

Week10. 모션센서



개발환경 구축 절차

2

주 차	수업 내용
1	수업 소개
2	개발 환경 구축과 맛보기 프로젝트
3	텍스트 출력과 레이아웃
4	이미지의 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

<http://github.com/hopypark>

센서의 이해

4

- 안드로이드에서 지원하는 모션, 환경, 위치 센서

구분	기능	센서 종류
모션 센서	스마트폰의 세 축에 따른 가속도와 회전력 측정	중력 센서, 자이로스코프, 회전 벡터 센서
환경 센서	온도, 기압, 조도, 습도와 같은 환경 변수 측정	기압계, 광도계, 온도계
위치 센서	디바이스의 물리적인 위치 측정	방향 센서, 자력계

스마트폰에 내장된 센서들...



(앞면)



(앞면)

삼성(갤럭시 S7)



(앞면)

LG(G5)



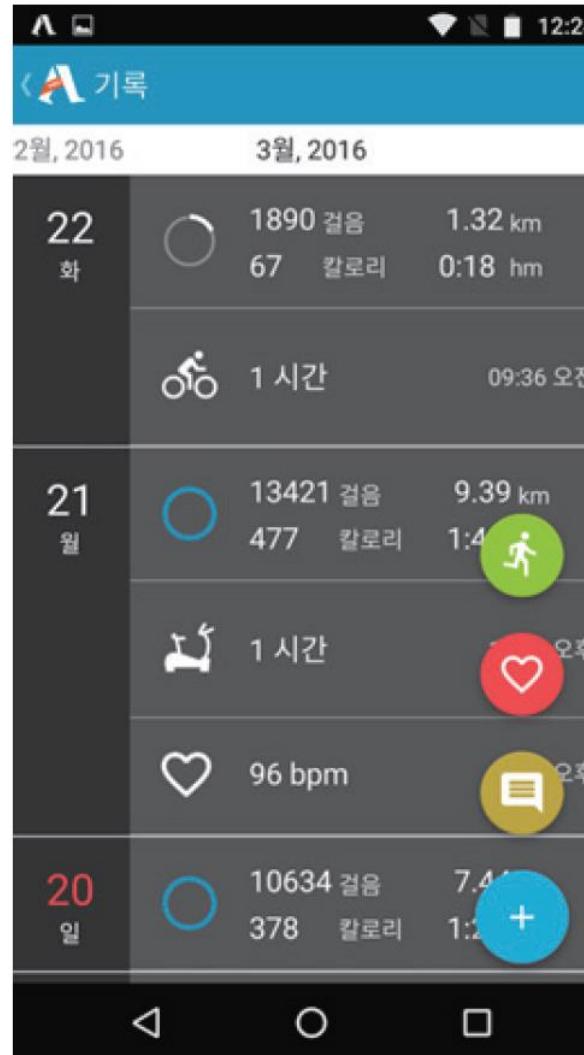
(뒷면)

모션센서를 이용한 앱의 예

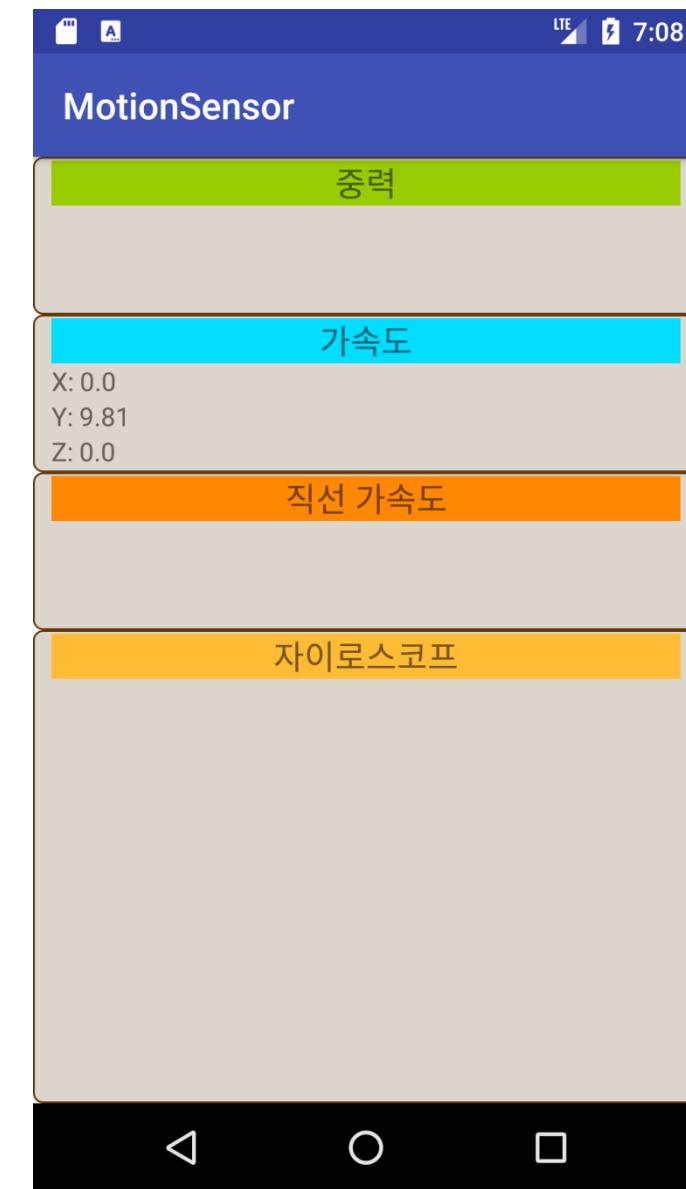


현재 상황

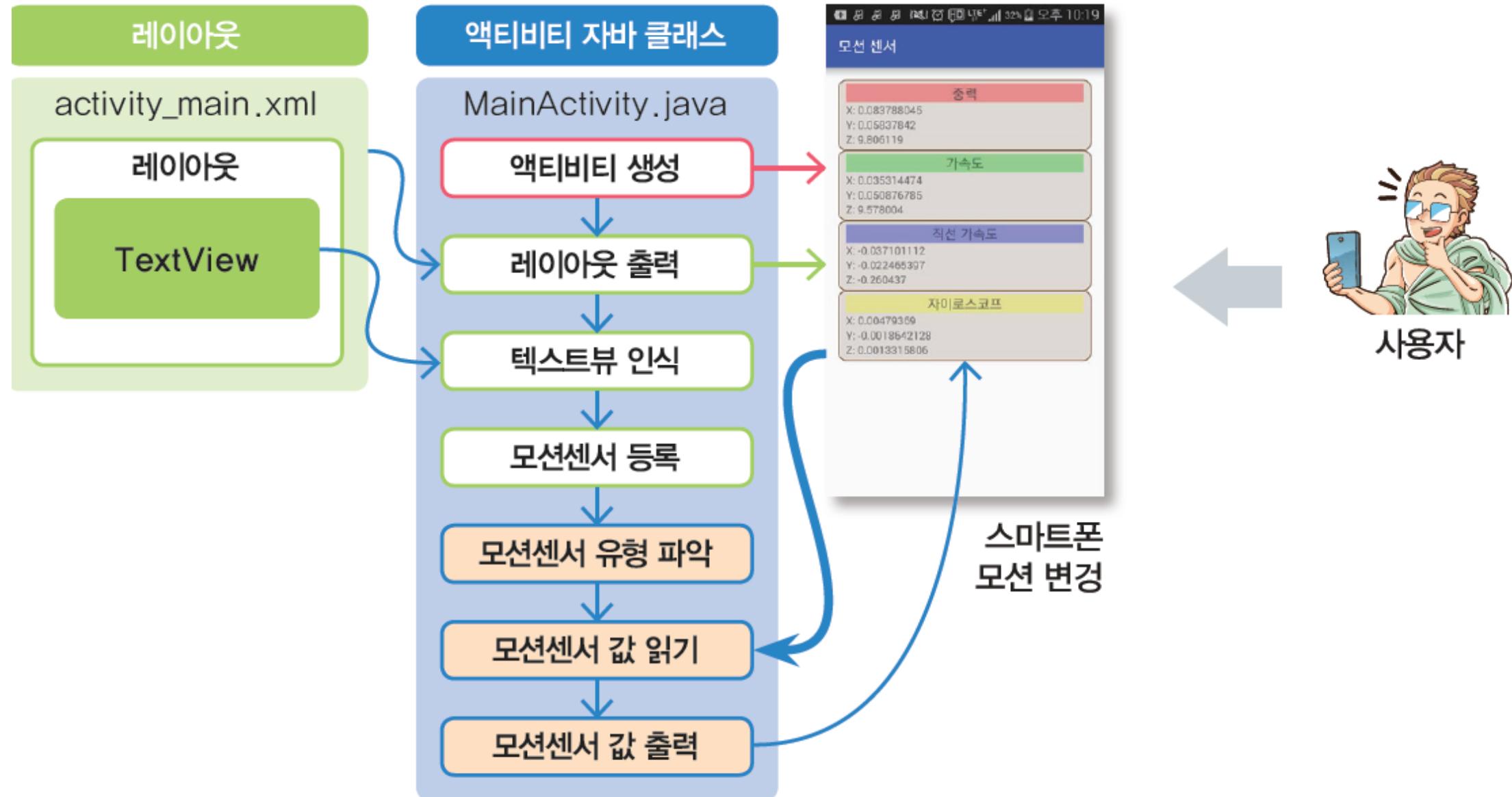
● 애쿠페도 만보기



일별 기록현황

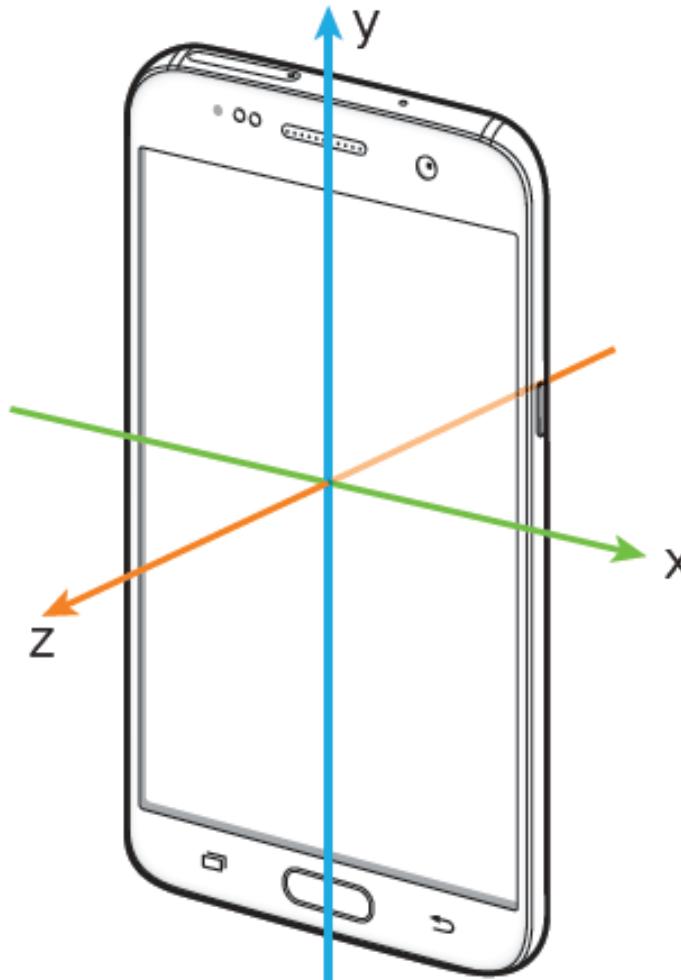


모션 센서 원리

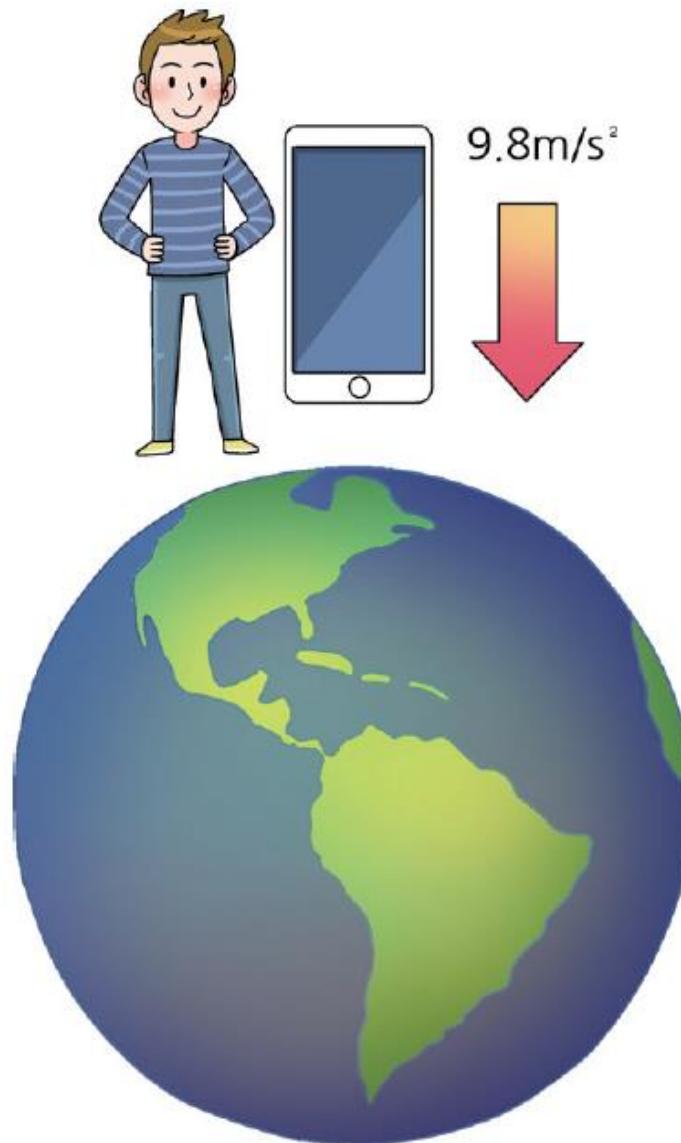


모션 센서는 센서 값을 표현하기 위해 세가지 축 사용

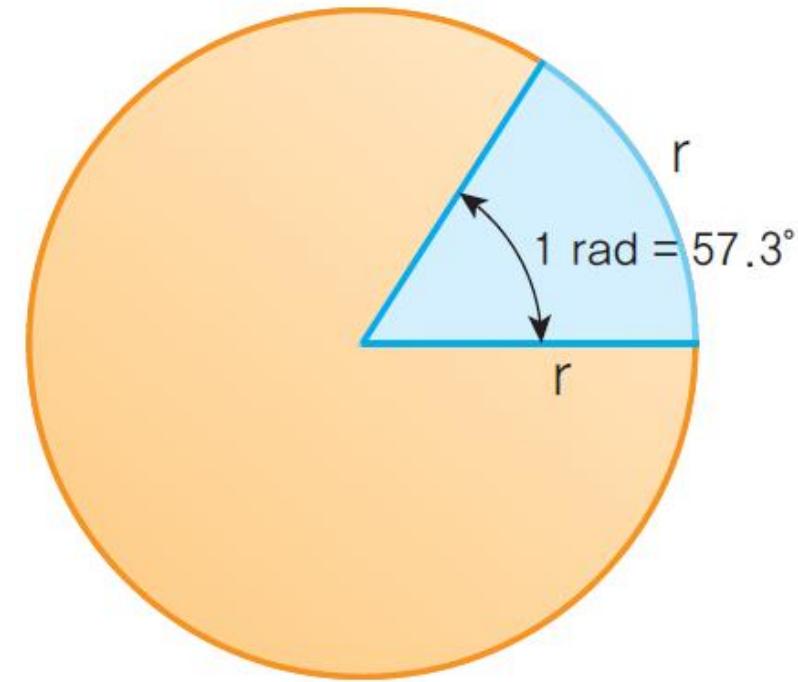
8



- 중력
 - 디바이스에 가해지는 중력의 크기
 - 지표면에서 9.8 m/s^2
 - 고도가 높을 수록 작아짐.
- 가속도
 - 중력을 포함해서 디바이스에 가해지는 가속도를 더하거나 뺀 값
 - 중력과 반대 방향으로 가속도가 작용하면 중력에 가속도를 더한 값이며, 같은 방향이면 가속도를 뺀 값으로 측정
- 직선 가속도
 - 중력을 제외하고 디바이스에 가해지는 가속도의 크기를 측정한 값
- 자이로스코프
 - x, y, z축 방향에 대한 회전율을 측정(rad/s)
 - 1라디안은 원의 반지름과 같은 원호에 대한 중심각(57.3도)



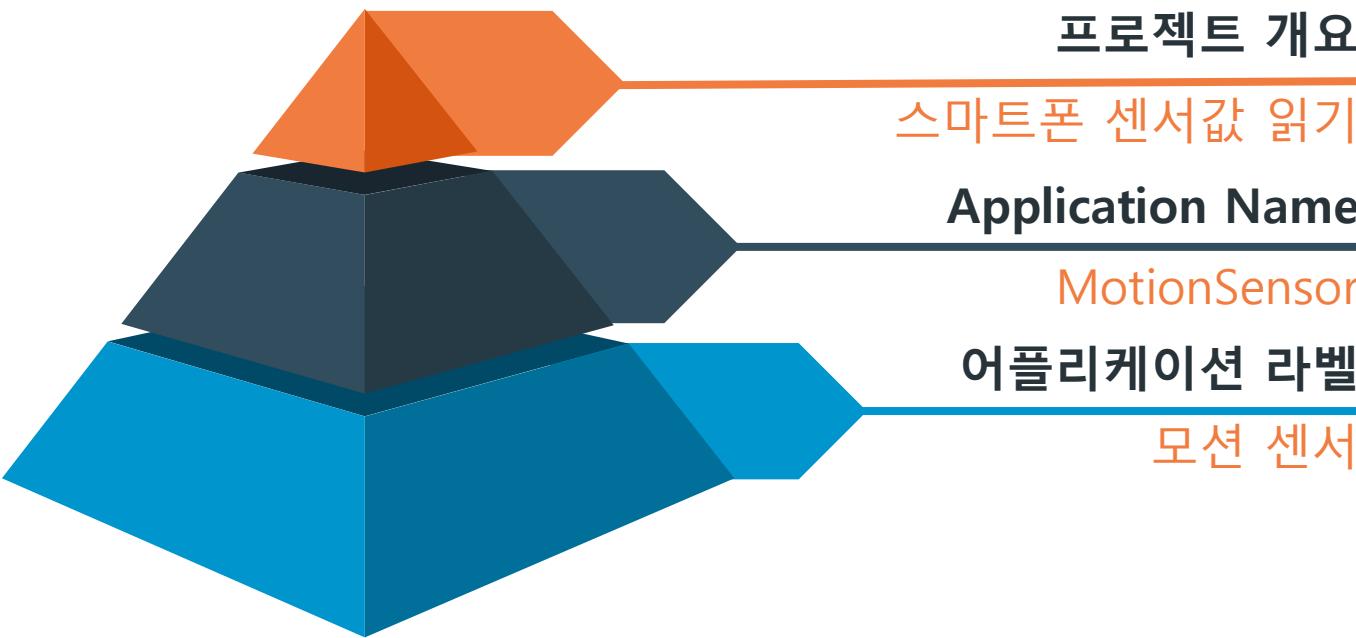
지표면의 중력



라디안(radian)

Step 0. 프로젝트 개요

10



Step 1. 프로젝트 생성

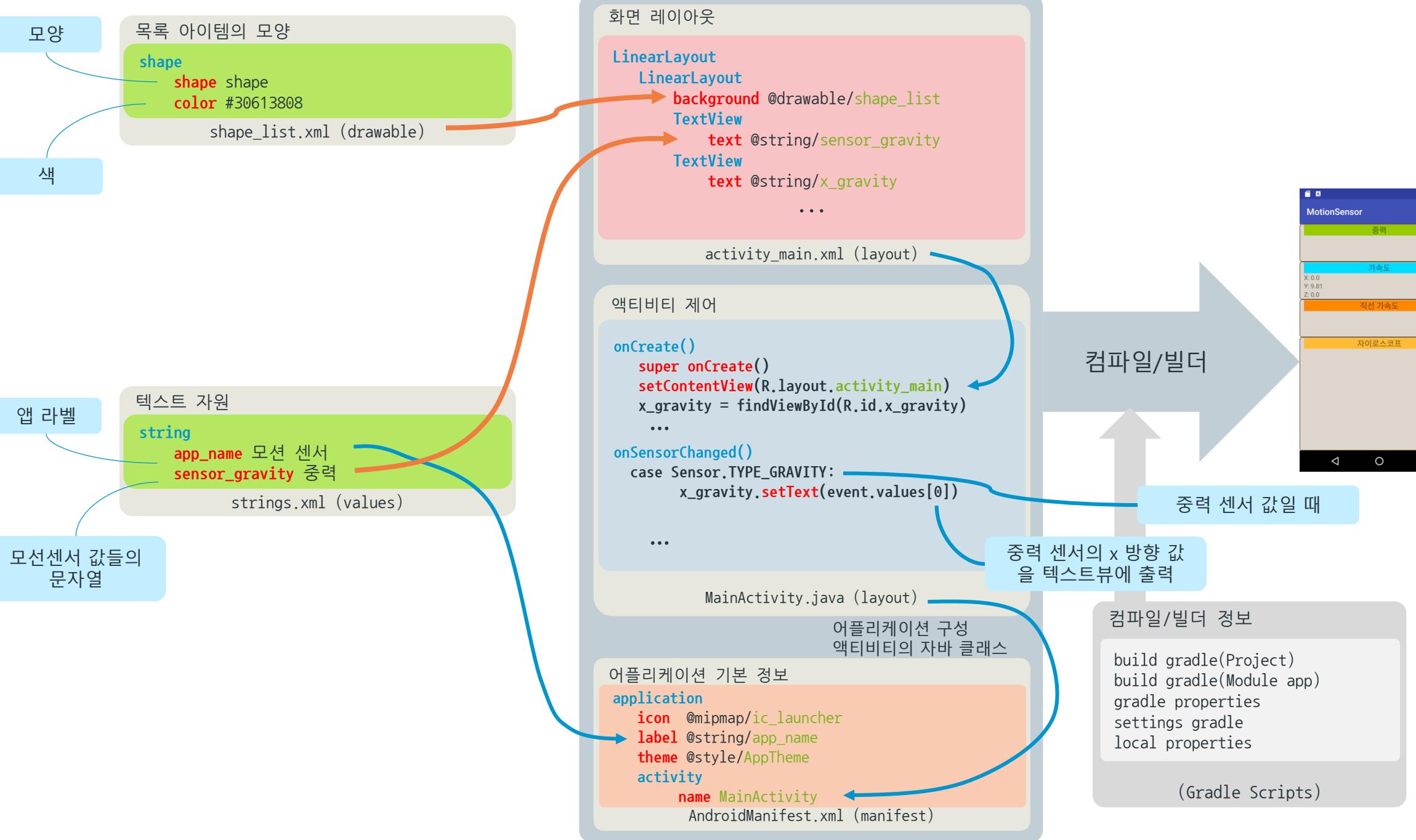
11

절차	내용
① 프로젝트 시작	메뉴에서 ‘File → New Project’ 클릭
② 프로젝트 구성	Application Name: MotionSensor Company Domain: kyungtae.example.com (디폴트 사용) Project Location: ~/AndroidStudioProject/ktpark/MotionSensor
③ 제품형태	Phone and Tablet(사용할 안드로이드 버전 지정: Android 7.0 Nougat)
④ 액티비티 유형	Empty Activity
⑤ 파일 옵션	Activity Name: MainActivity (디폴트 사용) Layout Name: activity_main (디폴트 사용)

Step 2. 파일 편집

12

모듈	폴더	소스 파일	편집 내용
manifests		AndroidManifest.xml	
java	com.example.kyungtae.video1	MainActivity.java	<ul style="list-style-type: none">• 센서 등록• 센서 값 변경 확인• 센서 값 출력
res	drawable	shape_list	<ul style="list-style-type: none">• 출력모양 설계
	layout	activity_main.xml	<ul style="list-style-type: none">• 모션센서 측정값 배치(TextView)
	mipmap	ic_launcher.png	
	values	colors.xml	
		dimens.xml	
		strings.xml	<ul style="list-style-type: none">• 어플리케이션 라벨 수정• 모션센서 이름의 문자열 추가
		styles.xml	



Step 2.1 텍스트 자원의 편집

14

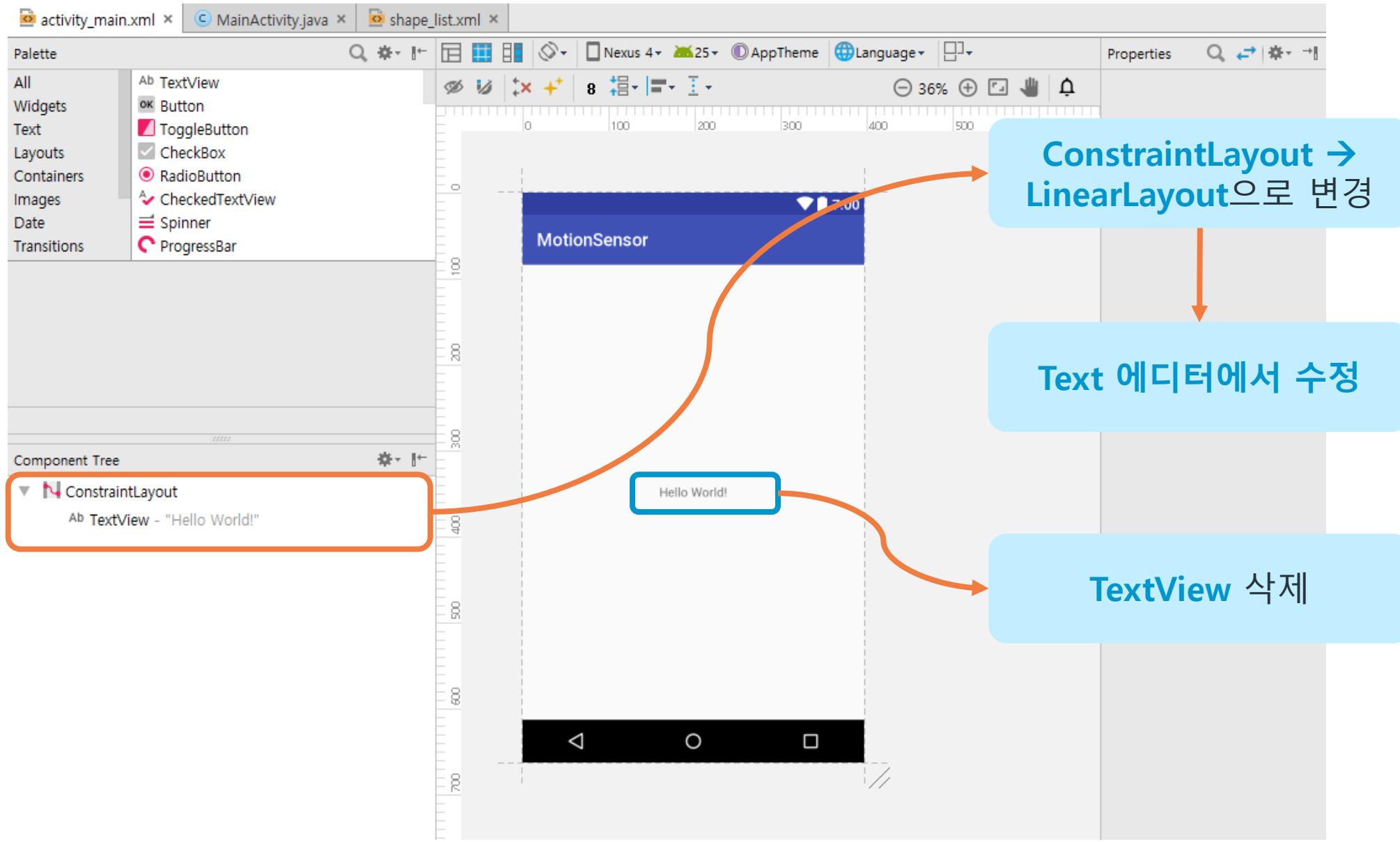
- strings.xml

The screenshot shows the Android Studio interface with the tabs: activity_main.xml, strings.xml, dimens.xml, MainActivity.java, and shape_list.xml. The strings.xml tab is active. The code in strings.xml is as follows:

```
1 <resources>
2   <string name="app_name">MotionSensor</string>
3
4   <string name="sensor_gravity">중력</string>
5   <string name="sensor_accelerometer">가속도</string>
6   <string name="sensor_linear_acceleration">직선 가속도</string>
7   <string name="sensor_gyroscope">자이로스코프</string>
8 </resources>
9
10 |
```

A blue rounded rectangle highlights the four string definitions from line 4 to line 7. A light blue callout bubble with a curved arrow points to the Korean text "중력", "가속도", "직선 가속도", and "자이로스코프". The text inside the bubble is "센서 값들의 제목".

2.2 화면 설계



• Layout 변경 및 기본 TextView 삭제

The image shows two screenshots of the Android Studio code editor for the file `activity_main.xml`.
The top screenshot displays the XML code using a `ConstraintLayout` as the root container. A red box highlights the opening tag `<android.support.constraint.ConstraintLayout`. A red arrow points from this tag to the corresponding closing tag `</android.support.constraint.ConstraintLayout>` in the bottom screenshot.
The bottom screenshot shows the same XML code, but the root container has been changed to a `LinearLayout`. A red box highlights the opening tag `<LinearLayout`. A red box also highlights the `<TextView` element, which contains the text "Hello World!". A large orange rounded rectangle surrounds the entire `<LinearLayout>` block. An orange arrow points from the text "삭제" (Delete) at the end of this box to the bottom right corner of the `<LinearLayout>` block.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.kyungtae.audio1.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

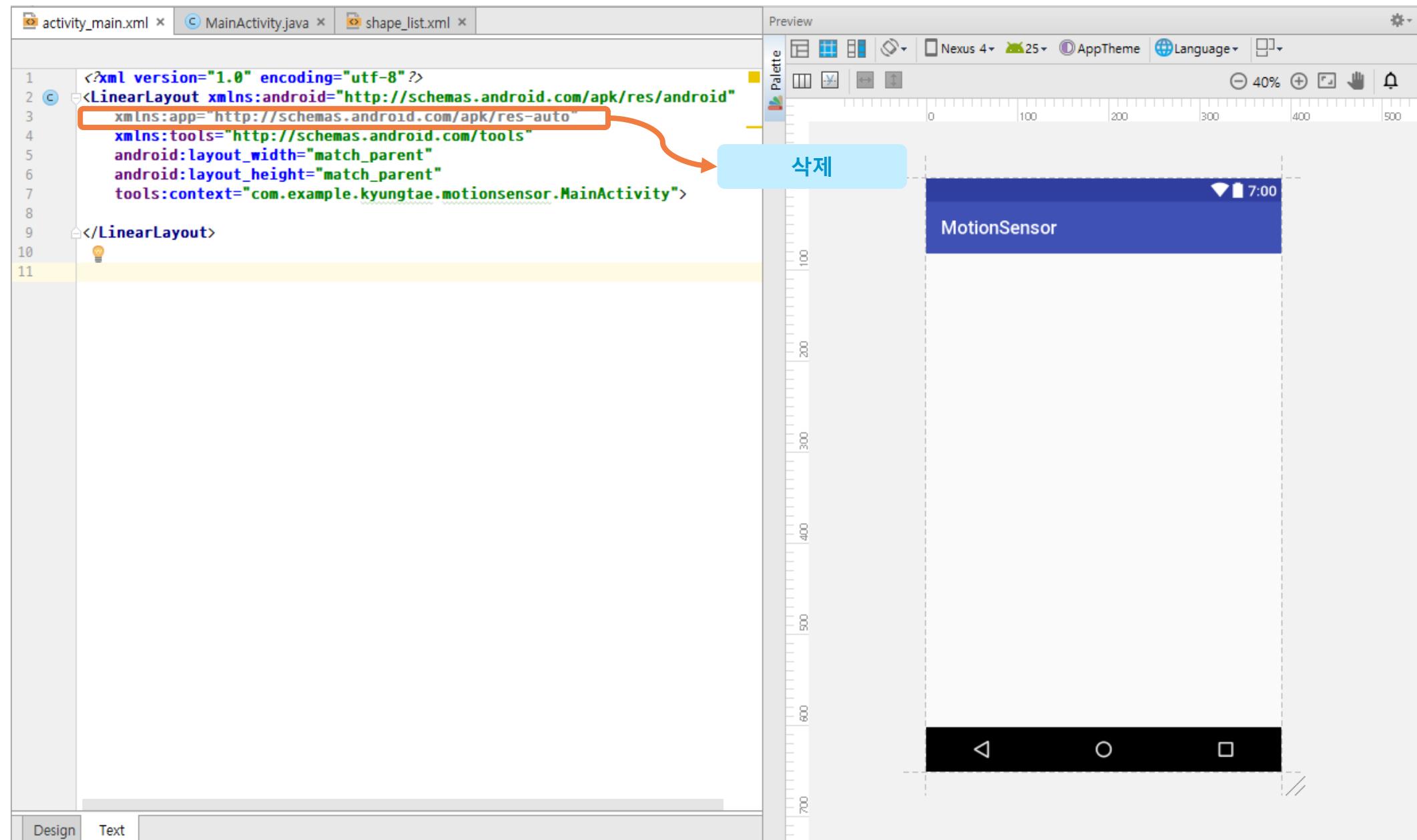


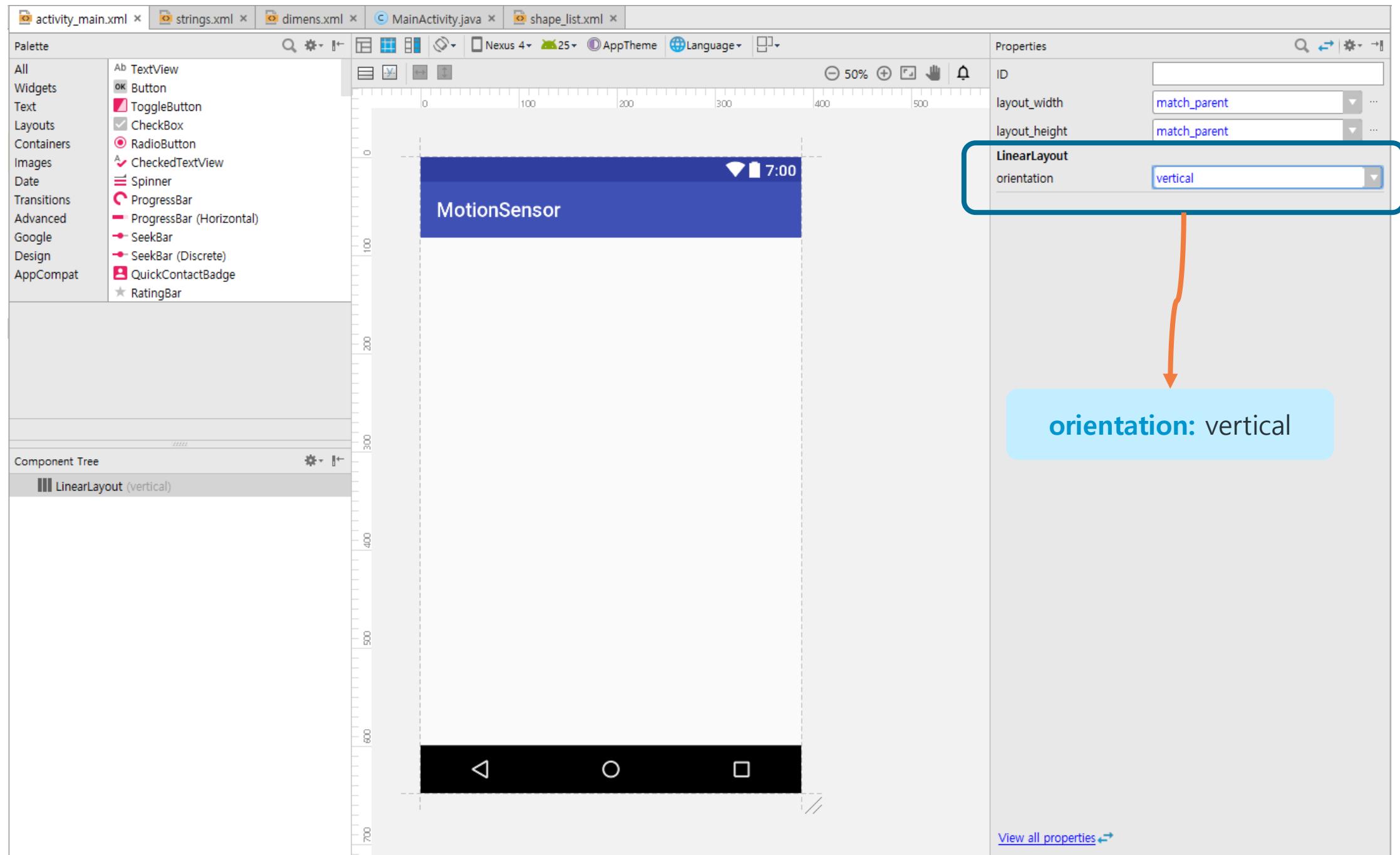
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.kyungtae.audio1.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</LinearLayout>
```

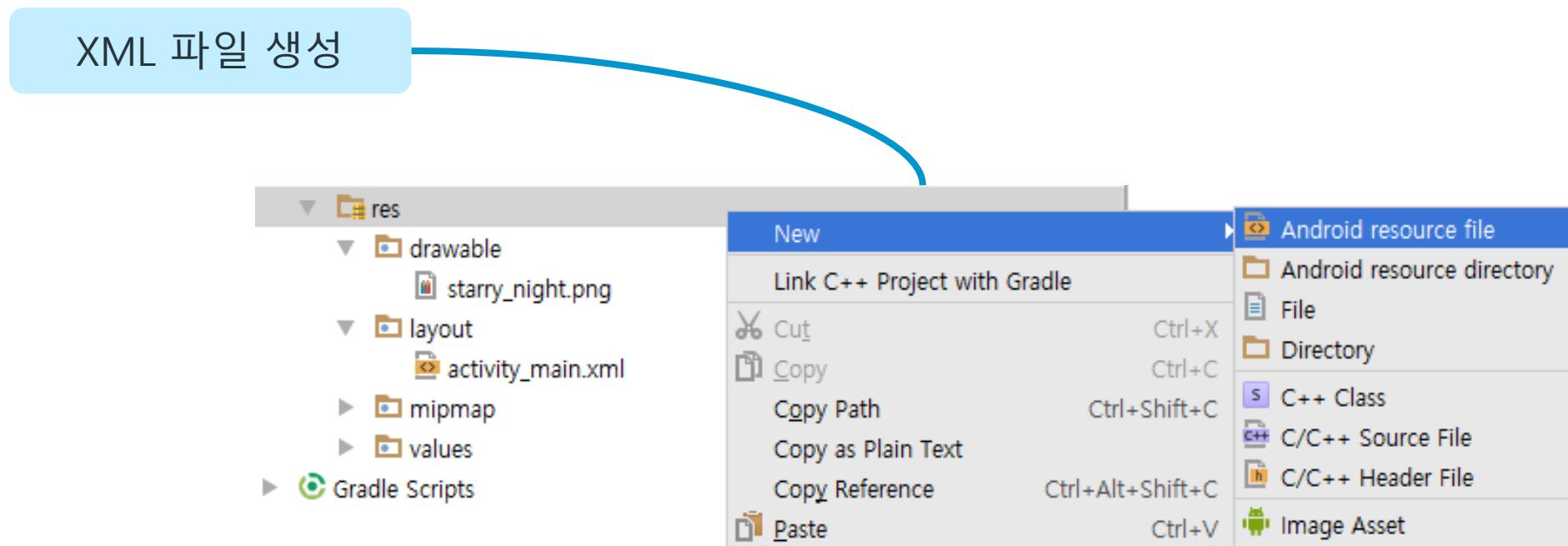
- activity_main.xml





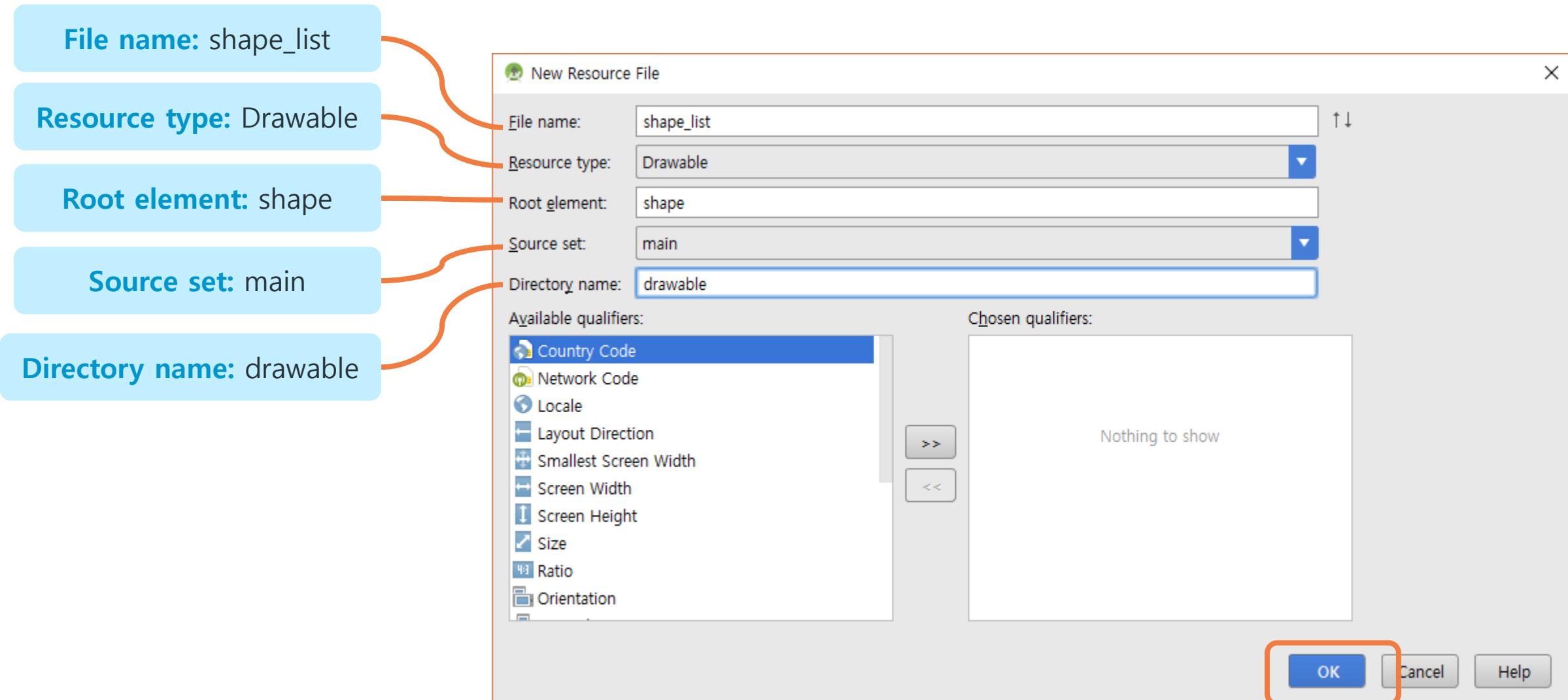
2.3 drawable 리소스 - shape_list.xml 추가

- **shape_list.xml** 생성(res/drawable 폴더)
 - drawable resource를 이용한 그림 출력



- Set New Resource File - shape_list.xml

20



• shape_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#3061380B"/>
    <stroke android:width="1dp" android:color="#61380B"/>
    <padding
        android:top="2dp"
        android:bottom="2dp"
        android:left="10dp"
        android:right="10dp">
    </padding>
    <corners android:radius="5dp"></corners>
</shape>
```

• 출력모양을 내부의 색

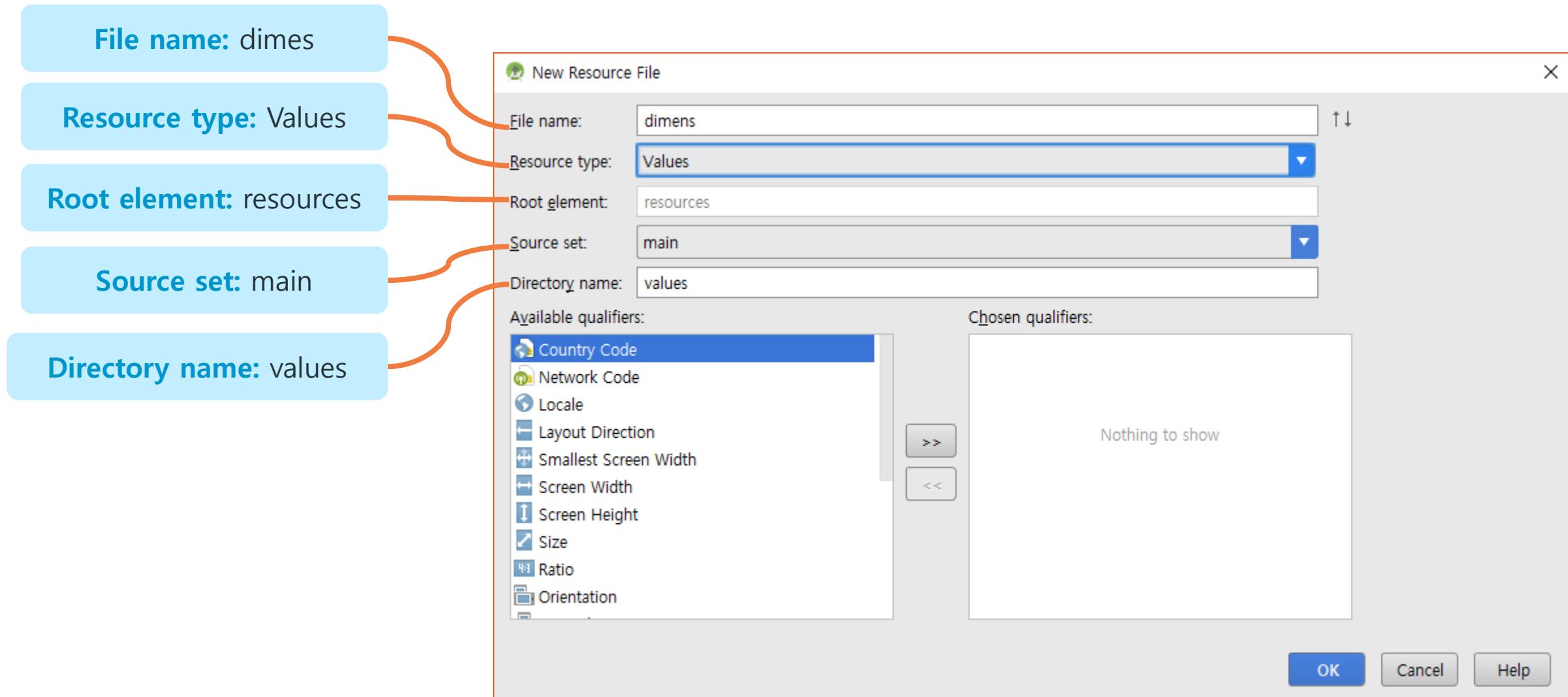
• 출력모양을 테두리의 색

• 내부 패딩 정보

• 출력모양 모서리를 둥근 모양
으로 지정(반지름은 5dp)

The screenshot shows the Android Studio interface with three tabs: activity_main.xml, MainActivity.java, and shape_list.xml. The shape_list.xml tab is active, displaying the XML code for a rectangle shape. The code includes elements for color, stroke, padding, and corner radius. Callout boxes with Korean text point to specific parts of the code: one to the stroke element, one to the padding section, and one to the corners element. To the right of the code is a preview window showing a brown rectangle with a thin black border, white padding, and rounded corners.

- Set New Resource File - dimens.xml

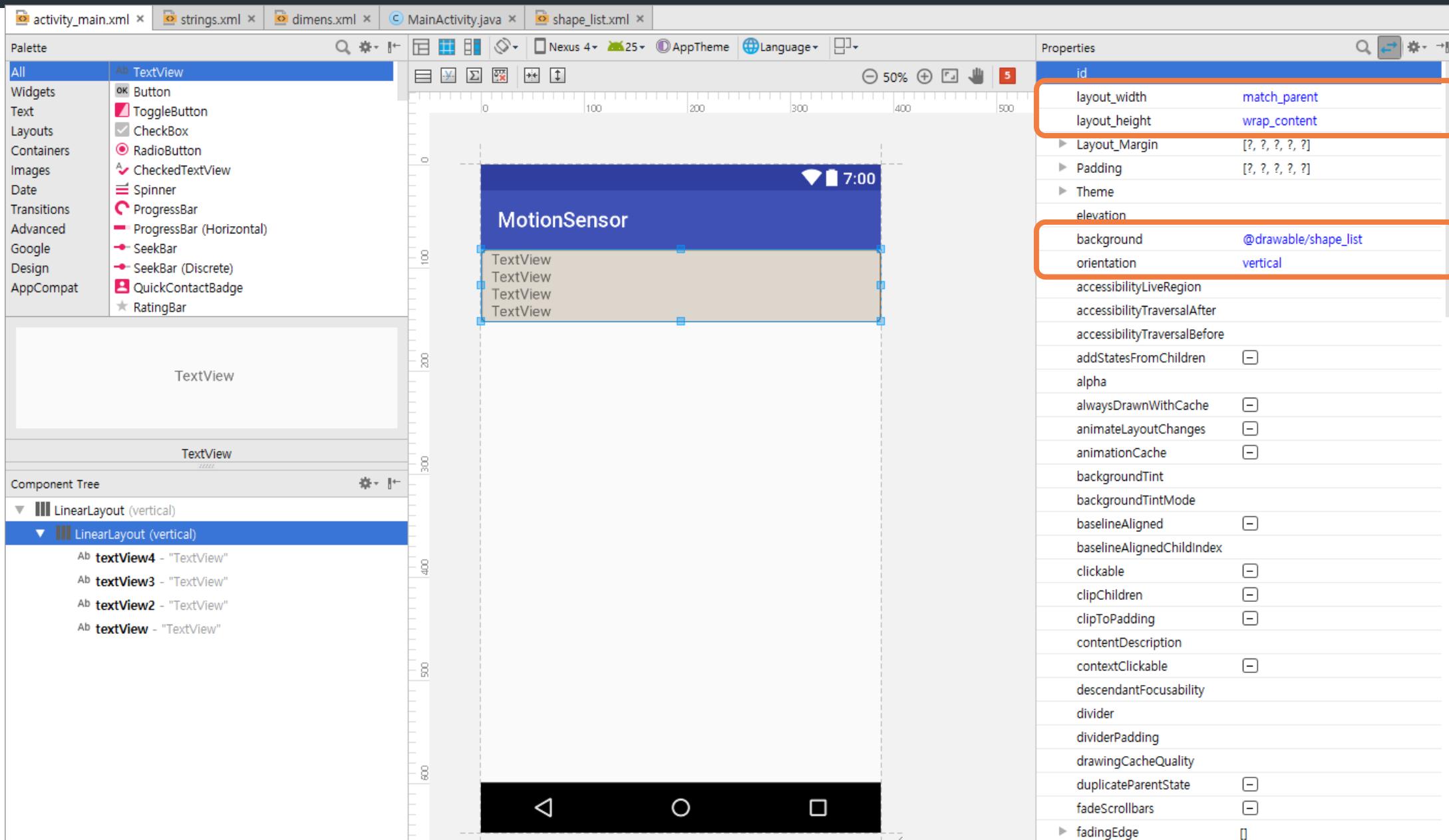


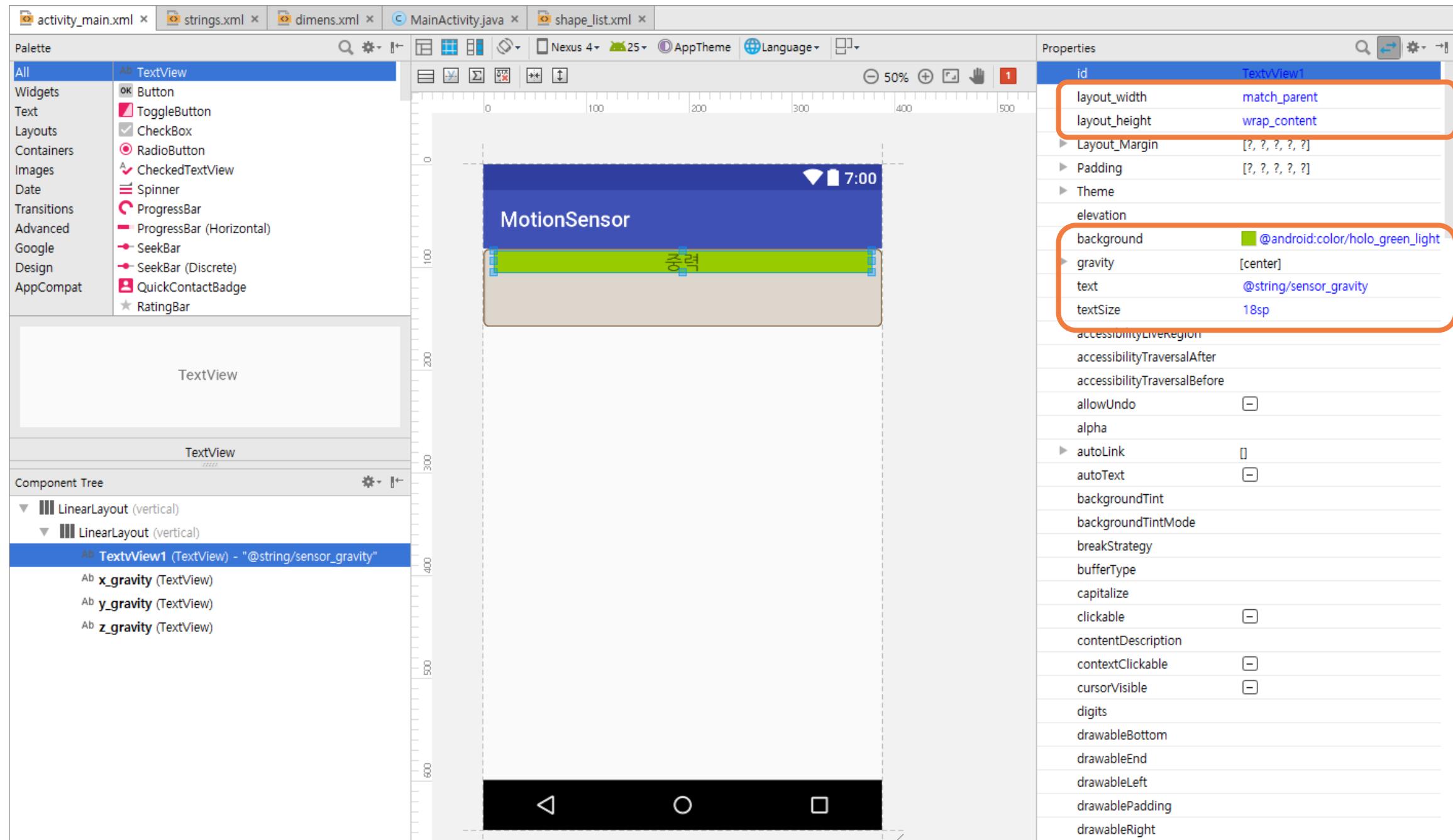
- dimens.xml

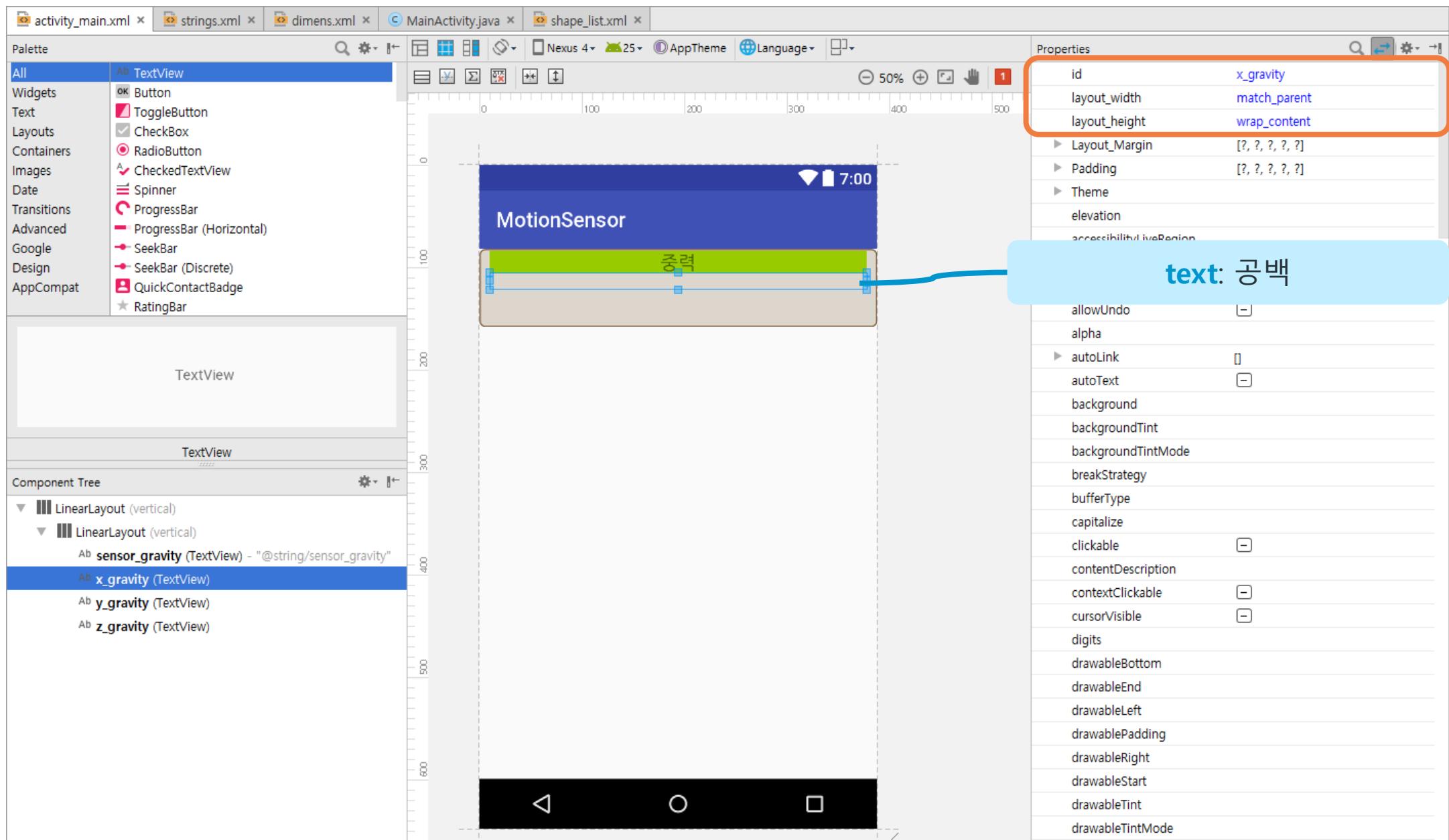
The screenshot shows the Android Studio interface with the dimens.xml file open in the code editor. The tab bar at the top includes activity_main.xml, strings.xml, dimens.xml (which is the active tab), MainActivity.java, and shape_list.xml. The code in dimens.xml defines two dimensions: 'activity_horizontal_margin' and 'activity_vertical_margin', both set to 8dp.

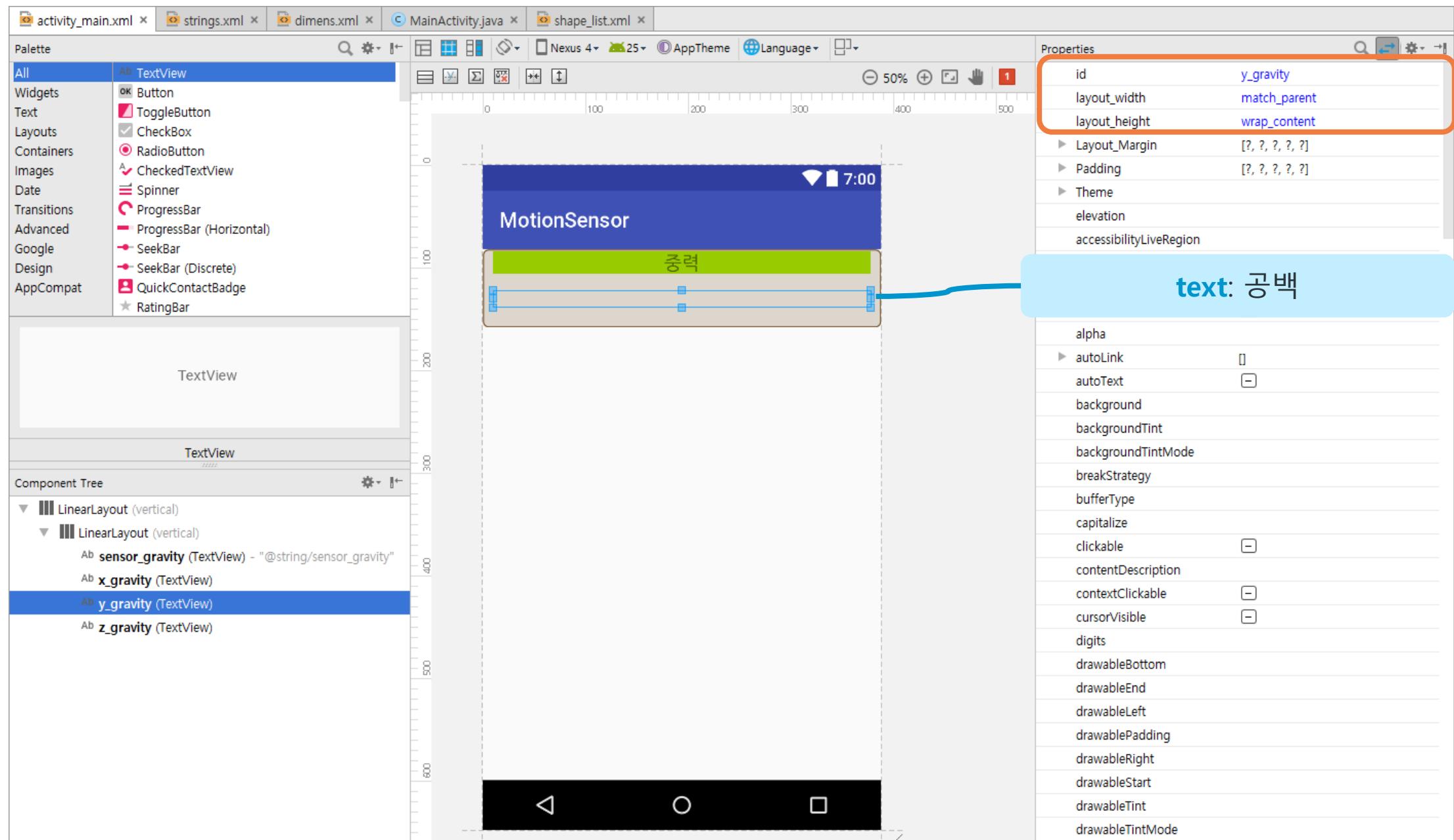
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="activity_horizontal_margin">8dp</dimen>
    <dimen name="activity_vertical_margin">8dp</dimen>
</resources>
```

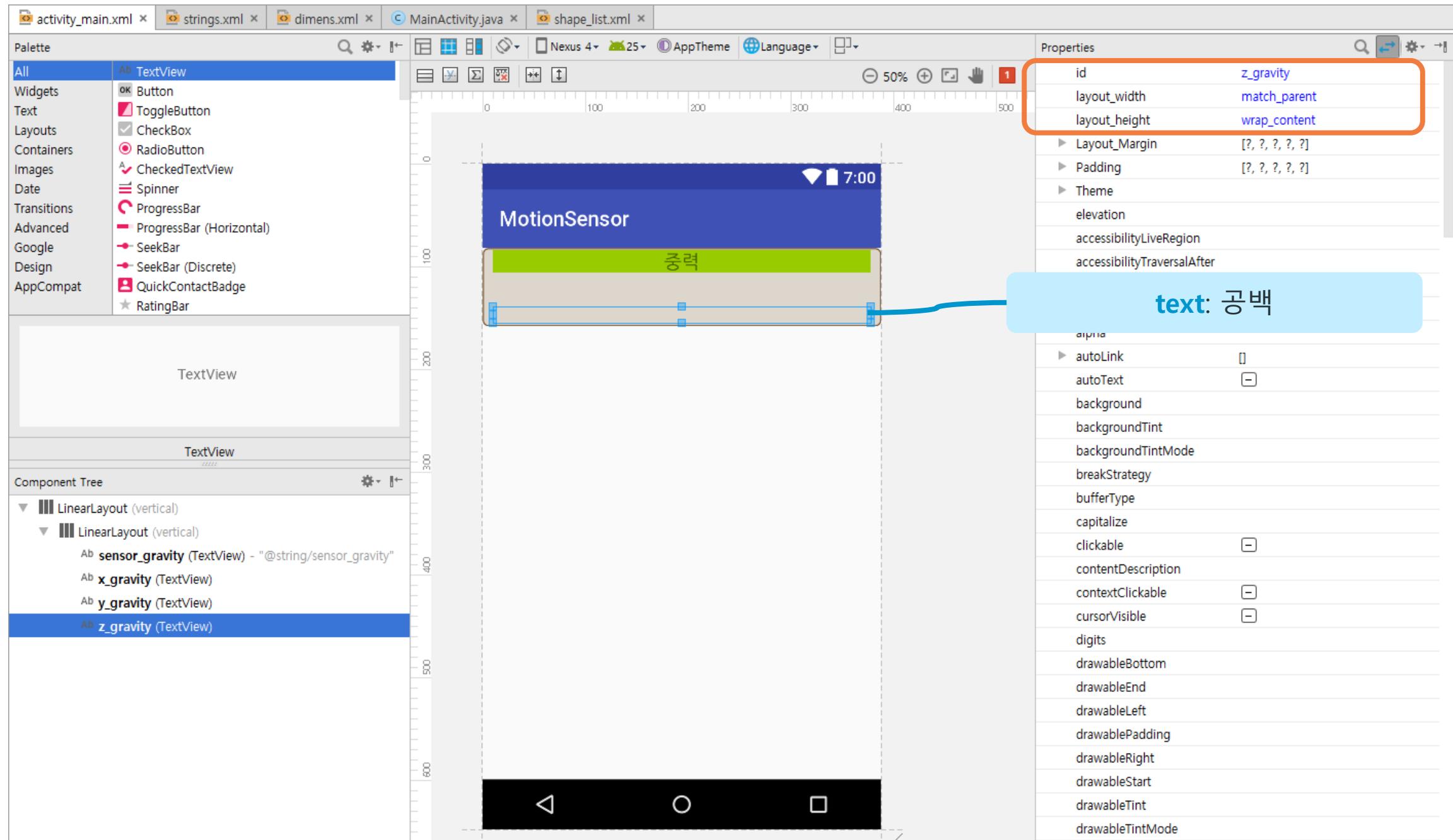
2.4 화면 설계-중력(입력하지 않음)





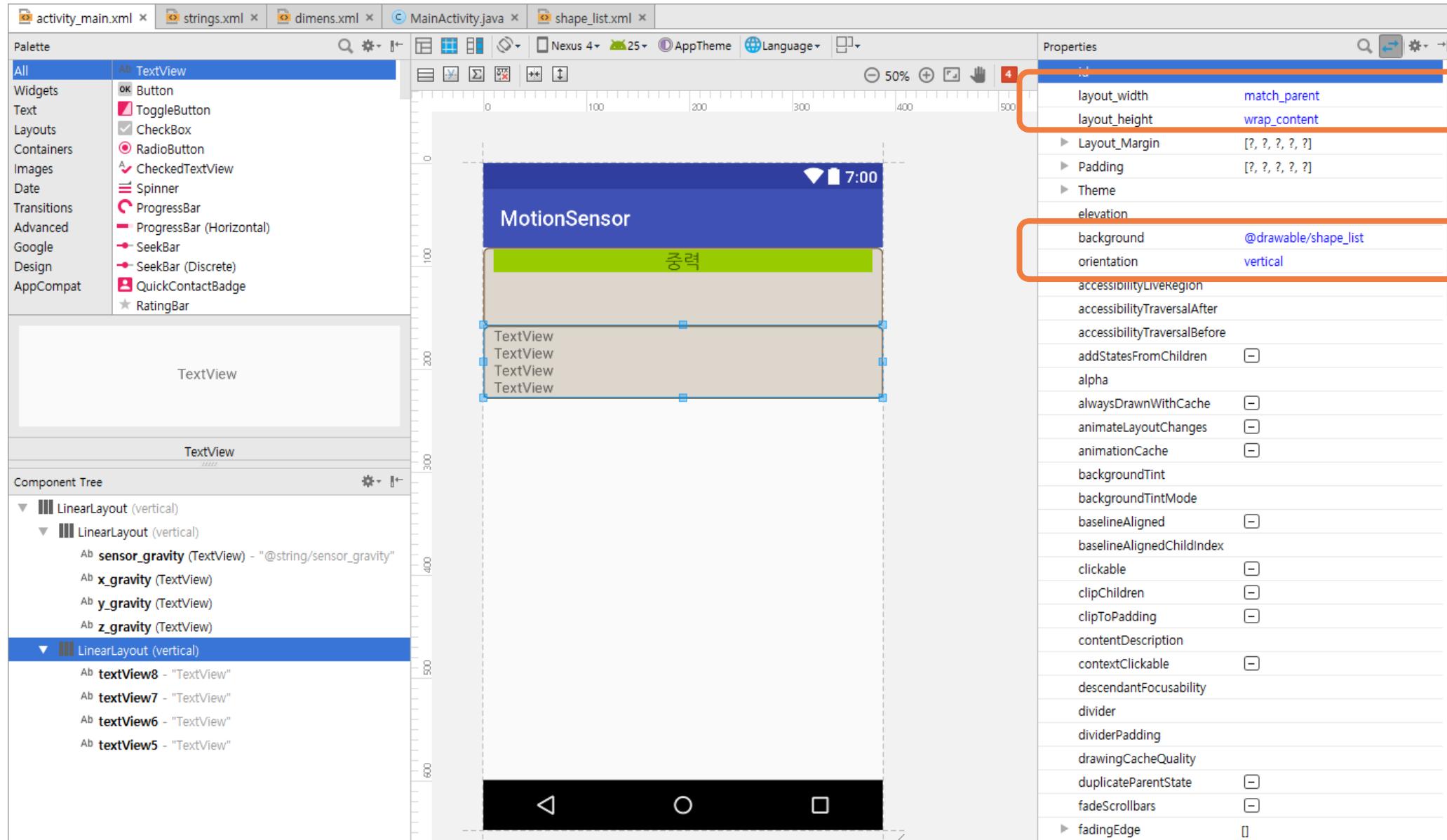


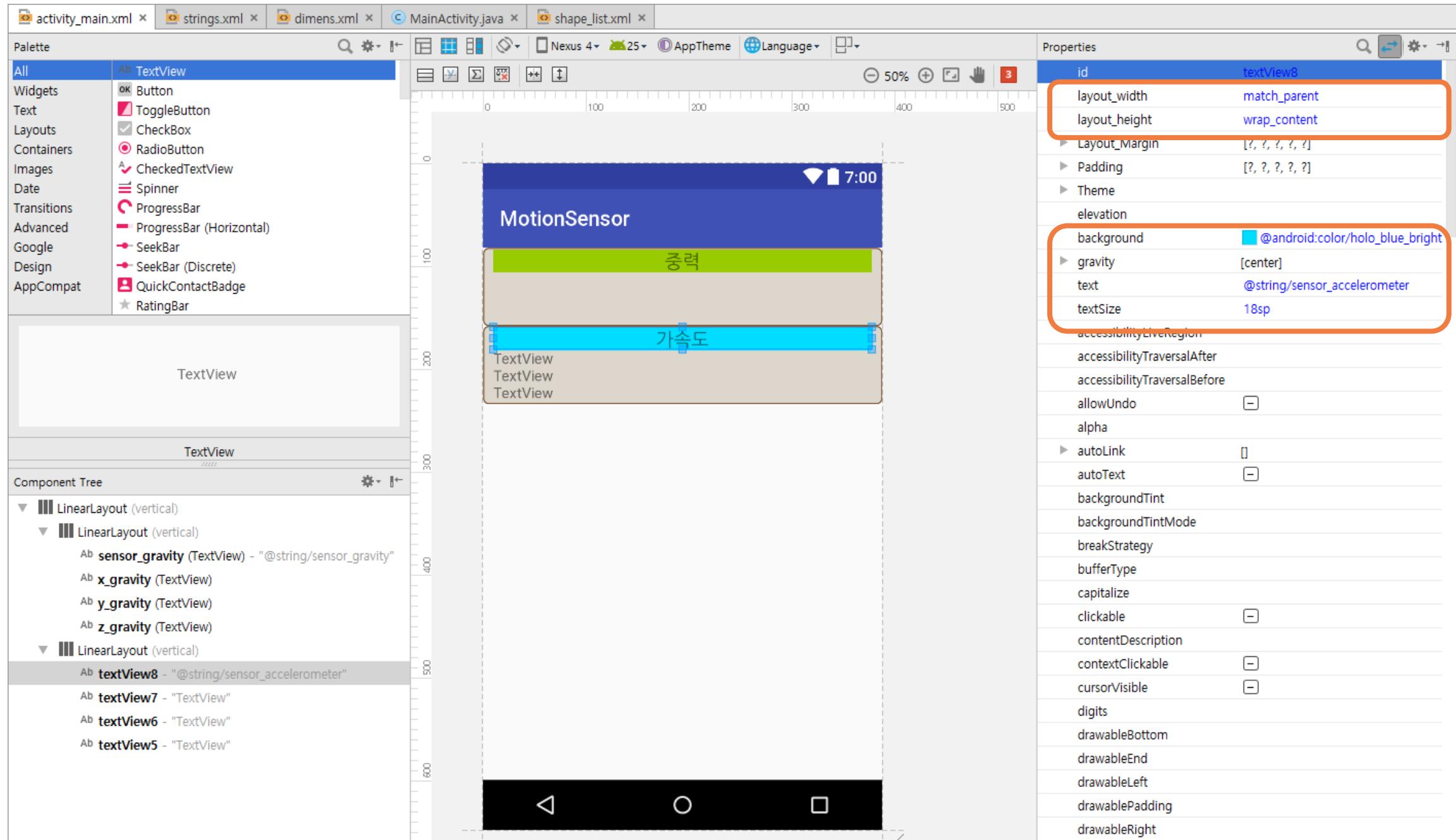


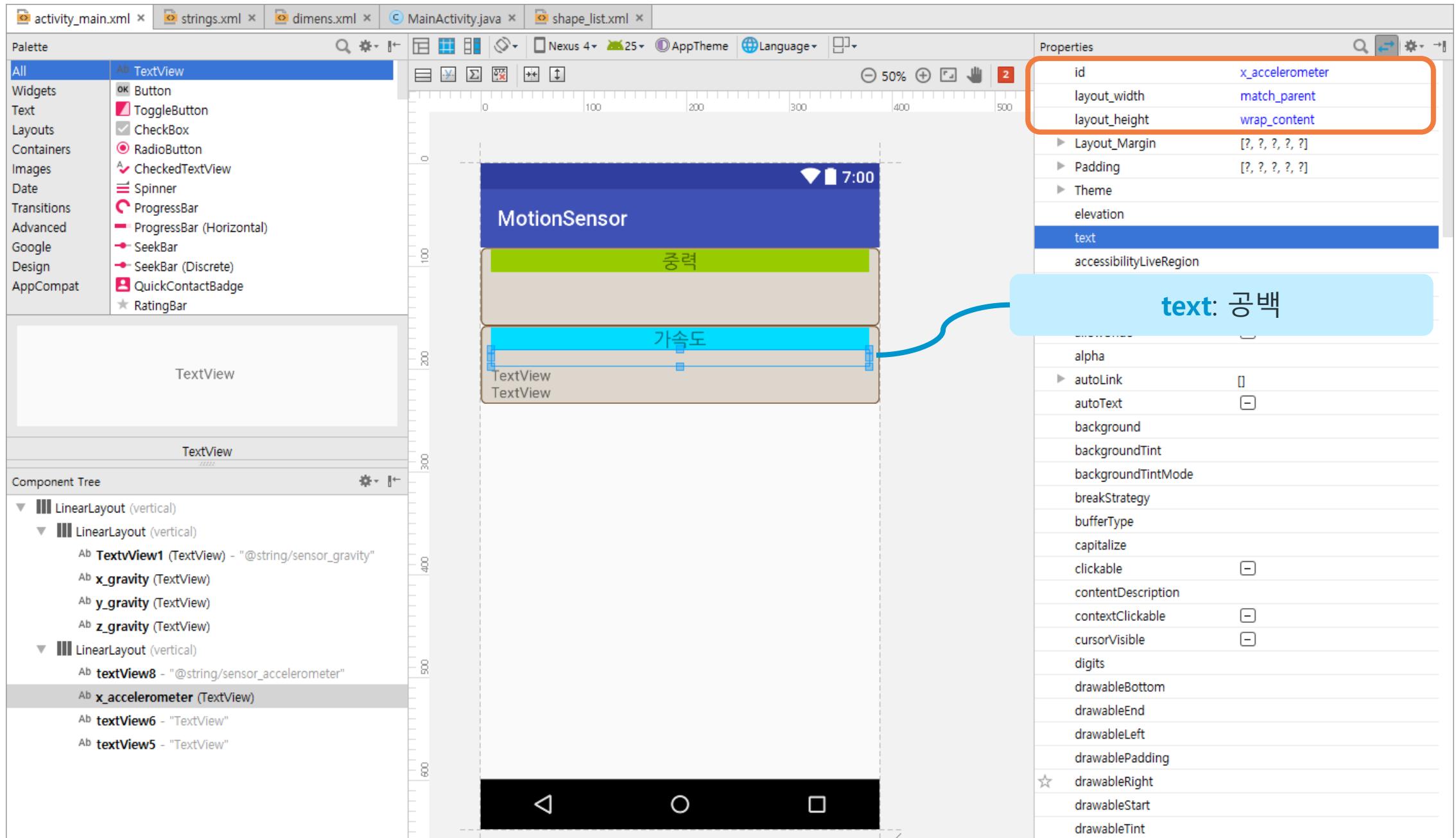


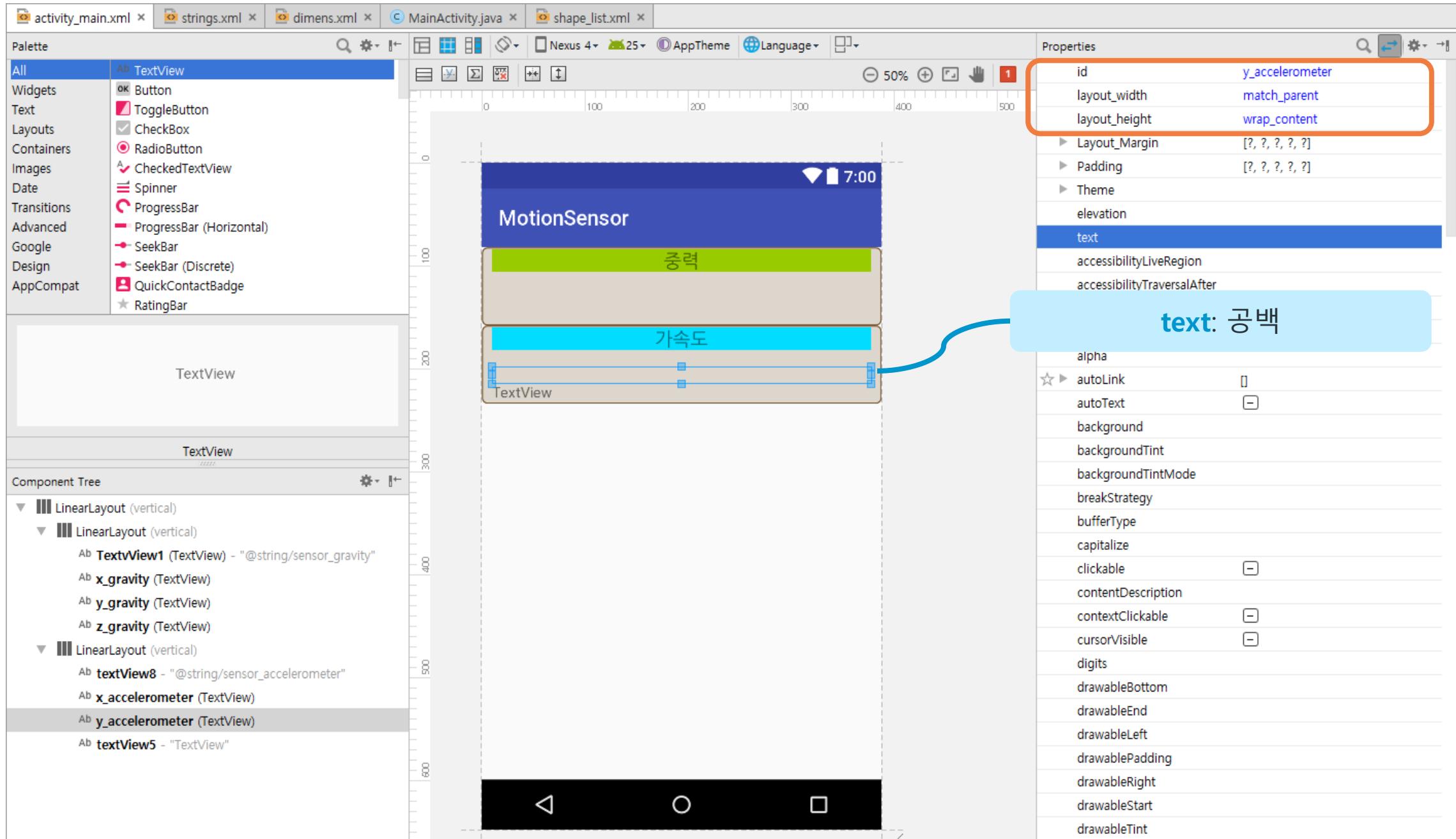
화면설계-가속도(가속도 만 입력)

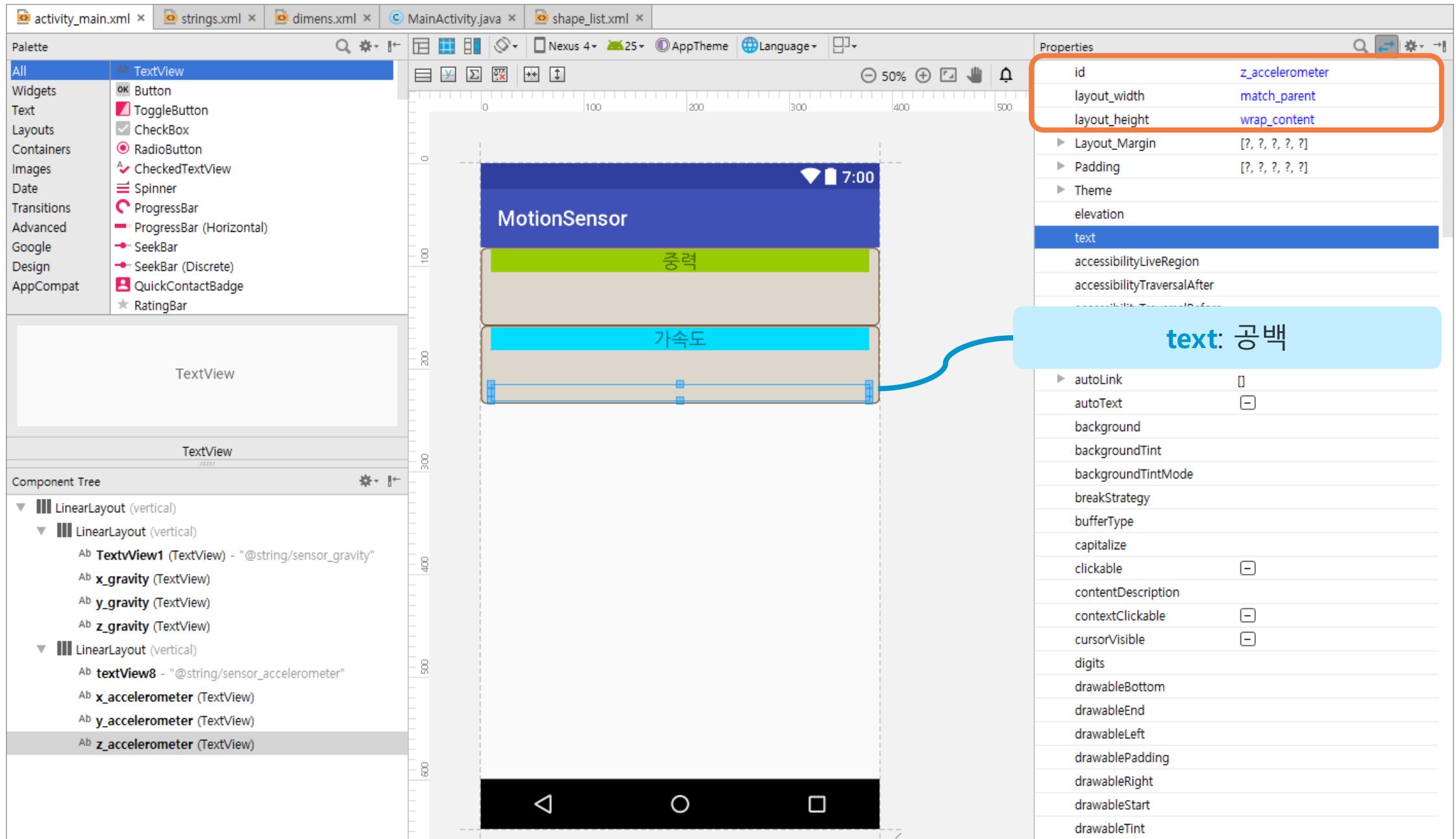
29





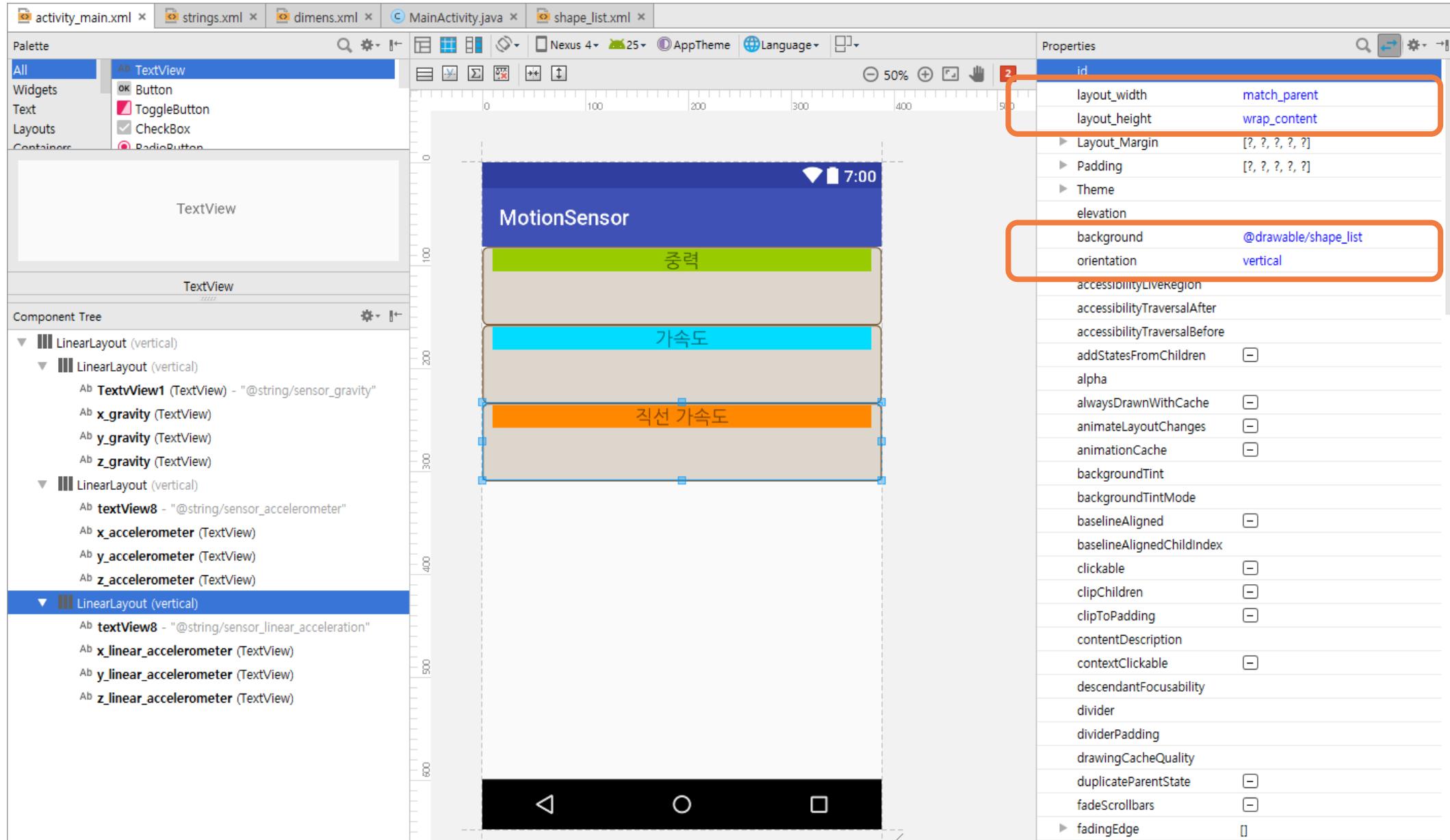


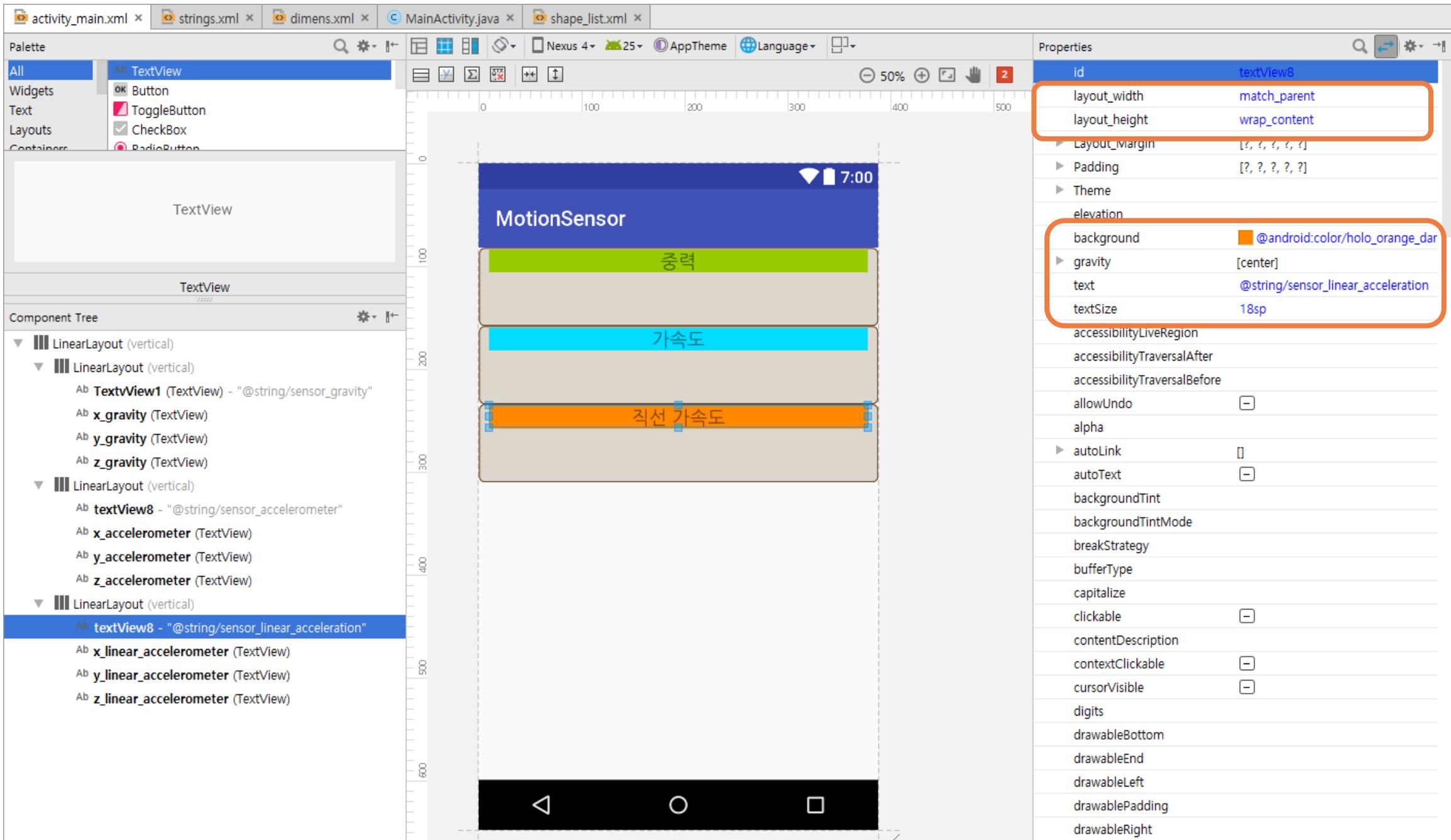


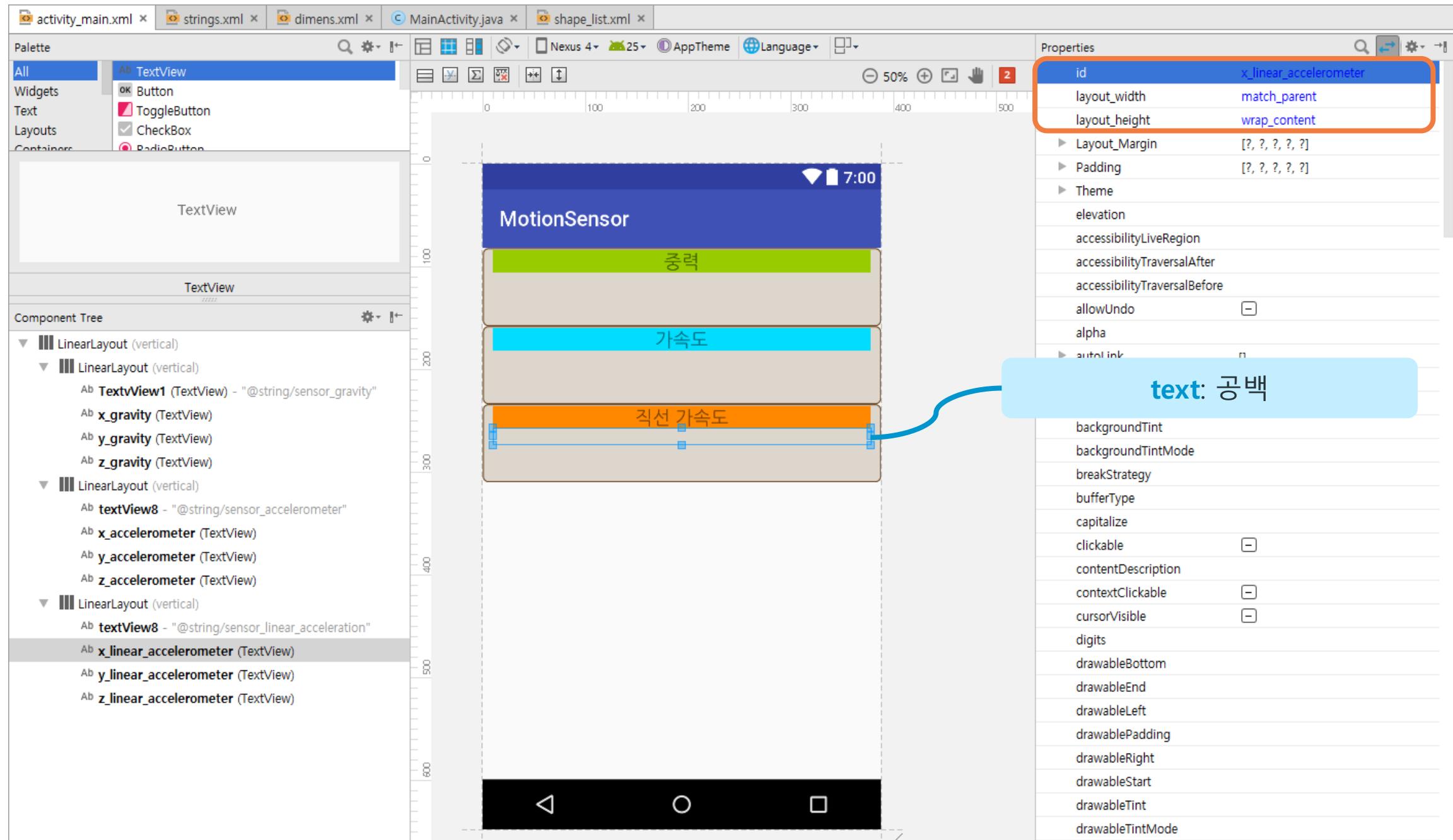


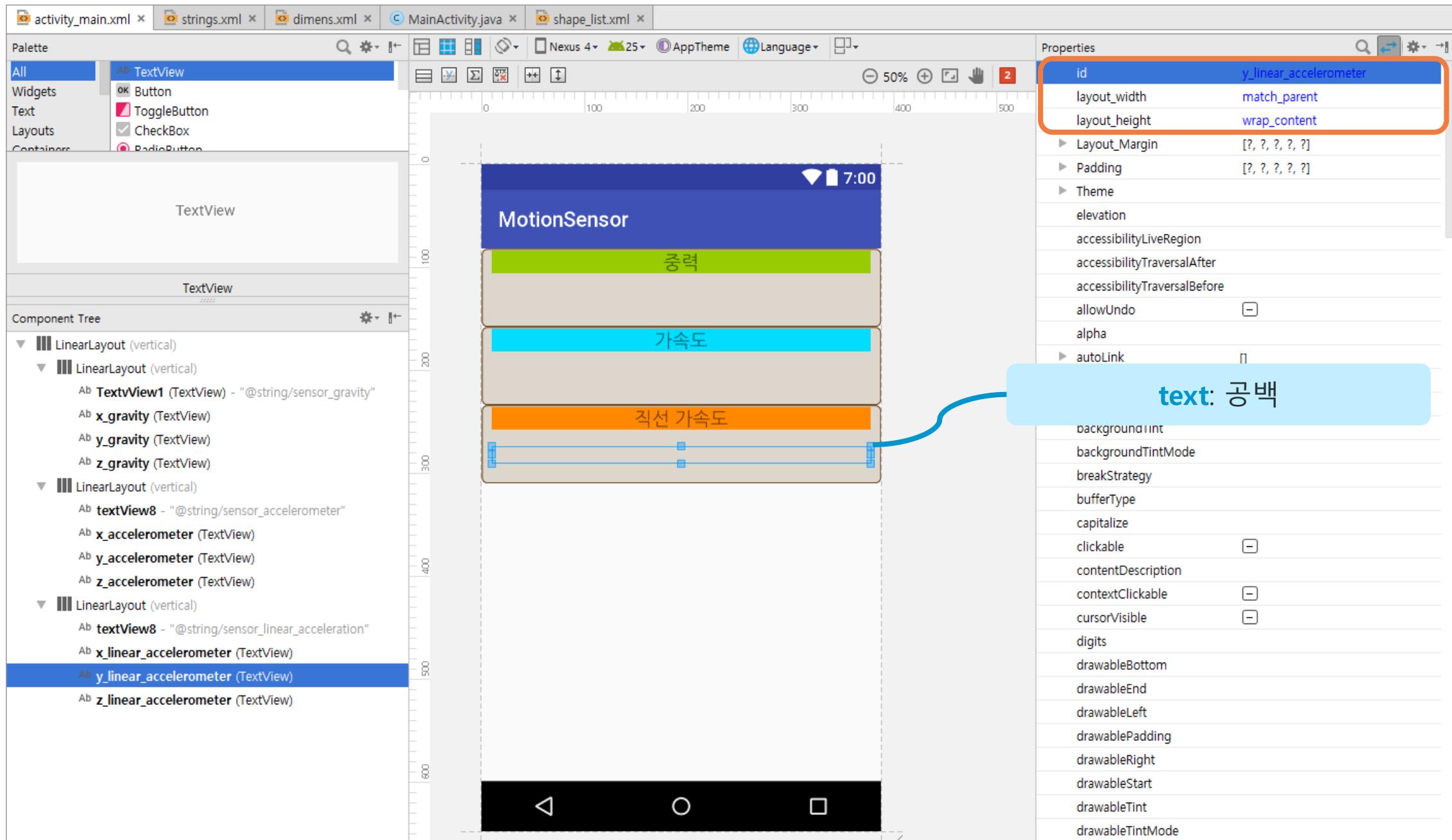
화면설계-직선가속도 (입력하지 않음)

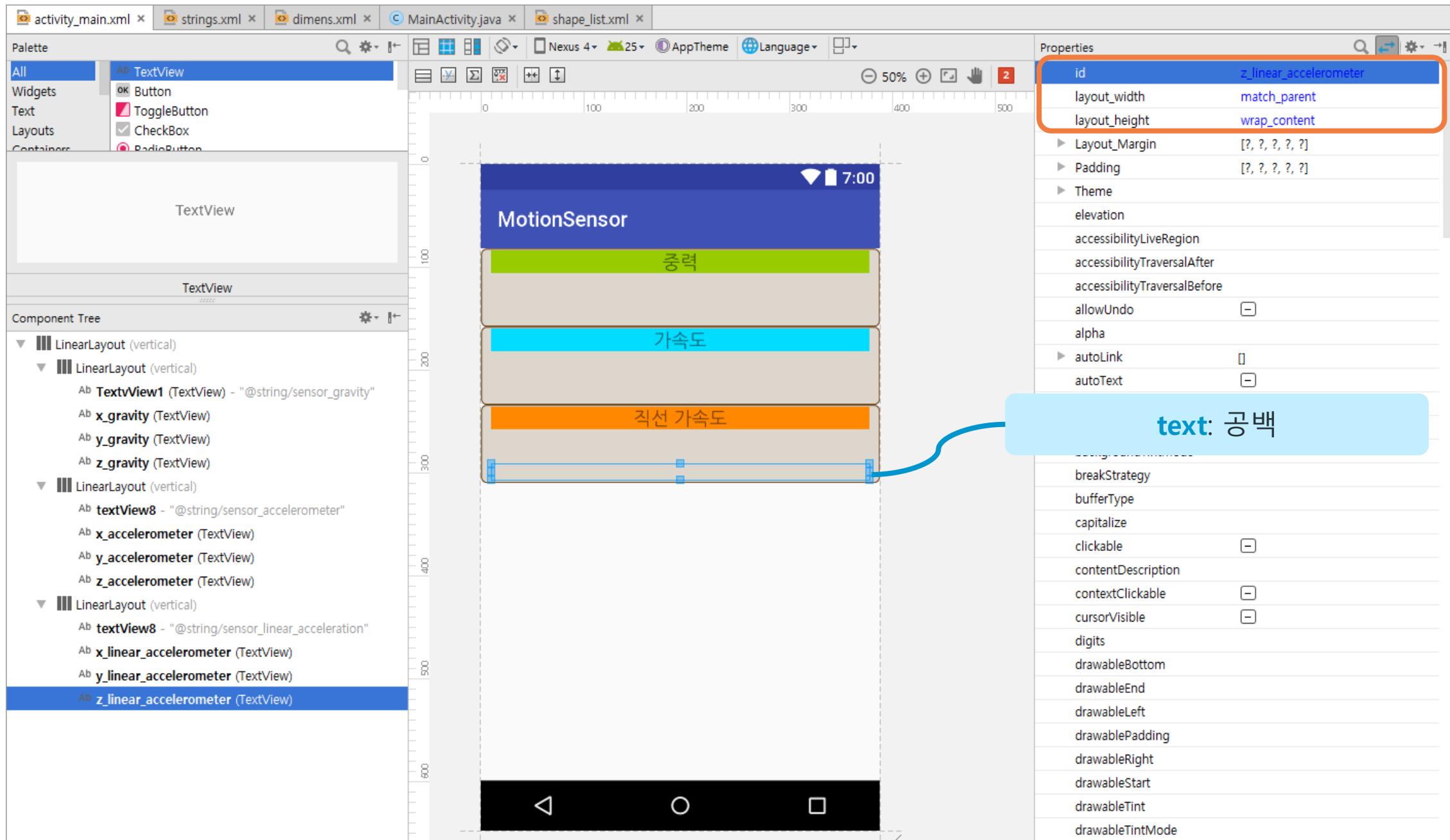
34





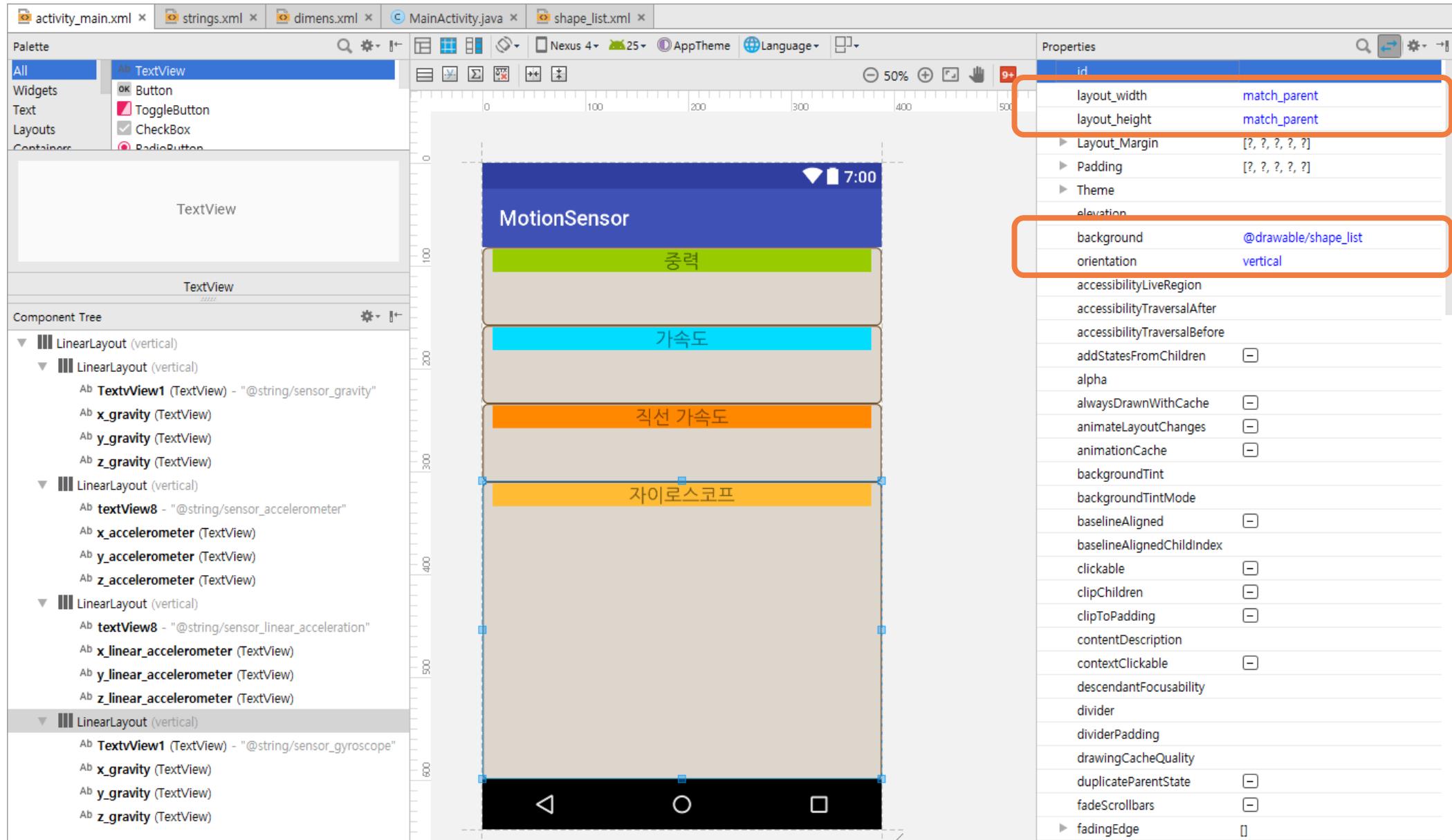


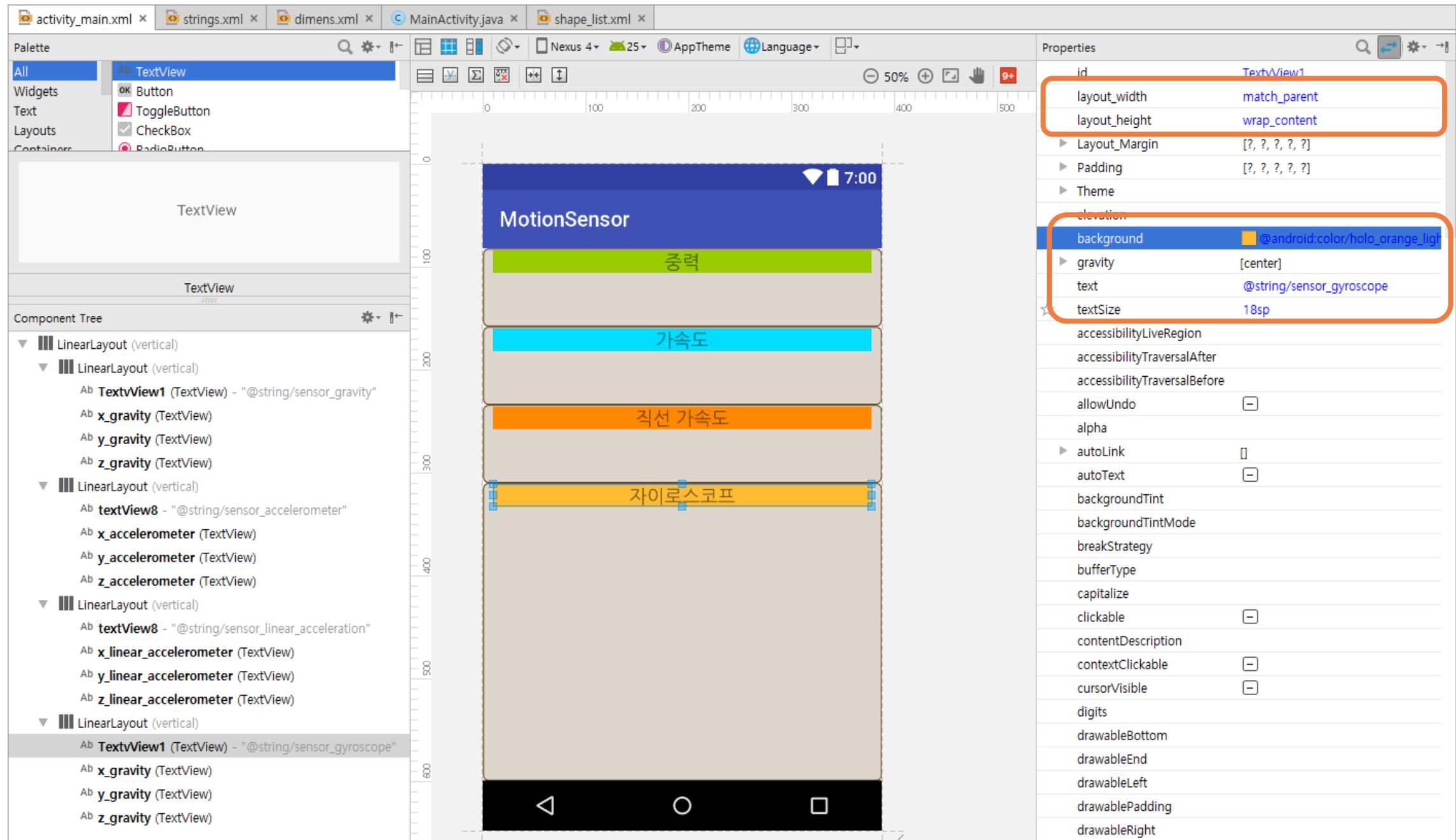


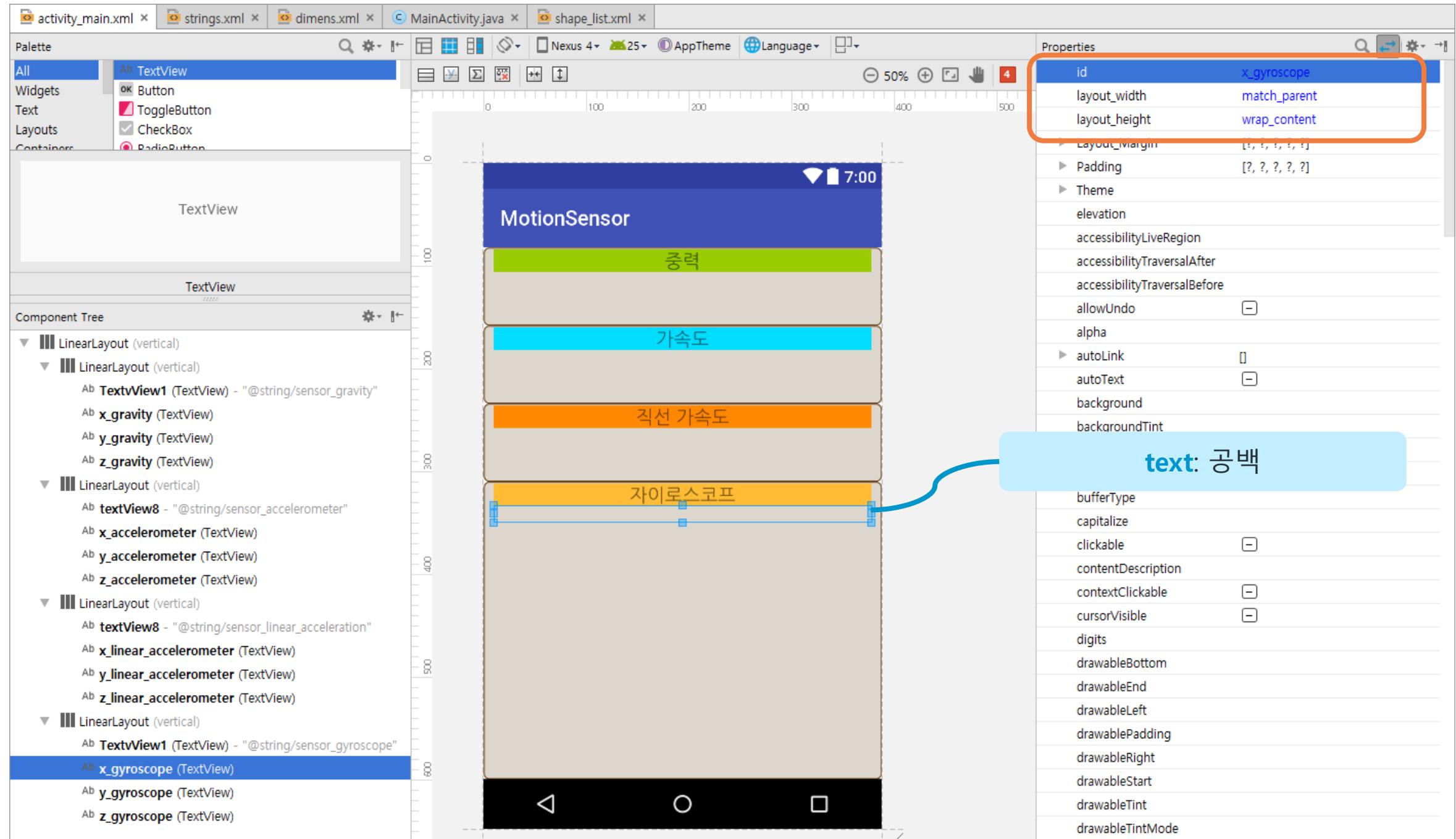


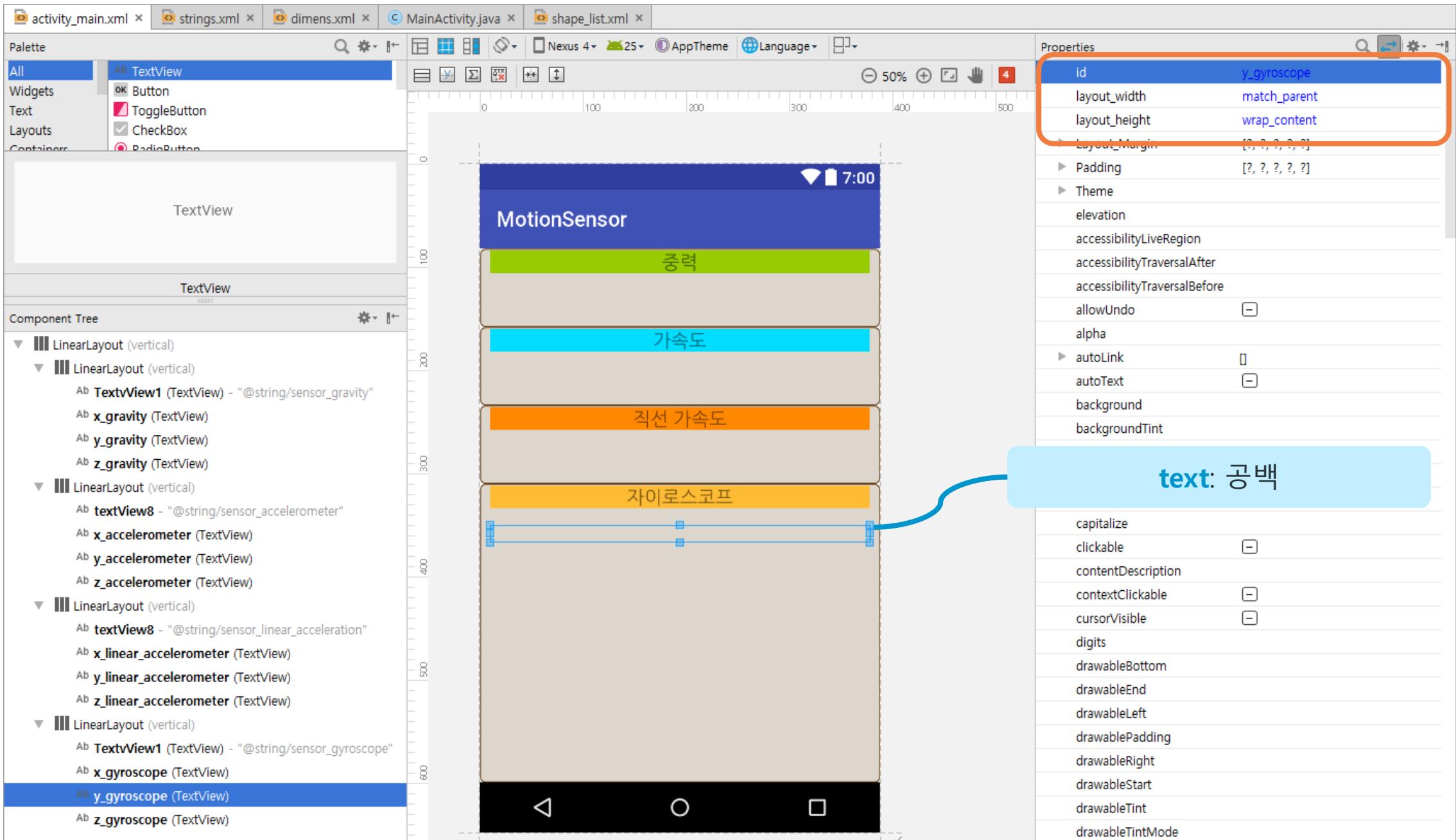
화면설계-자이로스코프 (입력하지 않음)

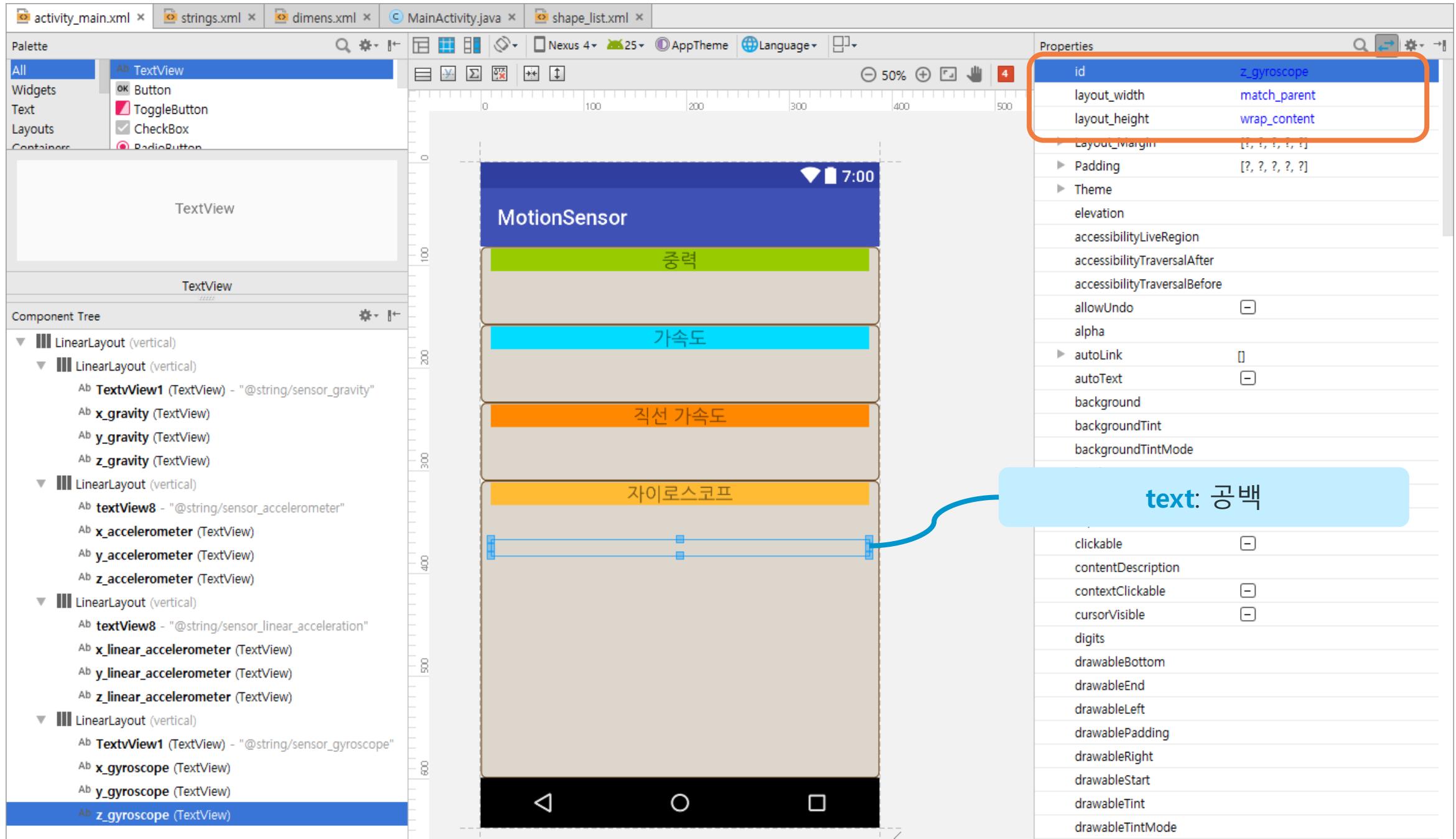
39











2.5 Activity 제어(MainActivity.java)

- 센서이벤트 처리를 위한 액티비티 인터페이스 추가

```

1 package com.example.kyungtae.motionsensor;
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

```

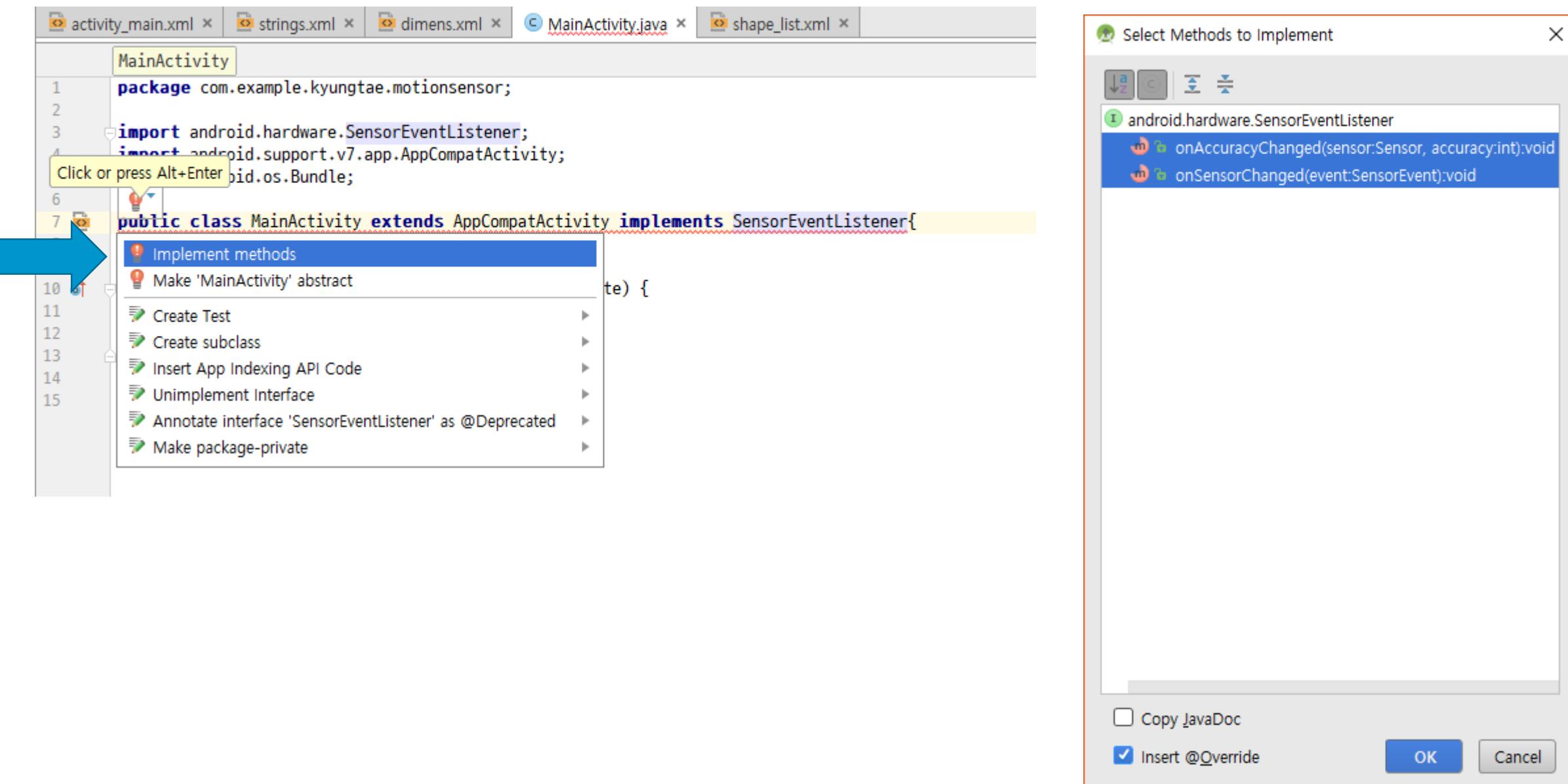
센서값 변화에 따른 이벤트 처리를
위한 클래스

```

1 package com.example.kyungtae.motionsensor;
2
3 import android.hardware.SensorEventListener;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6
7 public class MainActivity extends AppCompatActivity implements SensorEventListener {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14
15

```

• 센서값 처리를 위한 매소드 구현(@override)



```
activity_main.xml x strings.xml x dimens.xml x MainActivity.java x shape_list.xml x
```

```
1 package com.example.kyungtae.motionsensor;
2
3 import android.hardware.Sensor;
4 import android.hardware.SensorEvent;
5 import android.hardware.SensorEventListener;
6 import android.support.v7.app.AppCompatActivity;
7 import android.os.Bundle;
8
9 public class MainActivity extends AppCompatActivity implements SensorEventListener{
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_main);
15     }
16
17     @Override
18     public void onSensorChanged(SensorEvent event) {
19     }
20
21     @Override
22     public void onAccuracyChanged(Sensor sensor, int accuracy) {
23     }
24
25 }
26
27 }
```

센서 값이 변할 때 호출

등록된 센서의 정확도가 변할 때 호출

- 센서와 센서 값을 표시하기 위한 변수 선언 (빨간상자만 입력)

```
14  public class MainActivity extends AppCompatActivity implements SensorEventListener{  
15  
16      // x, y, z 중력 표시 텍스트뷰  
17      TextView x_gravity;  
18      TextView y_gravity;  
19      TextView z_gravity;  
20      // x, y, z 가속도 표시 텍스트뷰  
21      TextView x_accelerometer;  
22      TextView y_accelerometer;  
23      TextView z_accelerometer;  
24      // x, y, z 직선 가속도 표시 텍스트 뷰  
25      TextView x_linear_acceleration;  
26      TextView y_linear_acceleration;  
27      TextView z_linear_acceleration;  
28      // x, y, z 자이로스코프 텍스트 뷰  
29      TextView x_gyroscope;  
30      TextView y_gyroscope;  
31      TextView z_gyroscope;  
32  
33      // 센서 관리자 생성  
34      SensorManager sm;  
35  
36      // 센서  
37      Sensor sensor_gravity;  
38      Sensor sensor_accelerometer; Sensor sensor_accelerometer;  
39      Sensor sensor_linear_acceleration;  
40      Sensor sensor_gyroscope;  
41
```

• 센서값 출력을 위한 텍스트뷰 인식하기 (빨간상자만 입력)

```
42  
43 @Override  
44 protected void onCreate(Bundle savedInstanceState) {  
45     super.onCreate(savedInstanceState);  
46     setContentView(R.layout.activity_main);  
47  
48     // 중력 센서 값 출력을 위한 텍스트뷰 인식  
49     x_gravity = (TextView) findViewById(R.id.x_gravity);  
50     y_gravity = (TextView) findViewById(R.id.y_gravity);  
51     z_gravity = (TextView) findViewById(R.id.z_gravity);  
52  
53     // 중력가속도 센서 값 출력을 위한 텍스트뷰 인식  
54     x_accelerometer = (TextView) findViewById(R.id.x_accelerometer);  
55     y_accelerometer = (TextView) findViewById(R.id.y_accelerometer);  
56     z_accelerometer = (TextView) findViewById(R.id.z_accelerometer);  
57  
58     // 직선가속도 센서 값 출력을 위한 텍스트뷰 인식  
59     x_linear_acceleration = (TextView) findViewById(R.id.x_linear_accelerometer);  
60     y_linear_acceleration = (TextView) findViewById(R.id.y_linear_accelerometer);  
61     z_linear_acceleration = (TextView) findViewById(R.id.z_linear_accelerometer);  
62  
63     // 자이로스코프 센서 값 출력을 위한 텍스트뷰 인식  
64     x_gyroscope = (TextView) findViewById(R.id.x_gyroscope);  
65     x_gyroscope = (TextView) findViewById(R.id.x_gyroscope);  
66     x_gyroscope = (TextView) findViewById(R.id.x_gyroscope);  
67     sm = (SensorManager) getSystemService(SENSOR_SERVICE);  
68  
69     sensor_gravity = sm.getDefaultSensor(Sensor.TYPE_GRAVITY);  
70     sensor_accelerometer = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);  
71     sensor_linear_acceleration = sm.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);  
72     sensor_gyroscope = sm.getDefaultSensor(Sensor.TYPE_GYROSCOPE);  
73  
74 }
```

디바이스 센서 접근을 위한 객체 생성

중력 센서 측정을 위한 객체 생성

중력가속도 센서 측정을 위한 객체 생성

직선 가속도 센서 측정을 위한 객체 생성

자이로스코프 센서 측정을 위한 객체 생성

• 센서값이 변할 때 호출 (빨간상자만 입력)

```
48  
49 ❶@Override  
50 ❶public void onSensorChanged(SensorEvent event) {  
51      switch (event.sensor.getType()){ // 센서의 유형에 따른 실행  
52  
53          case Sensor.TYPE_GRAVITY:  
54              x_gravity.setText("X: " + event.values[0]);  
55              y_gravity.setText("Y: " + event.values[1]);  
56              z_gravity.setText("Z: " + event.values[2]);  
57              break;  
58  
59          case Sensor.TYPE_ACCELEROMETER:  
60              x_accelerometer.setText("X: " + event.values[0]);  
61              y_accelerometer.setText("Y: " + event.values[1]);  
62              z_accelerometer.setText("Z: " + event.values[2]);  
63              break;  
64  
65          case Sensor.TYPE_LINEAR_ACCELERATION:  
66              x_linear_acceleration.setText("X: " + event.values[0]);  
67              y_linear_acceleration.setText("Y: " + event.values[1]);  
68              z_linear_acceleration.setText("Z: " + event.values[2]);  
69              break;  
70  
71          case Sensor.TYPE_GYROSCOPE:  
72              x_gyroscope.setText("X: " + event.values[0]);  
73              y_gyroscope.setText("Y: " + event.values[1]);  
74              z_gyroscope.setText("Z: " + event.values[2]);  
75              break;  
76      }  
77 }
```

중력 센서 값 출력

중력가속도 센서 값 출력

직선가속도 센서 값 출력

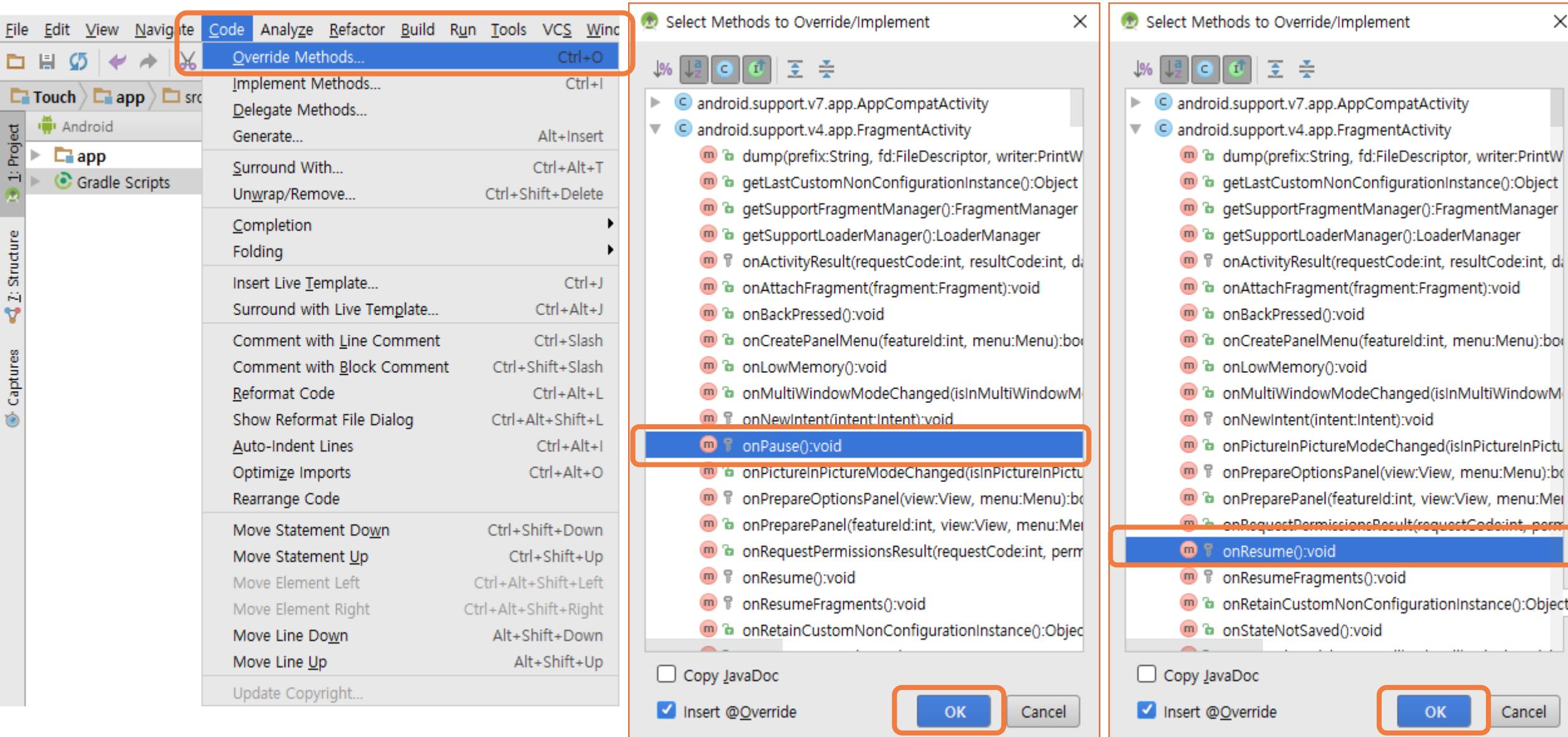
자이로스코프 센서 값 출력

Activity LifeCycle

메소드	설명	다음 메소드
onCreate()	액티비티가 생성될 때 호출되며 사용자 인터페이스 초기화에 사용됨.	onStart()
onRestart()	액티비티가 멈췄다가 다시 시작되기 바로 전에 호출됨.	onStart()
onStart()	액티비티가 사용자에게 보여지기 바로 직전에 호출됨.	onResume() 또는 onStop()
onResume()	액티비티가 사용자와 상호작용하기 바로 전에 호출됨.	onPause()
onPause()	다른 액티비티가 보여질 때 호출됨. 데이터 저장, 스레드 중지 등의 처리를 하기에 적당한 메소드.	onResume() 또는 onStop()
onStop()	액티비티가 더이상 사용자에게 보여지지 않을 때 호출됨. 메모리가 부족할 경우에는 onStop() 메소드가 호출되지 않을 수도 있음.	onRestart() 또는 onDestroy()
onDestroy()	액티비티가 소멸될 때 호출됨. finish() 메소드가 호출되거나 시스템이 메모리 확보를 위해 액티비티를 제거할 때 호출됨.	없음

onPause()/onResume() 메소드 재정의(Override)

51



- 재정의를 위한 매소드 추가

```
40  
41 ↗ @Override  
42     protected void onPause() {  
43         super.onPause();  
44     }  
45  
46 ↗ @Override  
47     protected void onResume() {  
48         super.onResume();  
49     }
```

화면에 표시되는 상태에서 사용자와 상호 작용하지 않을 때

액티비티가 일시정지(pause)상태에서 복귀할 때 호출

• onPause() / onResume() 매소드 재정의(빨간상자만 추가입력)

```
40  
41     @Override  
42     protected void onPause() {  
43         super.onPause();  
44  
45         sm.unregisterListener(this);  
46  
47     @Override  
48     protected void onResume() {  
49         super.onResume();  
50  
51         sm.registerListener(this, sensor_gravity, SensorManager.SENSOR_DELAY_NORMAL);  
52         sm.registerListener(this, sensor_accelerometer, SensorManager.SENSOR_DELAY_NORMAL);  
53         sm.registerListener(this, sensor_linear_acceleration, SensorManager.SENSOR_DELAY_NORMAL);  
54         sm.registerListener(this, sensor_gyroscope, SensorManager.SENSOR_DELAY_NORMAL);  
55  
56 }
```

등록된 센서 리스너 해제

중력센서 값 변화에 대한 이벤트를 감지하기 위한 리스너 등록

(스크린 방향 변화에 적당한 비율)

자이로스코프 센서 값 변화에 대한 이벤트를 감지하기 위한 리스너 등록

직선가속도 센서 값 변화에 대한 이벤트를 감지하기 위한 리스너 등록

클래스와 속성/메소드

• 클래스

클래스/인터페이스	설명
Sensor	센서 표현을 위한 클래스
SensorEvent	센서 이벤트를 표현하는 클래스
SensorEventListener	센서 값이 변할 때 센서 매니저로부터 공지를 받는 데 사용
SensorManager	디바이스의 센서에 접근할 수 있도록 함.

• 상수

클래스	상수	설명
Context	String <code>SENSOR_SERVICE</code>	센서 이용을 위한 센서 매니저를 추출하기 위해 <code>getSystemService(Class)</code> 와 함께 사용
Sensor	int <code>TYPE_GRAVITY</code>	중력센서 타입을 기술하는 상수
	int <code>TYPE_ACCELEROMETER</code>	가속도계 타입을 기술하는 상수
	int <code>TYPE_LINEAR_ACCELERATION</code>	직선 가속도 센서 타입을 기술하는 상수
	int <code>TYPE_GYROSCOPE</code>	자이로스코프 센서 타입을 기술하는 상수
SensorManager	int <code>SENSOR_DELAY_NORMAL</code>	스크린 방향 변화에 적당한 비율

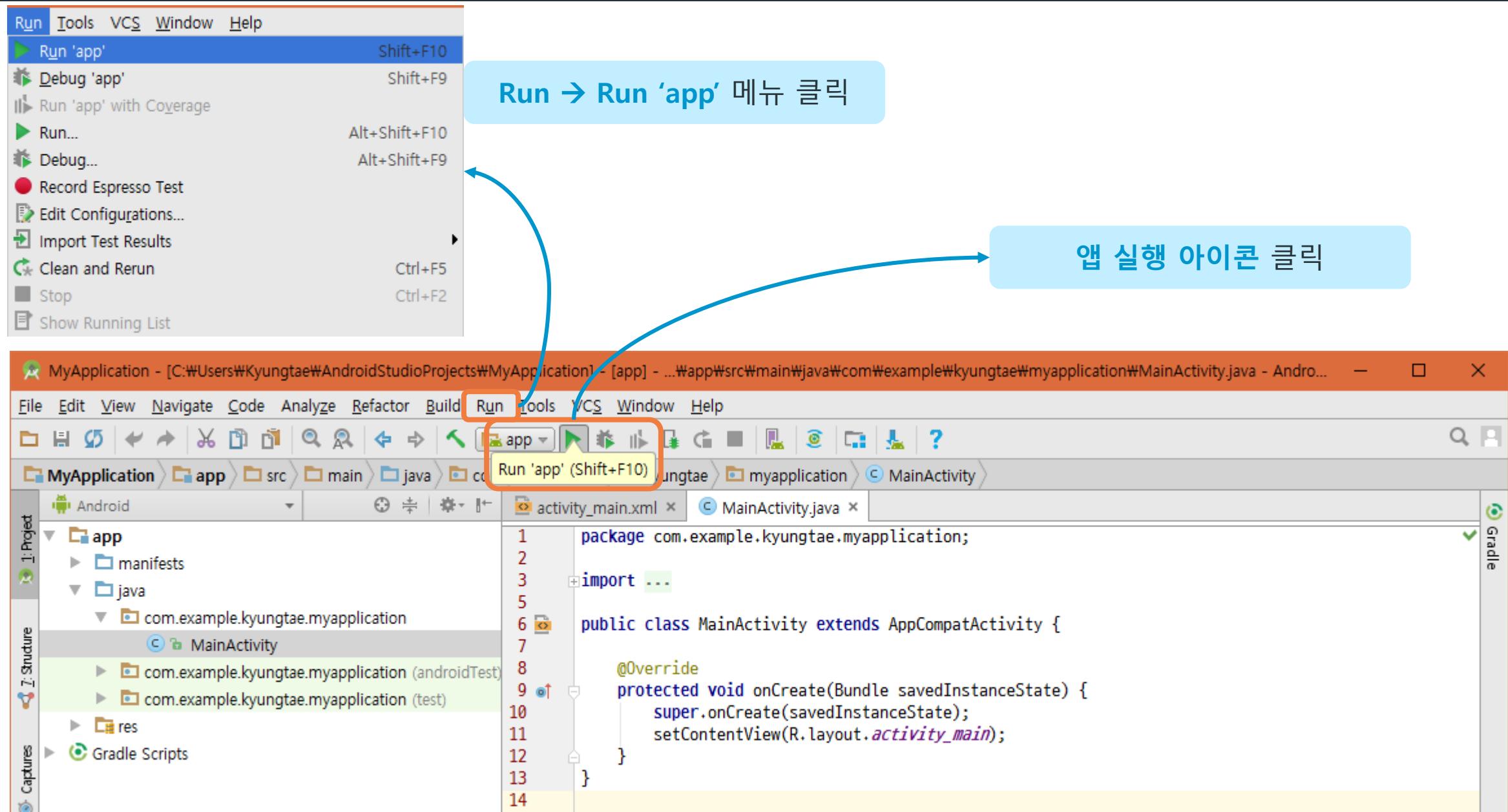
클래스와 속성/메소드

• 메소드

클래스	메소드	설명
Sensor	int <code>getType()</code>	센서 타입을 반환함
SensorEvent	public final float[] <code>values</code>	센서가 인식한 값들을 저장하는 배열
SensorEventListener	abstract void <code>onAccuracyChanged(Sensor, int accuracy)</code>	등록된 센서의 정확도가 변할 때 호출됨
	abstract void <code>onSensorChanged(SenseorEvent event)</code>	센서 값이 변할 때 호출됨
SensorManager	Sensor <code>getDefaultSensor(int type)</code>	주어진 type을 위한 디폴트 센서를 얻기 위해 사용
	Boolean <code>registerListener(SensorEventListener listener, Sensor sensor, int samplingPeriodUs)</code>	주어진 샘플링 주파수에서 주어진 센서를 위한 SensorEventListener를 등록함
	void <code>unregisterListener(SensorEventListener listener)</code>	모든 센서에 대한 리스너를 해제함

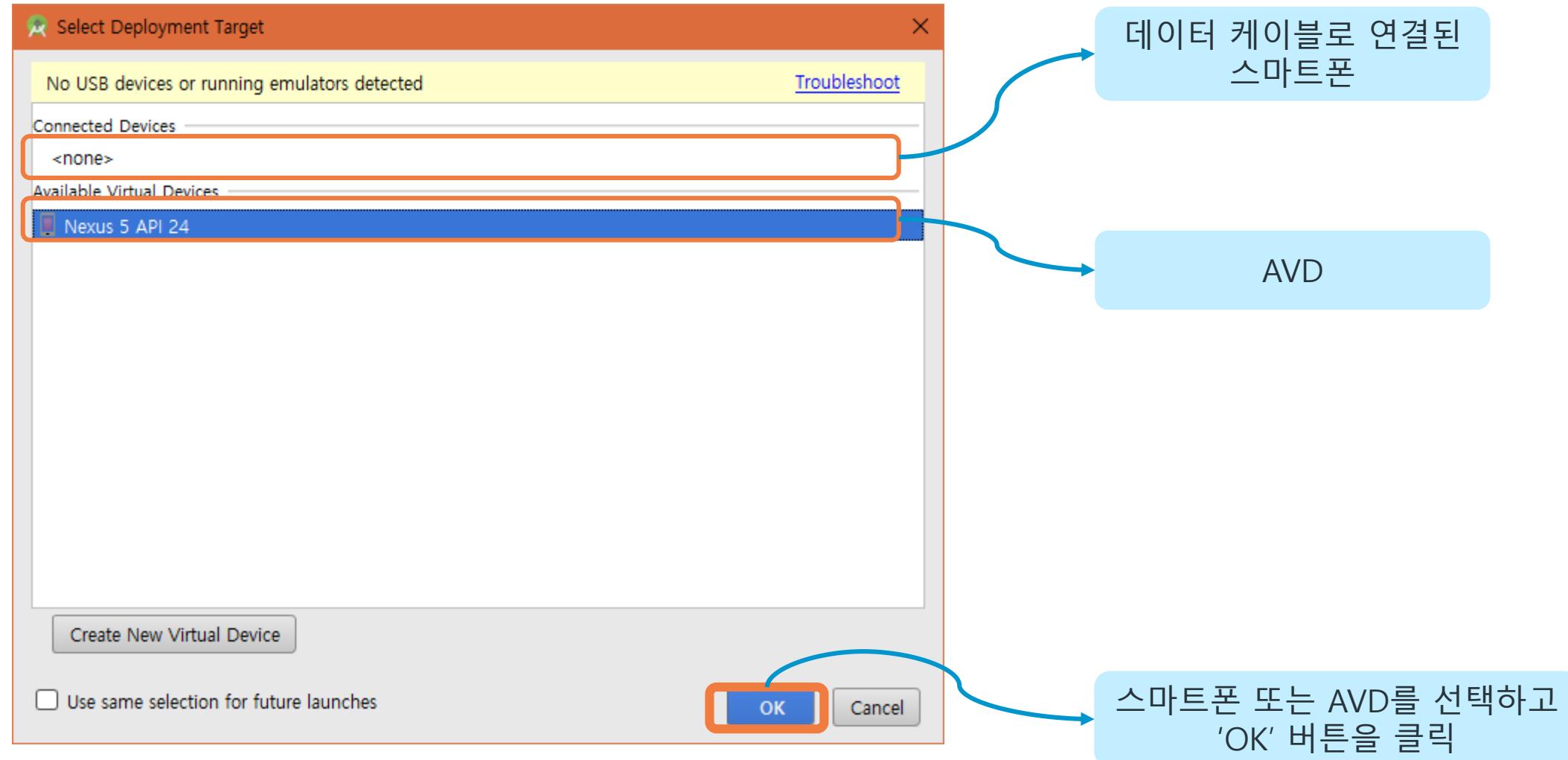
Step 3. 프로젝트 실행

56



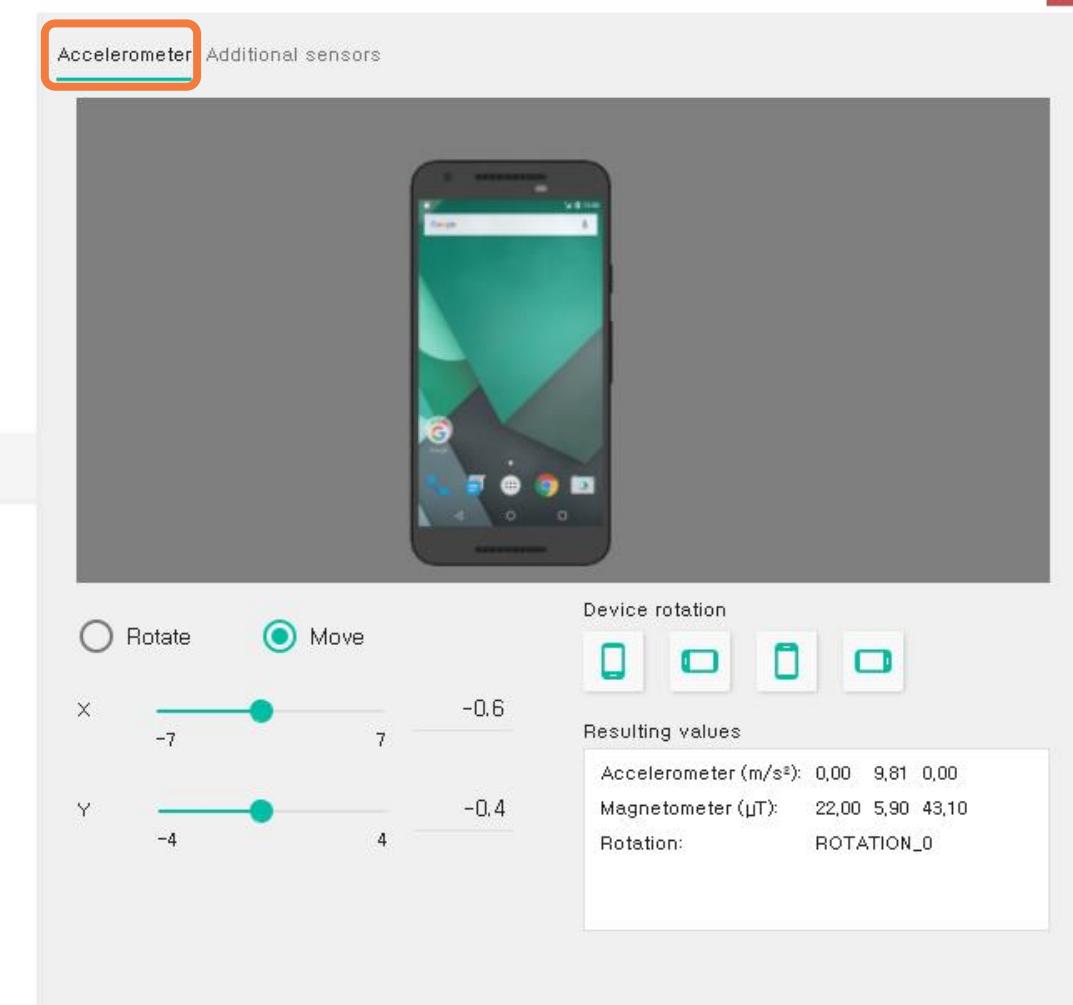
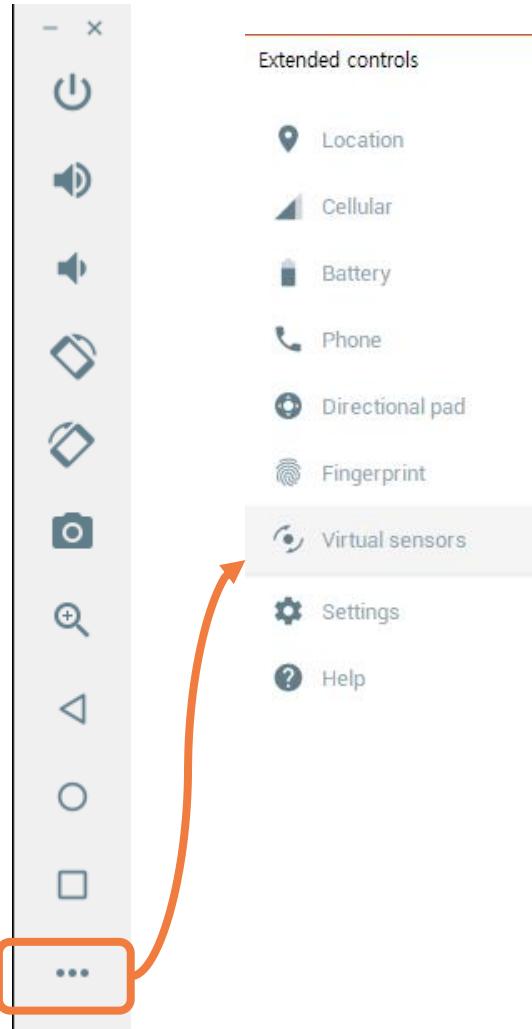
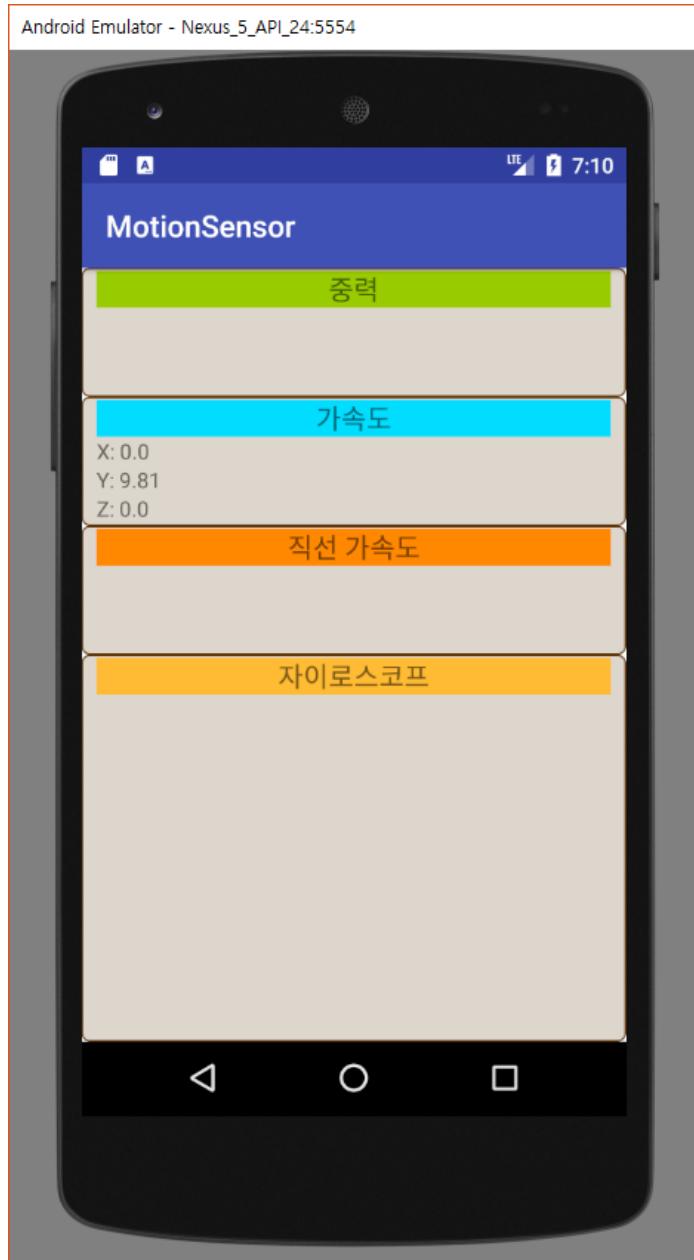
• AVD 장비 선택하기

57



• 실행 결과

58



O outputs



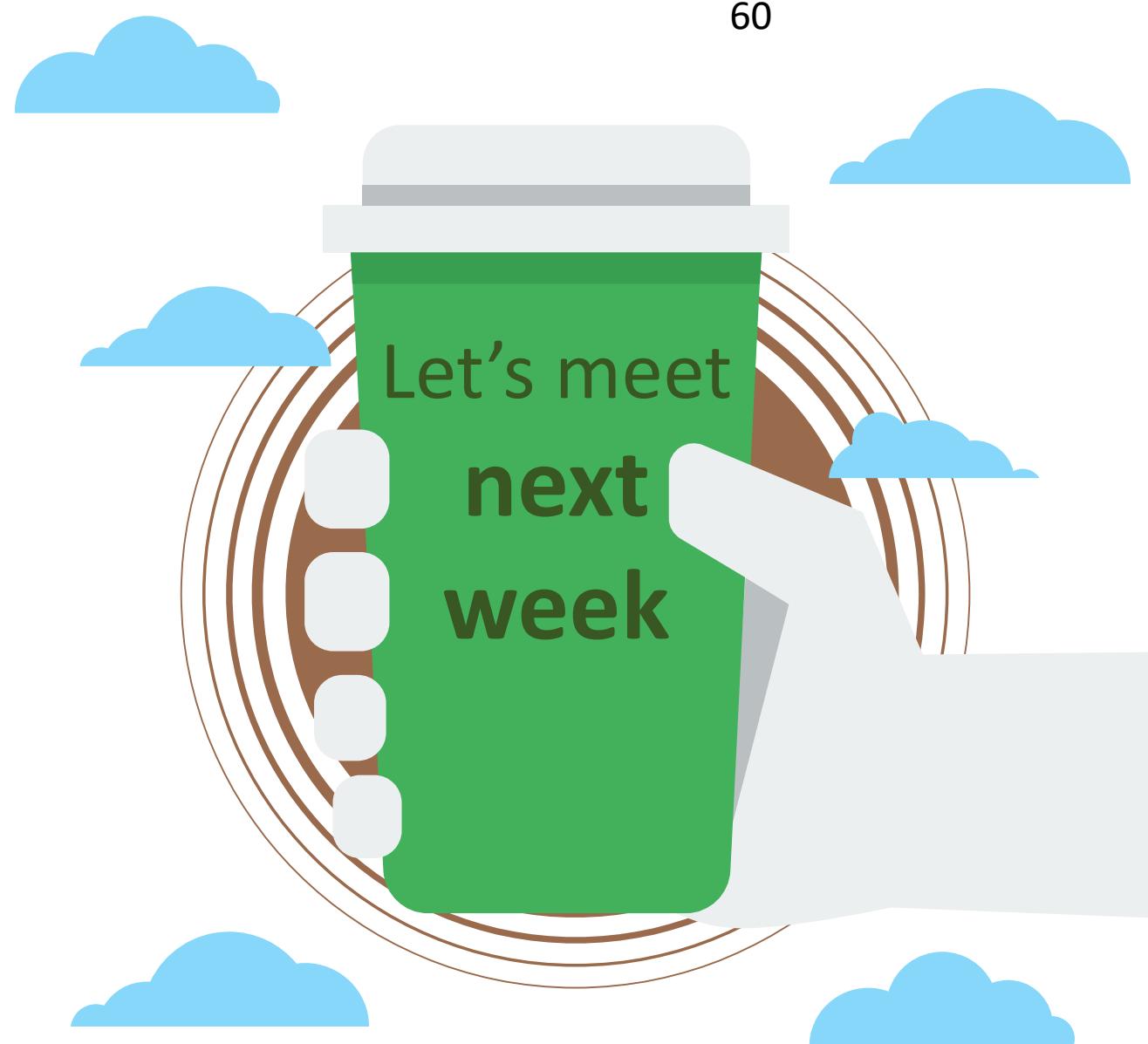


question

&



answer



60