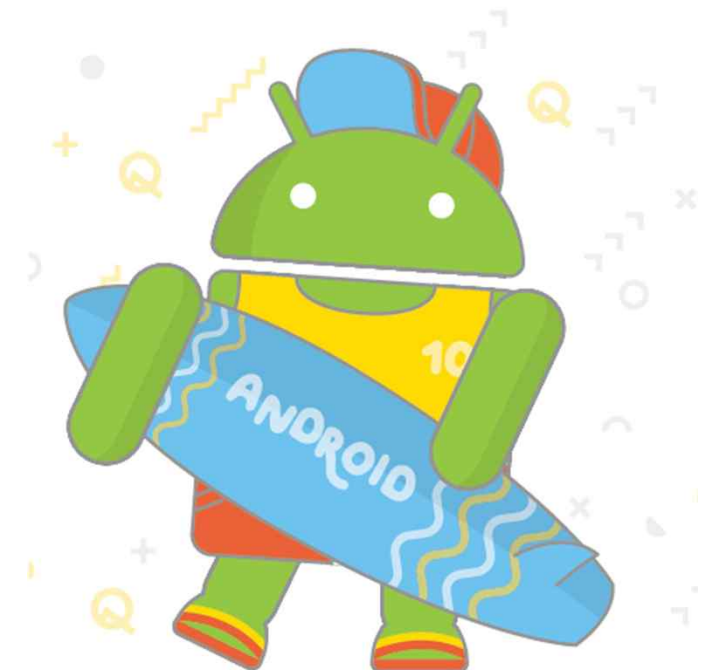


04 기본 위젯 익히기



IT CookBook, 코틀린을 활용한 안드로이드 프로그래밍

학습목표

- ❖ 뷰와 뷰 상속을 이해한다.
- ❖ 기본 위젯을 다루는 방법을 익힌다.
- ❖ 안드로이드 앱의 기본적인 프로그래밍을 숙달한다.

차례

1. 뷰의 개요
2. 기본 위젯 다루기
3. 기본 위젯 활용하기

01 뷰와 뷰그룹

■ 뷰(View) 클래스

- 안드로이드 화면에서 실제로 사용되는 것들은 모두 View 클래스의 상속을 받음
 - 버튼, 라디오버튼, 이미지 등은 모두 View 클래스의 서브클래스임
- '위젯'이라고도 부름
 - 화면에서의 버튼 : 버튼 위젯
 - 실제 코드에서의 버튼 : 버튼 클래스

■ 레이아웃

- 다른 위젯을 담을 수 있는 위젯을 특별히 레이아웃이라고 함
 - 즉, 위젯을 담아 배치하는 틀
- ViewGroup 클래스 아래에 존재함
 - 레이아웃도 크게 보면 위젯에 포함됨

01 뷰와 뷰그룹

■ 안드로이드에서의 위젯

- 다른 프로그래밍 언어에서 컨트롤(control)이라고 부르는 것들
 - 버튼(Button), 텍스트뷰(TextView), 에디트텍스트(EditText), 라디오버튼(RadioButton), 이미지뷰(ImageView) 등

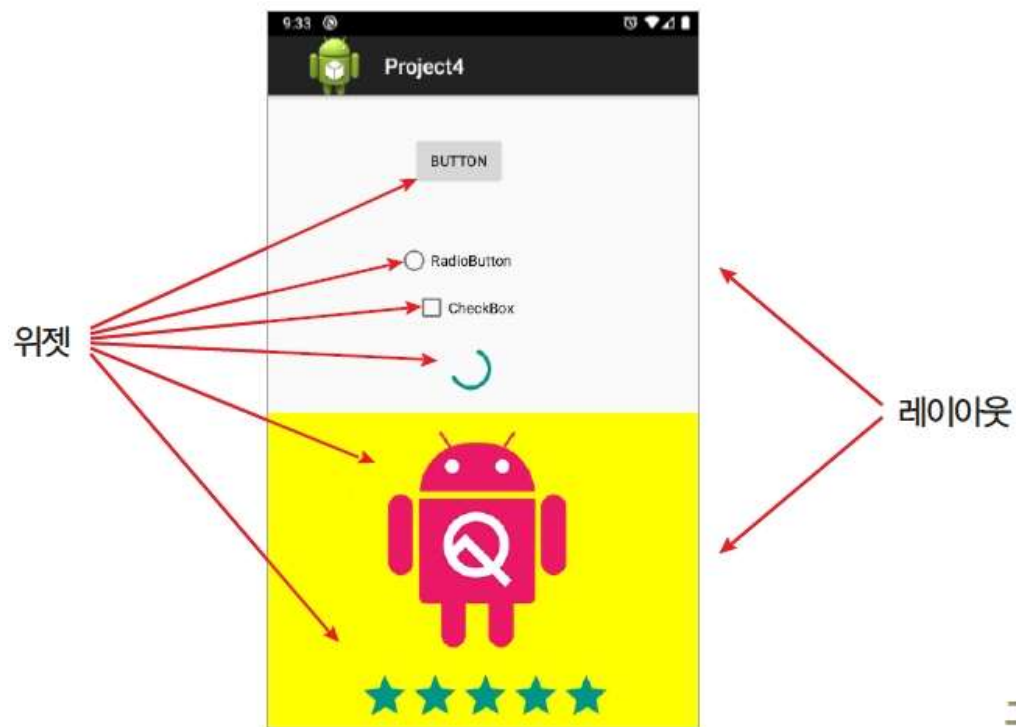


그림 4-1 위젯과 레이아웃

01 뷰와 뷰그룹

■ View 클래스 계층도

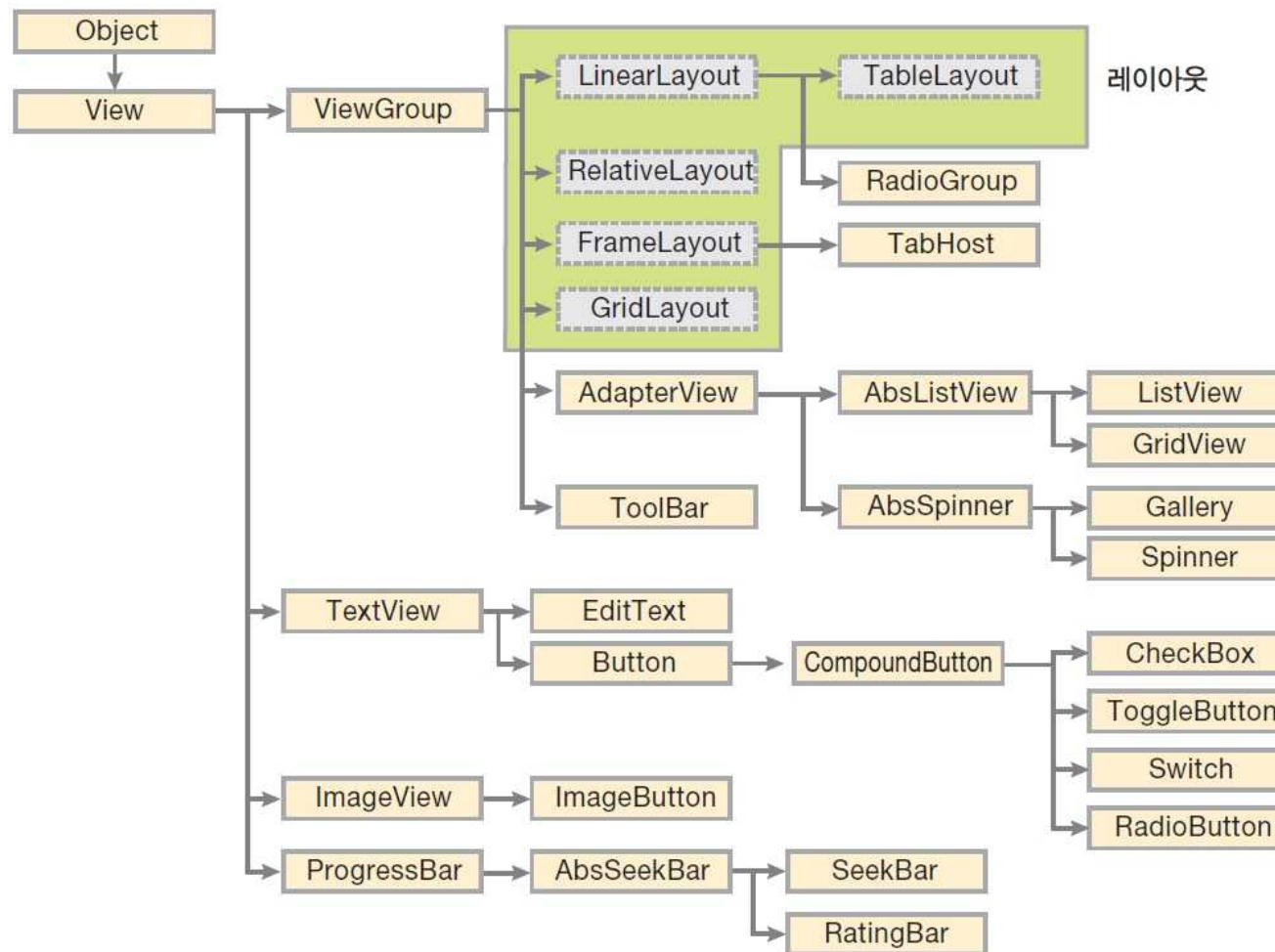


그림 4-2 안드로이드의 View 클래스 계층도

01 뷰와 뷰그룹

■ View 클래스 계층도

- 최상위에 Object(java.lang.Object) 클래스가 있고 이를 상속받은 View 클래스가 있음
- 안드로이드 화면에 나타나는 모든 위젯은 View 하위에 존재함
 - 레이아웃은 ViewGroup을 상속받은 LinearLayout, RelativeLayout, FrameLayout, GridLayout, TableLayout을 지칭함.
 - 뷰 컨테이너(view container)
 - 레이아웃이라고 부르지는 않지만 다른 뷰를 포함하는 ListView, GridView, TabHost, Gallery 등
 - 뷰 컨테이너도 ViewGroup 클래스에서 상속받음

01 뷰와 뷰그룹

■ 버튼의 속성

- XML 속성이 거의 없고 대개 상위 클래스인 TextView나 View에서 상속받음

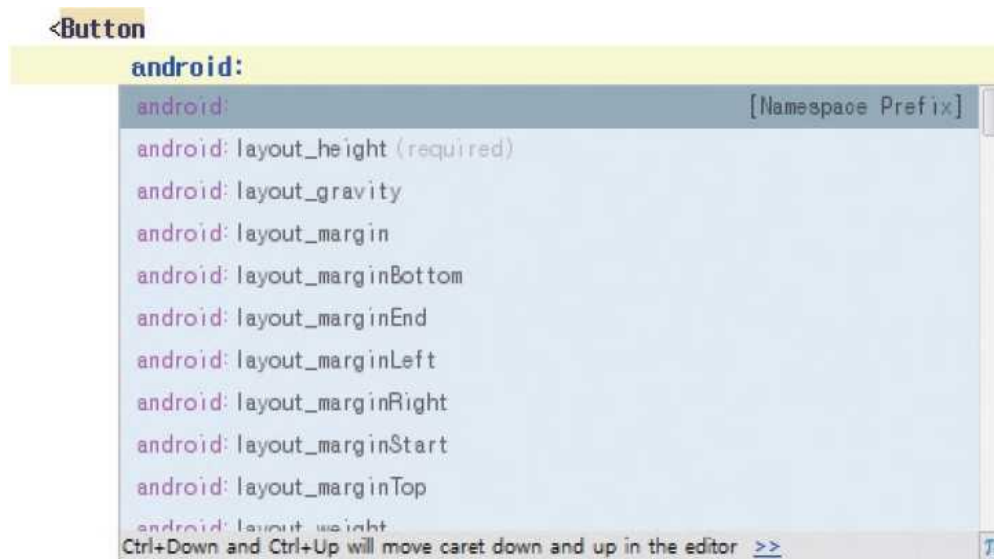


그림 4-3 Button의 XML 속성

01 뷰와 뷰그룹

■ 버튼의 속성

- <https://developer.android.com/reference/packages>에서 자세히 확인 가능

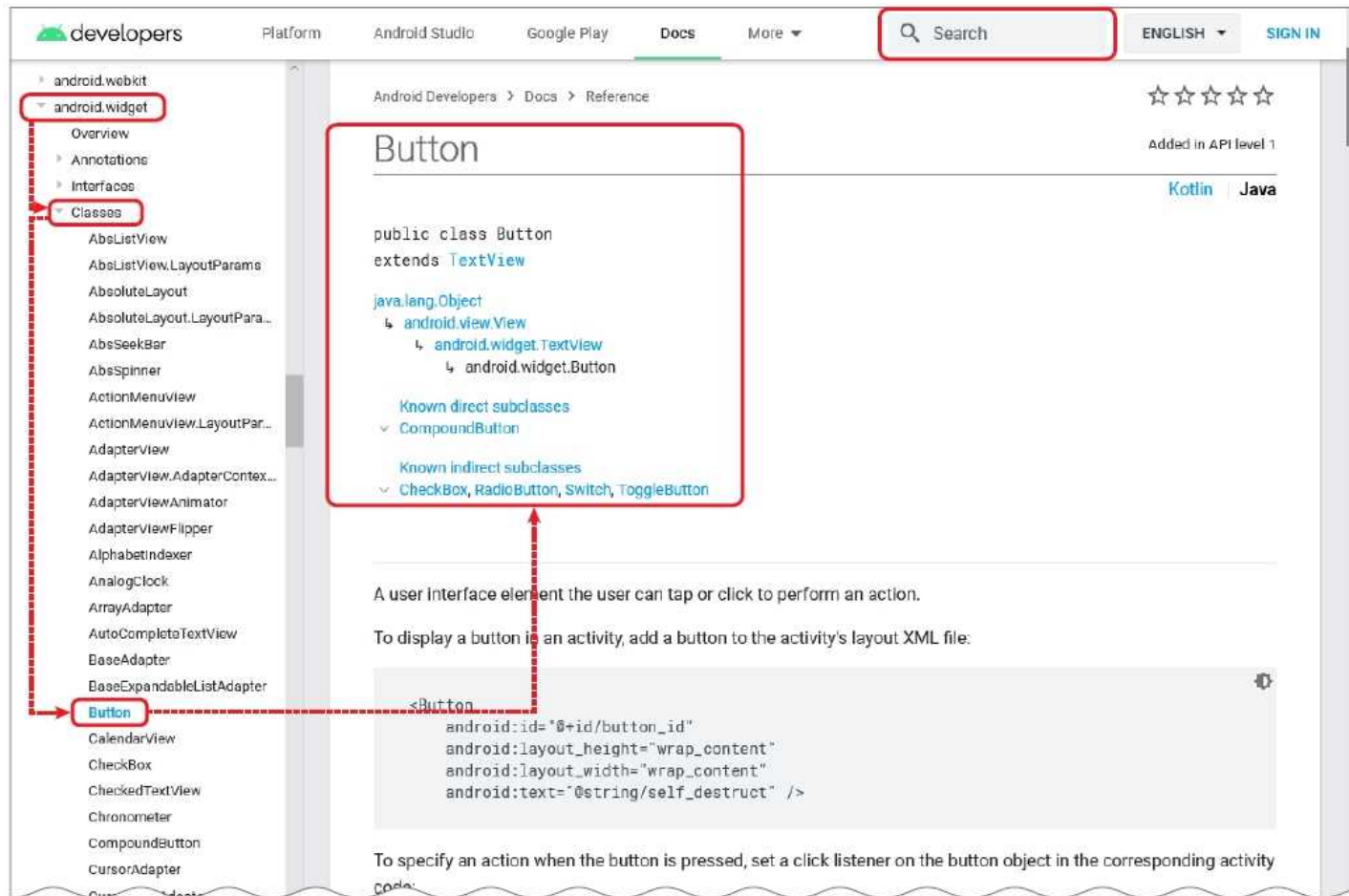


그림 4-4 클래스의 상속 관계를 찾는 방법

02 View 클래스의 XML 속성

■ View 클래스의 주요 XML 속성

- 안드로이드 계층도를 보면 View 클래스가 모든 위젯의 부모
 - 즉, 위젯과 레이아웃 등은 모두 View 클래스의 속성과 메소드를 상속받음
- View 클래스의 중요한 속성을 파악하면 하위 클래스도 쉽게 이해할 수 있음
 - Button의 기본 형태

```
<Button  
    android:id="@+id/btn1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:background="#ff0000"  
    android:text="버튼입니다"  
>
```

02 View 클래스의 XML 속성

저자
한마디

책의 XML 코드 표기(★주의)

텍스트뷰 1개와 버튼 1개가 있는 기본적인 activity_main.xml의 전체 코드는 다음과 같다.

예제 4-8 XML 전체 코드

```

1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/1
2             android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:orientation="vertical" >
7
8     <TextView
9         android:layout_width="match_parent"
10        android:layout_height="wrap_content"
11        android:text="@string/hello_world" />
12
13    <Button
14        android:id="@+id/button1"
15        android:layout_width="wrap_content"
16        android:layout_height="wrap_content"
17        android:text="Button" />
18
19 </LinearLayout>

```

02 View 클래스의 XML 속성

1~3행의 xmlns 이후에는 항상 동일한 내용이 나오고 4행, 5행, 9행, 10행, 15행, 16행에 나오는 layout_width와 layout_height는 모든 위젯의 필수 요소로 자주 등장할 것이다. 5행의 orientation 속성도 거의 고정되어 나온다. 따라서 이 책에서 이러한 반복적인 속성은 꼭 표현할 필요가 있을 때를 제외하고는 생략할 것이다. 즉 앞으로는 다음과 같이 필수 또는 반복되는 코드를 생략하고 간략하게 표현한다.

예제 4-9 앞으로 책에서 표기할 XML 코드

```

1  <LinearLayout>
2
3      <TextView
4          android:text="@string/hello_world" />
5
6      <Button
7          android:id="@+id/button1"
8          android:text="Button" />
9
10 </LinearLayout>

```

<LinearLayout>과 </LinearLayout>도 따로 추가할 설명이 없다면 생략하겠다.

간단히 추려놓으니 확인해야 할 코드가 명확하게 보인다. 앞으로는 이런 식으로 XML 코드를 표기하지만, 실제로는 전체 코드를 다 표현해야 동작함을 잊지 말기 바란다. 주의할 점은 [예제 4-8] 6행의 orientation 속성을 생략하면 LinearLayout이 horizontal이 된다는 것이다. 이 책의 화면은 대부분 vertical 정렬이므로 예제에는 생략되어 있어도 6행을 잊지 말고 써야 한다. 이는 5장에서 다시 설명하겠다. 생략하지 않은 모든 소스코드는 자료실(<http://www.hanbit.co.kr/src/4496/>)에 올려놓았다.

02 View 클래스의 XML 속성

■ id 속성

- 모든 위젯의 아이디를 나타냄
- Kotlin 코드에서 버튼 등의 위젯에 접근할 때 id 속성에 지정한 아이디를 사용
- 일반적으로 id 속성은 위젯에 아이디를 새로 부여하는 개념
 - '@+id/' 형식
 - / 다음에는 새로 지정할 id를 넣음
 - android:id="@+id/btn1" : 버튼 위젯의 아이디로 btn1을 부여함
 - Kotlin 코드에서 다음과 같은 접근 방식을 사용할 수 있음

```
위젯 변수 = findViewById<위젯형>(R.id.위젯id)
```

```
var button1 : Button  
button1 = findViewById<Button>(R.id.button1)
```

02 View 클래스의 XML 속성

- 터치했을 때 어떤 동작이 필요한 경우 id 지정함
 - Button, RadioButton, CheckBox 등
 - TextView, ImageView(배경 이미지일 경우)등은 굳이 id 속성을 지정하지 않아도 됨

예제 4-1 id 속성의 XML 코드

```

1 <LinearLayout ~~~ 생략 ~~~
2     android:orientation="vertical" >
3     <TextView
4         android:id="@+id/textView1"
5         android:layout_width="wrap_content"
6         android:layout_height="wrap_content"
7         android:text="성별 선택" />
8     <RadioButton
9         android:id="@+id/female"
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content"
12        android:text="여성" />
13    <RadioButton
14        android:id="@+id/male"
15        android:layout_width="wrap_content"
16        android:layout_height="wrap_content"
17        android:text="남성" />
18 </LinearLayout>

```

성별 선택

- ☐ 여성
☒ 남성

02 View 클래스의 XML 속성

- **layout_width, layout_height 속성**
 - 모든 위젯에 필수로 들어감
 - 이 둘은 각각 위젯의 너비와 높이를 나타냄
 - match_parent와 wrap_content 값으로 설정할 수 있음
- **wrap_content**
 - 버튼의 너비가 그 안의 글자인 '버튼입니다'에 꼭 맞는 크기가 됨
- **match_parent(또는 fill_parent)**
 - '버튼입니다' 글자와 관계없이 버튼을 싸고 있는 부모(리니어레이아웃)의 너비에 꼭 차는 크기가 됨
- **layout_height는 버튼의 높이에 대해 적용됨**

02 View 클래스의 XML 속성

예제 4-2 layout_width, layout_height 속성의 XML 코드 1

```

1 <LinearLayout
2     ~~~ 생략 ~~~ >
3     <Button
4         android:layout_width="wrap_content"
5         android:layout_height="wrap_content"
6         android:text="버튼입니다" />
7 </LinearLayout>

```

버튼입니다

부모 레이아웃

예제 4-3 layout_width, layout_height 속성의 XML 코드 2

```

1 <LinearLayout
2     ~~~ 생략 ~~~ >
3     <Button
4         android:layout_width="match_parent"
5         android:layout_height="wrap_content"
6         android:text="버튼입니다" />
7 </LinearLayout>

```

버튼입니다

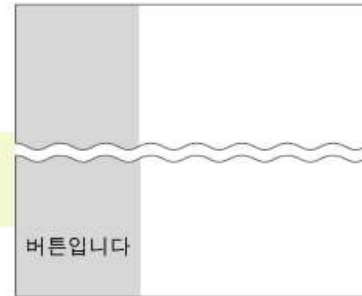
02 View 클래스의 XML 속성

예제 4-4 layout_width, layout_height 속성의 XML 코드 3

```

1 <LinearLayout
2     ~~~ 생략 ~~~ >
3     <Button
4         android:layout_width="wrap_content"
5         android:layout_height="match_parent"
6         android:text="버튼입니다" />
7 </LinearLayout>

```



예제 4-5 layout_width, layout_height 속성의 XML 코드 4

```

1 <LinearLayout
2     ~~~ 생략 ~~~ >
3     <Button
4         android:layout_width="match_parent"
5         android:layout_height="match_parent"
6         android:text="버튼입니다" />
7 </LinearLayout>

```



02 View 클래스의 XML 속성

- 값을 숫자로 직접 지정하는 경우
 - 단위에 주의 해야함
 - px(PiXe) 단위
 - 가장 단순한 단위
 - AVD는 해상도가 1080×1920인 경우 너비를 1080px, 높이를 1920px로 지정하면 match_parent로 지정한 것처럼 버튼이 화면에 꽉 참

예제 4-6 layout_width, layout_height 속성의 XML 코드 5

```
1 <LinearLayout
2   ~~~ 생략 ~~~ >
3   <Button
4       android:layout_width="1080px"
5       android:layout_height="1920px"
6       android:text="버튼입니다" />
7 </LinearLayout>
```



02 View 클래스의 XML 속성

■ background 속성

- 위젯의 색상을 주로 #RRGGBB 값으로 지정
- 각 값은 빨간색, 초록색, 파란색을 의미함
- RR, GG, BB의 위치는 16진수 00~FF로 표현할 수 있음

예제 4-7 background 속성의 XML 코드

```
1 <LinearLayout
2     android:background="#ff0000"
3     ~~~ 생략 ~~~ >
4     <Button
5         android:layout_width="wrap_content"
6         android:layout_height="wrap_content"
7         android:background="#00ff00"
8         android:text="버튼입니다" />
9 </LinearLayout>
```



02 View 클래스의 XML 속성

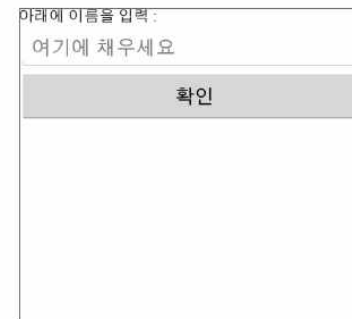
- padding, layout_margin 속성
 - padding 속성을 사용하면 레이아웃의 경계선과 위젯 사이에 여백을 둘 수 있음

예제 4-10 간격이 없는 XML 코드

```

1 <LinearLayout >
2     <TextView
3         android:text="아래에 이름을 입력 : " />
4     <EditText
5         android:hint="여기에 채우세요" />
6     <Button
7         android:text="확인" />
8 </LinearLayout>

```



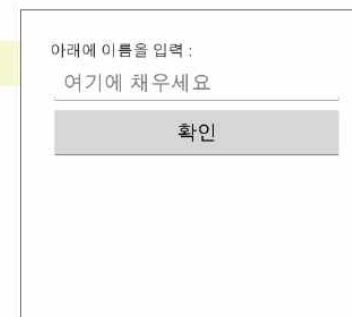
- 레이아웃에 padding 속성을 사용

예제 4-11 padding 속성의 XML 코드

```

1 <LinearLayout
2     android:padding="30dp" >
3     <TextView
4         android:text="아래에 이름을 입력 : " />
5     <EditText
6         android:hint="여기에 채우세요" />
7     <Button
8         android:text="확인" />
9 </LinearLayout>

```

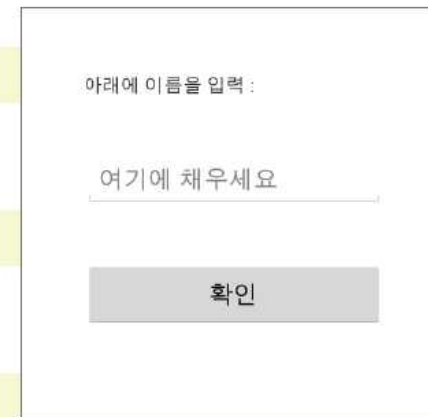


02 View 클래스의 XML 속성

- padding, layout_margin 속성
 - Button에 padding을 설정하면 버튼 안의 글자가 버튼의 경계선에서 일정 간격 떨어져서 표현됨
 - 위젯과 위젯 사이에 여유를 두고 싶다면 layout_margin 속성을 사용

예제 4-12 layout_margin 속성의 XML 코드

```
1 <LinearLayout
2     android:padding="30dp" >
3     <TextView
4         android:layout_margin="20dp"
5         android:text="아래에 이름을 입력 :" />
6     <EditText
7         android:layout_margin="20dp"
8         android:hint="여기에 채우세요" />
9     <Button
10        android:layout_margin="20dp"
11        android:text="확인" />
12 </LinearLayout>
```



아래에 이름을 입력 :

여기에 채우세요

확인

02 View 클래스의 XML 속성

■ visibility 속성

- 위젯을 보일 것인지 여부를 결정함
 - visible
 - 디폴트로 설정되어 있으며, 보이는 상태
 - invisible
 - 안 보이는 상태이지만, 보이지 않을 뿐 원래의 자리를 계속 유지함
 - gone
 - 안 보이는 상태이며, 원래의 자리까지 사라짐

예제 4-13 visibility 속성의 XML 코드

```

1 <Button
2     android:text="버튼 1" />
3 <Button
4     android:visibility="invisible"
5     android:text="버튼 2" />
6 <Button
7     android:visibility="visible"
8     android:text="버튼 3" />
9 <Button
10    android:visibility="gone"
11    android:text="버튼 4" />
12 <Button
13    android:text="버튼 5" />

```



02 View 클래스의 XML 속성

- **enabled, clickable 속성**
 - XML 보다 Kotlin 코드에서 주로 사용됨
 - 값은 true와 false이며 디폴트 값은 true임
 - enabled 속성
 - 위젯의 동작 여부
 - clickable 속성
 - 클릭이나 터치가 동작 여부

예제 4-14 enabled, clickable 속성의 XML 코드

```
1 <Button
2     android:text="버튼 1" />
3 <Button
4     android:enabled="false"
5     android:text="버튼 2" />
6 <Button
7     android:clickable="false"
8     android:text="버튼 3" />
```

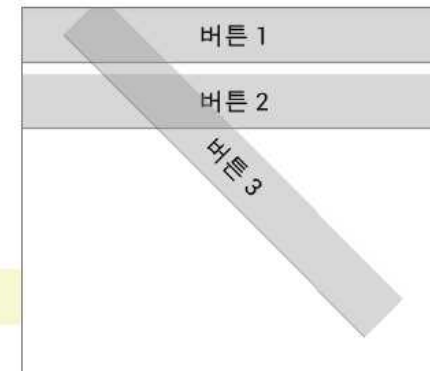


02 View 클래스의 XML 속성

- rotation 속성
 - 위젯을 회전시켜서 출력함
 - 값은 각도로 지정

예제 4-15 rotation 속성의 XML 코드

```
1 <Button
2     android:text="버튼 1" />
3 <Button
4     android:text="버튼 2" />
5 <Button
6     android:rotation="45"
7     android:text="버튼 3" />
```



02 View 클래스의 XML 속성

▶ 직접 풀어보기 4-2

다음과 같은 화면을 XML로 코딩하라. 버튼, 텍스트뷰, 에디트텍스트, 버튼을 차례로 지정하고 앞에서 배운 다양한 속성을 사용한다.



그림 4-5 다양한 XML 속성을 적용한 화면

01 텍스트뷰

텍스트뷰 계층도

```
java.lang.Object
└─ android.view.View
    └─ android.widget.TextView
```

- View 클래스 바로 다음에 위치하며 다양한 위젯이 그 하위에 존재함
- **text 속성**
 - 텍스트뷰에 나타나는 문자열을 표현
 - “문자열” 형식으로 값을 직접 입력하거나 “@string/변수명” 형식으로 지정한 후 strings.xml 파일에 지정할 수 있음
- **textColor 속성**
 - 글자의 색상을 지정
 - background 속성처럼 값은 #RRGGBB나 #AARRGGBB 형식을 사용함

01 텍스트뷰

- **textSize 속성**
 - 글자의 크기를 dp, px, in, mm, sp 단위로 지정
- **typeface 속성**
 - 글자의 글꼴을 지정
 - 값으로 sans, serif, monospace를 설정할 수 있고 디폴트는 normal
- **textStyle 속성**
 - 글자의 스타일을 지정
 - 값으로 bold, italic, bold|italic을 설정할 수 있고 디폴트는 normal
- **singleLine 속성**
 - 글이 길어 줄이 넘어갈 경우 강제로 한 줄까지만 출력하고 문자열의 맨 뒤에 '...'를 표시
 - 값으로 true와 false를 설정할 수 있고 디폴트는 false

01 텍스트뷰

예제 4-16 글자 관련 속성의 XML 코드


<pre> 1 <TextView 2 android:textSize="30dp" 3 android:text="textSize 속성" /> 4 <TextView 5 android:textSize="30dp" 6 android:textColor="#00FF00" 7 android:text="textColor 속성" /> 8 <TextView 9 android:textSize="30dp" 10 android:textStyle="bold italic" 11 android:text="textStyle 속성" /> 12 <TextView 13 android:textSize="30dp" 14 android:typeface="serif" 15 android:text="typeface 속성" /> 16 <TextView 17 android:textSize="30dp" 18 android:singleLine="true" 19 android:text="singleLine 속성 singleLine 속성 singleLine 속성" /> </pre>	<div style="border: 1px solid black; padding: 5px;"> <p>textSize 속성</p> <p>textColor 속성</p> <p>textStyle 속성</p> <p>typeface 속성</p> <p>singleLine 속성 single...</p> </div>
--	---

02 Kotlin 코드로 XML 속성 설정

- activity_main.xml 파일에서 지정한 XML 속성을 Kotlin 코드에서 설정하는 방법
- 기본적인 텍스트뷰만 만들어놓고 id 속성과 text만 설정한 XML 파일

예제 4-17 텍스트뷰가 3개 있는 activity_main.xml

```
1 <TextView
2     android:text="TextView 연습 1"
3     android:id="@+id/textView1" />
4 <TextView
5     android:text="TextView 연습 2"
6     android:id="@+id/textView2" />
7 <TextView
8     android:text="TextView 연습 3"
9     android:id="@+id/textView3" />
```



TextView 연습 1
TextView 연습 2
TextView 연습 3

02 Kotlin 코드로 XML 속성 설정

- activity_main.xml 파일에서 지정한 XML 속성을 Kotlin 코드에서 설정하는 방법
 - Kotlin 코드를 다음과 같이 설정하여 화면에 적용

예제 4-18 텍스트 속성을 변경하는 Kotlin 코드

```

1  override fun onCreate(savedInstanceState: Bundle?) {
2      super.onCreate(savedInstanceState)
3      setContentView(R.layout.activity_main)
4
5      var tv1 : TextView
6      var tv2 : TextView
7      var tv3 : TextView
8
9      tv1 = findViewById<TextView>(R.id.textView1)
10     tv2 = findViewById<TextView>(R.id.textView2)
11     tv3 = findViewById<TextView>(R.id.textView3)
12
13     tv1.setText("안녕하세요?")
14     tv1.setTextColor(Color.RED)
15     tv2.setTextSize(30.0f)
16     tv2.setTypeface(android.graphics.Typeface.SERIF,
17                     android.graphics.Typeface.BOLD_ITALIC)
18     tv3.setText("가나다라마바사아자차카타파하가나다라마바사아자차카타파하")
19     tv3.setSingleLine()
20 }

```

안녕하세요?

TextView 연습 2

가나다라마바사아자차카타파하가나다라마바사아자차카타파하

02 Kotlin 코드로 XML 속성 설정

- 많이 사용되는 View 클래스 또는 TextView 클래스의 XML 속성과 메소드

표 4-1 XML 속성과 관련 메소드

XML 속성	관련 메소드	비고
background	setBackgroundColor()	View 클래스
clickable	setClickable()	View 클래스
focusable	setFocusable()	View 클래스
id	setId()	View 클래스
longClickable	setLongClickable()	View 클래스
padding	setPadding()	View 클래스
rotation	setRotation()	View 클래스
scaleX, scaleY	setScaleX(), setScaleY()	View 클래스
visibility	setVisibility()	View 클래스
gravity	setGravity()	TextView 클래스
inputType	setRawInputType()	TextView 클래스
password	setTransformationMethod()	TextView 클래스
text	setText()	TextView 클래스
textColor	setTextColor()	TextView 클래스
textSize	setTextSize()	TextView 클래스

03 버튼과 에디트텍스트

- 버튼과 에디트텍스트는 사용자에게서 어떤 값을 입력받기 위한 가장 기본적인 위젯
- 두 위젯은 View 클래스와 TextView 클래스를 상속받으므로 비슷하게 사용 가능함

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="나는 어떤 위젯일까요?" />
```

- 여기서 텍스트뷰를 버튼으로 변경하려면 'TextView'를 'Button'으로만 변경

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="나는 어떤 위젯일까요?" />
```

- 필요시 TextView의 하위 클래스인 EditText, RadioButton, CheckBox, ToggleButton 등으로 바꿈

03 버튼과 에디트텍스트

■ 버튼

```
java.lang.Object
└─ android.view.View
    └─ android.widget.TextView
        └─ android.widget.Button
```

버튼 계층도

- 버튼을 클릭하는 이벤트를 가장 많이 사용
- 일반적인 버튼의 XML 코드

```
<Button
    android:id="@+id/button1"
    android:text="확인" />
```

03 버튼과 에디트텍스트

- 버튼을 클릭했을 때 동작하는 Kotlin 코드를 세 단계로 작성
 - ① 버튼 변수 선언
 - `var mybutton : Button`
 - ② 변수에 버튼 위젯 대입
 - `mybutton = findViewById<Button>(R.id.button1)`
 - ③ 버튼을 클릭할 때 동작하는 람다식 정의
 - `mybutton.setOnClickListener { // 동작 내용을 이 부분에 }`
- 세 단계는 대부분의 위젯(라디오버튼, 이미지버튼, 체크박스, 토글버튼 등)에서 거의 동일하게 사용됨

03 버튼과 에디트텍스트

■ 에디트텍스트

```
java.lang.Object
└─ android.view.View
    └─ android.widget.TextView
        └─ android.widget.EditText
```

에디트텍스트 계층도

- 값을 입력받은 후 해당 값을 Kotlin 코드에서 가져와 사용하는 용도로 많이 쓰임
- 일반적인 에디트텍스트의 XML 코드

```
<EditText
    android:id="@+id/edittext1" />
```

03 버튼과 에디트텍스트

- 에디트텍스트도 변수를 선언하고 이 변수에 해당 id 값을 넣은 후에 접근함
 - ① 에디트텍스트 변수 선언
 - `var myEdit : EditText`
 - ② 변수에 에디트텍스트 위젯 대입
 - `myEdit = findViewById<EditText>(R.id.edittext)`
 - ③ 에디트텍스트에 입력된 값 가져오기
 - 주로 버튼 클릭 이벤트 람다식 안에 넣음
 - `var myStr : String = myEdit.getText().toString()`

03 버튼과 에디트텍스트

■ <실습 4-1> 초간단 계산기 앱 만들기

- 가장 기본적인 위젯인 텍스트뷰, 에디트텍스트, 버튼을 이용한 앱 만들기
- 두 정수를 입력하고 버튼을 누르면 계산 결과가 나오는 계산기

■ 안드로이드 프로젝트 생성

- (1) 새 프로젝트를 만들기
 - 프로젝트 이름: 'Project4_1'
 - 패키지 이름: 'com.cookandroid. project4_1'
 - 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름

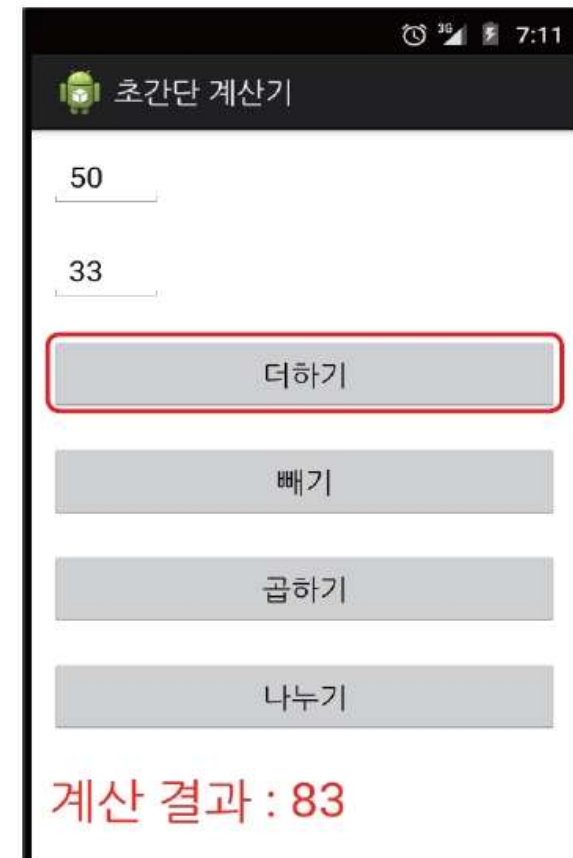


그림 4-6 초간단 계산기 앱 결과 화면

03 버튼과 에디트텍스트

- 화면 디자인 및 편집
 - (2) Project Tree에서 [app]-[res]-[layout]-[activity_main.xml]을 열고, 아래쪽의 [Text] 탭을 클릭하여 화면 코딩
 - 에디트텍스트 2개, 버튼 4개, 텍스트뷰 1개를 생성
 - 각 위젯에 layout_margin을 적절히 지정(예: 10dp)
 - 결과를 보여줄 TextView는 색상을 빨간색으로, 글자 크기를 30dp로 함
 - 각 위젯의 id는 위에서부터 차례로 Edit1, Edit2, BtnAdd, BtnSub, BtnMul, BtnDiv, TextResult로 함

03 버튼과 에디트텍스트

예제 4-19 activity_main.xml

```

1  <LinearLayout>
2      <EditText
3          android:id="@+id/Edit1"
4          android:layout_width="wrap_content"
5          android:layout_height="wrap_content"
6          android:layout_margin="10dp"
7          android:hint="숫자1" />
8      ~~~ 생략(에디트텍스트 1개) ~~~
9      <Button
10         android:id="@+id/BtnAdd"
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         android:layout_margin="10dp"
14         android:text="더하기" />
15     ~~~ 생략(버튼 3개) ~~~
16     <TextView
17         android:id="@+id/TextResult"
18         android:layout_width="wrap_content"
19         android:layout_height="wrap_content"
20         android:textSize="30dp"
21         android:textColor="#FF0000"
22         android:layout_margin="10dp"
23         android:text="계산 결과 : " />
24 </LinearLayout>

```

숫자1

숫자2

더하기

빼기

곱하기

나누기

계산 결과 :

03 버튼과 에디트텍스트

- Kotlin 코드 작성 및 수정
 - (3) Project Tree에서 [app]-[java]-[패키지 이름]-[MainActivity]를 열고, 다음과 같은 변수를 선언
 - activity_main.xml의 7개 위젯에 대응할 위젯 변수 7개
 - 입력될 2개 문자열을 저장할 문자열 변수 2개
 - 계산 결과를 저장할 정수 변수 1개
 - 이 변수들이 파일 내의 모든 클래스에서 사용되도록 전역변수로 선언
 - 메인 클래스인 MainActivity 바로 아래에 선언

예제 4-20 Kotlin 코드 1

```

1  ~~~ 생략(import 문) ~~~
2  class MainActivity : AppCompatActivity() {
3      lateinit var edit1 : EditText; lateinit var edit2 : EditText
4      lateinit var btnAdd : Button;   lateinit var btnSub : Button
5      lateinit var btnMul : Button;   lateinit var btnDiv : Button
6      lateinit var textResult : TextView
7      lateinit var num1 : String;      lateinit var num2 : String
8      var result : Int? = null
9
10     override fun onCreate(savedInstanceState: Bundle?) {
11         ~~~ 생략 ~~~

```


03 버튼과 에디트텍스트

- (4) 메인 메소드인 onCreate() 내부를 코딩
 - 먼저 <더하기>에 대한 부분을 코딩
 - 에디트텍스트 2개를 변수에 대입
 - 버튼(더하기) 1개를 변수에 대입
 - 텍스트뷰 1개를 변수에 대입

예제 4-21 Kotlin 코드 2

```
1  override fun onCreate(savedInstanceState: Bundle?) {  
2      super.onCreate(savedInstanceState)  
3      setContentView(R.layout.activity_main)  
4      title = "초간단 계산기"  
5  
6      edit1 = findViewById<EditText>(R.id.Edit1)  
7      edit2 = findViewById<EditText>(R.id.Edit2)  
8      btnAdd = findViewById<Button>(R.id.BtnAdd)  
9      textResult = findViewById<TextView>(R.id.TextResult)  
10 }
```

03 버튼과 에디트텍스트

- (5) 다음으로 <더하기> 버튼을 클릭했을 때 동작하는 클래스를 정의
 - 역시 onCreate() 메소드 안에([예제 4-21]의 9행 다음에) 코딩

예제 4-22 Kotlin 코드 3

```
1 btnAdd.setOnTouchListener { view, motionEvent ->
2     num1 = edit1.text.toString()
3     num2 = edit2.text.toString()
4     result = Integer.parseInt(num1) + Integer.parseInt(num2)
5     textResult.text = "계산 결과 : " + result.toString()
6     false
7 }
```

03 버튼과 에디트텍스트

- 프로젝트 실행 및 결과 확인
 - (6) Android Studio 메뉴의 [Run As]-[Run 'app']을 선택하거나 [Run 'app'] 아이콘을 클릭 하여 프로젝트를 실행
- [반복] Kotlin 코드 작성 및 수정
 - (7) 빼기, 곱하기, 나누기 코드를 직접 완성해 보기
 - 최종적으로 완성된 onCreate()의 완전한 코드는 다음과 같음

예제 4-23 완성된 Kotlin 코드

```
1  override fun onCreate(savedInstanceState: Bundle?) {
2      super.onCreate(savedInstanceState)
3      setContentView(R.layout.activity_main)
4      title = "초간단 계산기"
5
6      edit1 = findViewById<EditText>(R.id.Edit1)
7      edit2 = findViewById<EditText>(R.id.Edit2)
8      btnAdd= findViewById<Button>(R.id.BtnAdd)
9      btnSub= findViewById<Button>(R.id.BtnSub)
10     btnMul= findViewById<Button>(R.id.BtnMul)
11     btnDiv= findViewById<Button>(R.id.BtnDiv)
12
13     textResult= findViewById<TextView>(R.id.TextResult)
14 }
```

03 버튼과 에디트텍스트

```
15 btnAdd.setOnTouchListener { view, motionEvent ->
16     num1 = edit1.text.toString()
17     num2 = edit2.text.toString()
18     result = Integer.parseInt(num1) + Integer.parseInt(num2)
19     textResult.text= "계산 결과 : "+ result.toString()
20     false
21 }
22
23 btnSub.setOnTouchListener { view, motionEvent ->
24     num1 = edit1.text.toString()
25     num2 = edit2.text.toString()
26     result = Integer.parseInt(num1) - Integer.parseInt(num2)
27     textResult.text = "계산 결과 : "+ result.toString()
28     false
29 }
30
31 btnMul.setOnTouchListener { view, motionEvent ->
32     num1 = edit1.text.toString()
33     num2 = edit2.text.toString()
34     result = Integer.parseInt(num1) * Integer.parseInt(num2)
35     textResult.text = "계산 결과 : "+ result.toString()
36     false
37 }
38
39 btnDiv.setOnTouchListener { view, motionEvent ->
40     num1 = edit1.text.toString()
41     num2 = edit2.text.toString()
42     result = Integer.parseInt(num1) / Integer.parseInt(num2)
43     textResult.text = "계산 결과 : "+ result.toString()
44     false
45 }
46 }
```

03 버튼과 에디트텍스트

▶ 직접 풀어보기 4-3

[실습 4-1]을 다음과 같이 수정하라.

- 터치가 아닌 클릭으로 변경한다.
- 나머지값을 구하는 버튼을 추가한다.
- 값을 입력하지 않고 버튼을 클릭할 때 오류 메시지를 토스트 메시지로 나타낸다.
- 실숫값을 계산한다.
- 0으로 나누면 토스트 메시지를 나타내고 계산하지 않는다.



그림 4-7 개선된 계산기

01 컴파운드버튼

컴파운드버튼 계층도

```
java.lang.Object
└─ android.view.View
    └─ android.widget.TextView
        └─ android.widget.Button
            └─ android.widget.CompoundButton
                └─ android.widget.CheckBox
                └─ android.widget.RadioButton
                └─ android.widget.Switch
                └─ android.widget.ToggleButton
```

- CompoundButton 클래스
 - Button 클래스의 하위 클래스로 체크박스, 라디오버튼, 스위치, 토글버튼의 상위 클래스
 - 이 네 가지는 공통적으로 체크 또는 언체크 상태가 될 수 있음
 - 실제로 비슷한 형태를 띠지만 용도는 조금씩 다름.

01 컴파운드버튼

■ 체크박스

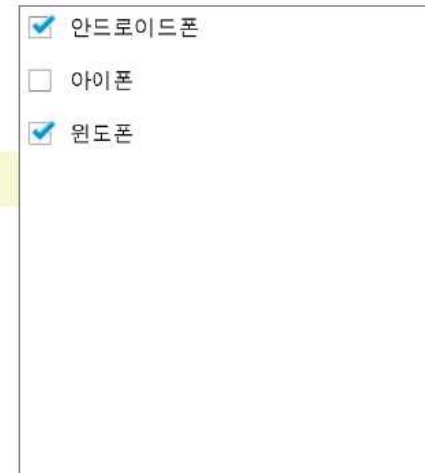
- 체크박스는 클릭할 때마다 상태가 체크, 언체크로 바뀜
- 여러 개의 체크박스가 있어도 서로 독립적으로 동작한다는 특징이 있어 여러 개를 동시에 체크할 수 있음

예제 4-24 체크박스의 XML 코드

```

1 <CheckBox
2     android:id="@+id/android"
3     android:text="안드로이드폰"
4     android:checked="true"/>
5 <CheckBox
6     android:id="@+id/iphone"
7     android:text="아이폰" />
8 <CheckBox
9     android:id="@+id/window"
10    android:text="윈도폰"
11    android:checked="true" />

```



☒ 안드로이드폰
☐ 아이폰
☒ 윈도폰

01 컴파운드버튼

- 체크와 언체크가 바뀌었을 때 Kotlin 코드의 처리 절차
 - ① 체크박스 변수 선언
 - `var mycheck : CheckBox`
 - ② 변수에 체크박스 위젯 대입
 - `mycheck = findViewById<CheckBox>(R.id.android)`
 - ③ 체크박스가 변경될 때 동작하는 람다식 정의
 - `mycheck.setOnCheckedChangeListener { compoundButton, b ->
// 동작 내용을 이 부분에 코딩
}`

01 컴파운드버튼

■ 스위치와 토글버튼

- 스위치와 토글버튼은 모양만 조금 다를 뿐 용도는 거의 동일함
- 스위치의 주 용도는 온/오프 상태 표시
 - XML 속성이나 관련 메소드는 모두 체크박스와 동일하게 사용 가능

예제 4-25 스위치와 토글버튼의 XML 코드

```

1 <Switch
2     android:checked="true" />
3 <Switch
4     android:checked="false" />
5 <ToggleButton
6     android:checked="true" />
7 <ToggleButton
8     android:checked="false" />

```



- checked 속성은 true와 false에 따라서 색상과 글자가 다르게 표현

01 컴파운드버튼

■ 라디오버튼과 라디오그룹

■ 라디오버튼

- XML 속성이나 메소드가 체크박스와 거의 동일하지만 용도가 다름
- 여러 개 중 하나만 선택해야 하는 경우에 사용
 - 그러나 라디오버튼만 여러 개 나열하면 클릭하는 것마다 모두 중복 선택되므로 라디오그룹과 함께 사용해야 함
 - 각 라디오버튼의 id 속성이 꼭 있어야 함
 - » id 속성이 없으면 해당 라디오버튼이 계속 선택된 것으로 지정되어 해제되지 않음

■ 라디오그룹

- ViewGroup-LinearLayout의 하위 클래스로 존재함
- TextView 하위의 위젯들과는 성격이 조금 다름
- 라디오그룹에서 가끔 사용되는 메소드인 clearCheck()
 - 해당 라디오그룹 안에 체크된 것을 모두 해제함

01 컴파운드버튼

예제 4-26 라디오그룹과 라디오버튼의 XML 코드

```

1 <RadioGroup
2     android:id="@+id/rGroup1" >
3     <RadioButton
4         android:text="남성" />
5     <RadioButton
6         android:text="여성" />
7 </RadioGroup>

```



- 2개의 라디오버튼을 1~7행의 라디오그룹으로 묶었기 때문에 이 라디오그룹 안의 모든 라디오 버튼은 한 번에 하나씩만 선택됨

02 이미지뷰와 이미지버튼

■ 이미지뷰

- 그림을 출력하는 위젯
- 그림을 넣거나 화면을 화려하게 구성할 때 사용함
 - 이미지뷰에 보여줄 그림 파일은 일반적으로 프로젝트의 [res]-[drawable] 폴더에 있어야 함
 - 접근은 XML에서 "@drawable/그림 아이디" 형식으로 함

```
java.lang.Object
└─ android.view.View
    └─ android.widget.ImageView
        └─ android.widget.ImageButton
```

이미지뷰 계층도

02 이미지뷰와 이미지버튼

- 이미지뷰와 이미지버튼의 XML 속성
 - src
 - 이미지의 경로를 나타냄
 - maxHeight/maxWidth
 - 이미지의 크기를 지정함
 - scaleType
 - 이미지의 확대/축소 방식을 지정
 - scaleType의 속성: matrix, fitXY, fitStart, fitEnd, center 등
지정한 값에 따라 이미지를 확대/축소하는 방식이 결정됨

02 이미지뷰와 이미지버튼

- 이미지뷰와 이미지버튼의 XML 속성
 - 사용하려는 그림 파일이 [res]-[drawable] 폴더에 있어야 함
 - 파일 포맷
 - png, jpg, gif를 지원하지만 png나 jpg를 권장함
 - [res]-[mipmap]
 - 같은 이름의 앱 아이콘 이미지(ic_launcher.png)가 디폴트로 들어 있음
 - 각각은 같은 이미지이지만 해상도가 mdpi(48×48), hdpi(72×72), xhdpi(96×96), xxhdpi(144×144), xxxhdpi(192× 192) 등으로 다름
 - XML 파일에서는 @mipmap/ic_launcher로, Kotlin 코드에서는 R.mipmap.ic_launcher로 사용됨
 - 이 앱 아이콘인 ic_launcher를 화면에 출력하면 안드로이드 운영체제가 현재 안드로이드폰의 해상도에 적절한 이미지를 알아서 선택함

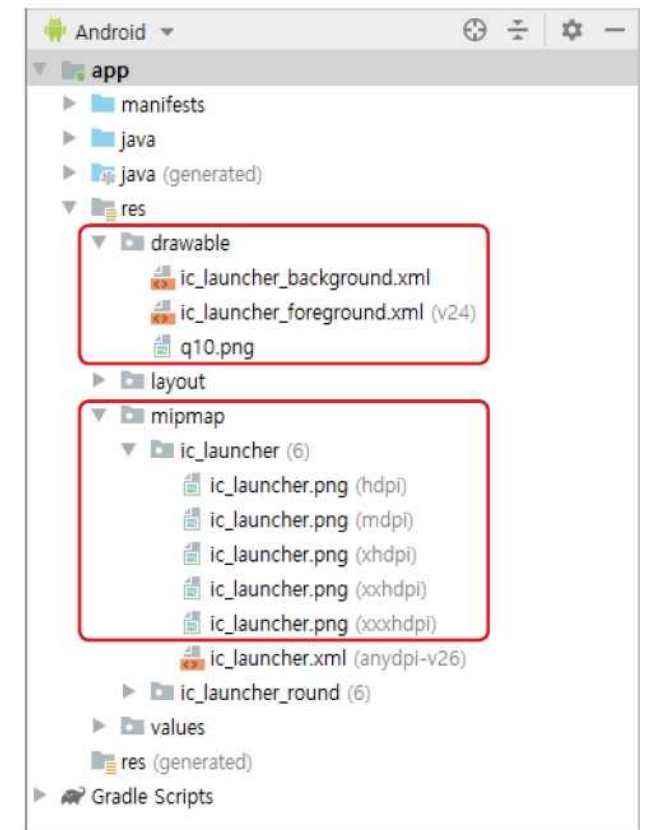
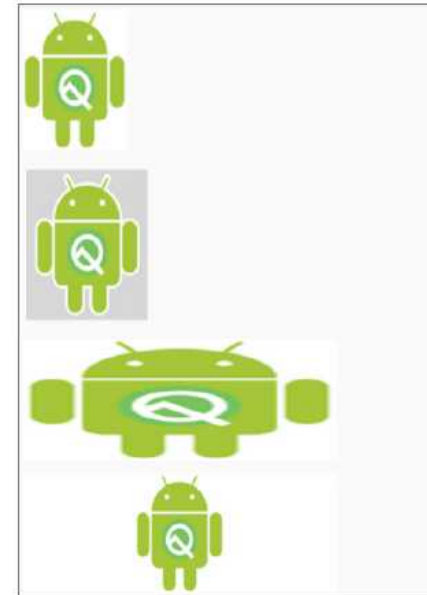


그림 4-8 drawable 폴더와 mipmap 폴더

02 이미지뷰와 이미지버튼

예제 4-27 이미지뷰와 이미지버튼의 XML 코드

```
1 <ImageView
2     android:src="@drawable/q10.png" />
3 <ImageButton
4     android:src="@drawable/q10.png" />
5 <ImageView
6     android:layout_width="300dp"
7     android:layout_height="100dp"
8     android:scaleType="fitXY"
9     android:src="@drawable/q10.png" />
10 <ImageView
11     android:layout_width="300dp"
12     android:layout_height="100dp"
13     android:scaleType="fitCenter"
14     android:src="@drawable/q10.png" />
```



02 이미지뷰와 이미지버튼

■ <실습 4-2> 애완동물 사진 보기 앱 만들기

- 애완동물의 사진을 출력하는 앱 만들기
- '시작함'에 체크하면 좋아하는 애완동물 세 가지 중에서 하나를 선택하라는 내용이 나옴
- 선택 후에 <선택 완료> 버튼을 클릭하면 해당 애완동물의 이미지가 나타남

■ 안드로이드 프로젝트 생성

- (1) 새 프로젝트 만들기
 - 프로젝트 이름 : 'Project4_2'
 - 패키지 이름 : 'com.cookandroid. project4_2'
 - 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름

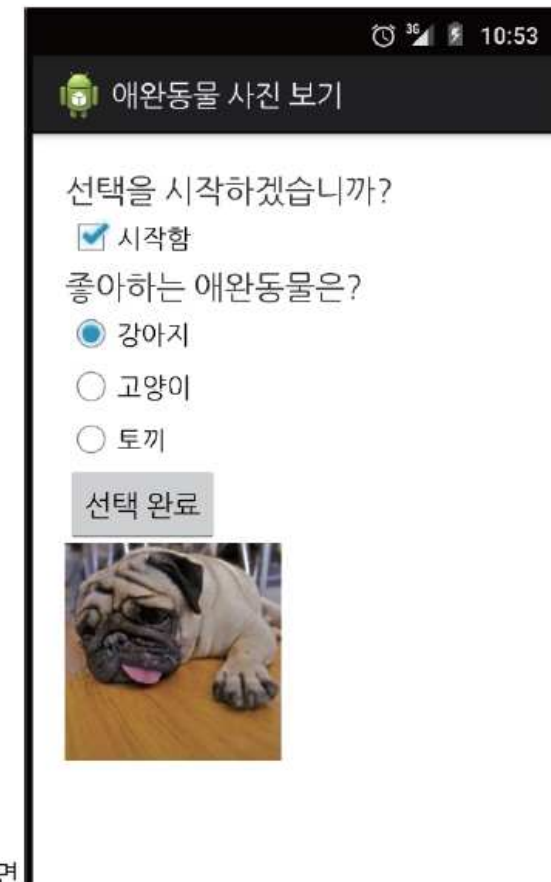


그림 4-9 애완동물 사진 보기 앱 결과 화면

02 이미지뷰와 이미지버튼

- 화면 디자인 및 편집
 - (2) 이번 프로젝트에서는 3개의 그림을 사용함
 - 프로젝트의 [res]-[drawable] 폴더에 강아지, 고양이, 토끼 그림 파일 복사
 - Windows 탐색기에서 그림 파일을 drawable 폴더에 복사/붙여넣기 한 후, [Choose Destination Directory] 창이 나오면 위쪽의 drawable 폴더를 선택
 - [Copy] 창에서 기본값으로 두고 <OK>를 클릭하면 해당 그림 파일이 복사됨

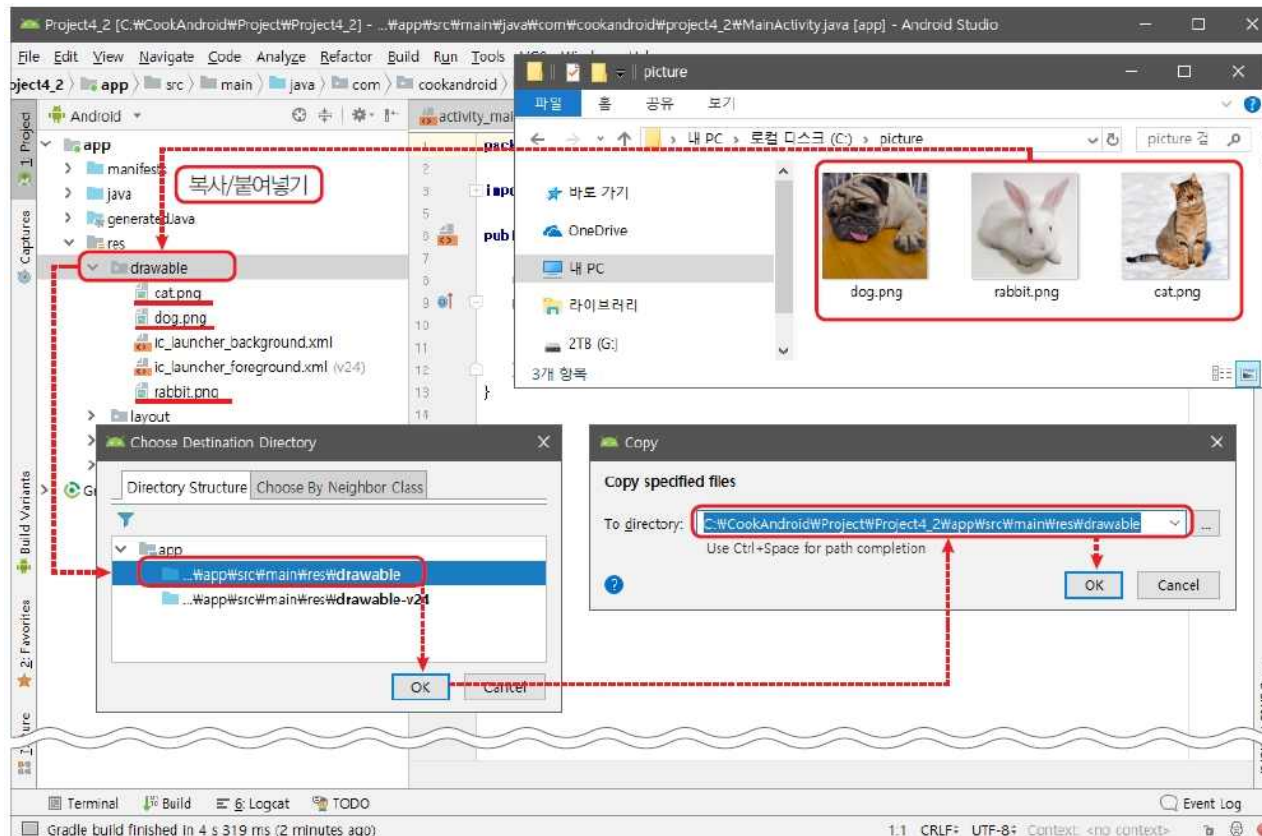


그림 4-10 그림 파일 복사

02 이미지뷰와 이미지버튼

- 화면 디자인 및 편집
 - (3) [app]-[res]-[layout]-[activity_main.xml]을 열고, 아래쪽의 [Text] 탭을 클릭하여 화면을 코딩
 - TextView, CheckBox, TextView, RadioGroup, RadioButton 3개, Button, ImageView의 차례로 만들
 - 레이아웃에 padding을 적절히 지정함
 - 맨 위의 TextView와 CheckBox를 제외하고 나머지 위젯은 visibility 속성을 invisible로 지정
 - 각 위젯의 id는 위에서부터 Text1, ChkAgree, Text2, Rgroup1, RdoDog, RdoCat, RdoRabbit, BtnOK, ImgPet로 함

02 이미지뷰와 이미지버튼

예제 4-28 activity_main.xml 코드

```

1  <TextView
2      android:id="@+id/Text1"
3      android:text="선택을 시작하겠습니까?" />
4  <CheckBox
5      android:id="@+id/ChkAgree"
6      android:text="시작함" />
7  <TextView
8      android:text="좋아하는 애완동물은?"
9      android:visibility="invisible" />
10 <RadioGroup
11     android:id="@+id/Rgroup1"
12     android:visibility="invisible" >
13     <RadioButton
14         android:id="@+id/RdoDog"
15         android:text="강아지" />
16     ~~~ 생략(라디오버튼 2개) ~~~
17 </RadioGroup>
18 <Button
19     android:id="@+id/BtnOK"
20     android:text="선택 완료"
21     android:visibility="invisible" />
22 <ImageView
23     android:id="@+id/ImgPet"
24     android:visibility="invisible" />

```

선택을 시작하겠습니까?

☐ 시작함

02 이미지뷰와 이미지버튼

- Kotlin 코드 작성 및 수정
 - (4) [app]-[java]-[패키지 이름]-[MainActivity]를 열기
 - (5) 다음과 같은 변수를 전역변수로 선언
 - activity_main.xml의 9개 위젯에 대응할 위젯 변수 9개

예제 4-29 Kotlin 코드 1

```
1  ~~~ 생략(import 문) ~~~
2  class MainActivity : AppCompatActivity() {
3      lateinit var text1 : TextView
4      lateinit var text2 : TextView
5      lateinit var chkAgree : CheckBox
6      lateinit var rGroup1 : RadioGroup
7      lateinit var rdoDog : RadioButton
8      lateinit var rdoCat : RadioButton
9      lateinit var rdoRabbit : RadioButton
10     lateinit var btnOK : Button
11     lateinit var imgPet : ImageView
12
13     override fun onCreate(savedInstanceState: Bundle?) {
14         ~~~ 생략 ~~~
```

02 이미지뷰와 이미지버튼

- (6) 각 위젯을 변수에 대입함
 - onCreate() 메소드 안에서 처리

예제 4-30 Kotlin 코드 2

```
1  override fun onCreate(savedInstanceState: Bundle?) {  
2      super.onCreate(savedInstanceState)  
3      setContentView(R.layout.activity_main)  
4      title = "애완동물 사진 보기"  
5  
6      text1 = findViewById<TextView>(R.id.Text1)  
7      chkAgree = findViewById<CheckBox>(R.id.ChkAgree)  
8  
9      text2 = findViewById<TextView>(R.id.Text2)  
10     rGroup1 = findViewById<RadioGroup>(R.id.Rgroup1)  
11     rdoDog = findViewById<RadioButton>(R.id.RdoDog)  
12     rdoCat = findViewById<RadioButton>(R.id.RdoCat)  
13     rdoRabbit = findViewById<RadioButton>(R.id.RdoRabbit)  
14  
15     btnOK = findViewById<Button>(R.id.BtnOK)  
16     imgPet = findViewById<ImageView>(R.id.ImgPet)  
17 }
```

02 이미지뷰와 이미지버튼

- (7) '시작함' 체크박스를 체크, 언체크할 때마다 동작하는 람다식을 onCreate() 내부에 정의함

예제 4-31 Kotlin 코드 3

```
1  chkAgree.setOnCheckedChangeListener { compoundButton, b ->
2
3      if (chkAgree.isChecked == true) {
4          text2.visibility = android.view.View.VISIBLE
5          rGroup1.visibility = android.view.View.VISIBLE
6          btnOK.visibility = android.view.View.VISIBLE
7          imgPet.visibility = android.view.View.VISIBLE
8      } else {
9          text2.visibility = android.view.View.INVISIBLE
10         rGroup1.visibility = android.view.View.INVISIBLE
11         btnOK.visibility = android.view.View.INVISIBLE
12         imgPet.visibility = android.view.View.INVISIBLE
13     }
14 }
```


02 이미지뷰와 이미지버튼

- (8) <선택 완료>를 클릭하면 동작하는 람다식을 정의
 - 역시 onCreate() 내부에 정의

예제 4-32 Kotlin 코드 4

```
1 btnOK.setOnClickListener {  
2     when (rGroup1.checkedRadioButtonId) {  
3         R.id.RdoDog -> imgPet.setImageResource(R.drawable.dog)  
4         R.id.RdoCat -> imgPet.setImageResource(R.drawable.cat)  
5         R.id.RdoRabbit -> imgPet.setImageResource(R.drawable.rabbit)  
6         else -> Toast.makeText(applicationContext,  
7             "동물 먼저 선택하세요", Toast.LENGTH_SHORT).show()  
8     }  
9 }
```

- 4 프로젝트 실행 및 결과 확인
 - (9) 프로젝트를 실행하여 결과를 확인

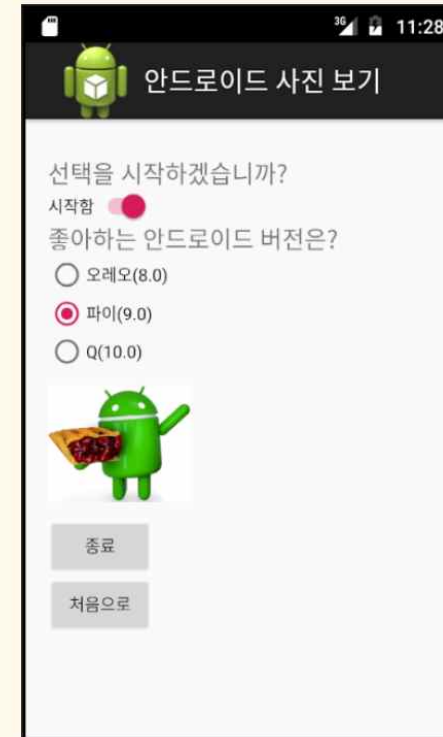
02 이미지뷰와 이미지버튼

▶ 직접 풀어보기 4-4

[실습 4-2]를 다음과 같이 수정하라.

- ‘좋아하는 안드로이드 버전은?’으로 질문을 변경한다.
- ‘시작함’을 Switch로 변경한다.
- <선택 완료>를 없애고, 라디오버튼을 선택할 때마다 즉시 해당 이미지가 나오도록 변경한다.
- 마지막에 <종료>와 <처음으로>를 추가한다. <종료>를 클릭하면 응용 프로그램이 완전히 종료되게 하고, <처음으로>를 클릭하면 다시 초기화되고 처음 화면이 나오게 한다.

그림 4-11 안드로이드 사진 보기 앱



Thank You !