

텍스트 출력과 레이아웃



개발환경 구축 절차

2

주 차	수 업 내 용
1	수업 소개
2	개발 환경 구축과 맛보기 프로젝트
3	텍스트 출력과 레이아웃
4	이미지 출력
5	이벤트 처리와 액티비티 간 이동
6	오디오 재생
7	비디오 재생
8	중간고사
9	애니메이션
10	사물인터넷과 센서 - 터치 센서, 모션 센서
11	사물인터넷과 센서 - 위치 센서, 환경 센서
12	NFC 활용
13	공공 DB 오픈 API 활용
14	구글 맵과 위치 추적
15	기말 고사



Lecture Notes: <https://goo.gl/vSdwLq>

3

The screenshot shows a web browser window displaying the GitHub repository page for 'hopypark / LectureNotes'. The browser's address bar shows the URL 'https://github.com/hopypark/LectureNotes/tree/master/AndroidApp'. The repository page includes a header with the repository name, a search bar, and navigation links like 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the header, there are tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Settings', and 'Insights'. The 'Code' tab is selected, showing a list of files and folders. A blue box highlights the file 'Week03.Chap06.텍스트 출력과 레이아웃.pdf'. The file list includes 'README.md', 'Week02.Chap02.앱 개발환경 구축.pdf', 'Week02.Chap03.앱 프로젝트 구조와 실행원리.pdf', 'Week03.Chap06.텍스트 출력과 레이아웃.pdf', and 'ic_launcher_new-web.png'. The file 'Week03.Chap06.텍스트 출력과 레이아웃.pdf' is the latest commit, made 'just now'.

Branch: master ▾ LectureNotes / AndroidApp /

Create new file Upload files Find file History

hopypark committed on GitHub Week03.Chap06.텍스트 출력과 레이아웃 ... Latest commit f1ad8b8 just now

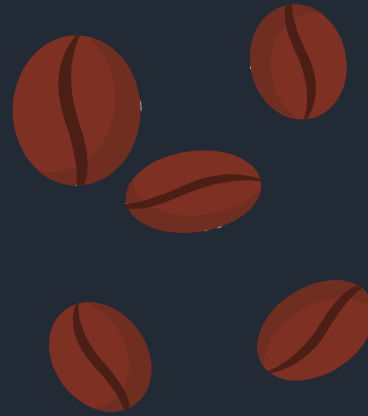
..

README.md	Create readme.md	14 days ago
Week02.Chap02.앱 개발환경 구축.pdf	Add files via upload	14 days ago
Week02.Chap03.앱 프로젝트 구조와 실행원리.pdf	Add files via upload	7 days ago
Week03.Chap06.텍스트 출력과 레이아웃.pdf	Week03.Chap06.텍스트 출력과 레이아웃	just now
ic_launcher_new-web.png	Add files via upload	10 days ago

README.md

심터

- XML의 이해
- 자바클래스의 이해





- XML(eXtensible Markup Language*)은 W3C(인터넷 표준제정 단체)에서 제안한 사람과 기계가 읽을 수 있는 형태의 문서를 만들 수 있는 규칙들의 집합
- 인터넷에 연결된 시스템끼리 데이터를 쉽게 주고 받을 수 있게 할 목적
- XML 문서는 선언부분과 엘리먼트(Element)들로 구성된다.

*마크업 언어(Markup Language): 태그 등을 이용하여 문서나 데이터의 구조를 명기하는 언어의 한 가지
XML: <https://ko.wikipedia.org/wiki/XML>

• 선언 부분

- XML 문서 저장시의 인코딩에 이용되는 문자 코드셋과 XML 버전을 지정

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

• 엘리먼트

- XML은 하나의 root element를 가지며, 하나 이상의 하위 child element를 가진다.
- child element도 하나 이상의 하위 child element를 가질 수 있다.

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

- 데이터가 없는 경우 empty element라고 하고, <element></element> 또는 </element>로 표현
- element는 여러 개의 속성을 지정할 수 있으며, 속성명과 속성값으로 표현

```
<element 속성명="속성값">
```

- XML의 예시 (<https://www.w3schools.com/xml/default.asp>)

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```


네임스페이스(NameSpace)

9

- (개인)정보 - <http://www.inje.ac.kr/abc/ML>

```
<?xml version="1.0" encoding="euc-kr"?>
<개인정보>
  <주민번호>123453-122241</주민번호>
  <이름>홍길동</이름>
  <이메일>aaaa@gmail.com</이메일>
  <주소>부산 광역시</주소>
</개인정보>
```

- 고객정보 - <http://www.inje.ac.kr/123/ML>

```
<?xml version="1.0" encoding="euc-kr"?>
<정보>
  <주민번호>123453-122241</주민번호>
  <이름>홍길동</이름>
  <이메일>aaaa@gmail.com</이메일>
</정보>
```

- 두 XML 문서를 병합할 때, 같은 이름의 element 구분은



- **NameSpace**는 W3C에서 문서의 element 속성 이름을 유일하게 구분할 수 있도록 고안
- NameSpace의 형태는 **URI(Uniform Resource Identifier)**로 표현된다.
 - URI는 웹페이지에서 사용하고 있으며, URL은 URI의 일부분
- URI 표현 방법

```
<엘리먼트명 xmlns:접두어="네임스페이스이름">  
  <접두어:엘리먼트명></접두어:엘리먼트명>  
</엘리먼트명>
```

- URI 표현 예시

```
<주문정보 xmlns:정보="http://inje.ac.kr/abc/ML"  
  xmlns:고객정보="http://inje.ac.kr/123/ML">  
  <정보:주민번호>123453-122241</정보:주민번호>  
  <정보:이름>홍길동</정보:이름>  
</주문정보>
```



- 프로그램 작성 언어
- 기계어(machine language)
 - 0, 1의 이진수로 구성된 언어
 - 컴퓨터의 CPU는 기계어만 이해하고 처리가능
- 어셈블리어
 - 기계어 명령을 ADD, SUB, MOVE 등과 같은 표현하기 쉬운 상징적인 단어인 **니모닉 기호(mnemonic symbol)**로 일대일 대응시킨 언어
- 고급언어
 - 사람이 이해하기 쉽고, 복잡한 작업, 자료 구조, 알고리즘을 표현하기 위해 고안된 언어
 - Pascal, Basic, C/C++, **Java**, C#
 - 절차 지향 언어와 **객체 지향 언어**로 나눌 수 있음

자바의 플랫폼 독립성

13

Write Once !!



자바
응용 프로그램

Run Anywhere!!

실행



자바 가상 기계

인텔 CPU + 리눅스

실행



자바 가상 기계

Apple 사의 MAC PC

실행



자바 가상 기계

인텔 CPU + 윈도우 노트북

- **객체**: 실세계에 존재하는 다른 것과 구별되는 추상 또는 구체적인 것
 - 사람, 자동차, TV, ...
- 객체는 상태와(state)와 행동(behavior)
 - 사람의 상태: 성별, 키, 몸무게, ...
 - 사람의 행동: 먹는다, 생각한다, 말한다, 걷는다, ...
- **클래스**: 객체(사람, 자동차)의 상태와 행동을 기술한 것
 - 객체의 상태는 클래스의 속성(Attribute)
 - 객체의 행동은 메소드(Method)로 기술

객체-사람



상태

- 성별: 남자
- 키: 180 cm
- 몸무게: 80 kg

행동

- 일한다.
- 먹는다.

클래스(CLASS)

class 사람{

Attribute

gender= 남자
height = 180
weight = 80

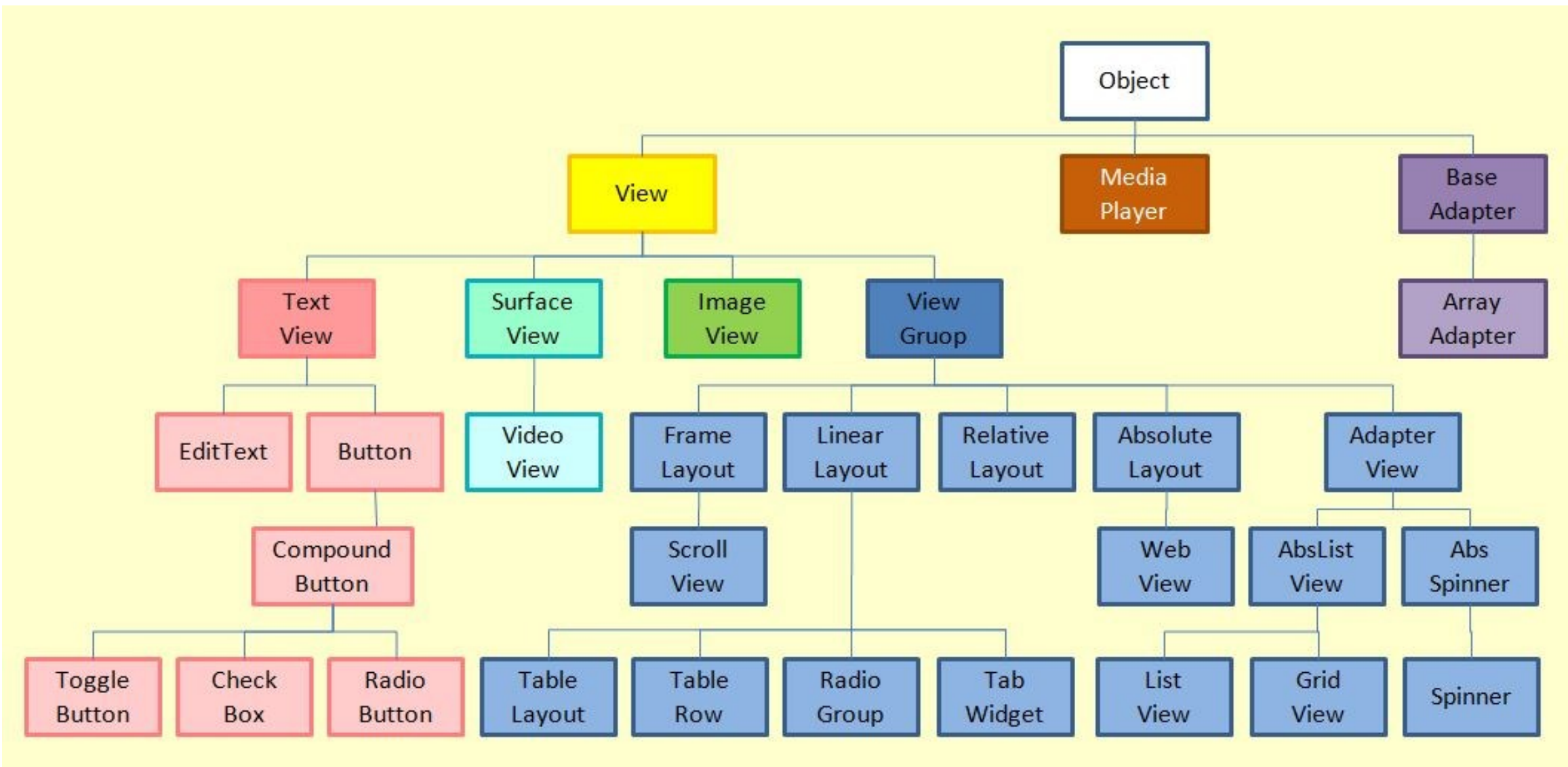
Method

working()
eating()

}

안드로이드 자바 뷰(View) 클래스 계층도

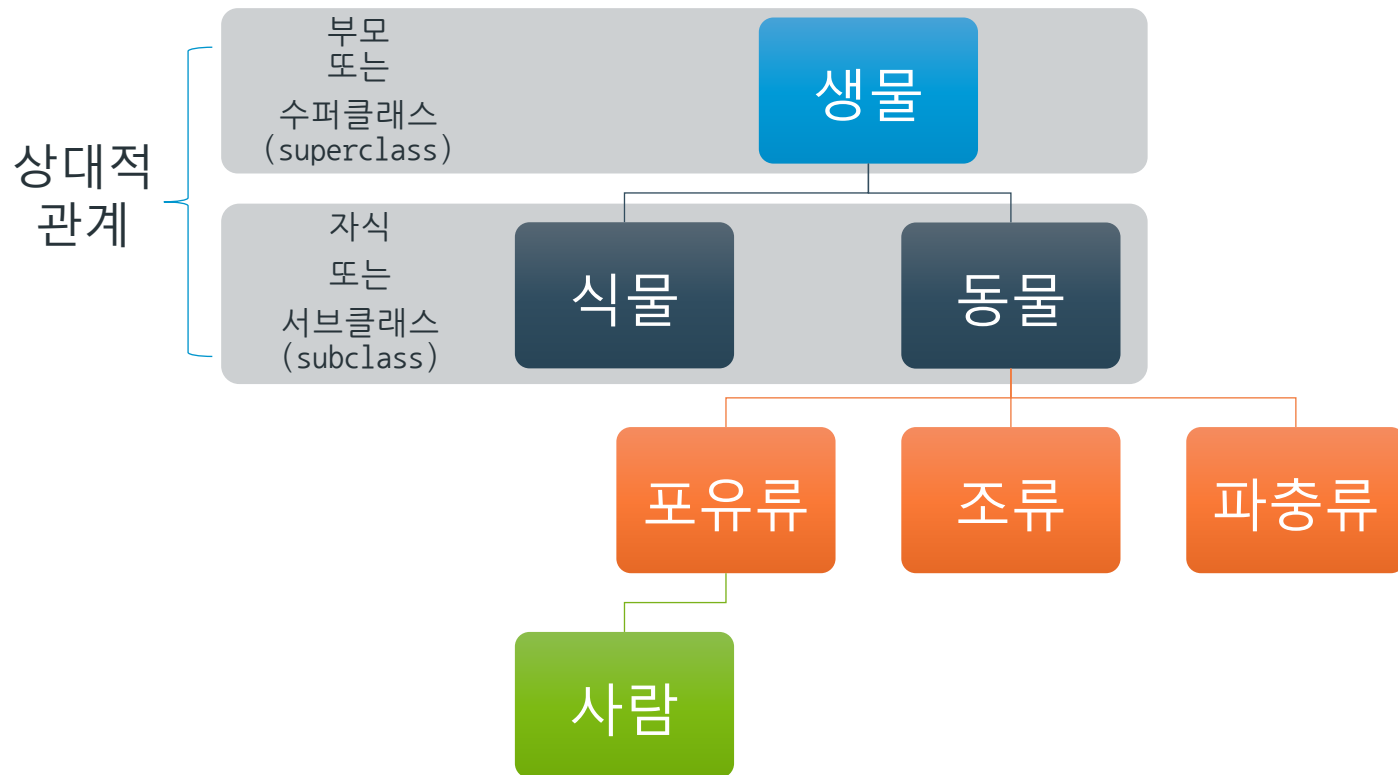
16



안드로이드 자바 클래스 계층도[상속]

17

- 자바는 상속이다.
- 상속이라는 것은 **is a** 관계가 성립한다.
- “**자식 is a 부모다**” 라고 말할 수 있어야 상속관계가 성립
- 자식은 부모의 모든 것(**생성자 제외**)을 물려 받는다.



class 사람 **extends** 포유류{

Attribute

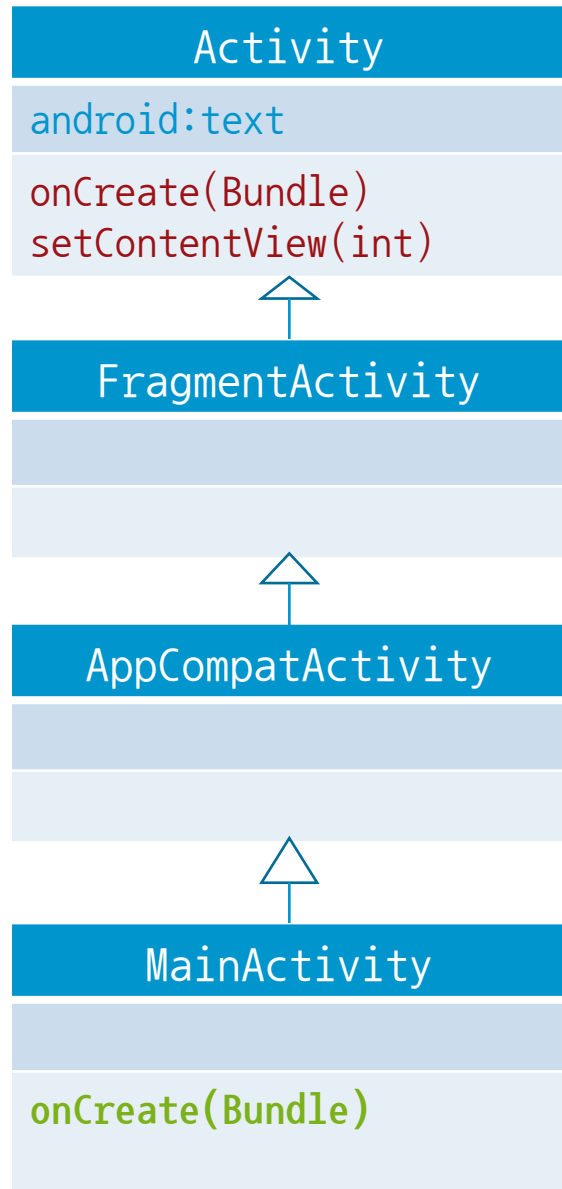
gender: 남자
height: 180 cm
weight: 80 kg

Method

working()

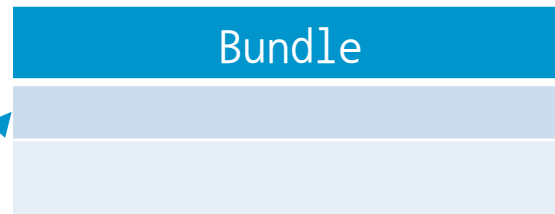
}

• MainActivity 클래스 다이어그램



- MainActivity 클래스는 Activity 클래스로부터 상속받은 AppCompatActivity 클래스로부터 상속 받으며, onCreate(Bundle) 메소드를 재정의 (override)하여 사용(녹색)

- MainActivity 클래스는 Bundle 클래스를 사용



- **Bundle**은 상태/값 등을 저장하기 위한 객체

- **Intent**는 저장이 아닌 전달하는 수단으로의 객체

예)

- 택배차량 – intent
- 택배물건 – bundle

• MainActivity.java

19

```
1 package com.example.kyungtae.poems;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5
6 public class MainActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13 }
14
```

MainActivity 클래스는 AppCompatActivity 클래스(수퍼클래스)로부터 상속받아 정의함

수퍼 클래스 AppCompatActivity 클래스의 onCreate()를 이용하여 액티비티 생성

@Override는 수퍼클래스로부터 상속받은 메소드를 재정의한다는 의미

- 클래스와 속성/메소드(MainActivity.java)

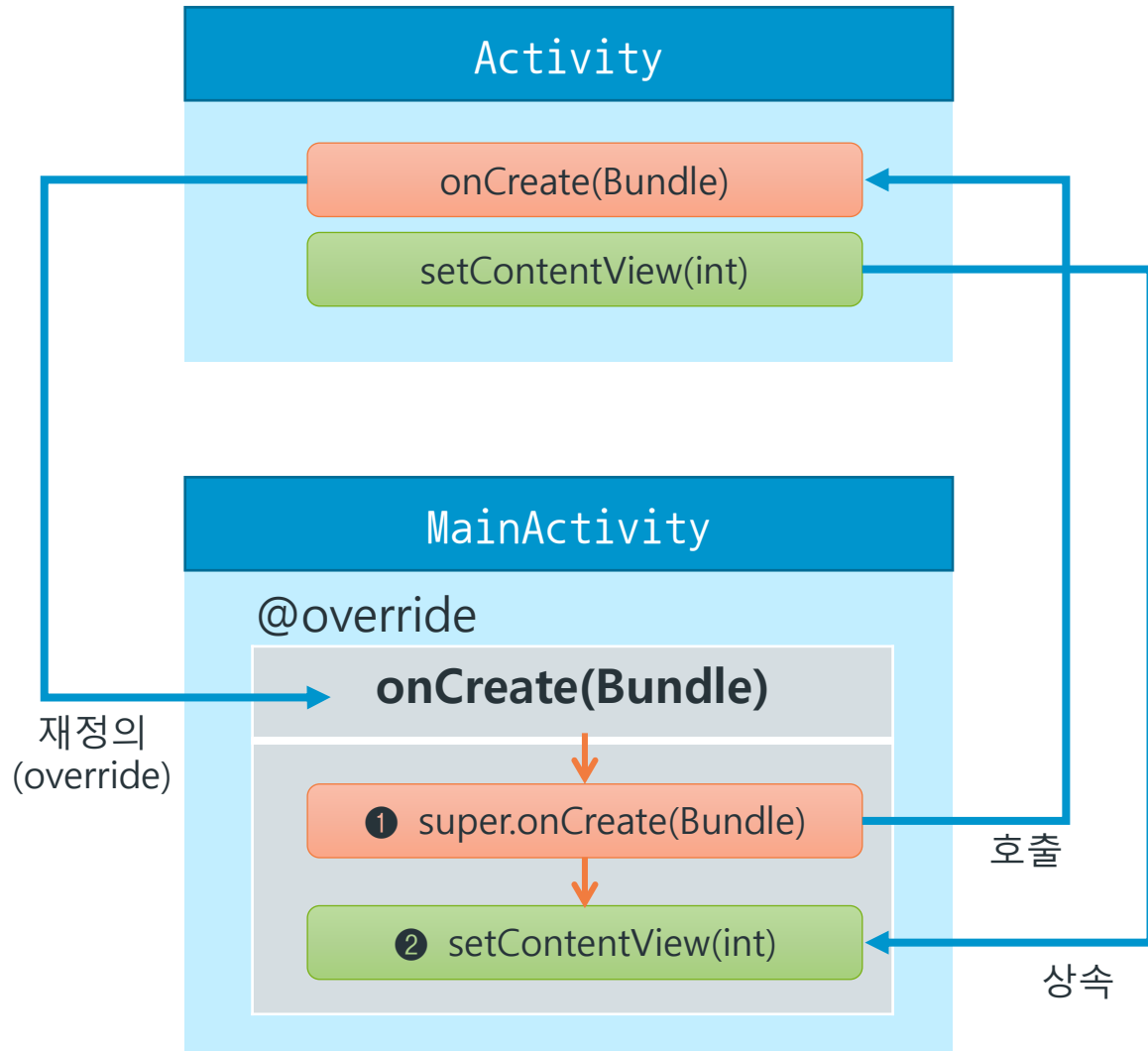
- 클래스

클래스	설명
Activity	사용자에게 윈도우 화면을 출력함
AppCompatActivity	액션 바를 사용하는 기능을 제공함
Bundle	문자열을 다양한 형태의 메시지를 담는 컨테이너로 매핑함

- 메소드

클래스		
Activity	void onCreate(Bundle savedInstanceState)	액티비티를 생성함. Bundle 클래스는 액티비티가 갑자기 정지(shut down)될 때 상태 정보를 가지고 있다. 액티비티가 다시 초기화될 때 활용되는 역할을 함
	void setContentView(int layoutResId)	레이아웃을 출력함(layoutResId는 레이아웃이 정의된 xml 파일의 ID를 의미함)

• MainActivity 클래스의 실행원리



- MainActivity 클래스가 실행되면 **onCreate() 메소드가 자동 호출**된다. onCreate() 메소드에는 액티비티가 수행될 절차를 기술한다.

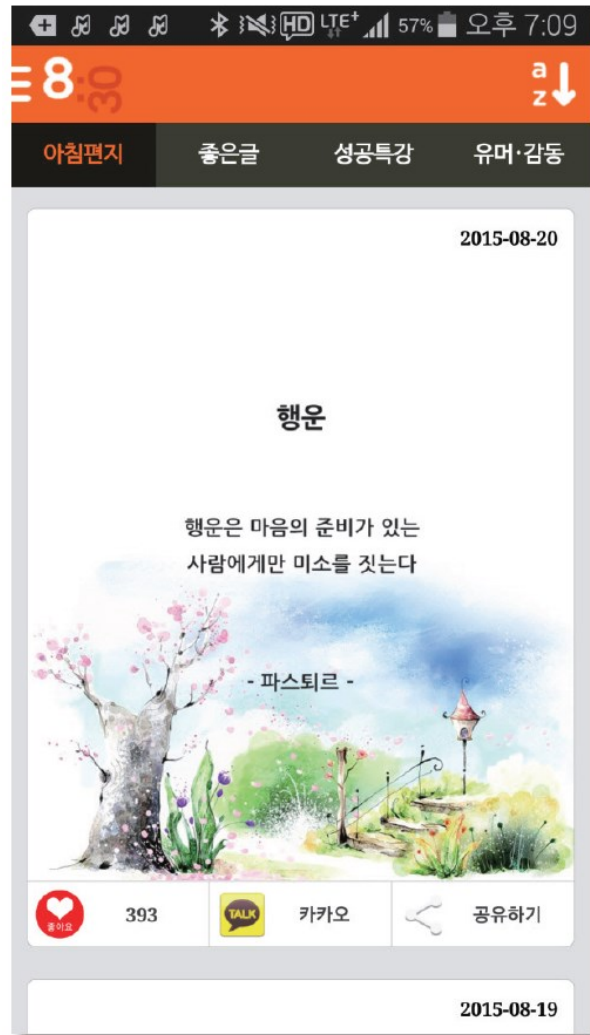
① 수퍼 클래스인 AppCompatActivity 클래스의 onCreate() 메소드를 이용 (super.onCreate())하여 액티비티를 생성함

② Activity 클래스로부터 상속받은 setContentView() 메소드를 이용하여 레이아웃을 출력함

- 텍스트 활용 앱과 출력 원리
- 텍스트 출력과 폰트의 변화

텍스트 출력과 레이아웃

- 명언 공지 앱: 여덟시삼십분

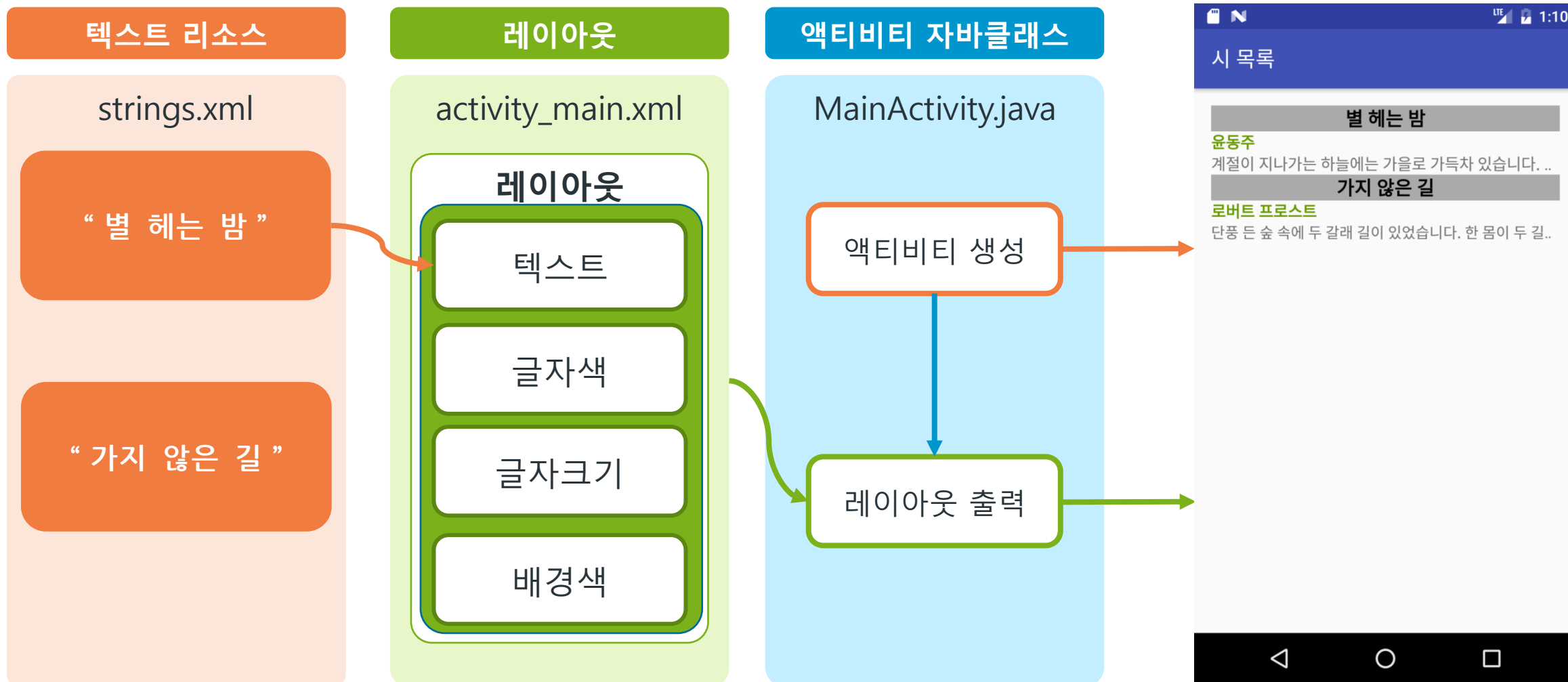


- 명언 공지 앱: 여덟시삼십분

텍스트 활용 앱과 출력 원리

24

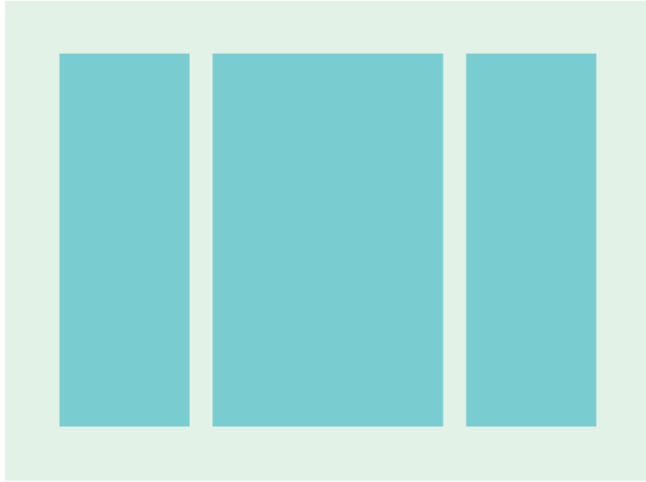
- 텍스트는 정해진 레이아웃에 따라 출력된다.



레이아웃(Layout)

25

- 레이아웃 유형



(a) 리니어 레이아웃

수평 또는 수직 방향의 화면 배치(화면 길이 초과 시는 스크롤바가 나타남)



(b) 렐러티브 레이아웃

개체들 간의 상대적인 위치에 의한 배치



(c) 웹뷰

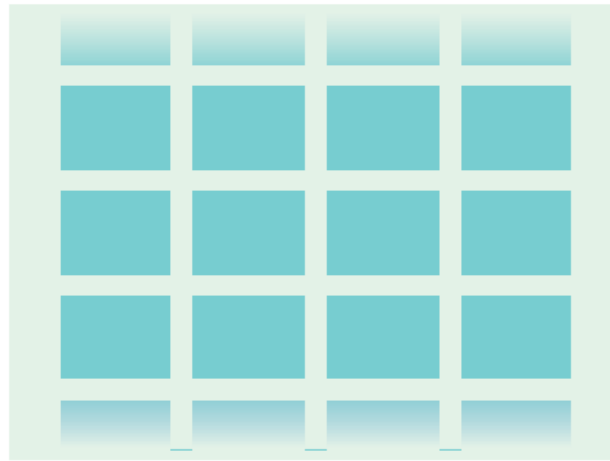
웹 문서의 출력

- 이미지: http://kunny.github.io/lecture/ui/2016/05/22/constraint_layout_1/



(a) 리스트 뷰

단일 열의 목록 출력(수직 방향의 화면 길이 초과 시는 스크롤바가 나타남)



(b) 그리드 뷰

정해진 수의 열과 행으로 구성되는 격자 모양의 출력(화면 길이 초과 시는 스크롤바가 나타남)

- 어댑터를 이용한 레이아웃, 자료원: developer.android.com

TIP 어댑터(Adapter)

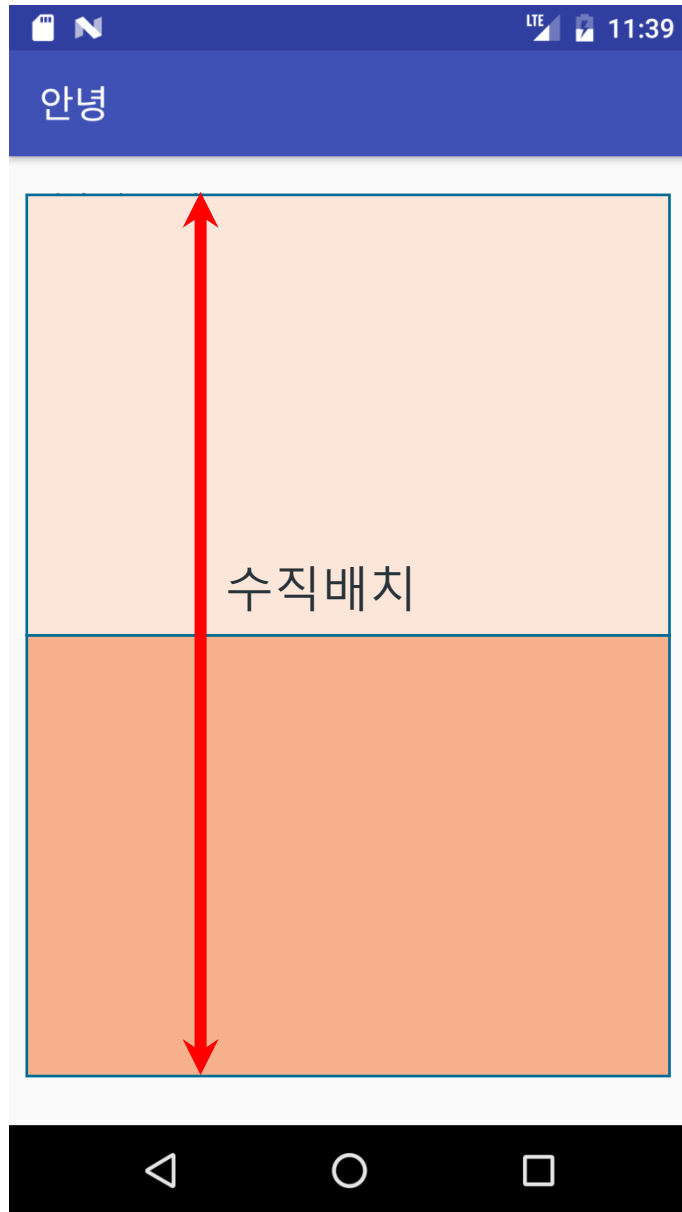
어댑터 (Adapter)란 레이아웃과 그 레이아웃에 출력될 데이터를 바인딩하는 클래스를 의미한다.

어댑터: 다른 전기나 기계 장치를 서로 연결해서 작동할 수 있도록 만들어 주는 결합 도구 (<https://ko.wikipedia.org/wiki/%EC%96%B4%EB%8C%91%ED%84%B0>)

레이아웃(Layout) – LinearLayout

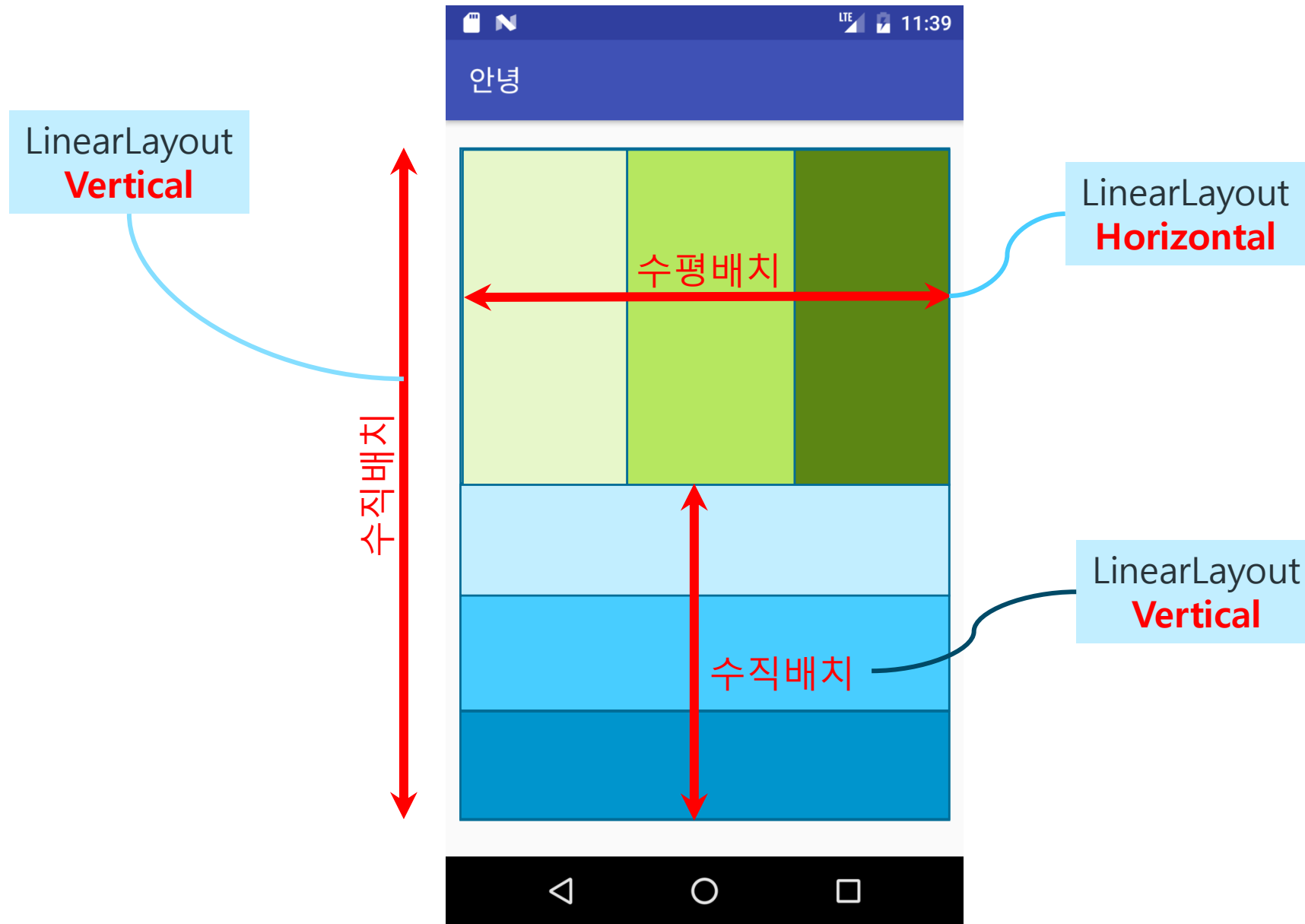
27

LinearLayout
Vertical



LinearLayout
Horizontal





- ConstraintLayout - <https://developer.android.com/training/constraint-layout/index.html>
- RelativeLayout - <http://lsit81.tistory.com/entry/Android-RelativeLayout-%EB%B0%B0%EC%B9%98-%EA%B4%80%EB%A0%A8>
- RelativeLayout : <https://www.android-tech.io/2016/01/19/%EB%91%98%EC%A7%B8%EB%A7%88%EB%8B%B9-04%EC%9E%A5-02%EC%A0%88-%EB%A0%90%EB%9F%AC%ED%8B%B0%EB%B8%8C%EB%A0%88%EC%9D%B4%EC%95%84%EC%9B%83/>



Layout Editor 소개

31

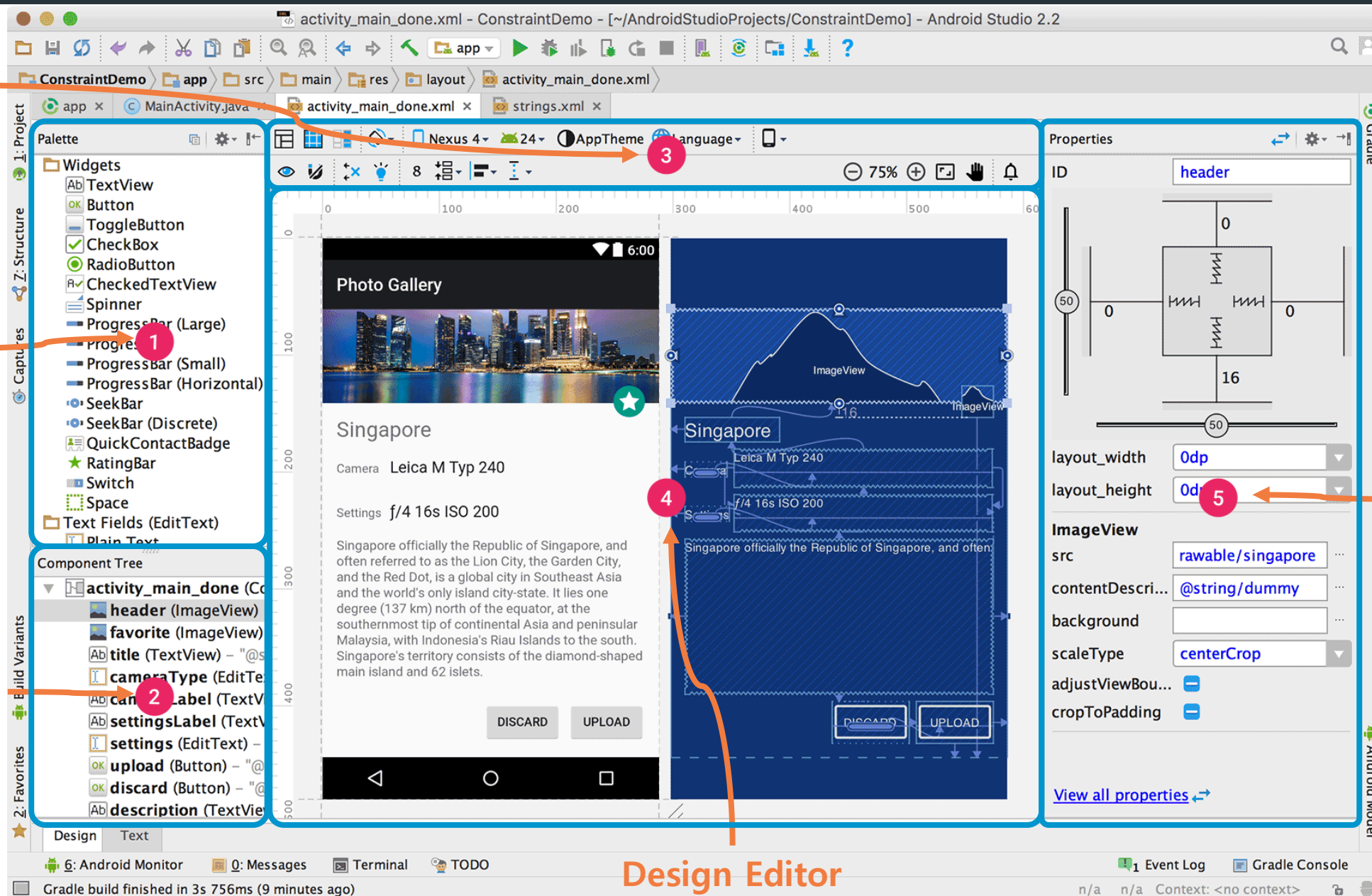
Toolbar

Palette

Component Tree

Properties

Design Editor



• Layout Editor 소개

01

Palette

편집기에서 레이아웃으로 드래그할 수 있는 위젯 및 레이아웃의 목록을 제공

02

Component Tree

레이아웃의 뷰 계층 구조를 표시. 여기서 항목을 클릭하면 편집기에 선택한 항목이 표시.

03

Toolbar

편집기에서 레이아웃 모양을 구성하고 레이아웃 속성을 편집할 수 있는 버튼을 제공

04

Design Editor

Design 및 Blueprint 뷰가 결합된 형태로 레이아웃을 표시

05

Properties

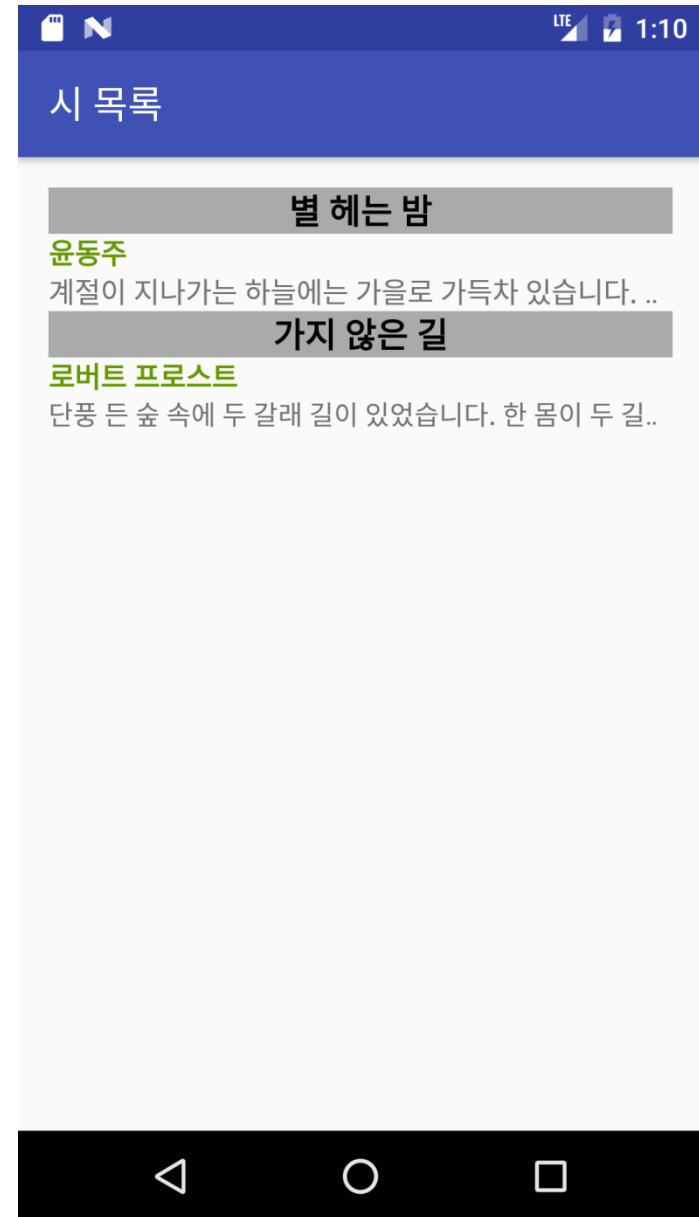
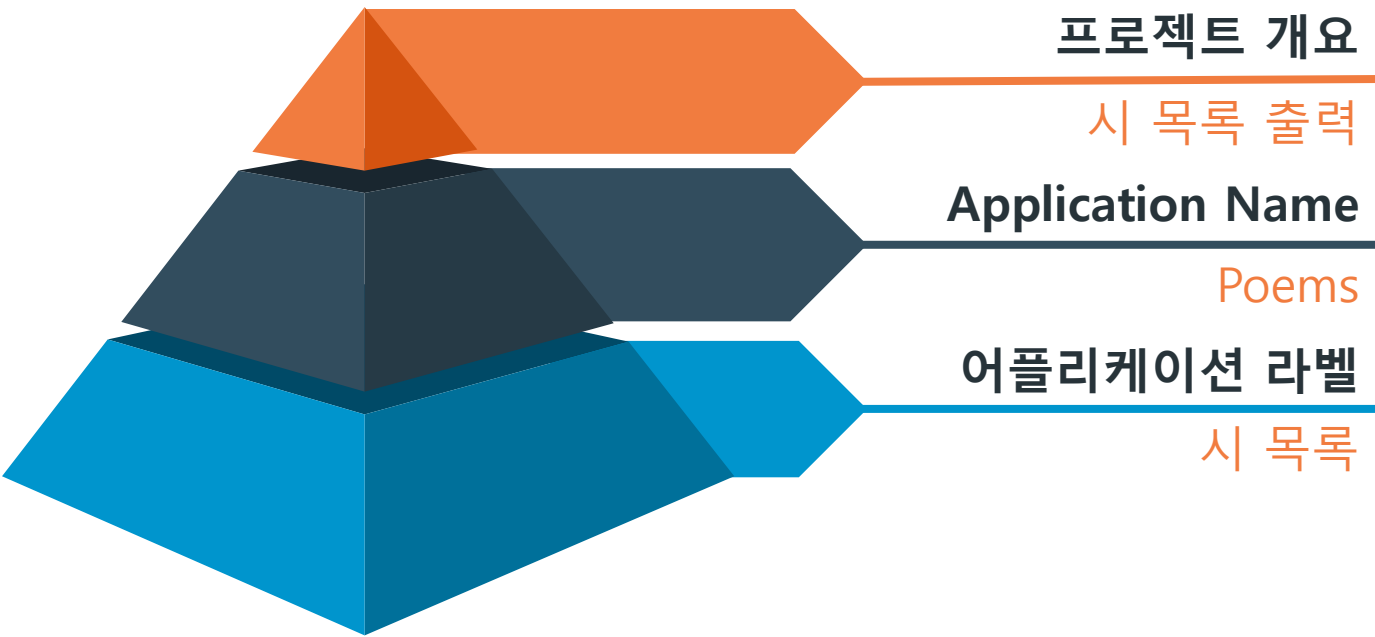
현재 선택된 뷰에 대한 속성 제어를 제공



텍스트 출력과 폰트의 변화

Step 0.프로젝트 개요

34



Step 1. 프로젝트 생성

절차	내용
①프로젝트 시작	메뉴에서 ‘ File → New Project ’ 클릭
②프로젝트 구성	Application Name: Poems
	Company Domain: 사용자계정.example.com (디폴트 사용)
③제품 형태	Phone and Tablet (사용할 안드로이드 버전 지정: Android 4.4 kitkat)
④액티비티 유형	Empty Activity
⑤파일 옵션	Activity Name: MainActivity
	Layout Name: activity_main

Step 2. 파일 편집

• 파일 구조와 편집 내용

모듈(App)	폴더	소스 파일	편집 내용
manifests		AndroidManifest.xml	
java	com.example.kyungtae.poems	MainActivity.java	
res	drawable		
	layout	activity_main.xml	• 두 편의 시에 대한 출력화면 배치
	mipmap	ic_launcher.png	
	values	dimens.xml	
		strings.xml	• 어플리케이션 라벨 수정 • 두 편의 시를 출력하는데 사용되는 텍스트 리소스 정의
		styles.xml	

수정

앱 라벨

시 제목

작가

시 본문

화면 구성 자원 크기

`dimen``activity_horizontal_margin: 16dp`

dimens.xml (values)

아이콘 이미지



ic_launcher.png (mipmap)

텍스트 자원

`string``app_name: 시 목록1``title01: 별 해는 밤``author01: 윤동주``body01: 계절이 ...`

strings.xml (values)

화면 테마

`style``AppTheme:``Theme.AppCompat.Light.DarkActionBar``item``colorPrimary:``@color/colorPrimary`

styles.xml (values)

화면 테마 구성 색상

`color``colorPrimary: #3F51B5`

colors.xml (values)

화면 레이아웃

`LinearLayout``android:paddingLeft:``@dimen/activity_horizontal_margin``TextView``text @string/title01``TextView``text @string/author01``TextView``text @string/body01`

activity_main.xml (layout)

화면 출력 소스

액티비티 제어

`onCreate``super.onCreate()``setContentView(R.layout.activity_main)`

MainActivity.java (layout)

어플리케이션 구성
액티비티의 자바 클래스

어플리케이션 기본 정보

`application``icon @mipmap/ic_launcher``label @string/app_name``theme @style/AppTheme``activity``name MainActivity`

AndroidManifest.xml (manifest)

컴파일/빌더

컴파일/빌더 정보

```
build.gradle(Project)
build.gradle(Module app)
gradle.properties
settings.gradle
local.properties
```

(Gradle Scripts)



Step 2.1 텍스트 리소스 편집(strings.xml)

38

① Edit translations for all locales in the translations editor.

```
1 <resources>
2   <string name="app_name">시 목록</string>
3
4   <string name="title01">별 헤는 밤</string>
5   <string name="author01">윤동주</string>
6   <string name="body01">
7     계절이 지나가는 하늘에는
8     가을로 가득차 있습니다.
9     나는 아무 걱정도 없이
10    가을 속의 별들을 다 헤일듯합니다.
11  </string>
12
13  <string name="title02">가지 않은 길</string>
14  <string name="author02">로버트 프로스트</string>
15  <string name="body02">
16    단풍 든 숲 속에 두 갈래 길이 있었습니다.
17    한 몸이 두 길을 가지 못하기에
18    탄식하며 한참을 서서
19    낮은 수풀로 꺾여 내려가는 길을
20    멀리 바라보았습니다.
21  </string>
22
23 </resources>
```

② app_name의 데이터를 '시 목록'으로 수정

③ 첫 번째 시에 대한 속성

④ 첫 번째 시의 제목

첫 번째 시의 저자

첫 번째 시의 본문

두 번째 시에 대한 속성

Step 2.2 화면 설계(activity_main.xml)

39

- android.support.constraint.ConstraintLayout → LinearLayout 으로 변경
- ~~android:orientation="vertical" 항목 추가~~

The image shows two side-by-side views of the `activity_main.xml` file in an IDE. The top view shows the original code using `android.support.constraint.ConstraintLayout`. The bottom view shows the modified code using `LinearLayout`. A red arrow points from the `ConstraintLayout` tag in the top view to the `LinearLayout` tag in the bottom view, indicating the replacement. The `android:orientation="vertical"` attribute has been removed from the bottom view.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   tools:context="com.example.research.myapplication.MainActivity">
8
9   <TextView
10     android:layout_width="wrap_content"
11     android:layout_height="wrap_content"
12     android:text="Hello World!"
13     app:layout_constraintBottom_toBottomOf="parent"
14     app:layout_constraintLeft_toLeftOf="parent"
15     app:layout_constraintRight_toRightOf="parent"
16     app:layout_constraintTop_toTopOf="parent" />
17
18 </android.support.constraint.ConstraintLayout>
19
20
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   tools:context="com.example.research.myapplication.MainActivity">
8
```

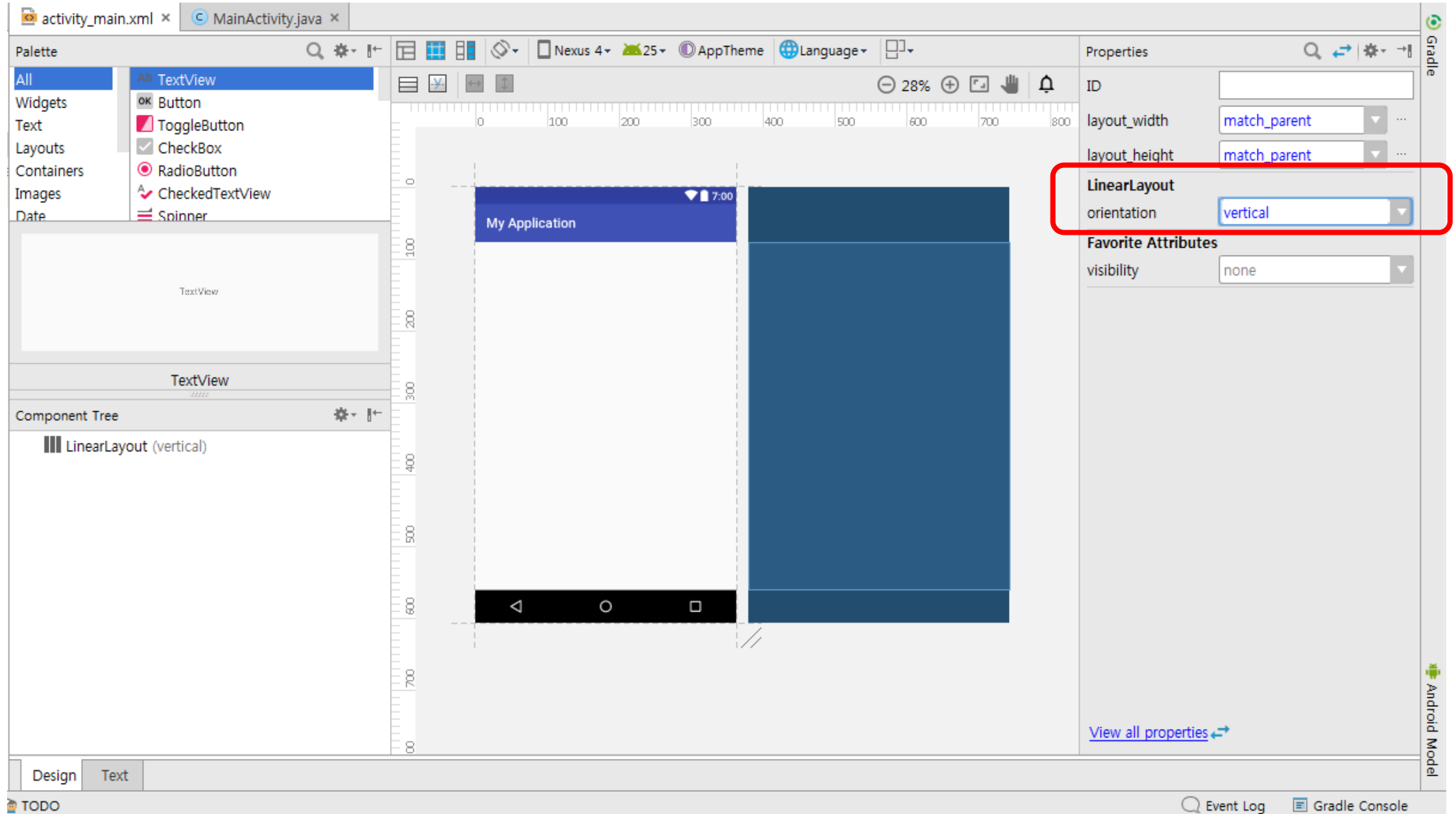
activity_main.xml x MainActivity.java x

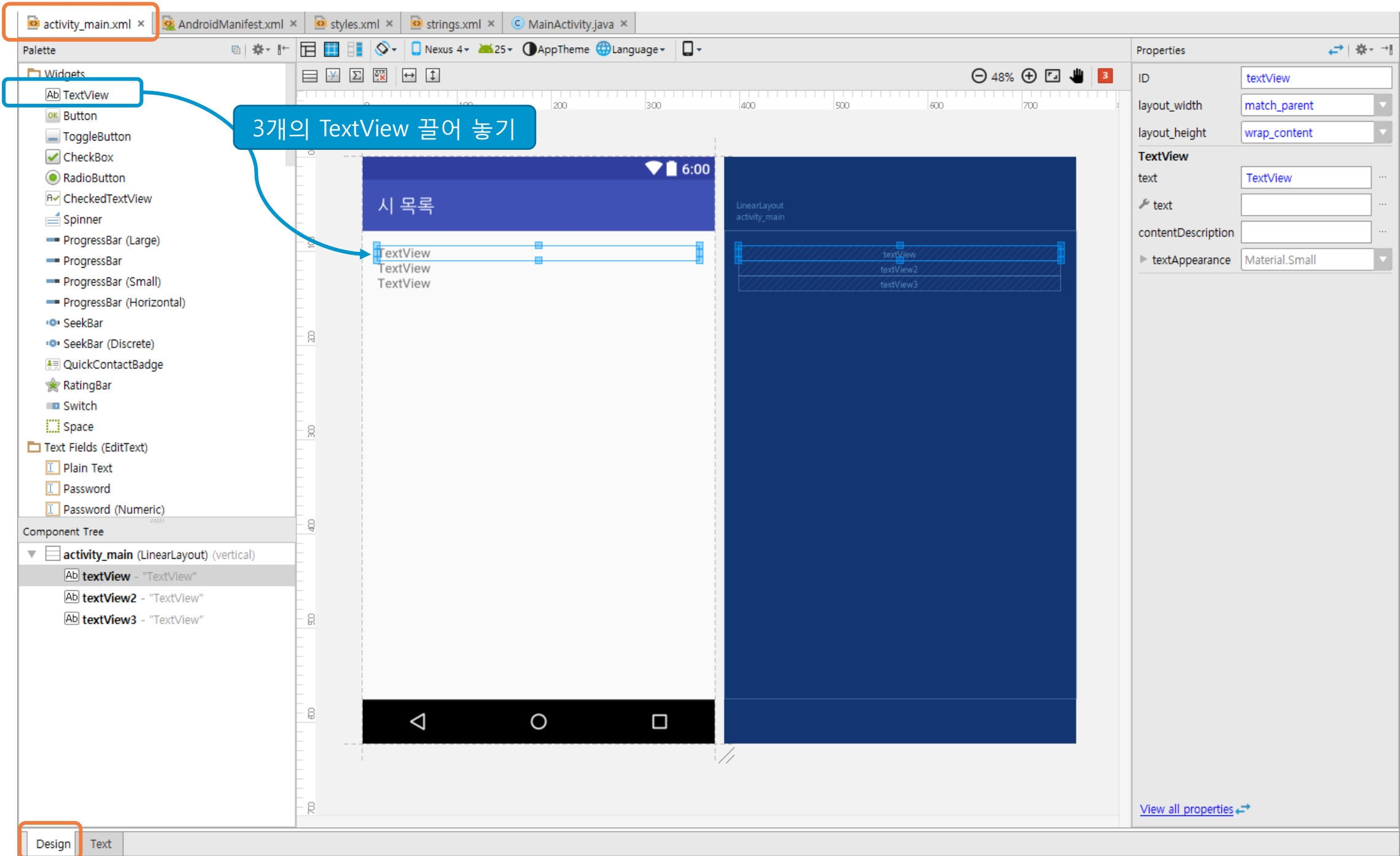
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   tools:context="com.example.research.myapplication.MainActivity">
8
9   <TextView
10     android:id="@+id/textView"
11     android:layout_width="wrap_content"
12     android:layout_height="wrap_content"
13     android:text="Hello World!"
14     app:layout_constraintBottom_toBottomOf="parent"
15     app:layout_constraintLeft_toLeftOf="parent"
16     app:layout_constraintRight_toRightOf="parent"
17     app:layout_constraintTop_toTopOf="parent" />
18
19 </LinearLayout>
```

삭제


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

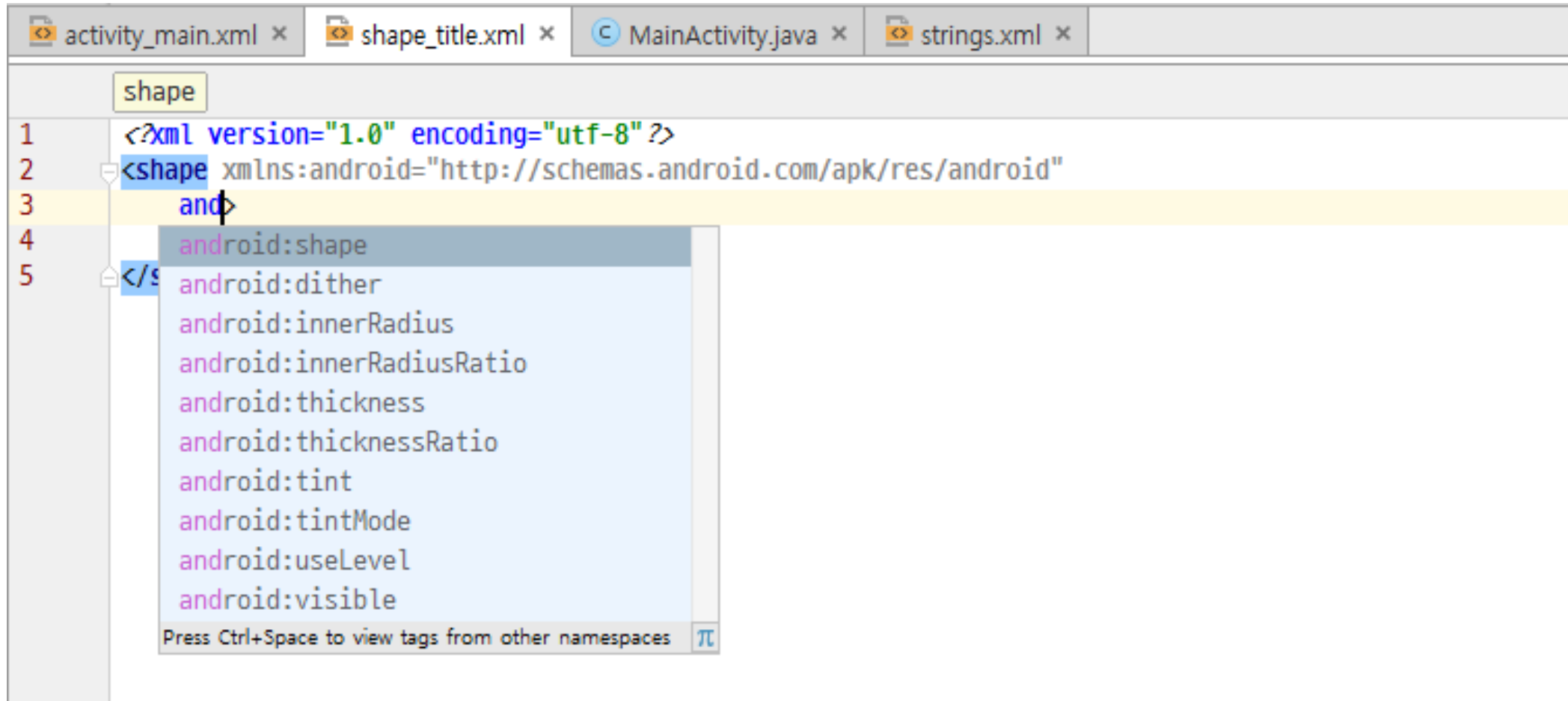
• LinearLayout의 orientation 속성 변경







- code completion 기능을 이용한 코드 작성



Step 2.2 화면 설계 – 속성 설정[첫 번째 시]

Properties

ID: textView

layout_width: match_parent

layout_height: wrap_content

TextView

text: TextView

contentDescription:

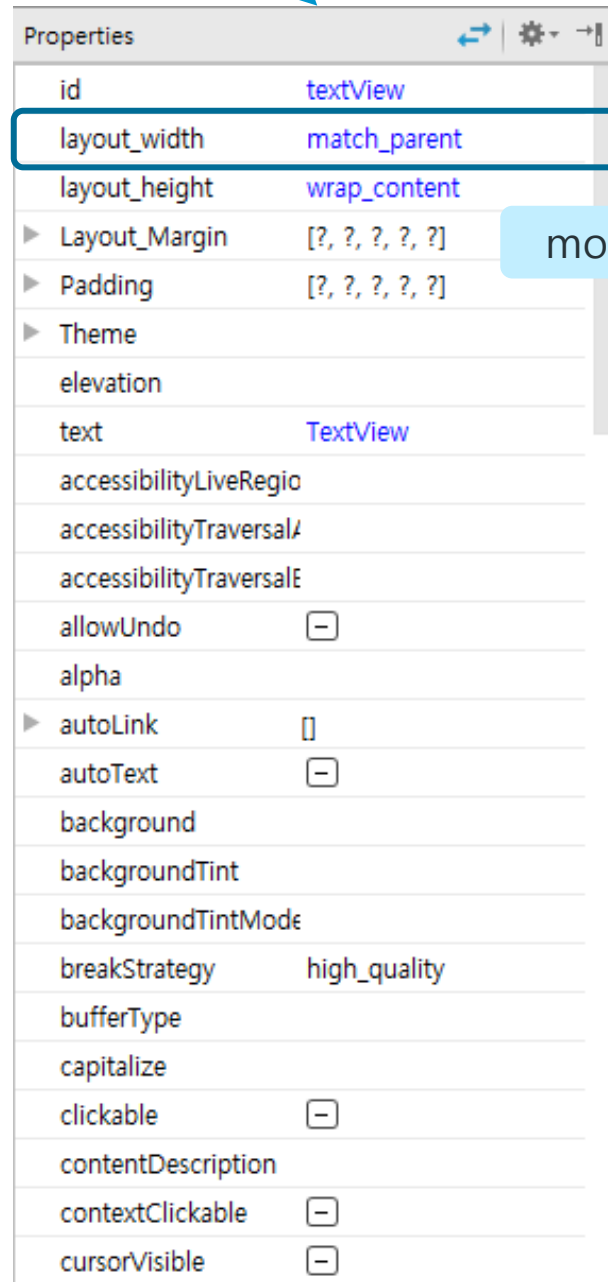
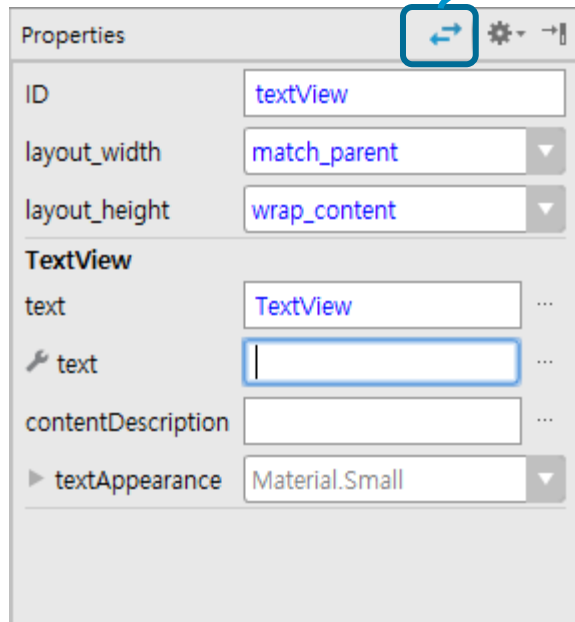
textAppearance: Material.Small

구성요소 간 구분자: textView

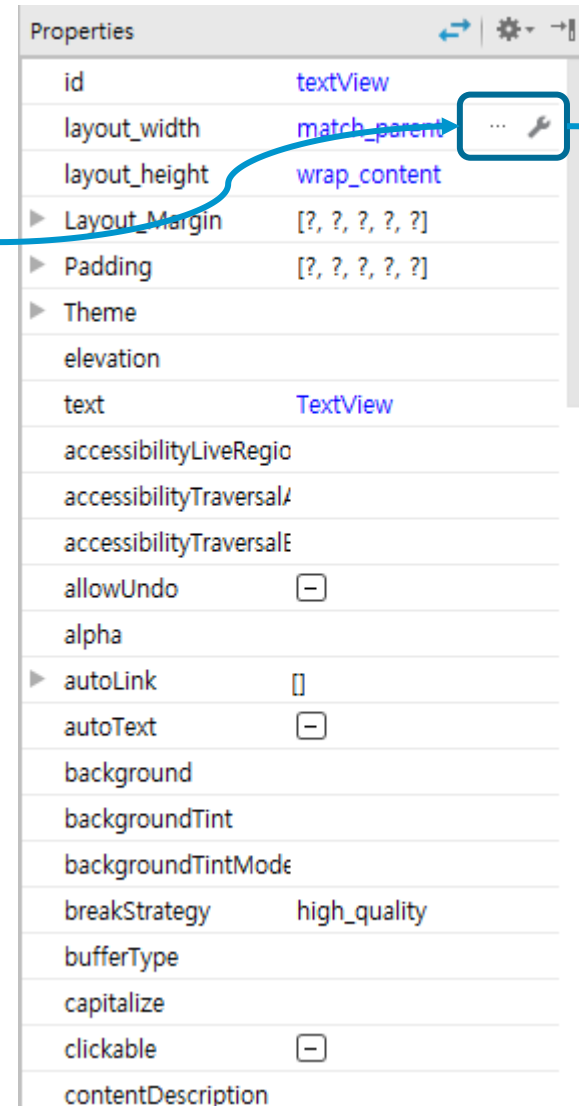
가로 폭 설정: match_parent

세로 폭 설정: wrap_content

표시할 텍스트: TextView



mouseover



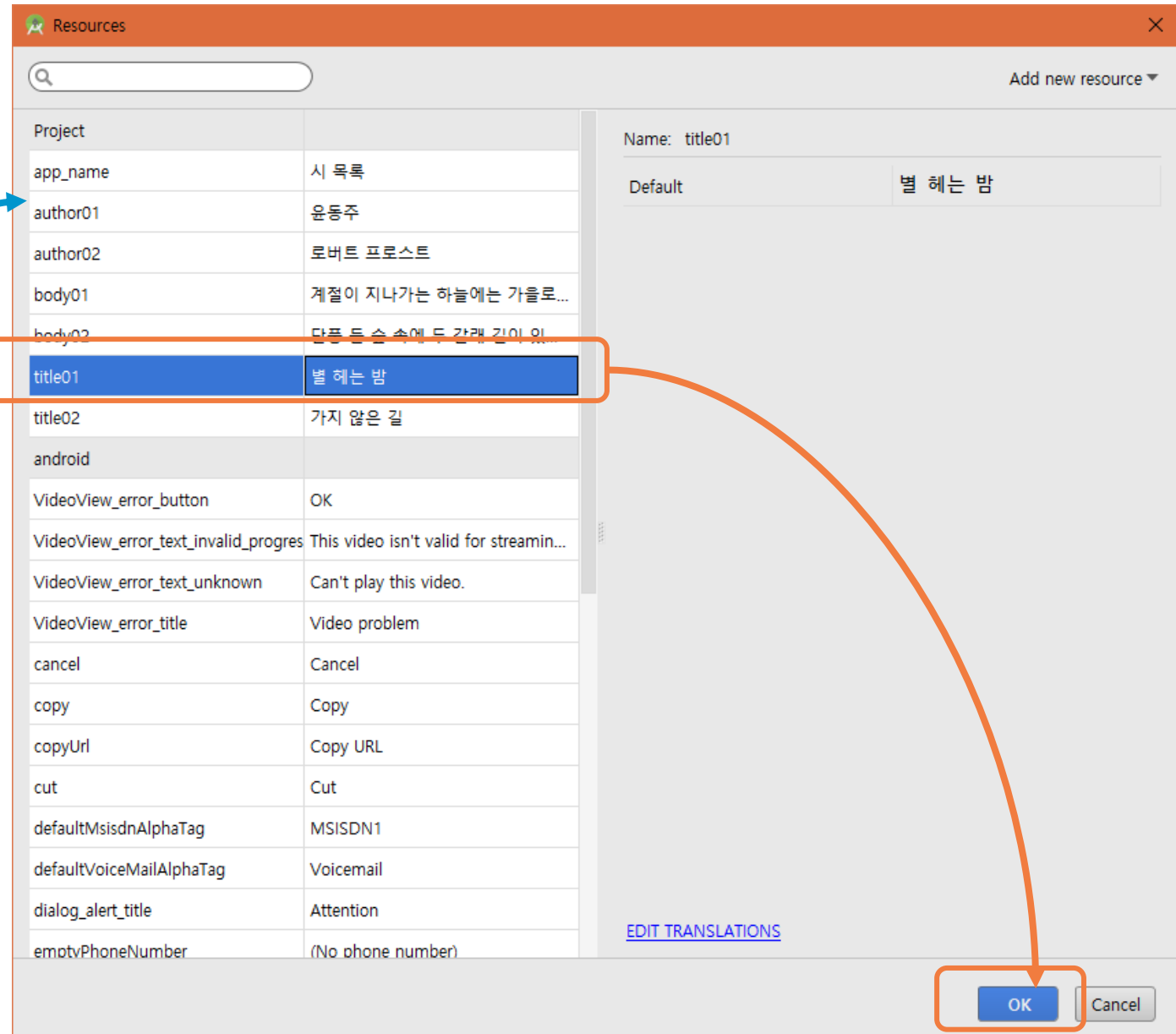
“...” click!!!

해당 속성과
관련된 설정창이
뜯는다.

첫 번째 TextView의 설정(title 설정)

- Text 설정

Properties	
id	textView
layout_width	match_parent
layout_height	wrap_content
Layout_Margin	[?, ?, ?, ?]
Padding	[?, ?, ?, ?]
Theme	
elevation	
text	TextView
accessibilityLiveRegion	
accessibilityTraversalAfter	
accessibilityTraversalBefore	
allowUndo	<input type="checkbox"/>
alpha	



The screenshot shows the Android Studio interface. On the left is the design view of a Nexus 4 device. A blue header bar at the top contains the text '시 목록' and status icons for Wi-Fi, battery, and time (6:00). Below the header, there are two TextView widgets. The first one contains the text '별 헤는 밤'. An orange box highlights this TextView, and an orange arrow points from it to the 'text' property in the Properties panel on the right. The Properties panel shows various attributes for the selected TextView, with 'text' set to '@string/title01'. A blue box with the text 'author, body 도 같은 방법으로 수정' (Modify author, body in the same way) has an arrow pointing to the second TextView widget below the first one.

Properties

id	textView
layout_width	match_parent
layout_height	wrap_content
Layout_Margin	[?, ?, ?, ?]
Padding	[?, ?, ?, ?]
Theme	
elevation	
text	@string/title01
accessibilityLiveRegion	
accessibilityTraversal	
accessibilityTraversalE	
allowUndo	<input type="checkbox"/>
alpha	
autoLink	<input type="checkbox"/>
autoText	<input type="checkbox"/>
background	
backgroundTint	
backgroundTintMode	
breakStrategy	high_quality
bufferType	
capitalize	
clickable	<input type="checkbox"/>

author, body 도 같은 방법으로 수정

결과 화면

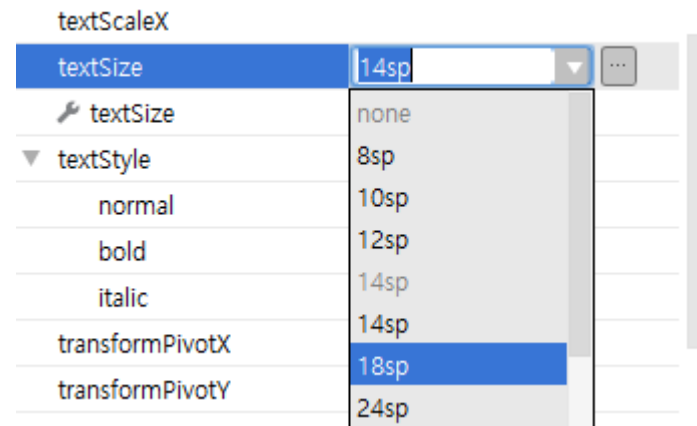
시 목록

별 헤는 밤
윤동주
계절이 지나가는 하늘에는 가을로 가득차 있습니다. 나는 아무 걱정도 없이 가을 속의 별들을 다 헤일듯합니다.

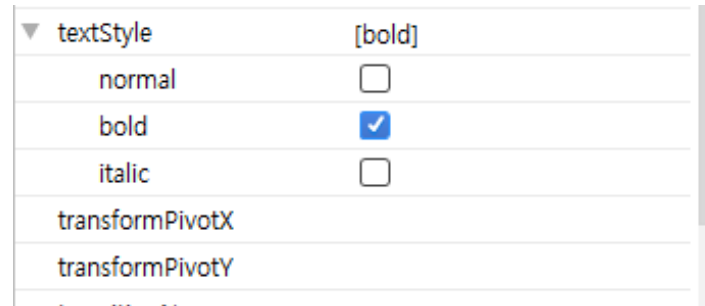
• 제목 속성 변경

- textSize, textStyle, textColor, background, gravity 등을 설정

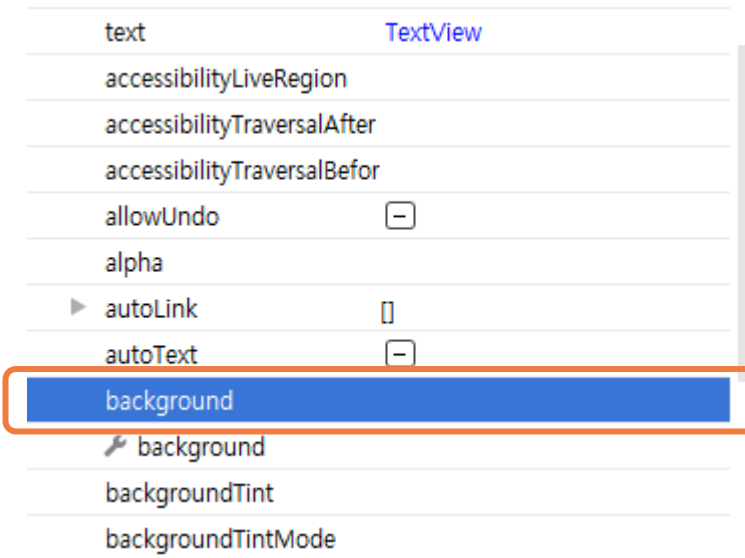
textSize

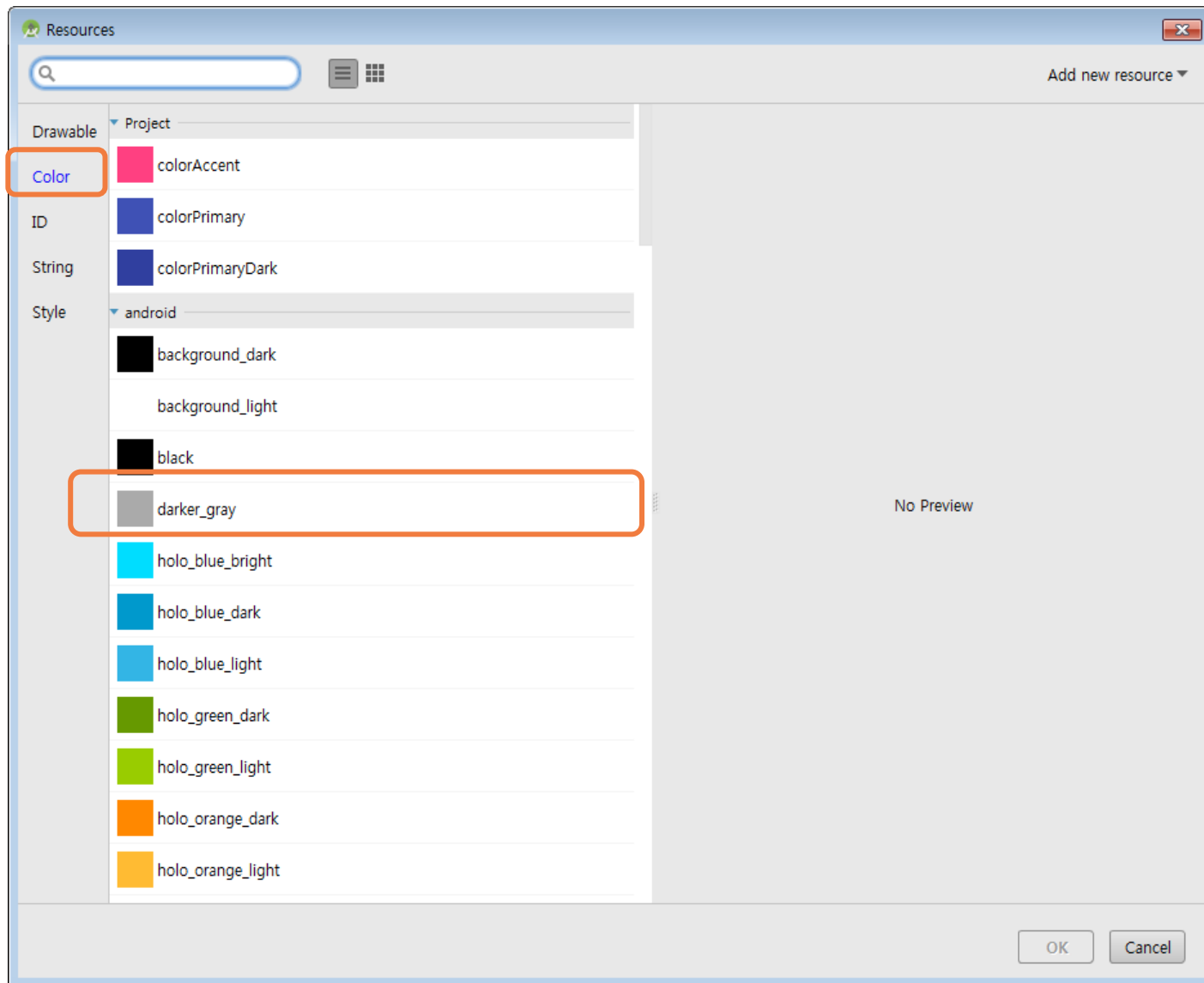


textStyle

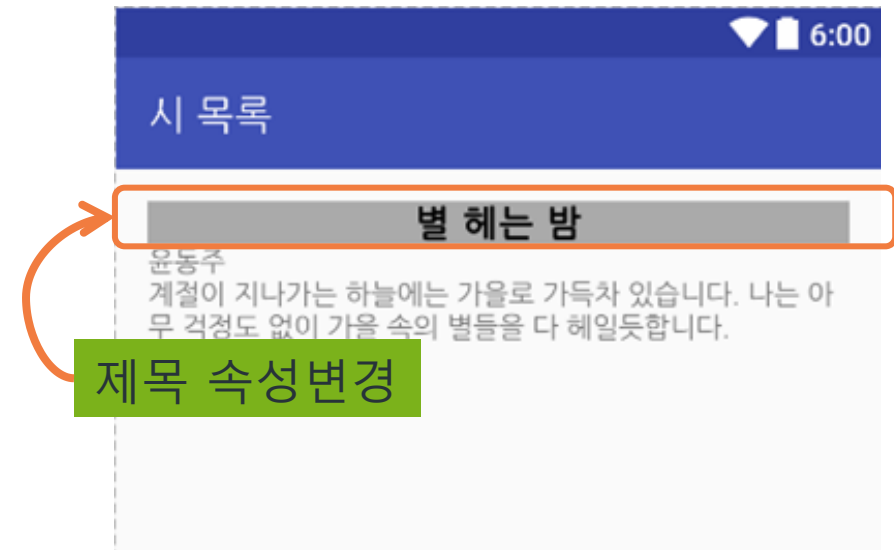


클릭



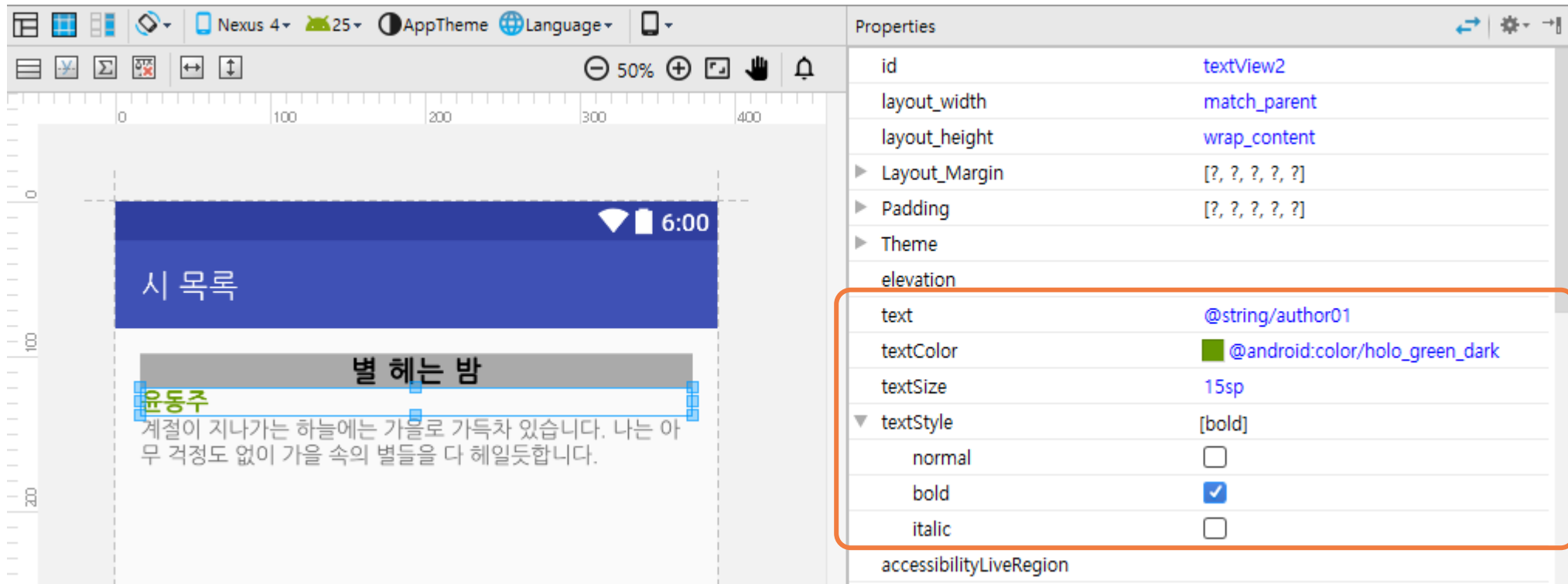


background	@android:color/darker_gray
gravity	[center]
text	@string/title01
textColor	@android:color/black
textSize	18sp
textStyle	[bold]
normal	<input type="checkbox"/>
bold	<input checked="" type="checkbox"/>
italic	<input type="checkbox"/>



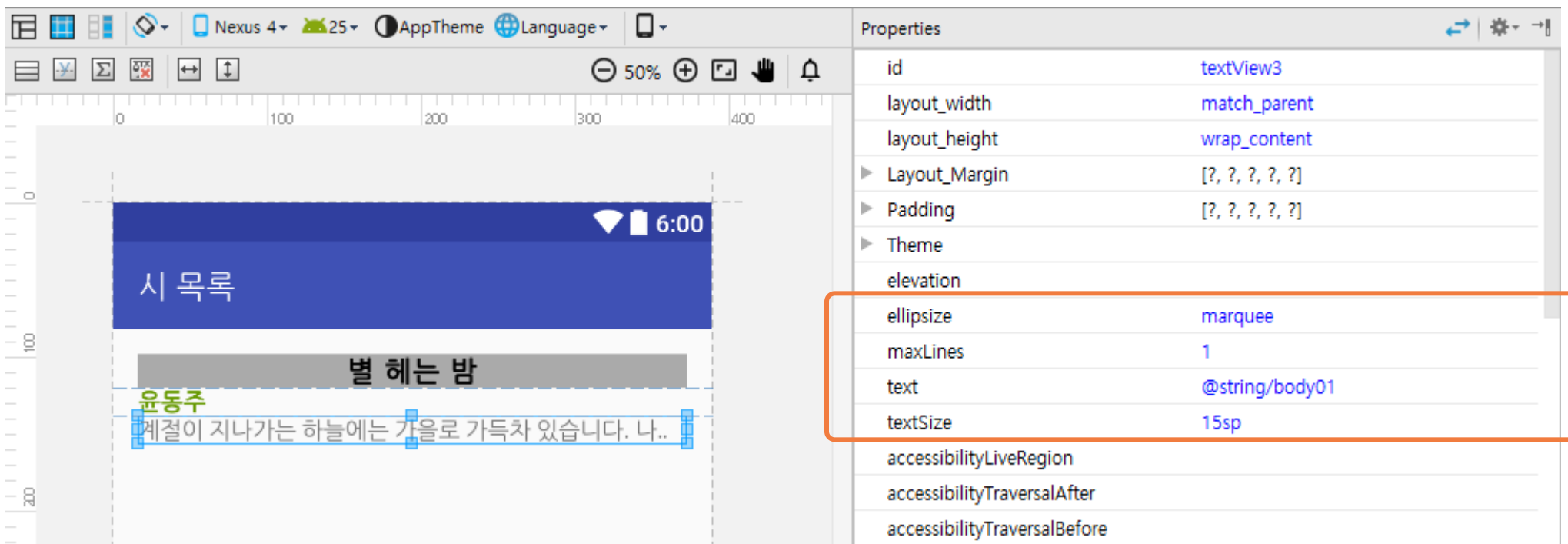
두 번째 TextView의 설정(author 설정)

53



세 번째 TextView의 설정(body 설정)

54



marquee: 일정 시간을 두고 왼쪽으로 글자가 흘러가게 함

```
activity_main.xml x colors.xml x styles.xml x strings.xml x AndroidManifest.xml

1 <?xml version="1.0" encoding="utf-8" ?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:tools="http://schemas.android.com/tools"
4   android:id="@+id/activity_main"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   android:orientation="vertical"
8   android:paddingBottom="16dp"
9   android:paddingLeft="16dp"
10  android:paddingRight="16dp"
11  android:paddingTop="16dp"
12  tools:context="com.example.kyungtae.poems.MainActivity">
13
14  <TextView
15    android:text="@string/title01"
16    android:layout_width="match_parent"
17    android:layout_height="wrap_content"
18    android:id="@+id/textView"
19    android:background="@android:color/darker_gray"
20    android:textColor="@android:color/black"
21    android:gravity="center"
22    android:textStyle="bold"
23    android:textSize="18sp" />
```

```
25 <TextView
26   android:text="@string/author01"
27   android:layout_width="match_parent"
28   android:layout_height="wrap_content"
29   android:id="@+id/textView2"
30   android:textColor="@android:color/holo_green_dark"
31   android:textStyle="bold"
32   android:textSize="15sp" />
33
34 <TextView
35   android:text="@string/body01"
36   android:layout_width="match_parent"
37   android:layout_height="wrap_content"
38   android:id="@+id/textView3"
39   android:textSize="15sp"
40   android:ellipsize="marquee"
41   android:maxLines="1" />
42 </LinearLayout>
43
```

속성 설정[두 번째 시]

56

- 두 번째 시의 “title”, “author”, “body”를 수정하여 첫번째 시와 같도록 설정

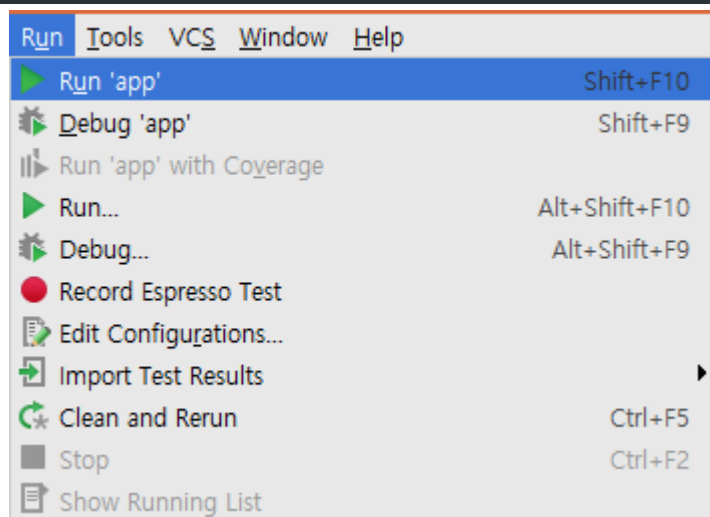


- 이전 xml 소스의 </LinearLayout> 태그 바로 앞에 추가 됨

```
42
43 <TextView
44     android:text="@string/title02"
45     android:layout_width="match_parent"
46     android:layout_height="wrap_content"
47     android:id="@+id/textView4"
48     android:textStyle="bold"
49     android:textSize="18sp"
50     android:background="@android:color/darker_gray"
51     android:gravity="center"
52     android:textColor="@android:color/black" />
53
54 <TextView
55     android:text="@string/author02"
56     android:layout_width="match_parent"
57     android:layout_height="wrap_content"
58     android:id="@+id/textView5"
59     android:textSize="15sp"
60     android:textStyle="bold"
61     android:textColor="@android:color/holo_green_dark" />
62
63 <TextView
64     android:text="@string/body02"
65     android:layout_width="match_parent"
66     android:layout_height="wrap_content"
67     android:id="@+id/textView6"
68     android:maxLines="1"
69     android:ellipsize="marquee" />
```

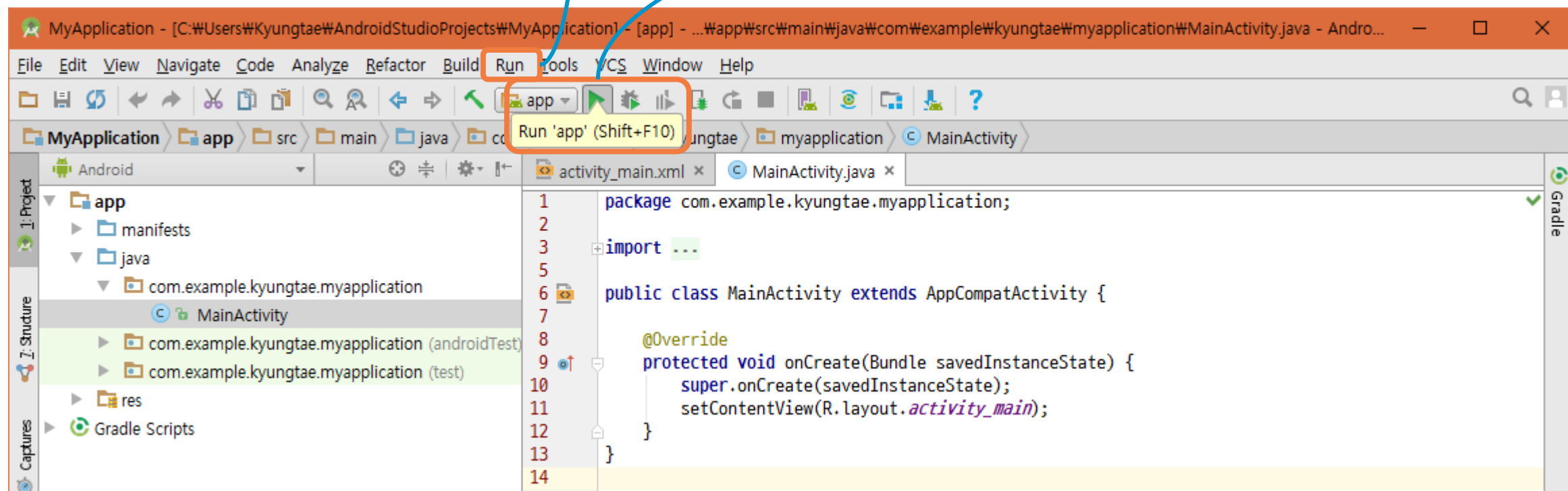

Step 3. 프로젝트 실행

58



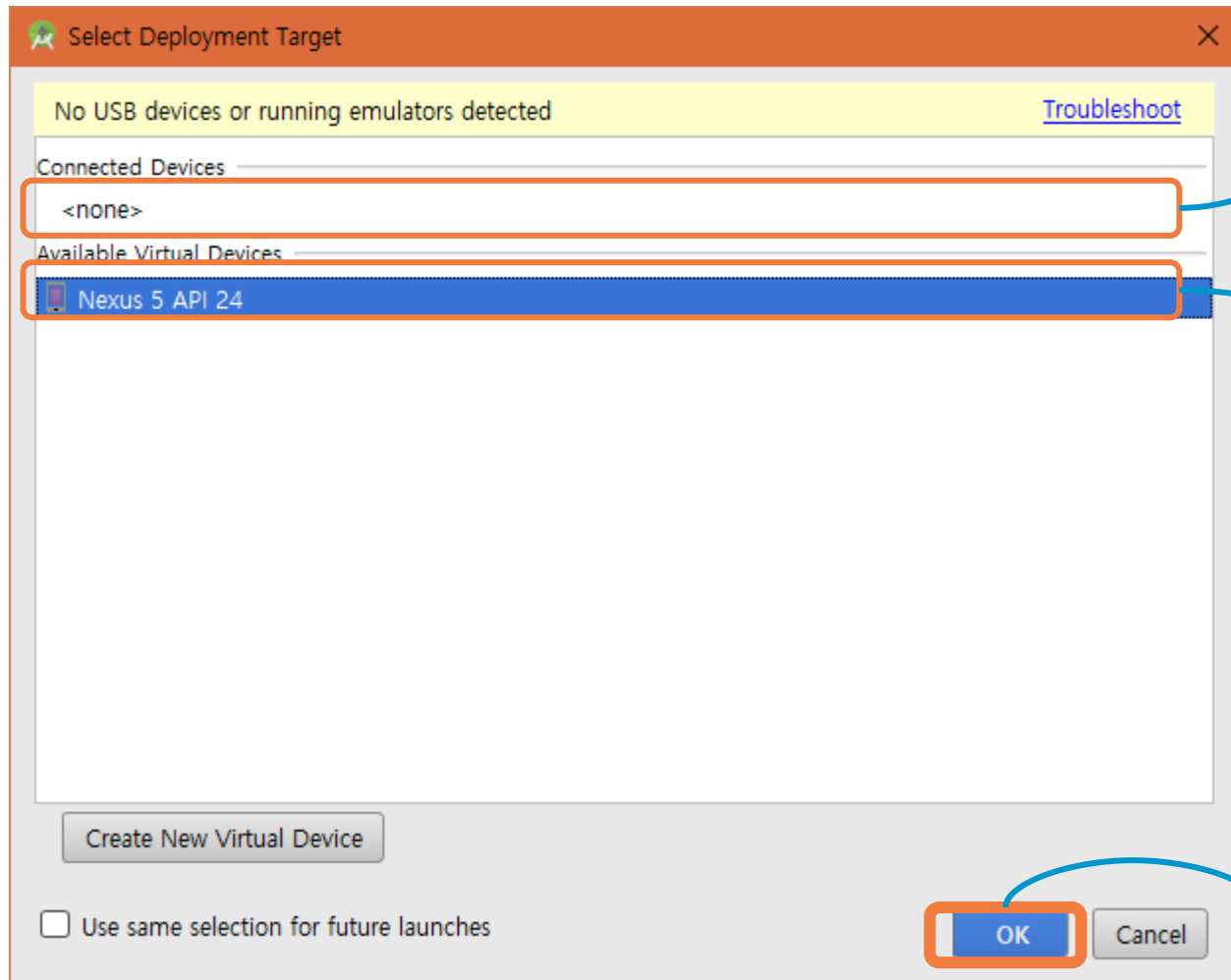
Run → Run 'app' 메뉴 클릭

앱 실행 아이콘 클릭



• AVD 장비 선택하기

59

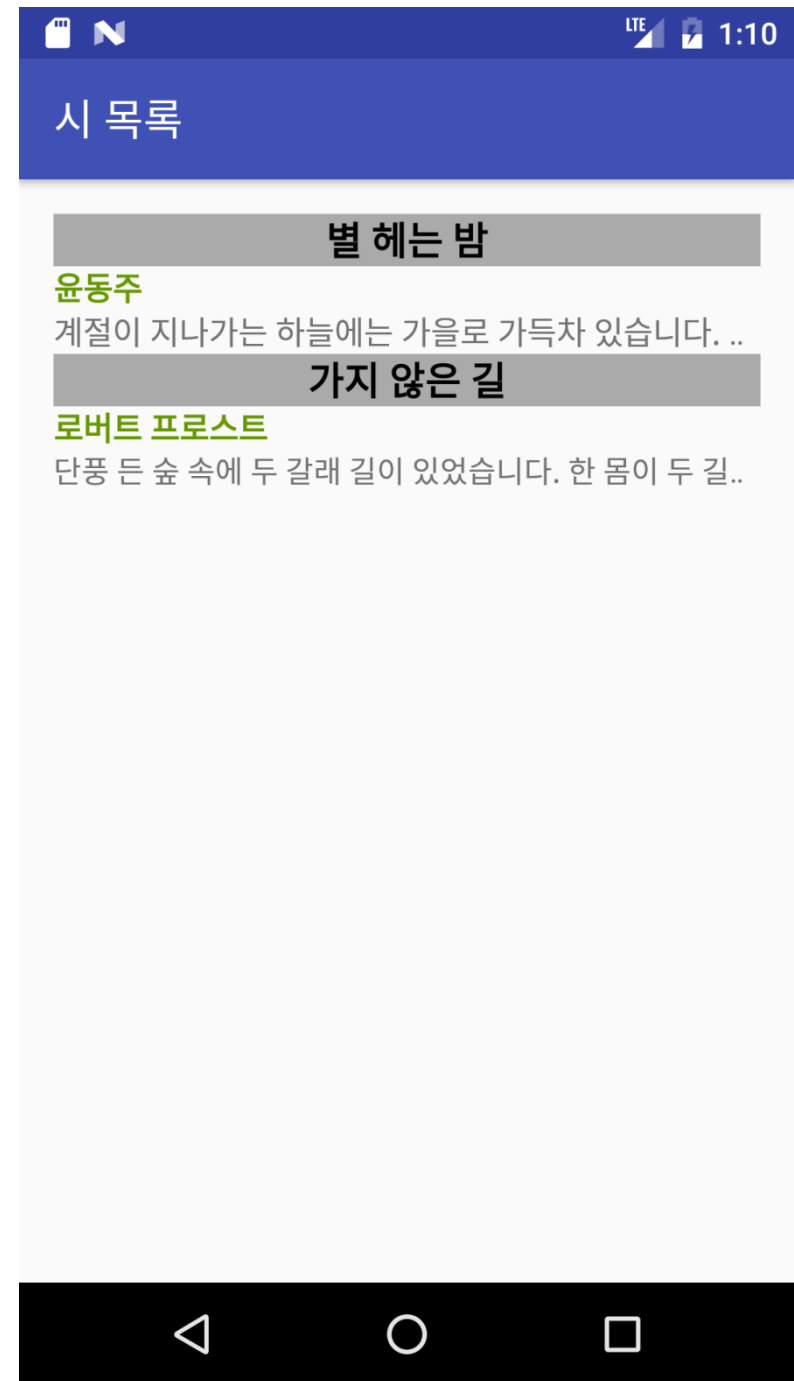


데이터 케이블로 연결된
스마트폰

AVD

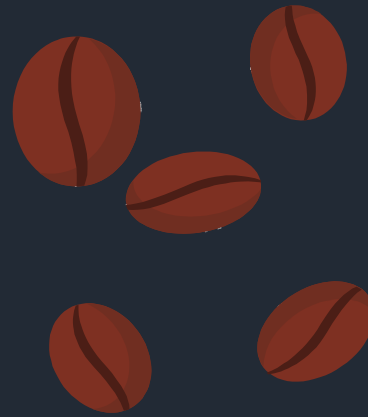
스마트폰 또는 AVD를 선택하고
'OK' 버튼을 클릭

- 실행 결과



심터

다양한 화면 크기를 지원하기 위한 화면 출력 단위
(화면 출력 용어와 개념)

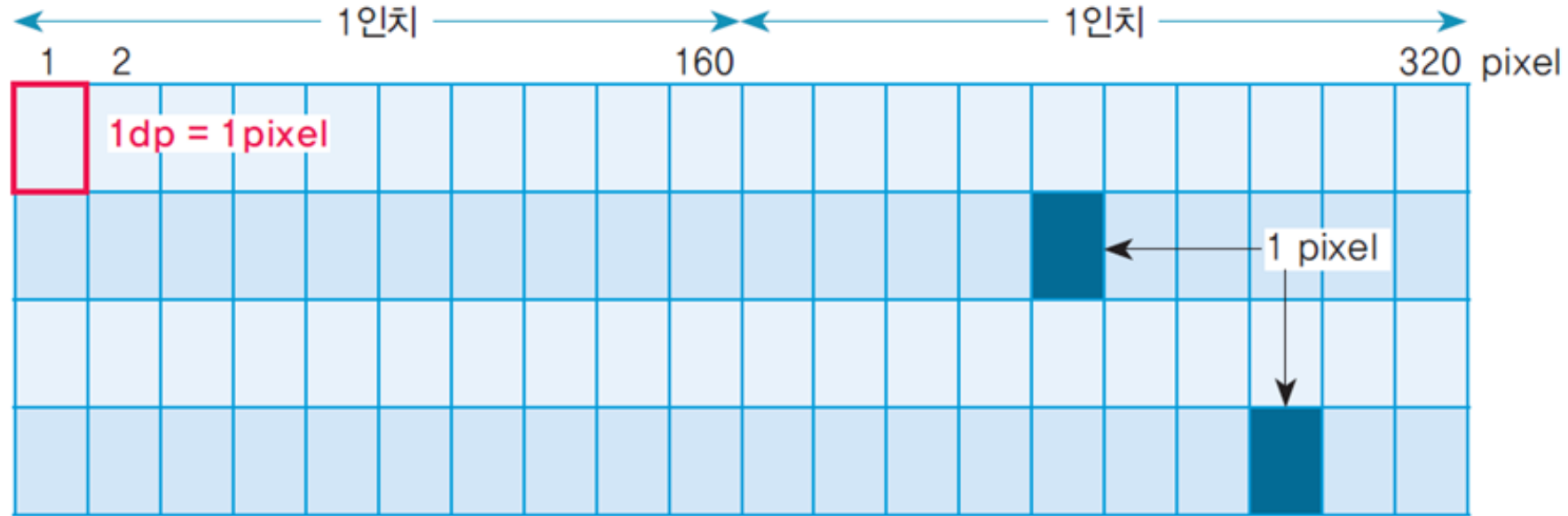


용어	개념														
화면 크기 (screen size)	<div><ul style="list-style-type: none">화면 대각선의 실제 길이안드로이드의 4가지 분류: small, normal, large, extra-large<table><tr><th>화면크기 분류</th><th>최소 화면밀도</th></tr><tr><td>small</td><td>426dp x 320dp</td></tr><tr><td>normal</td><td>470dp x 320dp</td></tr><tr><td>large</td><td>640dp x 480dp</td></tr><tr><td>extra-large</td><td>960dp x 720dp</td></tr></table></div>	화면크기 분류	최소 화면밀도	small	426dp x 320dp	normal	470dp x 320dp	large	640dp x 480dp	extra-large	960dp x 720dp				
화면크기 분류	최소 화면밀도														
small	426dp x 320dp														
normal	470dp x 320dp														
large	640dp x 480dp														
extra-large	960dp x 720dp														
화면 밀도 (screen density)	<div><ul style="list-style-type: none">화면 면적 당 픽셀 수, 대개 1인치 당 픽셀 수를 나타내는 dpi(dot per inch) 또는 pd(pixel density)를 사용함안드로이드의 6가지 화면밀도 분류와 dpi 수<table><tr><th>화면크기 분류</th><th>최소 화면밀도</th></tr><tr><td>ldpi(low)</td><td>120</td></tr><tr><td>mdpi(medium)</td><td>160</td></tr><tr><td>hdpi(high)</td><td>240</td></tr><tr><td>xhdpi(extra-high)</td><td>320</td></tr><tr><td>xxhdpi(extra-extra-high)</td><td>480</td></tr><tr><td>xxxhdpi(extra-extra-extra-high)</td><td>640</td></tr></table></div>	화면크기 분류	최소 화면밀도	ldpi(low)	120	mdpi(medium)	160	hdpi(high)	240	xhdpi(extra-high)	320	xxhdpi(extra-extra-high)	480	xxxhdpi(extra-extra-extra-high)	640
화면크기 분류	최소 화면밀도														
ldpi(low)	120														
mdpi(medium)	160														
hdpi(high)	240														
xhdpi(extra-high)	320														
xxhdpi(extra-extra-high)	480														
xxxhdpi(extra-extra-extra-high)	640														

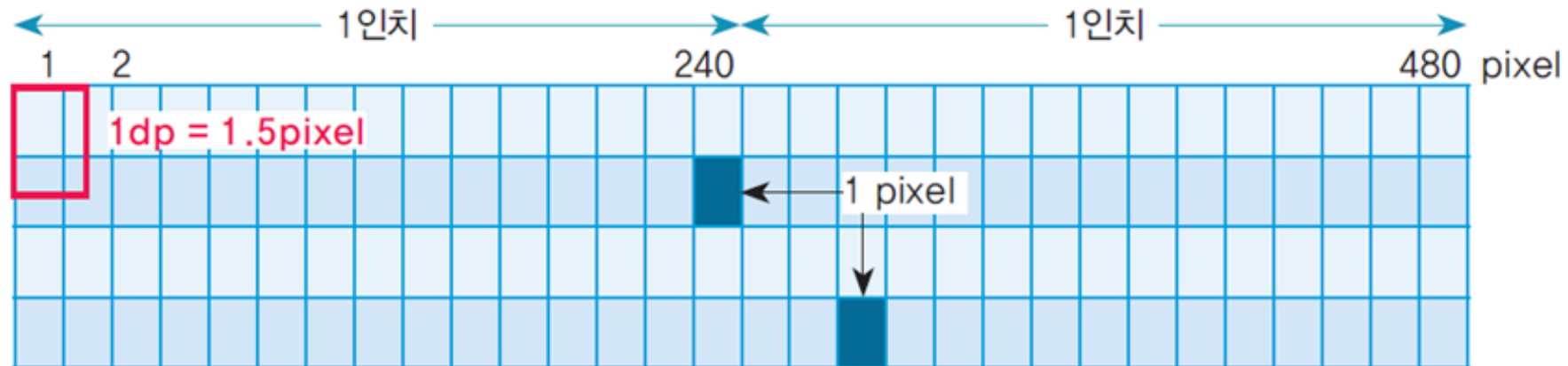
용어	개념
방향 (orientation)	<ul style="list-style-type: none"> ▪ 사용자 관점의 화면 방향 ▪ landscape(가로 방향이 길게 보임), portrait(세로 방향이 길게 보임)
해상도 (resolution)	<ul style="list-style-type: none"> ▪ 화면 상의 물리적인 픽셀 수
dp (density-independent pixel)	<ul style="list-style-type: none"> ▪ 밀도와 무관한 가상 픽셀 ▪ 이론상 어떠한 해상도에서도 같은 크기를 보여 주는 것이 핵심 개념 ▪ 1인치 당 160 픽셀 수를 가진 medium 화면 밀도 유형을 기준으로 볼 때 1dp는 1pixel에 대응됨 ▪ 픽셀과 dp의 관계 <ul style="list-style-type: none"> ▪ $px = dp \times dpi / 160$ ▪ 예) dpi가 160인 경우 1dp는 1pixel과 같음. 그러나, dpi가 240인 경우는 1dp는 1.5pixel이 됨. <p>즉, 모바일 기기의 해상도는 다르더라도 dp를 사용하면 같은 비율로 화면에 출력됨</p>

- dp와 pixel의 관계($px = dp \times dpi / 160$)

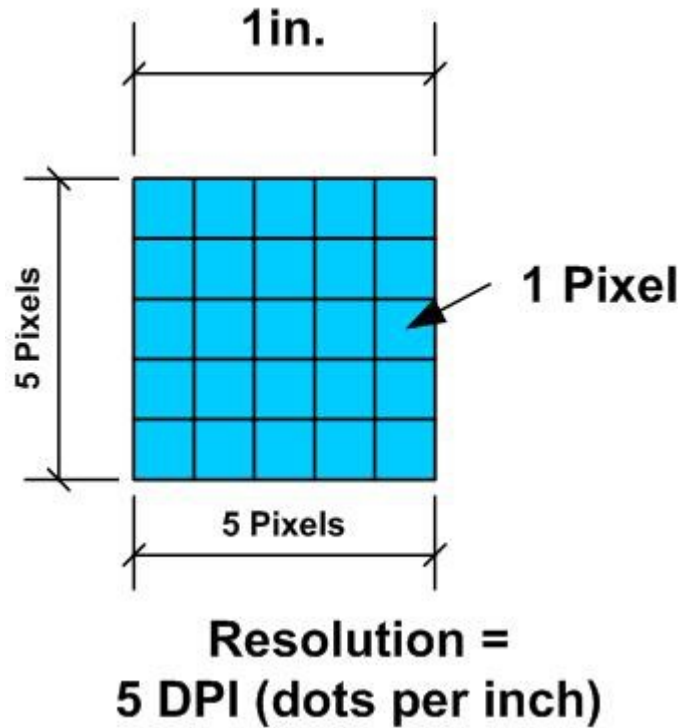
① 160 dpi의 경우, $1\text{pixel} = 1\text{dp} \times 160\text{dpi} / 160$



② 240 dpi의 경우, $1.5\text{pixel} = 1\text{dp} \times 240\text{dpi} / 160$

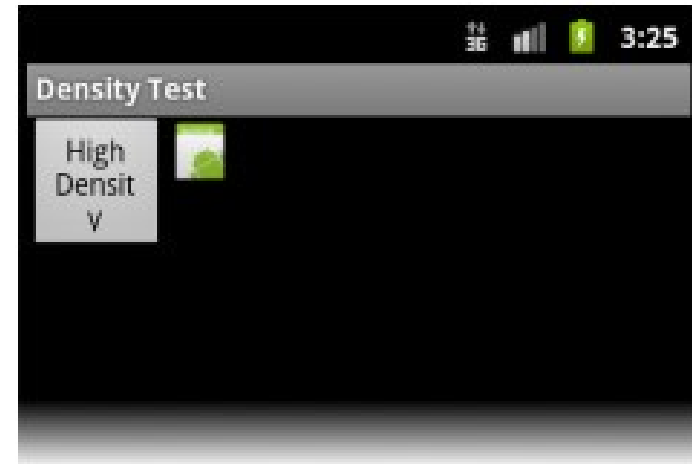
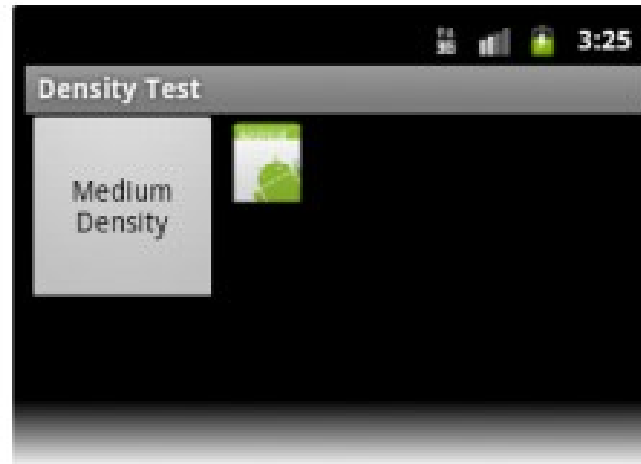
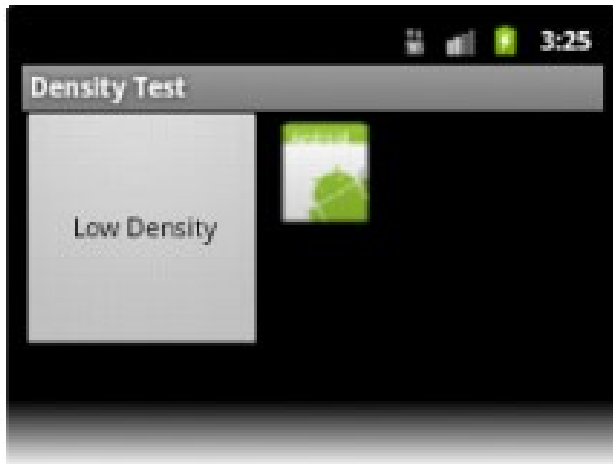


- DPI(Dots Per Inch)라는 단위

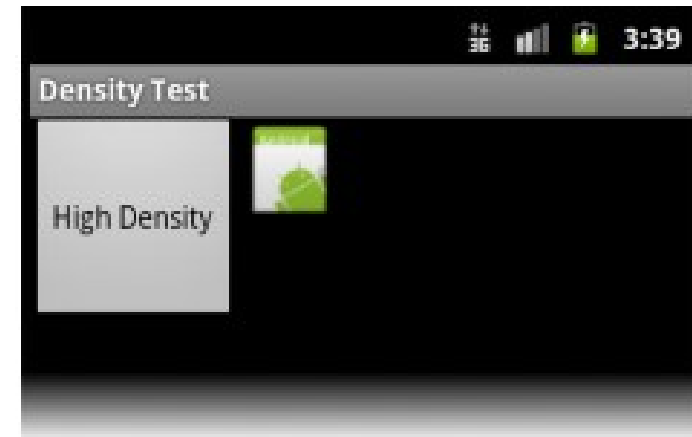
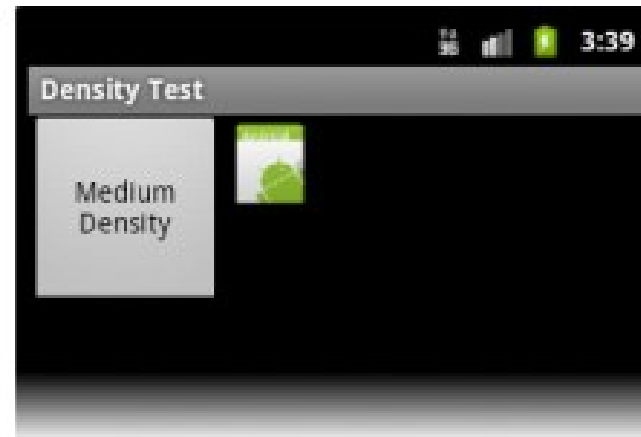
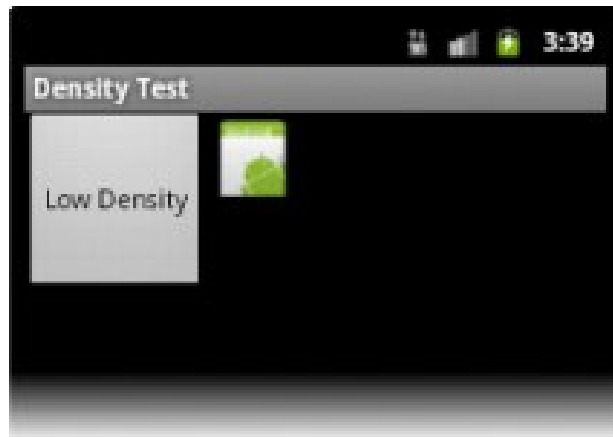


- 출처: <http://solarisailab.com/archives/179>

Low, Medium, High-density 화면 밀도에 **dpi 단위**로 크기를 지정했을 때



Low, Medium, High-density 화면 밀도에 **DP 단위**를 지원했을 때



- 출처: <http://solarisailab.com/archives/179>

