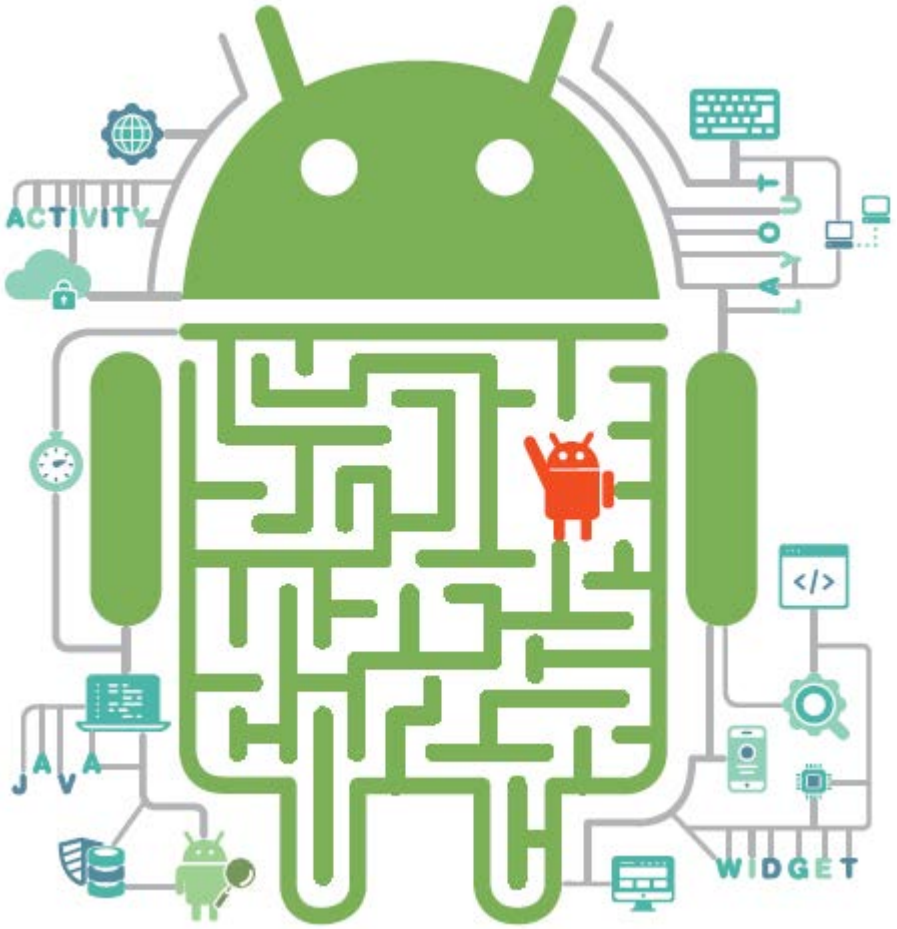
원리를 알면 IT가 맛있다
IT COOKBOOK

단계별로 배우는 안드로이드 프로그래밍

[강의교안 이용 안내]



단계별로 배우는

안드로이드 프로그래밍

Chapter 02. 레이아웃

한빛아카데미 Hanbit Academy, Inc.

목차

학습목표

- 뷰의 공통 속성을 이해하고 활용한다.
- 뷰의 크기와 여백 지정 방법을 익힌다.
- 대표적인 레이아웃의 사용법을 학습한다

01 뷰 = 레이아웃 + 위젯

뷰(View)

- 클래스 혹은 그 서브클래스로 만든 객체를 뜻함
- 액티비티 화면은 한 개 이상의 뷰로 구성
- 크게 두 종류(레이아웃과 위젯으)로 나눔

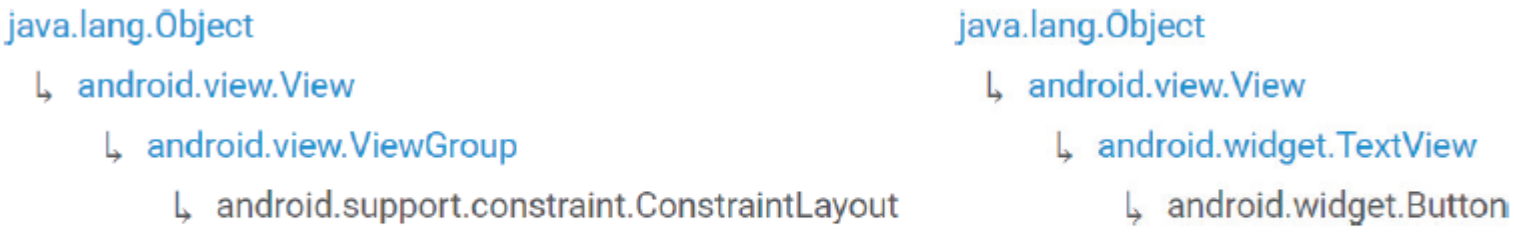


그림 2-1 Hello 앱 화면을 구성하는 뷰의 클래스 계층

01 뷰 = 레이아웃 + 위젯

레이아웃(Layout)

- ViewGroup의 서브클래스.
- 여러 개의 뷰('자식 뷰'라 부름)를 규칙대로 화면에 배치

위젯(Widget)

- View 또는 ViewGroup의 서브클래스로서 단독으로 사용

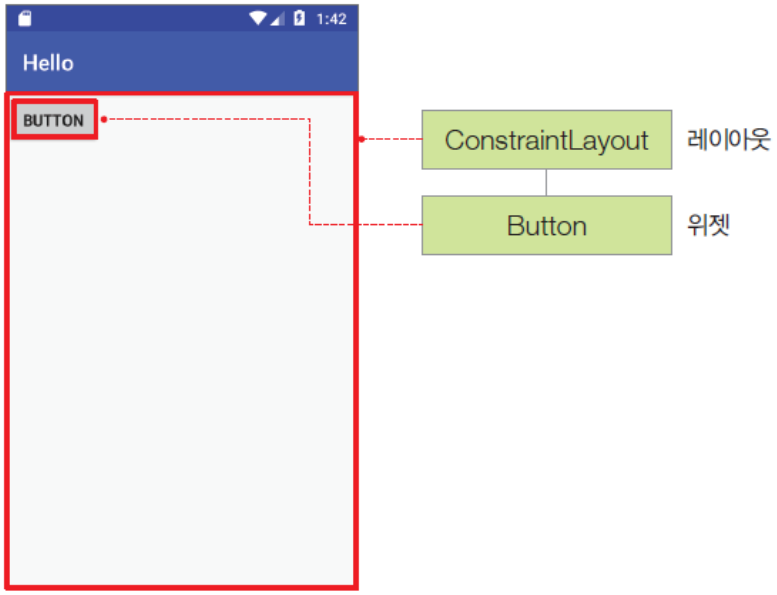


그림 2-2 Hello 앱의 화면을 구성하는 뷰 객체 = 레이아웃 + 위젯

레이아웃과 위젯은 View 클래스가 가진 속성을 공통 적용

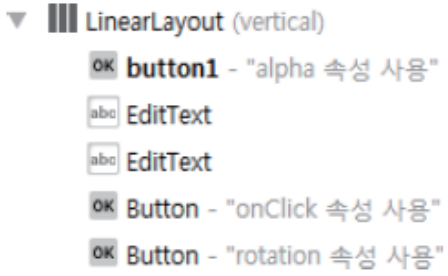
표 2-1 뷰의 공통 속성

XML 속성	관련 메서드	기능
android:alpha	setAlpha(float)	투명도를 0(완전 투명)~1(완전 불투명) 범위로 지정한다. [안드로이드 3.0부터 지원]
android:background	setBackgroundResource(int) setBackgroundColor(int) setBackground(Drawable)	배경 그림 또는 배경색을 지정한다.
android:focusable	setFocusable(boolean)	뷰가 키 입력을 받을 수 있는지를 "true" 또는 "false"로 지정한다.
android:id	setId(int)	자바 코드 또는 XML 파일 내에서 참조할 수 있도록 뷰에 식별자를 부여한다.
android:keepScreenOn	setKeepScreenOn(boolean)	값이 "true"이면 디바이스 화면이 계속 켜진 상태로 유지된다.
android:onClick	setOnClickListener(View.OnClickListener)	뷰를 클릭하면 호출될 메서드를 지정한다.
android:padding	setPadding(int, int, int, int)	패딩(padding; 내부 여백)을 지정한다.
android:rotation	setRotation(float)	뷰의 회전각을 도(degree) 단위로 지정한다. 양숫값은 시계 방향이고 음숫값은 반시계 방향이다. [안드로이드 3.0부터 지원]
android:visibility	setVisibility(int)	뷰의 가시성(visibility)을 설정한다. 화면에 보일 수도 있고(View.VISIBLE), 숨겨질 수도 있고(View.INVISIBLE), 없어질 수도 있다(View.GONE).

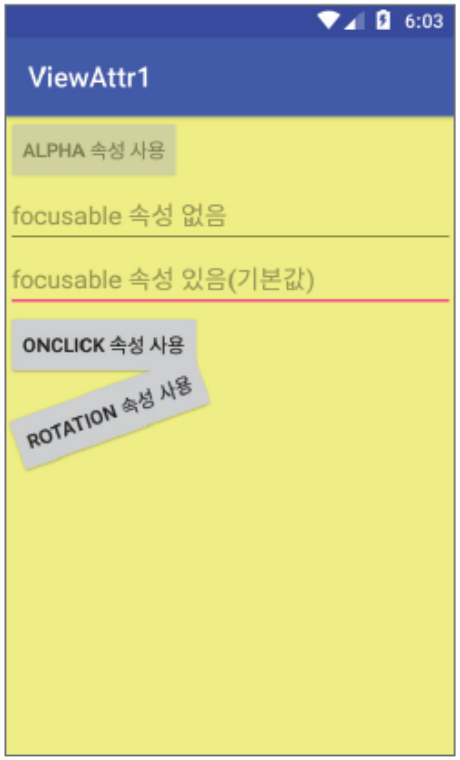
01 뷰 = 레이아웃 + 위젯 ▶ 뷰의 공통 속성

실습 2-1 ViewAttr1

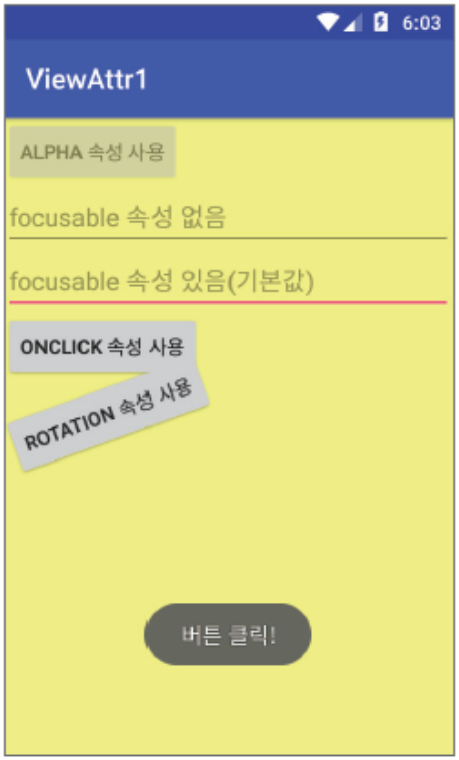
뷰의 공통 속성 분석



(a) 컴포넌트 트리



(b) 초기 화면



(c) 'ONCLICK 속성 사용' 클릭

그림 2-3 컴포넌트 트리와 실행 화면

01 뷰 = 레이아웃 + 위젯 ▶ 뷰의 공통 속성

실습 2-1 ViewAttr1

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:background="#ffff77"
6       android:orientation="vertical">
7       <Button
8           android:id="@+id/button1"
9           android:layout_width="wrap_content"
10          android:layout_height="wrap_content"
11          android:alpha="0.5"
12          android:text="alpha 속성 사용"/>
13      <EditText
```

실습 2-1 ViewAttr1

```
14          android:layout_width="match_parent"
15          android:layout_height="wrap_content"
16          android:focusable="false"
17          android:hint="focusable 속성 없음"/>
18      <EditText
19          android:layout_width="match_parent"
20          android:layout_height="wrap_content"
21          android:hint="focusable 속성 있음(기본값)"/>
22      <Button
23          android:layout_width="wrap_content"
24          android:layout_height="wrap_content"
25          android:onClick="mOnClick"
26          android:text="onClick 속성 사용"/>
27      <Button
28          android:layout_width="wrap_content"
29          android:layout_height="wrap_content"
30          android:rotation="-20"
31          android:text="rotation 속성 사용"/>
32  </LinearLayout>
```

01 뷰 = 레이아웃 + 위젯 ▶ 뷰의 공통 속성

실습 2-1 ViewAttr1

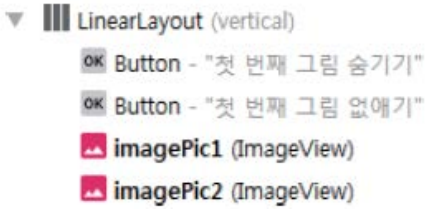
MainActivity.java

```
1   public class MainActivity extends AppCompatActivity {
2
3       @Override
4       protected void onCreate(Bundle savedInstanceState) {
5           super.onCreate(savedInstanceState);
6           setContentView(R.layout.activity_main);
7       }
8
9       public void mOnClick(View v) {
10          Toast.makeText(this, "버튼 클릭!", Toast.LENGTH_SHORT).show();
11      }
12  }
```

01 뷰 = 레이아웃 + 위젯 ▶ 뷰의 공통 속성

실습 2-2 ViewAttr2

■ 컴포넌트 트리와 초기 화면



(a) 컴포넌트 트리

(b) 초기 화면

01 뷰 = 레이아웃 + 위젯 ▶ 뷰의 공통 속성

실습 2-2 ViewAttr2

- (c)는 '첫 번째 그림 숨기기'를 클릭하여 INVISIBLE 속성을 적용한 결과
- (d)는 '첫 번째 그림 없애기'를 클릭하여 GONE 속성을 적용한 결과

(c) '첫 번째 그림 숨기기' 클릭 (d) '첫 번째 그림 없애기' 클릭

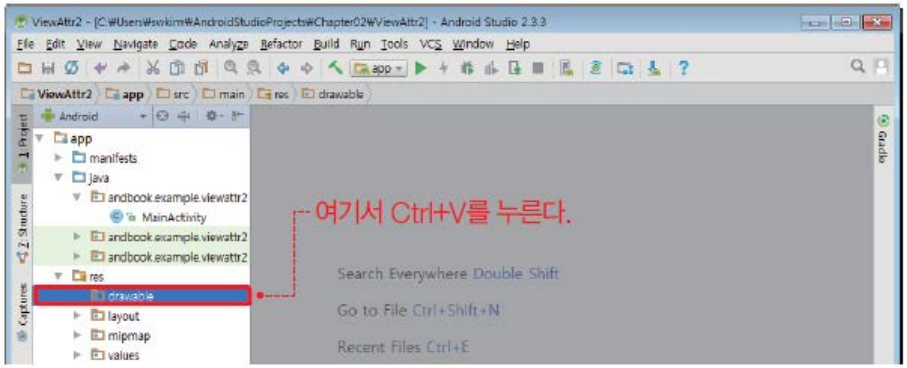
그림 2-4 컴포넌트 트리와 실행 화면

01 뷰 = 레이아웃 + 위젯 ▶ 뷰의 공통 속성

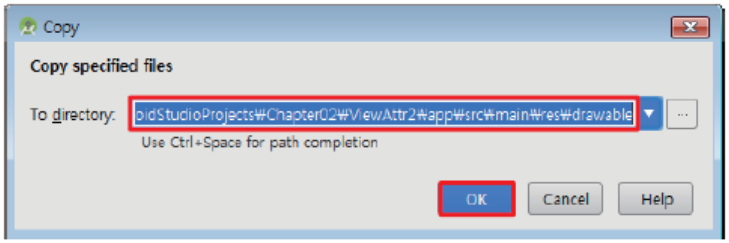
실습 2-2 ViewAttr2

- 1장의 Hello 예제처럼 프로젝트를 생성

(a) 준비한 그림 파일



(b) 그림 파일 붙여넣기



(c) Copy 대화상자

그림 2-5 drawable 리소스 준비

01 뷰 = 레이아웃 + 위젯 ▶ 뷰의 공통 속성

실습 2-2 ViewAttr2

- 컴포넌트 트리를 참고하여 res/layout/activity_main.xml 수정

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:background="@drawable/background"
6       android:orientation="vertical">
7       <Button
8           android:layout_width="wrap_content"
9           android:layout_height="wrap_content"
10          android:onClick="mOnClick1"
11          android:text="첫 번째 그림 숨기기"/>
12      <Button
```

01 뷰 = 레이아웃 + 위젯 ▶ 뷰의 공통 속성

실습 2-2 ViewAttr2

```
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:onClick="mOnClick2"
16          android:text="첫 번째 그림 없애기"/>
17      <ImageView
18          android:id="@+id/imagePic1"
19          android:layout_width="wrap_content"
20          android:layout_height="wrap_content"
21          android:src="@drawable/icon1"/>
22      <ImageView
23          android:id="@+id/imagePic2"
24          android:layout_width="wrap_content"
25          android:layout_height="wrap_content"
26          android:src="@drawable/icon2"/>
27  </LinearLayout>
```

실습 2-2 ViewAttr2

MainActivity 클래스에 다음 코드 작성

MainActivity.java

```
1   public class MainActivity extends AppCompatActivity {
2
3       private ImageView mImagePic1;
4
5       @Override
6       protected void onCreate(Bundle savedInstanceState) {
7           super.onCreate(savedInstanceState);
8           setContentView(R.layout.activity_main);
9           mImagePic1 = (ImageView) findViewById(R.id.imagePic1);
10      }
11
12      public void mOnClick1(View v) {
13          mImagePic1.setVisibility(View.INVISIBLE);
14      }
15
16      public void mOnClick2(View v) {
17          mImagePic1.setVisibility(View.GONE);
18      }
19  }
```

알파값 0으로 주는 것.

실습 2-2 ViewAttr2

제공되는 크기 지정 단위

- 레이아웃 리소스에서 뷰의 크기를 지정할 때 숫자 뒤에 붙여서 사용

표 2-2 크기 지정 단위

단위	의미
px	픽셀pixel이다. 특수한 경우를 제외하고는 사용하지 않는 편이 좋다.
mm in	각각 밀리미터millimeter와 인치inch이다. 다양한 디바이스에서 일관된 크기로 출력할 수 있다는 점에서 픽셀 단위보다는 좋지만 세밀한 크기를 지정하려면 소수점 이하 숫자가 필요한 경우가 많아서 사용을 권장하지 않는다.
pt	1/72인치다. 인쇄 기술에서 유래한 단위로 주로 글자 크기를 지정할 때 사용된다.
dp 또는 dip (사용 권장)	**밀도 독립적 픽셀**Density-Independent Pixel이다. 160dp로 출력하면 논리적 1인치가 되도록 한다는 약속된 단위다. 여기서 논리적 1인치는 실제 1인치 길이와는 조금 다를 수 있다. 실제 출력 시에는 디바이스에 따라 물리적 픽셀 수가 결정된다. 다양한 디바이스에서 일관된 크기로 출력할 수 있고 세밀한 크기를 지정할 때 소수점 이하 숫자가 필요한 경우가 적다는 장점이 있다. 참고 예를 들어 160dp로 출력한 결과는 160dpi 디바이스에서는 160픽셀로, 320dpi 디바이스에서는 320픽셀로 출력된다. 여기서 dpidot-per-inch는 디스플레이의 물리적 특징이 아니라 안드로이드 플랫폼에서 정한 값이다. 안드로이드 7.0부터는 이전 버전과 달리 시스템 설정에서 dpi를 사용자가 변경할 수 있다. 안드로이드 에뮬레이터의 [설정]-[디스플레이]-[디스플레이 크기]로 들어가 살펴보자.
sp (사용 권장)	**스케일 독립적 픽셀**Scale-Independent Pixel이다. dp와 기본 정의는 같지만 안드로이드 시스템 전체의 글꼴 크기를 설정에서 증감하면 그에 맞게 크기가 증감한다는 차이가 있다. 시력이 좋지 않은 사용자를 고려하여 화면을 디자인할 때 (특히 글자 크기를 지정할 때) 사용을 권장한다.

실습 2-3 DimensionUnits1

- 크기 지정 단위를 버튼의 폭에 적용한 결과

(a) 에뮬레이터(넥서스 S)
→ 480*800 해상도, 240dpi

(b) 넥서스 5X
→ 1080*1920 해상도, 420dpi

그림 2-6 실행 화면

01 뷰 = 레이아웃 + 위젯 ▶ 크기 지정 단위

실습 2-3 DimensionUnits1

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:orientation="vertical">
6       <Button
7           android:layout_width="240px"
8           android:layout_height="wrap_content"
9           android:text="폭 240px"/>
10      <Button
11          android:layout_width="25.4mm"
12          android:layout_height="wrap_content"
13          android:text="폭 25.4mm"/>
14      <Button
15          android:layout_width="1in"
16          android:layout_height="wrap_content"
17          android:text="폭 1in"/>
```

01 뷰 = 레이아웃 + 위젯 ▶ 크기 지정 단위

실습 2-3 DimensionUnits1

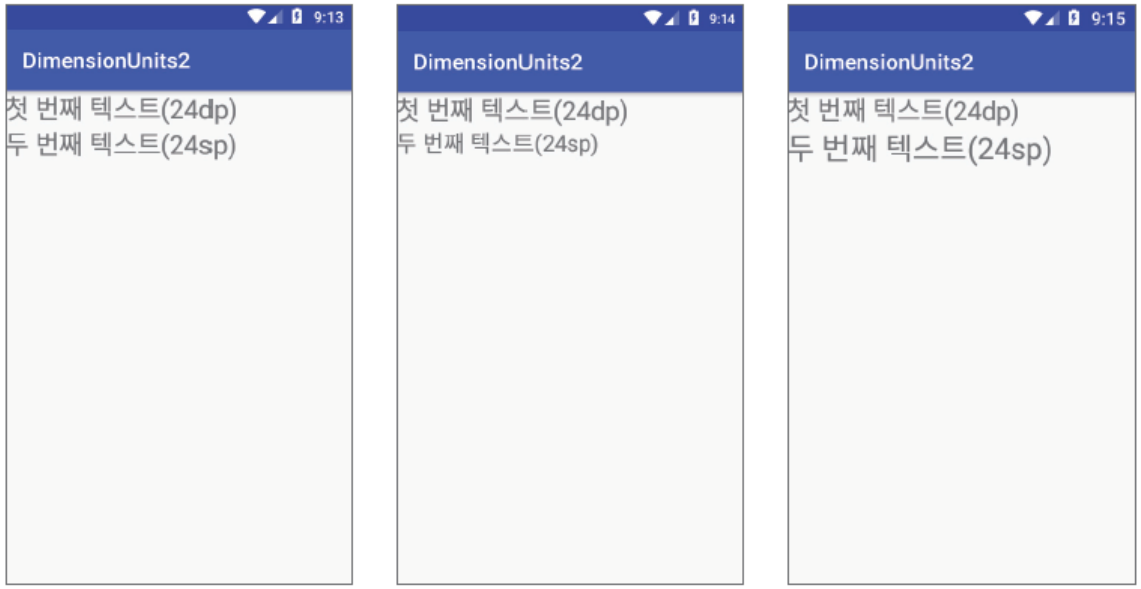
```
18      <Button
19          android:layout_width="72pt"
20          android:layout_height="wrap_content"
21          android:text="폭 72pt"/>
22      <Button
23          android:layout_width="160dp"
24          android:layout_height="wrap_content"
25          android:text="폭 160dp"/>
26      <Button
27          android:layout_width="160sp"
28          android:layout_height="wrap_content"
29          android:text="폭 160sp"/>
30  </LinearLayout>
```

01 뷰 = 레이아웃 + 위젯 ▶ 크기 지정 단위

실습 2-4 DimensionUnits2

dp와 sp의 차이

- 시스템 설정의 [디스플레이]-[글꼴 크기] 메뉴에서 시스템 전체 글꼴의 크기를 변경하면 sp 단위로 출력한 텍스트의 크기만 변경



(a) 초기 화면(기본 글꼴) (b) 작은 글꼴 선택 시 (c) 큰 글꼴 선택 시

그림 2-7 실행 화면

01 뷰 = 레이아웃 + 위젯 ▶ 크기 지정 단위

실습 2-4 DimensionUnits2

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:orientation="vertical">
6       <TextView
7           android:layout_width="wrap_content"
8           android:layout_height="wrap_content"
9           android:text="첫 번째 텍스트(24dp)"
10          android:textSize="24dp"/>
11      <TextView
12          android:layout_width="wrap_content"
13          android:layout_height="wrap_content"
14          android:text="두 번째 텍스트(24sp)"
15          android:textSize="24sp"/>
16  </LinearLayout>
```

마진margin

- 외부 여백, 즉 뷰와 부모 뷰 사이의 공간
- 마진으로 지정한 여백은 뷰 자신의 영역에 포함되지 않음

패딩padding

- 내부 여백, 즉 뷰와 뷰의 내용물 사이의 공간이다. 패딩으로 지정한 여백은 뷰 자신의 영역에 포함된다.

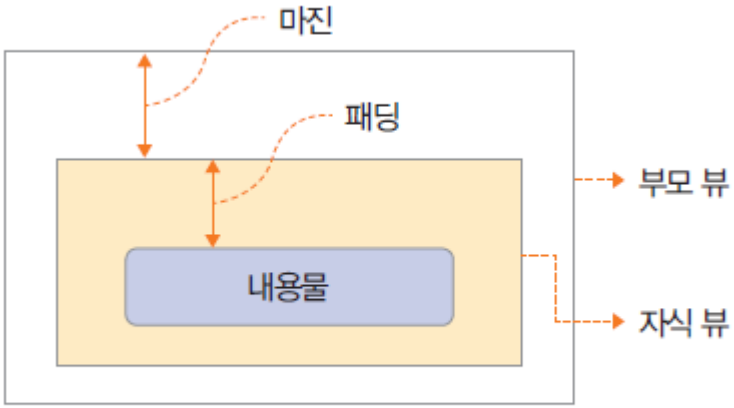


그림 2-8 마진과 패딩

마진과 패딩 관련 속성

표 2-4 패딩 관련 속성

XML 속성	기능
padding	상하좌우 네 방향 패딩을 지정한다.
paddingTop	위쪽 패딩을 지정한다.
paddingBottom	아래쪽 패딩을 지정한다.
paddingLeft	왼쪽 패딩을 지정한다.
paddingRight	오른쪽 패딩을 지정한다.

표 2-3 마진 관련 속성

XML 속성	기능
layout_margin	상하좌우 네 방향 마진을 지정한다.
layout_marginTop	위쪽 마진을 지정한다.
layout_marginBottom	아래쪽 마진을 지정한다.
layout_marginLeft	왼쪽 마진을 지정한다.
layout_marginRight	오른쪽 마진을 지정한다.

실습 2-5 MarginPadding

마진과 패딩을 부모 뷰와 자식 뷰에 적용한 예

(a) 실행 화면 (b) 여백 분석

그림 2-9 실행 화면과 여백 분석

01 뷰 = 레이아웃 + 위젯 ▶ 마진과 패딩

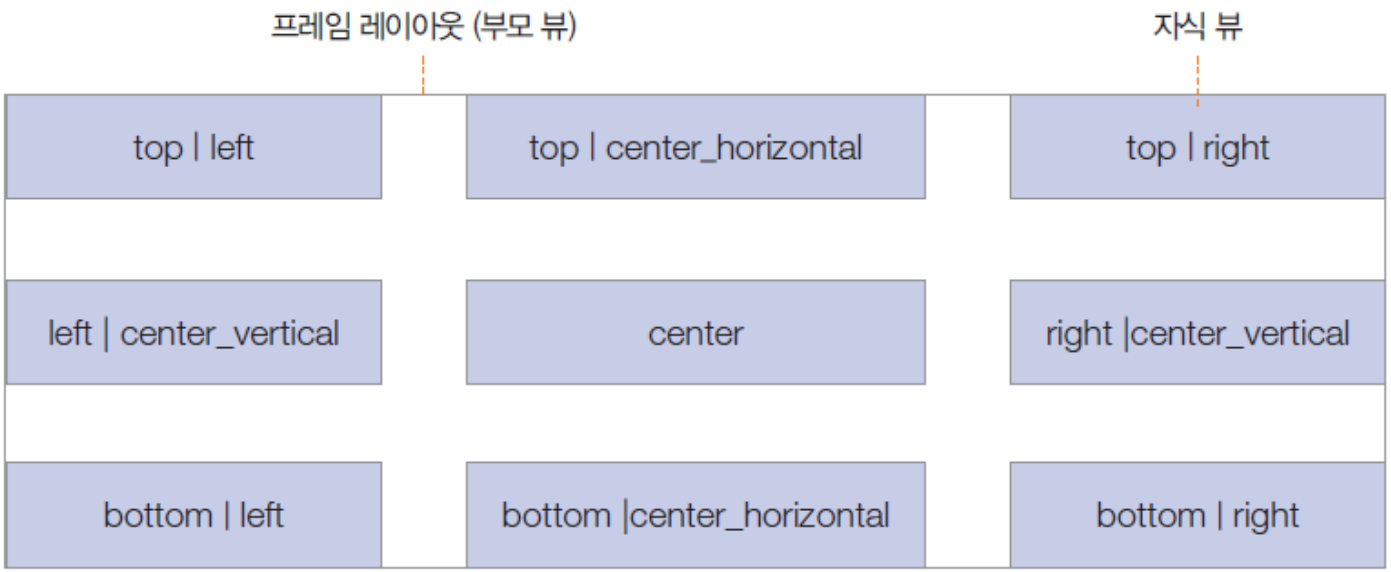
실습 2-5 MarginPadding

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:background="#00ff00"
6       android:orientation="vertical"
7       android:padding="30dp">
8       <TextView
9           android:layout_width="wrap_content"
10          android:layout_height="wrap_content"
11          android:layout_marginTop="60dp"
12          android:background="#ffff00"
13          android:paddingLeft="30dp"
14          android:text="마진과 패딩 연습"
15          android:textSize="28sp"/>
16  </LinearLayout>
```

프레임 레이아웃(FrameLayout)

- 화면의 일정 영역을 차지하는 틀frame 안에 자식 뷰를 배치하는 뷰 그룹
- 기본 동작은 자식 뷰를 프레임 레이아웃의 좌측 상단에 배치하는 것
- 그래비티(layout_graivity)속성을 사용하면 자식 뷰의 위치를 변경 가능



- top과 left는 기본값이므로 생략 가능
- center = center_horizontal | center_vertical

그림 2-10 그래비티 속성값에 따른 자식 뷰의 위치

실습 2-6 FrameLayout1

- 프레임 레이아웃 안에 자식 뷰를 그래비티 속성값을 바꿔가며 배치
- 우측 상단의 텍스트뷰와 아날로그 시계는 겹쳐서 출력

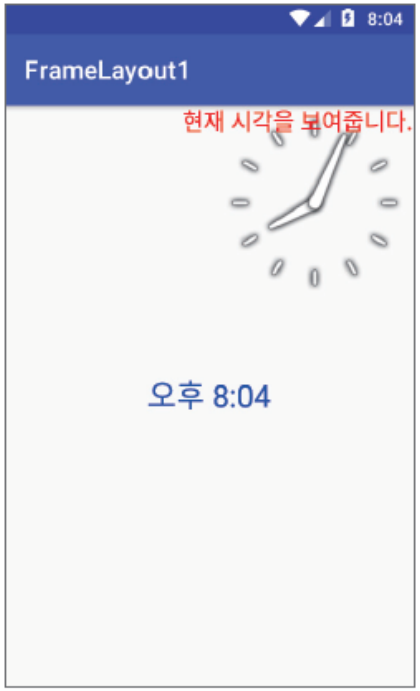


그림 2-11 실행 화면

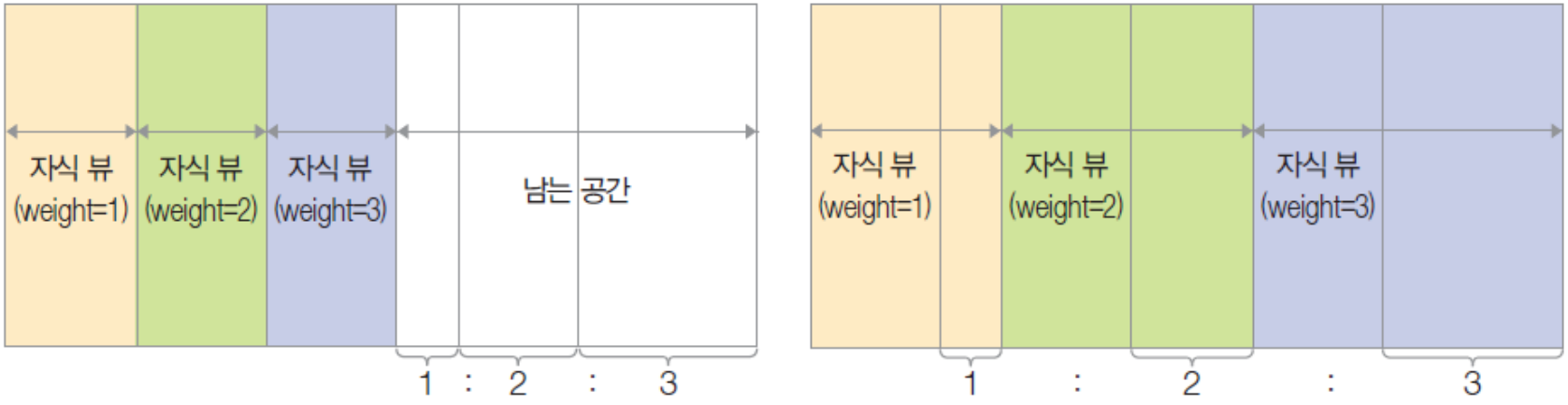
실습 2-6 FrameLayout1

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent">
5       <AnalogClock
6           android:layout_width="wrap_content"
7           android:layout_height="wrap_content"
8           android:layout_gravity="right"/>
9       <TextView
10          android:layout_width="wrap_content"
11          android:layout_height="wrap_content"
12          android:layout_gravity="right"
13          android:text="현재 시각을 보여줍니다."
14          android:textColor="#ff0000"
15          android:textSize="18sp"/>
16      <DigitalClock
17          android:layout_width="wrap_content"
18          android:layout_height="wrap_content"
19          android:layout_gravity="center"
20          android:textColor="#0000ff"
21          android:textSize="24sp"/>
22  </FrameLayout>
```

리니어 레이아웃(LinearLayout)

- 자식 뷰를 가로 또는 세로 방향으로 일렬로 배치하는 뷰 그룹
- 기본값은 가로 방향이며 orientation 속성값을 "vertical"로 하면 세로 방향이 됨
- 리니어 레이아웃에 자식 뷰를 차례로 배치하다 남는 공간이 생기면, layout_weight 속성을 자식 뷰에 적용하여 일정 비율로 나누어줄 수 있음
- 남는 공간이 없으면 layout_weight 속성이 제대로 동작하지 않으므로 주의



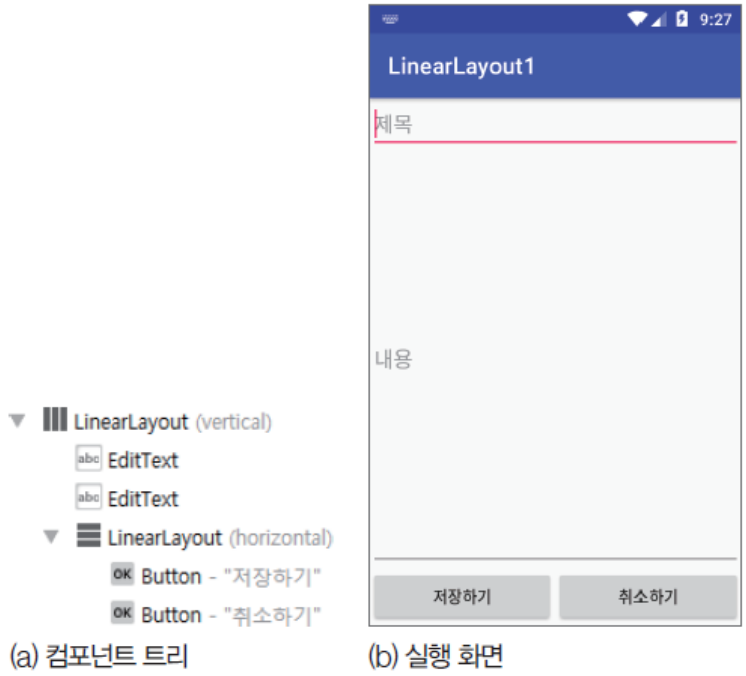
(a) 리니어 레이아웃 내부의 남는 공간

(b) 자식 뷰에 1:2:3으로 나누어준다.

그림 2-12 layout weight 속성

실습 2-7 LinearLayout1

리니어 레이아웃의 기본 사용법



(a) 컴포넌트 트리 (b) 실행 화면

그림 2-13 컴포넌트 트리와 실행 화면

실습 2-7 LinearLayout1

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="제목"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:hint="내용"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="저장하기"/>
        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="취소하기"/>
    </LinearLayout>
</LinearLayout>
```

실습 2-8 LinearLayout2

baselineAligned의 기능

- 텍스트를 가로 방향으로 배치할 때 기준선(baseline)에 맞출지를 결정하는 값
- 기본값은 "true"
- 속성값을 "false"로 변경하면 기준선이 맞지 않아 부자연스러움
- 한글은 기준선 개념은 없지만 baselineAligned 속성의 영향을 받음

그림 2-14 기준선 개념

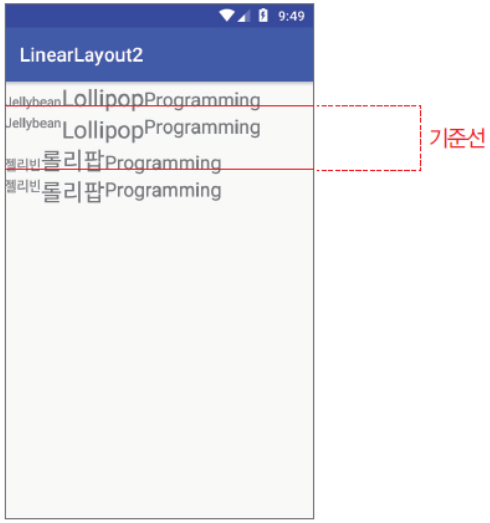


그림 2-15 실행 화면

실습 2-8 LinearLayout2

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="…생략…"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Jellybean"
            android:textSize="14sp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Lollipop"
            android:textSize="24sp"/>
<TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
android:text="Programming"
            android:textSize="20sp"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="젤리빈"
            android:textSize="14sp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="롤리팝"
            android:textSize="24sp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Programming"
            android:textSize="20sp"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
```

실습 2-8 LinearLayout2

```
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:baselineAligned="false">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Jellybean"
        android:textSize="14sp"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Lollipop"
        android:textSize="24sp"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Programming"
        android:layout_height="wrap_content"
        android:baselineAligned="false">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="젤리빈"
            android:textSize="14sp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="롤리팝"
            android:textSize="24sp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Programming"
            android:textSize="20sp"/>
    </LinearLayout>
</LinearLayout>
```

렐러티브 레이아웃(RelativeLayout)

- 자식 뷰를 부모(parent) 뷰나 형제(sibling) 뷰를 기준으로 상대적인 위치를 주어 배치하는 뷰 그룹

- 자식 뷰의 기본 위치: 부모 뷰의 좌측 상단이지만 다양한 layout_~ 속성이 마련되어 있어 자유롭게 위치 지정

부모 뷰를 기준으로 위치 지정

- 자식 뷰의 속성 이름으로 layout_alignParent~ 또는 layout_center~ 형태를 사용하며 속성값으로 "true" 또는 "false"를 지정한다.

표 2-5 부모 뷰를 기준으로 자식 뷰의 위치 지정

XML 속성	기능
layout_alignParentTop	부모 뷰의 윗면과 자식 뷰의 윗면을 일치시킨다.
layout_alignParentBottom	부모 뷰의 아랫면과 자식 뷰의 아랫면을 일치시킨다.
layout_alignParentLeft	부모 뷰의 왼쪽 면과 자식 뷰의 왼쪽 면을 일치시킨다.
layout_alignParentRight	부모 뷰의 오른쪽 면과 자식 뷰의 오른쪽 면을 일치시킨다.
layout_alignWithParentIfMissing	앵커가 없어지면 부모 뷰를 기준으로 자식 뷰를 정렬한다. 참고 이 속성은 설명만으로 이해하기 어려우므로 잠시 후 RelativeLayout2 예제로 테스트할 것이다. 참고 자식 뷰의 위치 지정 시 기준이 되는 형제 뷰를 안드로이드 개발자 문서에서 **앵커**anchor라 한다.
layout_centerHorizontal	부모 영역 내에서 수평으로 중앙에 자식 뷰를 배치한다.
layout_centerVertical	부모 영역 내에서 수직으로 중앙에 자식 뷰를 배치한다.
layout_centerInParent	부모 영역 내의 정중앙에 자식 뷰를 배치한다.

부모 뷰를 기준으로 위치 지정

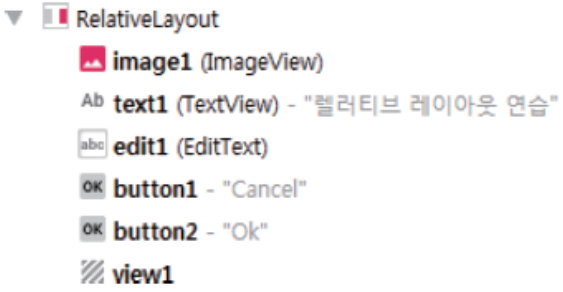
- 자식 뷰의 속성 이름으로 layout_align~ 또는 layout_{above|below} 또는 layout_to{Left|Right}Of 형태를 사용
- 속성값으로 "@+id/식별자" 또는 "@id/식별자"를 지정

표 2-6 형제 뷰를 기준으로 자식 뷰의 위치 지정

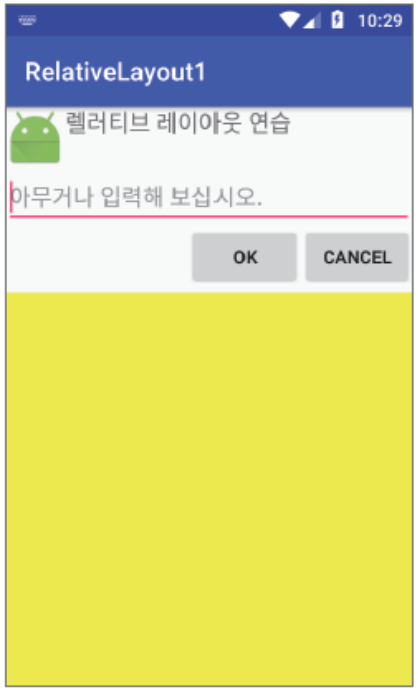
XML 속성	기능
layout_alignTop	앵커의 윗면과 자식 뷰의 윗면을 일치시킨다.
layout_alignBottom	앵커의 아랫면과 자식 뷰의 아랫면을 일치시킨다.
layout_alignLeft	앵커의 왼쪽 면과 자식 뷰의 왼쪽 면을 일치시킨다.
layout_alignRight	앵커의 오른쪽 면과 자식 뷰의 오른쪽 면을 일치시킨다.
layout_alignBaseline	앵커의 기준선과 자식 뷰의 기준선을 일치시킨다. 참고 이 속성도 잠시 후 RelativeLayout2 예제로 테스트할 것이다.
layout_above	앵커의 위쪽에 자식 뷰를 배치한다.
layout_below	앵커의 아래쪽에 자식 뷰를 배치한다.
layout_toLeftOf	앵커의 왼쪽에 자식 뷰를 배치한다.
layout_toRightOf	앵커의 오른쪽에 자식 뷰를 배치한다.

실습 2-9 RelativeLayout1

렐러티브 레이아웃의 기본 사용법



(a) 컴포넌트 트리



(b) 실행 화면

그림 2-16 컴포넌트 트리와 실행 화면

실습 2-9 RelativeLayout1

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent">
5       <ImageView
6           android:id="@+id/image1"
7           android:layout_width="wrap_content"
8           android:layout_height="wrap_content"
9           android:src="@mipmap/ic_launcher"/>
10      <TextView
11          android:id="@+id/text1"
12          android:layout_width="wrap_content"
13          android:layout_height="wrap_content"
14          android:layout_alignParentTop="true"
15          android:layout_toRightOf="@+id/image1"
16          android:text="렐러티브 레이아웃 연습"
17          android:textSize="18sp"/>
```

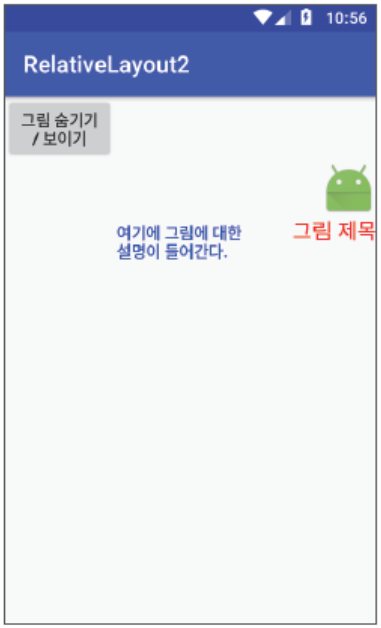
실습 2-9 RelativeLayout1

```
19          android:id="@+id/edit1"
20          android:layout_width="match_parent"
21          android:layout_height="wrap_content"
22          android:layout_alignParentLeft="true"
23          android:layout_below="@+id/image1"
24          android:hint="아무거나 입력해보십시오."/>
25      <Button
26          android:id="@+id/button1"
27          android:layout_width="wrap_content"
28          android:layout_height="wrap_content"
29          android:layout_alignParentRight="true"
30          android:layout_below="@+id/edit1"
31          android:text="Cancel"/>
32      <Button
33          android:id="@+id/button2"
34          android:layout_width="wrap_content"
35          android:layout_height="wrap_content"
36          android:layout_below="@+id/edit1"
37          android:layout_toLeftOf="@+id/button1"
38          android:text="Ok"/>
39      <View
40          android:id="@+id/view1"
41          android:layout_width="match_parent"
42          android:layout_height="0dp"
43          android:layout_alignParentBottom="true"
44          android:layout_below="@+id/button2"
45          android:layout_marginTop="4dp"
46          android:background="#ffff00"/>
47  </RelativeLayout>
```

실습 2-10 RelativeLayout2

- alignBaseline과 alignWithParentIfMissing 속성 사용

(a) 컴포넌트 트리

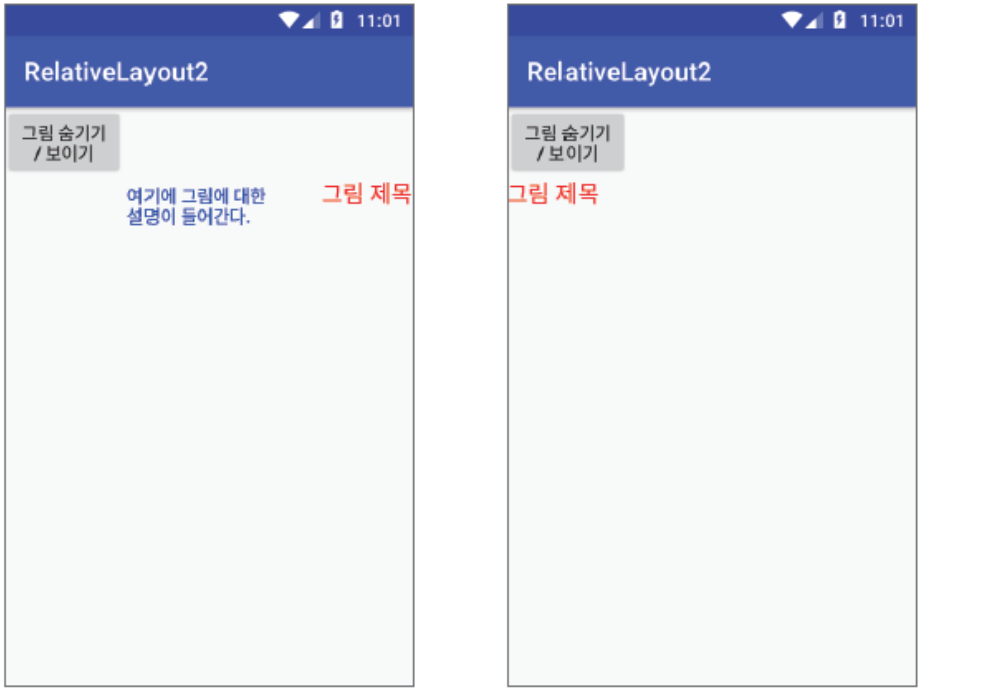


(b) 실행 화면

그림 2-17 컴포넌트 트리와 실행 화면

실습 2-10 RelativeLayout2

- 버튼을 클릭하면 앵커 역할을 하는 이미지뷰가 화면에서 없어짐



(a) alignWithParentIfMissing="true" (b) alignWithParentIfMissing="false"

그림 2-18 앵커가 없어졌을 때 alignWithParentIfMissing 속성값에 따른 차이

실습 2-10 RelativeLayout2

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="fill_parent"
4       android:layout_height="fill_parent">
5       <Button
6           android:id="@+id/button1"
7           android:layout_width="wrap_content"
8           android:layout_height="wrap_content"
9           android:onClick="mOnClick"
10          android:text="그림 숨기기\n/ 보이기"/>
11      <ImageView
12          android:id="@+id/image1"
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:layout_alignParentRight="true"
16          android:layout_below="@+id/button1"
17          android:src="@mipmap/ic_launcher"/>
18      <TextView
19          android:id="@+id/text1"
20          android:layout_width="wrap_content"
21          android:layout_height="wrap_content"
22          android:layout_alignRight="@+id/image1"
```

실습 2-10 RelativeLayout2

```
23          android:layout_alignWithParentIfMissing="true"
24          android:layout_below="@+id/image1"
25          android:text="그림 제목"
26          android:textColor="#ff0000"
27          android:textSize="18sp"/>
28      <TextView
29          android:id="@+id/text2"
30          android:layout_width="wrap_content"
31          android:layout_height="wrap_content"
32          android:layout_alignBaseline="@+id/text1"
33          android:layout_marginLeft="2dp"
34          android:layout_marginRight="2dp"
35          android:layout_toLeftOf="@+id/text1"
36          android:layout_toRightOf="@id/button1"
37          android:text="여기에 그림에 대한\n설명이 들어간다."
38          android:textColor="#0000ff"
39          android:textSize="14sp"/>
40  </RelativeLayout>
```

실습 2-10 RelativeLayout2

MainActivity.java

```
1   public class MainActivity extends AppCompatActivity {
2
3       private ImageView mImage1;
4
5       @Override
6       protected void onCreate(Bundle savedInstanceState) {
7           super.onCreate(savedInstanceState);
8           setContentView(R.layout.activity_main);
9           mImage1 = (ImageView) findViewById(R.id.image1);
10      }
11
12      public void mOnClick(View v) {
13          if (mImage1.getVisibility() == View.VISIBLE)
14              mImage1.setVisibility(View.GONE);
15          else
16              mImage1.setVisibility(View.VISIBLE);
17      }
18  }
```

테이블 레이아웃(TableLayout)

- 테이블table (표)의 각 셀에 자식 뷰를 차례로 배치하는 뷰 그룹
- 한 개 이상의 행(row)으로 구성
- 주로 TableRow 객체가 사용되며 가끔은 일반 View 객체가 사용
- TableRow 객체에는 각 셀에 자식 뷰가 한 개씩 들어갈 수 있으며 전체 열column의 개수는 셀 개수가 가장 많은 TableRow 객체로 결정
- TableRow 객체 대신 일반 View 객체가 사용되면 해당 View 객체가 한 행 전체를 무조건 차지
- 각각의 자식 뷰는 폭을 지정할 수 없고(폭은 항상 MATCH_PARENT) 높이만 지정할 수 있다.

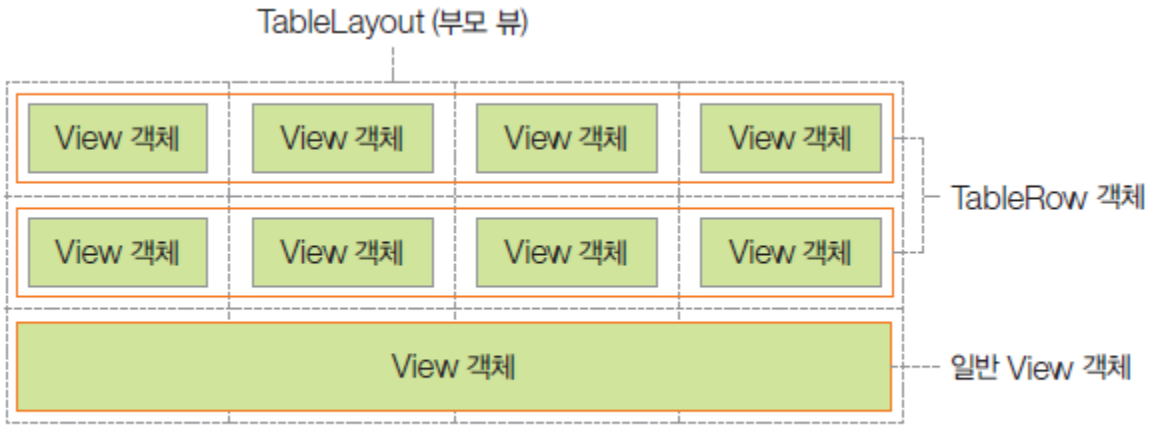


그림 2-19 테이블 레이아웃과 TableRow 객체

실습 2-11 TableLayout1

- 네 개의 행을 가진 테이블 레이아웃

(a) 컴포넌트 트리 (b) 실행 화면 (1) – 세로 방향 (c) 실행 화면 (2) – 가로 방향

그림 2-20 컴포넌트 트리와 실행 화면

실습 2-11 TableLayout1

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:shrinkColumns="2"
6       android:stretchColumns="2">
7       <TableRow
8           android:layout_width="wrap_content"
9           android:layout_height="wrap_content">
10          <TextView
11              android:layout_width="wrap_content"
```

실습 2-11 TableLayout1

```
12              android:layout_height="wrap_content"
13              android:layout_gravity="center"
14              android:text="사과"/>
15          <TextView
16              android:layout_width="wrap_content"
17              android:layout_height="wrap_content"
18              android:layout_gravity="center"
19              android:text="딸기"/>
20          <TextView
21              android:layout_width="wrap_content"
22              android:layout_height="wrap_content"
23              android:layout_gravity="center"
24              android:text="수박"/>
25      </TableRow>
26      <TableRow
27          android:layout_width="wrap_content"
28          android:layout_height="wrap_content">
29          <ImageView
30              android:layout_width="80dp"
31              android:layout_height="80dp"
```

실습 2-11 TableLayout1

```
32              android:src="@drawable/apple"/>
33          <ImageView
34              android:layout_width="80dp"
35              android:layout_height="80dp"
36              android:src="@drawable/strawberry"/>
37          <ImageView
38              android:layout_width="80dp"
39              android:layout_height="80dp"
40              android:src="@drawable/watermelon"/>
41      </TableRow>
42      <TableRow
43          android:layout_width="wrap_content"
44          android:layout_height="wrap_content">
45          <Button
46              android:layout_width="wrap_content"
47              android:layout_height="wrap_content"
48              android:text="사과 선택"/>
49          <Button
50              android:layout_width="wrap_content"
51              android:layout_height="wrap_content"
52              android:text="딸기 선택"/>
```

실습 2-11 TableLayout1

```
53          <Button
54              android:layout_width="wrap_content"
55              android:layout_height="wrap_content"
56              android:text="수박 선택"/>
57      </TableRow>
58      <EditText
59          android:layout_width="wrap_content"
60          android:layout_height="wrap_content"
61          android:hint="일반 뷰는 한 행을 차지"/>
62  </TableLayout>
```

그리드 레이아웃(GridLayout)

- 격자(grid) 내부의 셀에 자식 뷰를 배치하는 뷰 그룹
- 테이블 레이아웃과 비슷하지만, 각각의 자식 뷰가 자신의 위치와 차지하는 셀의 개수를 독립적으로 지정 가능
- 테이블 레이아웃과 달리 안드로이드 4.0부터 지원된다는 제약

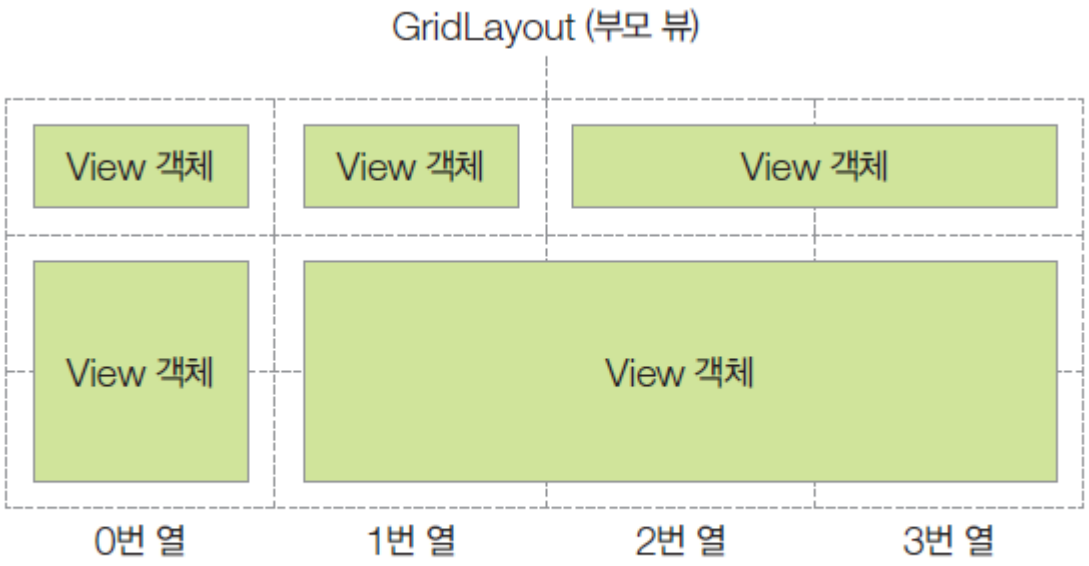
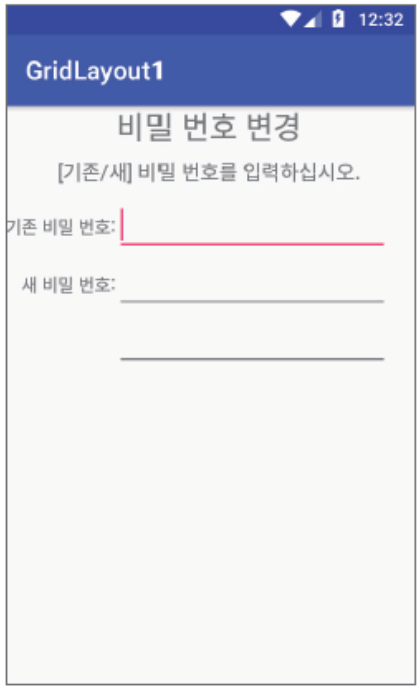


그림 2-21 그리드 레이아웃

실습 2-12 GridLayout1

- 그리드 레이아웃을 활용하여 비밀 번호 변경 화면 구성하기

(a) 컴포넌트 트리



(b) 실행 화면

그림 2-22 컴포넌트 트리와 실행 화면

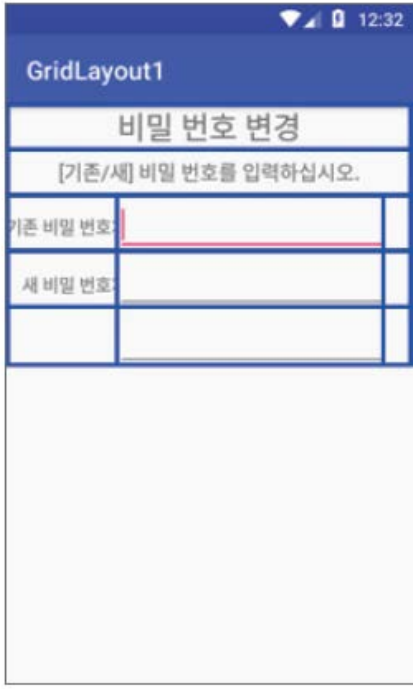


그림 2-23 격자 구조

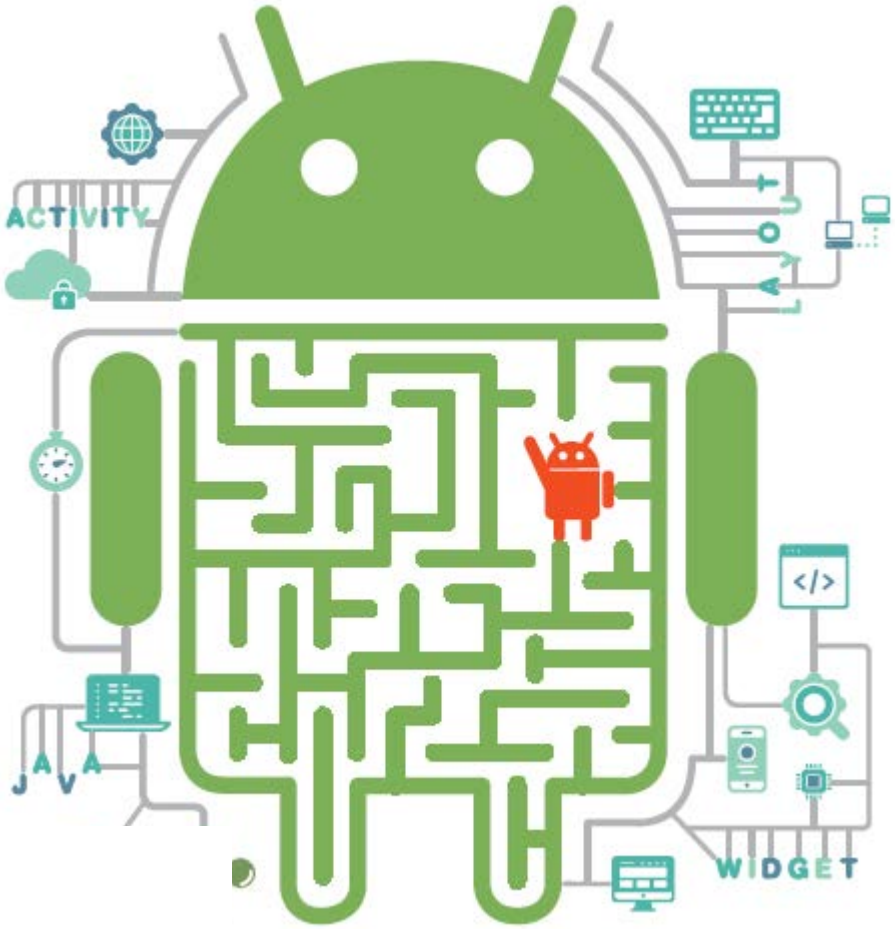
실습 2-12 GridLayout1

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:columnCount="3">
6       <TextView
7           android:layout_column="0"
8           android:layout_columnSpan="3"
9           android:layout_gravity="center_horizontal"
10          android:layout_row="0"
11          android:text="비밀번호 변경"
12          android:textSize="24sp"/>
13      <TextView
14          android:layout_column="0"
15          android:layout_columnSpan="3"
16          android:layout_gravity="center_horizontal"
17          android:layout_margin="8dp"
18          android:layout_row="1"
19          android:text="[기존/새] 비밀번호를 입력하십시오."
20          android:textSize="16sp"/>
21      <TextView
22          android:layout_gravity="right"
23          android:text="기존 비밀번호:"/>
```

실습 2-12 GridLayout1

```
24    <EditText
25          android:ems="10"
26          android:inputType="textPassword"/>
27       <TextView
28          android:layout_column="0"
29          android:layout_gravity="right"
30          android:text="새 비밀번호:"/>
31       <EditText
32          android:ems="10"
33          android:inputType="textPassword"/>
34       <EditText
35          android:layout_column="1"
36          android:ems="10"
37          android:inputType="textPassword"/>
38    </GridLayout>
```



단계별로 배우는

안드로이드 프로그래밍