Chapter 04

기본 위젯 익히기

학습목표

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

01 뷰의 개요

앱을 실행하면 화면에 무엇인가 나와야 사용자가 인식하고 사용할 수 있다. 이렇게 앱 실행화면을 구성하는 요소를 통칭하여 뷰(View)라고 한다. 텍스트뷰, 버튼, 라디오버튼, 이미지 등이 모두 뷰에 속한다.

1 뷰와 뷰그룹

안드로이드 화면에서 실제로 사용되는 것들은 모두 View라는 클래스의 상속을 받는다. 즉 버튼, 라디오버튼, 이미지 등 모두 View 클래스의 자손 클래스인 것이다. 뷰 클래스는 다른 말로 '위젯'이라고도 하며, 쉽게 이야기하면 화면에서는 버튼을 버튼 위젯, 실제 코드에서는 버튼 클래스라고 부르는 식이다. 그리고 다른 위젯을 담을 수 있는 위젯을 특별히 레이아웃이라고 하며, 레이아웃은 ViewGroup이라는 클래스 아래 존재한다. (레이아웃도 크게 보면 위젯에 포함된다.)

다른 프로그래밍 언어에서 컨트롤(Control)이라 부르는 버튼(Button), 텍스트뷰(TextView), 에디트텍스트(EditText), 라디오버튼(RadioButton), 이미지뷰(ImageView) 등을 안드로이드에서는 위젯이라고 부른다.

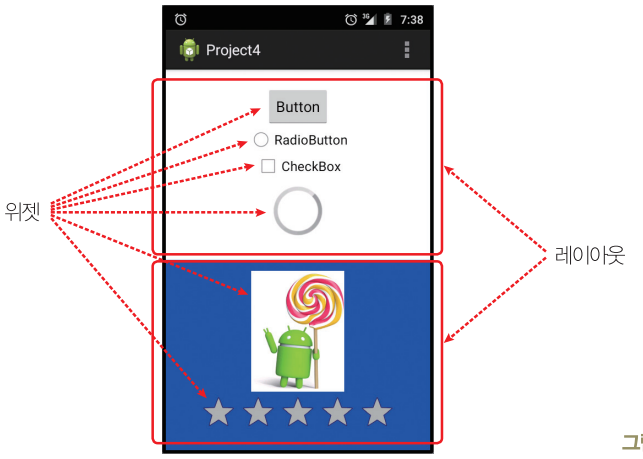


그림 4-1 위젯과 레이아웃

그런데 위젯은 단독으로 존재하지 않으며, 위젯을 담아 배치하는 틀이 바로 '레이아웃'이다. 레이아웃은 위젯들을 포함하는 컨테이너 역할을 하므로 눈에 보이는 개념은 아니다. 따라서 이 책에서는 버튼, 텍스트뷰, 체크박스 등 눈에 보이는 요소들을 '위젯', 위젯을 담는 틀은 '레이아웃'으로 구분하여 부른다.

TIP 위젯은 큰 의미로 View 클래스 하위의 모든 클래스를 지칭하기도 하며, 작은 의미로 레이아웃 이외의 클래스를 지칭하기도 한다. 이 책에서도 종종 상황에 맞게 부를 것이다.

View 클래스 계층도

안드로이드에서 View 클래스의 상속을 받은 클래스(위젯)의 계층도의 일부를 나타냈다.

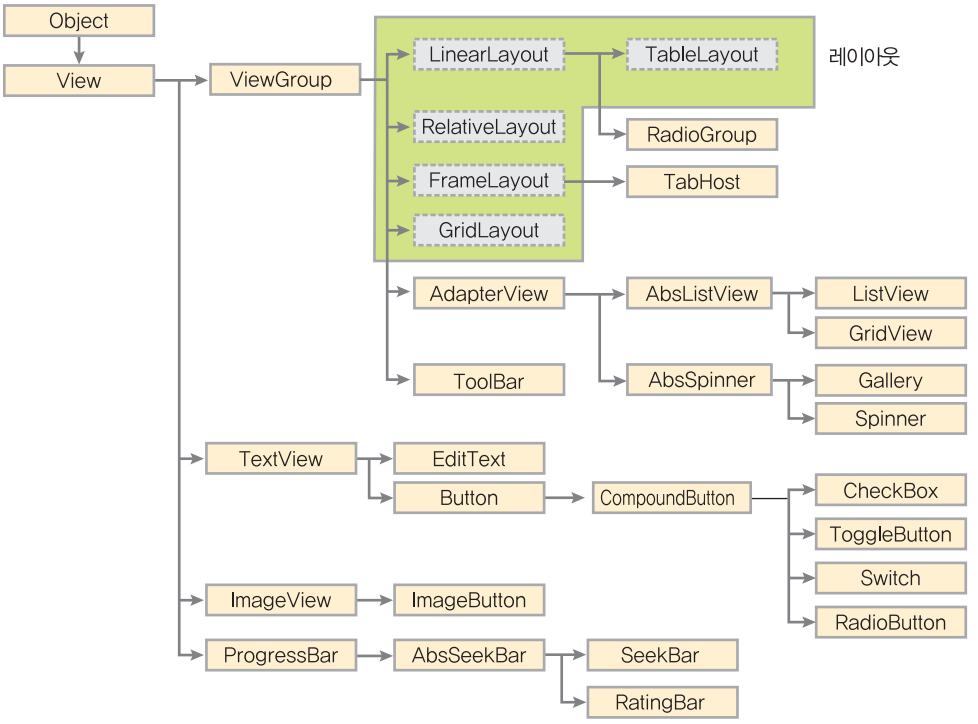


그림 4–2 안드로이드 View 클래스 계층도

그림을 보면 최상위에 Object(java.lang.Object) 클래스가 있고 이를 상속받은 View 클래스가 존재한다. 안드로이드 화면에 보이게 될 모든 위젯은 View 하위에 존재한다. 레이아웃은 ViewGroup을 상속받은 LinearLayout, RelativeLayout, FrameLayout, GridLayout, TableLayout을 지칭한다. 또한 레이아웃이라고 부르지는 않지만 다른 뷰를 포함하는

ListView, GridView, TabHost, Gallery 등을 뷰 컨테이너(View Container)라고 한다. 뷰 컨테이너도 ViewGroup 클래스에서 상속을 받는다.

TIP ToolBar는 API 21(Lollipop, Android 5.0)부터 지원되며, TabHost 및 ActionBar와 비슷한 용도로 사용된다. 6장에서 다시 언급하겠다.

TIP 앞으로 종종 위젯과 그에 해당하는 클래스라는 용어가 나오게 될 것이다. 버튼 위젯과 Button 클래스라는 용어가 나온다면, 버튼 위젯은 1개의 Button 클래스에 대응되므로 같은 개념으로 이해하면 된다. 문맥상 버튼 위젯은 화면에 보이는 개체를 지칭할 때, Button 클래스는 XML 속성이나 Java 코드와 연계될 때 사용된다.

Button이 위젯, ListView가 뷰 컨테이너, LinearLayout이 레이아웃이라는 것을 외울 필요는 없으며 앞으로 자연스럽게 익히게 될 것이다. 위젯이 어떤 클래스를 상속했는지는 알아야 할 때가 있는데, 상속 관계는 계속 학습해가면서 알 수 있다.

가장 자주 사용하게 될 위젯인 버튼의 속성을 살펴보자. Android Studio에서 XML 파일의 〈Button 안에 android:을 입력하면 버튼의 XML 속성 목록이 나오는데 100개가 훨씬 넘는다.

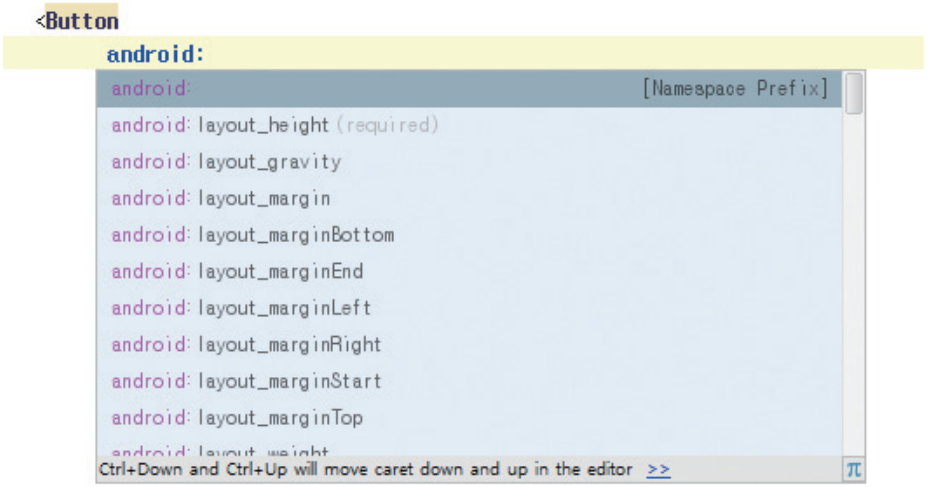


그림 4-3 Button의 XML 속성

Button에 있는 XML 속성은 거의 없고 대개 상위 클래스인 TextView나 View에서 상속받는다. 이는 http://developer.android.com/reference에서 자세히 확인할 수 있다. 왼쪽 상단의 패키지 이름에서 android.widget을 클릭하고, 하단에서 Button 클래스를 클릭하면 상속관계도와 설명을 확인할 수 있다.

앞에서 얘기했지만 Button에는 XML 속성에 대한 내용이 거의 없다. Button의 XML 속성은 대부분 TextView나 View 클레스에서 상속받기 때문이다. 이럴 때는 TextView를 클릭해서 확

인해봐야 한다. 앞으로 안드로이드에서 사용되는 클래스에 대한 내용은 이러한 방식으로 찾아볼 수 있다. 특히 화면을 구성하는 뷰들은 대부분 android.widget 패키지에 포함되므로 이 부분을 살펴보면 된다. 바로 찾으려면 화면 오른쪽 위에 클래스 이름을 입력한다.

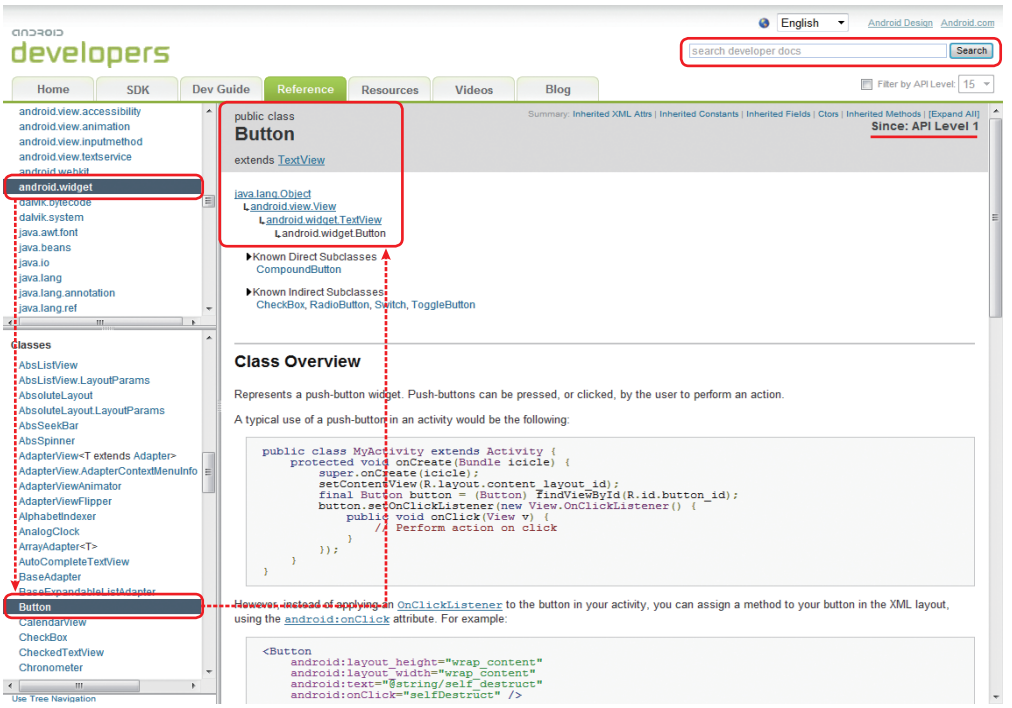


그림 4-4 클래스의 상속 관계를 찾는 방법

▶ **직접 풀어보기 4-1**

http://developer.android.com/reference에서 ScrollView, DigitalClock, CalendarView의 각 클래스 상속 관계를 찾아보자.

2 View 클래스의 XML 속성

안드로이드 계층도를 보면 View 클래스가 모든 위젯의 부모 클래스임을 알 수 있다. 그러므로 위젯과 레이아웃 등은 모두 View 클래스의 속성과 메소드를 상속받는다고 생각하면 된다. View 클래스의 중요한 속성을 파악해놓으면 하위 클래스의 이해도 쉬워진다. 이번에는 View 클래스나 해당 클래스의 상위 클래스에서 상속받은 XML 속성을 살펴보겠다.

View 클래스의 XML 속성은 수십 개가 넘으므로 자주 사용할 것들만 살펴보자. 다음은 앞에서 배운 Button의 기본 형태이다. View 클래스로부터 상속을 받았다.

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

버튼의 id, layout_width 등 몇 가지 기본적인 XML 속성이 표현되었다. 이 속성들에 대해 하나씩 살펴보자. 여기에는 표현되지 않았지만 종종 추가로 필요한 속성을 알아볼 것이다.

TIP 2장에서도 언급한 적이 있지만 XML 속성은 모든 xmlns에 지정된 "android:" 이후에 붙게 된다. 그러므로 앞으로 XML 속성을 지칭할 때 "android:" 부분은 생략하고 얘기할 것이다.

id 속성

id 속성은 모든 위젯의 아이디를 나타내며 Java 코드에서 버튼 등의 위젯에 접근할 때 id 속성에 지정한 아이디를 사용한다. 일반적으로 id 속성은 위젯에 아이디를 새로 부여하는 개념이므로 "@+id/" 형식으로 지정한다. / 다음에는 새로 지정할 id를 적어준다. 그러므로 android:id="@+id/btn1"의 의미는 버튼 위젯의 아이디를 btn1로 부여한 것이다. 2장에서 잠깐 살펴보았지만 위젯에 접근하기 위해 Java 코드에서 다음과 같은 형식을 사용했다.

```
위젯 변수 = (위젯 형) findViewById(R.id.위젯id);
```

앞의 버튼 예제는 Java 코드에서 다음과 같은 접근 방식을 사용할 수 있다.

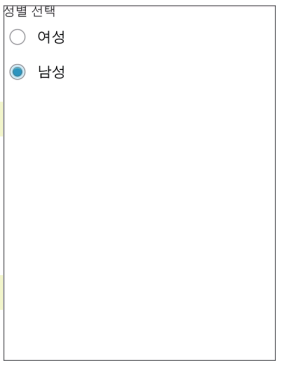
```
Button  button1;
button1 = (Button) findViewById(R.id.btn1);
```

Button, RadioButton, CheckBox 등의 위젯은 일반적으로 클릭 혹은 터치했을 때 어떤 동작을 하기 위한 것이므로 id를 지정해준다. 하지만 클릭이나 터치를 해도 아무 동작이 필요 없는 글자(텍스트뷰)나 배경 이미지(이미지뷰) 등은 굳이 id 속성을 지정하지 않아도 된다. 예를 들어 다

음 화면에서 "성별 선택" 텍스트뷰는 클릭 또는 터치할 일이 없다. 결국 Java 코드에서 id 속성을 사용할 일이 없기 때문에 4행을 삭제해도 무방하다. 반면 라디오버튼은 클릭이나 터치하면 Java 코드에서 어떤 동작을 해야 하므로 9행과 14행이 필요하다.

예제 4-1 id 속성 XML 코드

```
1  <LinearLayout   ~~~ 중간 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>
```



layout_width, layout_height 속성

layout_width, layout_height 속성은 모든 위젯에 필수로 들어가야 하므로 확실히 정리하고 넘어가자. 이 둘은 위젯의 폭과 높이를 나타내는데, match_parent와 wrap_content 값으로 설정할 수 있다.

TIP match_parent와 fill_parent는 동일한 용어이다. Android 2.1 이하 버전에서 동작하는 앱을 작성하려면 match_parent 대신 fill_parent를 사용해야 한다. match_parent는 Android 2.2부터 인식한다.

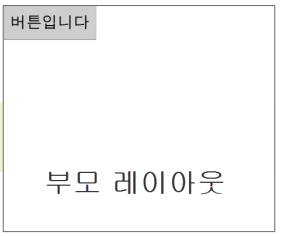
match_parent는 이름 그대로 자신의 부모(대개는 레이아웃)에 폭이나 높이를 맞춘다는 의미이고 wrap_content는 자신의 폭이나 높이를 자신 안의 글자가 꼭 들어갈 정도로만 설정한다는 의미이다.

TIP 버튼 예에서 그 안에 들어갈 것이 글자이기 때문에 글자가 꼭 들어갈 정도라고 표현했다. 하지만 다른 위젯은 글자 외의 것이 들어갈 수 있다. 예를 들어 이미지뷰는 이미지가 들어가므로 wrap_content는 이미지가 꼭 들어갈 정도로 폭이나 높이가 설정된다.

다음 예제를 보면 쉽게 이해할 수 있을 것이다.

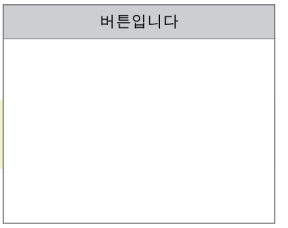
예제 4-2 layout_width, layout_height 속성 XML 코드 1

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



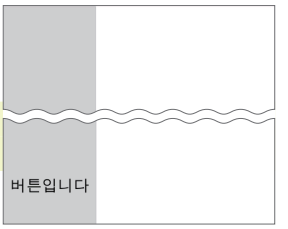
예제 4-3 layout_width, layout_height 속성 XML 코드 2

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



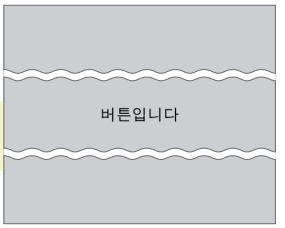
예제 4-4 layout_width, layout_height 속성 XML 코드 3

```
1  <LinearLayout
2      ~~~~ 중간 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 속성 XML 코드 4

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

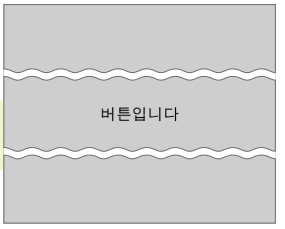


예제를 보면 알 수 있듯이 layout_width 속성에 wrap_content 값을 사용하면 그 안의 글자인 "버튼입니다"가 폭에 꼭 맞게 버튼 크기가 정해지고, match_parent(또는 fill_parent) 값을 사용하면 "버튼입니다" 글자와 관계 없이 버튼을 싸고 있는 부모(리니어레이아웃)의 폭에 꽉 차도록 버튼의 크기가 정해진다. layout_height도 버튼의 높이에 대해 적용된다.

값을 숫자로 직접 지정할 수도 있는데 이때는 단위에 주의해야 한다. 가장 단순한 것이 px(PiXel) 단위이다. 필자의 AVD는 480x800의 해상도이므로 폭은 480px로, 높이는 화면 위쪽 제목 부분을 제외한 690px 정도로 지정했더니 match_parent로 지정한 것처럼 버튼이 화면에 꽉 찼다.

예제 4-6 layout_width, layout_height 속성 XML 코드 5

```
1 <LinearLayout
2      ~~~~ 중간 생략 ~~~~  >
3      <Button
4          android:layout_width="480px"
5          android:layout_height="690px"
6          android:text="버튼입니다" />
7 </LinearLayout>
```



background 속성

background 속성은 위젯의 색상을 주로 #RRGGBB 값으로 지정한다. 각 값은 Red, Green, Blue를 의미하며 RR, GG, BB 위치는 16진수 00~FF로 표현할 수 있다. 예를 들어 빨간색은 #FF0000로, 파란색은 #0000FF로 지정한다. 적절히 조합하면 필요한 색을 만들 수도 있다.

저자 한마디

위젯의 크기 지정 단위

위젯의 크기를 in, mm, pt, dp, sp 등의 단위로 직접 지정할 수도 있다. 주로 사용되는 것은 dp(=dip), sp, px이다. dp(Density-independent Pixel)는 화면 밀도에 독립적으로 사용되는 단위인데, 이를 사용하면 실행되는 안드로이드폰의 해상도가 다르더라도 결과는 같은 비율로 출력되는 효과를 낼 수 있다.

즉 dp로 지정하면 WVGA(480x800)든 QVGA(240x320)든 WXGA(1280x800)든 화면에 비슷한 비율로 버튼이 적절히 표현될 것이다. 그래서 안드로이드폰의 해상도와 관계없이 잘 동작되는 화면을 만들 때 dp 단위를 많이 사용한다. 폰트를 지정할 때는 sp(Scaled Pixel) 단위를 많이 사용한다.

TIP 각 값의 조합은 16진수 색상표에 따라 조합된다. "#7878FF"는 보라색 계열, "#20B2AA"는 청록색 계열이다.

다음 예제는 레이아웃은 빨간색, 버튼은 초록색으로 표현했다. 2행에서 레이아웃의 색을 빨간색으로 지정하고, 7행에서 레이아웃에 있는 버튼을 초록색으로 지정하였다.

예제 4-7 background 속성 XML 코드

```
1  <LinearLayout
2      android:background="#ff0000"
3      ~~~~ 중간 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>
```

저자 한마디

책의 XML 코드 표기(★ 주의)

텍스트뷰 1개와 버튼 1개가 있는 기본적인 activity_main.xml 코드의 전체는 다음과 같다.

예제 4-8 전체 XML 코드

```
1  <LinearLayout  xmlns:android="http://schemas.android.com/apk/res/
   android"
2      xmlns:tools="http://schemas.android.com/tools"
3      android:layout_width="fill_parent"
4      android:layout_height="fill_parent"
5      android:orientation="vertical" >
6
```

```
7       <TextView
8           android:layout_width="fill_parent"
9           android:layout_height="wrap_content"
10          android:text="@string/hello_world" />
11
12      <Button
13          android:id="@+id/button1"
14          android:layout_width="wrap_content"
15          android:layout_height="wrap_content"
16          android:text="Button" />
17
18  </LinearLayout>
```

1행과 2행의 xmlns 이후는 항상 동일한 내용이 나오고 3행, 4행, 8행, 9행, 14행, 15행에 나오는 layout_width와 layout_height는 모든 위젯의 필수 요소로 자주 등장할 것이다. 5행의 orientation 속성도 거의 고정되어 나온다. 따라서 이 책에서는 이러한 반복적인 속성들은 꼭 표현할 필요가 있을 때를 제외하고는 생략할 것이다. 즉 앞으로 다음과 같이 필수나 반복되는 코드를 생략하고 간략하게 표현하기로 한다.

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 실제 확인해야 할 코드들이 명확하게 보인다. 앞으로는 이런 식으로 XML 코드를 표기할 것인데, 실제로는 전체 코드를 다 표현해야 동작함을 잊지 말자. 주의할 점은 5행의 orientation 속성을 생략하면 LinearLayout은 horizontal이 된다는 것이다.

이 책의 대부분 화면은 vertical 정렬이므로 예제에는 생략되어 있어도 5행을 잊지 말고 써주자. 5장에서 다시 설명하겠다. 참고로 생략하지 않은 모든 소스코드는 한빛 사이트(http://www.hanbit.co.kr/exam/4179/)에 등록해놓았다.

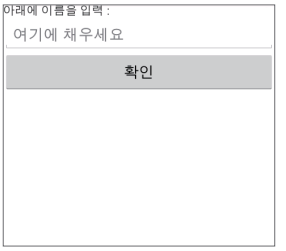
이외에 색상을 지정하는 방법으로 #AARRGGBB 방식을 사용할 수도 있다. AA는 알파 값으로 투명도를 지정한다. 00~FF를 지정할 수 있는데 00은 완전 투명을, FF는 완전 불투명을 나타낸다. 즉 #FF0000FF는 불투명 파란색이고 #000000FF는 완전 투명이므로 색상이 나오지 나오지 않는다. #770000FF는 반투명 파란색이다.

padding과 layout_margin 속성

padding 속성을 사용하여 위젯의 경계선으로부터 위젯 안의 요소가 떨어지도록 설정할 수 있다. 기본적으로는 레이아웃 내에 버튼, 텍스트뷰 등을 여러 개 뒀을 때 레이아웃의 경계선에 딱 붙어서 표현된다. 그런데 padding 속성을 사용하면 레이아웃의 경계와 위젯 사이에 여백을 둘 수 있다. 우선 텍스트뷰, 에디트텍스트, 버튼이 있는 XML 파일을 살펴보자.

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>
```

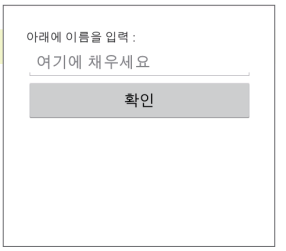


세 위젯이 레이아웃의 경계선에 딱 붙어 답답해 보인다. 레이아웃에 padding 속성을 사용해보자.

TIP padding은 상하좌우 모두에 지정하는 속성이다. 따로 지정하고 싶으면 paddingTop, paddingBottom, paddingLeft, paddingRight를 이용한다.

예제 4-11 padding 속성 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>
```

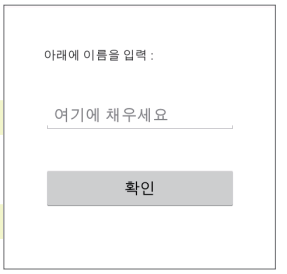


LinearLayout에 지정된 padding 속성 때문에 그 안의 요소들이 경계선에서 30dp만큼 떨어져 출력되었다. 예에서는 LinearLayout에 padding을 설정했지만, Button에 padding을 설정하면 버튼 안의 글자가 버튼의 경계선에서 일정 간격 떨어져서 표현된다. 이와 달리 위젯과 위젯 사이에 여유를 두고 싶다면 layout_margin 속성을 사용한다.

TIP layout_margin은 상하좌우 모두에 지정하는 속성이다. 따로 지정하고 싶으면 layout_marginTop, layout_marginBottom, layout_marginLeft, layout_marginRight를 이용한다.

예제 4-12 layout_margin 속성 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>
```



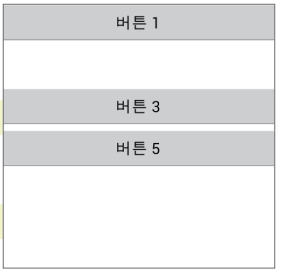
정리하면 padding 속성은 자신의 내부에 들어 있는 위젯과 자신의 경계선과의 간격을 지정하고, layout_margin은 자신과 부모 레이아웃이나 위젯과의 간격이나 주위의 다른 위젯과의 간격을 지정한다. 따라서 layout_margin은 각 위젯의 속성으로 지정해야 한다.

visibility 속성

visibility 속성은 위젯을 보일 것인지 여부를 결정한다. 세 가지 값을 지정할 수 있는데 디폴트인 visible은 보이는 상태, invisible과 gone은 안 보이는 상태이다. invisible과 gone의 차이는, invisible은 보이지 않을 뿐 원래의 자리를 계속 유지하지만 gone은 그 자리까지 아예 내놓는다는 것이다. 다음 예제를 보면 이해하기 쉬울 것이다.

예제 4-13 visibility 속성 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" />
```



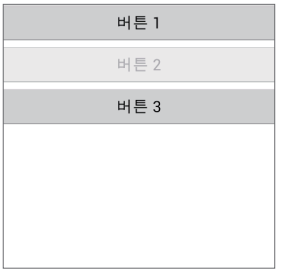
visibility 속성은 XML보다 Java 코드에서 상황에 따라서 동적으로 필요한 버튼 등을 보이거나 안 보이도록 하는 경우에 주로 활용된다.

enabled, clickable 속성

위젯의 동작 여부는 enabled 속성으로, 클릭이나 터치가 가능하도록 하는 것은 clickable 속성으로 지정한다. 값은 true와 false로 지정하며 디폴트 값은 true이다. 두 속성도 XML보다는 Java 코드에서 주로 사용한다.

예제 4-14 enabled 및 clickable 속성 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" />
```



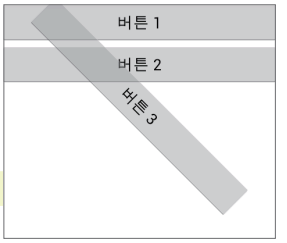
rotation 속성

rotation은 위젯을 회전시켜서 출력한다. 값은 각도(degree)로 지정한다. 특별한 화면을 만들 때 가끔 사용하곤 한다.

TIP rotation 속성은 안드로이드 3.0(API 11)부터 지원된다.

예제 4-15 rotation 속성 XML 코드

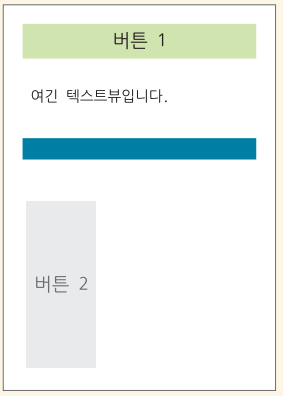
```
1  <Button
2      android:text="버튼 1" />
3  <Button
4      android:text="버튼 2" />
5  <Button
6      android:rotation="45"
7      android:text="버튼 3" />
```



지금까지 많이 사용되는 XML 속성을 살펴보았다. 이외에도 상당히 많지만 전부 나열하는 것보다는 필요할 때마다 익히도록 하겠다.

▶ **직접 풀어보기 4-2**

다음 그림과 같은 화면을 XML로 코딩하자. 버튼, 텍스트뷰, 에디트텍스트, 버튼을 차례로 지정하고 앞에서 배운 다양한 속성을 지정해본다.



02 기본 위젯 다루기

이번에는 안드로이드에서 가장 기본적으로 사용되는 텍스트뷰, 버튼, 에디트텍스트에 대해서 학습한다. XML로 화면을 구성한 후 실제 Java 코드로 동작하는 기법을 익힌다.

1 텍스트뷰

TextView 계층도

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

[그림 4-2]를 보면 텍스트뷰(TextView)는 View 클래스 바로 다음에 위치하며 다양한 위젯들이 그 하위에 존재한다. 텍스트뷰 하위(에디트텍스트, 버튼, 체크박스 등)는 모두 텍스트뷰로부터 상속을 받기 때문에 텍스트뷰의 속성을 잘 이해하면 다른 위젯의 속성도 이해하기 쉬울 것이다.

▶ text 속성

텍스트뷰에 나타나는 문자열을 표현한다. "문자열" 형식으로 값을 직접 입력하거나 "@string/변수명" 형식으로 지정한 후에 strings.xml 파일에 지정할 수 있다.

TIP 안드로이드 매뉴얼에서는 activity_main.xml 파일에 문자열을 직접 입력하는 것보다 "@string/변수명" 형식의 사용을 권장한다. 이는 string.xml의 내용만 수정하면 다른 언어로도 앱을 쉽게 변환할 수 있기 때문이다. 이 책에서는 편의상 activity_main.xml의 위젯에 문자열을 직접 입력하는 방법을 위주로 사용했다.

▶ textColor 속성

글자의 색상을 지정하며, background 속성처럼 값은 #RRGGBB나 #AARRGGBB 형식이다.

▶ textSize 속성

글자의 크기를 dp, px, in, mm, sp 단위로 지정한다.

▶ typeface 속성

글자의 글꼴을 지정한다. 값으로 sans, serif, monospace를 설정할 수 있고 디폴트는 normal이다.

▶ textStyle 속성

글자의 스타일을 지정한다. 값으로 bold, italic, bold|italic을 설정할 수 있고 디폴트는 normal이다.

▶ singleLine 속성

글이 길어 줄이 넘어갈 경우 강제로 한 줄까지만 출력하고 문자열의 맨 뒤에 "…"를 표시한다. 값으로 true와 false를 설정할 수 있으며 디폴트는 false다.

TIP 이외에도 글자와 관련된 속성으로 textAllCaps, textAppearance, textColorHighlight, textColorHint, textColorLink, textIsSelectable, textScaleX 등이 있는데 활용도가 높지는 않다.

예제 4-16 글자 관련 속성 XML 코드

```
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="textStype 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" />
```

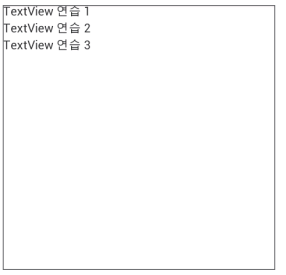
2 Java 코드로 XML 속성 설정

XML 속성은 View 클래스와 TextView 클래스에서 배운 것으로도 충분히 활용할 수 있다. 배우지 않은 것들도 거의 비슷한 형식이므로 쉽게 응용할 수 있을 것이다. 이번에는 activity_main.xml 파일에서 지정한 XML 속성을 Java 코드에서 설정하는 방법을 알아보겠다.

기본적인 텍스트뷰만 만들어놓고 id 속성과 text만 설정한 XML 파일은 다음과 같다.

예제 4-17 텍스트뷰가 3개 있는 activity_main.xml

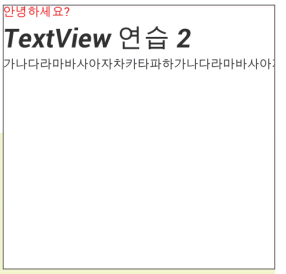
```
1  <TextView
2      android:text="TextView 연습 1"
3      android:id="@+id/textView1" />
4  <TextView
5      android:text="TextView 연습 2"
6      android:id="@+id/textView2" />
7  <TextView
8      android:text="TextView 연습 3"
9      android:id="@+id/textView3" />
```



Java 코드를 다음과 같이 설정하여 화면에 적용한다.

예제 4-18 텍스트 속성을 변경하는 Java 코드

```
1  public void onCreate(Bundle savedInstanceState) {
2    super.onCreate(savedInstanceState);
3    setContentView(R.layout.main);
4
5    TextView tv1, tv2, tv3;
6    tv1 = (TextView) findViewById(R.id.textView1);
7    tv2 = (TextView) findViewById(R.id.textView2);
8    tv3 = (TextView) findViewById(R.id.textView3);
9
10   tv1.setText("안녕하세요?");
11   tv1.setTextColor(Color.RED);
12   tv2.setTextSize(30);
13   tv2.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD_ITALIC);
14   tv3.setText("가나다라마바사아자차카타파하가나다라마바사아자차카타파하");
15   tv3.setSingleLine();
16 }
```



1~3 자동 생성되는 코드이다. 3행은 R.layout.activity_main(즉, activity_main.xml) 파일을 화면에 출력해준다는 의미인데 추후에 변경해야 할 필요가 있으므로 기억해두자.

5 텍스트뷰 변수를 선언한다. TextView에 빨간줄이 표시되면 Ctrl+Shift+O를 누른다.

6~8 activity_main.xml에서 생성한 3개의 텍스트뷰를 tv1~tv3 변수에 대입한다.

10 첫 번째 텍스트뷰의 문자열을 변경한다. XML 속성의 text와 동일한 역할을 한다.

11 첫 번째 텍스트뷰의 색상을 빨간색으로 지정한다. 색상은 android.graphics.Color 클래스에 상수로 지정되어 있는데 RED, BLACK, GREEN 등으로 사용한다. XML 속성의 textColor 속성과 동일한 역할을 한다.

12 두 번째 텍스트뷰에 sp 단위의 폰트 크기를 지정한다. XML 속성의 textSize="30sp"와 동일한 역할을 한다.

13 두 번째 텍스트 뷰에 XML 속성의 typeFace와 textStyle 속성을 동시에 지정하는 메소드다. android.graphics.Typeface 클래스에 상수로 지정되어 있다.

14~15 세 번째 텍스트뷰에 긴 글을 대입하고, XML 속성에서 singleLine="true" 효과를 줬다.

위 예제를 보면 XML 파일에서 설정하는 내용의 대부분이 메소드로 제공되어 Java 코드에서도 XML의 설정이 가능함을 확인할 수 있다. 이러한 XML 속성과 Java 메소드를 일일이 외울 필요는 없지만, 예제를 이해함으로써 응용할 수 있을 것이다. 많이 사용되는 View 클래스 또는 TextView 클래스의 XML 속성과 메소드를 [표 4-1]에 정리했다.

TIP [표 4-1]의 XML 속성과 메소드는 앞으로 사용되는 많은 위젯들에서 거의 동일한 방식으로 사용된다. [그림 4-2]에서 View 클래스와 TextView 클래스의 상속을 받은 클래스가 많기 때문이다.

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

표 4-1 XML 속성과 관련 메소드(계속)

XML 속성	관련 메소드	비고
password	setTransformationMethod()	TextView 클래스
text	setText()	TextView 클래스
textColor	setTextColor()	TextView 클래스
textSize	setTextSize()	TextView 클래스

3 버튼과 에디트텍스트

버튼과 에디트텍스트는 사용자에게서 어떤 값을 입력받기 위한 가장 기본적인 위젯으로 활용도가 높다. 두 위젯은 View와 TextView 클래스를 상속받으므로 거의 비슷하게 사용할 수 있다.([그림 4-2] 참조)

간단한 텍스트뷰를 예로 들어보자.

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

여기서 텍스트뷰를 버튼으로 변경하려면 "TextView" 글자를 "Button"으로만 바꿔주면 된다.

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

필요하다면 TextView의 하위 클래스인 EditText, RadioButton, CheckBox, ToggleButton 등으로 바꿔도 된다.

버튼

Button 계층도

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

버튼(Button)에서는 버튼을 클릭하는 이벤트를 가장 많이 사용한다. 2장에서는 따라하기만 했으므로 버튼에서 가장 많이 사용하는 기능을 다시 요약해보자. 일반적인 버튼의 XML 코드는 다음과 같다.

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

이 버튼을 클릭했을 때 동작하는 Java 코드를 세 단계로 작성했다.

❶ 버튼 변수 선언

```
Button  mybutton;
```

❷ 변수에 버튼 위젯 대입

```
mybutton = (Button) findViewById(R.id.button1);
```

❸ 버튼 클릭할 때 동작하는 클래스 정의

```
mybutton.setOnClickListener( new View.OnClickListener() {
      public void onClick(View v) {
      // 동작할 내용을 코딩
   }
});
```

위 세 단계는 대부분의 위젯(라디오버튼, 이미지버튼, 체크박스, 토글버튼 등)에서 거의 동일하게 사용되므로 통째로 외워두면 좋다. 잠시 후 실습에서 이 내용을 그대로 활용해보자.

에디트텍스트

EditText 계층도

```
java.lang.Object
   ㄴ android.view.View
     ㄴ android.widget.TextView
       ㄴ android.widget.EditText
```

에디트텍스트는 값을 입력받은 후에 해당 값을 Java 코드에서 가져와서 사용하는 용도로 많이 쓰인다. Java 코드에서 에디트텍스트에 값을 입력하는 경우도 종종 있다. 일반적인 에디트텍스트의 XML 코드는 다음과 같다.

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

에디트텍스트도 변수를 선언하고 이 변수에 해당 id 값을 넣은 후에 접근한다. 다음과 같은 형식을 많이 사용한다.

❶ 에디트텍스트 변수 선언

```
EditText  myEdit;
```

❷ 변수에 에디트텍스트 위젯 대입

```
myEdit = (EditText) findViewById(R.id.edittext1);
```

❸ 에디트텍스트에 입력된 값을 가져오기 → 주로 버튼 클릭 이벤트 리스너 안에 넣음

```
String myStr = myEdit.getText().toString();
```

getText() 메소드는 에디트텍스트에 입력한 값을 반환하는데, 이를 문자열로 바꾸기 위해서 toString()을 함께 사용했다. 반환 값을 문자열로 변경할 때 가장 많이 사용하는 방식이므로 잘 기억해놓자. 잠시 후 실습에서 그대로 활용해보자.

실습 4-1 초간단 계산기 앱 만들기

가장 기본적인 위젯인 텍스트뷰, 에디트텍스트, 버튼을 이용해 앱을 만들어보자. 두 정수를 입력하고 해당 버튼을 누르면 계산 결과가 나오는 계산기다. 이번 실습을 통해서 XML 작성부터 Java 코드를 연계하는 완전한 프로젝트 작성법을 숙지할 수 있다. 생성할 화면은 [그림 4-5]와 같다.

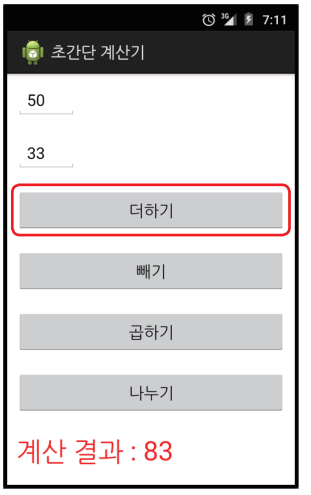


그림 4-5 초간단 계산기 결과 화면

1 안드로이드 프로젝트 생성

❶ 새 프로젝트를 만든다. 프로젝트 이름은 Project4_1로, 패키지 이름은 com.cookandroid.project4_1로 한다. 그 외 규칙은 [실습 2-4]의 ❶~❹ 과정을 따른다.

TIP 앞으로 작성하는 프로젝트는 대부분 "[그림 2-2] 안드로이드 프로젝트 개발 단계" ❶~❺를 따른다.

2 화면 디자인 및 편집

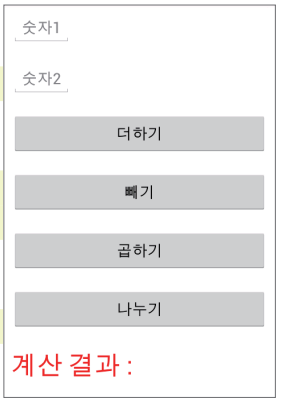
2 Android Studio의 Project Tree에서 [app]-[res]-[layout]-[activity_main.xml]을 더블클릭해서 열고, 아래쪽 [Text] 탭을 클릭해서 화면을 코딩한다. 화면 구성은 다음 규칙을 따른다.

- EditText 2개, Button 4개, TextView 1개를 생성한다.
- 각 위젯에 layout_margin을 적절히 지정한다(예 10dp).
- 결과를 보여줄 TextView는 색상을 빨간색으로, 글자 크기를 30dp로 한다.
- 각 위젯의 id는 위에서부터 차례로 Edit1, Edit2, BtnAdd, BtnSub, BtnMul, BtnDiv, TextResult로 한다.

TIP 이미 언급했지만 activity_main.xml이 기본적으로 RelativeLayout으로 되어 있다면 LinearLayout으로 고쳐준다. 또 LinearLayout에 orientation="vertical" 속성도 추가해준다. 앞으로 따로 언급하지 않아도 activity_main.xml는 이런 방식으로 고쳐야 한다.

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        ~~~~ 중간 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     ~~~~ 중간 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>
```



3, 10, 17 위젯에 id를 지정한다. 추후 Java 코드에서 "R.id.아이디" 형식으로 각 위젯에 접근하게 된다.

7 에디트텍스트에 약간 흐리게 글자를 표시해주는 기능을 하며, 실제로는 없는 것과 마찬가지다. 이 글자에 색상을 넣으려면 textColorHint 속성을 사용한다. 에디트텍스트에서 종종 사용된다.

6, 13, 22 각 위젯의 화면상 간격을 조절한다.

20~21 텍스트뷰의 글자 크기와 색상을 지정한다.

3 Java 코드 작성 및 수정

3 Android Studio의 Project Tree에서 [app]-[java]-[패키지 이름]-[MainActivity]를 더블클릭해서 연다.

4 다음 내용의 변수를 선언한다.

- 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 선언하자. 전역변수로 선언하려면 메인 클래스인 MainActivity 바로 아래에 두면 된다.

TIP 자동 생성된 Java 코드 중에 onCreateOptionMenu(), onOptionsItemSelected() 메소드가 있다면 사용하지 않으므로 삭제한다([그림 2-54] 참조). 앞으로는 별도로 언급하지 않겠다.

예제 4-20 Java 코드 1

```
1  ~~~ 중간생략 ~~~
2  public class MainActivity extends Activity {
3      EditText edit1, edit2;
4      Button btnAdd, btnSub, btnMul, btnDiv;
5      TextView textResult;
6      String num1, num2;
7      Integer result;
8
9       @Override
10      public void onCreate(Bundle savedInstanceState) {
11      ~~~ 중간생략 ~~~
```

3~5 7개의 위젯에 관계되는 변수를 선언한다.

6 에디트텍스트에 입력한 문자열 값을 저장할 변수

7 두 문자열을 계산한 결과를 저장할 변수

TIP 변수 선언 후에, Alt + Enter를 누르면 자동으로 import문이 추가된다.

5 메인 메소드인 onCreate() 내부를 코딩한다. 먼저 〈더하기〉에 대한 부분을 코딩하자.

- 에디트텍스트 2개를 변수에 대입한다.
- 버튼(더하기) 1개를 변수에 대입한다.
- 텍스트뷰 1개를 변수에 대입한다.

예제 4-21 Java 코드 2

```
1  public void onCreate(Bundle savedInstanceState) {
2      super.onCreate(savedInstanceState);
3      setContentView(R.layout.activity_main);
4      setTitle("초간단 계산기");
5
6      edit1 = (EditText) findViewById(R.id.Edit1);
7      edit2 = (EditText) findViewById(R.id.Edit2);
8      btnAdd = (Button) findViewById(R.id.BtnAdd);
9      textResult = (TextView) findViewById(R.id.TextResult);
10 }
```

4 화면의 제목 표시줄을 변경한다.

6~9 activity_main.xml에 생성한 위젯을 변수에 대입한다.

6 이번에는 더하기 버튼을 클릭했을 때 동작하는 클래스를 정의하자. 역시 onCreate() 메소드 안에([예제 4-21]의 9행 다음에) 코딩해야 한다.

- 버튼에 터치 이벤트 리스너를 정의한다.
- 터치 시에 동작하는 내용을 onTouch() 메소드 안에 코딩한다.

TIP 클릭(Click)과 터치(Touch)는 비슷하게 동작하며 작성법도 거의 동일하다. OnClickListener 대신에 OnTouchListener을 사용하면 된다.

예제 4-22 Java 코드 3

```
1  btnAdd.setOnTouchListener(new View.OnTouchListener() {
2      public boolean onTouch(View arg0, MotionEvent arg1) {
3          num1 = edit1.getText().toString();
4          num2 = edit2.getText().toString();
5          result = Integer.parseInt(num1) + Integer.parseInt(num2);
6          textResult.setText("계산 결과 : " + result.toString());
7          return false;
8      }
9  });
```

1 더하기 버튼을 터치했을 때 리스너를 설정한다.

2 onTouch() 메소드를 구현한다. 자동완성되는 부분이다.

3~4 에디트텍스트에 입력된 값을 num1, num2 변수에 대입한다.

5 num1, num2를 Interger.parseInt() 메소드를 사용해서 정수형으로 변환한 후 두 값을 더한다.

6 정수형 결과를 다시 문자열로 변경한 후 텍스트뷰에 setText()를 이용해서 대입한다.

7 false 값을 돌려준다. 자동완성되는 부분이다.

9 맨 뒤에 세미콜론을 입력해야 문장이 끝난다.

TIP Integer.parseInt() 메소드는 시스템에서 제공하는 Interger 클래스에 포함되어 있는 정적 메소드로, 문자열을 정수로 변경해준다.

4 프로젝트 실행 및 결과 확인

7 Android Studio 메뉴의 [Run As]-[Run 'app']을 선택하거나, [Run 'app'] 아이콘을 클릭해서 프로젝트를 실행한다. 적당한 두 값을 입력한 후 〈더하기〉를 터치(클릭)하면 [그림 4-5]의 화면이 나올 것이다.

[반복] 3 Java 코드 작성 및 수정

8 빼기, 곱하기, 나누기 코드를 직접 완성해보자. 최종 완성된 onCreate()의 완전한 코드는 다음과 같다.

예제 4-23 완성된 Java 코드

```
1  public void onCreate(Bundle savedInstanceState) {
2      super.onCreate(savedInstanceState);
3      setContentView(R.layout.activity_main);
4      setTitle("초간단 계산기");
5
6      edit1 = (EditText) findViewById(R.id.Edit1);
7      edit2 = (EditText) findViewById(R.id.Edit2);
8
9      btnAdd = (Button) findViewById(R.id.BtnAdd);
10     btnSub = (Button) findViewById(R.id.BtnSub);
11     btnMul = (Button) findViewById(R.id.BtnMul);
12     btnDiv = (Button) findViewById(R.id.BtnDiv);
13
14     textResult = (TextView) findViewById(R.id.TextResult);
15
16     btnAdd.setOnTouchListener(new View.OnTouchListener() {
17      public boolean onTouch(View arg0, MotionEvent arg1) {
18         num1 = edit1.getText().toString();
19         num2 = edit2.getText().toString();
20         result = Integer.parseInt(num1) + Integer.parseInt(num2);
21         textResult.setText("계산 결과 : " + result.toString());
22         return false;
23      }
24     });
25
26     btnSub.setOnTouchListener(new View.OnTouchListener() {
27      public boolean onTouch(View arg0, MotionEvent arg1) {
28         num1 = edit1.getText().toString();
29         num2 = edit2.getText().toString();
30         result = Integer.parseInt(num1) - Integer.parseInt(num2);
31         textResult.setText("계산 결과 : " + result.toString());
32         return false;
33      }
34     });
35
36     btnMul.setOnTouchListener(new View.OnTouchListener() {
37      public boolean onTouch(View arg0, MotionEvent arg1) {
38         num1 = edit1.getText().toString();
```

```
39          num2 = edit2.getText().toString();
40          result = Integer.parseInt(num1) * Integer.parseInt(num2);
41          textResult.setText("계산 결과 : " + result.toString());
42          return false;
43       }
44      });
45
46      btnDiv.setOnTouchListener(new View.OnTouchListener() {
47       public boolean onTouch(View arg0, MotionEvent arg1) {
48          num1 = edit1.getText().toString();
49          num2 = edit2.getText().toString();
50          result = Integer.parseInt(num1) / Integer.parseInt(num2);
51          textResult.setText("계산 결과 : " + result.toString());
52          return false;
53       }
54      });
55 }
```

6~14 위젯을 변수에 대입한다.

26~34 빼기 버튼을 터치했을 때 동작하는 리스너다.

36~14 곱하기 버튼을 터치했을 때 동작하는 리스너다.

46~54 나누기 버튼을 터치했을 때 동작하는 리스너다.

5 안드로이드 응용프로그램 개발 완료

9 간단한 계산기 프로젝트를 완료하였다.

저자 한마디 **리스너(Listener) 인터페이스**

버튼, 체크박스, 라디오버튼 등을 클릭(또는 터치)하여 어떤 동작이 일어나도록 할 때는 해당 위젯에 리스너(Listener) 인터페이스를 설정하여 처리하도록 해야 한다. 예를 들어 OnTouchListener 인터페이스에는 onTouch( ) 메소드가 포함되어 있는데, onTouch( ) 메소드를 구현하고 터치할 때 필요한 내용을 코딩하는 것이 바로 프로그래머가 할 일이다. [예제 4-22]가 가장 일반적인 리스너 인터페이스 사용 형식이다.

▶ **직접 풀어보기 4-3**

[실습 4-1]에 다음과 같이 기능을 추가하거나 변경해보자.

- 터치(Touch)가 아닌 클릭(Click)으로 변경
- 나머지 값 구하기 버튼 추가
- 값을 입력하지 않고, 버튼을 클릭할 때 오류 메시지를 토스트로 나타내기
- 실수 값 계산하기
- 0으로 나누면 토스트 메시지를 나타내고 계산하지 않기

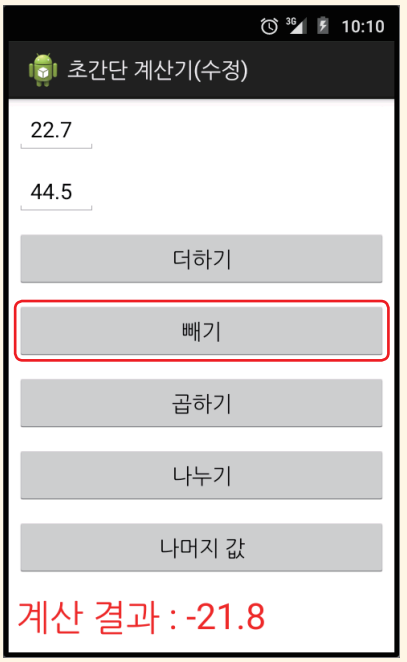


그림 4-6 개선된 계산기

03 기본 위젯 활용하기

[그림 4-2]를 보면 CompoundButton 클래스의 하위에 체크박스, 토글버튼, 라디오버튼을 확인할 수 있는데, 이 세 가지는 응용프로그램 개발에서 다양한 용도로 활용된다.

1 컴파운드버튼

CompoundButton 클래스는 Button 클래스의 하위 클래스로, 체크박스(CheckBox), 라디오버튼(RadioButton), 스위치(Switch), 토글버튼(ToggleButton)의 상위 클래스이다. 이 네 가지는 공통적으로 체크 또는 언체크 상태가 될 수 있다. 실제로 비슷한 형태를 띠지만 용도는 조금씩 다르다.

CompundButton 계층도

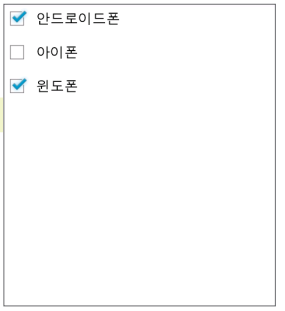
```
java.lang.Object
    └ android.view.View
        └ android.widget.TextView
            └ android.widget.Button
                └ android.widget.CompundButton
                    └ android.widget.CheckBox
                    └ android.widget.RadioButton
                    └ android.widget.Switch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

체크박스

체크박스(CheckBox)는 클릭할 때마다 상태가 체크와 언체크로 바뀐다. 여러 개의 체크박스가 있어도 서로 독립적으로 동작한다는 특징이 있어 여러 개를 동시에 체크할 수 있다.

예제 4-24 CheckBox의 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" />
```



4행과 11행에서 checked 속성이 true로 되어 있다. 이처럼 동시에 여러 개를 선택할 수 있다. "지금까지 사용해본 적이 있는 스마트폰을 모두 고르시오"처럼 선택을 여러 개 할 수 있는 경우에 사용하면 적절하다.

Java 코드에서는 강제로 체크를 켜거나 끄는 setChecked(), 체크가 되었는지를 확인하는 isChecked(), 체크 상태를 반대로 바꿔주는 toggle() 등의 메소드를 많이 사용한다. Button에서 클릭 이벤트 발생 시 OnClickListener 리스너를 사용했듯이, 체크박스에는 체크 또는 언체크 이벤트 발생 시 OnCheckedChangeListener 리스너를 사용할 수 있다.

TIP 체크박스도 TextView 클래스의 하위 클래스이므로 OnClickListener, OnTouchListener 등의 리스너를 사용할 수 있다.

즉 체크와 언체크가 바뀌었을 때 Java 코드에서 다음과 같이 처리할 수 있다. Button 클릭과 절차가 거의 동일하다.

❶ 체크박스 변수 선언

```
CheckBox mycheck;
```

❷ 변수에 체크박스 위젯 대입

```
mycheck = (CheckBox) findViewById(R.id.android);
```

❸ 체크박스가 변경될 때 동작하는 클래스 정의

```
mycheck.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
   public void onCheckedChanged(CompoundButton arg0, boolean arg1) {
```

```
            // 이 부분에 동작할 내용을 코딩
        }
    });
```

체크박스의 실제 적용 사례는 잠시 후 실습에서 더 살펴보자.

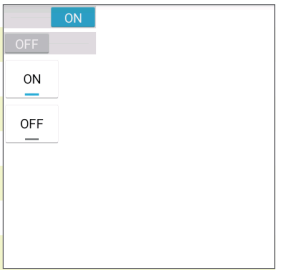
스위치, 토글버튼

스위치(Switch)와 토글버튼(ToggleButton)은 모양만 조금 다를 뿐 용도는 거의 동일하다. 스위치의 주 용도는 On/Off 상태 표시이다. 사실 체크박스로도 스위치나 토글버튼처럼 사용할 수 있지만 모양이 조금 다르므로 상황에 맞는 것을 고르는 것이 좋다. XML 속성이나 관련 메소드는 모두 체크박스와 동일하게 사용할 수 있다.

TIP 스위치는 안드로이드 4.0(API 14)부터 지원되는 위젯이다.

예제 4-25 Switch와 ToggleButton의 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 모양 및 글자가 다르게 표현된다.

라디오버튼과 라디오그룹

라디오버튼(RadioButton)은 체크박스와 XML 속성이나 메소드가 거의 동일하지만 용도는 다르다. 성별을 선택하는 것처럼 여러 개 중 하나만 선택해야 하는 경우에 사용한다. 그러나 라디오버튼만 여러 개 나열하면 클릭하는 것마다 모두 중복 선택되므로 라디오그룹(RadioGroup)과 함께 사용해야 한다.

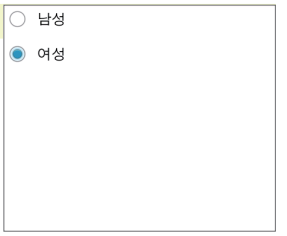
라디오그룹은 [그림 4-2]와 같이 ViewGroup-LinearLayout의 하위 클래스로 존재하며 지금 사용하고 있는 TextView 하위의 위젯들과는 성격이 조금 다르다. 대부분 RadioGroup은

RadioButton을 묶어주는 역할만 하므로 다음의 예제만 이해하면 된다. RadioGroup에서 가끔 사용되는 메소드는 clearCheck()인데, 이는 해당 라디오그룹 안에 체크된 것을 모두 해제해 준다.

두 개의 라디오버튼을 1행~7행의 라디오그룹으로 묶어주었다. 그러므로 이 라디오그룹 안의 모든 라디오버튼은 한 번에 하나씩만 선택된다. 라디오버튼 사용에서 주의할 점은 각 라디오버튼의 id 속성이 꼭 있어야 한다는 것이다. id 속성이 없으면 해당 라디오버튼이 계속 선택된 것으로 지정되며 선택 해제가 되지 않는다.

예제 4-26 RadioGroup과 RadioButton의 XML 코드

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



2 이미지뷰와 이미지버튼

이미지뷰(ImageView)는 그림을 출력하는 위젯으로 그림이 필요하거나 화면을 화려하게 구성할 때 사용한다. 이미지뷰에 보여줄 그림 파일은 일반적으로 프로젝트의 [res]-[drawable]에 있어야 한다. 접근은 XML에서 "@drawable/그림 아이디" 형식으로 한다.

ImageView 계층도

```
java.lang.Object
    ㄴ android.view.View
        ㄴ android.widget.ImageView
            ㄴ android.widget.ImageButton
```

[그림 4-2]를 보면 ImageVew 클래스는 View 클래스에서 바로 상속받기 때문에 앞에서 배운 TextView의 하위 위젯들과 속성이 좀 다르다. 특히 이미지와 관련된 속성 및 메소드를 주의 깊게 볼 필요가 있다. ImageButton 클래스는 ImageView 클래스에서 상속받으며 거의 동일한

용도로 사용하지만 버튼처럼 클릭하는 용도로 사용한다. 이미지 버튼은 그림으로 표현된 예쁜 버튼을 만들 때 사용할 수 있다.

이미지뷰 및 이미지버튼의 XML 속성

이미지뷰 및 이미지버튼의 XML 속성은 이미지의 경로를 나타내는 src, 이미지의 크기를 지정하는 maxHeight/maxWidth, 이미지의 확대/축소 방식을 지정하는 scaleType 등이 있다. scaleType 속성으로 matrix, fitXY, fitStart, fitEnd, center 등 여덟 가지 값을 지정할 수 있는데, 지정한 값에 따라 이미지를 확대/축소하는 방식을 결정한다.

이미지를 사용하려면 먼저 그림 파일을 [res]-[drawable] 폴더에 복사해놓아야 한다. 파일 포맷은 png, jpg, gif를 지원하지만 png나 jpg를 권장한다. drawable 폴더는 xxhdpi, xhdpi, hdpi, mdpi 네 가지로 나뉘는데, drawable-xxhdpi 폴더에는 440dpi용 초초고해상도 이미지를, drawable-xhdpi 폴더에는 320dpi용 초고해상도 이미지를, drawable-hdpi 폴더에는 240dpi용 고해상도 이미지를, drawable-mdpi 폴더에는 160dpi용 중해상도 이미지를 저장하면 된다.

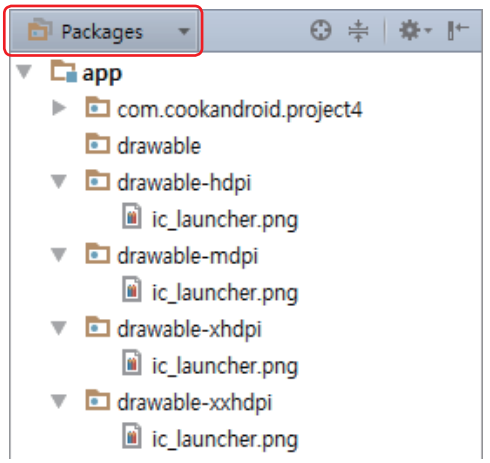


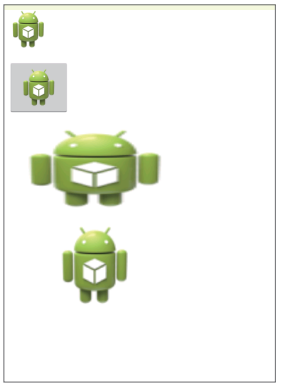
그림 4-7 한 이미지를 네 가지 해상도로 저장한 drawable 리소스(Packages 모드로 볼 때)

TIP 프로젝트를 생성하면 [res]-[drawable]에 같은 이름의 이미지(ic_launcher.png)가 디폴트로 들어 있는데, 각각은 같은 이미지이지만 해상도가 다르다. 모두 XML 파일에서 "@drawable/ic_launcher"로, Java 코드에서는 "R.drawable.ic_launcher"로 사용된다. "ic_launcher"를 화면에 출력하면 안드로이드 운영체제가 현재 안드로이드폰의 해상도에 적절한 이미지를 알아서 선택한다. 그러므로 상용 응용프로그램을 개발하는 경우라면 어떤 해상도에서도 출력이 잘 되도록 같은 이미지를 몇 가지 해상도로 만들어놓는 것이 좋다.

TIP 그림 파일을 [drawable] 폴더에 넣어 두는 간단한 방법도 있다.

예제 4-27 ImageView와 ImageButton의 XML 코드

```
1  <ImageView
2      android:src="@drawable/ic_launcher" />
3  <ImageButton
4      android:src="@drawable/ic_launcher" />
5  <ImageView
6      android:layout_width="200dp"
7      android:layout_height="100dp"
8      android:scaleType="fitXY"
9      android:src="@drawable/ic_launcher" />
10 <ImageView
11     android:layout_width="200dp"
12     android:layout_height="100dp"
13     android:scaleType="fitCenter"
14     android:src="@drawable/ic_launcher" />
```



1~2 기본적인 ImageView 형식이다. 3행에서 [res]-[drawable]의 이미지 id를 지정한다.

3~4 기본적인 ImageButton 형식이다.

6~7, 11~4 이미지뷰의 크기를 확장했다.

8, 13 이미지뷰의 이미지를 확장, 축소하는 방식을 지정했다. fitXY는 이미지뷰의 좌우에 꽉 맞춰서 이미지를 확장하고 fitCenter는 중앙에 맞춰서 확장하는 방식이다. 이외에도 matrix, fintStart, fitEnd, center, centerCrop, centerInside 방식이 있다.

이미지버튼은 이미지뷰와 거의 동일하지만, 버튼 형식이고 클릭이 가능하다. 물론 이미지뷰에도 클릭 이벤트 발생 시 버튼처럼 처리가 가능하다.

실습 4-2 좋아하는 애완동물 선택 앱 만들기

이번에는 보고 싶은 애완동물의 사진을 출력하는 앱을 만들어보자. "시작함"를 체크하면 좋아하는 애완동물 세 가지 중에서 하나를 선택하라는 내용이 나온다. 선택 후에 〈선택 완료〉 버튼을 클릭하면 해당 애완동물의 이미지가 나타난다.

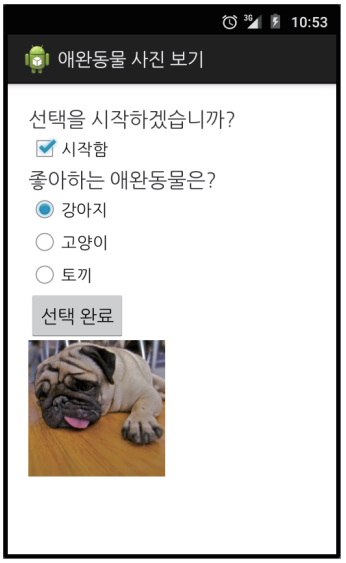


그림 4-8 실행 화면

1 안드로이드 프로젝트 생성

1 새 프로젝트를 만든다. 프로젝트 이름은 Project4_2로 하고, 패키지 이름은 com.cookandroid.project4_2로 한다. 그 외 규칙은 [실습 2-4]의 1~4를 따른다.

2 화면 디자인 및 편집

2 이번 프로젝트에서는 그림을 세 개 사용해야 한다. 우선 프로젝트의 [res]-[drawable]에 강아지, 고양이, 토끼 그림 파일을 미리 복사해놓는다.

TIP 복사할 그림 파일명은 꼭 모두 소문자로 해야 한다. 공백이 있으면 안 되므로 의미 단위를 끊어줄 필요가 있다면 언더스코어(_)를 이용한다. drawble 폴더에 그림을 복사해놓으면 그림의 아이디는 파일명과 동일해진다. 예를 들어 dog.png를 복사하면 XML에서는 @drawable/dog로, Java 코드에서는 R.drawable.dog로 접근한다.

[그림 4-9]와 같이 윈도 탐색기에서 그림 파일을 drawable 폴더에 복사/붙여넣기를 하고, drawble-hdpi를 선택하고 연속으로 〈OK〉를 클릭하면 해당 그림 파일이 복사된다.

3 Android Studio의 Project Tree에서 [app]-[res]-[layout]-[activity_main.xml]을 더블클릭해서 열고, 아래쪽 [Text] 탭을 클릭해서 화면을 코딩한다. 화면의 구성은 다음 규칙을 따른다.

- TextView, CheckBox, TextView, RadioGroup, RadioButton 각각 세 개, Button, ImageView의 차례로 만든다.
- 레이아웃에 padding을 적절히 지정한다.

• 맨 위의 TextView와 CheckBox를 제외하고, 나머지 위젯은 visibility 속성을 invisible로 지정한다.

• 각 위젯의 id는 위에서부터 Text1, ChkAgree, Text2, Rgroup1, RdoDog, RdoCat, RdoRabbit, BtnOK, ImgPet으로 한다.

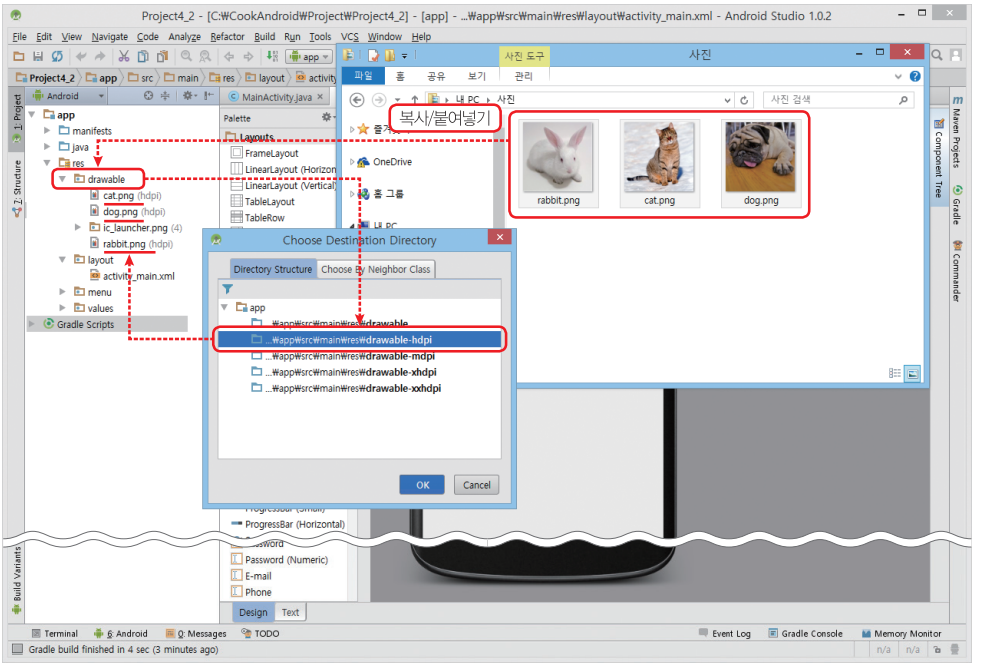


그림 4-9 그림 파일 복사

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" >
```

선택을 시작하겠습니까?
☐ 시작함

```
13      <RadioButton
14          android:id="@+id/RdoDog"
15          android:text="강아지" />
16       ~~~~ 중간 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" />
```

9, 12, 21, 24 위젯이 보이지 않도록 설정한다. 12행 RadioGroup에서 visibility 속성을 설정하면 그 안의 RadioButton은 visibility 속성을 설정하지 않아도 된다.

3 Java 코드 작성 및 수정

4 Android Studio의 Project Tree에서 [app]–[java]–[패키지 이름]–[MainActivity]를 더블클릭해서 연다.

5 다음 내용의 변수를 전역변수로 선언한다.

- activity_main.xml의 9개 위젯에 대응할 위젯 변수 9개

예제 4-29 Java 코드 1

```
1   ~~~~ 중간 생략 ~~~~
2   public class MainActivity extends Activity {
3       TextView text1, text2;
4       CheckBox chkAgree;
5       RadioGroup rGroup1;
6       RadioButton rdoDog, rdoCat, rdoRabbit;
7       Button btnOK;
8       ImageView imgPet;
9
10      @Override
11      public void onCreate(Bundle savedInstanceState) {
12  ~~~~ 중간 생략 ~~~~
```

6 각 위젯을 변수에 대입한다. onCreate() 메소드 안에서 처리한다.

예제 4-30 Java 코드 2

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

7 〈시작함〉 체크박스를 체크/언체크할 때마다 동작하는 리스너를 정의한다. onCreate() 내부에 정의한다.

예제 4-31 Java 코드 3

```
1  chkAgree.setOnCheckedChangeListener(new CompoundButton.
   OnCheckedChangeListener() {
2      public void onCheckedChanged(CompoundButton arg0, boolean arg1) {
3
4          if (chkAgree.isChecked() == true) {
5              text2.setVisibility(android.view.View.VISIBLE);
6              rGroup1.setVisibility(android.view.View.VISIBLE);
7              btnOK.setVisibility(android.view.View.VISIBLE);
8              imgPet.setVisibility(android.view.View.VISIBLE);
9          } else
10         {
11             text2.setVisibility(android.view.View.INVISIBLE);
12             rGroup1.setVisibility(android.view.View.INVISIBLE);
```

```
13              btnOK.setVisibility(android.view.View.INVISIBLE);
14              imgPet.setVisibility(android.view.View.INVISIBLE);
15          }
16      }
17  });
```

1 〈시작함〉 체크박스의 체크가 변경될 때 리스너를 설정한다. View.OnCheckedChange Listener()가 아니라 CompoundButton.OnCheckedChangeListener()라는 점을 주의한다.

2 onCheckedChanged() 메소드를 구현한다. 자동완성된다.

4 〈동의함〉 체크박스가 체크되어 있다면 실행된다.

5~8 숨겨졌던 위젯을 모두 보이도록 설정한다. 단 라디오버튼은 일일이 설정할 필요 없이 라디오버튼을 포함하는 라디오그룹만 보이도록 설정하면 된다. 보이거나 안 보이도록 설정하는 상수는 android.view.View 클래스에 지정되어 있는데, VISIBLE/INVISIBLE로 구분된다.

11~14 체크가 꺼지면 위젯을 모두 숨긴다.

8 〈선택 확인〉을 클릭하면 동작하는 리스너를 정의한다. 역시 onCreate() 내부에 정의한다.

예제 4-32 Java 코드 4

```
1  btnOK.setOnClickListener(new View.OnClickListener () {
2      public void onClick(View arg0) {
3          switch(rGroup1.getCheckedRadioButtonId()) {
4          case R.id.RdoDog:
5              imgPet.setImageResource(R.drawable.dog);
6              break;
7          case R.id.RdoCat:
8              imgPet.setImageResource(R.drawable.cat);
9              break;
10         case R.id.RdoRabbit:
11             imgPet.setImageResource(R.drawable.rabbit);
12             break;
13         default:
14                 Toast.makeText(getApplicationContext(), "동물 먼저 선택하세요", Toast.
   LENGTH_SHORT).show();
15         }
16     }
17 });
```

1 〈선택 완료〉를 클릭했을 때 리스너를 설정한다.

2 onClick() 메소드를 구현한다. 자동으로 완성되는 부분이다.

3~15 switch()~case문으로 다중 분기를 한다. getCheckedRadioButtonId()는 현재 라디오그룹에서 선택된 라디오버튼의 아이디 값을 반환한다.

4~6 id 값이 강아지 라디오버튼이면 이미지뷰에 drawable의 dog 그림을 출력한다.

13~14 라디오버튼 중에서 아무 것도 선택되지 않았다면 토스트 메시지를 보여준다. 토스트는 2장의 [예제 2-3]에서 잠깐 설명했다. 더 상세한 내용은 7장에서 다룰 것이다.

4 프로젝트 실행 및 결과 확인

⑨ 프로젝트를 실행해서 결과를 확인한다. [그림 4-8]의 화면이 나올 것이다.

5 안드로이드 응용프로그램 개발 완료

⑩ 애완동물 사진 보기의 프로젝트를 완료하였다.

이상으로 안드로이드 기본 위젯의 XML 속성과 Java 코딩 방식에 대해서 알아보았다. 중요한 내용은 View 클래스의 위젯들이 어떻게 상속받는지와 XML 속성을 사용하는 방식이다. 이번 장의 내용은 앞으로도 계속 사용될 것이므로 이해가 되지 않는다면 복습하길 바란다.

▶ **직접 풀어보기 4-4**

[실습 4-2]를 다음과 같이 수정해보자.

- "좋아하는 안드로이드 버전은?"으로 질문 변경
- 〈시작함〉을 Switch로 변경
- 〈선택 완료〉를 없애고, 라디오버튼을 선택할 때마다 즉시 해당 이미지가 나오도록 변경
- 제일 마지막에 〈종료〉와 〈처음부터〉를 추가하고 〈종료〉 클릭시에는 응용프로그램이 완전히 종료되도록 함. 〈처음부터〉 클릭 시에는 다시 초기화가 되고 처음 화면이 나오도록 함.

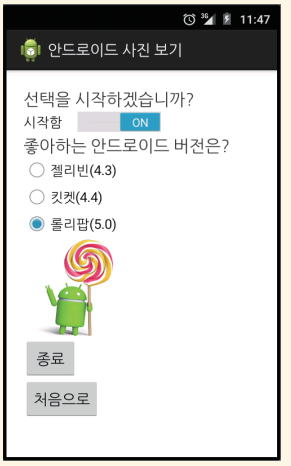
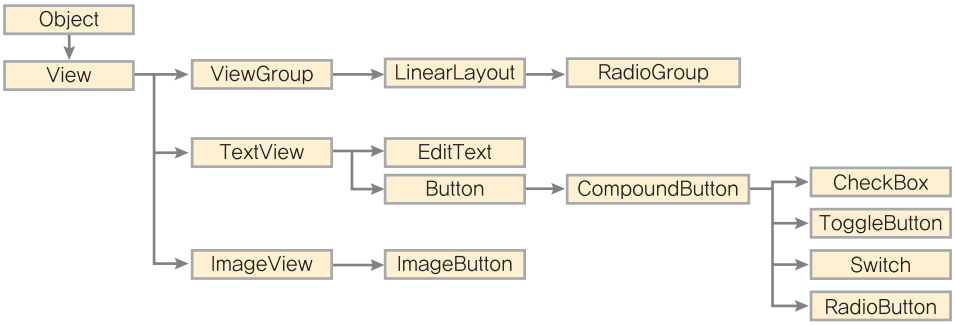


그림 4-10 안드로이드 사진 보기 프로젝트

요약

1 위젯은 넓은 의미로 View 클래스 아래의 모든 클래스들을 지칭하고, 좁은 의미로 버튼, 텍스트뷰, 체크박스 등 눈에 보이는 요소를 지칭한다. 그리고 위젯을 담는 틀을 레이아웃이라 부른다.

2 이번 장에서 사용한 위젯의 View 클래스 계층도는 다음과 같다.



3 View 클래스의 주요한 XML 속성으로는 id, layout_width, layout_height, background, padding, layout_margin, visibility, enable, clickable, rotation 등이 있다.

5 텍스트뷰의 주요한 XML 속성으로는 text, textColor, textSize, typeface, textStyle, singleLine 등이 있다.

5 XML 속성을 Java 코드로 설정할 수 있다. 예를 들어, background 속성은 setBackgroundColor() 메소드를 사용할 수 있다.

6 버튼은 Object → View → TextView → Button의 상속 관계를 갖는다. 그러므로 View와 TextView의 XML 속성 및 메소드를 대부분 동일하게 사용할 수 있다.

7 버튼의 Java 코드는 다음 3단계를 거쳐 작성한다.

❶ 버튼 변수 선언

```
Button  mybutton;
```

❷ 변수에 버튼 위젯 대입

```
mybutton = (Button) findViewById(R.id.button1);
```

요약

❸ 버튼을 클릭할 때 동작하는 클래스 정의

```
mybutton.setOnClickListener( new View.OnClickListener() {
    public void onClick(View v) {
        // 이 부분에 동작할 내용을 코딩
    }
});
```

8 에디트텍스트의 입력 값을 가져오는 코드는 다음과 같다.

```
String myStr =  myEdit.getText().toString();
```

9 이미지뷰 및 이미지버튼의 XML 속성에는 이미지의 경로를 나타내는 src, 이미지의 크기를 지정하는 maxHeight/maxWidth, 이미지의 확대/축소 방식을 지정하는 scaleType 등이 있다.

10 컴파운드버튼(CompoundButton)은 체크박스(CheckBox), 라디오버튼(RadioButton), 스위치(Switch), 토글버튼(ToggleButton)이라는 네 가지 하위 클래스를 갖는다.

11 체크박스의 Java 코드는 다음 3단계를 거쳐 작성한다.

❶ 체크박스 변수 선언

```
CheckBox mycheck;
```

❷ 변수에 체크박스 위젯 대입

```
mycheck = (CheckBox) findViewById(R.id.android);
```

❸ 체크박스가 변경될 때 동작하는 클래스 정의

```
mycheck.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton arg0, boolean arg1) {
        // 이 부분에 동작할 내용을 코딩
    }
});
```

12 라디오버튼은 보통 라디오그룹(RadioGroup) 안에 여러 개를 포함시켜서 사용한다.

13 이미지를 사용하려면 그림 파일을 [res]–[drawable] 폴더에 미리 복사해놓아야 한다.

연습문제

1 레이아웃의 종류에는 어떤 것이 있는지 찾아보고, 그 계층도를 그려라.

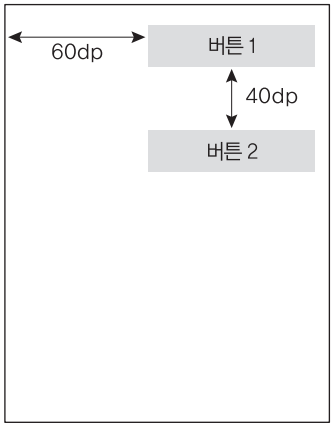
2 layout_width와 layout_height 속성 값으로 match_parent를 사용하려면 안드로이드 버전이 얼마 이상이어야 하는가?

3 텍스트뷰의 XML 속성과 각 속성과 관련된 메소드를 표로 정리해보자.

4 버튼에 클릭(Click)과 터치(Touch) 외에 어떤 이벤트가 가능한지 두 개 이상을 조사하고, 요약의 7번처럼 사용법의 예를 들어라.

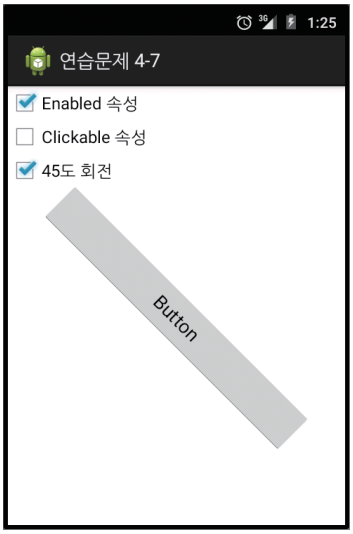
5 안드로이드에서 색상을 지정하는 속성 값은 #RRGGBB로 설정한다. 이에 대응하는 android.graphics.Color 클래스의 상수 필드에 대해서 조사해보자. 예를 들어, 0xffff0000은 RED에 대응된다.

6 다음 그림은 리니어레이아웃에 버튼 2개가 들어 있는 상태다. 그림과 같이 여백을 설정하기 위해서 리니어레이아웃과 버튼1에 각각 1개씩의 XML 속성 및 값을 설정하라.



연습문제

7 체크박스를 선택할 때마다 버튼의 속성이 설정되도록 프로젝트를 작성하라.



8 에디트텍스트에 키가 눌릴 때마다 바뀐 글자가 토스트 메시지로 나오도록 프로젝트를 작성하라.

HINT 에디트텍스트의 setOnKeyListener()를 사용한다.

연습문제

9 버튼에도 이미지를 넣을 수 있다. 다음과 같이 버튼을 클릭하면 이미지가 10°씩 회전하도록 프로젝트를 작성하라.

HINT 버튼의 XML 속성 중 drawableLeft, drawableRight 등을 이동한다.