

# **모바일프로그래밍기초**

## **- 안드로이드(Android) -**

## □ 목차

- 액티비티/리소스
  - 액티비티 상태와 생명주기
  - 리소스 접근
  
- 안드로이드 기능/실습 2
  - 레이아웃 구성
  
- 안드로이드 기능/실습 3
  - 액티비티

□ **액티비티/리소스**

## □ 애플리케이션 개념

### ○ 액티비티와 태스크

- 액티비티: 같은 애플리케이션 내에 존재하는 액티비티 뿐만 아니라 다른 애플리케이션 내에 존재하는 액티비티까지 **호출 가능**  
예) 서적 관리 애플리케이션 = 서적 관리 일반 기능 + 특정 기능 (바코드 스캔)
- 한 애플리케이션에서 다른 애플리케이션의 컴포넌트를 거의 자유 자재로 접근 가능  
→ 파일의 구성 면에서는 애플리케이션의 경계가 뚜렷함, 실제로 애플리케이션의 실행 면에서는 애플리케이션간 경계가 없음
- 각 컴포넌트들, 특히 화면에 표시되면서 사용자와 상호작용하는 액티비티는 애플리케이션 단위보다 태스크(Task) 단위로 관리
- 태스크(Task): 사용자가 실질적으로 “하나의 애플리케이션처럼” 느끼는 **액티비티들의 집합**임.

## □ 애플리케이션 개념

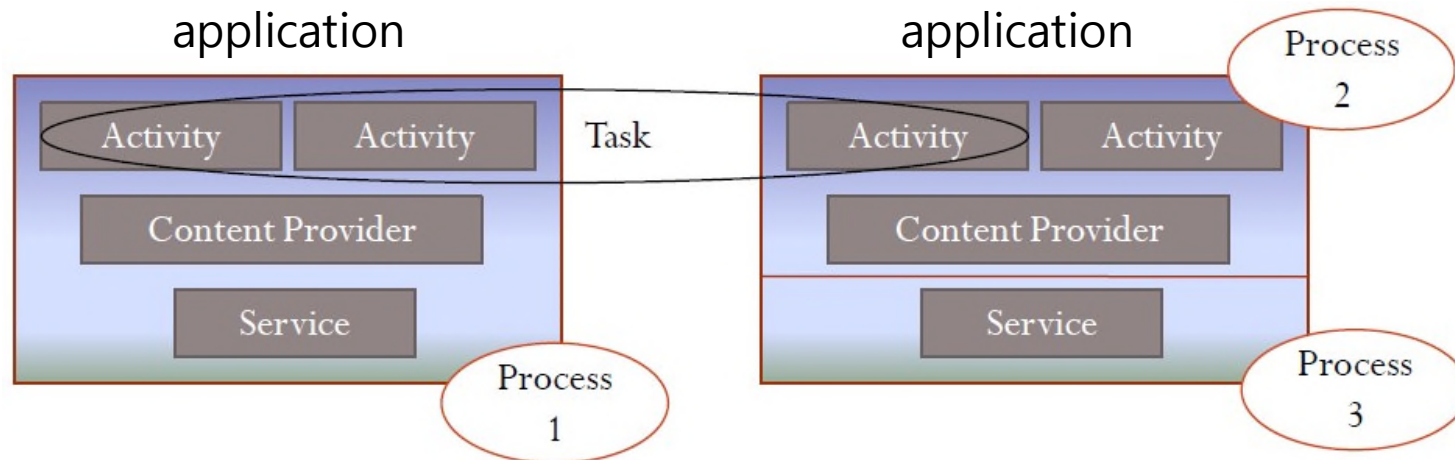
### ○ 태스크와 프로세스

#### □ 태스크

- ❖ 연관된 액티비티의 집합
- ❖ 다수의 프로세스와 APK에 걸쳐 존재 가능
- ❖ 다른 APK의 액티비티 호출 가능

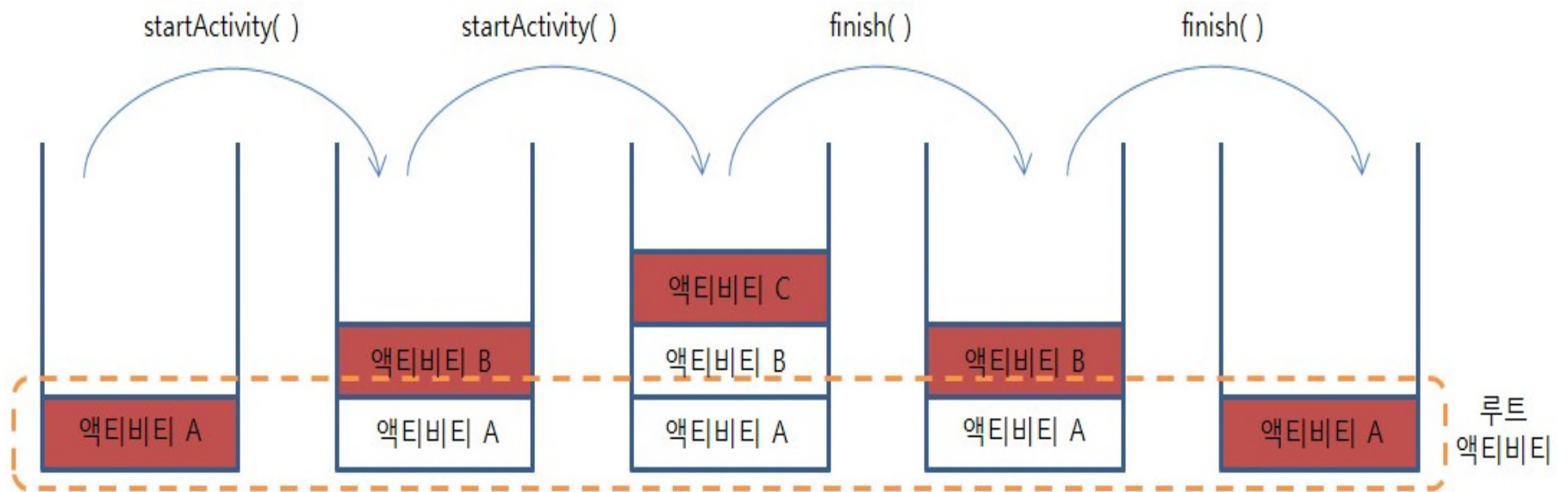
#### □ 프로세스

- ❖ 커널 프로세스
- ❖ 기본적으로 APK는 하나의 프로세스에서 동작
- ❖ 하나의 APK에서 다수의 프로세스 매핑 가능



## □ 애플리케이션 개념

### ○ 액티비티 스택



## □ 액티비티 상태와 생명주기

### ○개요

- 안드로이드는 모바일 기기에서 구동되기 때문에 데스크탑 애플리케이션에 비하여 더욱 효율적 메모리 관리가 요구
- 애플리케이션 컴포넌트의 중요 요소인 액티비티도 효율적인 메모리 관리를 위하여 액티비티 생성 및 소멸 과정인 **생명주기**가 있음
- 액티비티가 중지 혹은 정지 상태일 경우, 해당 프로세스가 메모리에서 제거될 수 있음. 사용자에게 다시 보여질 때 이전 상태로 복구
- 액티비티는 **액티비티 스택(activity stack)**을 통해 관리

액티비티는 사용자와 상호 작용하는 단위이며 일반적으로 전체 화면을 차지

## □ 액티비티 상태와 생명주기

### ○ 액티비티의 3가지 상태

#### □ 활성(active) 혹은 실행(running) 상태

- ❖ 전경 화면에 있을 경우
- ❖ 해당 액티비티가 사용자 동작에 대한 초점을 가짐
- ❖ 사용자와 상호 작용 가능

#### □ 중지(paused) 상태

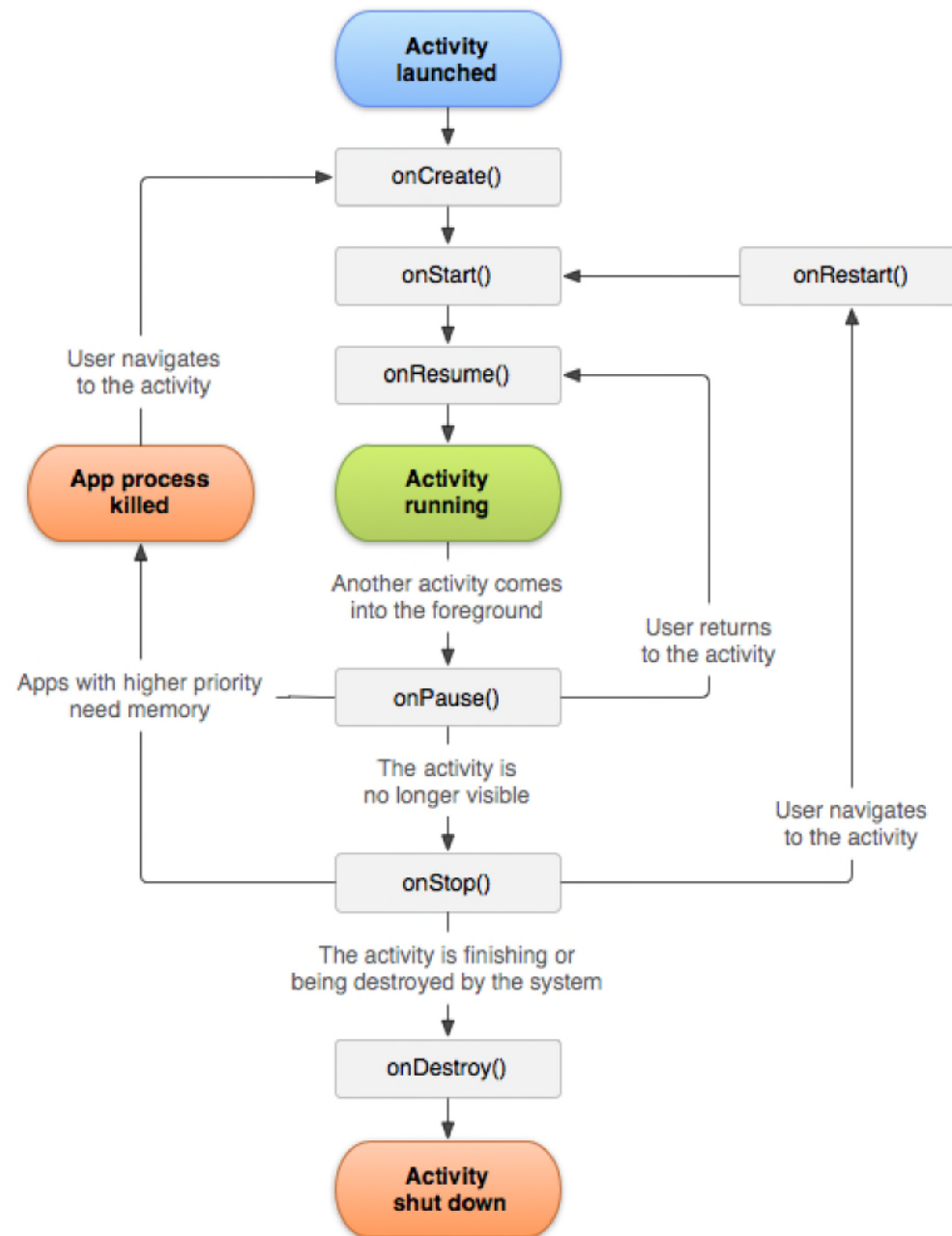
- ❖ 초점을 보유하지 않았지만 사용자에게 일부 보임
- ❖ 메모리가 극도로 부족할 경우 시스템에 의하여 강제 종료
- ❖ 사용자와 상호 작용 불가

#### □ 정지(stopped) 상태

- ❖ 사용자에게 전혀 보이지 않지만 여전히 모든 상태와 멤버 정보 유지
- ❖ 다른 컴포넌트가 메모리를 요청하면 시스템에 의하여 강제 종료



## □ 액티비티 생명주기



## □ 액티비티 상태와 생명주기

### ○ 액티비티를 위한 콜백 메소드

#### □ onCreate()

- ❖ 액티비티가 생성될 때 처음으로 호출되는 메소드
- ❖ 전역 상태의 모든 리소스를 초기화. 예) layout과 data binding 등
- ❖ 액티비티는 기본적으로 버튼, 리스트, 체크박스, 입력 표시줄 등과 같은 위젯들이 배치되어 있는 레이아웃을 구성하고, 위젯들이 사용자와 상호작용을 하는 코드를 포함

#### □ onStart()

- ❖ 액티비티가 초기화 과정을 마친 후 사용자에게 보여줄 준비가 되었을 때 호출

#### □ onResume()

- ❖ 액티비티가 사용자에게 보여지고 사용자의 입력을 처리할 수 있음
- ❖ 액티비티 스택의 최상위에 위치

## □ 액티비티 상태와 생명주기

### ○ 액티비티를 위한 콜백 메소드

#### □ onPause()

- ❖ 사용자와 상호작용이 중지
- ❖ 액티비티 종료 등을 대비해 상태를 저장하는 등의 작업을 수행

#### □ onStop()

- ❖ 더 이상 액티비티가 사용자에게 보이지 않음
- ❖ 더 이상 액티비티 스택의 최상위에 위치하지 않음

#### □ onDestroy()

- ❖ 존재하는 모든 리소스를 해제
- ❖ 시스템 내에 액티비티가 존재하지 않게 됨

#### □ onRestart()

- ❖ onStop() 메소드 호출 이후 정지 상태에 있던 액티비티를 다시 화면에 표시할 때 호출

## □ 액티비티 상태와 생명주기

### ○ 액티비티 상태 저장하기

- 일반적으로 정지된 액티비티는 사용자가 다시 사용할 것을 대비하여 메모리에 상주. 만약 메모리가 부족하게 되면 강제 종료
- 액티비티를 다시 호출하면,
  - ❖ 강제 종료된 상태: 다시 액티비티를 생성한 후 액티비티를 실행
  - ❖ 강제 종료되지 않은 상태: 액티비티를 다시 만들 필요가 없고 액티비티를 다시 시작하여 화면에 표시

## □ 액티비티 상태와 생명주기

### ○ 액티비티 상태 저장하기

#### □ onSaveInstanceState()

- ❖ onPause() 혹은 onStop() 이후 메모리가 부족할 경우 프로세스가 메모리에서 제거될 수 있음
- ❖ 메모리에서 제거되기 전 액티비티 상태를 저장
- ❖ 매개변수로 액티비티의 동적 상태를 기록할 수 있는 번들(Bundle) 오브젝트를 가짐

#### □ onRestoreInstanceState()

- ❖ onCreate() 혹은 onStart() 이후 저장된 액티비티 상태를 복원

#### □ onSaveInstanceState()와 onRestoreInstanceState()는 생명주기 메소드가 아니기 때문에 항상 호출되지 않음. 개발자가 해당 코드의 메소드를 재정의하여 구현

#### □ 매개변수로 사용되는 savedInstanceState는 저장된 인스턴스의 상태, 즉 액티비티의 UI 상태를 의미

## □ 리소스 이해

### ○ 리소스란?

- 애플리케이션 = 기능 + 리소스
- 기능은 애플리케이션 실행에 관계되는 모든 알고리즘을 포함하는 코드
- 리소스는 애플리케이션이 사용하는 자산
  - ❖ 텍스트 문자열, 이미지, 아이콘, 오디오, 동영상 등
  - ❖ 레이아웃
- 리소스를 소스 코드와 분리시 장점
  - ❖ 유지 보수 용이
  - ❖ 언어와 문화권에 맞는 애플리케이션의 현지화(localization) 가능

### ○ 리소스 저장

- 리소스는 Java 소스 코드와 별도로 외부 파일에 저장
- 리소스 파일은 대부분 XML 파일로 저장
- 이미지와 같은 원본 자료 파일은 그 자체로 저장
- 모든 리소스는 /res 디렉토리의 하위 디렉토리에 저장되며, 하위 디렉토리 이름은 **소문자+숫자+밑줄**로만 구성
  - ❖ 그래픽 및 애니메이션 파일 등 일부 리소스 파일은 파일 이름과 동일한 이름의 변수로 참조되기 때문에 Java 식별자 형태의 파일 이름으로 명명
- 같은 형식의 리소스는 동일한 하위 디렉토리에 저장
- 자동 생성되는 리소스 디렉토리: drawable, layout, values
- aapt(Android Asset Packaging Tool)가 리소스를 모두 파악하여 자원을 접근하기 위한 변수의 정의를 담은 ~/gen/R.java 파일을 생성



### ○ 기본적인 리소스 형식과 파일 이름

| 리소스 형식   | 디렉토리     | 권장 파일 이름               | 엘리먼트 이름                   |
|----------|----------|------------------------|---------------------------|
| 문자열      | values   | strings.xml            | <string>                  |
| 문자열 배열   |          | arrays.xml             | <string-array>            |
| 색상 값     |          | colors.xml             | <color>                   |
| 크기       |          | dimens.xml             | <dimen>                   |
| 단순 표시    |          | drawables.xml          | <drawable>                |
| 스타일 및 테마 |          | styles.xml, themes.xml | <style>                   |
| 그래픽      | drawable | drawables.xml          |                           |
| 애니메이션    | anim     |                        | <set>, <alpha>, <scale> 등 |
| 메뉴       | menu     |                        | <menu>                    |
| XML      | xml      |                        |                           |
| 원본       | raw      |                        |                           |
| 레이아웃     | layout   |                        |                           |
| 아이콘      | mipmap   |                        |                           |

## □ 리소스 형식

| 리소스 형식      | 디렉터리     | 리소스 설명<br>(권장 파일명, 엘리먼트 이름 등)   |
|-------------|----------|---|
| 속성 애니메이션    | animator | Property Animation 관련 정보가 설정된 xml 파일<br>Property Animation 시스템은 거의 모든 항목을 애니메이션으로 만들 수 있음<br>객체의 속성을 애니메이션으로 보여줄 수 있으며 기존 애니메이션(anim) 보다 더 강력한 기능을 제공 |
| 애니메이션       | anim     | 애니메이션 정보가 설정된 xml 파일<br>회전, 크기 확대축소, 알파값 등의 효과를 뷰에 적용하기 위해 만든 정보<br><set>, <alpha>, <scale>, ...  |
| 색상 상태 목록    | color    | Color State List Resource를 정의하는 xml 파일<br>색상이 적용되는 View 객체의 상태에 따라 색상을 변경함<br>xml 파일에 상태 목록을 설명할 수 있으며 단일 <selector> 요소 내에 <item> 요소에 정의              |
| 그래픽         | drawable | 비트맵, 나인패치, xml 등...   |
| 앱 아이콘       | mipmap   | 런처 아이콘  |
| 레이아웃        | layout   | 레이아웃 정의하는 xml 파일  |
| 메뉴          | menu     | 옵션 메뉴, 컨텍스트 메뉴 또는 하위 메뉴 등과 같은 앱 메뉴를 정의하는 xml 파일<br><menu>, <item>, ...  |
| 원본          | raw      | 원시 형태로 저장하기 위한 임의의 파일   |
| 폰트          | font     | .ttf, .otf 또는 .ttc 확장자가 붙은 글꼴 파일이나 <font-family> 요소를 포함한 xml 파일   |
| XML         | xml      | Resources.getXML()을 호출하여 런타임에 읽을 수 있는 임의의 xml 파일<br>다양한 xml 구성 파일을 여기에 저장   |
| 문자열         | values   | strings.xml, 문자열 관리   |
| 문자열 배열 등... |          | arrays.xml 등... 리소스 배열  |
| 색상 값        |          | colors.xml 색상 정보  |
| 크기          |          | dimens.xml, <dimen>...  |
| 스타일 및 테마    |          | styles.xml, themes.xml  |

## □ 리소스 접근

### ○ 코드에서 문자열 참조

- R.java 파일에 정의된 R 클래스와 하위 클래스를 이용
- 코드에서 리소스를 접근하려면 R 클래스와 하위 클래스의 멤버 변수를 통하여 접근.

예) hello라는 문자열을 접근하려면

```
String myString = getResources().getString(R.string.hello);
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Android!</string>
    <string name="app_name">Android</string>
</resources>
```

- ❖ **getResource()**: 리소스를 대표하는 Resources 객체를 반환
- ❖ **getString()**: 인수로 주어진 리소스 식별자에 해당하는 문자열의 실제 값을 반환
- ❖ R.string.hello는 R 클래스의 하위 클래스 string에 있는 멤버 변수 hello를 참조

## □ 리소스 접근

### ○ 코드에서 문자열 배열 참조

#### □ 문자열 배열 fruits의 접근

```
String[] fruits = getResources().getStringArray(R.array.fruits);
```

#### □ 문자열 배열 리소스 파일(~/res/values/arrays.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="fruits">
        <item>apple</item>
        <item>banana</item>
    </string-array>
    <string-array name="animal">
        <item>tiger</item>
        <item>lion</item>
    </string-array>
</resources>
```

## □ 리소스 접근

### ○ 코드에서 색상 참조

#### □ 문자열 배열 textColor의 접근

```
int myColor = getResources().getColor(R.color.textColor);
```

#### □ 색상 리소스 파일(~/res/values/colors.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="backgroundColor">#006600</color>
    <color name="textColor">#ffeec</color>
</resources>
```

## □ 리소스 접근

### ○ 코드에서 크기 참조

- 텍스트 크기 참조를 위한 `textPointSize`의 접근

```
float myTextSize = getResources().getDimension(R.dimen.textPointSize);
```

- 리소스 파일(~/`res/values/dimens.xml`)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="smallSize">6pt</dimen>
    <dimen name="textPointSize">11pt</dimen>
    <dimen name="largeSize">20pt</dimen>
</resources>
```

## □ 리소스 접근

### ○ 코드에서 레이아웃 참조

#### □ 레이아웃 내부에 정의된 ImageView01의 접근

```
ImageView iv = (ImageView)findViewById(R.id.ImageView01);
```

#### □ 레이아웃 파일(~/res/layout/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:background="#ffffff">
    <ImageView
        android:id="@+id/ImageView01"
        android:layout_width="fill_parent"
        ...
    ...
</LinearLayout>
```

## □ 리소스 접근

### ○ 코드에서 이미지 참조

- ~/res/drawable 디렉토리에 추가한 flag.png 파일 접근

```
ImageView iv = (ImageView)findViewById(R.id.ImageView01);  
iv.setImageResource(R.drawable.flag)
```

- 레이아웃 파일(~/res/layout/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:background="#ffffffff">  
    <ImageView  
        android:id="@+id/ImageView01"  
        android:layout_width="fill_parent"  
        ...  
    ...  
</LinearLayout>
```



## □ 리소스 접근

### ○ 리소스에서 리소스 참조

#### □ 참조 방법

@[패키지이름:]리소스형식/리소스이름

예) 애플리케이션 이름을 hello 문자열과 동일하게 하는 경우: strings.xml을 수정

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, ColorSize!</string>
    <string name="app_name">@string/hello</string>
</resources>
```

### ○ 시스템 리소스 참조

- android.R의 하위 클래스에는 다양한 시스템 리소스 보유
  - ❖ 표준 시스템 색상
  - ❖ 시스템 스타일과 테마
  - ❖ 애플리케이션 프로그램 썸네일(thumbnail) 이미지와 아이콘
  - ❖ 오류 문자열과 표준 버튼 텍스트
  - ❖ 페이드인/아웃을 위한 애니메이션 시퀀스
  - ❖ ...

## **□ 안드로이드 기능/실습 2**

### **- 레이아웃 구성 기초**

## □ 레이아웃 구성

### ○기능/실습 2

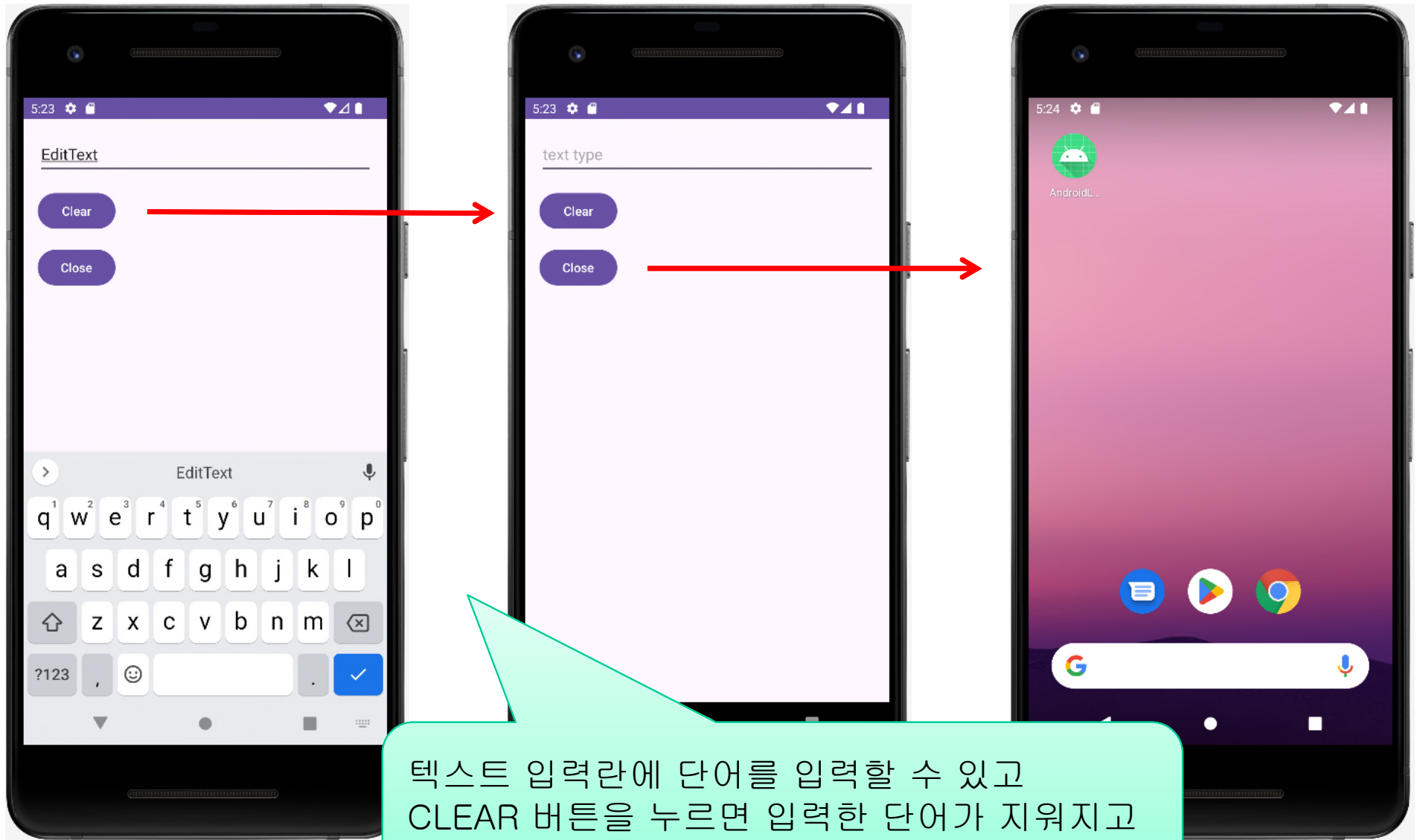
#### □ 내용

- ❖ 레이아웃을 구성하는 기초 방법을 습득하고 구성된 요소(기능/실습 2에서는 버튼)에 따른 명령 작성을 통한 레이아웃 구성 방법과 기초 코드 처리 과정 진행

#### □ 목적

- ❖ 레이아웃 설계를 위한 기초 내용 습득
  - 레이아웃 구성 방법 습득
  - 구성된 레이아웃에 대한 기초 코드 처리 방법 습득
  - ConstraintLayout 소개

## □ 결과 실행 모습



## □ 레이아웃 구성

### ○ ConstraintLayout

- Design Tool 중심의 작업
- 일정한 비율 구성, 쉽고 빠른 배치 등의 장점을 가짐
- 기준점 설정 등, 세부적인 기능 습득 과정이 일부 필요

### ○ 각 레이아웃 별 장, 단점이 존재

- RelativeLayout, LinearLayout, TableLayout, FrameLayout 등, 다양한 레이아웃이 존재
- 앞으로의 기능/실습을 통해서 LinearLayout, FrameLayout 등 각각의 개별 레이아웃 기능 습득 진행

## □ 레이아웃 구성

### ○ 다른 레이아웃 변경이 필요한 경우

- 해당 레이아웃 코드를 필요한 레이아웃 코드로 변경 작성하거나 추가적인 레이아웃 코드 생성시 해당 레이아웃으로 생성

❖ 예) ConstraintLayout → LinearLayout으로 변경

❖ 선택한 레이아웃에 따라 추가적으로 속성 정의 필요

```
<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.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:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".MainActivity">
```

```
    <TextView...>
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

## □ 레이아웃 구성

### ○레이아웃 구성을 위한 위젯 구성

선택한 위젯의 속성

텍스트 입력을 위한 공간  
Plain Text 선택

Attributes

editText01

Declared Attributes

| Attribute              | Value        |
|------------------------|--------------|
| id                     | editText01   |
| layout_width           | 0dp          |
| layout_height          | wrap_content |
| layout_constraintEn... | parent       |
| layout_constraintSt... | parent       |
| layout_constraintTo... | parent       |
| layout_marginStart     | 16dp         |
| layout_marginTop       | 16dp         |
| layout_marginEnd       | 16dp         |
| ems                    | 10           |
| hint                   | text type    |
| id                     | editText01   |
| inputType              | text         |
| minHeight              | 48dp         |

Layout

Constraint Widget

Constraints (4)

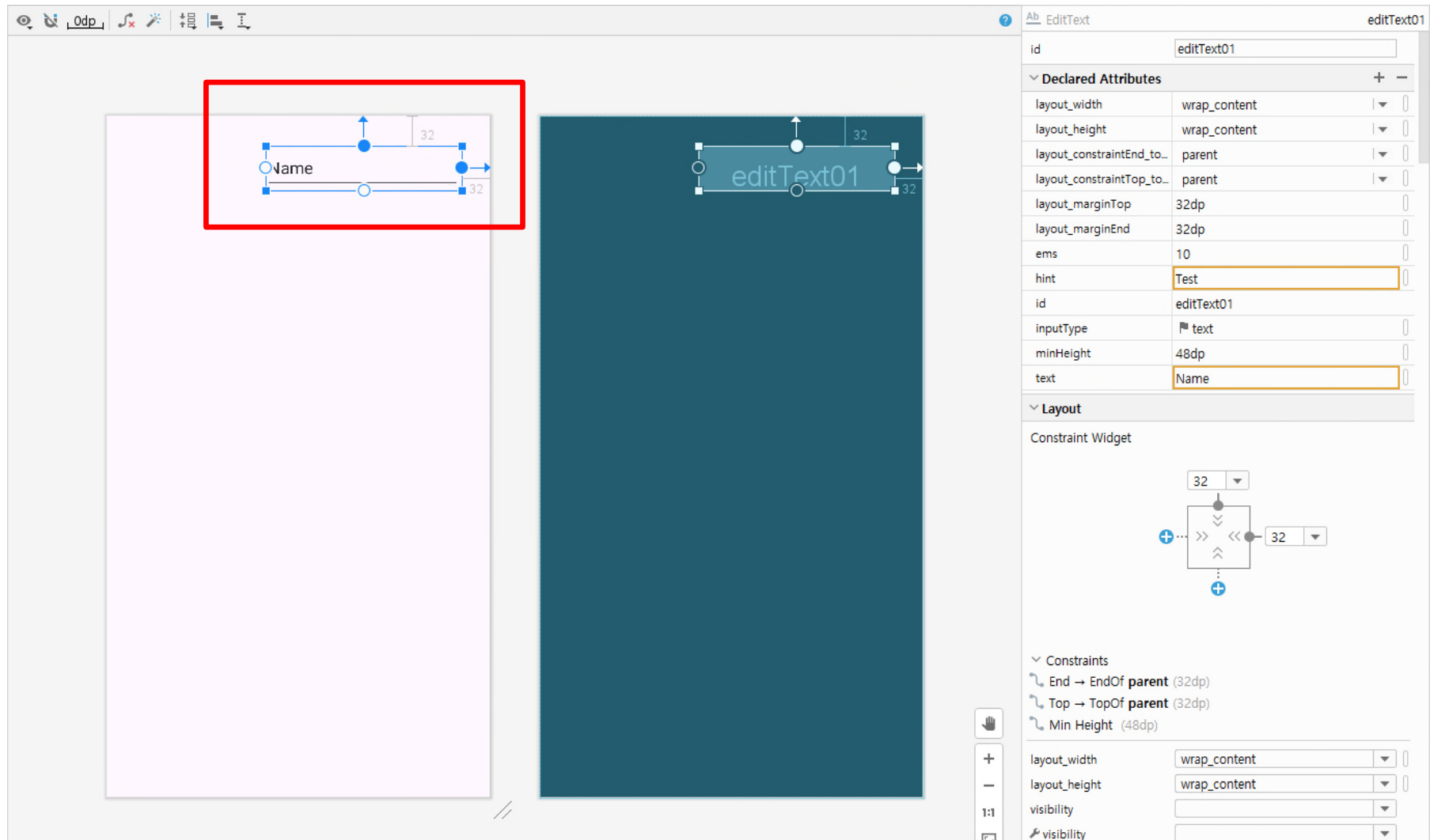
| Attribute     | Value        |
|---------------|--------------|
| layout_width  | 0dp          |
| layout_height | wrap_content |
| visibility    |              |
| visibility    |              |

Blueprint : 위젯의 종류와 테두리를 표시  
위젯 간 관계를 설정할 때 유용



## □ 레이아웃 구성

### ○필요한 위치 관계 정보 설정 예



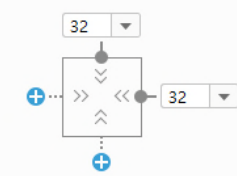
The screenshot displays the Android Studio IDE interface for configuring an `EditText` widget. The left pane shows a visual representation of the widget with its constraints. The right pane shows the 'Ab EditText' properties panel with various attributes and constraints.

**Declared Attributes:**

| Attribute                               | Value                     |
|---|---------------------------|
| <code>layout_width</code>               | <code>wrap_content</code> |
| <code>layout_height</code>              | <code>wrap_content</code> |
| <code>layout_constraintEnd_to...</code> | <code>parent</code>       |
| <code>layout_constraintTop_to...</code> | <code>parent</code>       |
| <code>layout_marginTop</code>           | <code>32dp</code>         |
| <code>layout_marginEnd</code>           | <code>32dp</code>         |
| <code>ems</code>                        | <code>10</code>           |
| <code>hint</code>                       | <code>Test</code>         |
| <code>id</code>                         | <code>editText01</code>   |
| <code>inputType</code>                  | <code>text</code>         |
| <code>minHeight</code>                  | <code>48dp</code>         |
| <code>text</code>                       | <code>Name</code>         |

**Layout:**

Constraint Widget



**Constraints:**

- End → EndOf parent (32dp)
- Top → TopOf parent (32dp)
- Min Height (48dp)

**Properties:**

| Property                   | Value                     |
|----------------------------|---------------------------|
| <code>layout_width</code>  | <code>wrap_content</code> |
| <code>layout_height</code> | <code>wrap_content</code> |
| <code>visibility</code>    | <code>visible</code>      |

## □ 레이아웃 구성

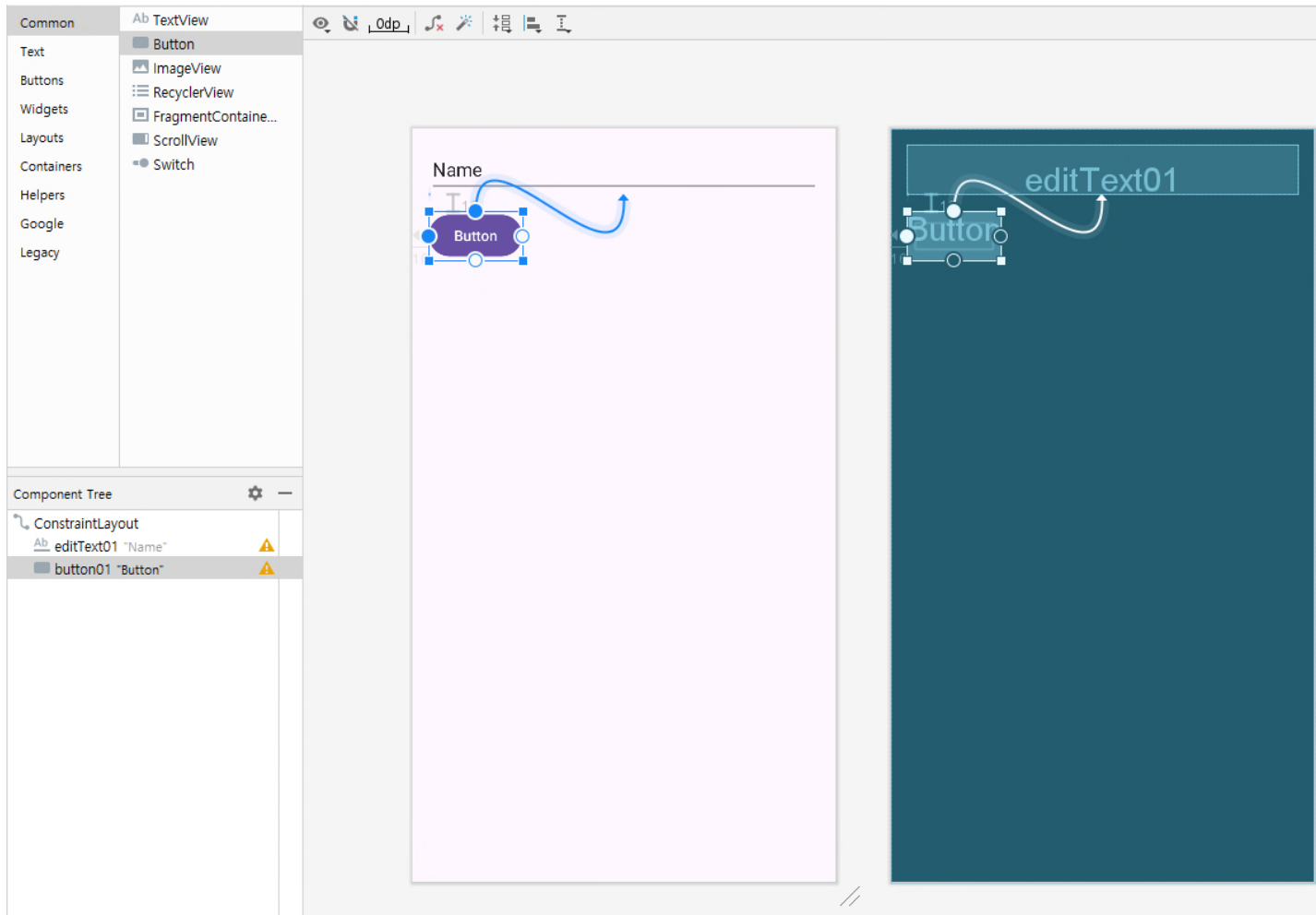
### ○필요한 위치 관계 정보 및 필요 속성 정의

The screenshot displays the Android Studio interface. On the left, the layout editor shows a design view of a text input field labeled 'Name' on a light pink background. On the right, the properties panel for the selected 'EditText' widget is shown, enclosed in a red border. The panel includes sections for 'Declared Attributes' and 'Layout'. The 'Declared Attributes' section lists various properties such as 'layout\_width', 'layout\_height', 'layout\_constraintEnd\_to...', 'layout\_constraintTop\_to...', 'layout\_marginTop', 'layout\_marginEnd', 'ems', 'hint', 'id', 'inputType', 'minHeight', and 'text'. The 'Layout' section shows a 'Constraint Widget' diagram and a list of 'Constraints' including 'End → EndOf parent (32dp)', 'Top → TopOf parent (32dp)', and 'Min Height (48dp)'. The 'text' property is set to 'Name' and the 'hint' property is set to 'Test'.

선택 위젯에 대한 개별 속성 설정

## □ 레이아웃 구성

### ○레이아웃 구성을 위한 위젯을 선택하여 화면 구성



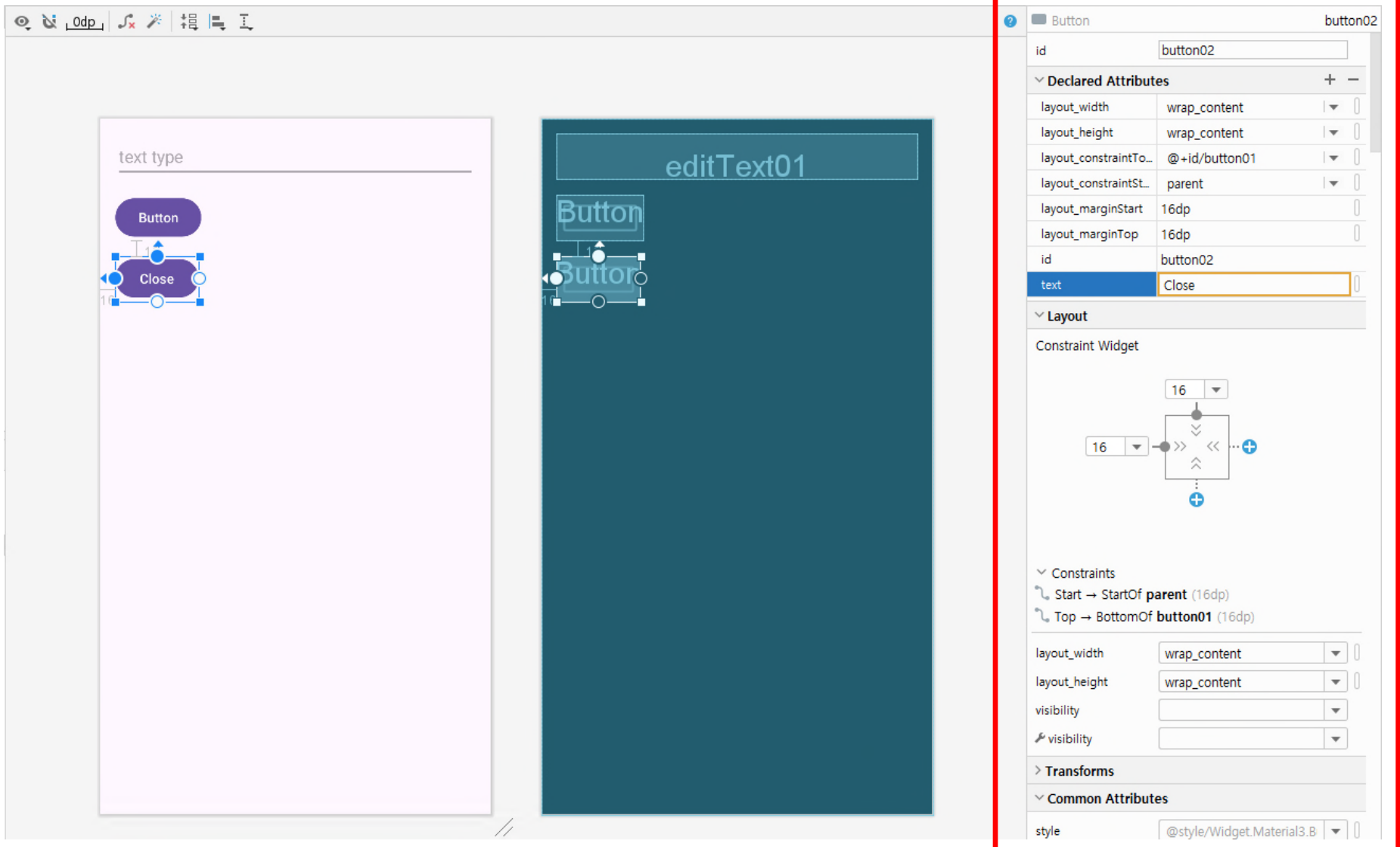
클릭 처리를 위한  
위젯 선택  
Button 선택

- 클릭 처리를 버튼에서  
만 지원하는 것은 아님
- 사용자의 직관적 인지  
에 도움

## □ 레이아웃 구성

○ Design 레이아웃을 이용하여 필요한 내용을 구성

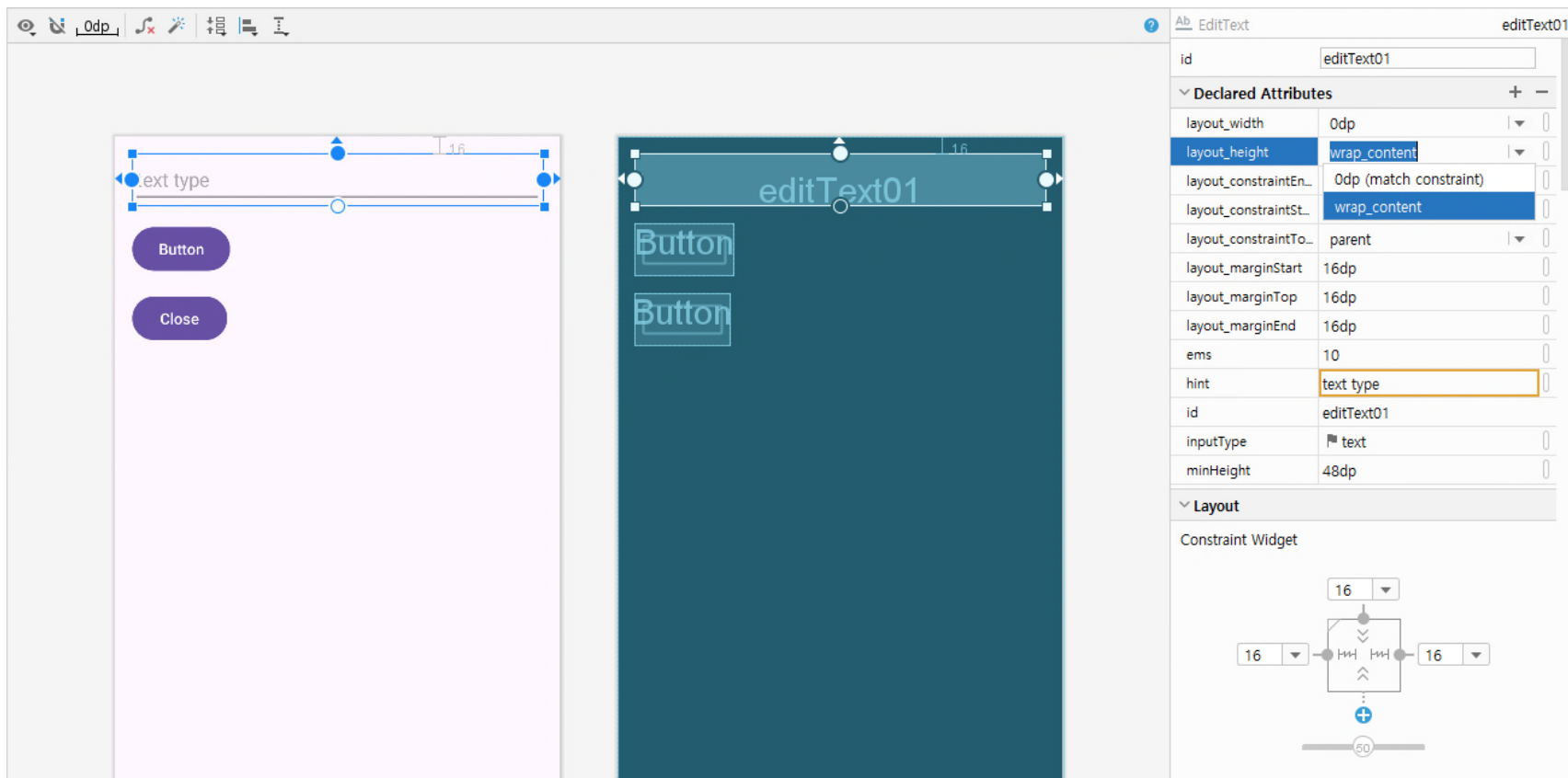
□ 해당 위젯을 선택한 후, 속성을 정의



## □ 레이아웃 설정 : Design

### ○ 각 구성 요소에 대한 속성 값 설정

- 텍스트를 입력할 Plain Text의 너비를 최대로 설정 (match\_constraint) : 0dp
- 버튼(Button)의 폭은 필요한 만큼으로 설정 (wrap\_content)
- 높이는 모두 필요한 만큼으로 설정 (wrap\_content)



## □ 레이아웃 설정 : XML

### ○ 코드로 이동

- 해당 위젯을 선택하고 Go To XML 선택
- 여러 동일 위젯(예, Button)이 존재할 때 쉽게 코드로 이동

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

해당 위젯에 대한 코드 부분이 선택되어 이동

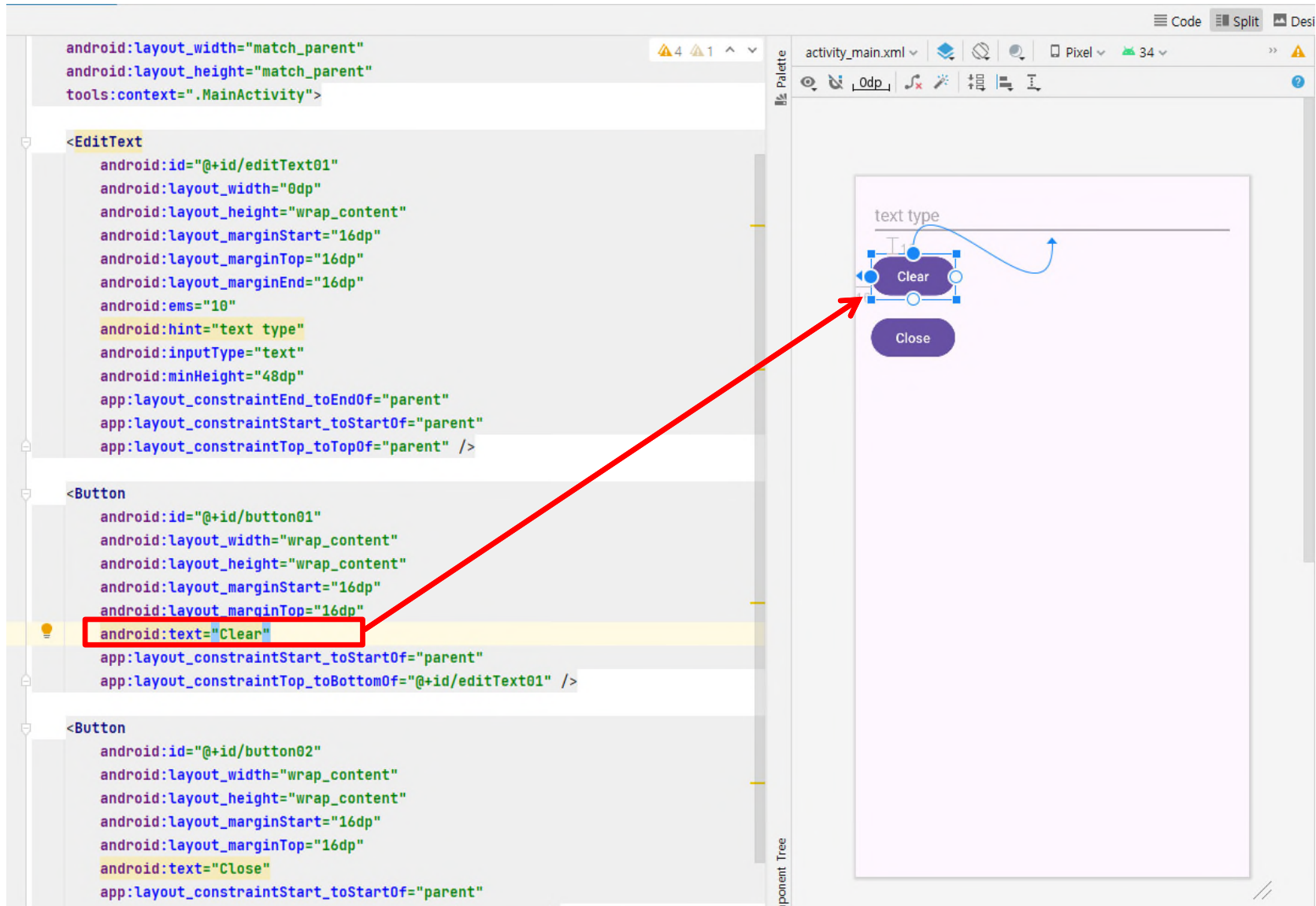
```
<EditText
    android:id="@+id/editText01"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:ems="10"
    android:hint="text type"
    android:inputType="text"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

- PlainText를 선택하여 이동한 예
- (선택한 위젯에 대한 코드로 이동하면서 해당 부분이 선택되어 쉽게 확인할 수 있도록 제공 (여러 동일한 위젯을 사용할 때 유용)

```
<Button
    android:id="@+id/button01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

## □ 레이아웃 설정 : XML

### ○코드를 통한 수정/작성





## □ 레이아웃 작성 결과

### ○ 레이아웃 편집 결과 확인

### ○ 각 객체에 주어진 id 확인

- 화면 구성을 하는 위젯을 구분

### ○ 편집된 xml 코드의 화면 반영 결과 확인

### ○ ConstraintLayout의 위젯 위치 속성 확인

### ○ 전체 layout 코드

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

```
<EditText
```

```
    android:id="@+id/editText01"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
```

```
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:ems="10"
    android:hint="text type"
    android:inputType="text"
    android:minHeight="48dp"
```

```
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

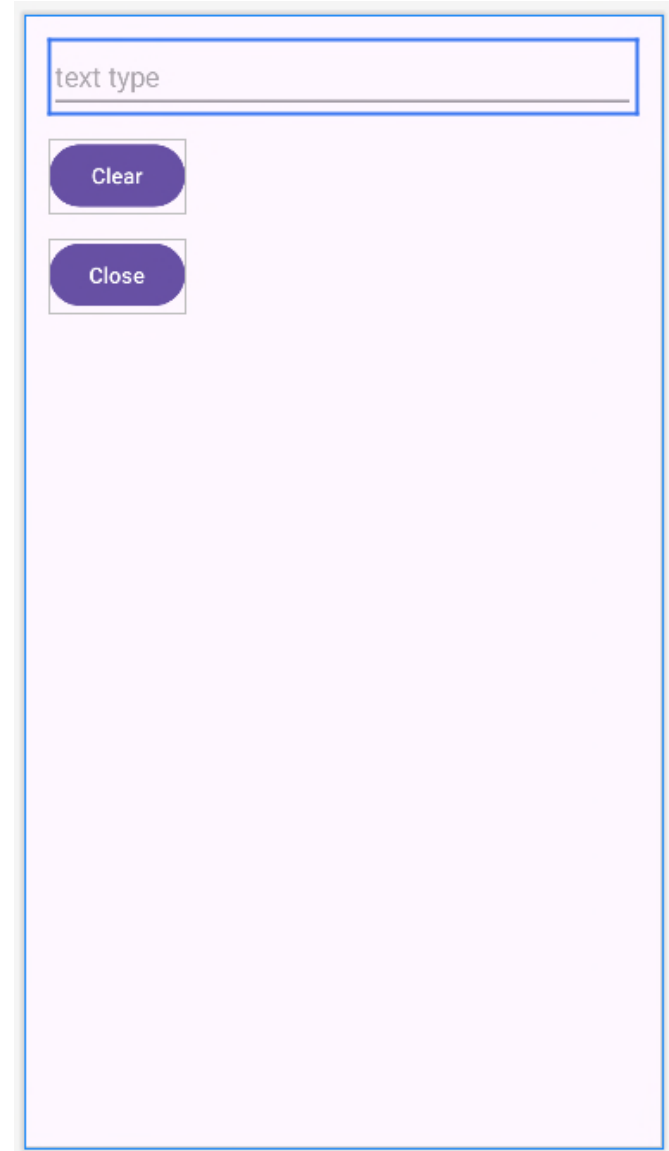


## □ 레이아웃 작성 결과

```
<Button
    android:id="@+id/button01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:text="Clear"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editText01" />

<Button
    android:id="@+id/button02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:text="Close"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button01" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



## □ 코드 분석 / 설명 - MainActivity

```
package com.practice.ex.androidlayout;
```

```
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.widget.Button;
```

```
import android.widget.EditText;
```

필요한 클래스 import  
- 자동으로 import 됨

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
```

2 usages

```
EditText editText01;
```

2 usages

```
Button btn01, btn02;
```

클릭 이벤트 처리를 위한 리스너 정의

레이아웃 객체와 연결한 변수 선언

## □ 코드 분석 / 설명 - MainActivity

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
}
```

기능/실습 1 설명 참조

```
editText01 = (EditText) findViewById(R.id.editText01);  
btn01 = (Button) findViewById(R.id.button01);  
btn02 = (Button) findViewById(R.id.button02);
```

레이아웃 객체와 연결

```
btn01.setOnClickListener(this);  
btn02.setOnClickListener(this);
```

클릭리스너에 등록

- 등록된 구성요소에 대해서만 클릭발생시  
이벤트가 클릭리스너에 통지됨

@Override

```
public void onClick(View v) {
```

버튼이 클릭되었을 때 호출되는 콜백메소드

```
    if(v.getId() == R.id.button01) {  
        editText01.setText("");  
    }  
    if(v.getId() == R.id.button02) {  
        finish();  
    }  
}
```

이벤트가 발생했을 때, 처리되어야 할 내용을 작성

- setText() : 텍스트 설정  
- finish() : 액티비티 종료

## **□ 안드로이드 기능/실습 3**

### **- 액티비티 추가/실행**

## □ 액티비티

### ○ 목적

- 인텐트를 이용한 액티비티 실행 과정 이해
  - ❖ 액티비티 추가 작성
  - ❖ 인텐트를 이용한 액티비티 실행 방법 습득

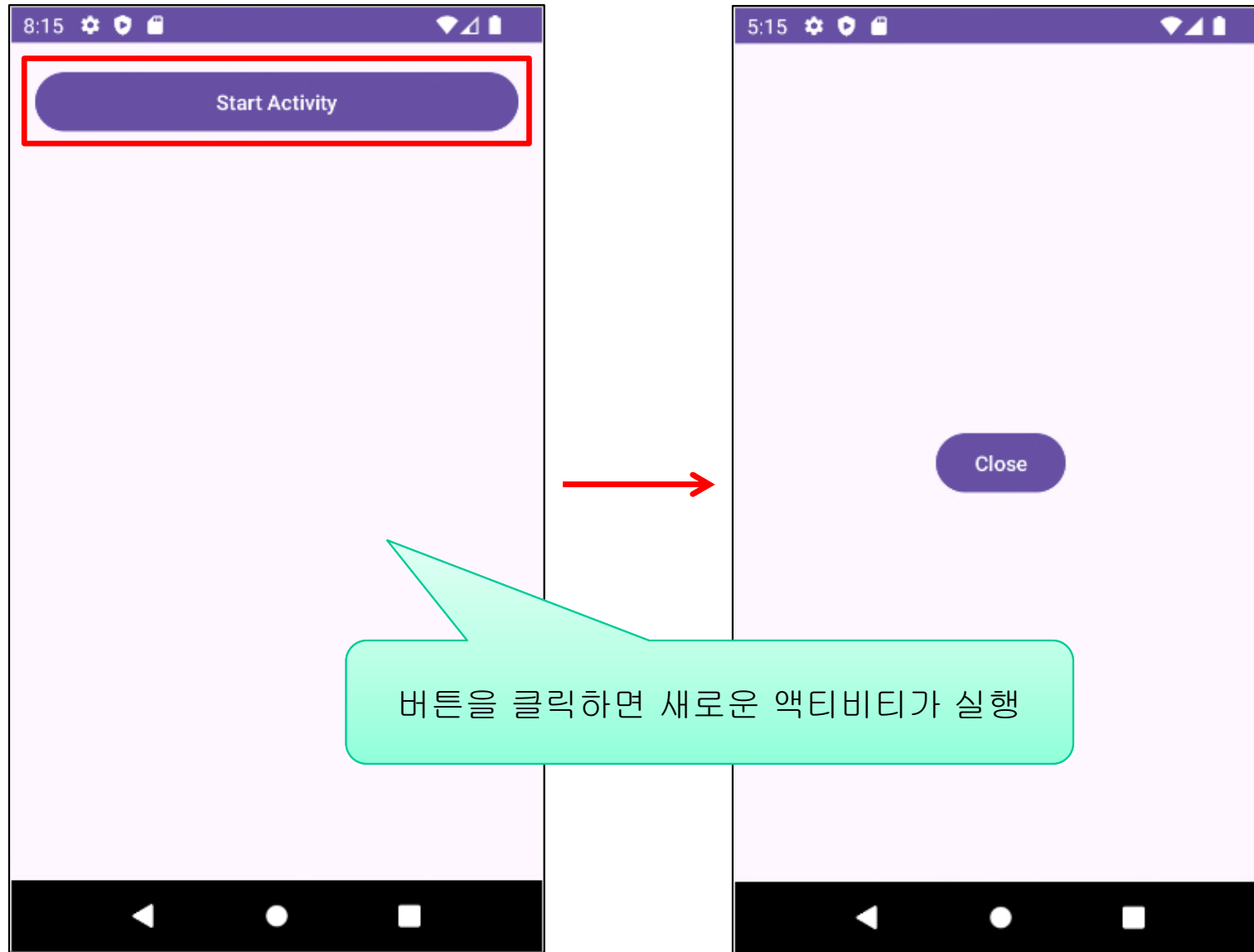
### ○ 기능/실습 3 내용

- 액티비티 추가 작성 과정과 인텐트를 이용한 액티비티 실행 방법을 습득하고 코드 처리 과정 진행

### ○ 액티비티

- 현재 액티비티에서 다른 액티비티 화면을 오픈
  - ❖ `startActivity(Intent);`

## □ 결과 실행 모습

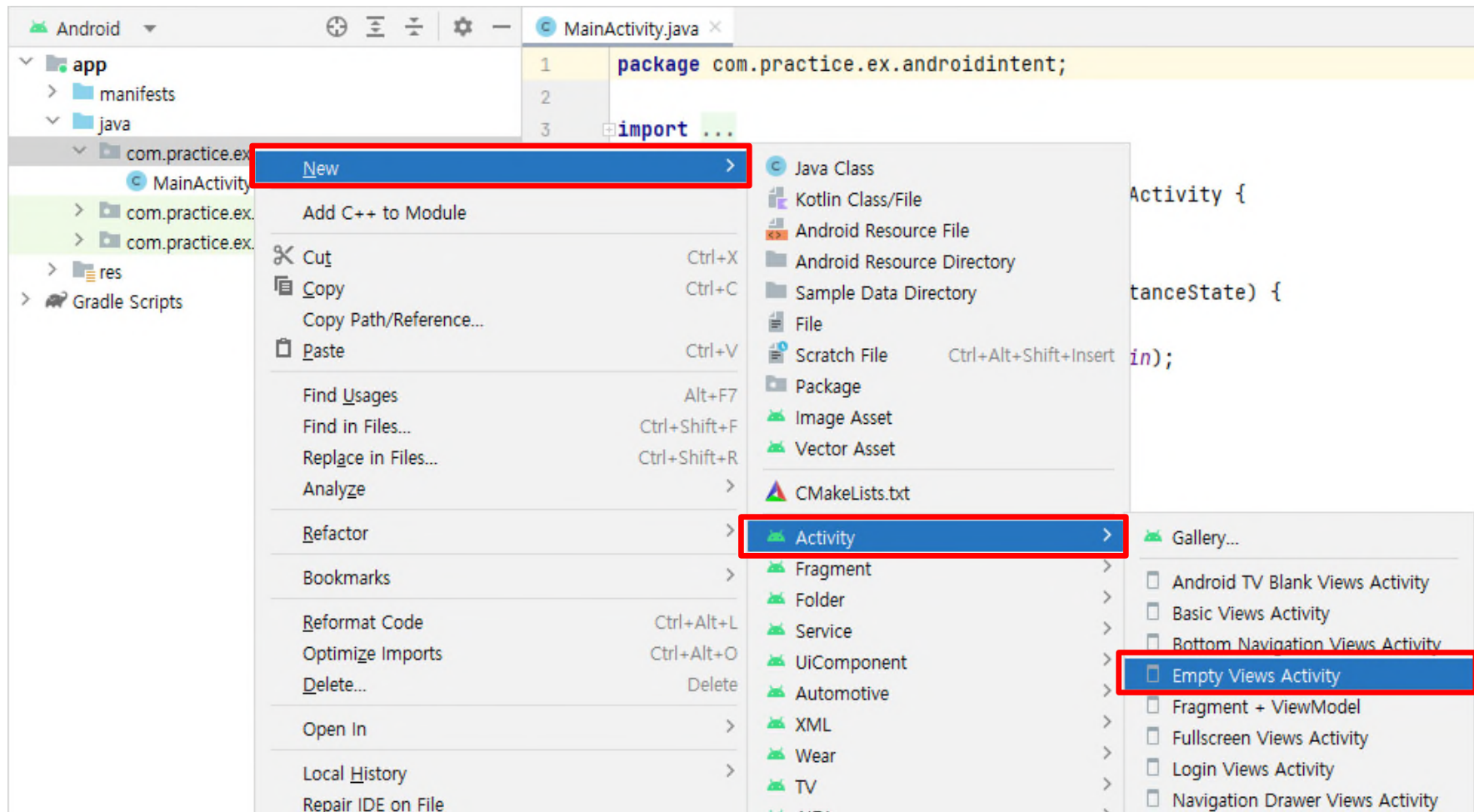


## □ 진행 단계

- 액티비티 준비
  - MainActivity
  - SecondActivity
- 레이아웃 준비
  - activity\_main.xml
  - activity\_second.xml
- 레이아웃 작성
- 액티비티 코드 작성

## □ 액티비티 추가 과정

- 코드에서 우클릭 한 후,
- new → Activity → EmptyViewActivity 선택
- Java Class 선택 X





## □ 액티비티 추가 과정

### ○ 추가할 액티비티 정보 입력

New Android Activity

Empty Views Activity  
Creates a new empty activity

Activity Name  
SecondActivity

☒ Generate a Layout File

Layout Name  
activity\_second

☐ Launcher Activity

Package name  
com.practice.ex.androidintent

Source Language  
Java

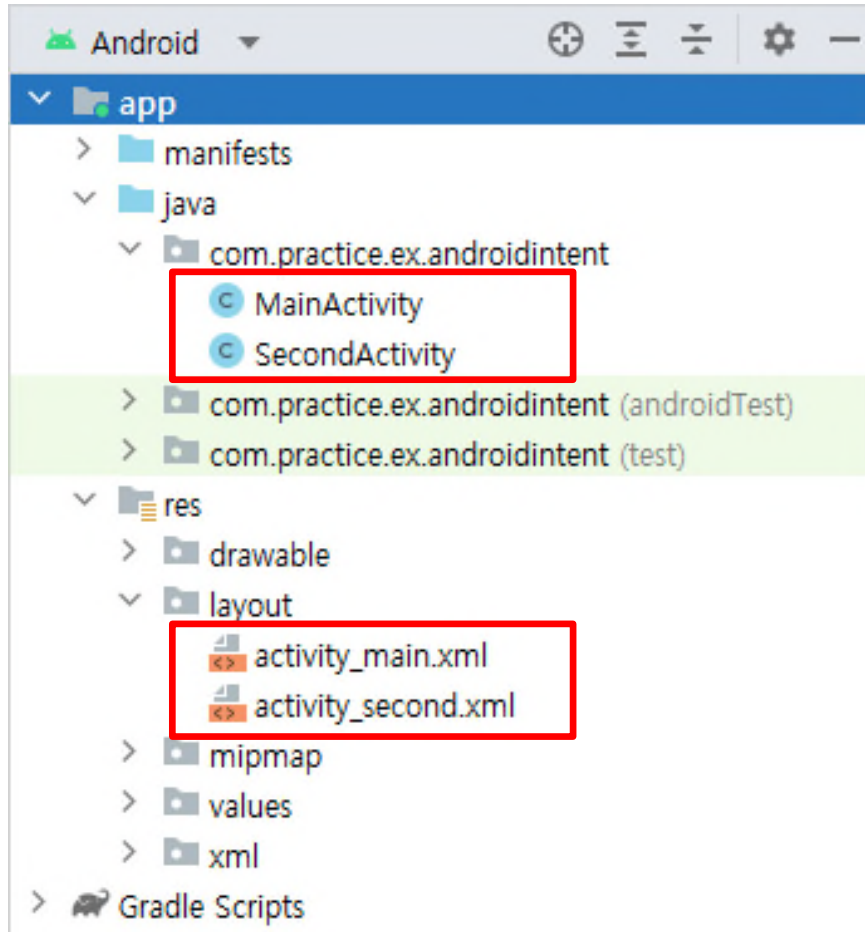
Previous Next Cancel Finish

Activity Name은  
-SecondActivity

Layout Name은  
-activity\_second

-이름은 자유롭게 작성

## □ 액티비티/레이아웃 생성 결과



### ○ 액티비티 구성

- MainActivity
- SecondActivity

### ○ 레이아웃 구성

- activity\_main.xml
- activity\_second.xml

## □ 수정된 *AndroidManifest.xml*

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="AndroidIntent"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AndroidIntent"
        tools:targetApi="31">

        <activity
            android:name=".SecondActivity"
            android:exported="false" />

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

추가한 액티비티 정보가  
자동으로 추가됨

안드로이드 스튜디오 버전에 따라 코드의 모습이 차이가 있음

- 액티비티를 호출하기 위해서는 매니페스트 파일에 액티비티가 등록되어 있어야 함.
- 액티비티 추가 과정을 정상적으로 진행한 경우에는 자동으로 추가가 되어 있음

## □ 레이아웃 작성 : *activity\_main.xml*

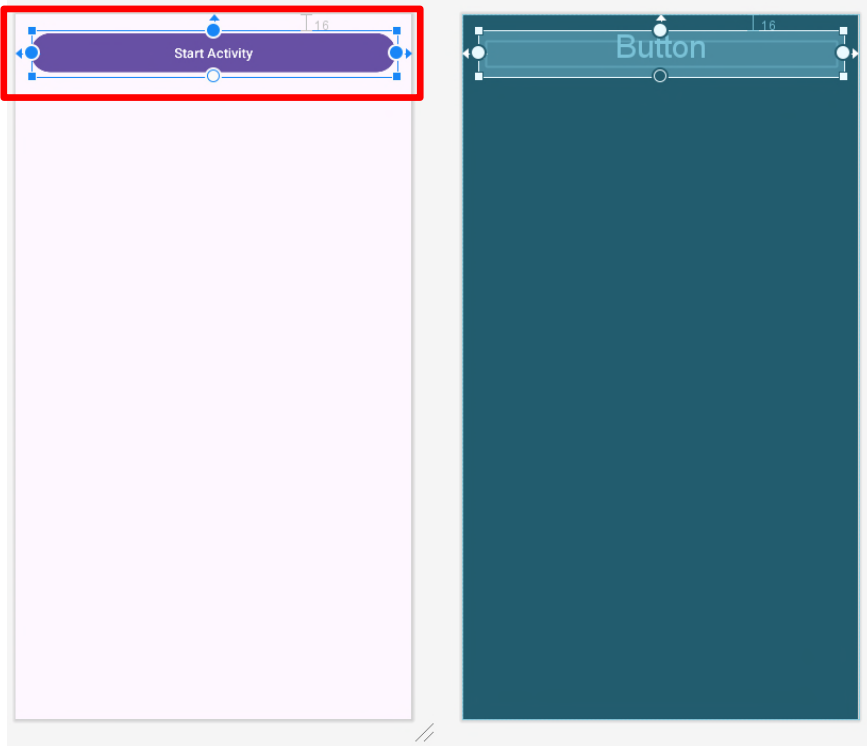
### ○ Layout 작성

#### □ 고려사항

- ❖ 액티비티 별로 레이아웃을 작성

#### □ *activity\_main.xml*

- ❖ 새로운 액티비티 실행을 위한 버튼 필요



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

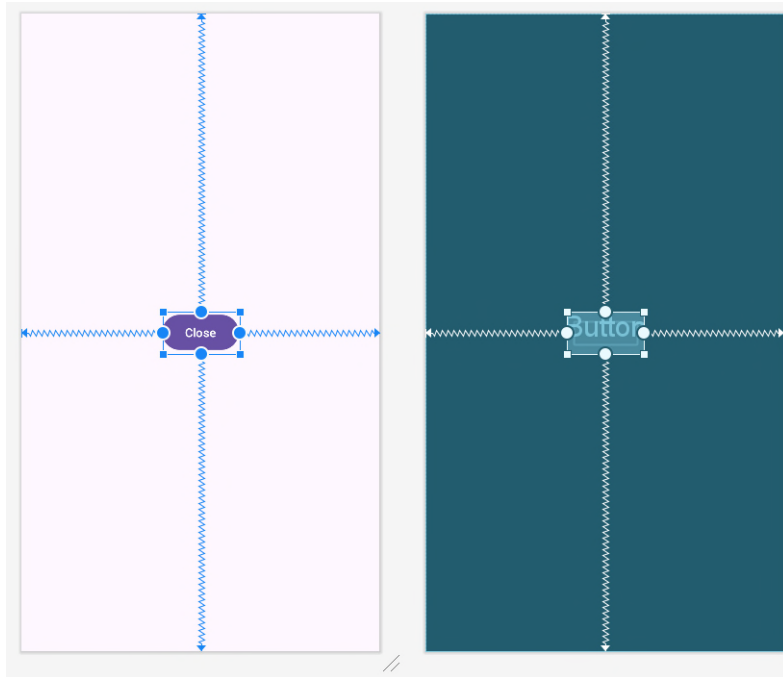
```
<Button
    android:id="@+id/button01"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:text="Start Activity"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

## □ 레이아웃 : *activity\_second.xml*

### ○ SecondActivity의 레이아웃인 *activity\_second.xml* 작성

#### □ 액티비티를 종료하기 위한 버튼만 추가



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SecondActivity">
```

```
    <Button
        android:id="@+id/button02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Close"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```



## □ 코드 분석 / 설명 - MainActivity

```
package com.practice.ex.androidintent;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {
```

2 usages

```
Button btn01;
```

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

    btn01 = (Button) findViewById(R.id.button01);
    btn01.setOnClickListener(this);
}
```

인텐트를 이용하여 다른 액티비티를  
실행시키기 위함  
startActivity(...) : 액티비티 실행

```
@Override
public void onClick(View v) {
    Intent intent01 = new Intent( packageContext: MainActivity.this, SecondActivity.class);
    startActivity(intent01);
}
```

## □ 코드 분석 / 설명 - *SecondActivity*

```
package com.practice.ex.androidintent;
```

```
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.widget.Button;
```

```
public class SecondActivity extends AppCompatActivity implements View.OnClickListener {
```

2 usages

```
Button btn01;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_second);
```

레이아웃 연결

```
    btn01 = (Button) findViewById(R.id.button02);
```

```
    btn01.setOnClickListener(this);
```

```
@Override
```

```
public void onClick(View v) {
```

```
    finish();
```

```
}
```

리스너에 등록된 대상이 1개이기 때문에  
대상 구분이 필요 없음