

안드로이드 애플리케이션 프로그래밍

- View 기초

View, Widget



부산대학교 정보·의생명공학대학
정보컴퓨터공학부



이론

강의 목표와 구성

❖ View

- View 와 실행화면
- View 의 종류와 배치
- View 의 속성과 크기
- View 의 영역
- View 의 다중 배치와 시각표현

❖ Widget

- Widget 배치하기
- Widget 연결하기
- 알고리즘 연결하기

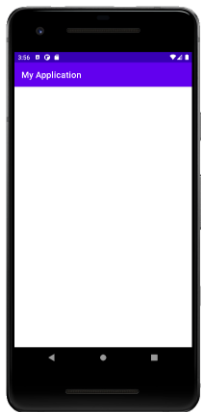
View와 실행화면

❖ 실행화면

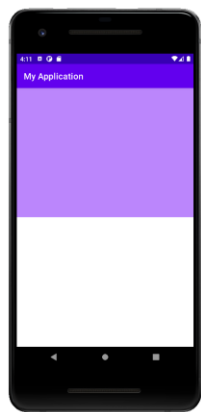
- 앱을 실행하면, 화면에 무엇이 나와야 사용자가 인식하고 사용할 수 있음
- 실행 화면은 **View들을 구성하고 배치한 결과물**임

❖ View란?

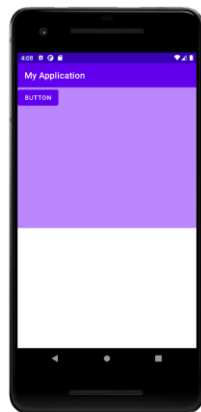
- 앱의 **실행화면**을 구성하는 요소



프로젝트 만들기



공간 만들기



배치(구성)하기

```
<LinearLayout
    android:layout_weight="1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/purple_200">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"/>
</LinearLayout>
```



View의 종류와 배치

❖ View의 종류

Button

BUTTON

Layout

▪ Widget

- View(View) 중에 **시각적인 요소**
- **ImageView, TextView, Button, ...**

▪ ViewGroup

- View(View) 중에 **영역적인 요소**
- **Layout, container**

❖ View의 요소 배치

- View의 요소 배치는 구조를 표현하는 언어인 **XML로 수행**됨
- 모든 View 요소는 **태그**라는 형태로 명시됨
 - <시작 태그>~</끝 태그> 혹은 <약식 태그/> 표현됨
- 모든 View 요소는 **Layout 태그** 내에 포함됨
 - 최상단에 존재하는 태그는 **선언 태그**와 **Layout 태그**임

선언 태그

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

시작 태그

```
<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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:weightSum="2"
    tools:context=".MainActivity">
```

약식 태그

```
<LinearLayout
    android:layout_weight="1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/purple_200">
```

끝 태그

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

```
</LinearLayout>
```

```
</LinearLayout>
```

View의 속성과 크기

❖ View의 속성(Attribute)

- View 요소에 부여하는 **특징 및 참조 정보**
- View 요소의 속성 중 **크기 속성은 필수 정보**
 - 속성 = “속성값”

❖ View의 크기

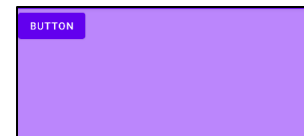
- View는 화면에 배치되는 요소로서, **공간의 크기에 대한 속성은 필수적으로 명시되어야 함**
- **공간의 크기**는 **참조하는 방식과 값을 지정해주는 방식**이 있음
 - **match_parent**: 상위 요소(부모 요소)의 공간 범위를 참조함
 - **wrap_content**: 본인 및 하위 요소의 텍스트 문자 크기를 참조함
 - **지정 값**: View 요소의 크기를 지정함
 - “범위 값 + 범위 단위”
 - **Widget 최적 단위**: dp
 - **텍스트 최적 단위**: sp

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



속성

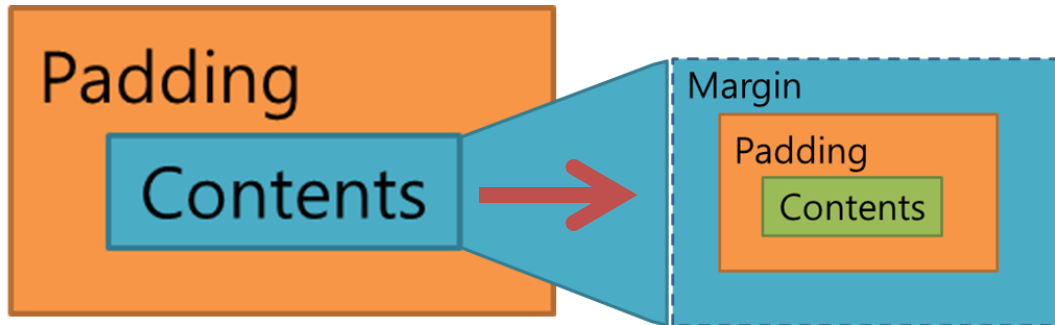
```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android=  
    "http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    android:weightSum="2"  
    tools:context=".MainActivity">  
    <LinearLayout  
        android:layout_weight="1"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:background="@color/purple_200">  
        <Button  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="Button"/>  
        </LinearLayout>  
    </LinearLayout>
```



View의 영역

❖ View의 영역

- 내용(Contents), 패딩(Padding), 마진(Margin)
- View의 간격은 margin과 padding 속성으로 설정함



❖ 패딩(Padding)과 마진(Margin)

- 패딩 속성은 Widget의 경계선으로부터 **Widget 내부의 요소가 멀어지게** 여백을 설정할 수 있음
- 마진 속성은 Widget의 경계선 밖으로 **부모 태그로부터 Widget이 멀어지게** 여백을 설정할 수 있음

```
<LinearLayout
    ...
    android:padding="30dp">
    <TextView
        ...
        android:background="#89EEFF"
        android:text="부모 태그의 패딩 30dp"/>
    <TextView
        ...
        android:layout_margin="30dp"
        android:background="#00FF00"
        android:text="태그의 마진 30dp"/>
</LinearLayout>
```

Linear Layout

Textview



View의 다중 배치와 시각 표현

❖ View의 다중 배치

- 하나의 Layout 안에 여러 개의 View 요소 배치가 가능함
- 다중 배치를 위해서는 적절한 시각 표현이 필요함

❖ View 의 시각 표현

- 시각 표현 관련 속성을 통해서 태그를 화면에 다양하게 표현할 수 있다.
- 색상 및 회전을 통해서 View 요소를 강조하는 표현할 수 있다.
- View 요소를 숨기거나, 드러낼 수 있다.
- View 요소의 상호작용을 비 활성화 하거나, 활성화 할 수 있다.

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:weightSum="2"
    tools:context=".MainActivity">
```

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="30dp">
```

```
        <Button...>
        <Button...>
        <Button...>
        <Button...>
        <Button...>
        <Button...>
        <Button...>
        <Button...>
```

```
    </LinearLayout>
```

```
</LinearLayout>
```

기본 버튼

VISIBILITY-VISIBLE

ENABLED-TRUE

ENABLED-FALSE

CLICKABLE-TRUE

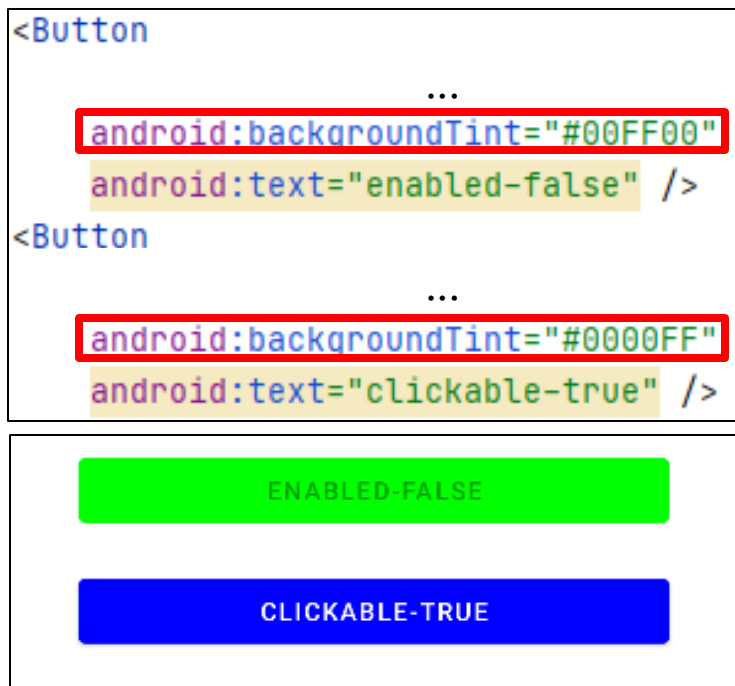
CLICKABLE-FALSE

ROTATION

View의 시각 표현 - 색상과 회전

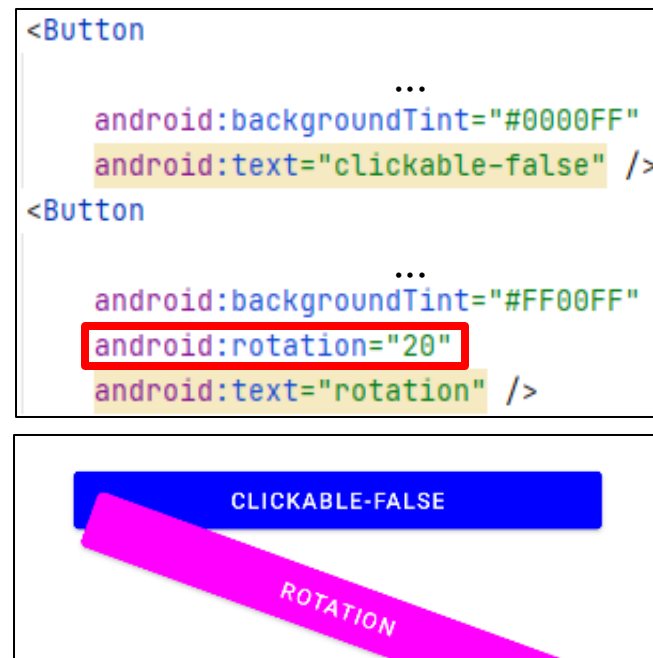
❖ View의 색상 속성

- View 요소에 색상을 결정
 - backgroundTint="RGB 색상 코드"



❖ View의 회전 속성

- View 요소에 회전 각도를 결정
 - rotation="각도"



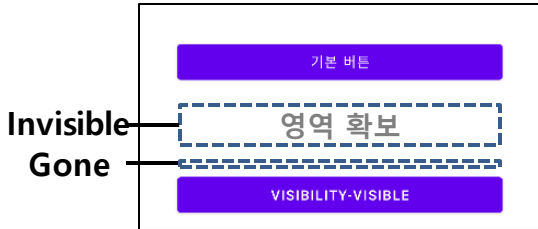
❖ View의 색상과 회전

- View의 색상과 회전은 제공하는 Widget을 강조하여 사용자의 이목을 끌 수 있다.
- Layout을 포함한 모든 View 요소 동일하게 적용이 가능하다.

View의 시각 표현 – 가시성과 활성화

❖ View의 가시성

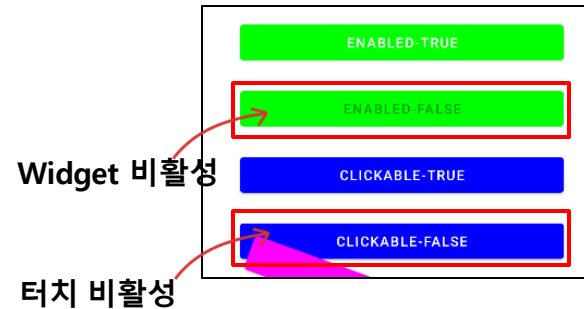
- View 요소가 보일 것인지 여부를 결정
 - 숨김 : "invisible"-영역 유지, "gone"-영역 비 유지



```
<Button
    ...
    android:text="기본 버튼" />
<Button
    ...
    android:visibility="invisible"
    android:text="visibility-invisible" />
<Button
    ...
    android:visibility="gone"
    android:text="visibility-gone" />
<Button
    ...
    android:visibility="visible"
    android:text="visibility-visible" />
```

❖ View의 활성화

- View 요소의 활성화 여부를 결정
 - enabled** : Widget 동작 활성화
 - Clickable** : 상호작용(클릭, 터치) 활성화



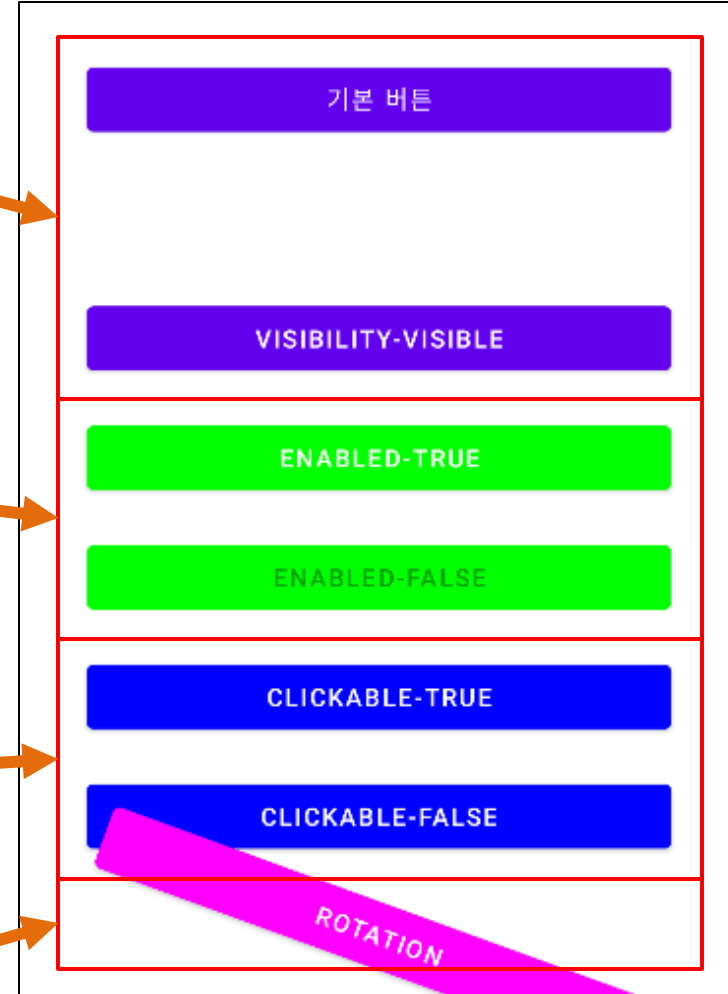
```
<Button
    ...
    android:enabled="true"
    android:text="enabled-true" />
<Button
    ...
    android:enabled="false"
    android:text="enabled-false" />
<Button
    ...
    android:clickable="true"
    android:text="clickable-true" />
<Button
    ...
    android:clickable="false"
    android:text="clickable-false" />
```

❖ View의 가시성과 활성화

- View 요소의 가시성과 활성화는 **이벤트 처리 측면에서 중요한 역할**을 수행한다.
- XML 코드에서 사용되기보다, 주로 **Kotlin 코드와 연계되어서 많이 사용되는 속성**이다.

View의 시각 표현 - 소스 코드

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout ...
  <LinearLayout ...
    <Button ...
      android:text="기본 버튼" />
    <Button ...
      android:visibility="invisible"
      android:text="visibility-invisible" />
    <Button ...
      android:visibility="gone"
      android:text="visibility-gone" />
    <Button ...
      android:visibility="visible"
      android:text="visibility-visible" />
    <Button ...
      android:backgroundTint="#00FF00"
      android:enabled="true"
      android:text="enabled-true" />
    <Button ...
      android:backgroundTint="#00FF00"
      android:enabled="false"
      android:text="enabled-false" />
    <Button ...
      android:backgroundTint="#0000FF"
      android:clickable="true"
      android:text="clickable-true" />
    <Button ...
      android:backgroundTint="#0000FF"
      android:clickable="false"
      android:text="clickable-false" />
    <Button ...
      android:backgroundTint="#FF00FF"
      android:rotation="20"
      android:text="rotation" />
  </LinearLayout>
</LinearLayout>
```



Widget 다루기

❖ Widget의 역할

- Widget은 구성된 **알고리즘을 시작하는 트리거 역할**을 수행함
- 알고리즘의 연산 결과**를 Widget으로 표현함

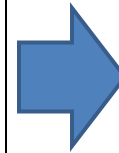
```
<Button  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="10dp"  
    android:id="@+id/BtnAdd"  
    android:text="더하기" />
```

Widget 배치(XML)



```
lateinit var btnAdd : Button;   lateinit var btnSub : Button  
lateinit var btnMul : Button;  lateinit var btnDiv : Button  
lateinit var textResult : TextView  
lateinit var num1 : String;    lateinit var num2 : String  
var result : Int? = null  
  
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    title = "초간단 계산기"  
  
    edit1 = findViewById<EditText>(R.id.Edit1)  
    edit2 = findViewById<EditText>(R.id.Edit2)  
  
    btnAdd = findViewById<Button>(R.id.BtnAdd)
```

Widget 연결(Kotlin)



```
btnAdd.setOnTouchListener{ view, motionEvent ->  
    num1 = edit1.text.toString()  
    num2 = edit2.text.toString()  
    result = Integer.parseInt(num1) + Integer.parseInt(num2)  
    textResult.text = "계산 결과 : "+result.toString()  
    false ^setOnTouchListener  
}
```

알고리즘 구성(Kotlin)

숫자1

숫자2

더하기

빼기

곱하기

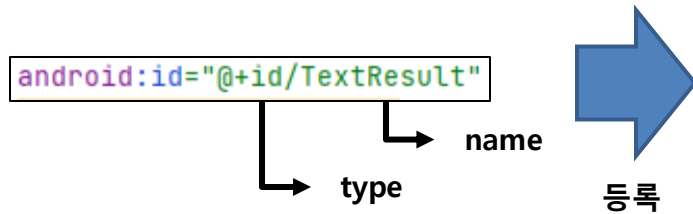
나누기

계산 결과 :

Widget 배치하기

❖ Widget의 연결 속성

- 개발 단계에서 자동 완성 지원
 - `tools:context=".클래스명"`
- 알고리즘과 태그 사이의 연결 / id 리스트 등록
 - `android:id="@+id/고유식별자"`



```
<resources>
  <item type="id" name="Edit1"/>
  ...
  <item type="id" name="BtnAdd"/>
  ...
  <item type="id" name="BtnTextResultt"/>
</resources>
```

❖ Widget 배치하기

- Widget은 화면에 배치되며, 터치를 통해서 사용자와의 상호작용으로 알고리즘 수행이 가능함
- 알고리즘 수행을 위해서 구성된 알고리즘을 Widget에 연결해야 함
- 연결을 위해서, XML 와 Kotlin의 각 코드에 연결을 위한 부분 추가가 필요함

tools

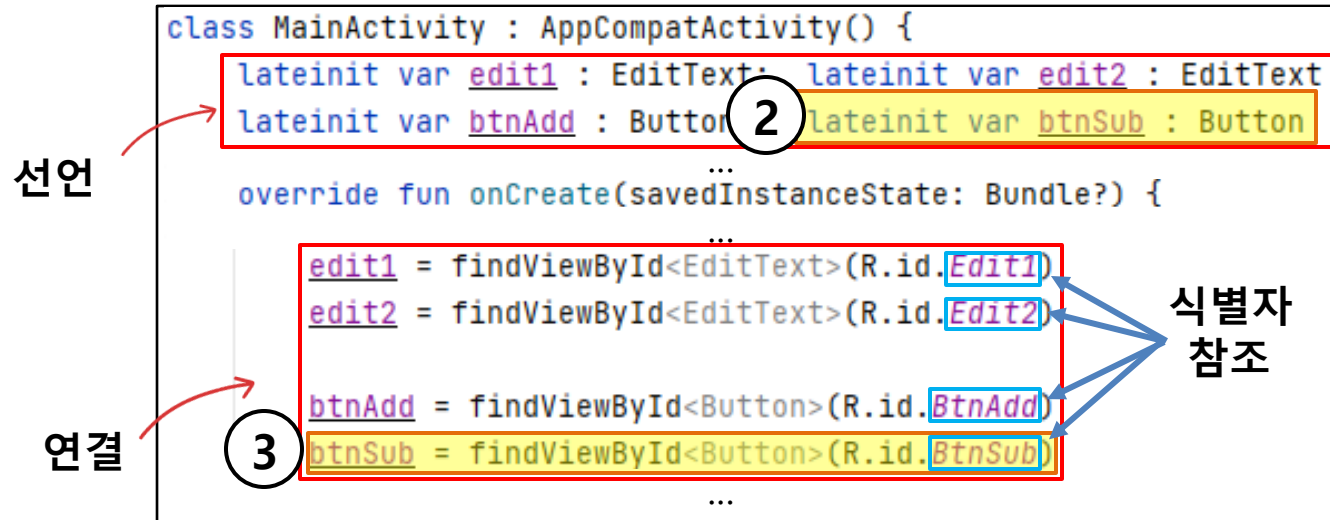
id

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    tools:context=".MainActivity">
    <LinearLayout
        <EditText
            android:id="@+id/Edit1"
            android:hint="숫자1" />
        <EditText android:layout_width="wrap_content"
            <Button android:layout_width="match_parent"
                android:id="@+id/BtnAdd"
                android:text="더하기" />
        <Button android:layout_width="match_parent"
            <Button android:layout_width="match_parent"
        <TextView android:layout_width="wrap_content"
            android:id="@+id/TextResultt"
            android:textSize="30dp"
            android:textColor="#FF0000"
            android:text="계산 결과 : " />
    </LinearLayout>
</LinearLayout>
```

Widget 연결하기

❖ Widget 객체

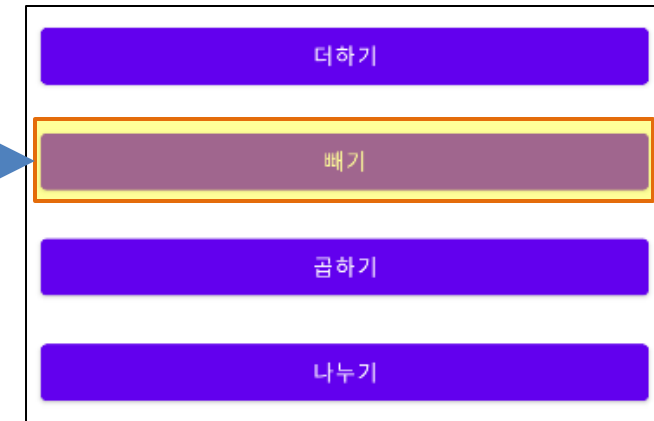
- 태그를 연결하기 위한 객체 생성
- 태그의 정보에 접근하기 위한 findViewById() 메소드 호출



❖ Widget 연결하기

- id 속성 값을 사용해서 XML 태그와 Kotlin 객체를

3 Var BtnSub



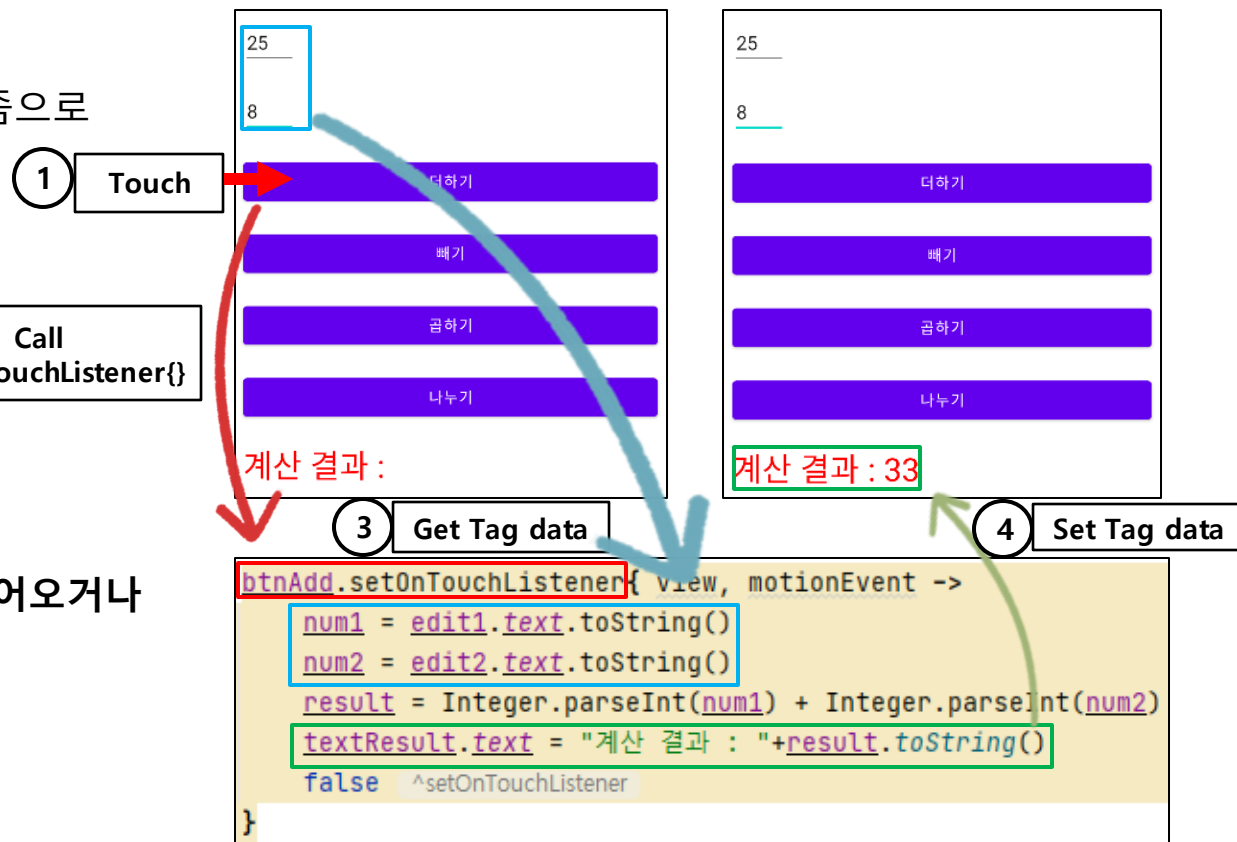
알고리즘 실행하기

❖ 알고리즘 실행

- 애플리케이션에서 처리하고 동작하는 과정은 모두 알고리즘으로 수행됨

❖ 알고리즘

- Widget의 백그라운드 동작 역할을 수행함
- 주로, 화면 내의 Widget 위치에 대한 상호작용을 인식함
- 알고리즘의 수행 결과를 Widget을 통해서 화면에 반영함
- 태그와 연결된 객체는 태그에 접근하여 태그의 데이터를 읽어오거나 변경할 수 있음



실습

실습 목표와 구성

1. 기초(따라하기)

- View 의 배치
- View 의 영역
- View 의 시각표현

2. 응용(로직구현)

- Widget 의 알고리즘 구성하기

3. 심화(완성하기)

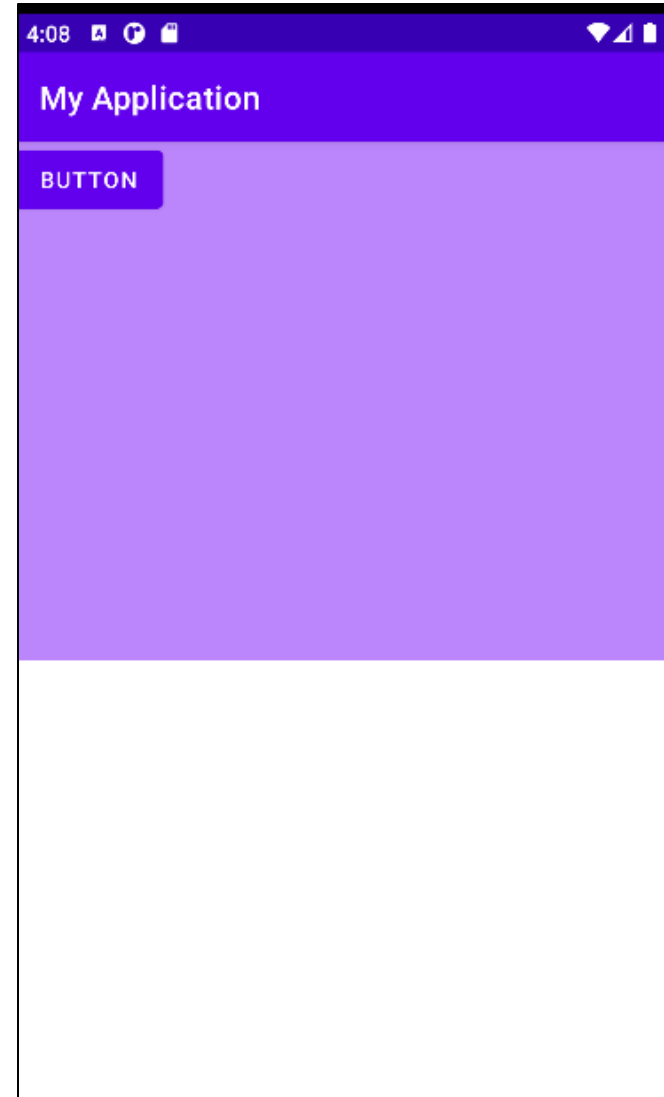
- 바탕화면 색 변경하기
- 버튼 릴레이
- 갤러리 만들기

4. 심화(과제물)

기초(따라하기) – 예제 1

❖ View 의 배치

- View 요소인 Layout 과 Widget 을 배치할 수 있다.
1. 메인 Layout 배치하기
 2. 내부 Layout 배치하기
 3. 버튼 Widget 배치하기



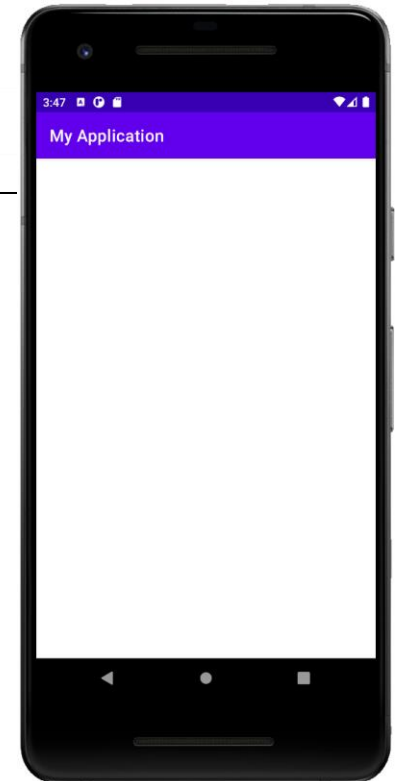
기초(따라하기) – 예제 1

activity_main.xml

1. 메인 Layout 배치하기

- (1 line) : xml 선언 태그 및
 - (2 line) : LinearLayout 태그 시작
 - (2-5 line) : XML Name Space 연결
 - (6-7 line) : Layout의 너비와 높이 부모 Layout을 참조
 - (8 line) : Layout의 내부 요소 나열 방향 설정
 - (9 line) : Layout의 내부 균등 분배 비율 설정
 - (10 line) : Kotlin 소스코드 연결, LinearLayout 태그의 끝
-
- 참조 사항
 - `<?xml ?>` : XML 문서의 시작을 알리며, 내부 인코딩 방식을 명시함
 - `<LinearLayout> </LinearLayout>` : LinearLayout 배치
 - `xmlns` 속성 : XML Name Space 연결
 - `layout_width, layout_height` 속성 : Layout의 너비와 높이 설정
 - `Match_parent` : 부모 크기 참조
 - `orientation` 속성 : Layout의 내부 요소 나열 방향 설정
 - `weightSum` 속성 : Layout의 크기를 균등한 비율로 나눔
 - `Tools` : 개발 시 이름 참조를 위한 연결

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android=
3     "http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     android:orientation="vertical"
9     android:weightSum="2"
10    tools:context=".MainActivity">
11
12 </LinearLayout>
```



기초(따라하기) – 예제 1

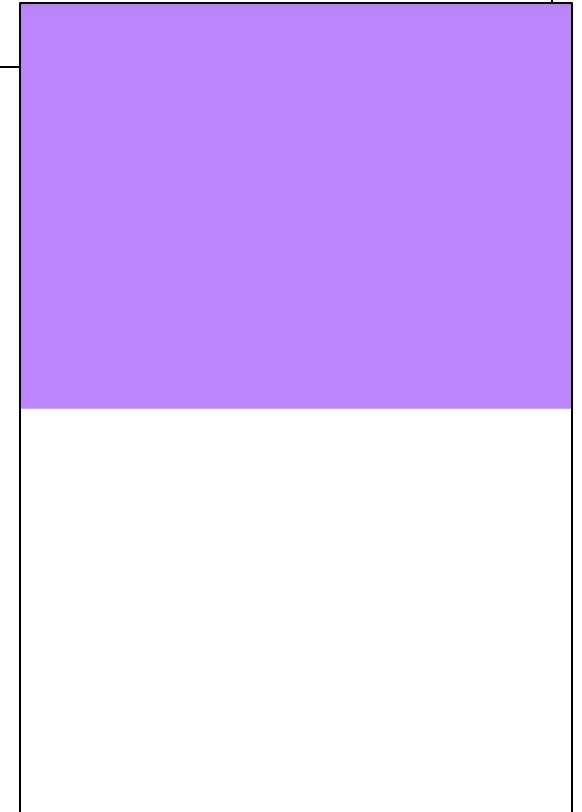
2. 내부 Layout 배치하기

- (12 line) : Layout 크기의 고정 비율 설정
- (13-14 line) : Layout의 크기 설정
- (15 line) : Layout 배경 색상 설정

activity_main.xml

```
11 <LinearLayout
12     android:layout_weight="1"
13     android:layout_width="match_parent"
14     android:layout_height="wrap_content"
15     android:background="@color/purple_200">
16
17 </LinearLayout>
```

- 참고사항
 - **Layout_weight 속성** : 부모 Layout의 크기 중, 해당 Layout이 차지할 크기 비율
 - **Background 속성** : Layout의 배경색상을 보라색으로 설정



기초(따라하기) – 예제 1

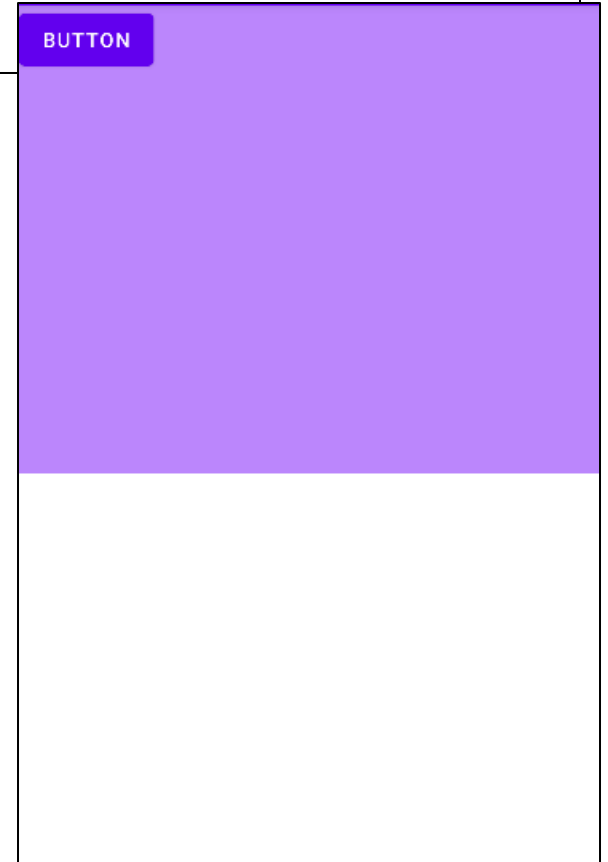
activity_main.xml

3. 버튼 Widget 배치하기

- (16 line) : 버튼 태그 시작
- (17-18 line) : 버튼의 크기 설정
- (19 line) : 버튼의 텍스트 설정

```
16      <Button
17          android:layout_width="wrap_content"
18          android:layout_height="wrap_content"
19          android:text="Button"/>
20    </LinearLayout>
21 </LinearLayout>
```

- 참고사항
 - **<Button />** : 버튼 Widget 태그, 화면을 통한 상호작용 제공
 - **text** : 버튼 내에 배치되는 문자열

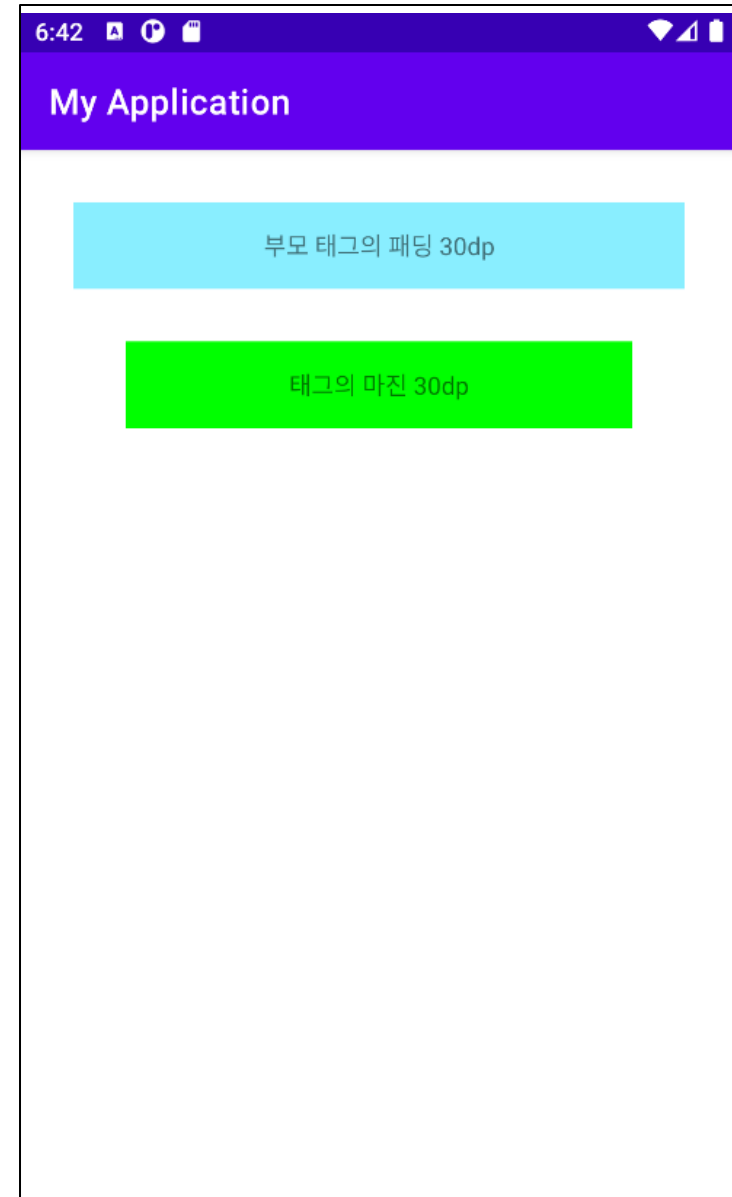


기초(따라하기) – 예제 2

❖ View 요소의 영역

- 부모 태그로부터 설정된 **패딩의 영역**을 파악할 수 있다.
- 태그 내부에 설정된 **마진의 영역**을 파악할 수 있다.

1. 패딩 Layout 배치하기
2. 기본 텍스트View 배치하기
3. 마진 텍스트View 배치하기



기초(따라하기) – 예제 2

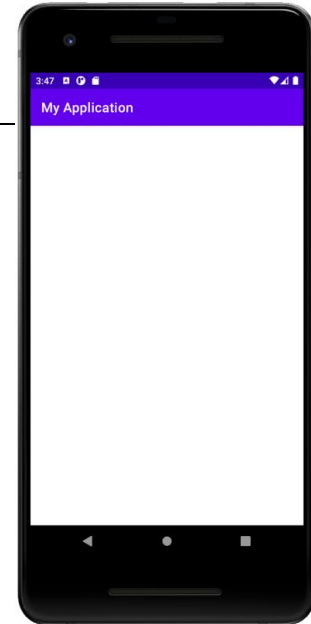
activity_main.xml

1. 패딩 레이어 배치하기

- (1-9 line) : (생략) 메인 Layout
- (11-12 line) : Layout 크기 설정
- (13 line) : Layout 내부 배치 방식 설정
- (14 line) : Layout 패딩 설정

```
10 <LinearLayout
11     android:layout_width="match_parent"
12     android:layout_height="wrap_content"
13     android:orientation="vertical"
14     android:padding="30dp">
15
16 </LinearLayout>
```

- 참고사항
 - **orientation 속성** : 내부 요소의 배치 방식
 - “vertical” : 세로 배치
 - **Padding 속성** : 본 태그의 윤곽선과 내부 요소 간의 여백 영역 확보



기초(따라하기) – 예제 2

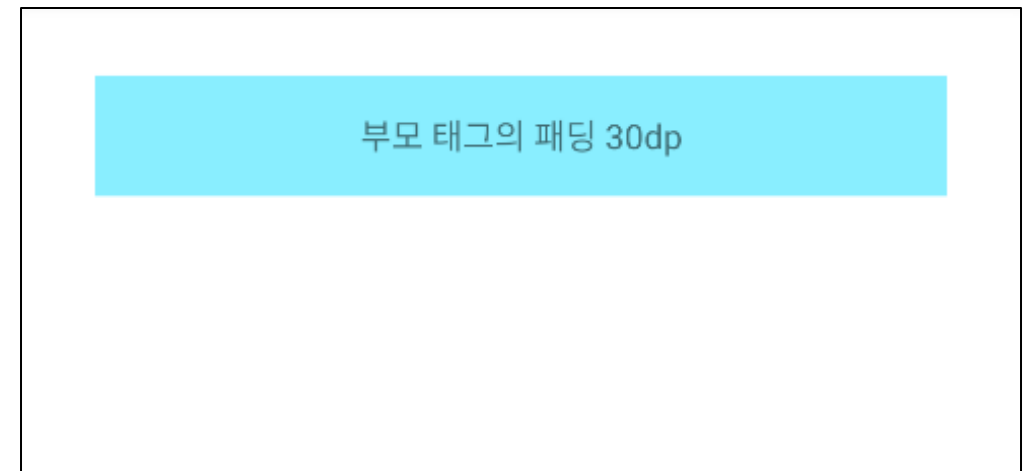
activity_main.xml

2. 기본 텍스트View 배치하기

- (16-17 line) : 너비, 높이 설정
- (18 line) : 텍스트 가로 정렬 설정
- (19 line) : 텍스트 세로 정렬 설정

```
15      <TextView
16          android:layout_width="match_parent"
17          android:layout_height="50dp"
18          android:textAlignment="center"
19          android:gravity="center"
20          android:background="#89EEFF"
21          android:text="부모 태그의 패딩 30dp"/>
22
23  </LinearLayout>
```

- 참고사항
 - **<TextView />** : 텍스트View Widget 태그, 문자열을 화면에 출력함
 - **textAlignment 속성** : 텍스트View 내부 텍스트의 가로 정렬 설정
 - “center” : 가운데 정렬
 - **gravity 속성** : 태그 내부의 콘텐츠의 세로 정렬 설정



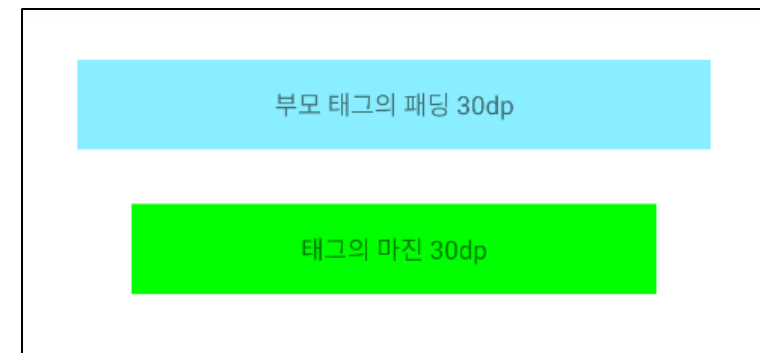
기초(따라하기) – 예제 2

activity_main.xml

3. 마진 텍스트View 배치하기

- (27 line) : 마진 설정
- 참고사항
 - **Layout_margin 속성** : Widget의 경계선 외부에 여백 영역 확보

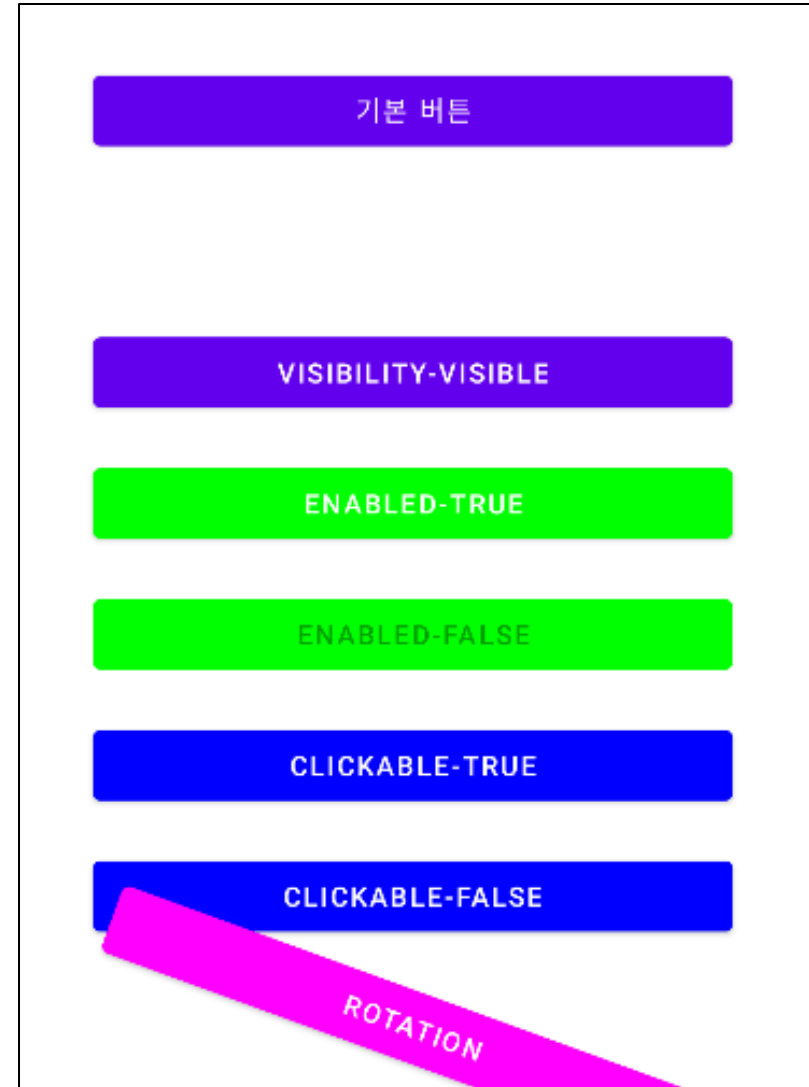
```
22     <TextView
23         android:layout_width="match_parent"
24         android:layout_height="50dp"
25         android:textAlignment="center"
26         android:gravity="center"
27         android:layout_margin="30dp"
28         android:background="#00FF00"
29         android:text="태그의 마진 30dp"/>
30 </LinearLayout>
```



기초(따라하기) – 예제 2

❖ View 의 시각 표현

- View 를 적절한 시각 표현을 사용하여 배치할 수 있다.
1. 기본 버튼 배치하기
 2. 가시 활성화 버튼 배치하기
 3. Widget 활성화 버튼 배치하기
 4. 상호작용 활성화 버튼 배치하기
 5. 버튼 회전 배치하기



기초(따라하기) – 예제 3

activity_main.xml

1. 기본 버튼 배치하기

- (1-9 line) : (생략) 메인 Layout
- (10-14 line) : Layout 배치 및 설정
- (15-19 line) : 버튼 배치 및 설정



```
10 <LinearLayout
11     android:layout_width="match_parent"
12     android:layout_height="wrap_content"
13     android:orientation="vertical"
14     android:padding="30dp">
15     <Button
16         android:layout_margin="10dp"
17         android:layout_width="match_parent"
18         android:layout_height="wrap_content"
19         android:text="기본 버튼" />
20
21 </LinearLayout>
```

기초(따라하기) – 예제 3

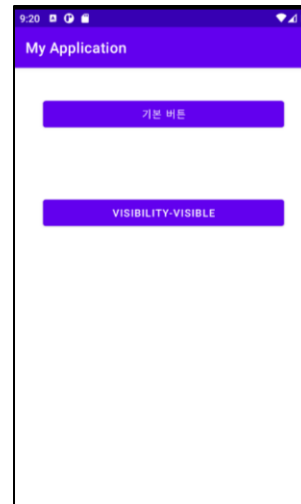
activity_main.xml

2. 버튼 가시 활성화 배치하기

- (24 line) : 버튼의 가시 속성 – 비 가시 설정
- (30 line) : 버튼의 가시 속성 – 숨김 설정
- (36 line) : 버튼의 가시 속성 – 가시 설정

참고사항

- **visibility 속성**: Widget의 가시에 대한 상태 설정
 - visible : (Default) 가시
 - invisible : 비 가시, Widget이 보이지만 않게
 - gone : 숨김, Widget의 영역까지 없음



```
20 <Button
21     android:layout_margin="10dp"
22     android:layout_width="match_parent"
23     android:layout_height="wrap_content"
24     android:visibility="invisible"
25     android:text="visibility-invisible" />
26 <Button
27     android:layout_margin="10dp"
28     android:layout_width="match_parent"
29     android:layout_height="wrap_content"
30     android:visibility="gone"
31     android:text="visibility-gone" />
32 <Button
33     android:layout_margin="10dp"
34     android:layout_width="match_parent"
35     android:layout_height="wrap_content"
36     android:visibility="visible"
37     android:text="visibility-visible" />
38
```

기초(따라하기) – 예제 3

3. Widget 활성화 버튼 배치하기

- (42 line) : 버튼의 색상 설정
- (43 line) : 버튼의 활성화 속성 – 활성화 설정
- (50 line) : 버튼의 활성화 설정 – 비활성화 설정
- 참고 사항
 - **backgroundTint 속성** : Widget의 색상 설정
 - “#00FF00”
 - **enabled 속성** : Widget 활성화 상태 설정, 시각적인 차이를 보임
 - “ture” : 활성
 - “false” : 비활성



activity_main.xml

```
38 <Button
39     android:layout_margin="10dp"
40     android:layout_width="match_parent"
41     android:layout_height="wrap_content"
42     android:backgroundTint="#00FF00"
43     android:enabled="true"
44     android:text="enabled-true" />
45 <Button
46     android:layout_margin="10dp"
47     android:layout_width="match_parent"
48     android:layout_height="wrap_content"
49     android:backgroundTint="#00FF00"
50     android:enabled="false"
51     android:text="enabled-false" />
52
```

기초(따라하기) – 예제 3

activity_main.xml

4. 상호작용 활성화 버튼 배치하기

- (57 line) : 버튼의 상호작용 활성화 설정
- (63 line) : 버튼의 상호작용 활성화 설정
- 참고사항
 - **clickable 속성** : 상호작용 활성화 상태 설정
 - 외관 변화 없음
 - “ture” : 활성화
 - “false” : 비활성



```
52 <Button
53     android:layout_margin="10dp"
54     android:layout_width="match_parent"
55     android:layout_height="wrap_content"
56     android:backgroundTint="#0000FF"
57     android:clickable="true"
58     android:text="clickable-true" />
59 <Button android:layout_margin="10dp"
60     android:layout_width="match_parent"
61     android:layout_height="wrap_content"
62     android:backgroundTint="#0000FF"
63     android:clickable="false"
64     android:text="clickable-false" />
65
```

기초(따라하기) – 예제 3

5. 회전된 버튼 배치하기

- (70 line) : 회전 속성 설정
- 참고 사항
 - **rotation 속성** : View 요소의 회전 각도 설정



activity_main.xml

```
65 <Button
66     android:layout_margin="10dp"
67     android:layout_width="match_parent"
68     android:layout_height="wrap_content"
69     android:backgroundTint="#FF00FF"
70     android:rotation="20"
71     android:text="rotation" />
72 </LinearLayout>
```

응용(로직구현) – 예제 4

❖ View 의 여러 요소 배치하기 와 시각 표현

- 여러 요소를 배치할 때 적절한 시각표현을 적용할 수 있다.
1. EditText Widget 배치하기
 2. Button Widget 배치하기
 3. TextView
 4. Widget 객체 연결하기
 5. 알고리즘 구성하기

숫자1

숫자2

더하기

빼기

곱하기

나누기

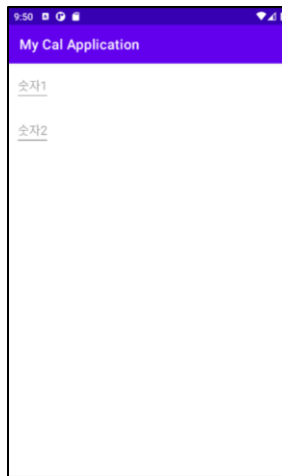
계산 결과 :

응용(로직구현) – 예제 4

activity_main.xml

1. EditText Widget 배치하기

- (14-19 line) : 에디트 텍스트 Widget 태그
- (18 line) : 태그 id 속성 값 등록
- (19 line) : 힌트 속성 문자열 설정
- 참고 사항
 - `<EditText />` : 문자열 입력 및 보관하는 Widget
 - **hint** 속성 : 문자열 입력 전, 문자열 입력에 대한 설명 제공
 - **id** 속성 : View 요소의 고유 식별자



```
10 <LinearLayout
11     android:layout_width="match_parent"
12     android:layout_height="match_parent"
13     android:orientation="vertical">
14     <EditText
15         android:layout_width="wrap_content"
16         android:layout_height="wrap_content"
17         android:layout_margin="10dp"
18         android:id="@+id/Edit1"
19         android:hint="숫자1" />
20     <EditText
21         android:layout_width="wrap_content"
22         android:layout_height="wrap_content"
23         android:layout_margin="10dp"
24         android:id="@+id/Edit2"
25         android:hint="숫자2" />
26
```

응용(로직구현) – 예제 4

activity_main.xml

2. Button Widget 배치하기

- (26 line) : 버튼 태그 배치
- (30 line) : id 속성 값 등록
- (32 line) : 버튼 태그 배치
- (38 line) : 버튼 태그 배치
- (44 line) : 버튼 태그 배치



- 참고 사항
 - ... : 태그의 내부 내용(속성, 예하 요소) 요약 표현

```
26 <Button
27     android:layout_width="match_parent"
28     android:layout_height="wrap_content"
29     android:layout_margin="10dp"
30     android:id="@+id/BtnAdd"
31     android:text="더하기" />
32 <Button...>
38 <Button...>
44 <Button...>
50
```

응용(로직구현) – 예제 4

activity_main.xml

3. TextView Widget 배치하기

- (55 line) : 텍스트 사이즈 설정
- (56 line) : 텍스트 색상 설정



```
50 <TextView
51     android:layout_width="wrap_content"
52     android:layout_height="wrap_content"
53     android:layout_margin="10dp"
54     android:id="@+id/TextResult"
55     android:textSize="30dp"
56     android:textColor="#FF0000"
57     android:text="계산 결과 : "/>
58 </LinearLayout>
```

- 참고사항
 - **textSize** 속성 : 화면에 출력하는 텍스트의 크기 설정
 - **textColor** 속성 : 화면에 출력하는 텍스트의 색상 설정

응용(로직구현) – 예제 4

MainActivity.kt

4. Widget 객체 생성하기 배치하기

- (1 line) : 패키지 등록
- (2-7 line) : 사용 라이브러리 추가
- (9 line) : 메인 클래스
- (10-13 line) : Widget 클래스 객체 선언
- (14-15 line) : 변수 선언
- 참고 사항
 - **package** : 클래스의 모음, 프로젝트 내부 폴더
 - com.example.mycalapplication : 프로젝트 이름
 - **import** : 패키지, 클래스를 추가함
 - AppCompatActivity : 호환 액티비티 메소드
 - Bundle : Map 클래스, 액티비티 간의 송수신 데이터 구조
 - Button / EditText / TextView : Widget 클래스
 - **MainActivity** : 메인 소스코드 파일의 이름이자 메인 클래스의 이름
 - **AppCompatActivity()** : 생성자 메소드
 - **Lateinit var / var** : 늦은 초기화 변수 / 변수
 - **Int?** : 자료형에 Null 상태 허용

```
1 package com.example.mycalapplication
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 import android.widget.Button
6 import android.widget.EditText
7 import android.widget.TextView
8
9 class MainActivity : AppCompatActivity() {
10     lateinit var edit1 : EditText; lateinit var edit2 : EditText
11     lateinit var btnAdd : Button;   lateinit var btnSub : Button
12     lateinit var btnMul : Button;   lateinit var btnDiv : Button
13     lateinit var textResult : TextView
14     lateinit var num1 : String; lateinit var num2 : String
15     var result : Int? = null
16 }
```

응용(로직구현) – 예제 4

MainActivity.kt

5. Widget 객체 생성하기

- (17-19 line) : 메인 클래스 형식
- (20 line) : 앱 제목 설정
- (22-23 line) : 에디트텍스트 객체 연결
- (25-28 line) : 버튼 객체 연결
- (29 line) : 텍스트View 객체 연결
- 참고 사항
 - `onCreate()` : 액티비티 생성
 - `SetContentView()` : 액티비티 화면 출력
 - `findViewById<태그>(id 속성 값)` : id 속성값을 통해서 객체와 태그 연결

```
17 override fun onCreate(savedInstanceState: Bundle?) {
18     super.onCreate(savedInstanceState)
19     setContentView(R.layout.activity_main)
20     title = "초간단 계산기"
21
22     edit1 = findViewById<EditText>(R.id.Edit1)
23     edit2 = findViewById<EditText>(R.id.Edit2)
24
25     btnAdd = findViewById<Button>(R.id.BtnAdd)
26     btnSub = findViewById<Button>(R.id.BtnSub)
27     btnMul = findViewById<Button>(R.id.BtnMul)
28     btnDiv = findViewById<Button>(R.id.BtnDiv)
29     textResult = findViewById<TextView>(R.id.TextResult)
30 }
```

응용(로직구현) – 예제 4

MainActivity.kt

6. 알고리즘 구성하기

- (31 line) : 버튼 터치 리스너 생성
- (39 line) : 버튼 터치 리스너 생성
- (47 line) : 버튼 터치 리스너 생성
- (55 line) : 버튼 터치 리스너 생성
- (56-57 line) : 태그 속성 값 가져오기
- (59 line) : 태그 속성 값 덮어쓰기

```
31      btnAdd.setOnClickListener{...}
38
39      btnSub.setOnClickListener{...}
46
47      btnMul.setOnClickListener{...}
54
55      btnDiv.setOnClickListener{ view, motionEvent ->
56          num1 = edit1.text.toString()
57          num2 = edit2.text.toString()
58          result = Integer.parseInt(num1) / Integer.parseInt(num2)
59          textResult.text = "계산 결과 : "+result.toString()
60          false ^setOnClickListener
61      }
62  }
63 }
```

- 참고 사항
 - **setOnClickListener()**: 실행 화면에서 해당 Widget의 터치 인식
 - **Widget객체.text**: 객체와 연결된 Widget의 text 값 접근(가져오기, 덮어쓰기)
 - **숫자toString()**: 숫자를 문자열로 변환

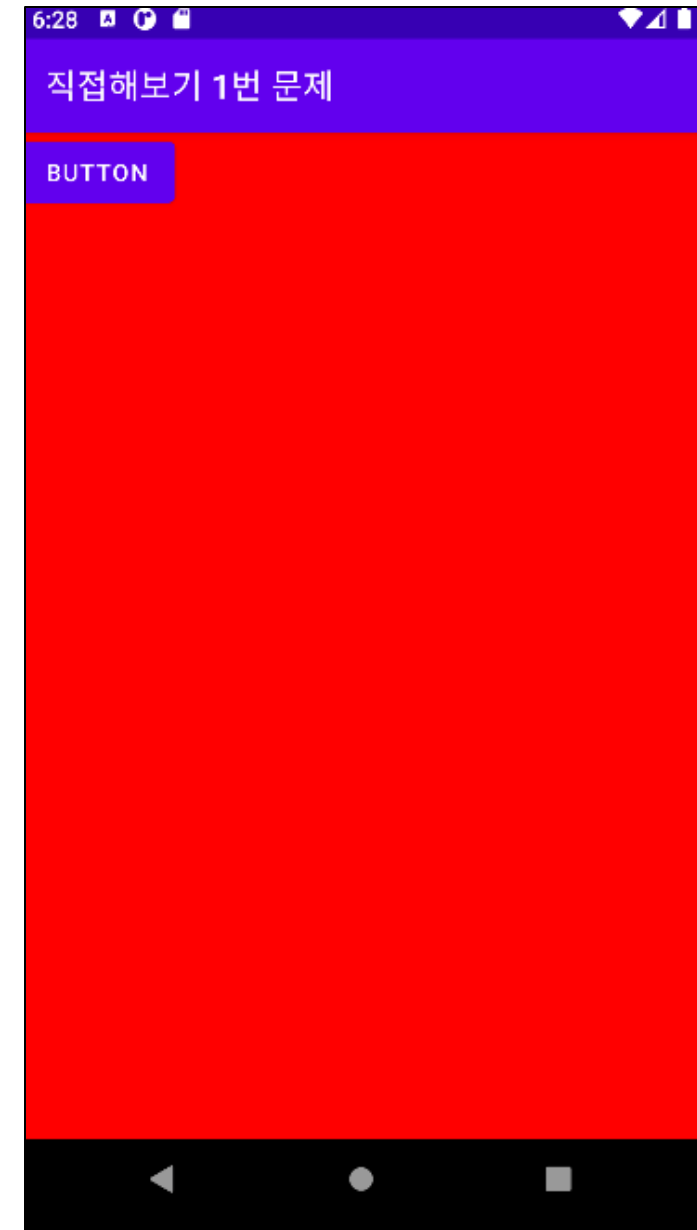
심화(완성하기) – 예제 5

❖ 버튼 터치로 배경색 변경하기

- 버튼 터치 시 배경화면 색상 전환
 - #FF0000, #00FF00, #0000FF
 - 순서 : 빨 – 초 – 파 – 빨 - ...
- 터치가 두 번씩 되는 문제는 본 예제에서 다루지 않음

Hint

```
lateinit var linLayer : LinearLayout
linLayer = findViewById<LinearLayout>(R.id.LinLay)
linLayer.setBackgroundColor(Color.parseColor( colorString: "#FF0000"))
```



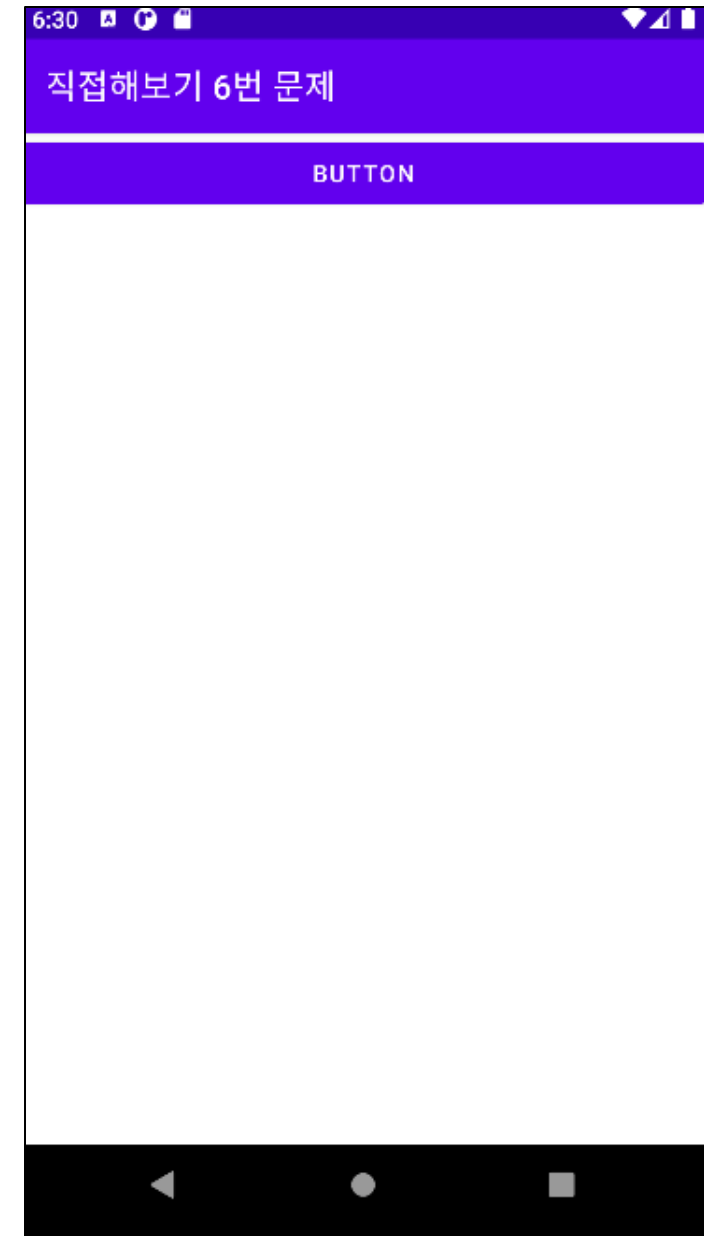
심화(완성하기) – 예제 6

❖ 버튼 릴레이

- 버튼을 누르면, 5개의 버튼이 순차적으로 활성화됨
 - 순서 : 버튼1 – 버튼2 – 버튼3 – 버튼4 – 버튼5 – 버튼1 -

Hint

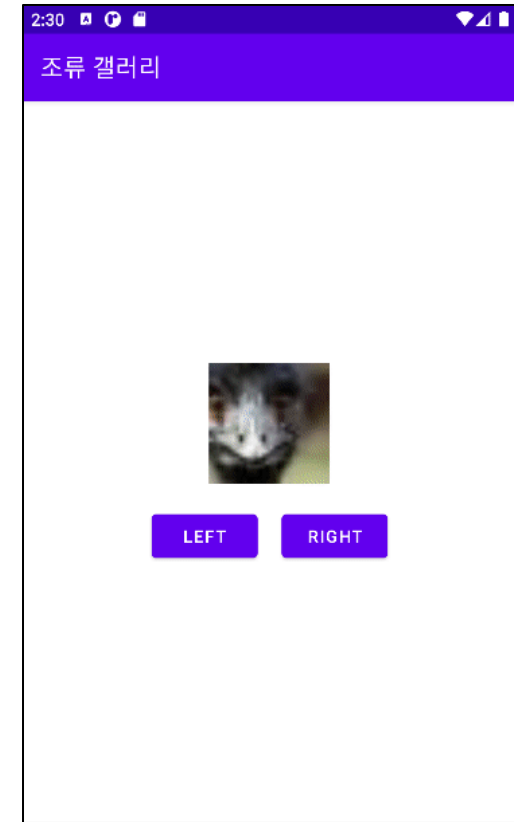
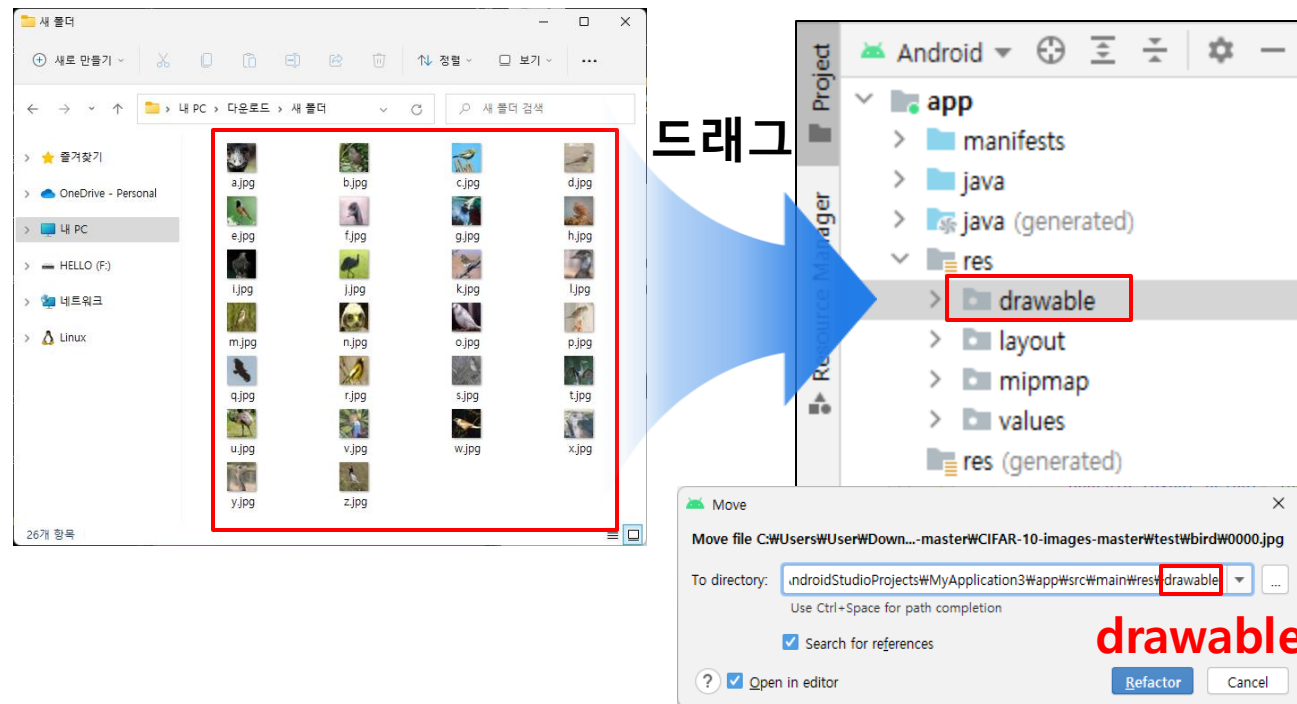
```
btn1.visibility = View.INVISIBLE  
btn2.visibility = View.VISIBLE
```



심화(완성하기) – 예제 7

❖ 갤러리 만들기

- 각 버튼으로 앞 / 뒤 이미지 화면에 출력하기
- 이미지 파일 26개 넣기
 - 본인이 좋아하는 이미지 가져오기
 - app/res/drawable 으로 드래그로 파일 넣기
 - 파일 이름은 'a' ~ 'z' 으로 지정



Hint

```
<ImageView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:layout_margin="10dp"
    android:layout_gravity="center"
    android:src="@drawable/a" />

lateinit var img : ImageView
img = findViewById<ImageView>(R.id.imgView)
img.setImageResource(R.drawable.a)
```

심화(과제물) – 예제 8

❖ 계산기 확장하기

- 예외 처리하기
 - 두 개의 숫자가 값이 존재해야만 연산 수행
- 기능 추가하기
 - 나머지 연산, 교체
 - 계산 결과, 결과값이 Num1에 업데이트되고, Num2는 비우기
- 애플리케이션 제목에 계산 횟수 넣기
- 애플리케이션이 비정상 종료되면 안됨

