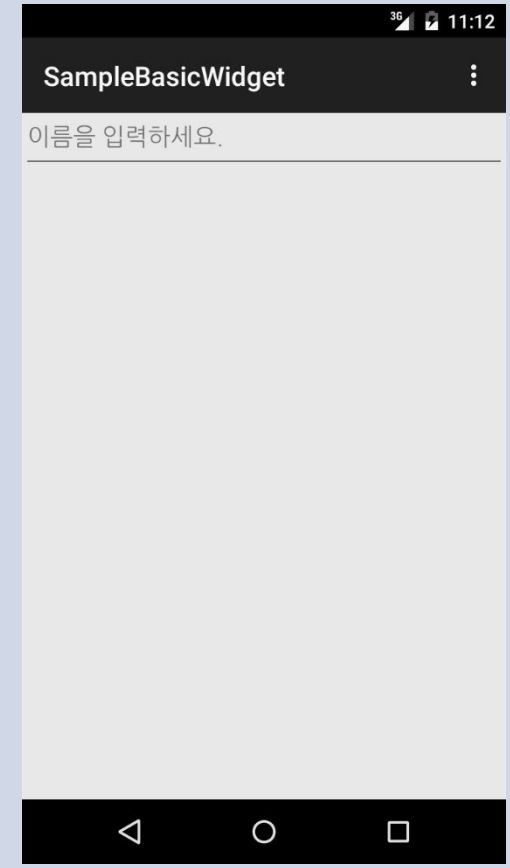
기본 위젯 – 입력상자의 속성 사용

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  >
  <EditText
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:autoText= "true"
    android:capitalize= "words"
    android:hint= "이름을 입력하세요."
    >
  </EditText>
</LinearLayout>
```

1 설정

2 변경

3 표시



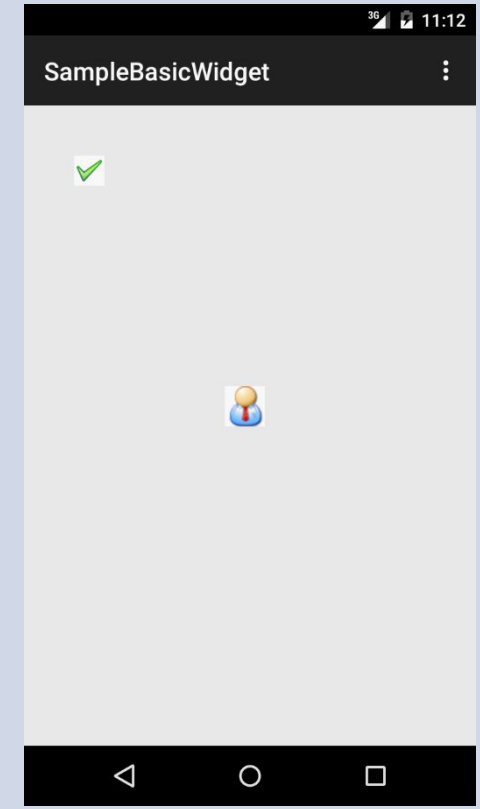
기본 위젯 – 이미지뷰의 속성 사용

```
<ImageButton
  android:id="@+id/ImageButton01"
  android:background="@drawable/ok_btn"
  android:layout_width="24dp"
  android:layout_height="24dp"
  …
  />

<ImageView
  android:id="@+id/ImageView01"
  android:background="@drawable/person"
  android:layout_width="32dp"
  android:layout_height="32dp"
  …
 />
```

1 이미지 버튼

2 이미지 뷰



[References]

• 기본 서적

2016, 정재곤, “Do it! 안드로이드 앱 프로그래밍(개정3판)”, 이지스퍼블리싱(주)

• Android Website

http://www.android.com/

• Google Developer’s Conference

http://code.google.com/events/io/

• Android SDK Documentation