

# 01.안드로이드 인터페이스 기초 뷰(View)



# 목차

---

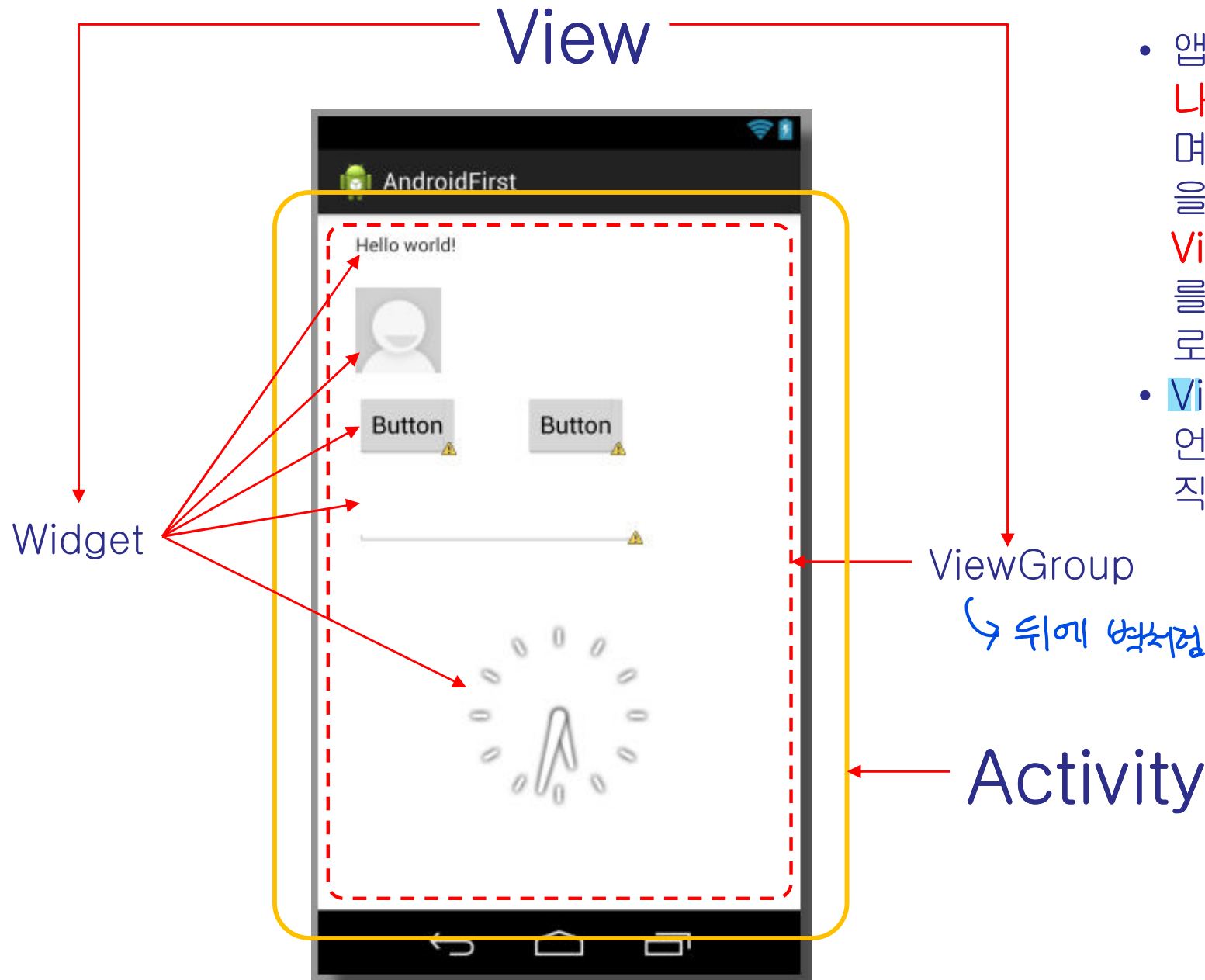
 안드로이드 화면 구성

 위젯(Widget) 과 뷰 그룹(ViewGroup)

 뷰의 기본 속성

 기본 위젯

# 안드로이드 화면 구성



- 앱의 하나의 화면은 하나의 Activity가 관리하며, 화면은 다른 View들을 내부에 담는 ViewGroup과 화면 요소를 구성하는 Widget으로 구성
- View는 주로 XML로 선언하나 java code에서 직접 작성도 가능

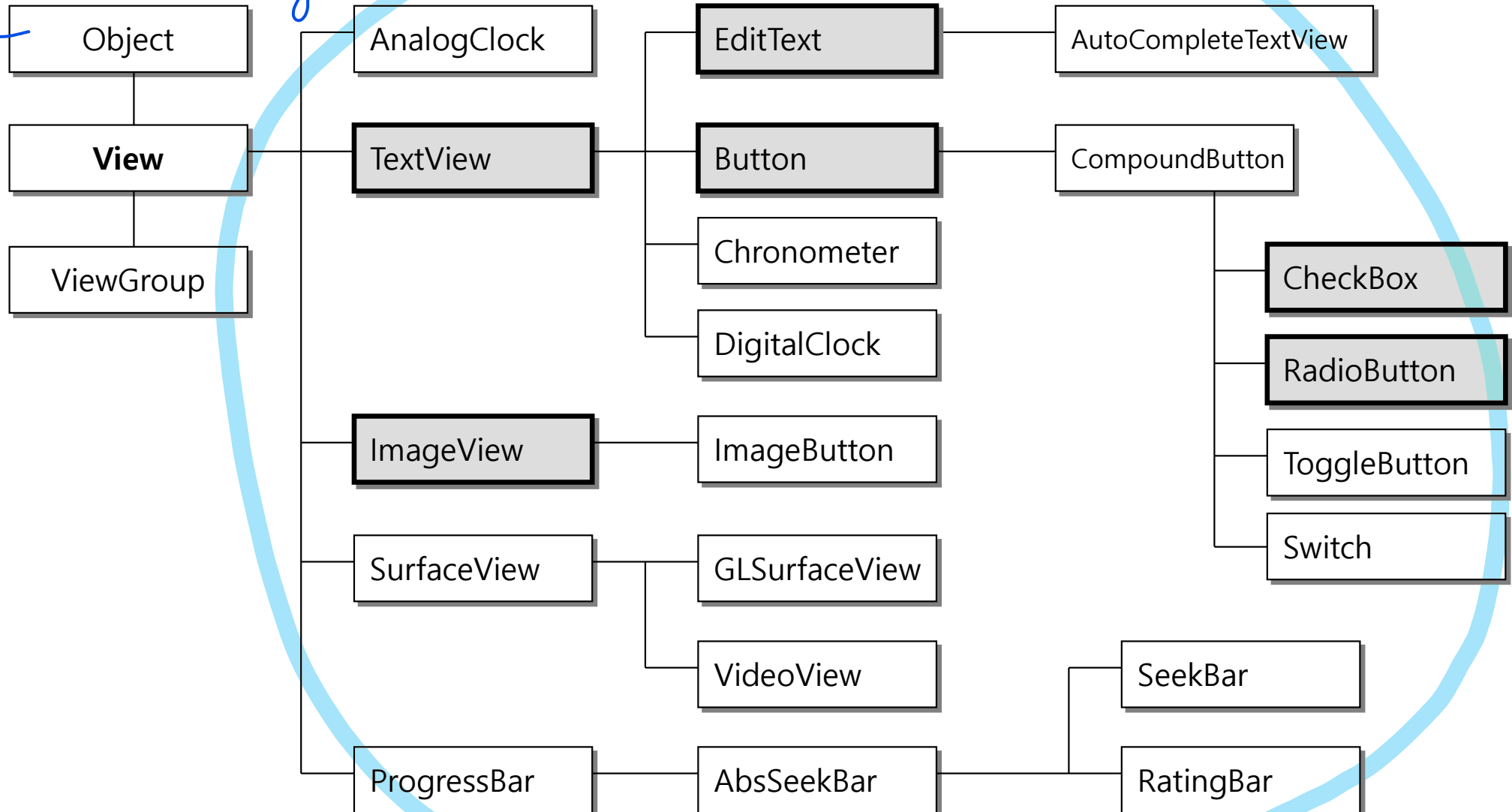
↳ 뒤에 배워줄 ...

# 위젯(Widget)

화면 입출력 요소를 표현하는 뷰 → 상속 관계

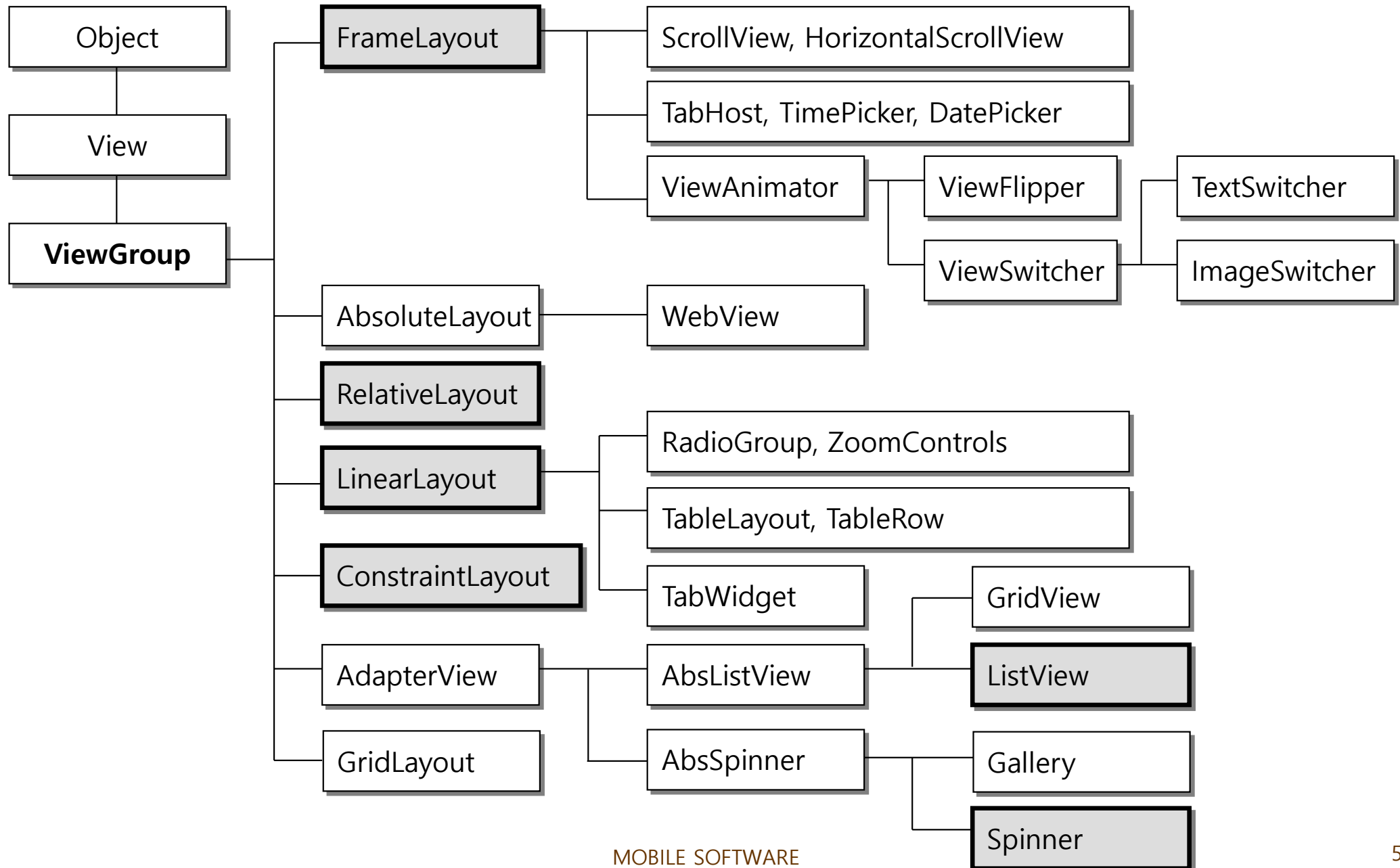
widget

오트라인에서 Any



# 뷰그룹(ViewGroup)

☞ 위젯 또는 다른 뷰를 grouping 하거나 배치할 때 사용하는 뷰



# View 공통 속성 1

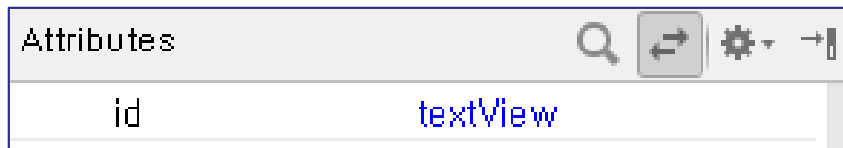
## 📄 ID: 특정 뷰를 참조하기 위한 식별정보

### ◆ 형식

@[+]id/view\_id

- Id를 새로 지을 때만 표시

### ◆ TextView 의 경우



Design 상태에서의 Properties 창

```
<TextView
    android:id="@+id/textView1"
```

XML 에디터 상의 코드

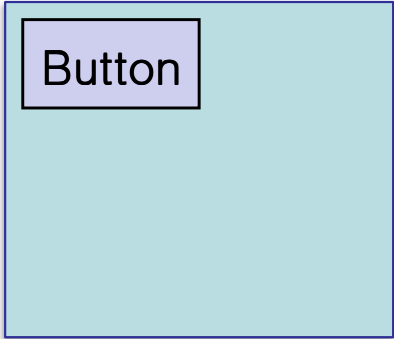
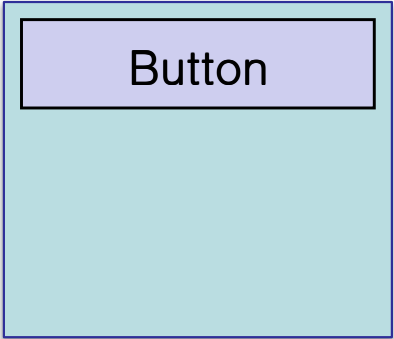
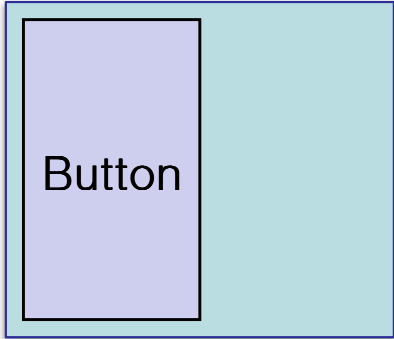
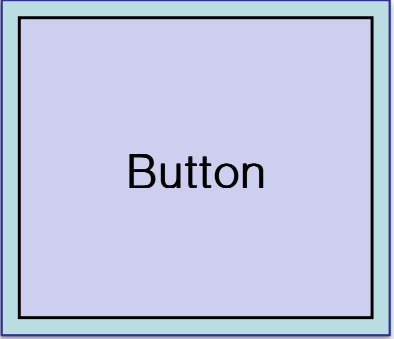
## 📄 layout\_width/layout\_height 필수!

- ◆ 뷰 그룹 안에서의 가로/세로 너비 지정
- ◆ match\_parent(fill\_parent): 담겨 있는 뷰그룹 크기에 맞춤
- ◆ wrap\_content: 담고 있는 항목의 크기에 맞춤
- ◆ 상수값: 값만큼 크기 고정 (dp 또는 dip 단위 사용)

어떤 폰트

# View 공통 속성 2

## wrap\_content vs. match\_parent

<i>layout_width</i>	wrap_content	match_parent	wrap_content	match_parent
<i>layout_height</i>	wrap_content	wrap_content	match_parent	match_parent
view	<p>layout</p> 	<p>layout</p> 	<p>layout</p> 	<p>layout</p> 

## 임의 크기로 지정할 때 → 값을 단위와 함께 입력

- ◆가능하면 논리 단위를 사용할 것(기기별 해상도 차 고려)
- ◆dp (density-independent pixel): view의 크기 지정 시
- ◆sp (scale-independent pixel): 문자열의 크기 지정 시

# View 공통 속성 3

background: 뷰의 배경 지정

- ◆ 색상 (#RGB | #ARGB | #RRGGBB | #AARRGGBB)
- ◆ 속성 (visible | invisible | gone)

padding: 뷰와 내용물 사이의 간격

- ◆ 예: 버튼 안의 텍스트

visibility: 가시성 결정

clickable/longClickable : 클릭/롱클릭 이벤트 반응 여부

focusable: 키보드 입력 가능 여부 T/F



# 기본 위젯 - TextView

## 문자열 표시 위젯

### 주요 속성

#### ◆text: 표시할 문자열

1. XML 상에서 직접 값 지정

```
<TextView
    android:text="hello!" />
```

strings.xml 에  
hello\_world 라는  
id 로 추가한  
string 을 읽어옴

2. 다른 xml 의 값을 읽어와 값 지정

```
<TextView
    android:text="@string/hello_world" />
```

findViewById() 를 사용하여 레이아웃에 배치한 위젯을 객체로 가져온 후 수행

```
TextView textView1
= findViewById<TextView>
(R.id.text_view)
```

3. 코드에서 직접 텍스트 입력

```
textView1.setText("hello!")
```

코틀린에서 .text 사용

#### ◆textColor : 문자열 색상

#### ◆textSize (sp 단위 사용 권장) : 문자 크기

#### ◆textStyle (normal | bold | italic)

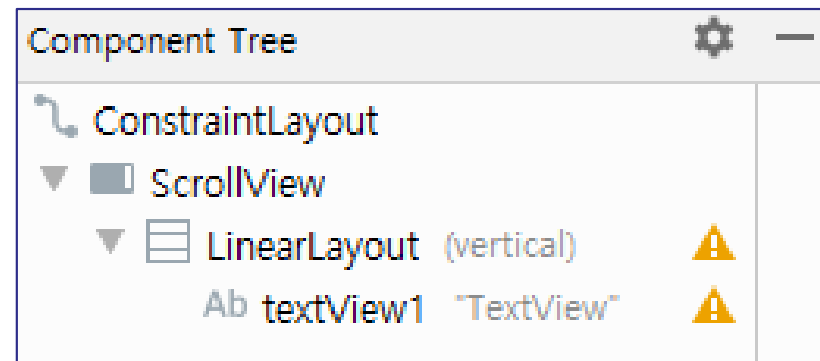
#### ◆typeface (font - 폰트는 한정적, 필요시 직접 추가)

#### ◆singleLine : 여러 줄 표기 가능 지정

# Scroll 가능한 TextView

## ❏ TextView의 출력 내용이 길어 Scroll이 필요할 경우

- ◆ [ScrollView] 사용
- ◆ [Design] 상태에서 [Palette] 항목 중 [Containers]의 [ScrollView] 선택 후 레이아웃에 배치
- ◆ [Properties]에서 배치한 [ScrollView] 속성 변경
  - ScrollView를 적절한 크기로 배치
  - ScrollView 내부의 LinearLayout 에 View 배치
- ◆ 스크롤 대상 View 를 [ScrollView] 내부에 배치가 어려울 때
  - [Component Tree] 활용



- TextView 일 경우 텍스트를 화면에 모두 표시할 수 없을 경우 스크롤이 가능해짐

# 기본 위젯 - ImageView

## 이미지를 표시하는 위젯

- ◆ ImageView 자체는 이미지를 표시하는 액자 역할
- ◆ 화면 배치 시 기본 이미지 설정 필요

## 주요 속성 *src Compat*

- ◆ src: 출력할 이미지의 아이디
  - @drawable/target\_id or @mipmap/target\_id
- ◆ maxHeight/maxWidth
- ◆ adjustViewBounds: 가로/세로 비율(종횡비) 유지 여부
- ◆ cropToPadding: 여백 조정 여부
- ◆ tint: 이미지 위에 색조 추가
- ◆ scaleType: 이미지 원본의 크기를 ImageView에 맞추기 위한 확대/축소 방식 지정

# 이미지 소스의 지정

## res/mipmap-xxx 폴더

- ◆ 앱에서 고정적으로 사용할 ~~간단한~~ 이미지를 저장 → drawable
- ◆ 동일한 이미지를 해상도를 달리하여 해상도별 폴더에 저장
  - 기기의 해상도에 따라 자동으로 선택됨

## 이미지 파일 명 지정 시 주의사항!

- ◆ 투명도 등을 조종하기 위해서 가능하면 png 파일 사용
- ◆ 이미지 **파일명**이 아이디가 되므로 반드시 명명 규칙 준수
  - **영어소문자, 밑줄, 숫자만 가능(한글 X, 대문자X)**
  - **확장자로 구분하지 않으므로 파일명을 다르게 사용하여야 함 →**  
dog.jpg == dog.png (중복으로 사용하면 안 됨)

확장자도 소문자!

## 이미지 파일 지정 시 주의 사항은 모든 XML 파일명에도 동일하게 적용

# 실습

- ❏ 레이아웃에 세 개의 Button을 배치하고 각각 다음 방법으로 자신의 이름을 표시하시오.
  - ◆ 속성에 직접 이름 기록
  - ◆ strings.xml 에 이름을 기록하고 TextView 에 연결
  - ◆ code 에서 직접 이름을 기록
- ❏ 위 실습의 버튼색과 글자색, 그리고 레이아웃 배경색을 변경하시오.
- ❏ 위 실습의 버튼 크기를 정해진 상수 및 dp 단위를 사용하여 변경하시오.
- ❏ ScrollView 를 사용하여 Text 를 표시하시오. (Text 내용은 기존 text 를 읽어 새로운 text 를 추가하는 코드를 반복)
- ❏ 웹 상에서 이미지를 다운 받아 ImageView 에 표시하시오.

# 기본 위젯 - Button

## ☞ 사용자 클릭 이벤트 처리

## ☞ 주요 속성

- ◆ text: 버튼에 표시할 문자열
- ◆ ~할 때 onClick: 버튼을 클릭하였을 때 동작할 메소드 명, 개발자가 임의로 지정 (자바 소스 코드 부분에 작성)
- ◆ onClick 메소드 구현: fun 메소드명 (매개변수: View) {…}

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:text="버튼 클릭" />
```

activity\_main.xml

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    fun onClick(view : View) {
        Toast.makeText(this, "버튼을 클릭함", Toast.LENGTH_SHORT).show()
    }

}
```

MainActivity.kt

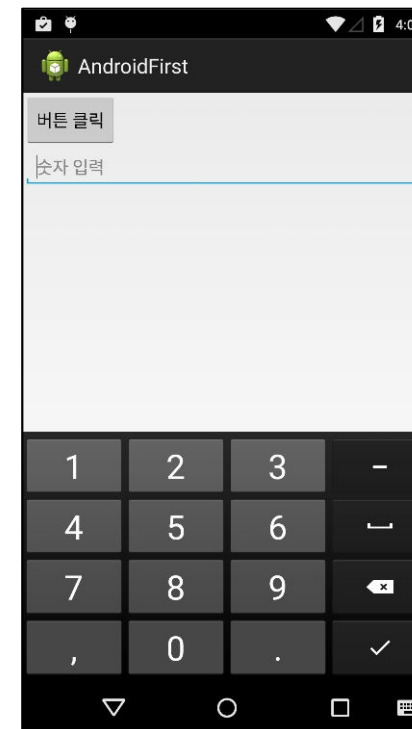
# 기본 위젯 - EditText

## ☞ 사용자의 키보드 입출력 처리

## ☞ 주요 속성

- ◆ text: EditText 입력/출력 텍스트
- ◆ hint: 입력할 텍스트 값 안내, 값 입력 시 자동으로 없어짐
- ◆ inputType: 입력할 값의 유형(일반/암호/숫자/전화번호 등)  
→ 유형에 따라 사용하는 키보드의 종류가 자동으로 변경

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="숫자 입력"
    android:inputType="number" >
</EditText>
```



# EditText 값 처리

## EditText에 입력한 문자열을 코드에서 처리

- ◆ XML의 EditText (id: editText) 객체를 찾아 지정

```
val editText = findViewById<EditText>(R.id.editText)
```

- ◆ EditText 객체 속성 **text** 를 사용하여 현재 쓰여진 값을 가져옴 (CharSequence 반환) → **getter** X

```
val text : String = editText.text.toString()
```

- **.toString()** 로 String 변환 → text 는 Editable 객체 반환

- ◆ EditText 객체에 **setText()** 를 사용하여 문자열 표시

```
editText1.setText("문자열");
```



# 토스트(Toast)

☞ 화면 하단에 일정 시간 동안 나타나는 안내용 출력창

- ◆ 간단한 상태 정보 표시
- ◆ 디버깅 시 값 확인용으로도 사용할 수 있음

☞ Toast 사용 예

```
Toast myToast = Toast.makeText(this, "Hello", Toast.LENGTH_SHORT);
// Toast myToast = Toast.makeText(this, R.string.hello_world, Toast.LENGTH_SHORT);

// myToast.setGravity(Gravity.CENTER_HORIZONTAL, 0, 0);
// myToast.setMargin(float, float);
// myToast.setView(View view);
```

```
myToast.show();
//myToast.cancel();
```

• 출력 메시지

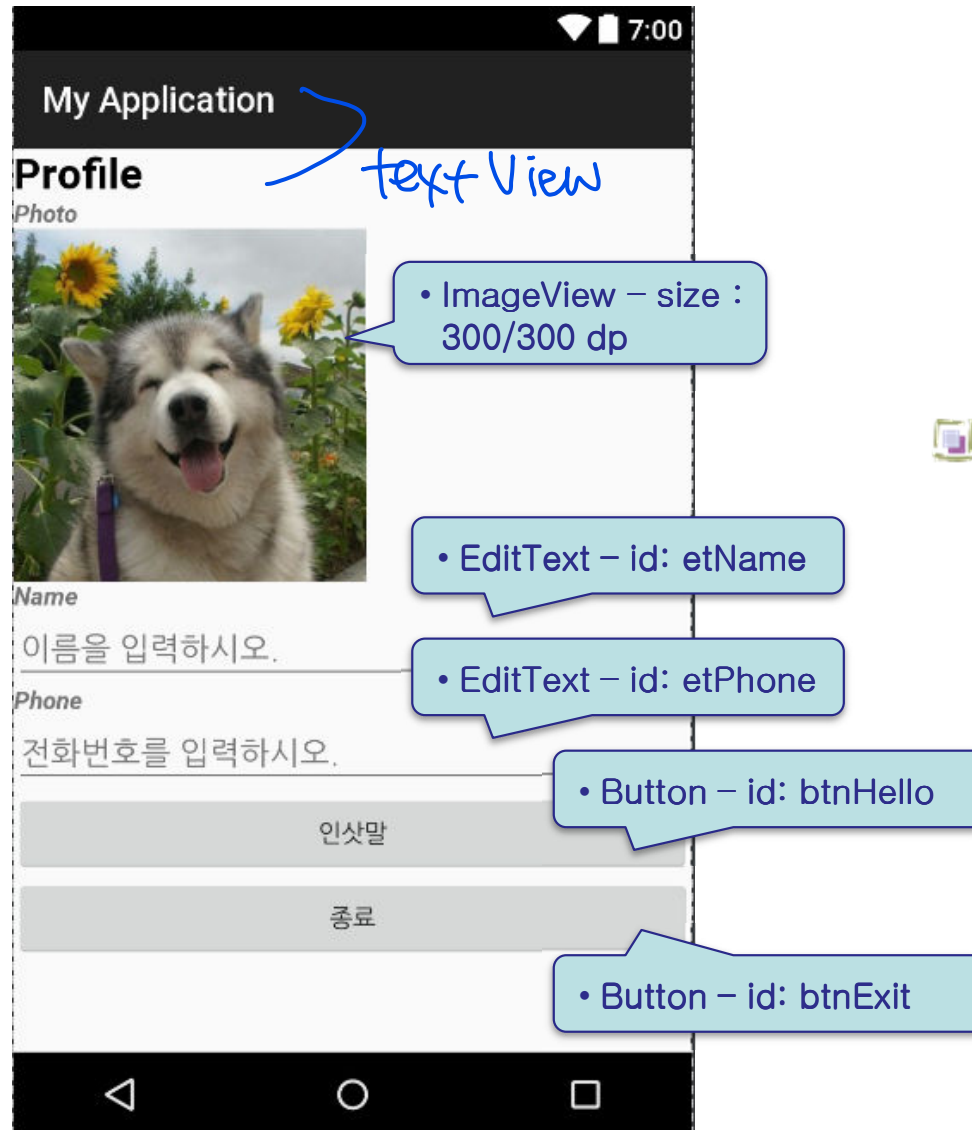
• show() 를 실행시켜야  
화면에 표시

```
Toast.makeText(this, "Hello", Toast.LENGTH_SHORT).show();
```

• Context  
(Activity 가 상속함)

• 출력 시간  
(LENGTH\_SHORT 또는 LENGTH\_LONG)

다음과 같이 앱을 구성하시오.



btnHello 클릭 시 다음 문장을 Toast로 출력

- ◆ 안녕하세요, 저는 XXX 입니다.  
전화번호는 XXX 입니다.
- ◆ XXX 부분은 etName과 etPhone  
에 입력한 값 *myEdit.text*

btnExit 클릭 시 앱 종료

- ◆ finish() → Activity의 멤버 메소드, Activity 종료 시 사용

다음과 같은 화면을 구성한 후 아래 내용과 같이 동작 하도록 앱을 구성하시오.



• EditText ( id: editText )

• Button ( id: btnOne, btnTwo, btnThree, btnClear )

숫자 버튼을 누르면 계속 숫자 추가

clear 를 누를 경우 EditText 의 글자를 모두 지움