

Android - Day 1

Widget을 만나러...



학습목표

이 학습을 마치면...

- Toast를 사용하여 사용자를 위한 메시지를 표시 할 수 있습니다.
- Log를 사용하여 개발자를 위한 메시지를 표시 할 수 있습니다.
- 위젯에 대해 알고 위젯의 크기를 조정할 수 있습니다.
- Button 위젯을 사용하여 버튼을 누르면
버튼의 이벤트를 받아서 메시지를 출력합니다.
- Image Button 위젯을 사용하여 이미지가 있는 버튼을 사용합니다.
- TextView 위젯을 사용하여 문자열을 표시합니다.
- EditText를 이용하여 사용자가 입력한 문자를 받아옵니다.
- ImageView를 사용하여 Drawable에 있는 사진을 표시합니다.
- LinearLayout을 사용하여 자신이 원하는 레이아웃을 구성합니다.

메시지 출력하기

Toast



HelloWorld1

hello toast

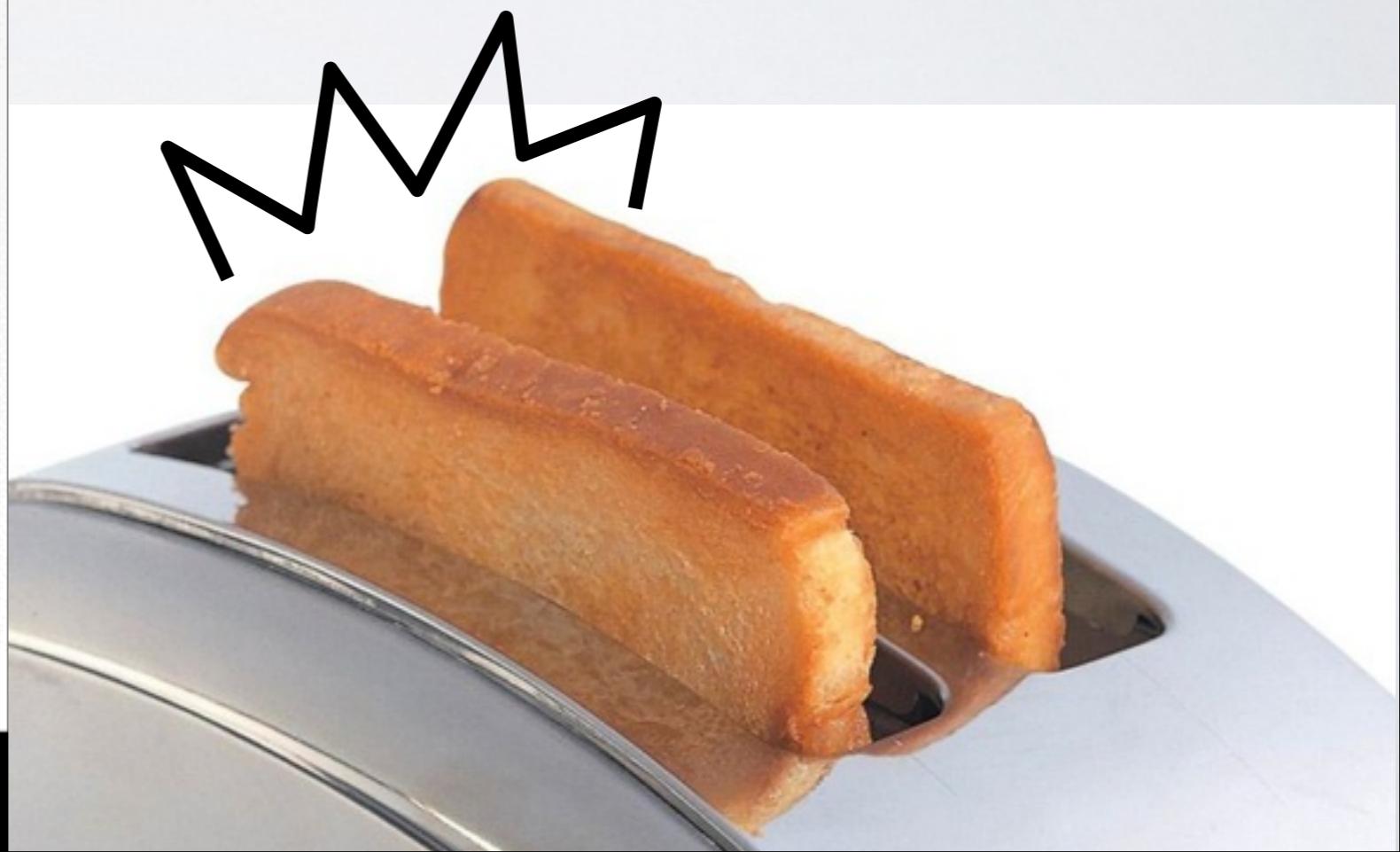
항상 최상위에서 보여지는 토스트 메시지

free for personal use

Toast

알림을 잠깐 동안 띄워주는 메시지 박스
(다이얼로그, Notification의 축소판)

자바스크립트의 alert와 유사



```
Toast toast = Toast.makeText(context, message, Toast.LENGTH_SHORT);  
toast.show();
```

Context

앱의 환경 정보들을
다루는 인터페이스

토스트로 띄워줄 텍스트

(문자열 리소스,
문자열 모두 가능)

토스트를 표시할 시간

Toast.LENGTH_SHORT
Toast.LENGTH_LONG

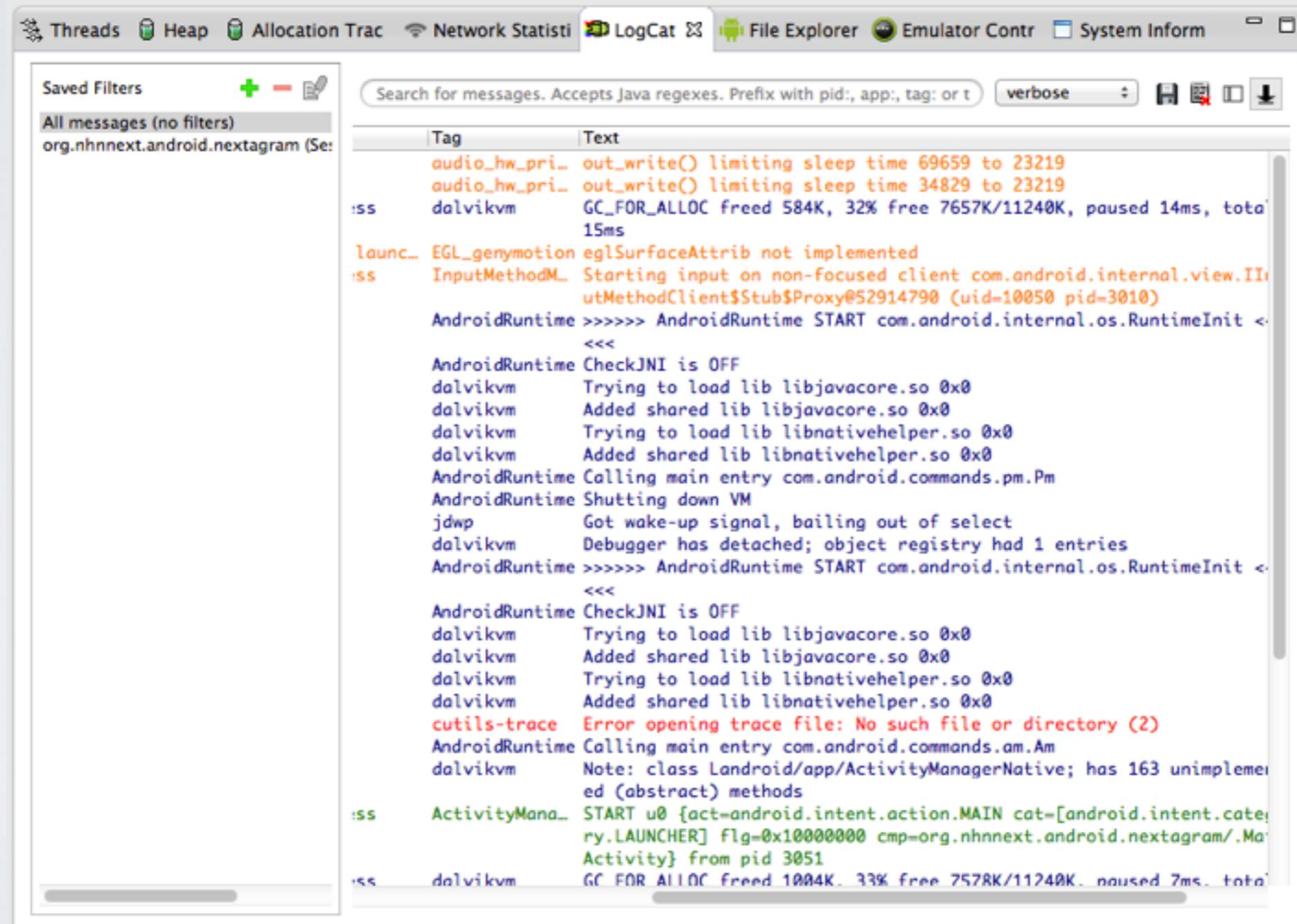
토스트를 화면에 띄워주기!

show()

Log

콘솔 창에 출력할 수 없는 안드로이드 환경에서 개발자 임의대로 로그를 출력하고 그러한 로그를 잡아 보여주는 기능

자바스크립트의 console.log와 유사



<http://developer.android.com/reference/android/util/Log.html>

로그캣

```
Log.i("test", "save Comp");
```

로그 종류

태그

출력할 메시지

- Log.e() - 오류 기록
- Log.w() - 경고 기록
- Log.i() - 메시지 기록
- Log.d() - 디버그 기록
- Log.v() - 상세 메시지 기록

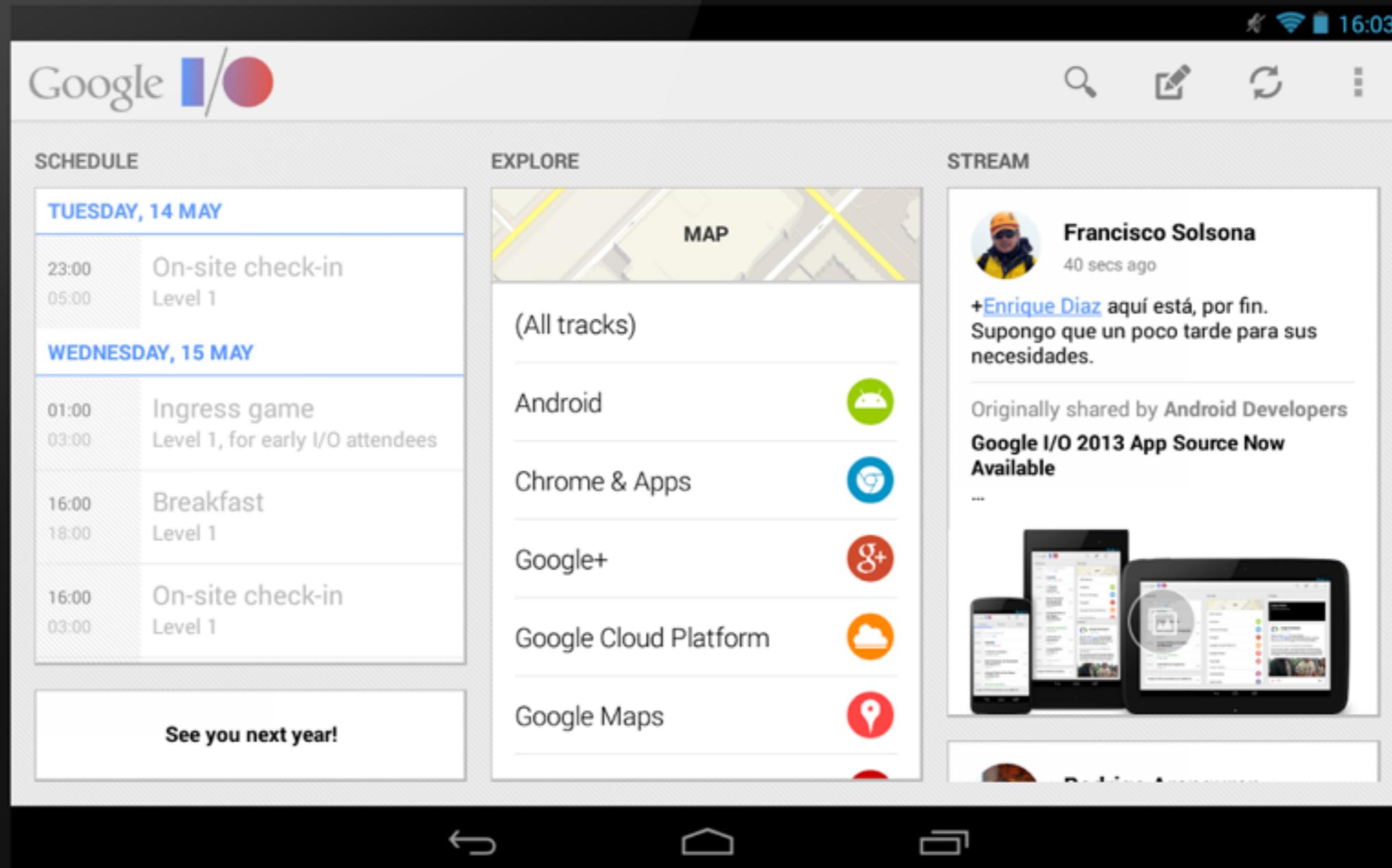
Lev	Time	PID	TID	Application	Tag	Text
E	12-07 12:03:02.405	1608	1608		test	LOG E TEST
W	12-07 12:03:02.405	1608	1608		test	LOG W TEST
I	12-07 12:03:02.405	1608	1608		test	LOG I TEST
D	12-07 12:03:02.405	1608	1608		test	LOG D TEST
V	12-07 12:03:02.405	1608	1608		test	LOG V TEST

로그 종류에 따라
로그의 색이 달라집니다.

토스트(Toast)
사용자를 위한 메시지 기능

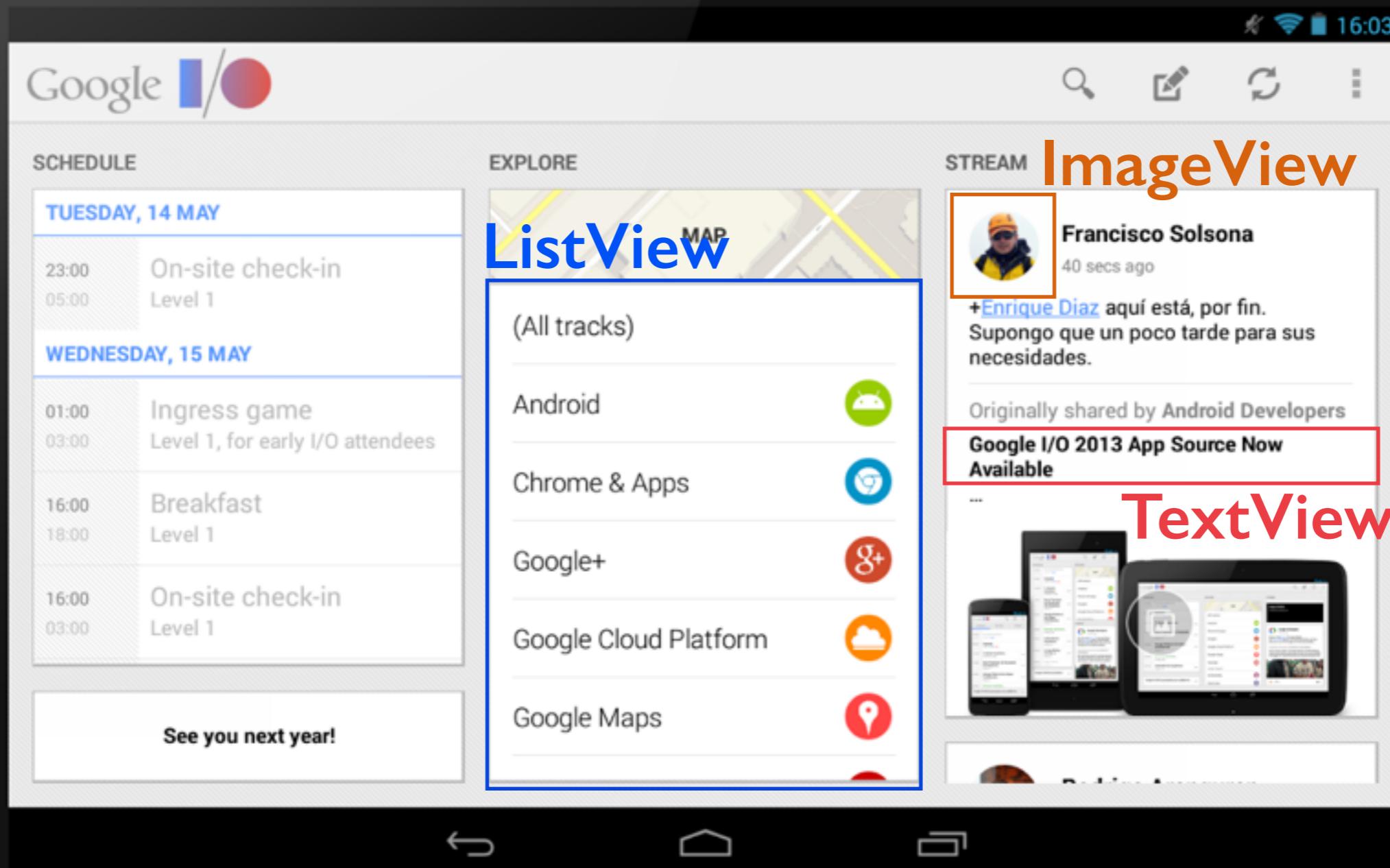
로그(Log)
개발자를 위한 메시지 기능

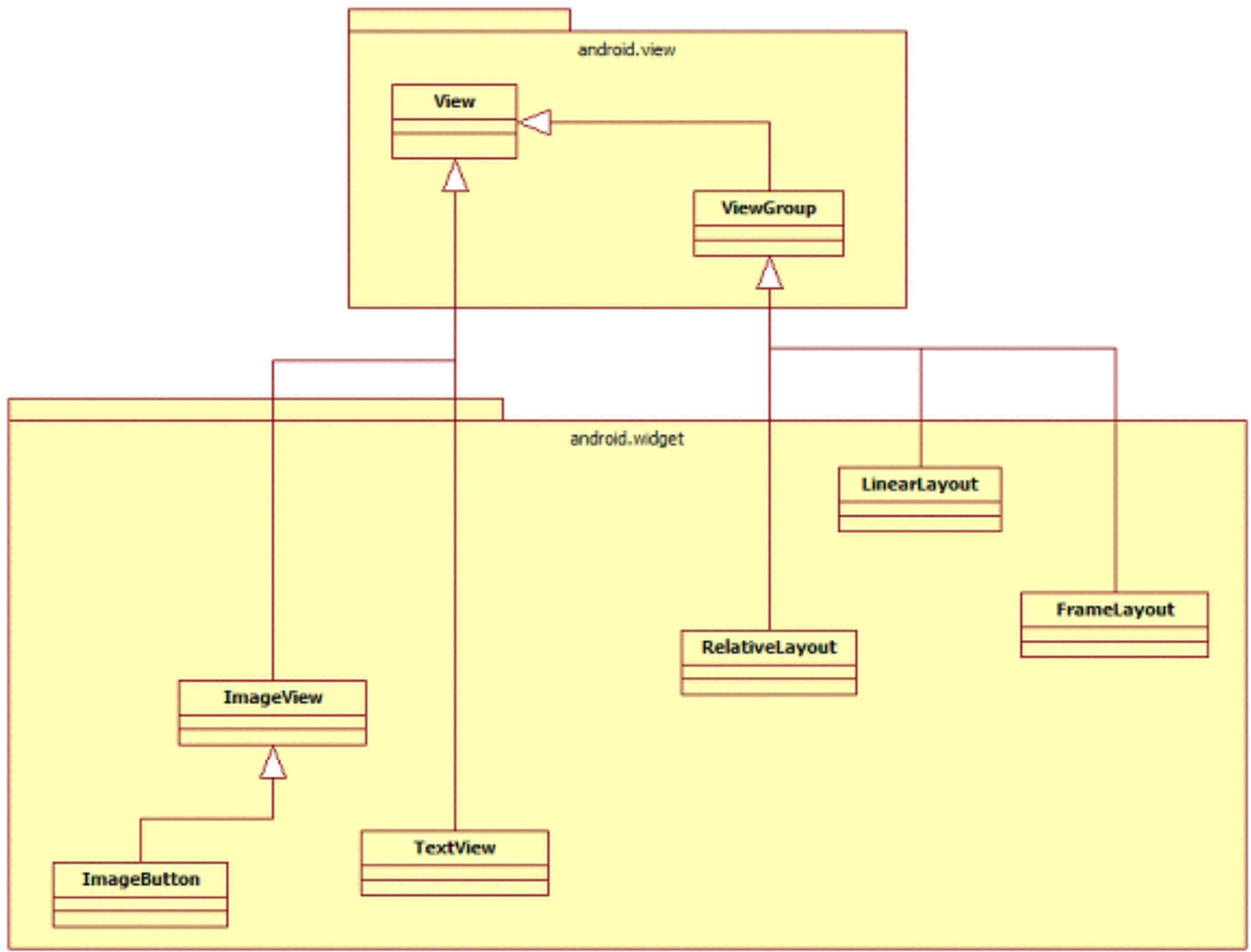
위젯이란?



위젯

사용자와 상호작용하는 요소





뷰(View)

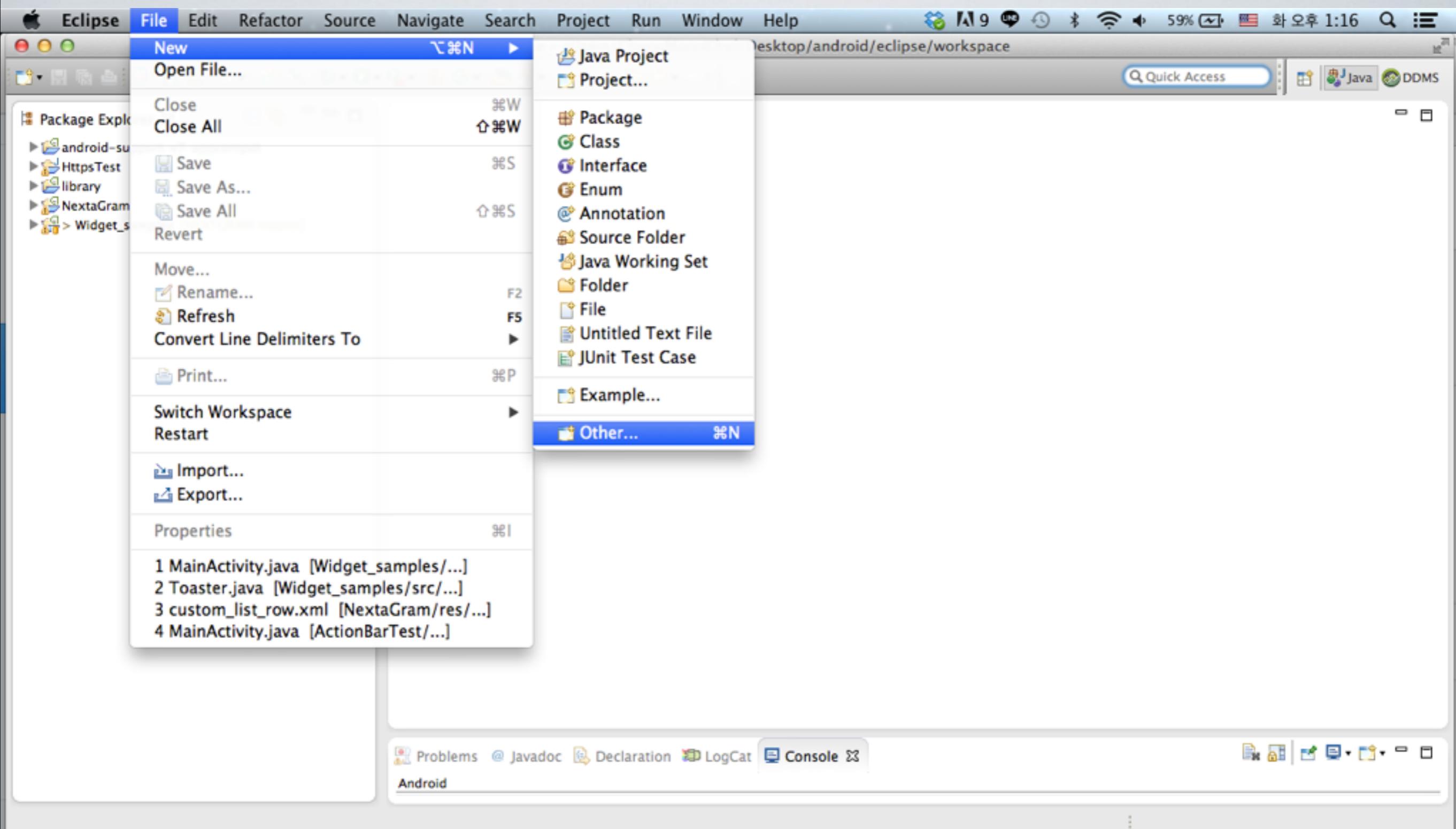
위젯 등 화면을 구성하는
요소인 객체들을 자식으로 갖는
다.

자세한 내용은 아래 링크 참조

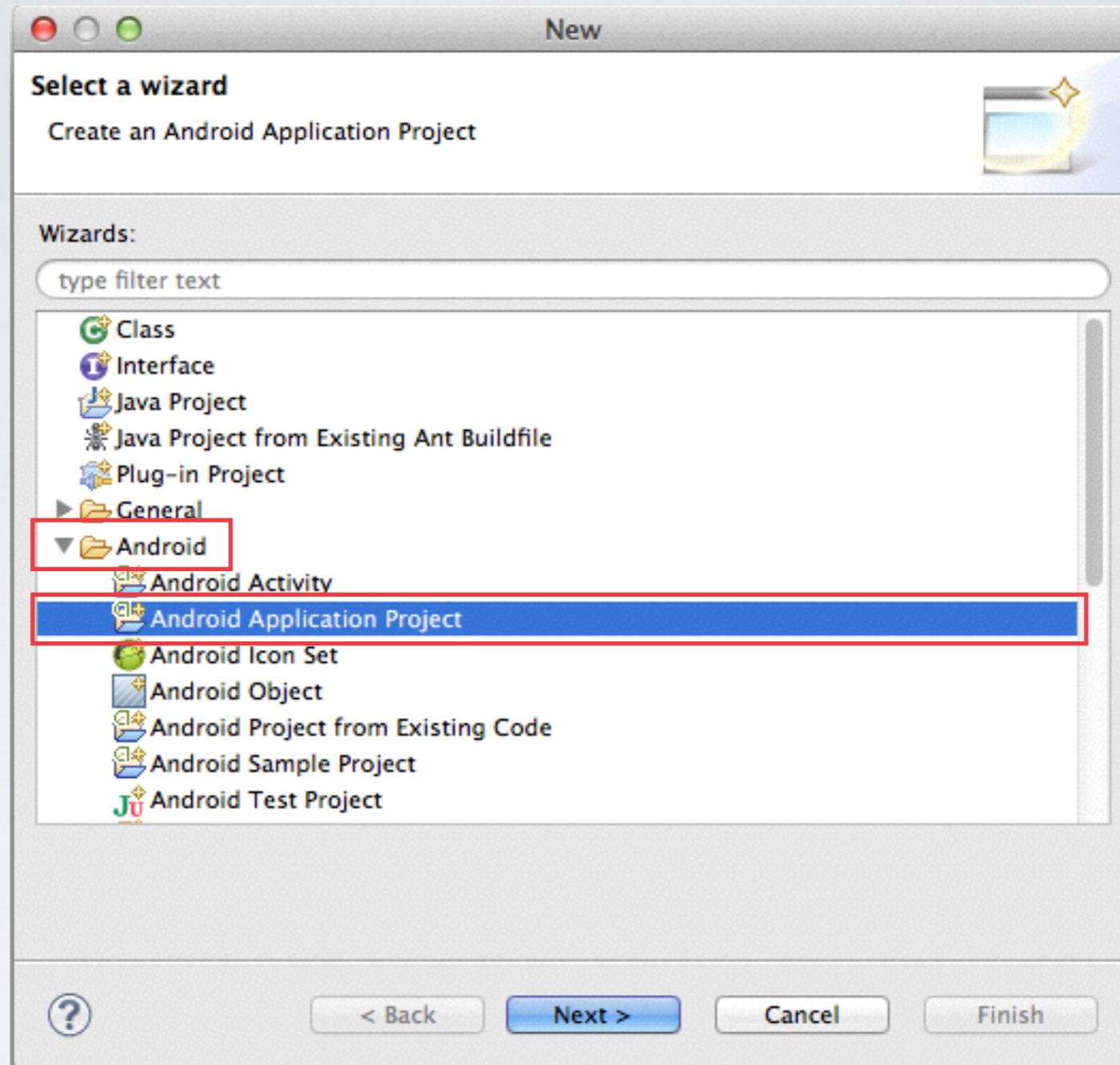
<http://croute.me/391>
<http://powerhan.tistory.com/107>

위젯 (Widget)

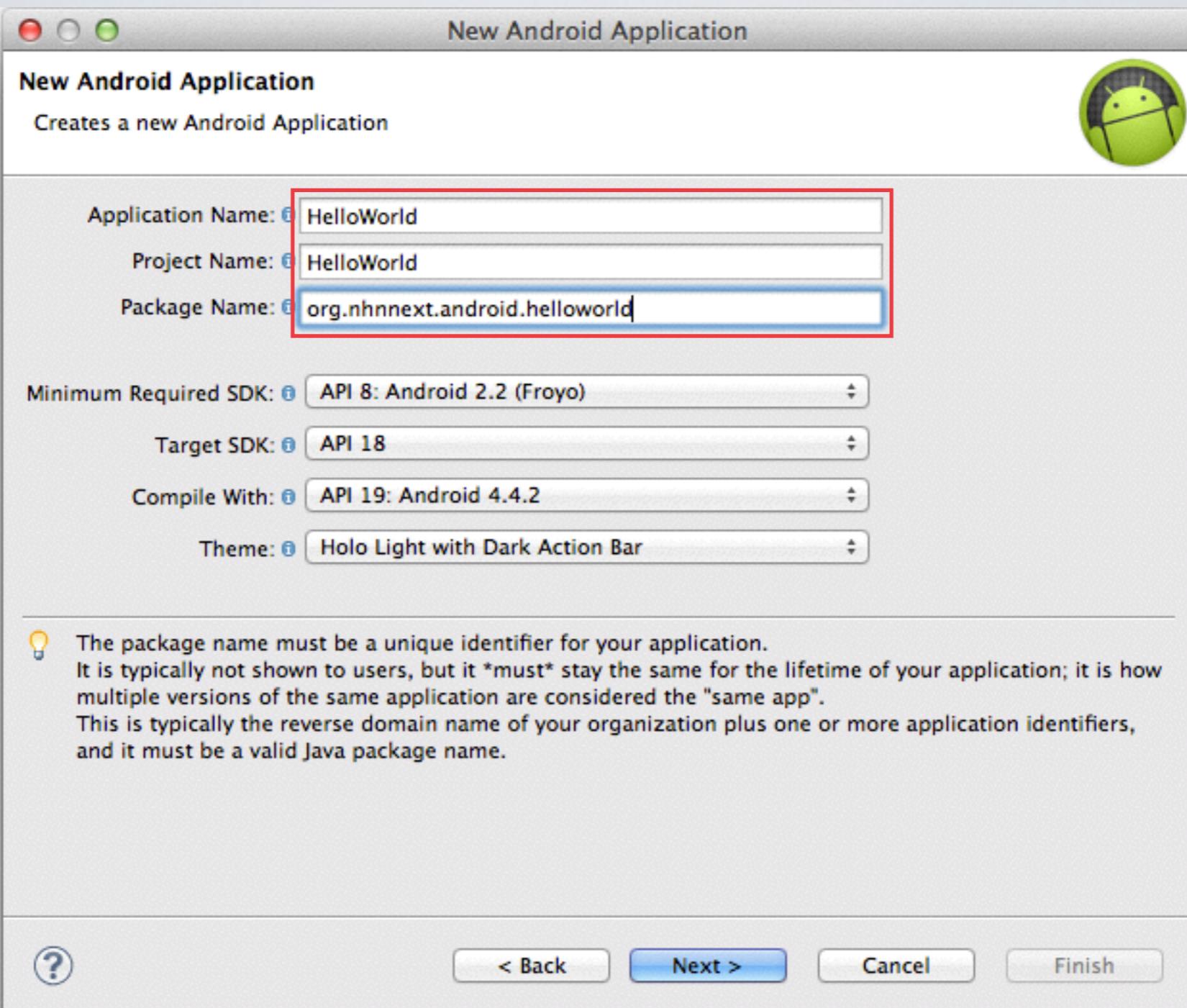
이제 새 안드로이드 프로젝트를 만들어 봅시다.



File-New-Other로 들어갑니다.

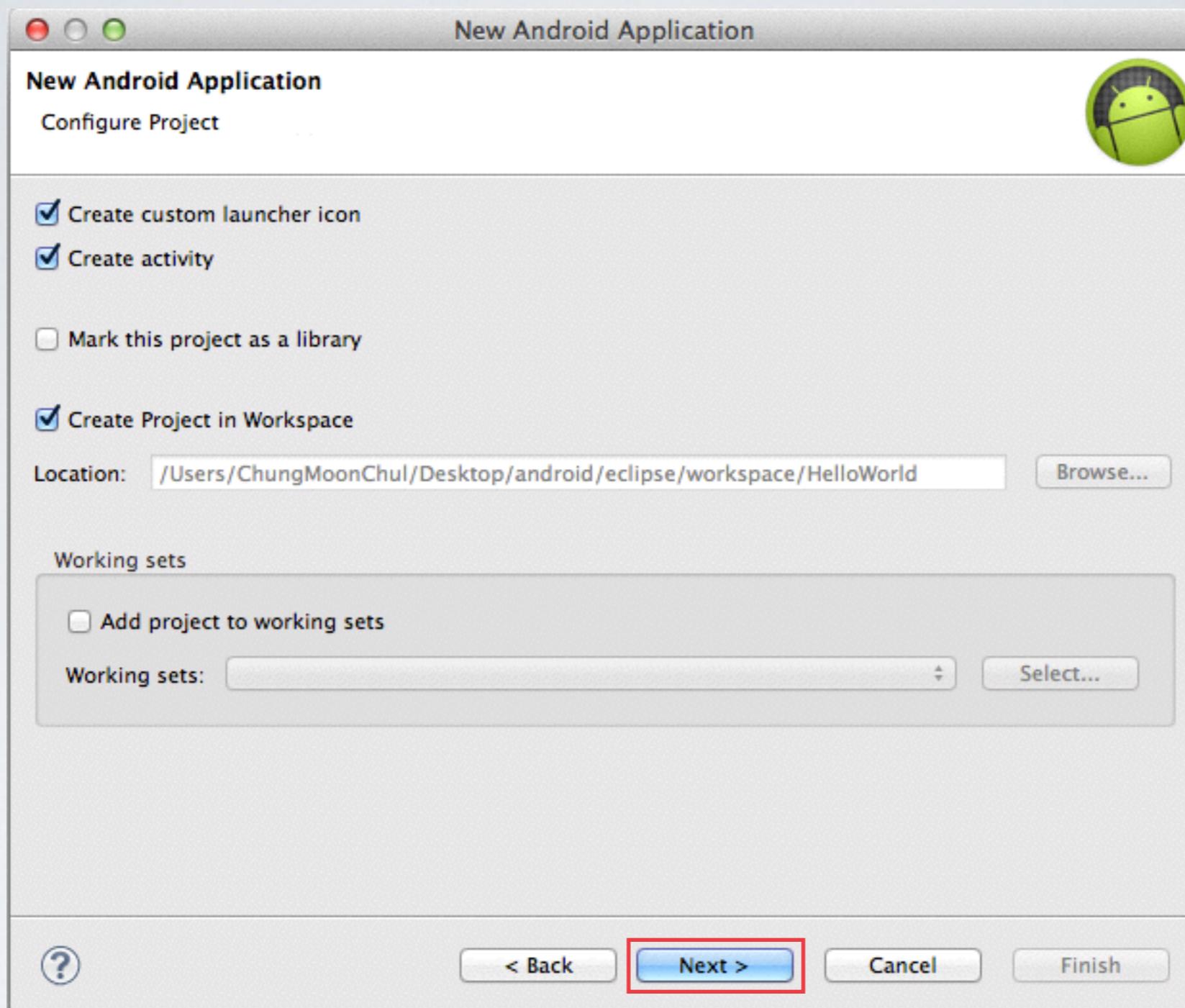


Android항목에서
Android Application Project를 선택합니다.

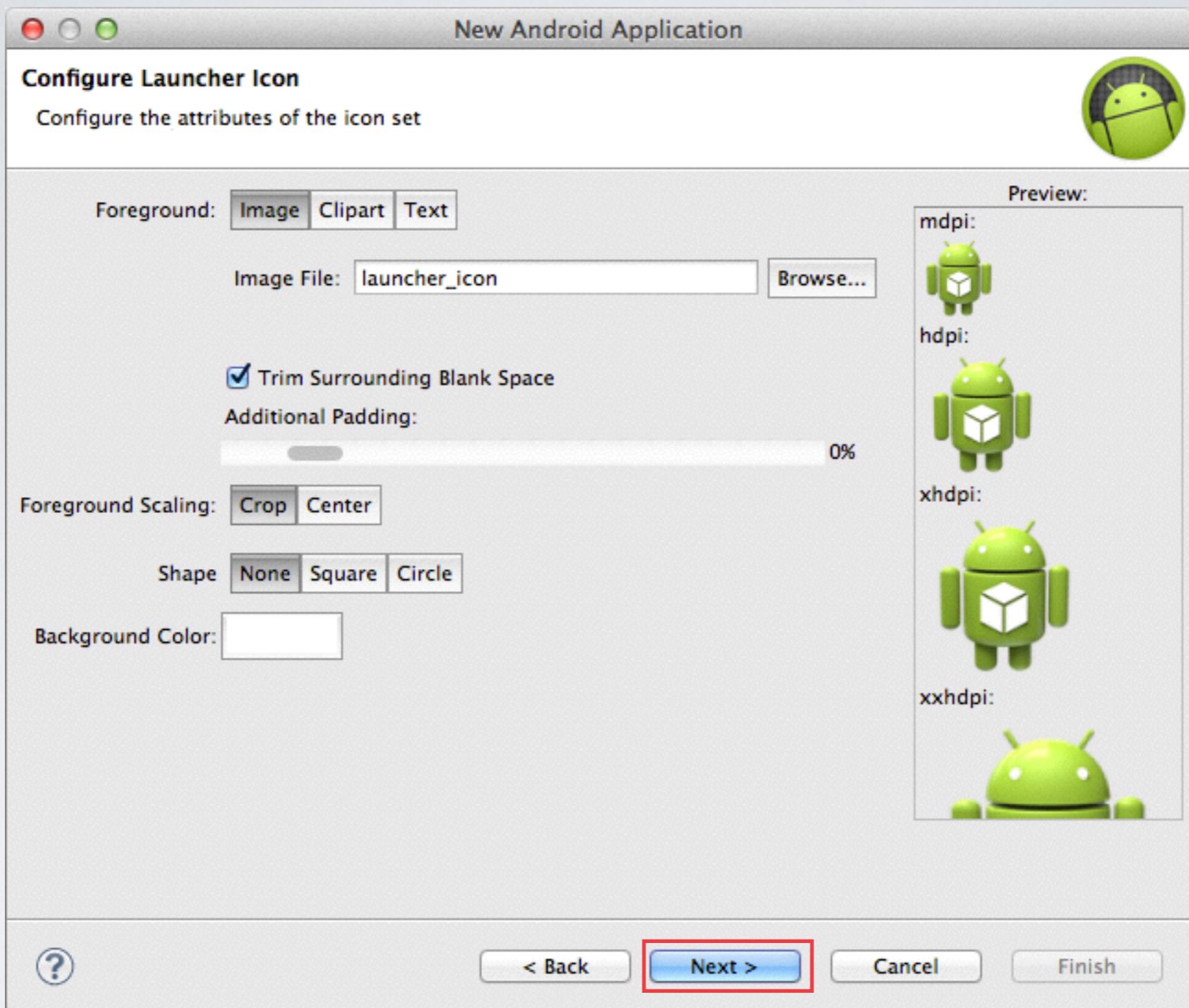


앱의 이름, 프로젝트 이름,
패키지 이름을 적어줍니다.

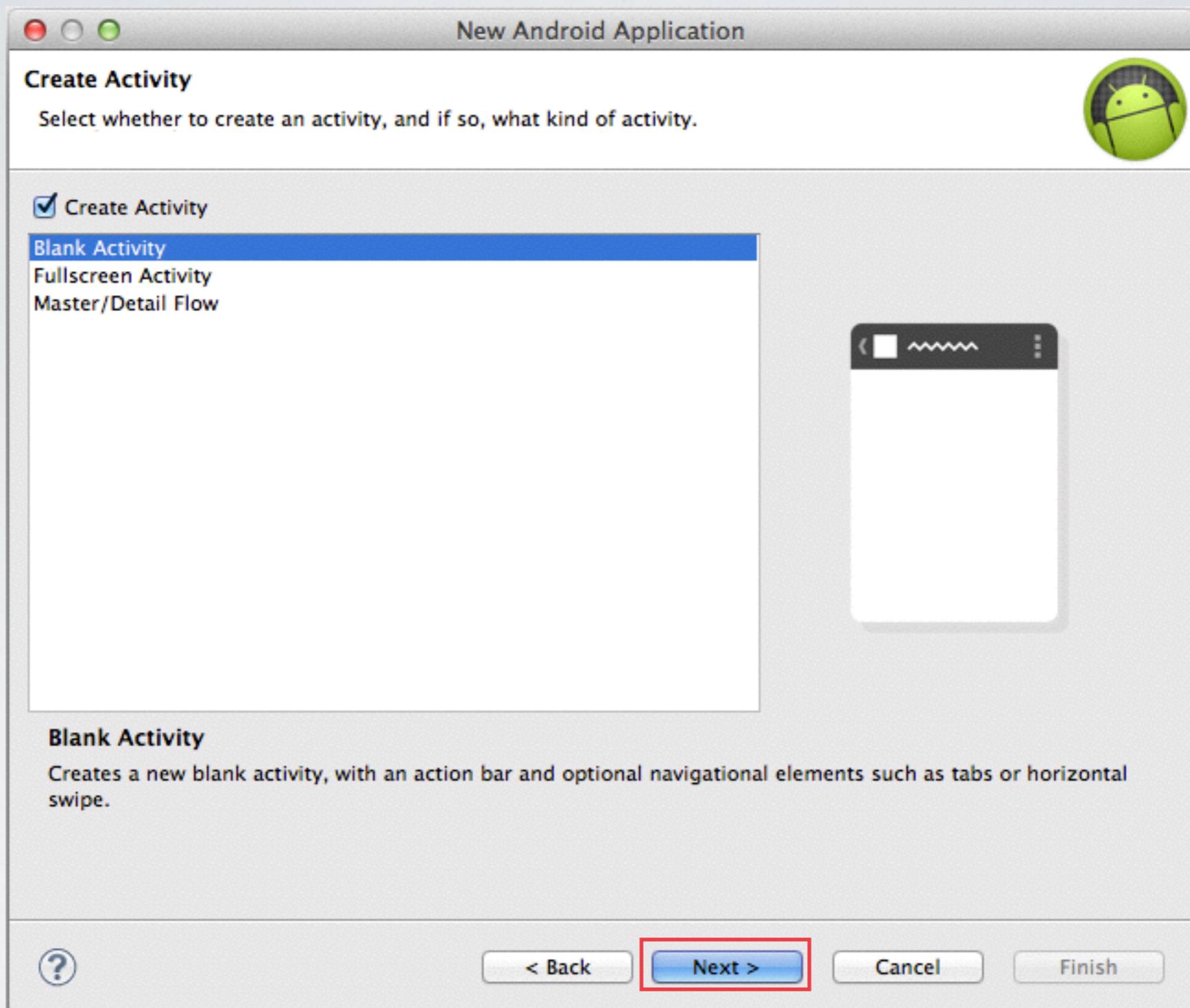
보통 앱, 프로젝트 이름은 첫글자가 대문자로 시작하고
패키지 이름은 중복되지 않는 주소 형태를 따릅니다.



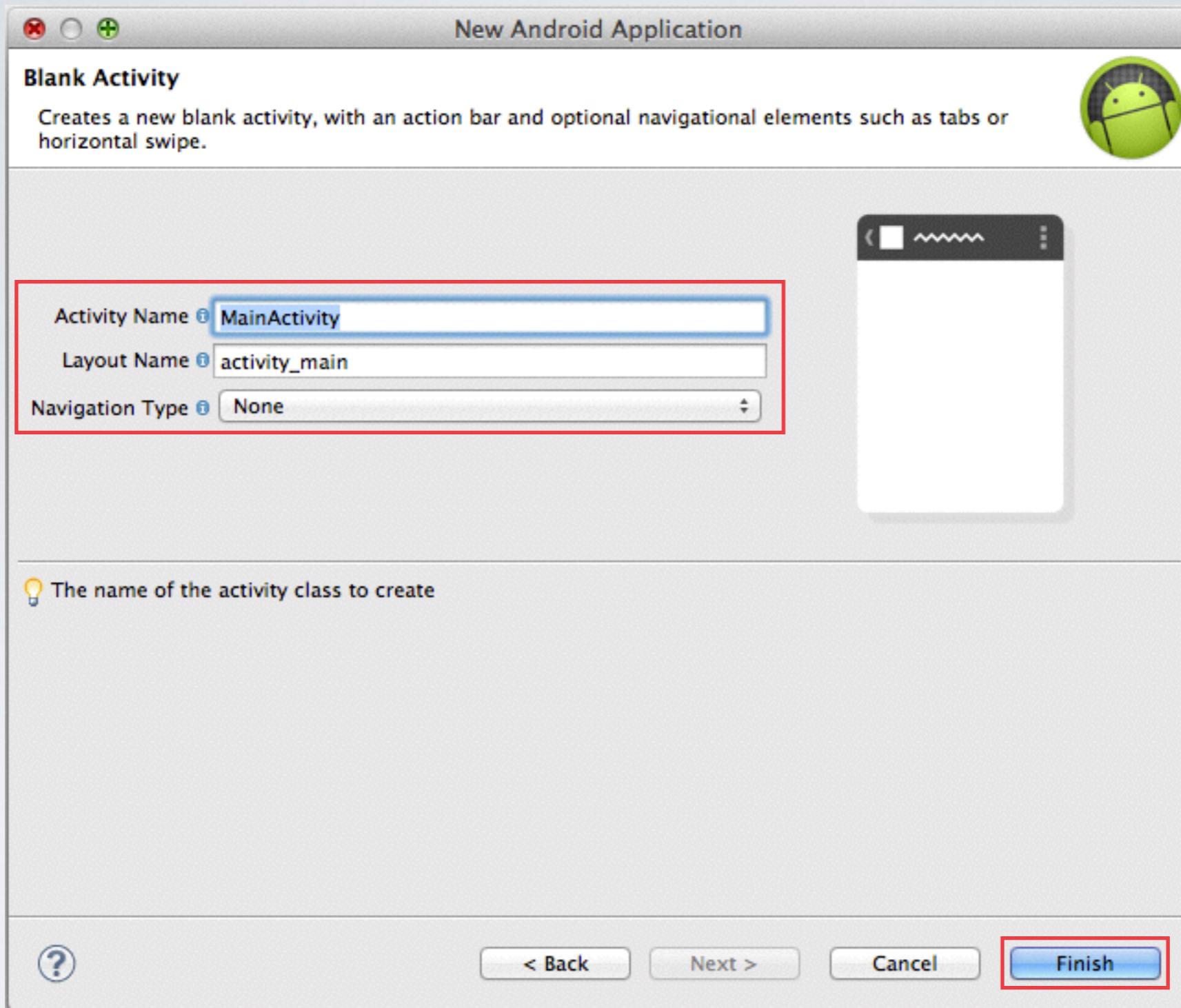
프로젝트 설정부분이지만
그냥 Next를 눌러줍니다...



기본 아이콘 설정 화면
그냥 넘어가셔도 무방합니다...



액티비티 종류를 선택합니다.
여기서는 Blank Activity를 사용합니다.



액티비티의 이름등의 설정을 하는 화면입니다.
역시 기본값을 사용하겠습니다...

Java - HelloWorld/src/org/nhnnext/android/helloworld/MainActivity.java – Eclipse – /Users/ChungMoonChul/Desktop/android/eclipse/workspace

Package Explorer

activity_main.xml MainActivity.java

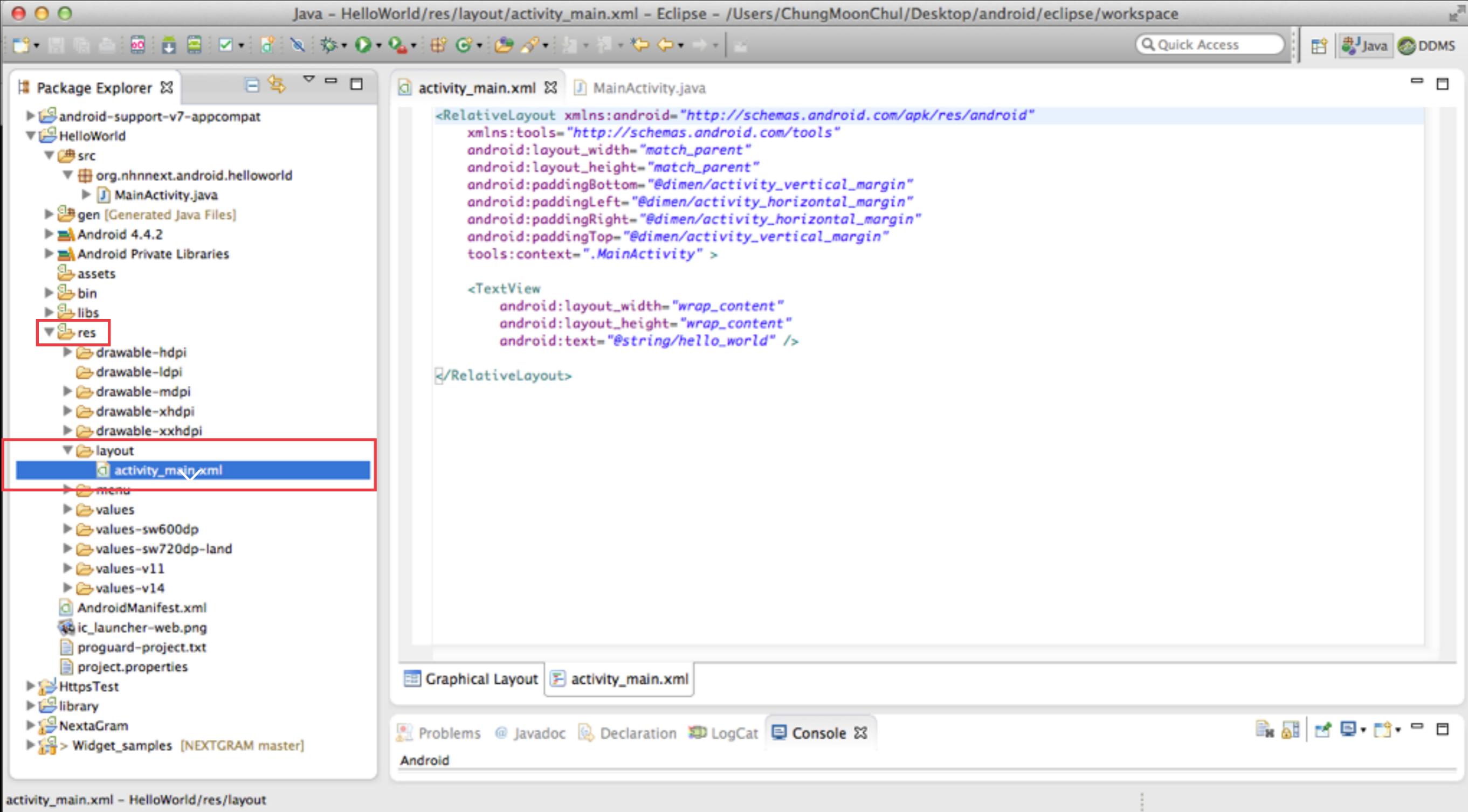
```
package org.nhnnext.android.helloworld;  
import android.os.Bundle;  
public class MainActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        // Inflate the menu; this adds items to the action bar if it is present.  
        getMenuInflater().inflate(R.menu.main, menu);  
        return true;  
    }  
}
```

Problems Javadoc Declaration LogCat Console

Writable Smart Insert 10 : 57

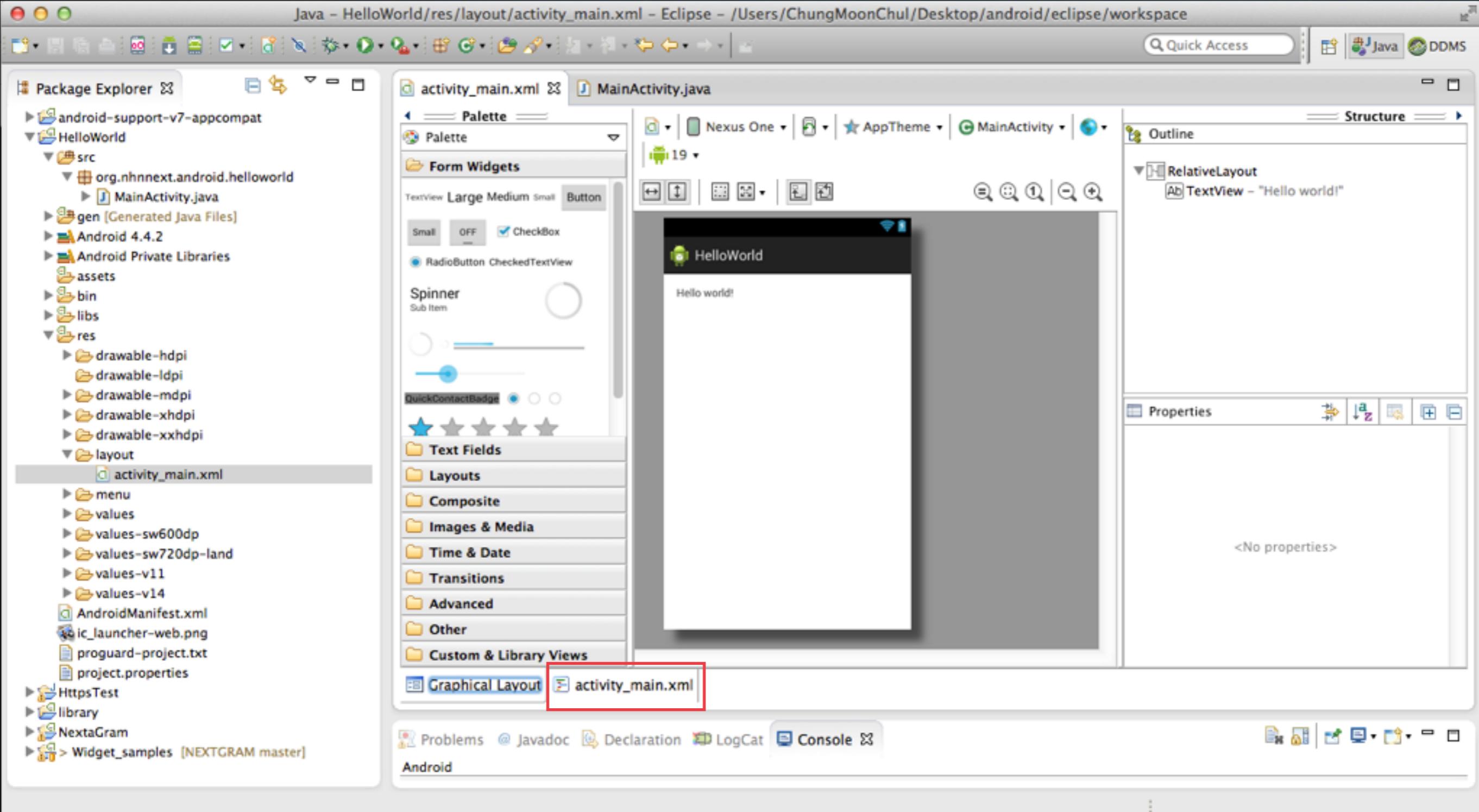
새 프로젝트가 생성되었습니다.
왼쪽에는 프로젝트 구조가 오른쪽에는 코드가 보입니다.

개발툴 버전에 따라 보이는 화면이 다를수 있습니다...



위젯에 대하여 알아보기 위해
레이아웃 파일을 수정해보겠습니다.

프로젝트의 res/layout/activity_main.xml 파일을 열어줍니다.



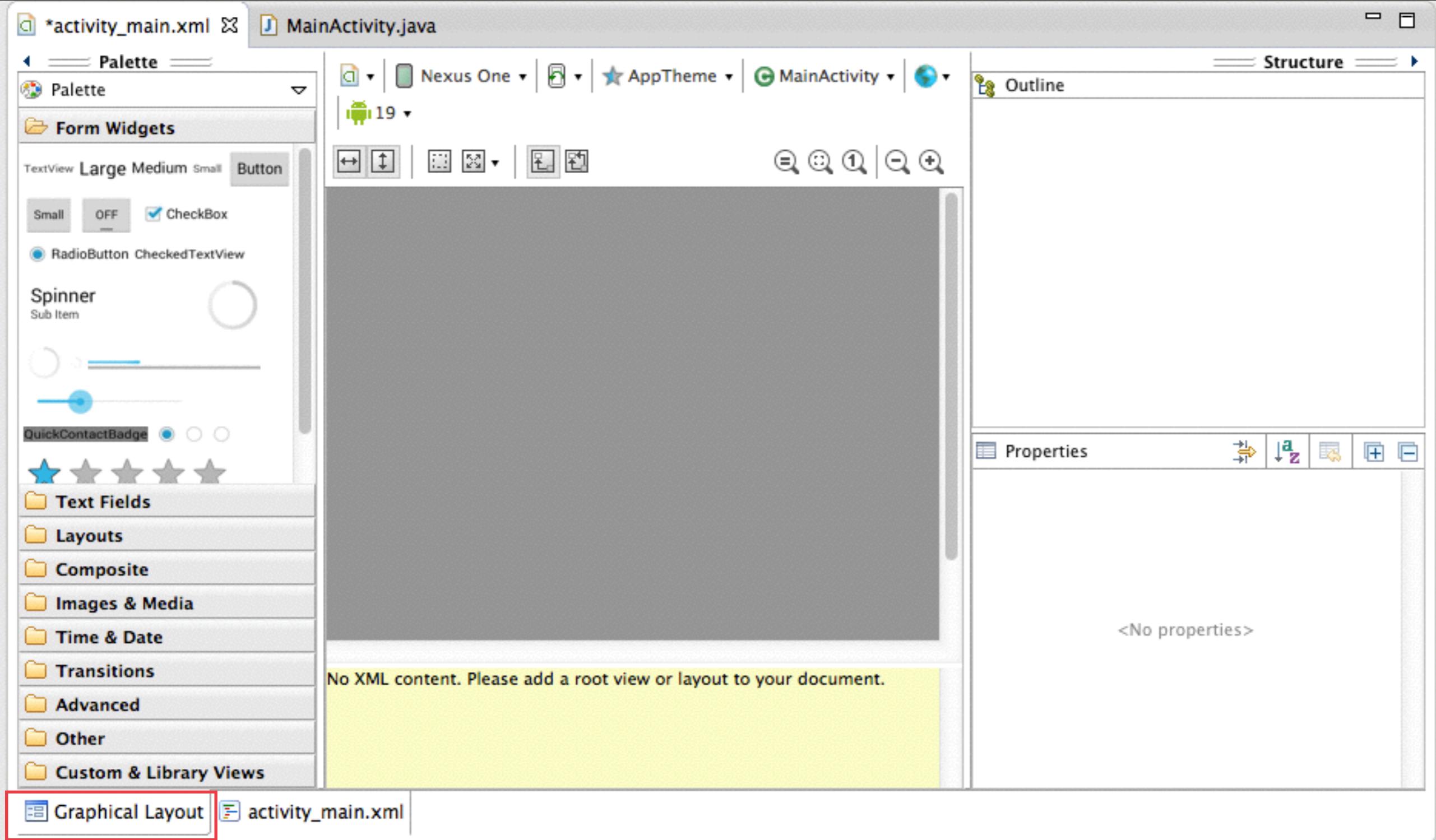
혹시 텍스트 에디터가 아니라
이런 레이아웃 에디터면 파일명.xml 탭을 눌러주세요

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context=".MainActivity" >  
  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/hello_world" />  
  
</RelativeLayout>
```

Graphical Layout

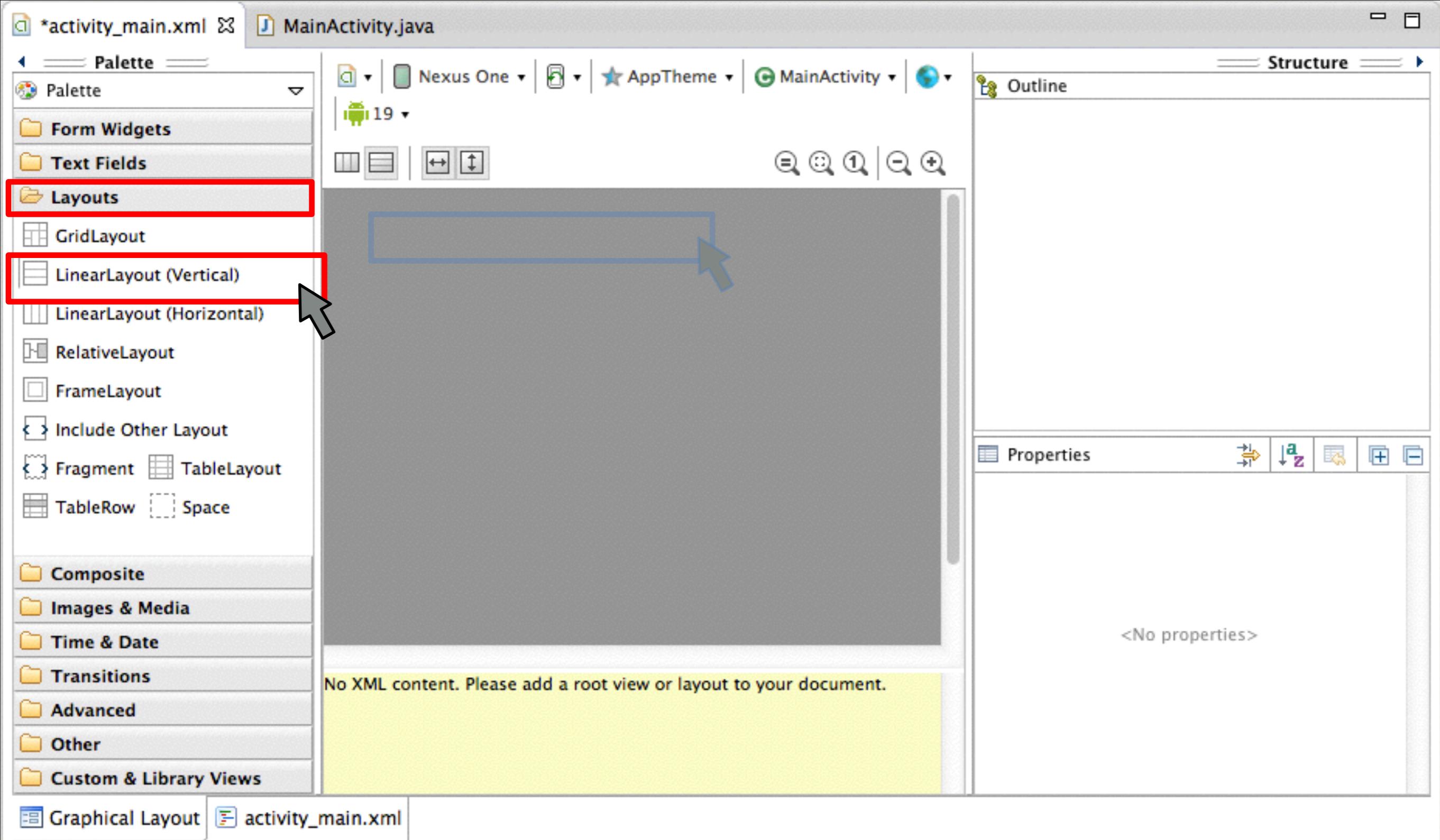
activity_main.xml

먼저 기존에 있던 레이아웃을 삭제를 하겠습니다.

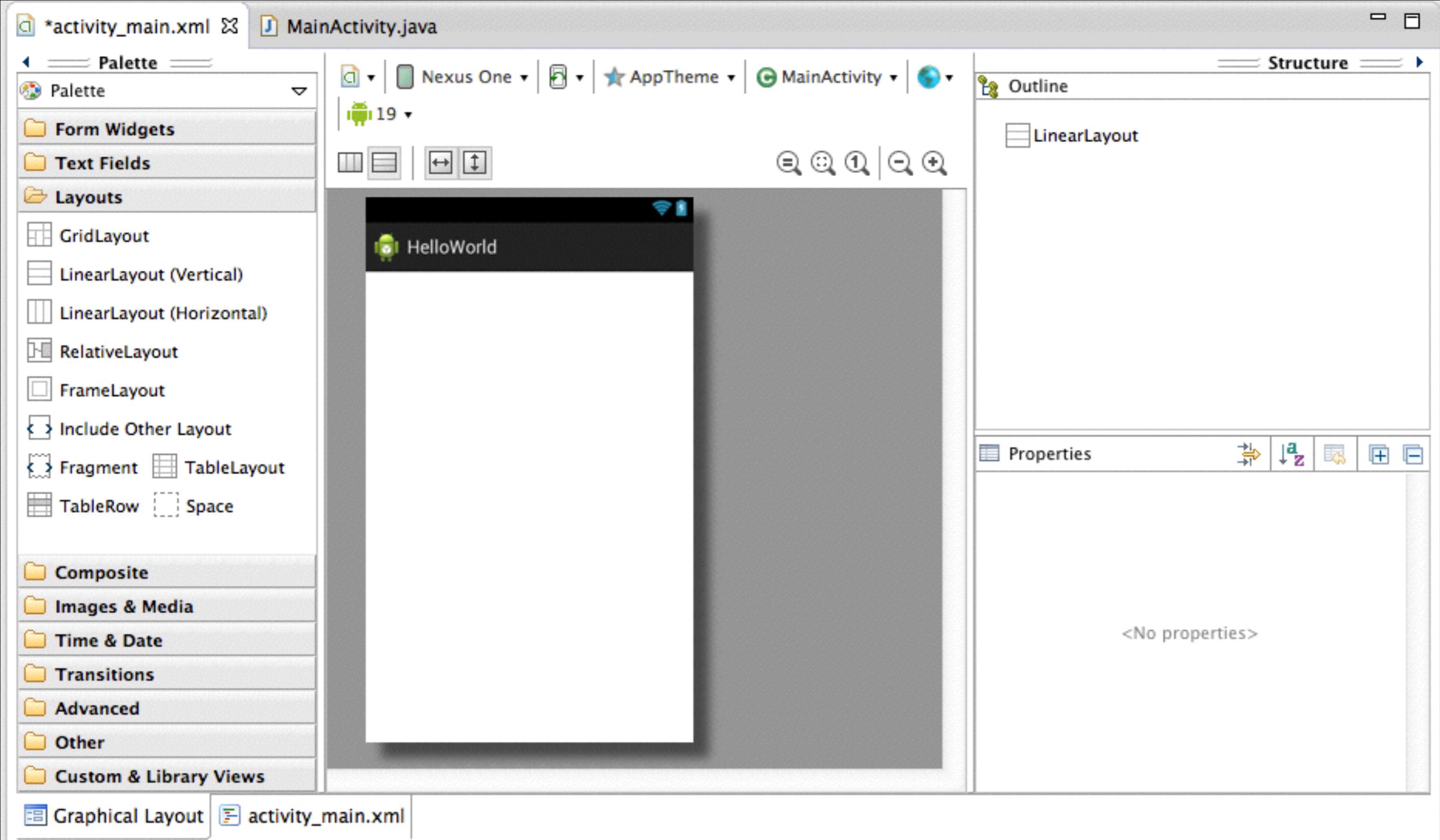


안드로이드 레이아웃을 수정하기 위해서는
xml파일을 직접 수정하거나 레이아웃 에디터를 사용할 수 있습니다.

이번에는 Graphical Layout 탭을 눌러 에디터를 사용해 보겠습니다.



가장 기본적인 레이아웃인 LinearLayout을 사용해 보겠습니다.
Layouts 탭에서 LinearLayout을 드래그해서 에디터에 삽입합니다.



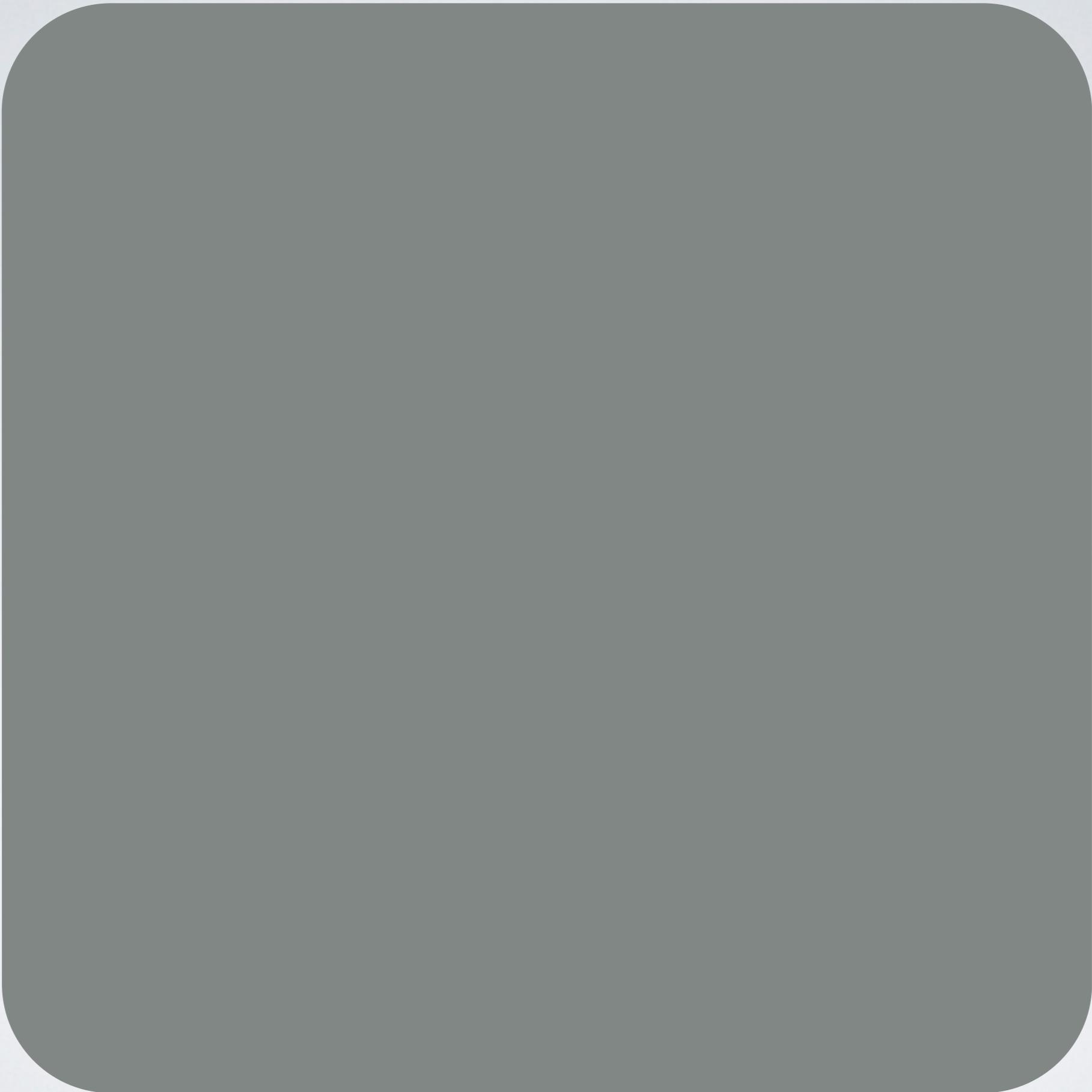
LinearLayout이 자동으로 생성되었습니다.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:orientation="vertical" >  
  
</LinearLayout>
```

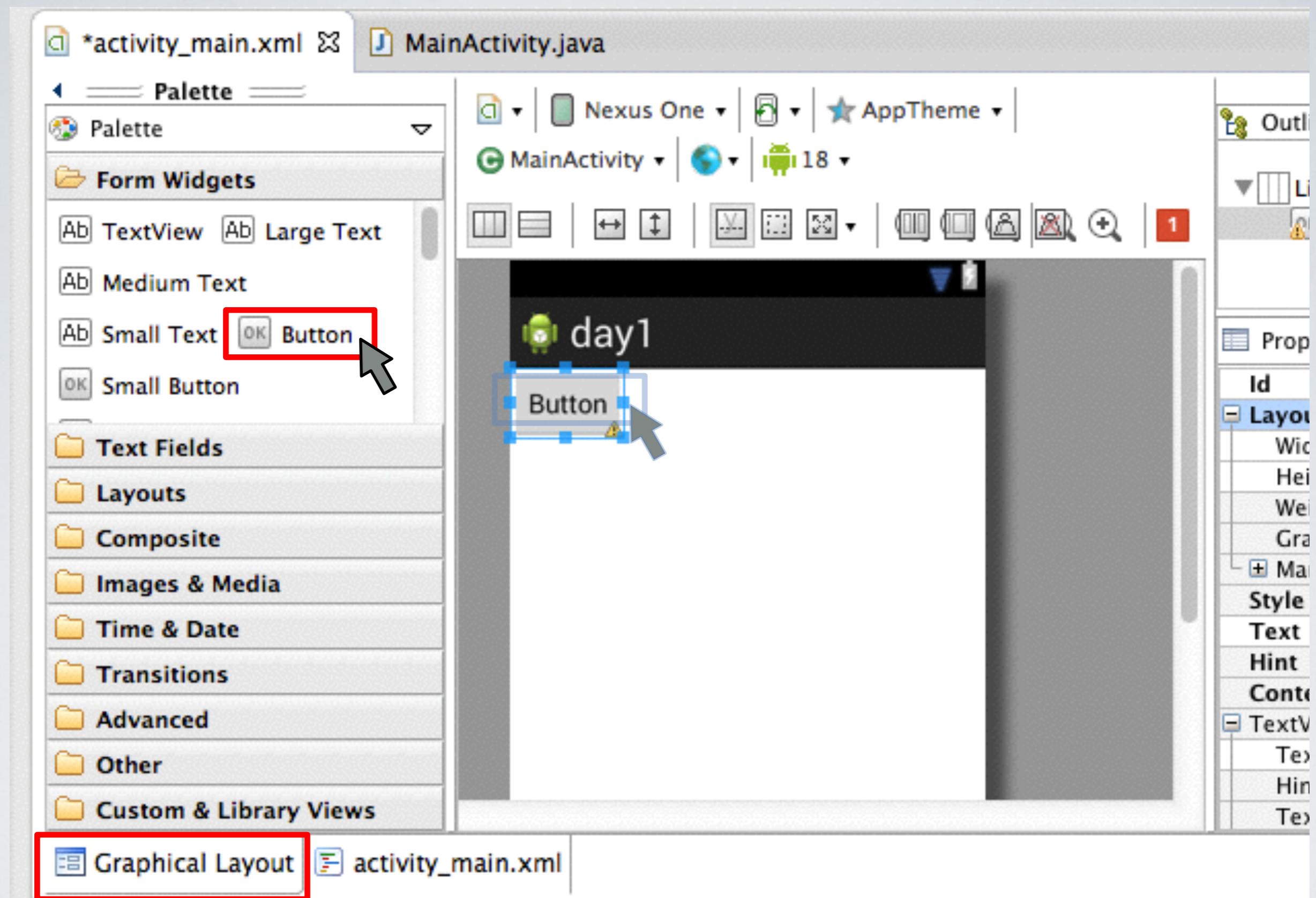
Graphical Layout

activity_main.xml

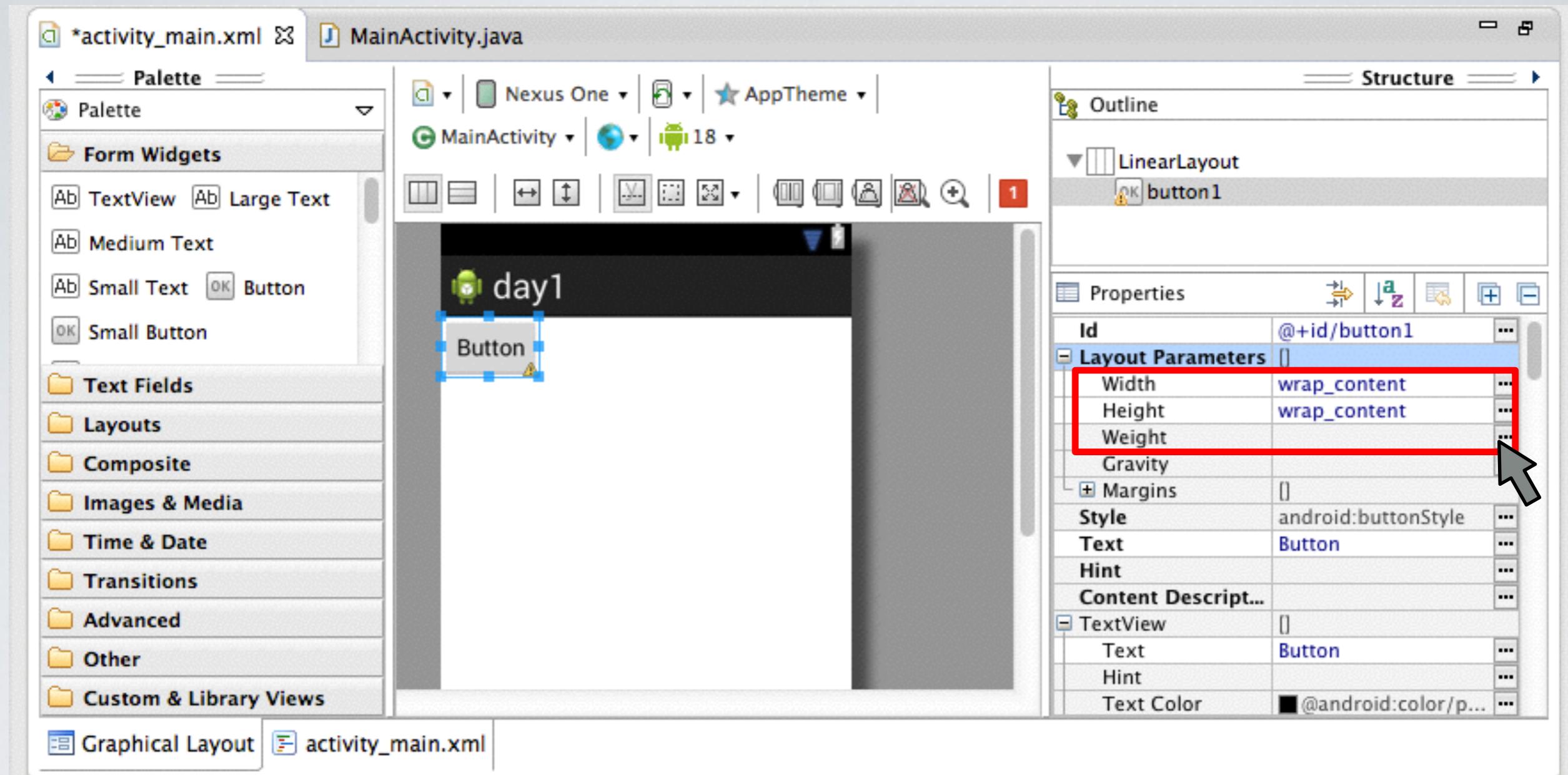
xml파일을 보면 에디터가 다음과 같은 내용을
자동으로 만들어 준 것을 확인할 수 있습니다.



위젯의 크기를 설정하는 방법을 알아보겠습니다.

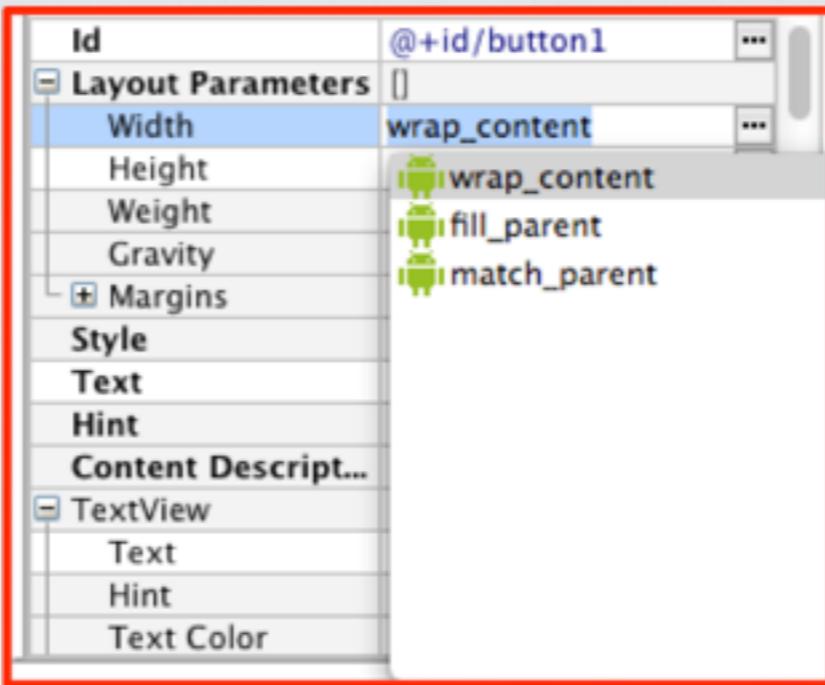


연습을 위해 레이아웃 에디터에서
버튼을 하나 생성을 해보겠습니다.



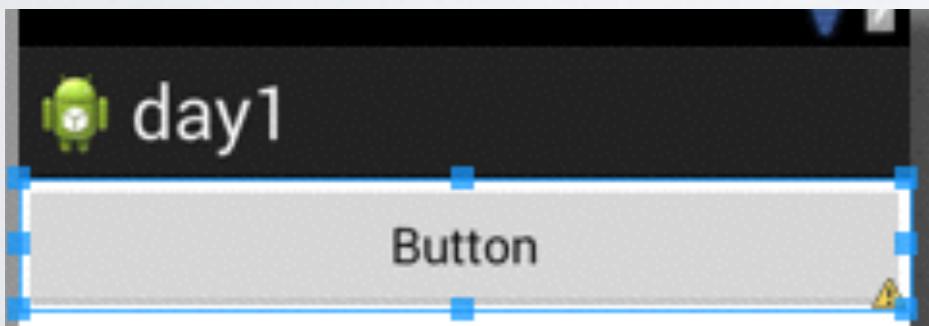
Properties에서 위젯을 설정 할 수 있습니다.

크기를 변화시키기 위해 LayoutParameters의 Width와 Height를
중점적으로 살펴보겠습니다.



컨텐츠의 크기로 (문자열 “Button”)
내용물을 감싸듯이 자동으로 크기를 조정합니다.

Width가 “`wrap_content`” 일 때



부모의 크기만큼(커질수 있을 최대한)
자동으로 크기를 조정합니다.

Width가 “`match_parent`(= `fill_parent`)” 일 때

```
<TextView  
    android:id="@+id/textView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignLeft="@+id/textView2"  
    android:layout_below="@+id/textView2"  
    android:layout_marginLeft="57dp"  
    android:layout_marginTop="25dp"  
    android:text="TextView" />
```

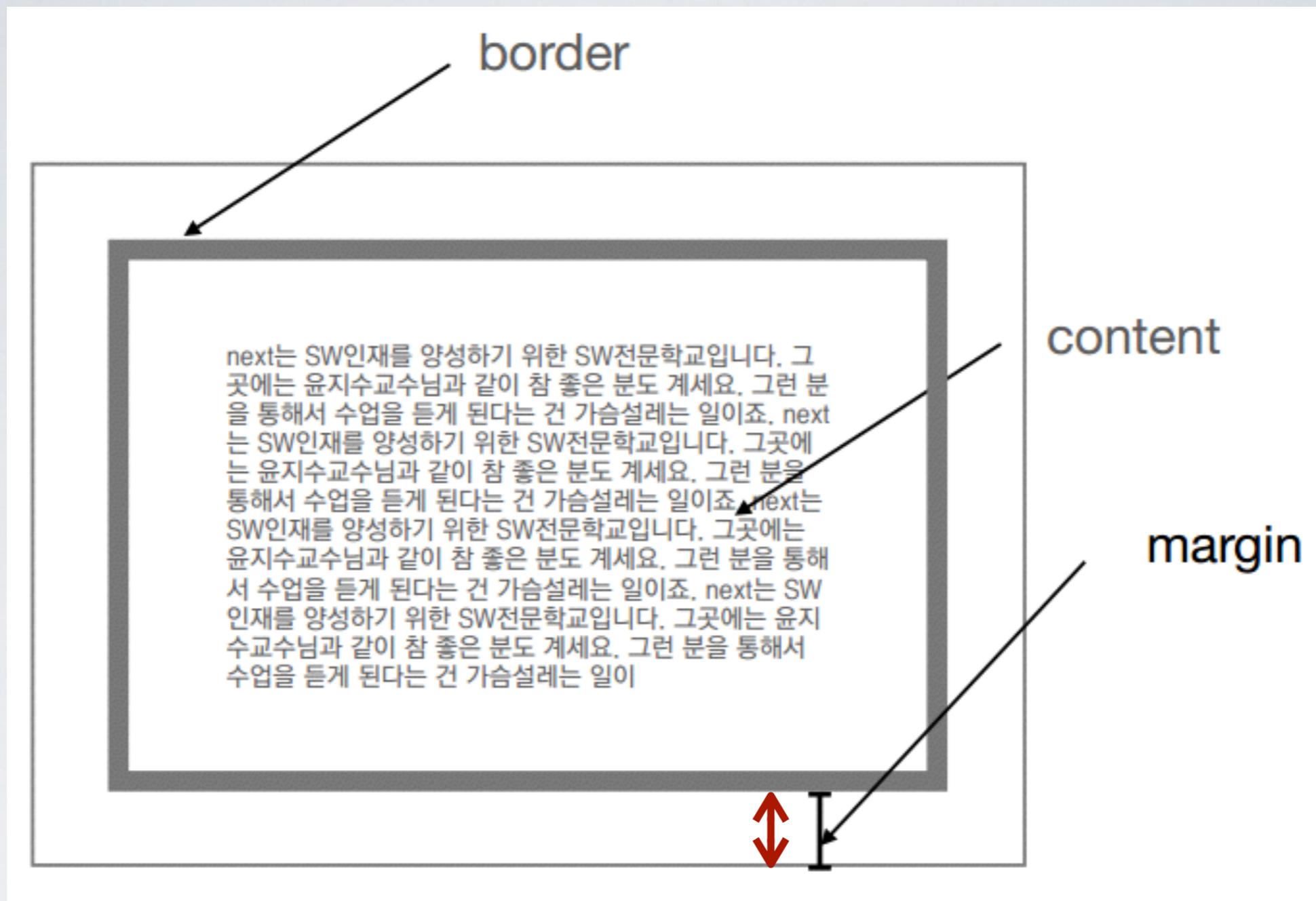
xml편집으로도 크기를 변화시킬수 있습니다.

“wrap_content” - 현재 뷰의 최대 크기로 자동 조절 (내용을 감싸게)

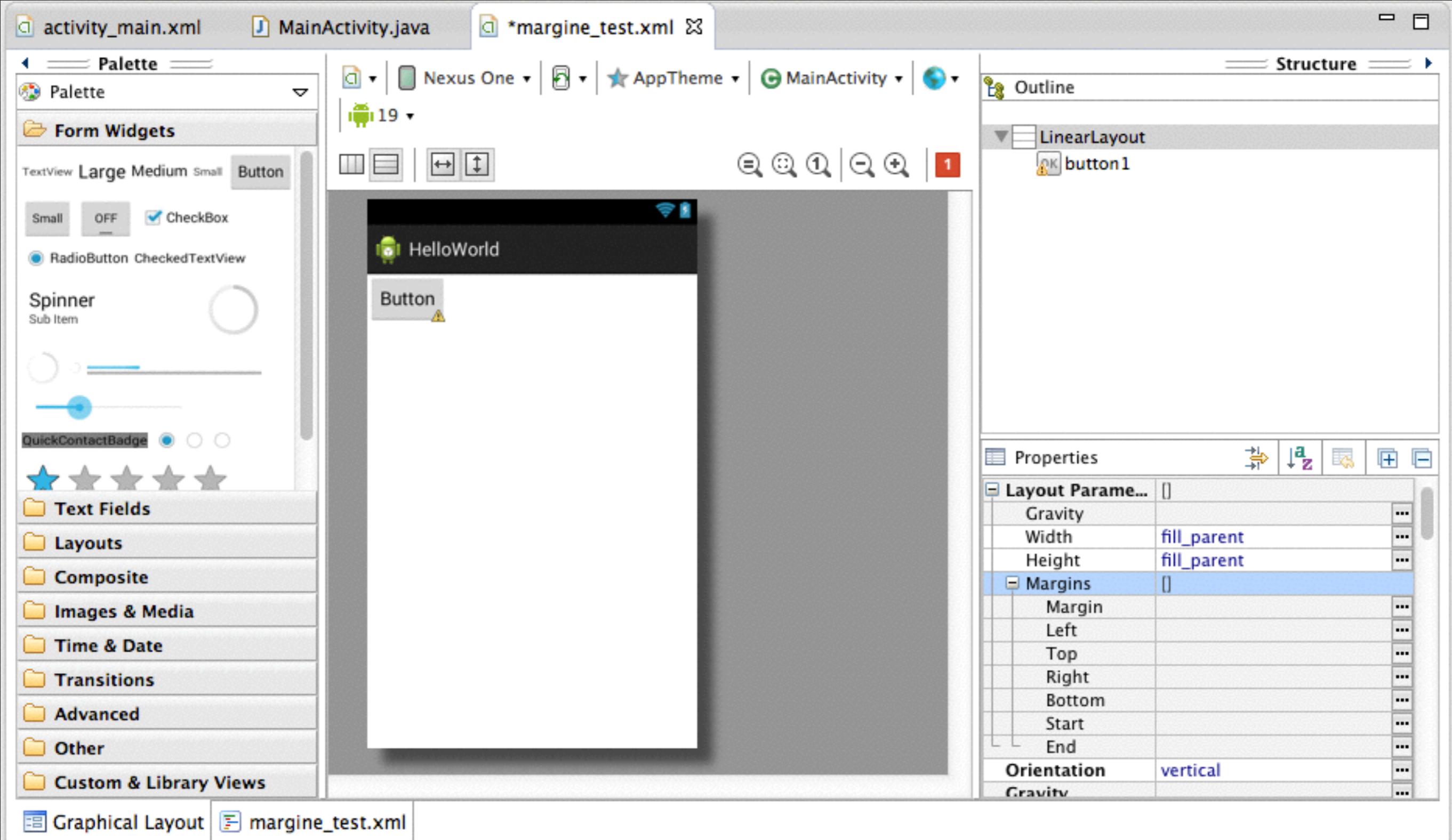
“match_parent” - 부모 뷰까지의 최대 크기로 자동 조절

(fill_parent하고 match_parent는 같은 기능이지만
API 8 이후부터 지원되는 match_parent를 사용하는 것이 권장하고 있습니다.)

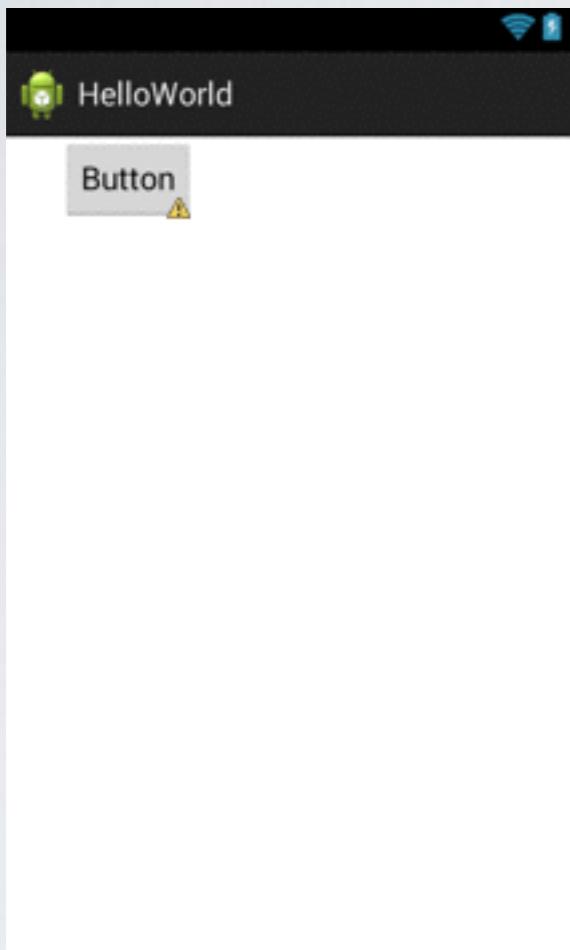
마진 (Margin)



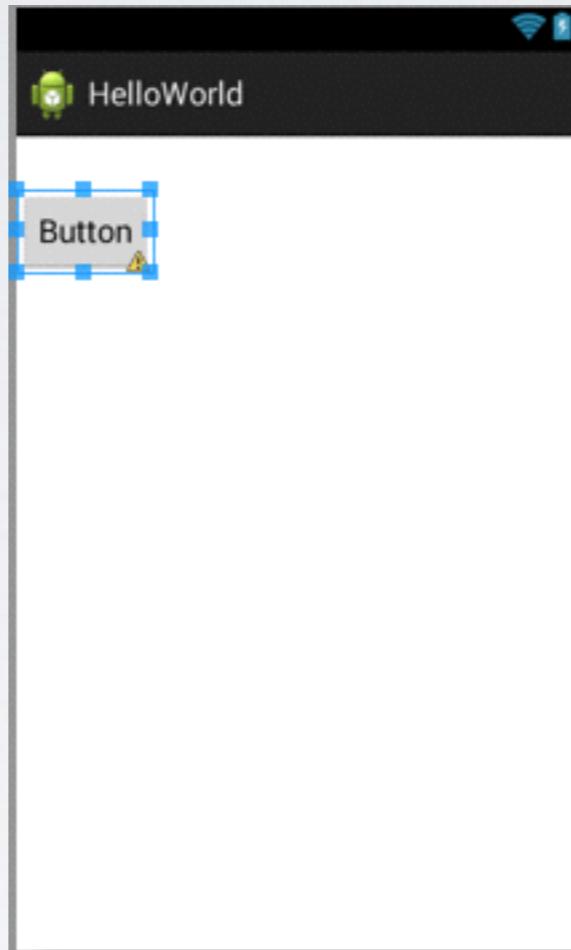
다른 Widget(Element)과의 간격을
마진으로 조정 할 수 있습니다.



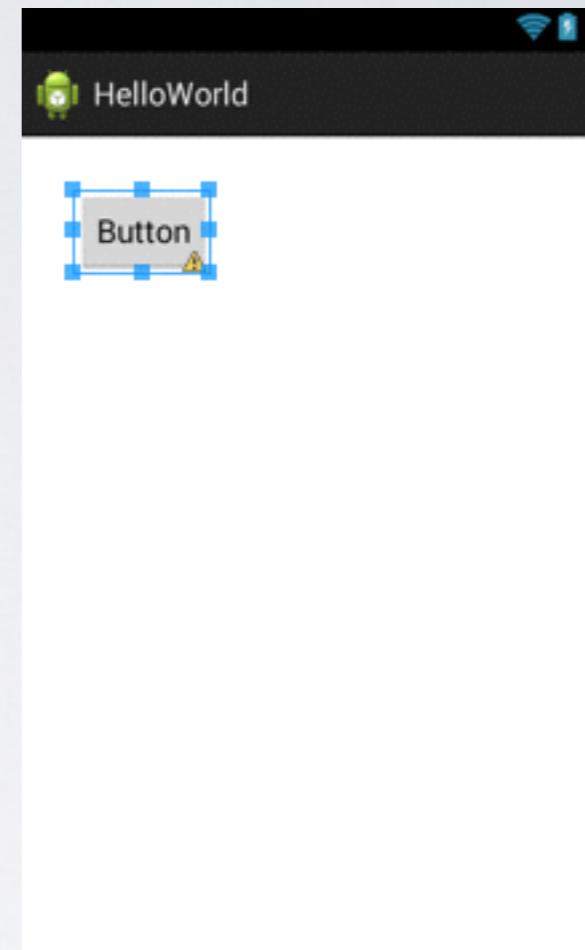
Margin은 Properties의 Layout Parameters의
Margins에서 적용가능합니다.



LeftMargin



TopMargin



Left + Top

```
<TextView  
    android:id="@+id/textView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignLeft="@+id/textView2"  
    android:layout_below="@+id/textView2"  
    android:layout_marginLeft="57dp"  
    android:layout_marginTop="25dp"  
    android:text="TextView" />
```

android:layout_margin
 marginLeft
 marginRight
 marginTop
 marginBottom

xml에서도 직접 수치를 입력하는 것으로
마진을 줄 수 있습니다.

(4방향 모두 마진을 줄 때에는 그냥 layout_margin을 주면 간편하게 됩니다.)

기기 크기에 상관없는
레이아웃 만들기

같은 앱이지만 기기의 크기가 다른 경우 어떻게 표시를 할까요?

기계마다 다 다르게 레이아웃을 만들어야 할까요?

사진/모든 크기

사용권

© NHNNEXT님이 모든 권리를 보유함

다운로드

이 사진의 중간 500 크기 다운로드

크기

사각형 75 (75 x 75)

사각형 150 (150 x 150)

썸네일 (100 x 67)

소형 240 (240 x 160)

소형 320 (320 x 213)

중간 500 (500 x 333)

중간 640 (640 x 427)

중형 800 (800 x 534)

대형 1024 (1024 x 683)

대형 1600 (1600 x 1067)

대형 2048 (2048 x 1365)

원본 (6000 x 4000)



333px

500px

우리가 흔히 알고 있는 픽셀로
화면을 구성할게 되면...



160DPI 저해상도
디바이스



320DPI 고해상도
디바이스

해상도가 다른 기계에서는
화면이 다르게 보일 수 있습니다.

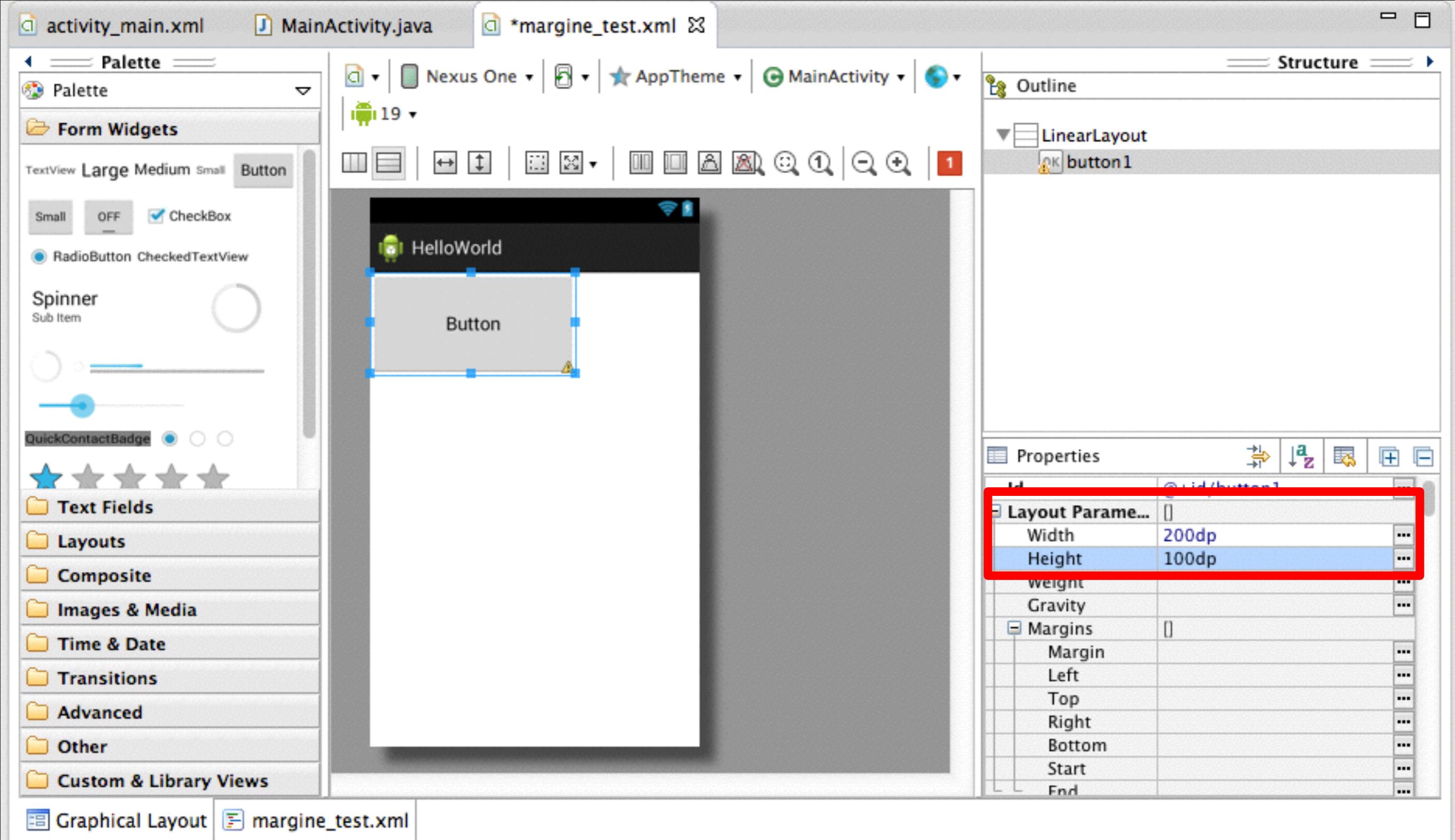


160DPI 저해상도
디바이스



320DPI 고해상도
디바이스

이를 막기 위해서는
가변적으로 크기가 변하는 단위인 DP를 사용해야합니다.



레이아웃 에디터의
Properties에서 크기값으로 dp를 사용하거나

The screenshot shows the Android Studio XML Editor. At the top, there are tabs for 'activity_main.xml', 'MainActivity.java', and '*margine_test.xml'. The '*margine_test.xml' tab is active. The code editor displays the following XML code:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:orientation="vertical">  
  
    <Button  
        android:id="@+id/button1"  
        android:layout_width="200dp"  
        android:layout_height="100dp"  
        android:text="button" />  
  
</LinearLayout>
```

A red box highlights the 'layout_width' and 'layout_height' attributes of the Button element, specifically the values '200dp' and '100dp'. A small yellow warning icon is visible on the left side of the editor.

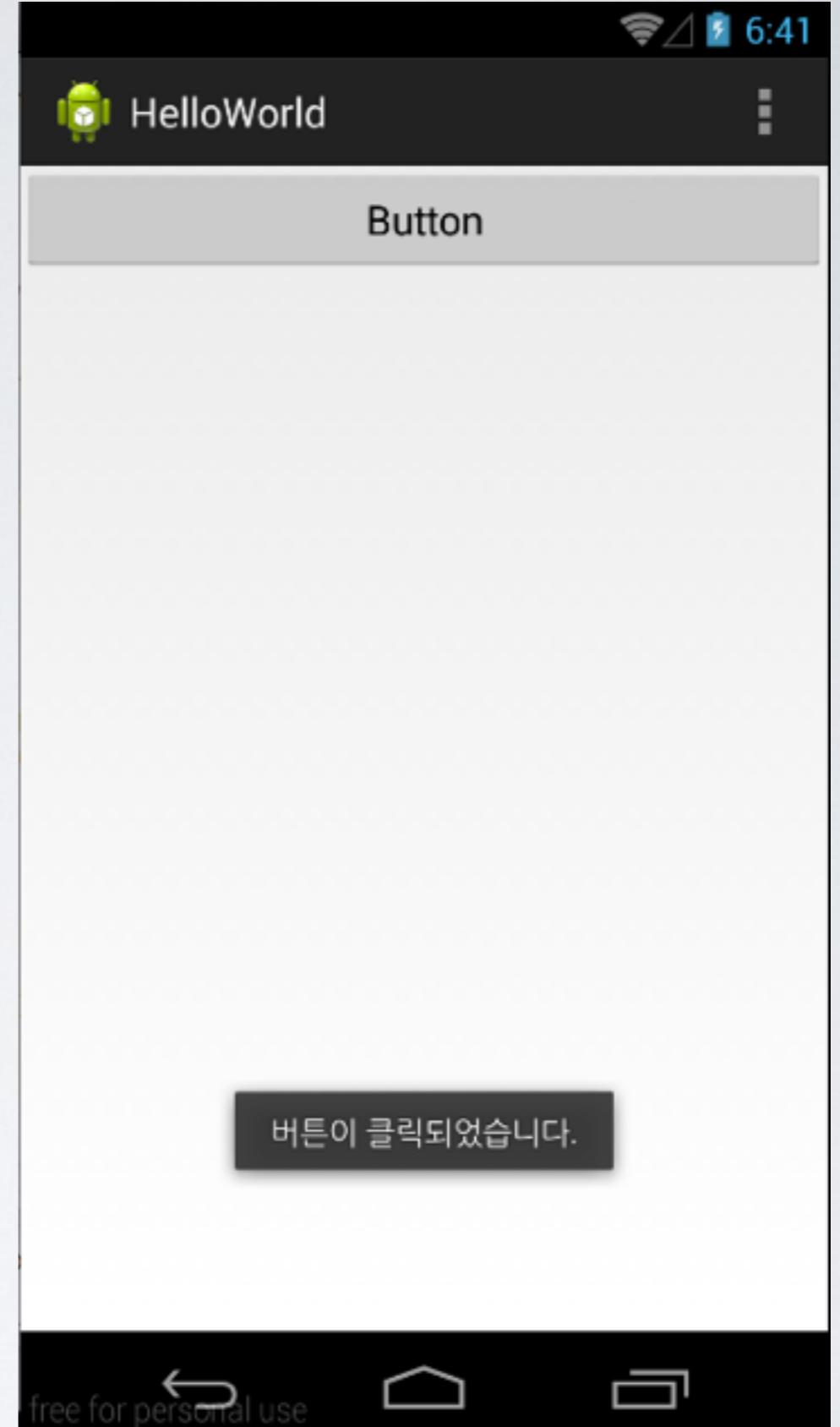
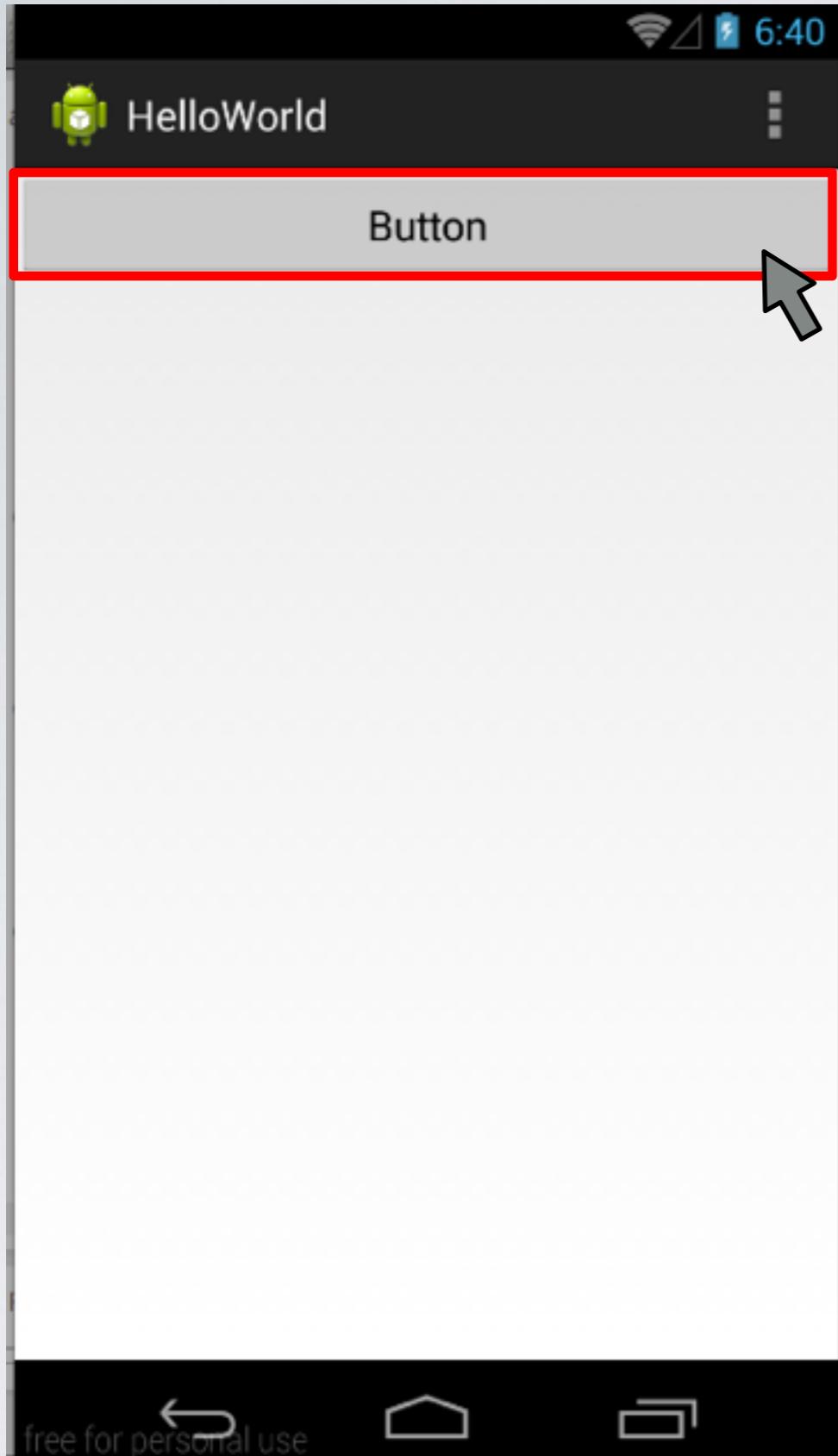
At the bottom, there are two tabs: 'Graphical Layout' (selected) and 'margine_test.xml'.

xml에서 레이아웃 크기 값을 dp로 사용하셔야 합니다.

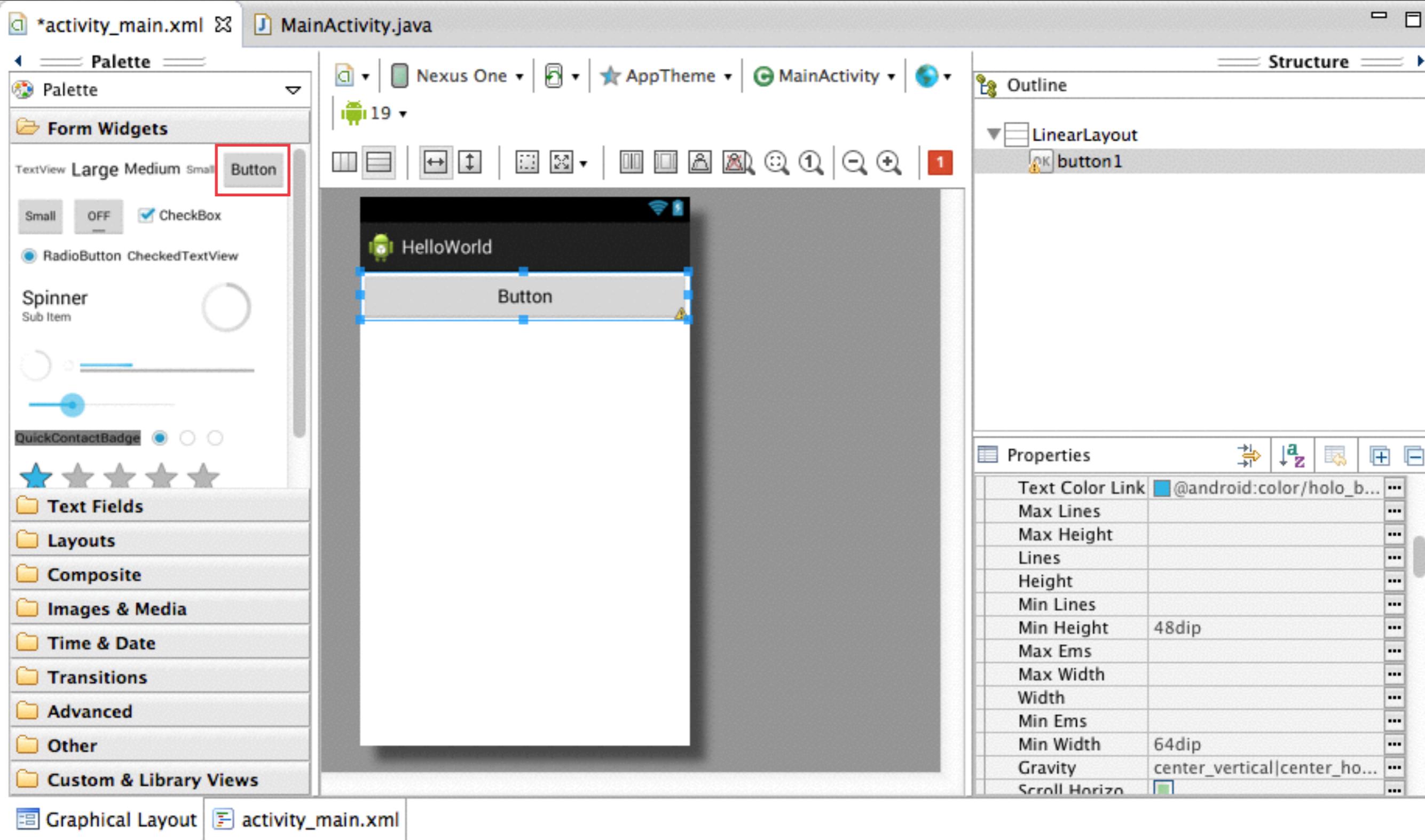
자세한 내용은 아래의 슬라이드를 참고하세요

<http://ui.nhnnext.org/crong/scope/android/files/DpPx.pdf>

버튼(Button) 위젯의 활용

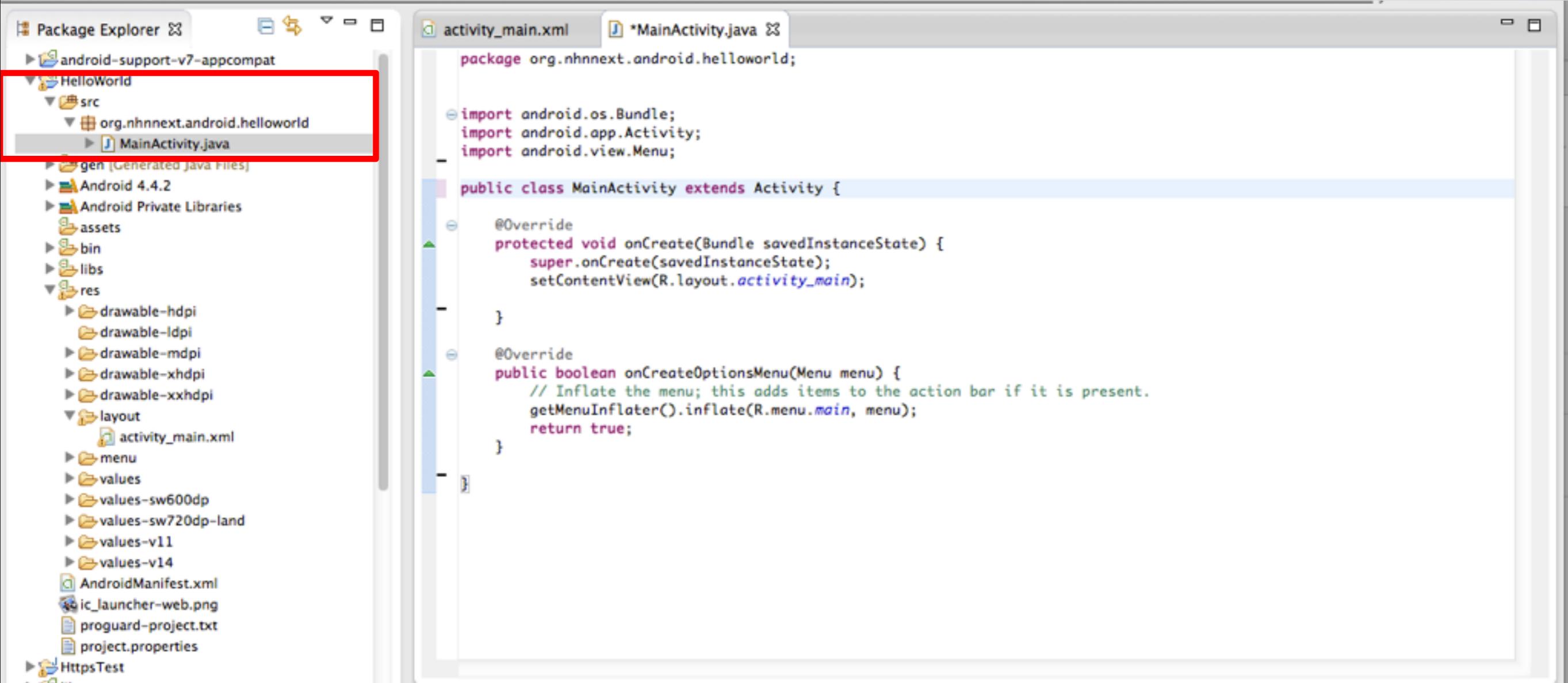


버튼을 터치하면 토스트 메시지가 출력되는 앱을 만들어 봅니다.



```
<Button  
    android:id="@+id/button1"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Button" />
```

먼저 버튼을 하나 생성을 합니다.
id는 button1
layout_width는 match_parent로 하였습니다.



레이아웃 작업은 완료를 하였고
소스코드를 수정하기 위해

프로젝트에서 src-package-MainActivity.java파일을 편집합니다.

```
package org.nhnnext.android.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

onCreate는 안드로이드 앱의 액티비티가 생성될때
실행되는 부분으로 한번만 실행이 되므로 초기화 등의 작업을 주로합니다.

onCreateOptionsMenu은 액션바나 메뉴의 아이템을 설정하는 부분으로
이번 예제에서는 사용되지 않으므로 지우셔도 무방합니다.

```
package org.nhnnext.android.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.widget.Button;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button1 = (Button) findViewById(R.id.button1);}

}
```

버튼을 사용하기 위해서는 Button 인스턴스를 만들고 레이아웃에 있는 Id를 연결해야합니다.

The screenshot shows the Android Studio interface. On the left is the Package Explorer with project files like android-support-v7-appcompat, HelloWorld, and various res/drawable folders. The main area has tabs for activity_main.xml, MainActivity.java, and R.java. The R.java code is displayed, showing a static final class R with various integer constants. One of these constants, button1=0x7f080000, is highlighted with a red box. Below it, the XML code for activity_main.xml is shown, specifically the <Button> tag which includes the android:id=" @+id/button1" attribute, also highlighted with a red box.

```
activity_main.xml MainActivity.java R.java
```

```
/* AUTO-GENERATED FILE. DO NOT MODIFY. */

package org.nhnnext.android.helloworld;

public final class R {
    public static final class attr {
    }
    public static final class dimen {
        /** Default screen margins, per the Android Design guidelines.

        Customize dimensions originally defined in res/values/dimens.xml (such as
        screen margins) for sw720dp devices (e.g. 10" tablets) in landscape here.

        */
        public static final int activity_horizontal_margin=0x7f040000;
        public static final int activity_vertical_margin=0x7f040001;
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int action_settings=0x7f080001;
        public static final int button1=0x7f080000;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    public static final class menu {
        public static final int main=0x7f070000;
    }
    public static final class string {
        public static final int action_settings=0x7f050001;
        public static final int app_name=0x7f050000;
    }
}
```

```
<Button
    android:id="@+id/button1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button" />
```

ID는 레이아웃에 적은 ID로 결정이 되며
프로젝트의 gen-R.java에 자동으로 추가가 됩니다.
(R.java는 임의로 건드릴 필요가 없는 파일입니다...)

```
activity_main.xml *MainActivity.java R.java

package org.nhnnext.android.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

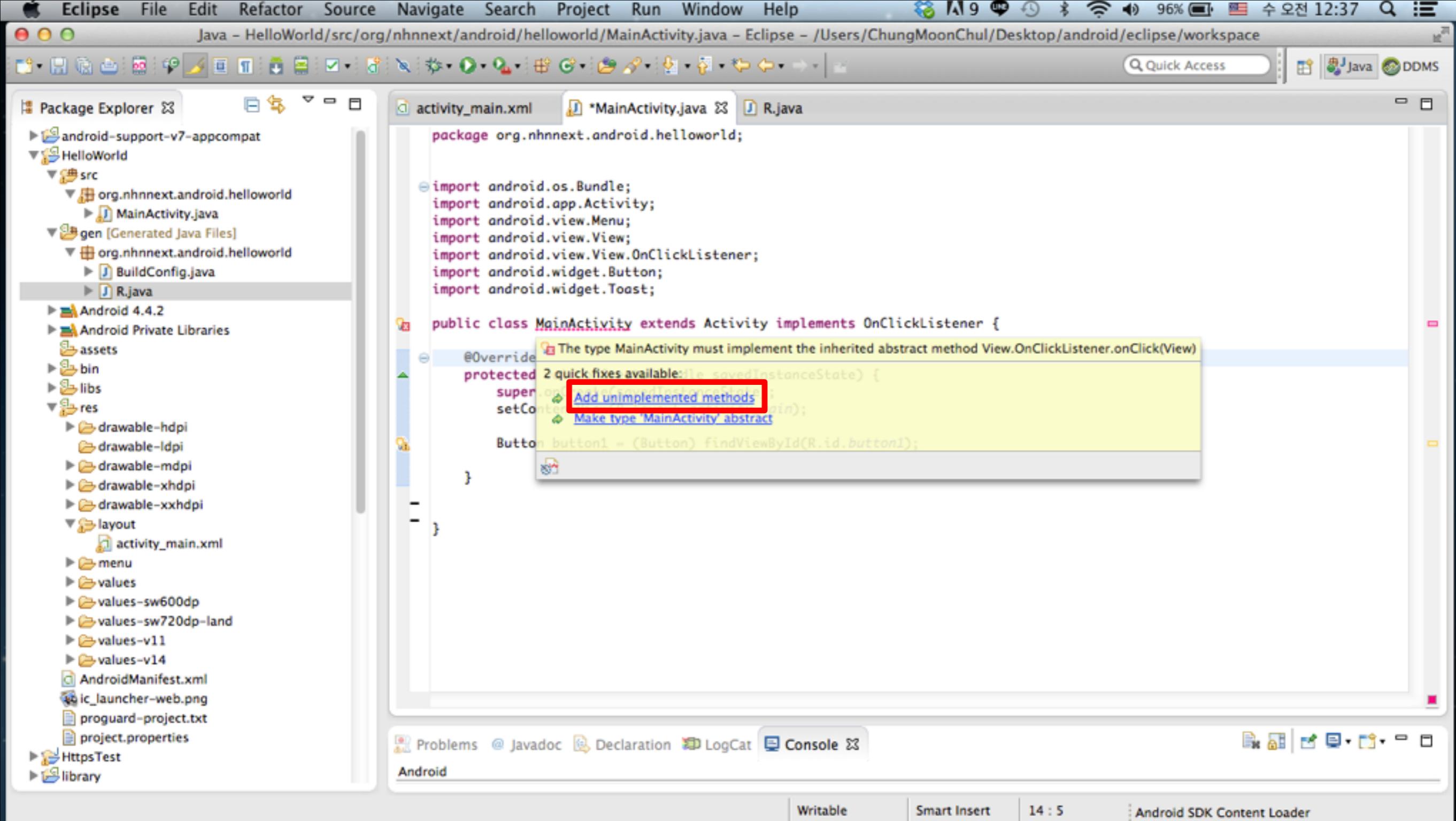
public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

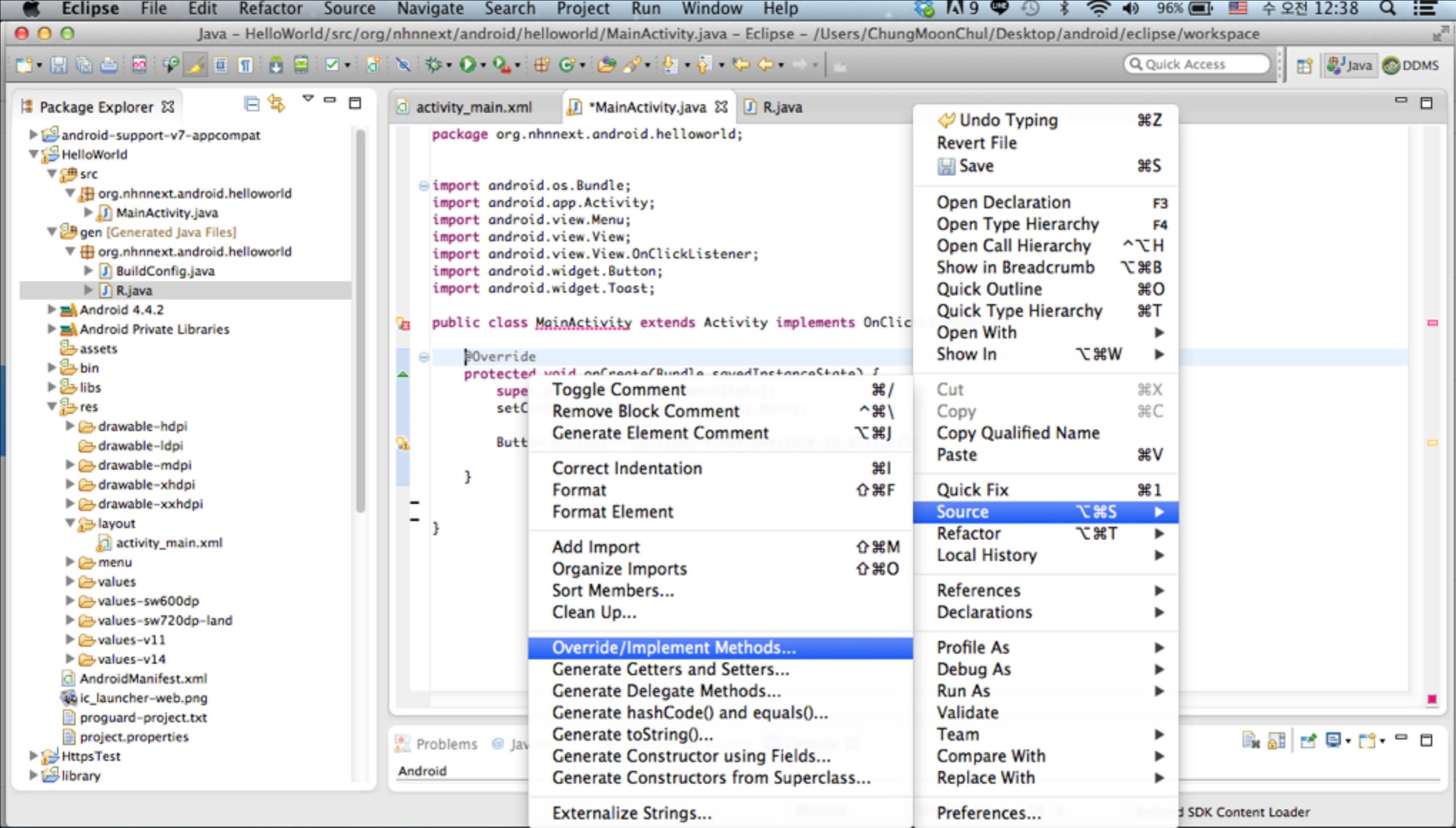
        Button button1 = (Button) findViewById(R.id.button1);

    }
}
```

버튼에 이벤트 리스너를 달기 위해
OnClickListener 인터페이스를 받아오겠습니다.



마우스를 올린후 Add unimplemented methods를 누르거나



마우스 오른쪽 클릭 후 Source-Implement Methods를 눌러
OnClickListener의 메소드를 생성해 주세요

```
activity_main.xml *MainActivity.java R.java

package org.nhnnext.android.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

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

    }

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
    }
}
```

onClick()이 자동으로 생성되었습니다.

(자동 생성말고 직접 입력하셔도 됩니다.)

```
activity_main.xml *MainActivity.java R.java
package org.nhnnext.android.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button1 = (Button) findViewById(R.id.button1);
        button1.setOnClickListener(this); // Line highlighted with a red box

    }

    @Override
    public void onClick(View v) {

    }

}

}
```

버튼에 setOnClickListener()로
리스너를 달아 주었습니다.

```
activity_main.xml *MainActivity.java R.java
```

```
package org.nhnnext.android.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity implements OnClickListener {

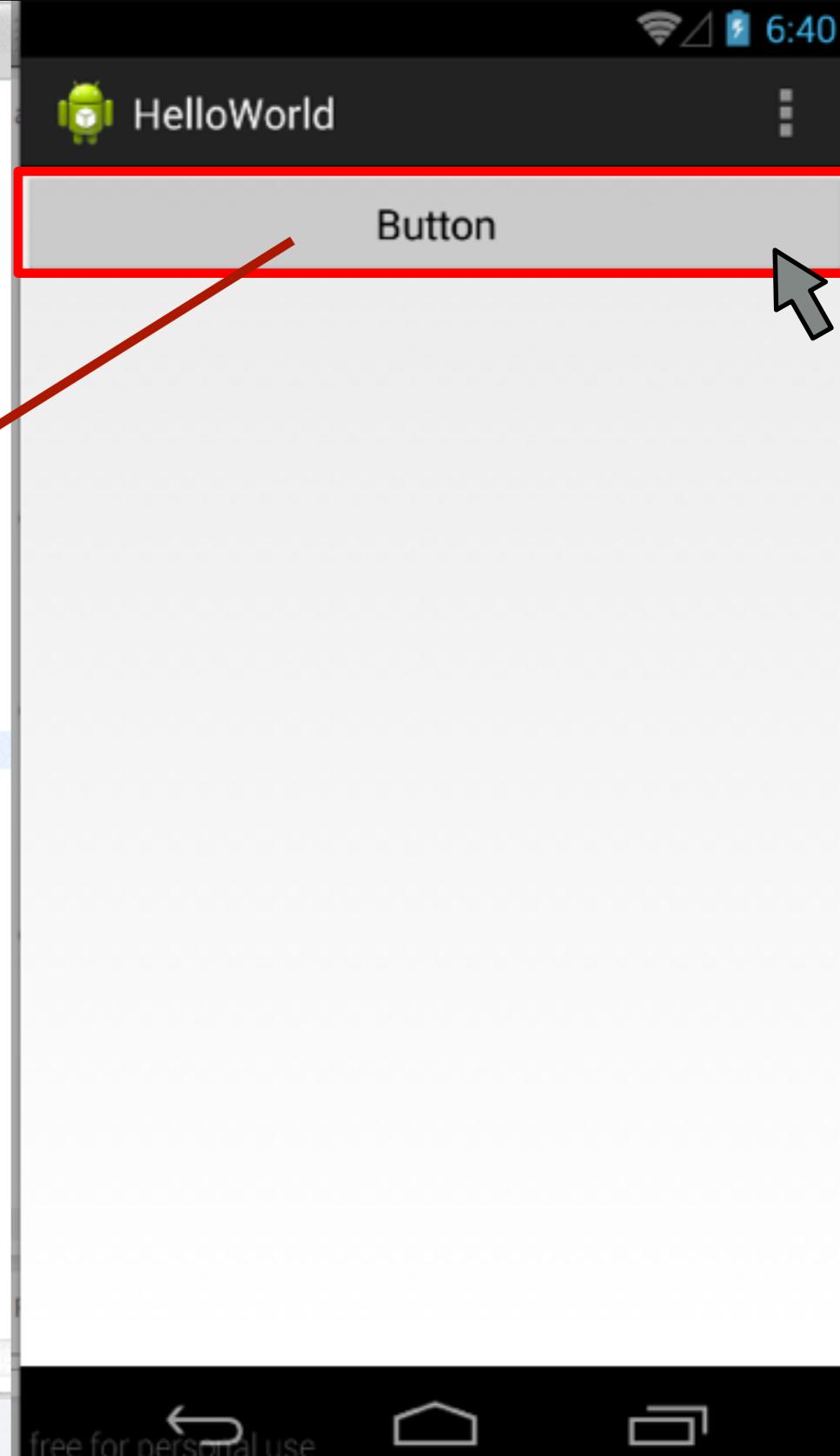
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button1 = (Button) findViewById(R.id.button1);
        button1.setOnClickListener(this); // This line is highlighted with a red arrow

    }

    @Override
    public void onClick(View v) {
    }

}
```



이제 버튼을 누르면 `onClick()`부분이 실행됩니다.

```
activity_main.xml>MainActivity.java>R.java>ButtonSample.java
```

```
package org.nhnnext.android.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener {

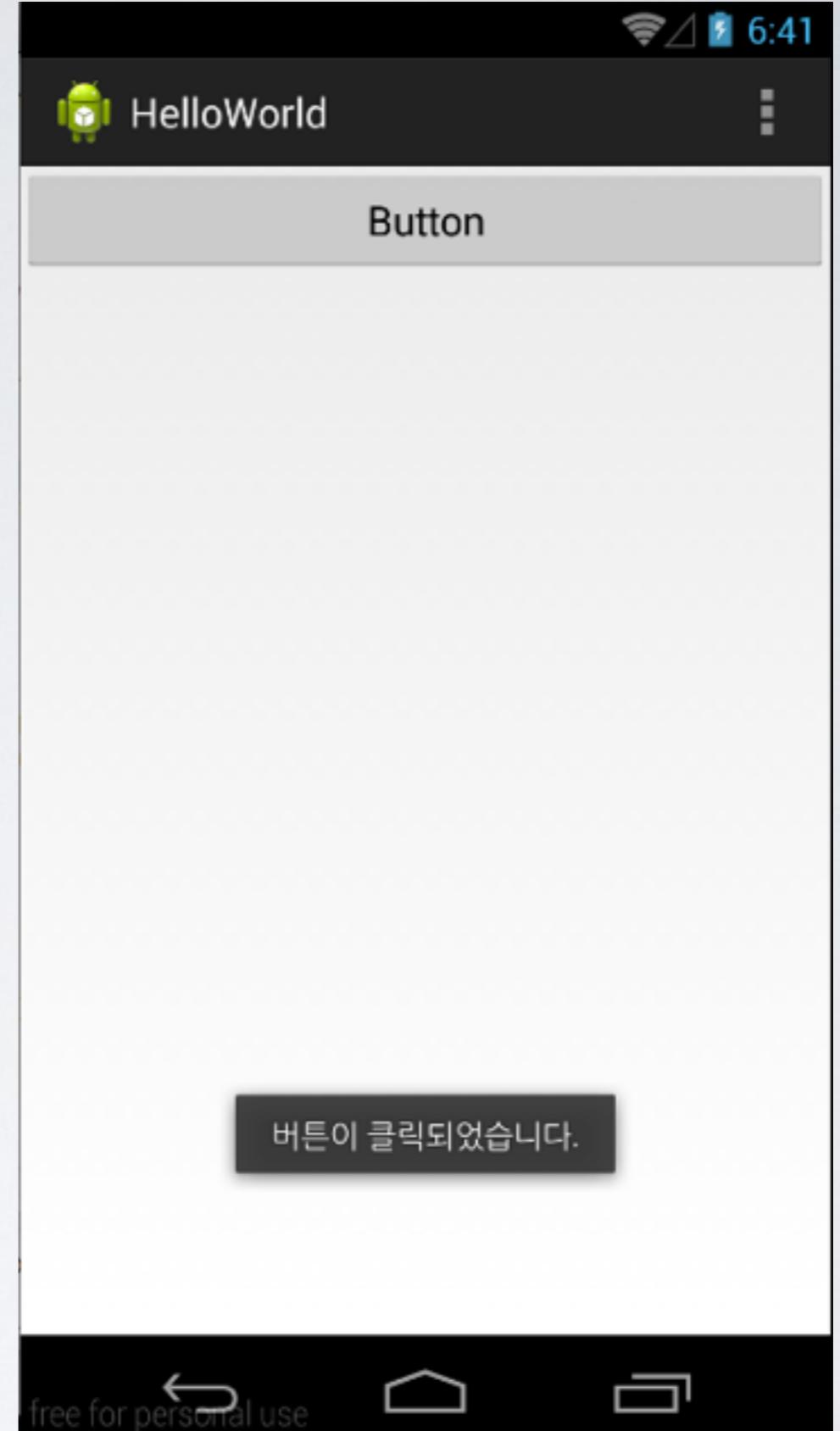
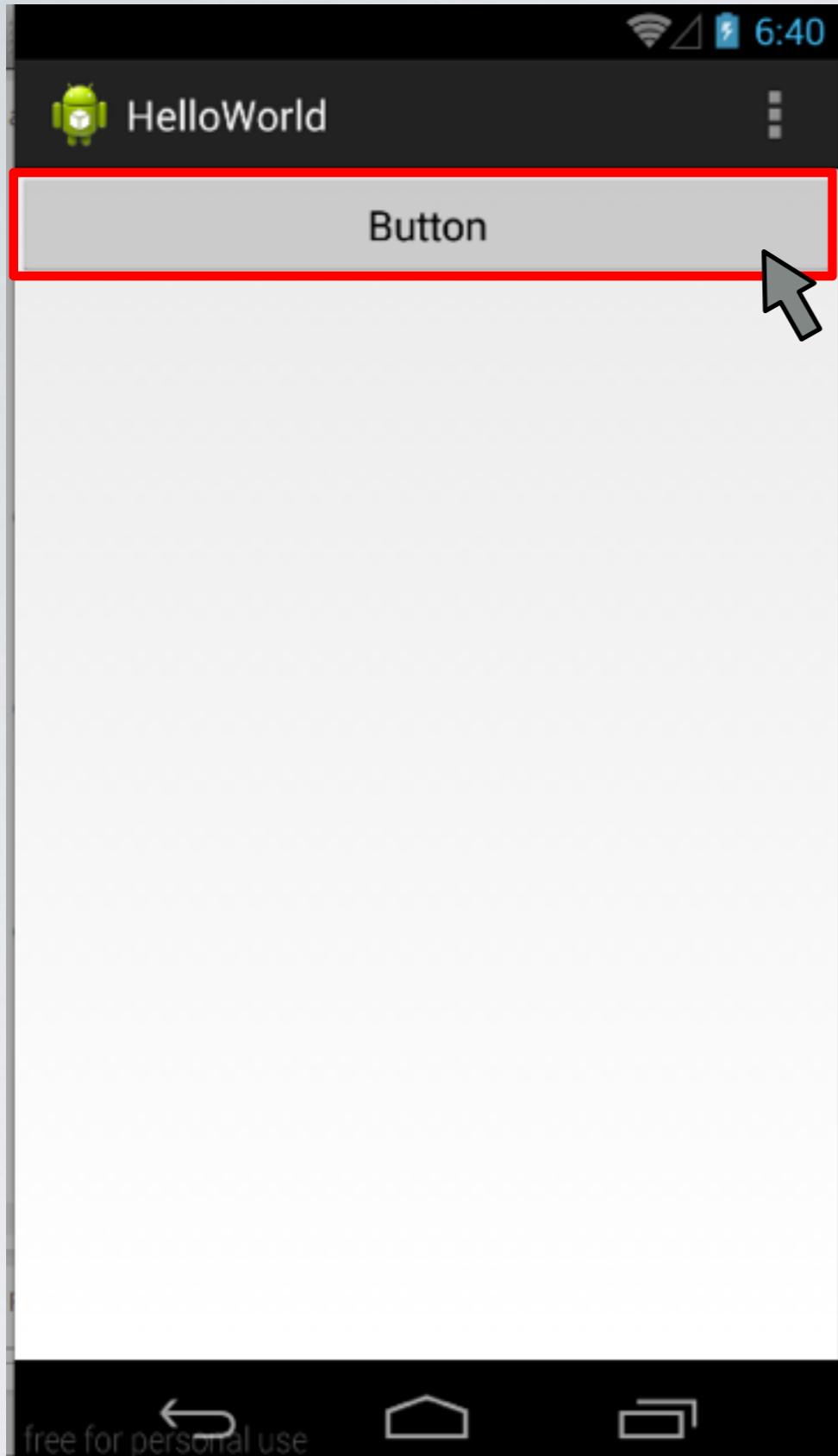
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button1 = (Button) findViewById(R.id.button1);
        button1.setOnClickListener(this);

    }

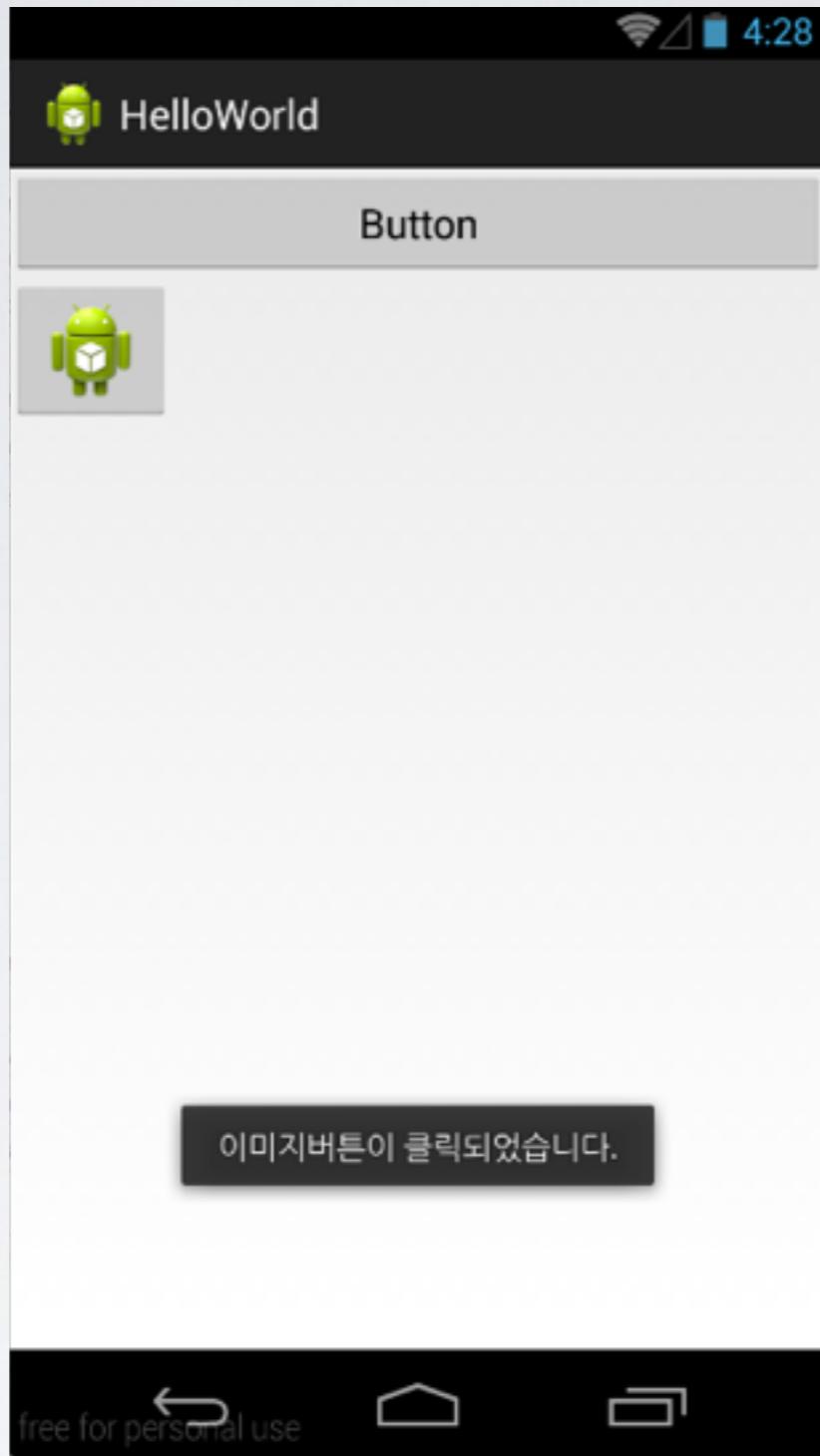
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button1:
                Toast.makeText(getApplicationContext(), "버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();
                break;
        }
    }
}
```

지금은 버튼이 하나만이여서 동작에는 차이가 없지만
리스너가 여러개가 달릴것을 대비하여 switch문으로
id값으로 명령어를 달리 합니다.

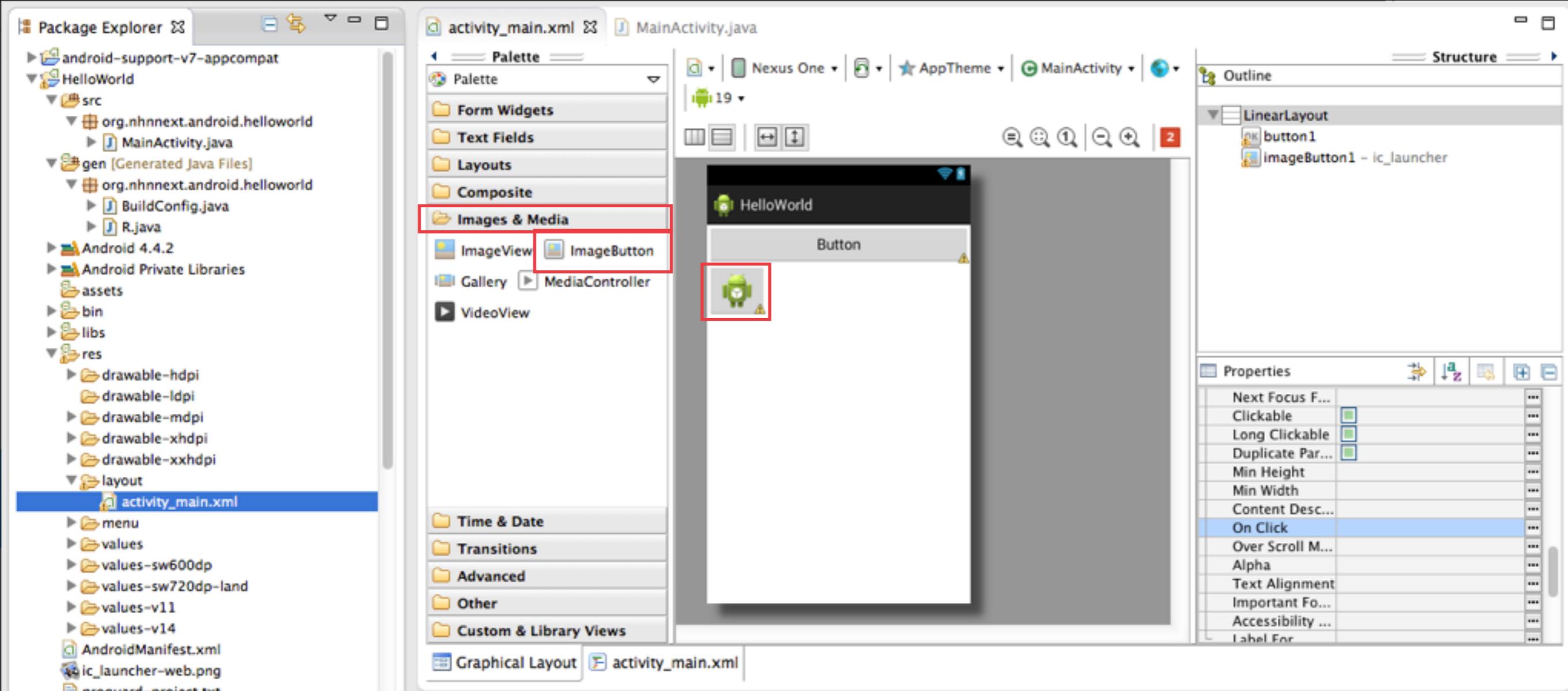


버튼 클릭 완성!

이미지 버튼(ImageButton)



이번에는 이미지 버튼을 사용해보겠습니다.
이미지 버튼은 버튼과 동일하나
버튼에 이미지를 쉽게 넣을 수 있습니다.



```
<ImageButton  
    android:id="@+id/imageButton1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/ic_launcher" />
```

버튼과 마찬가지로 레이아웃 파일에서
ImageButton을 만들어 줍니다.



이미지버튼을 만들 때 사용하는 이미지는
기본적으로 있는 앱 아이콘을 사용하겠습니다.

```

import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button1 = (Button) findViewById(R.id.button1);
        button1.setOnClickListener(this);

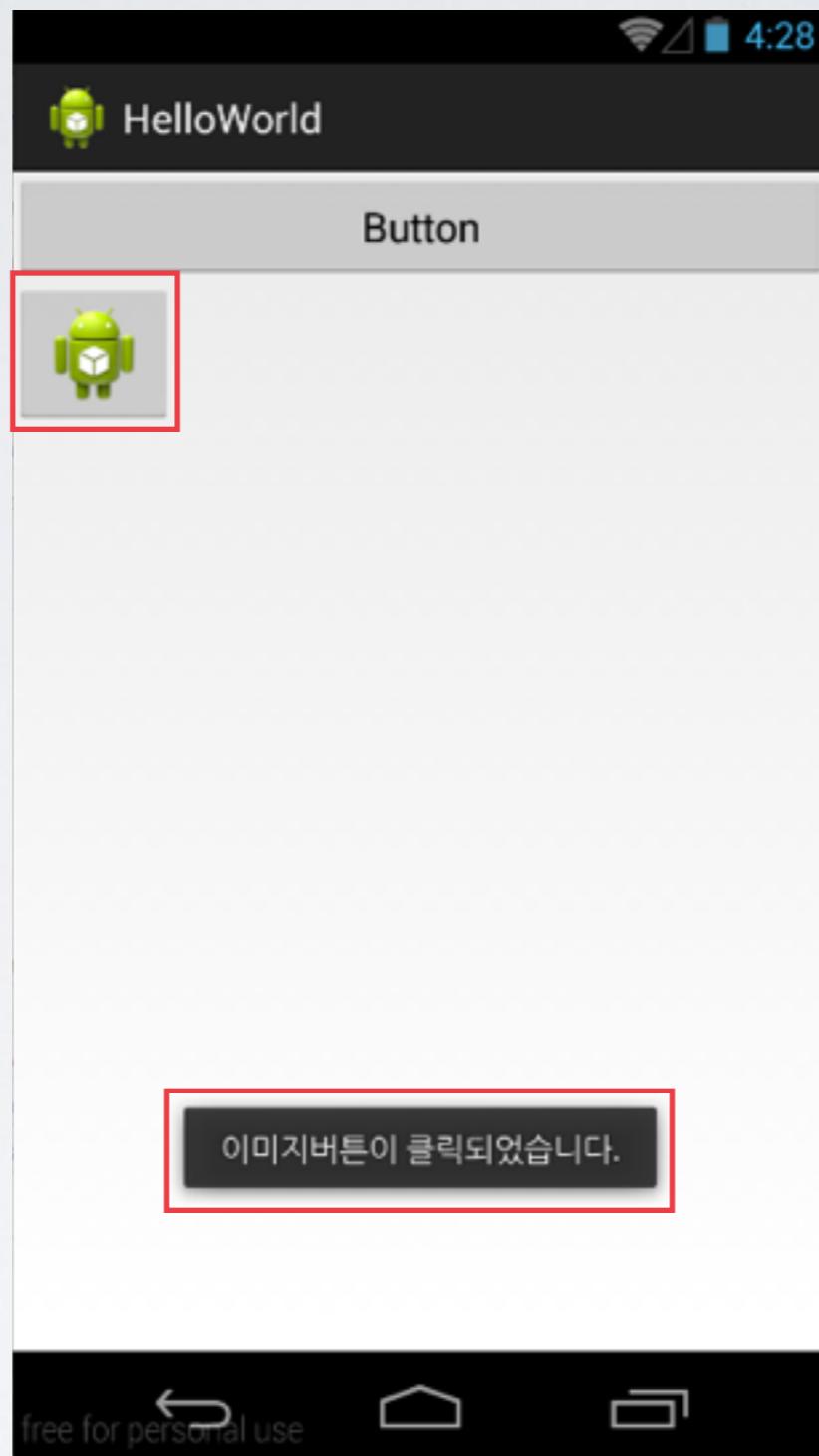
        ImageButton imageView1 = (ImageButton) findViewById(R.id.imageView1);
        imageView1.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button1:
                Toast.makeText(getApplicationContext(), "버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();
                break;
            case R.id.imageView1:
                Toast.makeText(getApplicationContext(), "이미지버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();
                break;
        }
    }
}

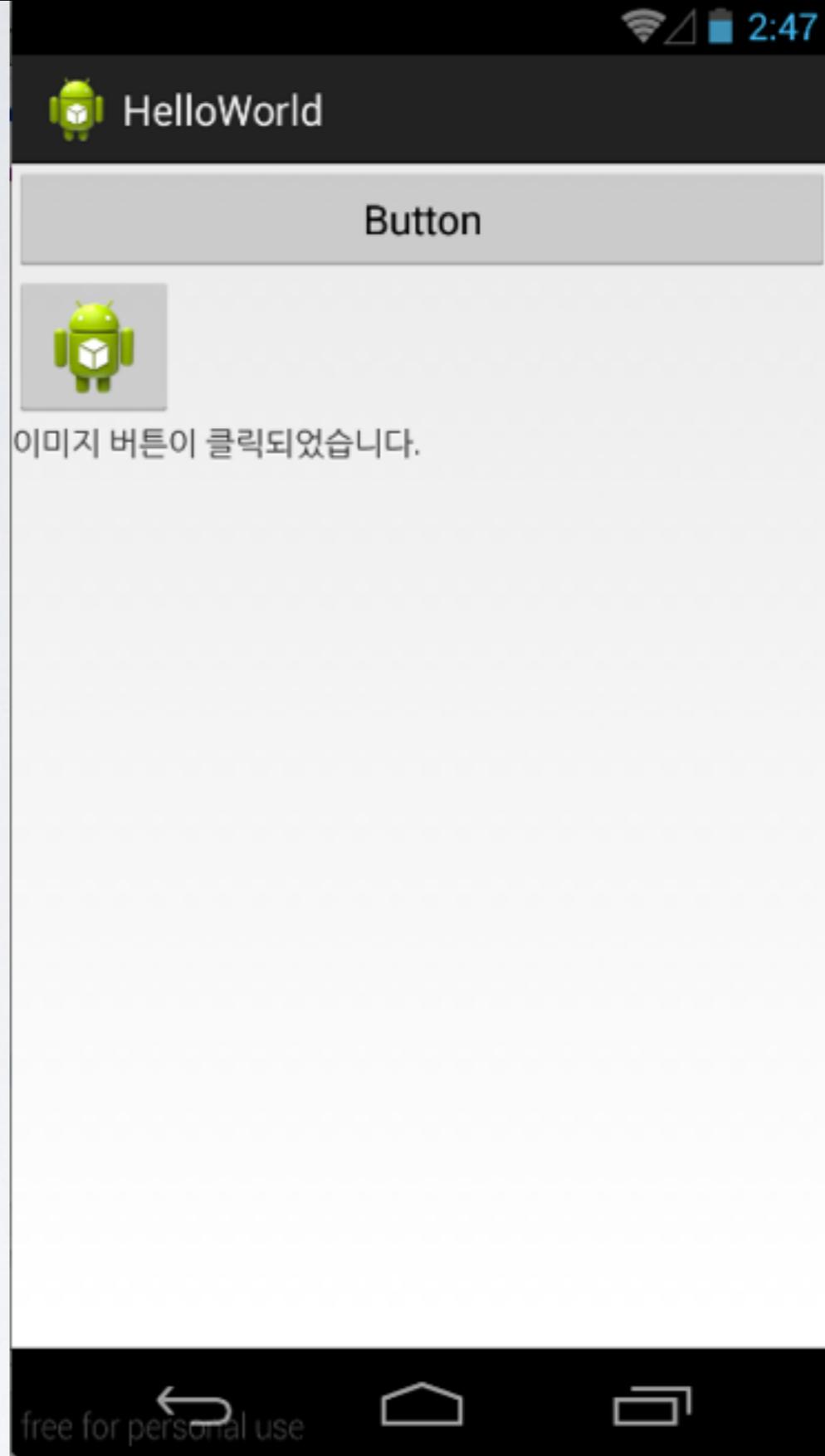
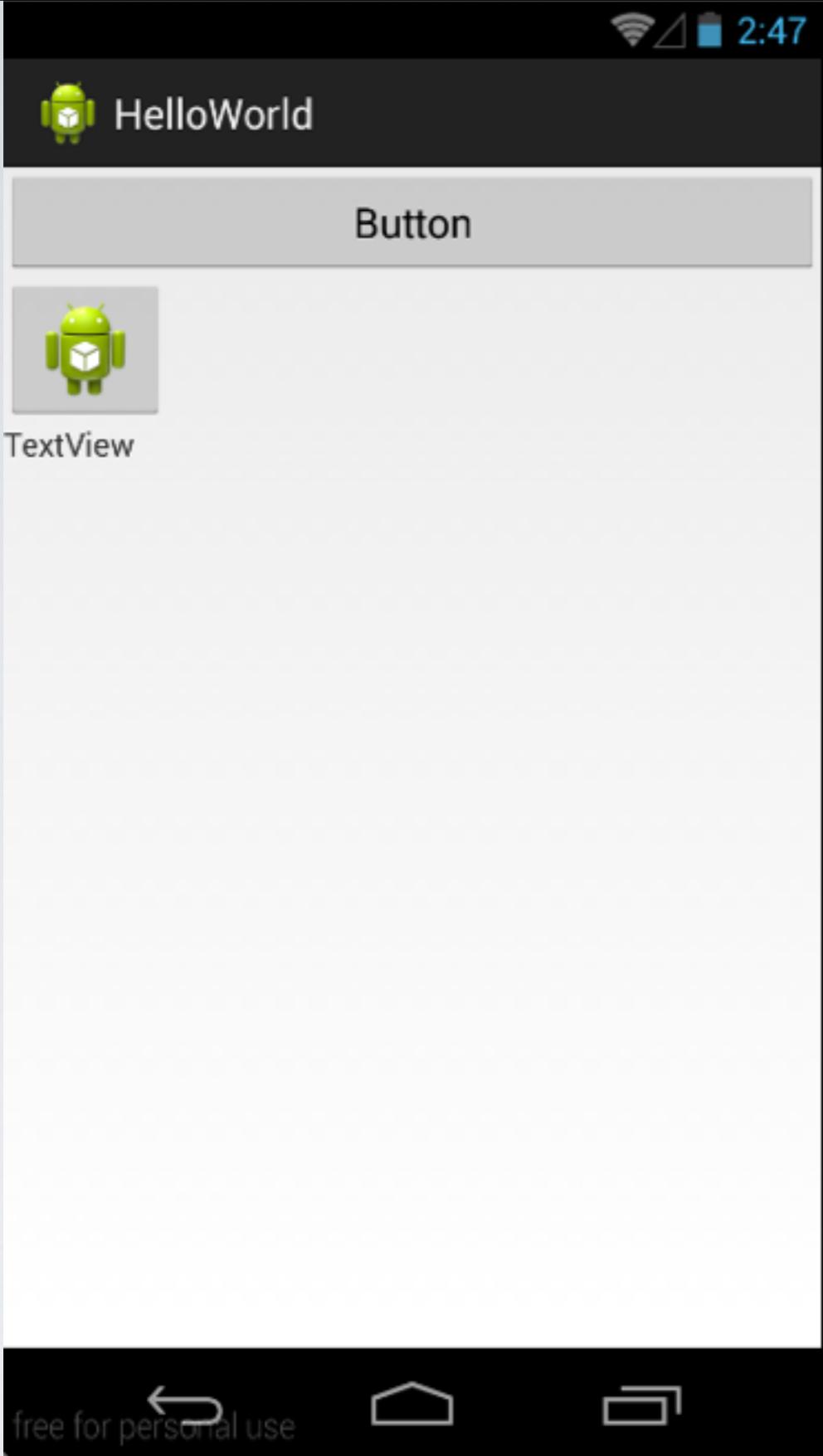
```

소스코드는 버튼과 흡사합니다.

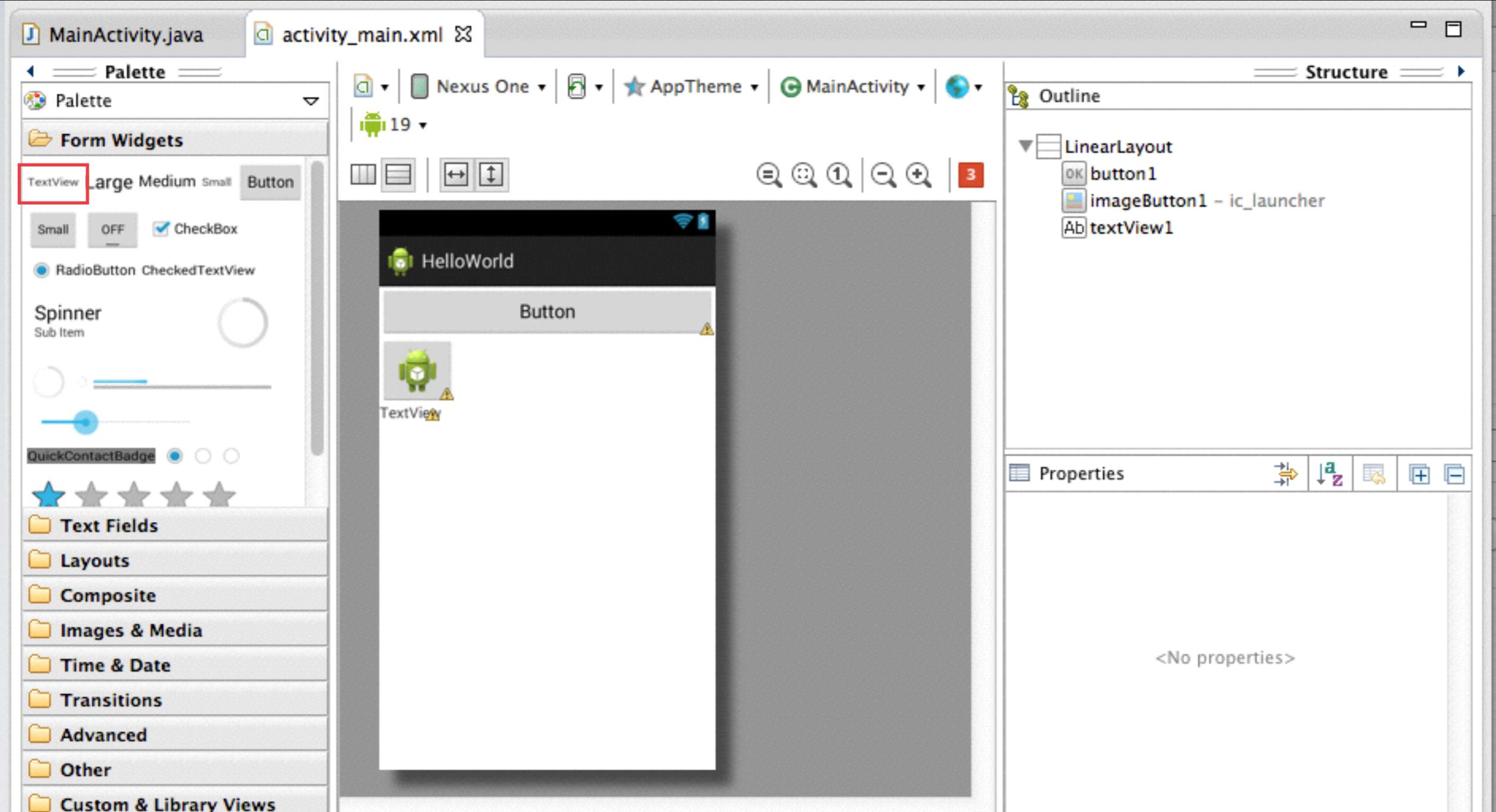
Button대신 ImageButton을 사용하고
리스너 구분을 위해 switch문을 사용합니다.



텍스트 뷰(TextView)



문자열을 표시하기 위해서는 TextView를 사용합니다.
이번에는 버튼을 누르면 TextView의 내용이 달라지는 방법을 알아봅니다.



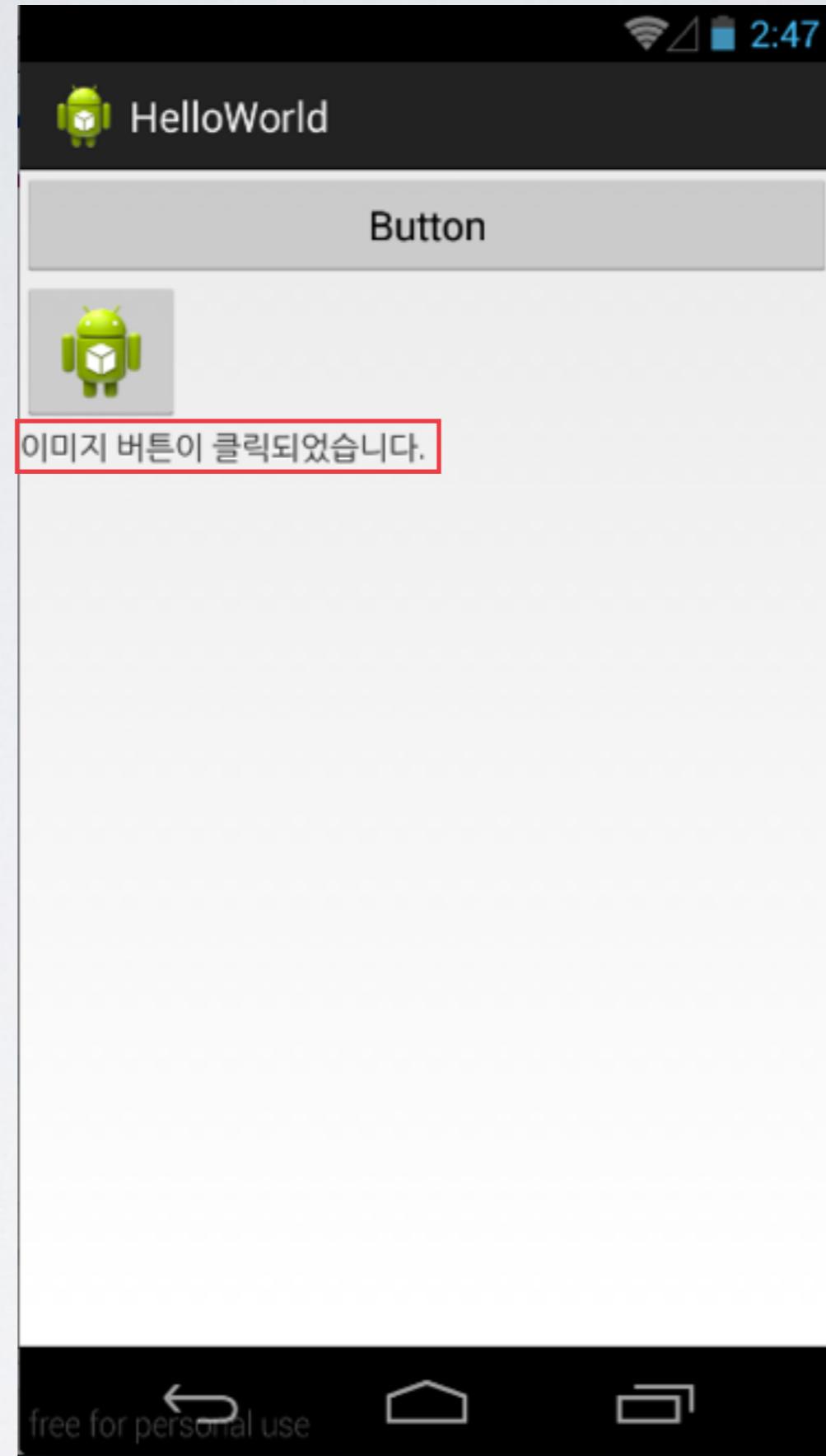
레이아웃에디터에서 TextView를 추가합니다.

```
public class MainActivity extends Activity implements OnClickListener {  
    private TextView textView1;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button button1 = (Button) findViewById(R.id.button1);  
        button1.setOnClickListener(this);  
  
        ImageButton imageButton1 = (ImageButton) findViewById(R.id.imageButton1);  
        imageButton1.setOnClickListener(this);  
  
        textView1 = (TextView) findViewById(R.id.textView1);  
    }  
  
    @Override  
    public void onClick(View v) {  
        switch (v.getId()) {  
            case R.id.button1:  
                Toast.makeText(getApplicationContext(), "버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();  
                break;  
            case R.id.imageButton1:  
                Toast.makeText(getApplicationContext(), "이미지버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();  
                break;  
        }  
    }  
}
```

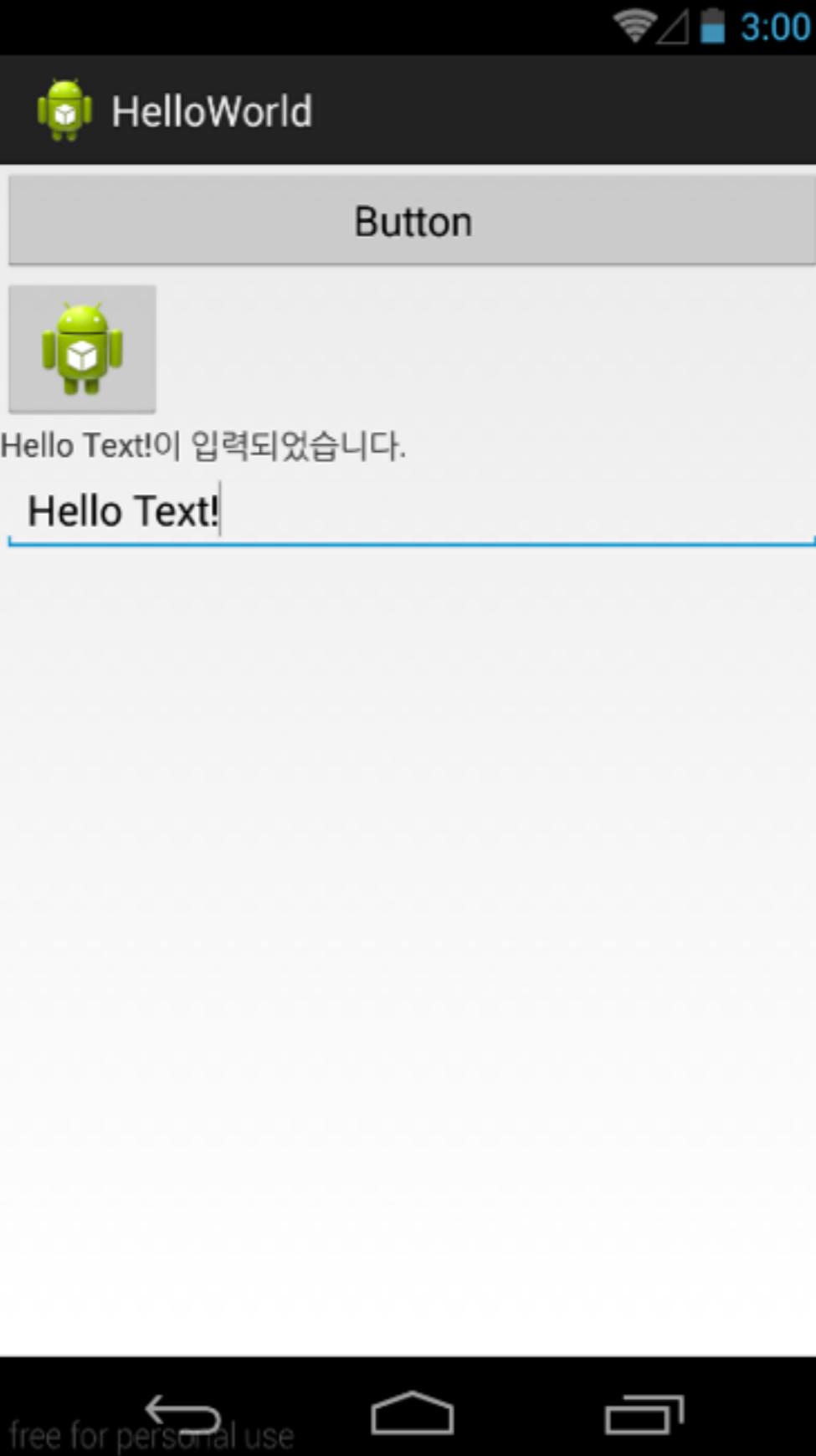
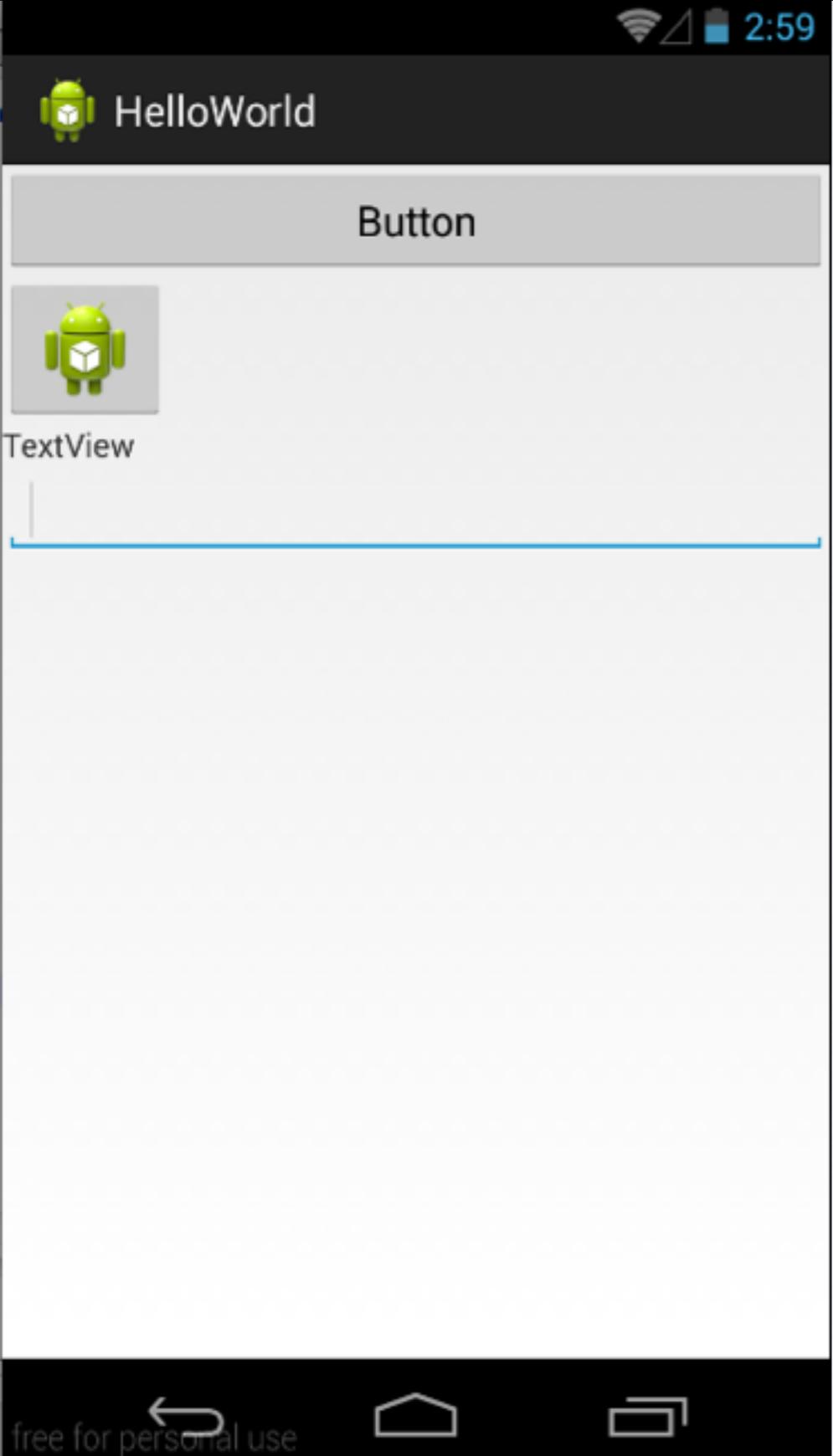
TextView인스턴스를 만들고 레이아웃의 Id를 연결합니다.
(버튼과 달리 onClick()에서 사용할 예정이므로 밖에서 선언합니다.)

```
public class MainActivity extends Activity implements OnClickListener {  
  
    private TextView textView1;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button button1 = (Button) findViewById(R.id.button1);  
        button1.setOnClickListener(this);  
  
        ImageButton imageButton1 = (ImageButton) findViewById(R.id.imageButton1);  
        imageButton1.setOnClickListener(this);  
  
        textView1 = (TextView) findViewById(R.id.textView1);  
    }  
  
    @Override  
    public void onClick(View v) {  
        switch (v.getId()) {  
            case R.id.button1:  
                Toast.makeText(getApplicationContext(), "버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();  
                textView1.setText("버튼이 클릭되었습니다.");  
                break;  
            case R.id.imageButton1:  
                Toast.makeText(getApplicationContext(), "이미지버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();  
                textView1.setText("이미지 버튼이 클릭되었습니다.");  
                break;  
        }  
    }  
}
```

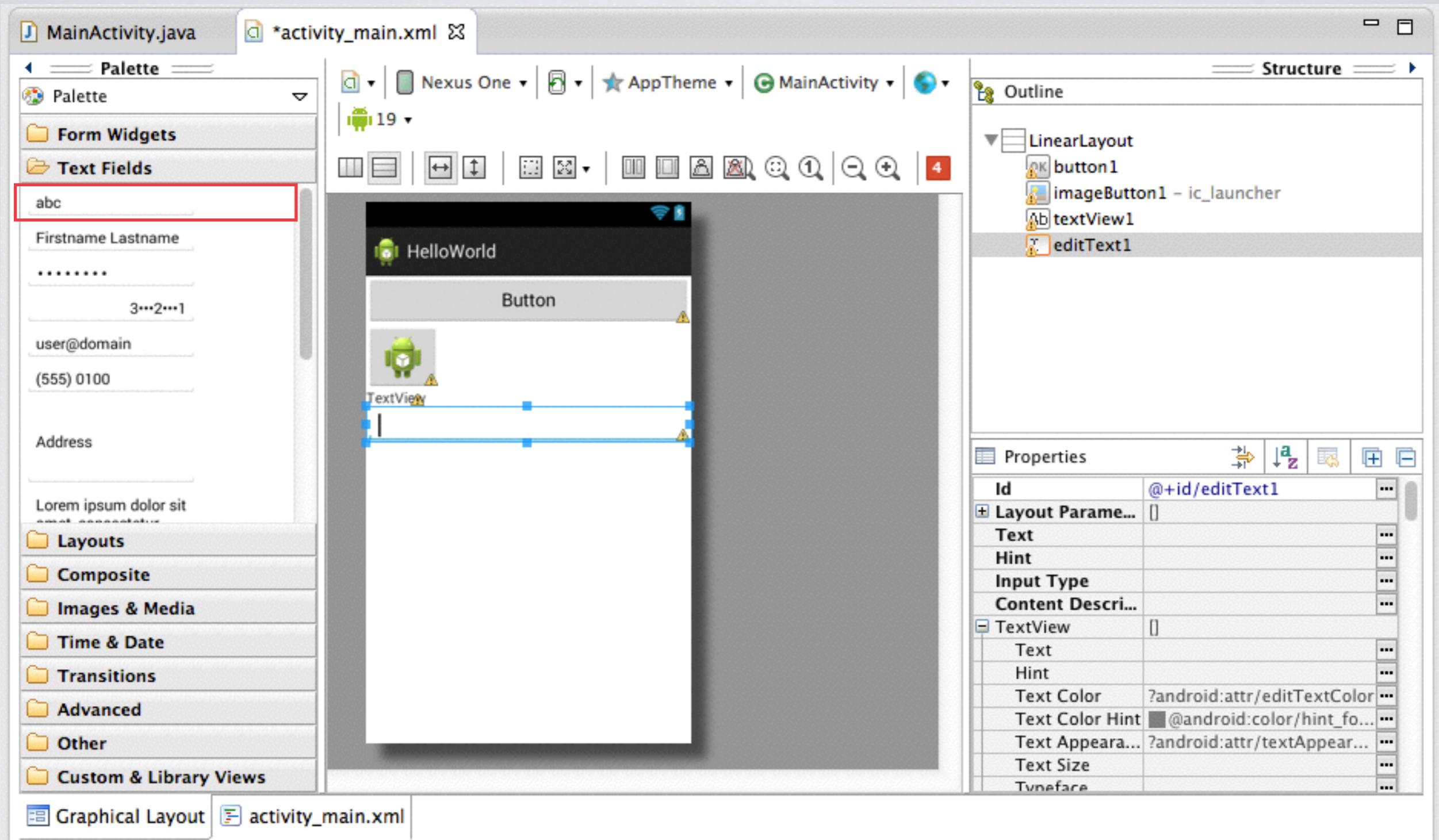
TextView의 내용을 바꾸기 위해서는 setText()를 사용합니다.
각각 버튼에 맞게 setText("메시지")를 입력합니다.



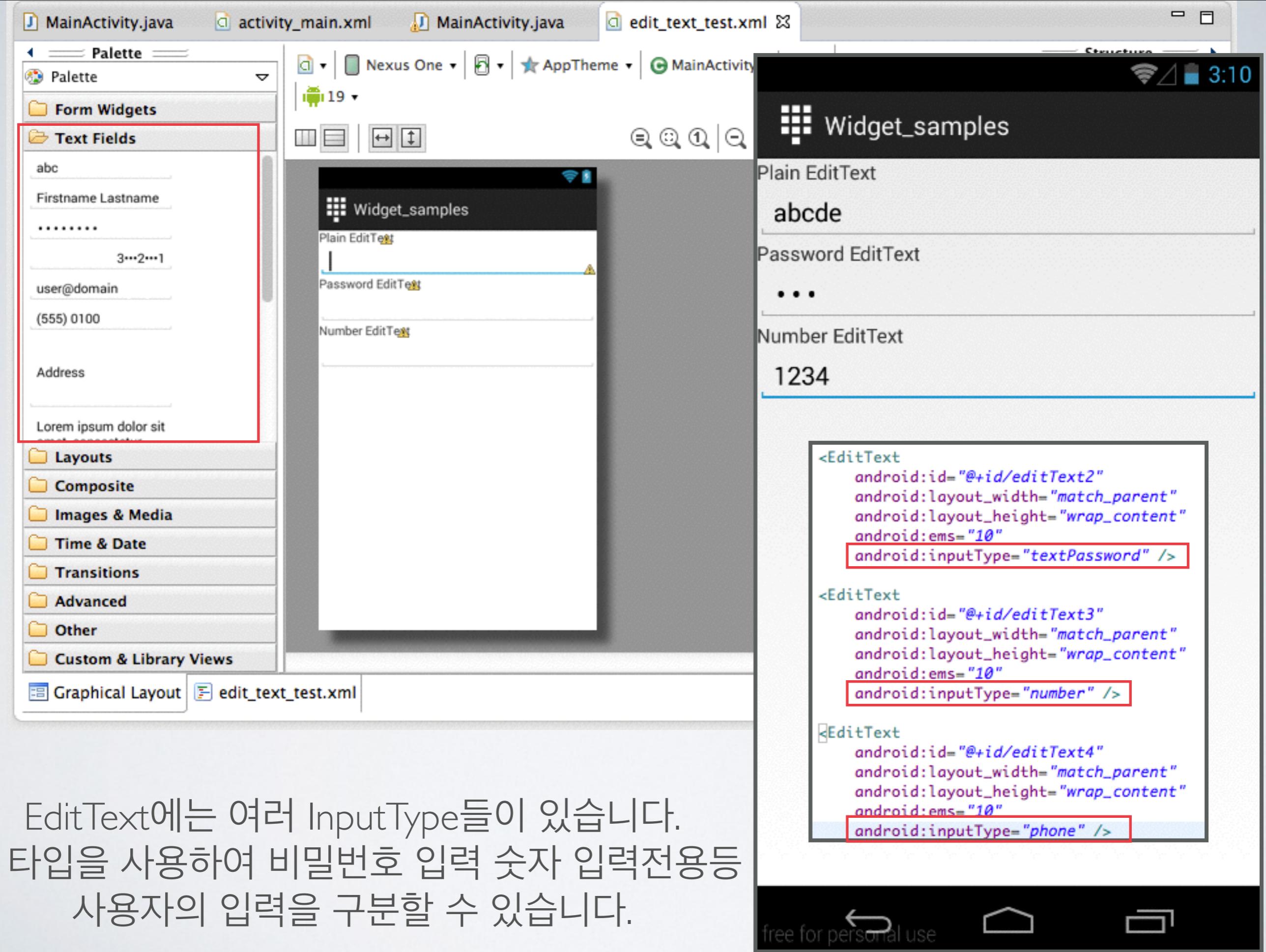
에디트 텍스트(EditText, 입력창)



버튼을 누르면 사용자가 EditText에 입력한 내용을 가져와서 TextView에 보여지도록 하겠습니다.



레이아웃에디터에서 EditText를 추가합니다.



*activity_main.xml>MainActivity.java

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context=".MainActivity" >  
  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/hello_world" />  
  
</RelativeLayout>
```



Graphical Layout activity_main.xml

TextView와 마찬가지로 EditText인스턴스를 만들고
레이아웃 Id를 연결해줍니다.

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button1 = (Button) findViewById(R.id.button1);
        button1.setOnClickListener(this);

        ImageButton imageButton1 = (ImageButton) findViewById(R.id.imageButton1);
        imageButton1.setOnClickListener(this);

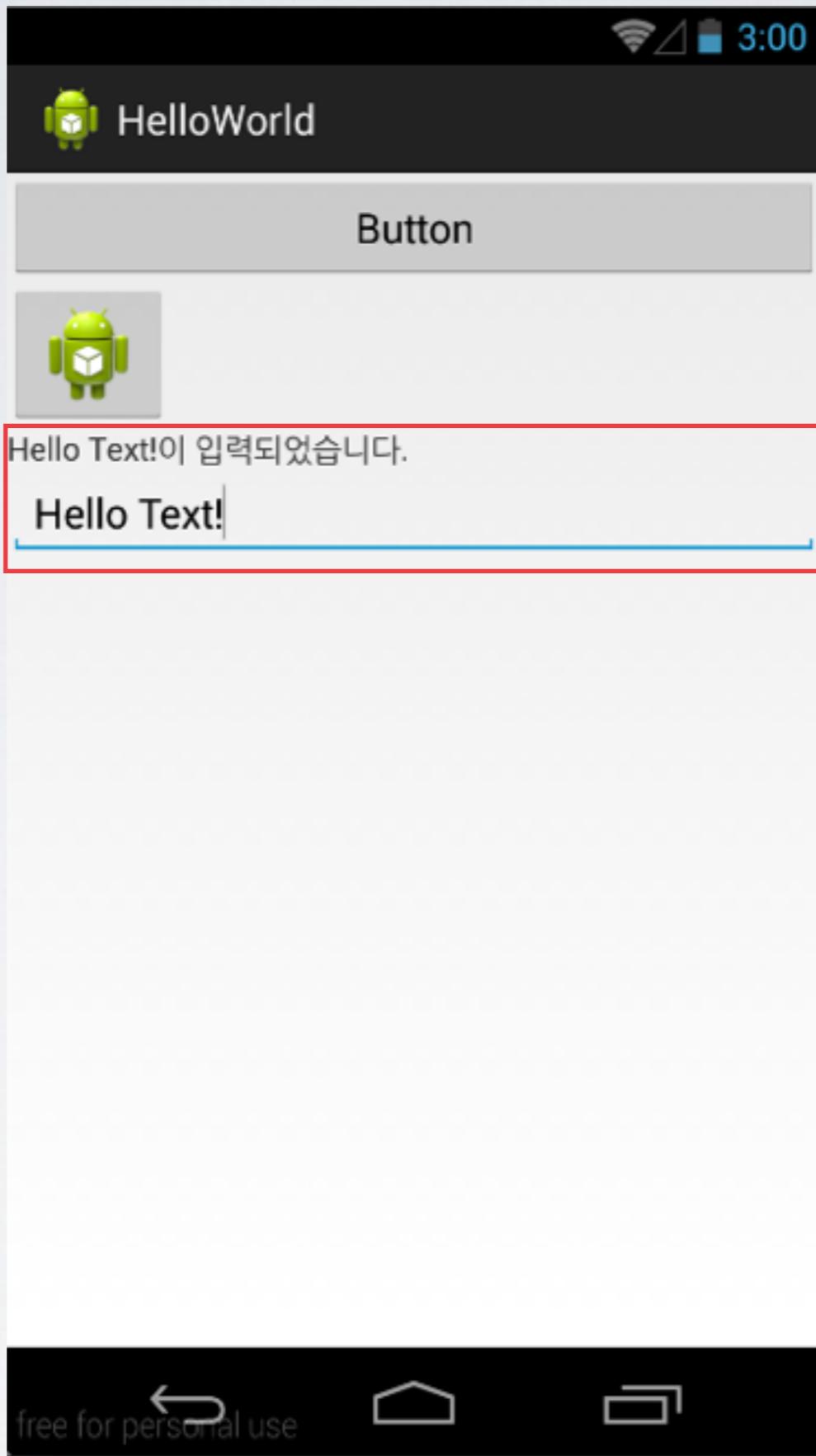
        textView1 = (TextView) findViewById(R.id.textView1);

        editText1 = (EditText) findViewById(R.id.editText1);

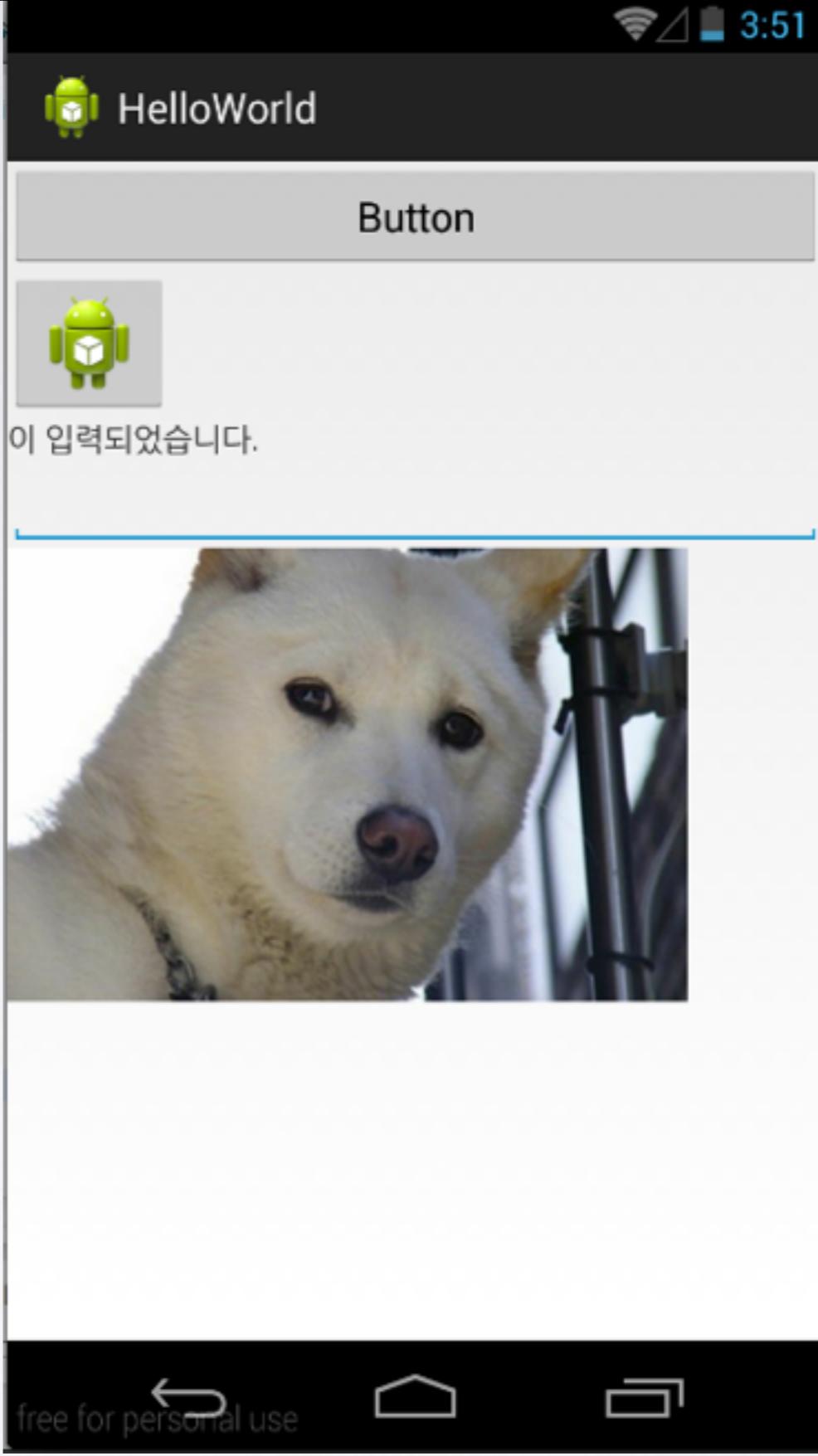
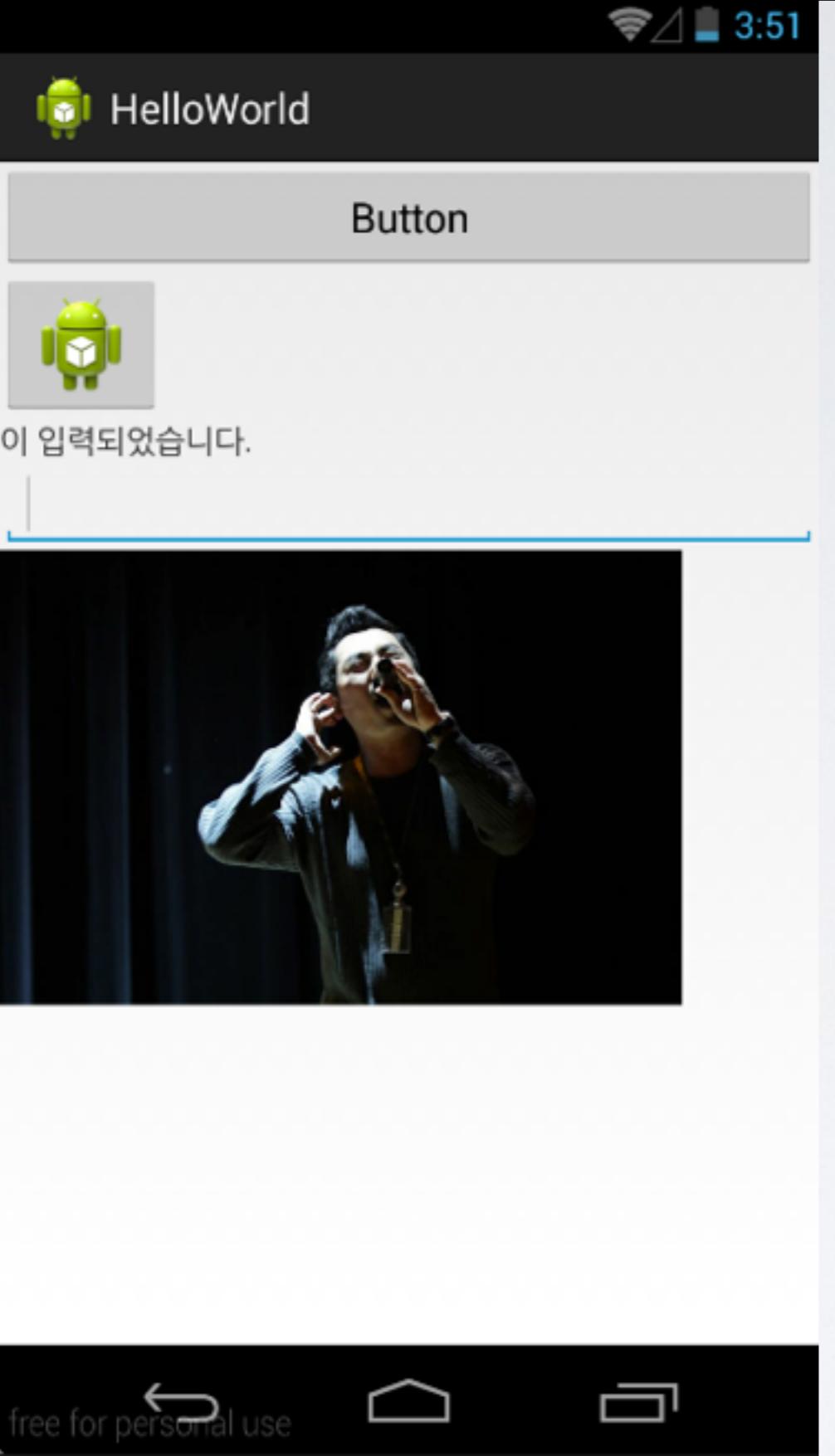
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button1:
                Toast.makeText(getApplicationContext(), "버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();
                textView1.setText(editText1.getText().toString() + "이 입력되었습니다.");
                break;
            case R.id.imageButton1:
                Toast.makeText(getApplicationContext(), "이미지버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();
                textView1.setText(editText1.getText().toString() + "이 입력되었습니다.");
                break;
        }
    }
}
```

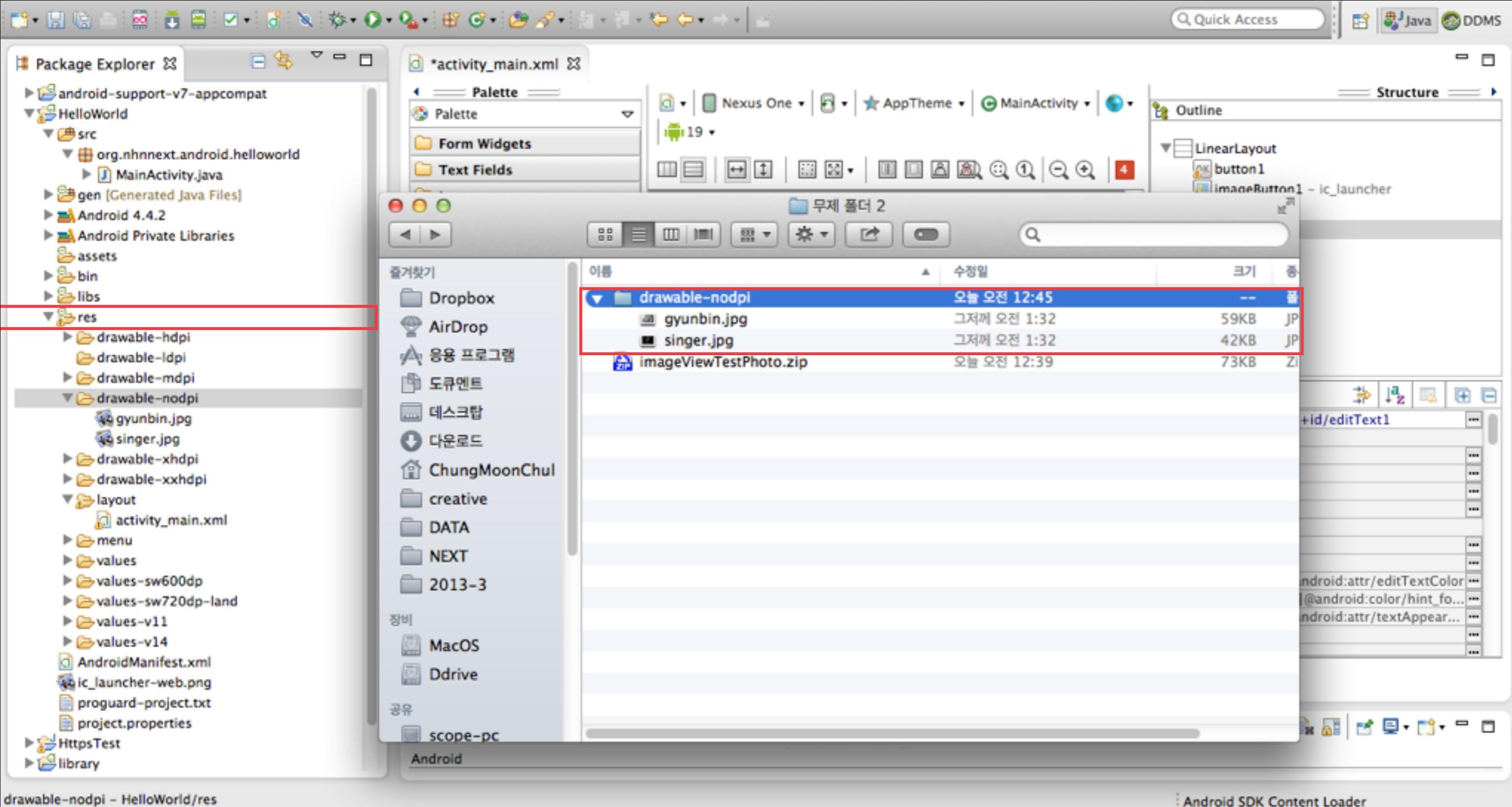
기존에 setText()가 있던곳을 수정하여
EditText의 getText()를 사용하여 EditText에 입력된 내용을
TextView에 표시하도록 합니다.



이미지 뷰(ImageView)



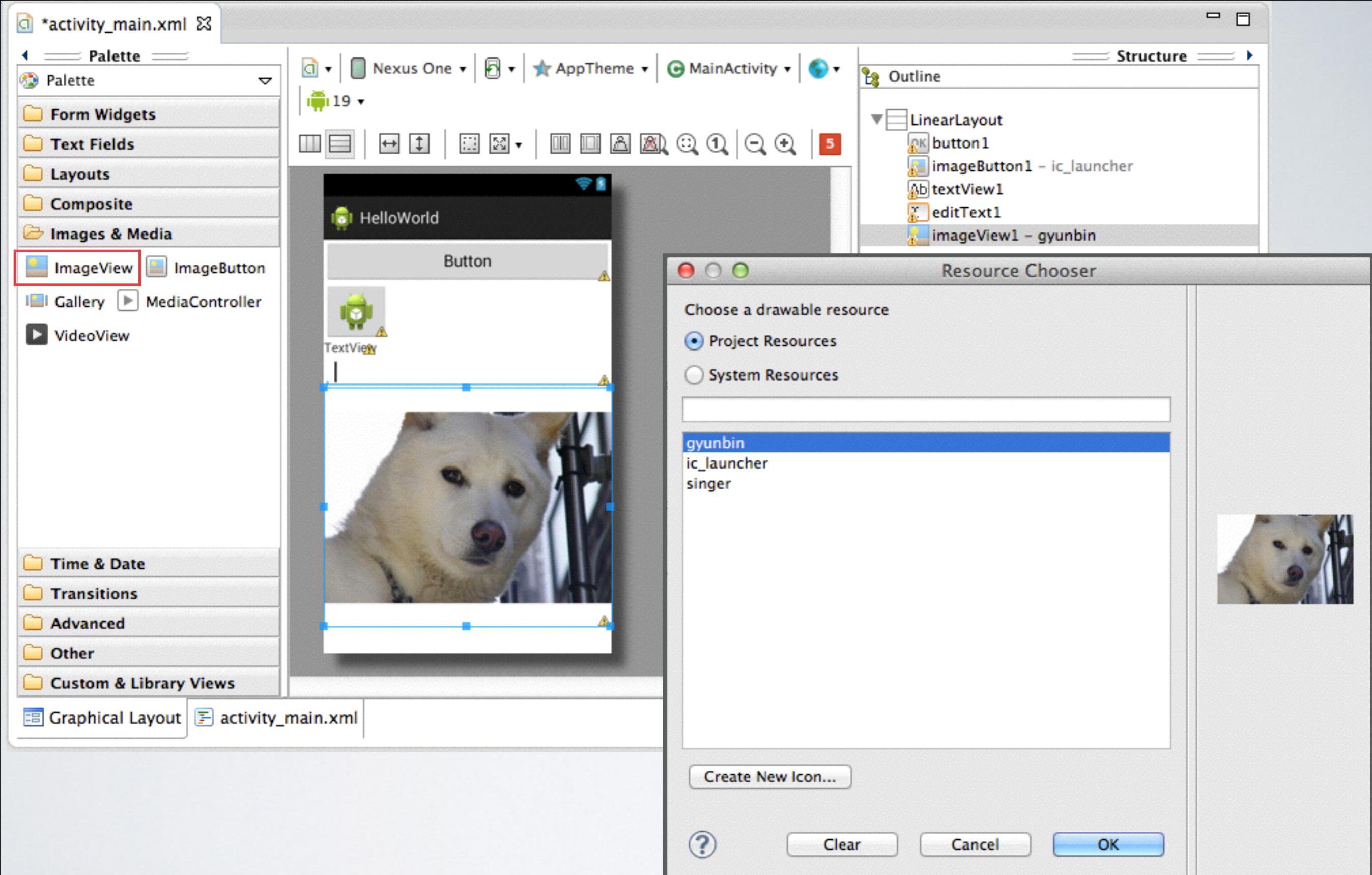
버튼을 누르면 사진이 바뀌는 앱을 만들어 보겠습니다.



먼저 테스트할 이미지를 프로젝트의 res 폴더 안으로 복사합니다.

다운로드 주소

<http://ui.nhnnext.org/crong/scope/android/files/imageViewTestPhoto.zip>



레이아웃에디터에서 ImageView를 추가합니다.
이미지는 테스트용으로 복사한 이미지를 선택합니다.

```
activity_main.xml MainActivity.java
```

```
public class MainActivity extends Activity implements OnClickListener {  
    private TextView textView1;  
    private EditText editText1;  
    private ImageView imageView1;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button button1 = (Button) findViewById(R.id.button1);  
        button1.setOnClickListener(this);  
  
        ImageButton imageButton1 = (ImageButton) findViewById(R.id.imageButton1);  
        imageButton1.setOnClickListener(this);  
  
        textView1 = (TextView) findViewById(R.id.textView1);  
  
        editText1 = (EditText) findViewById(R.id.editText1);  
  
        imageView1 = (ImageView) findViewById(R.id.imageView1);  
    }  
  
    @Override  
    public void onClick(View v) {  
        switch (v.getId()) {  
        case R.id.button1:  
            Toast.makeText(getApplicationContext(), "버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();  
            textView1.setText( editText1.getText().toString() + "이 입력되었습니다." );  
            break;  
        }  
    }  
}
```

ImageView를 추가하고
레이아웃 Id를 연결해줍니다.

```
activity_main.xml *MainActivity.java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Button button1 = (Button) findViewById(R.id.button1);
    button1.setOnClickListener(this);

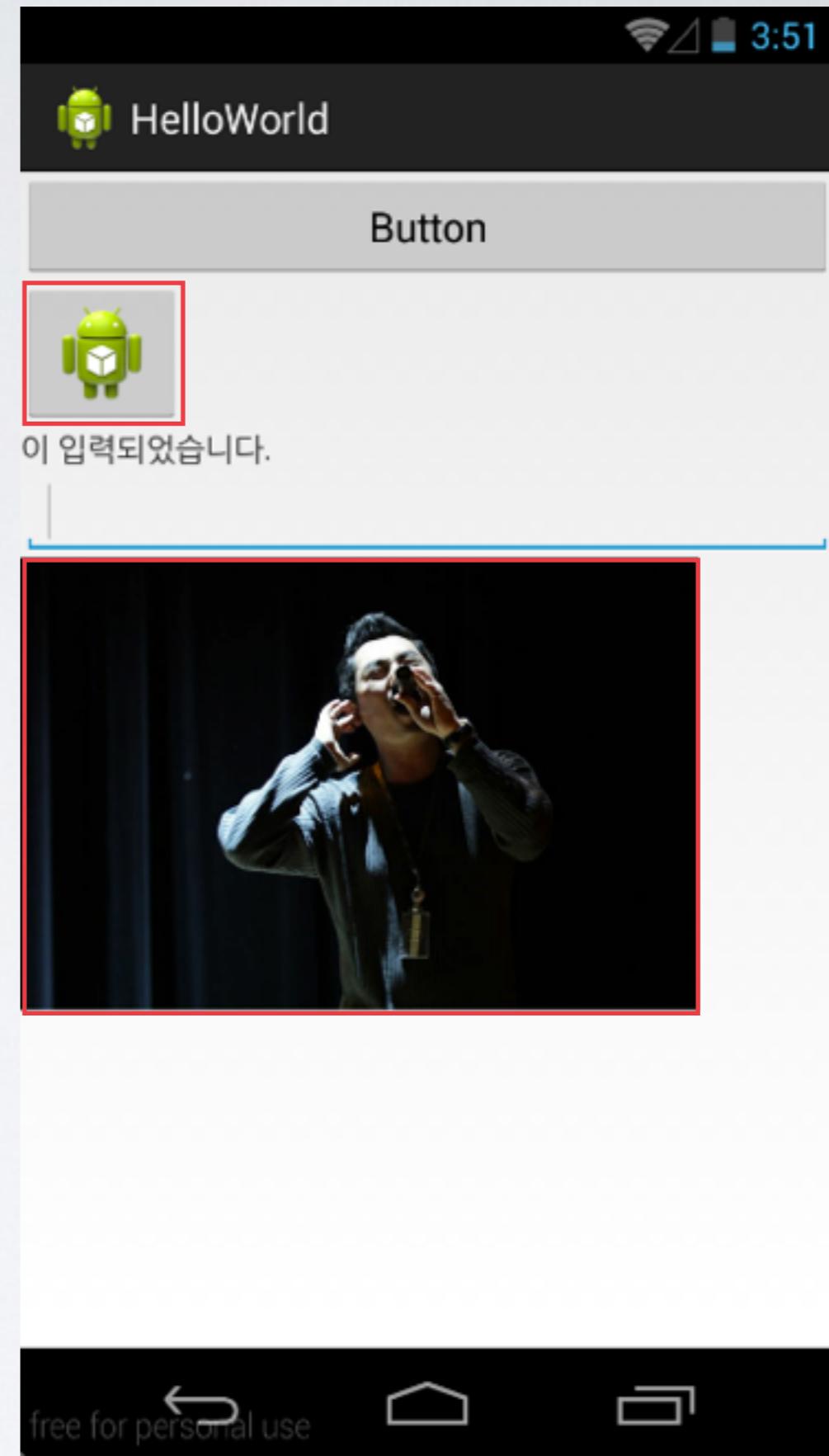
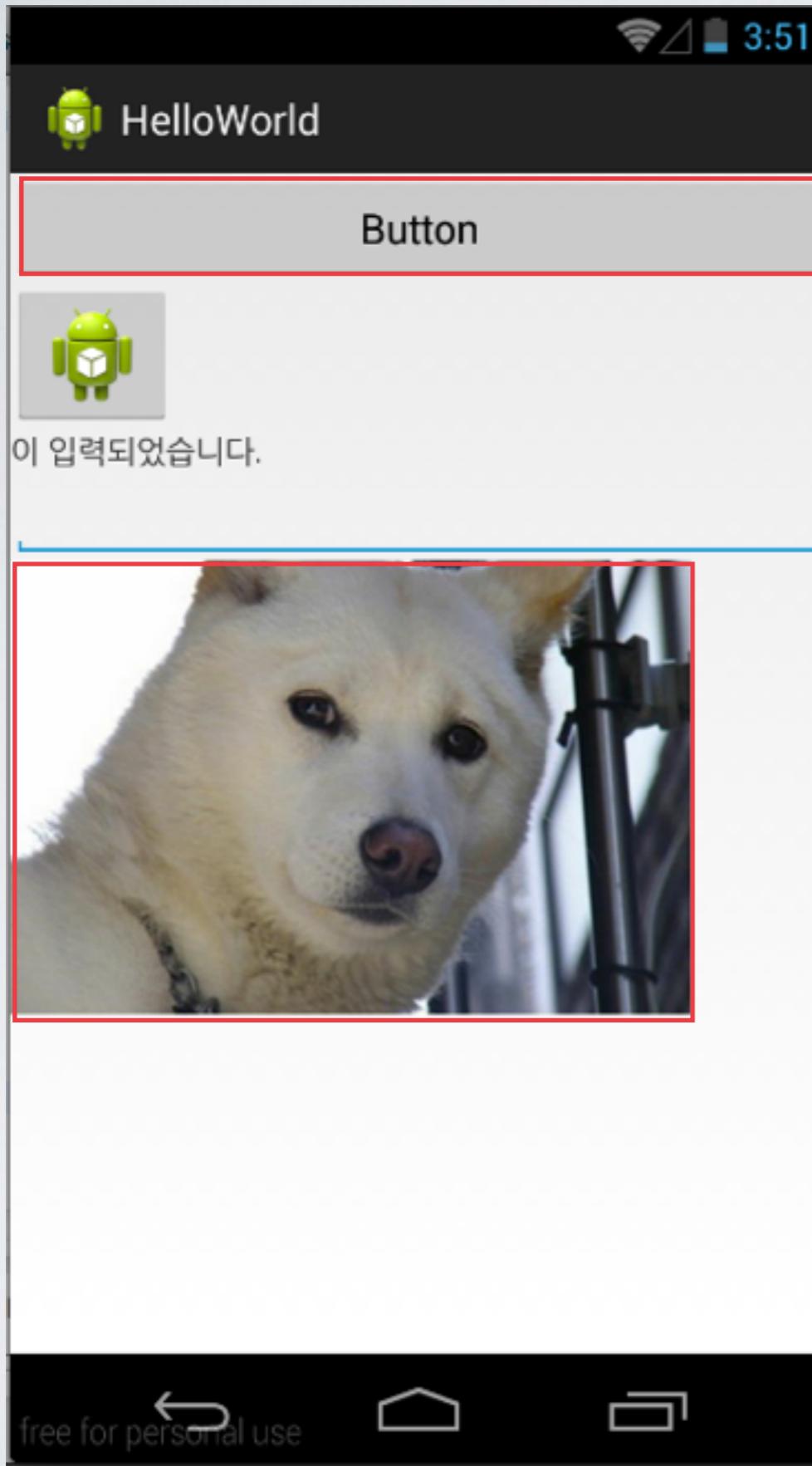
    ImageButton imageButton1 = (ImageButton) findViewById(R.id.imageButton1);
    imageButton1.setOnClickListener(this);

    textView1 = (TextView) findViewById(R.id.textView1);
    editText1 = (EditText) findViewById(R.id.editText1);
    imageView1 = (ImageView) findViewById(R.id.imageView1);

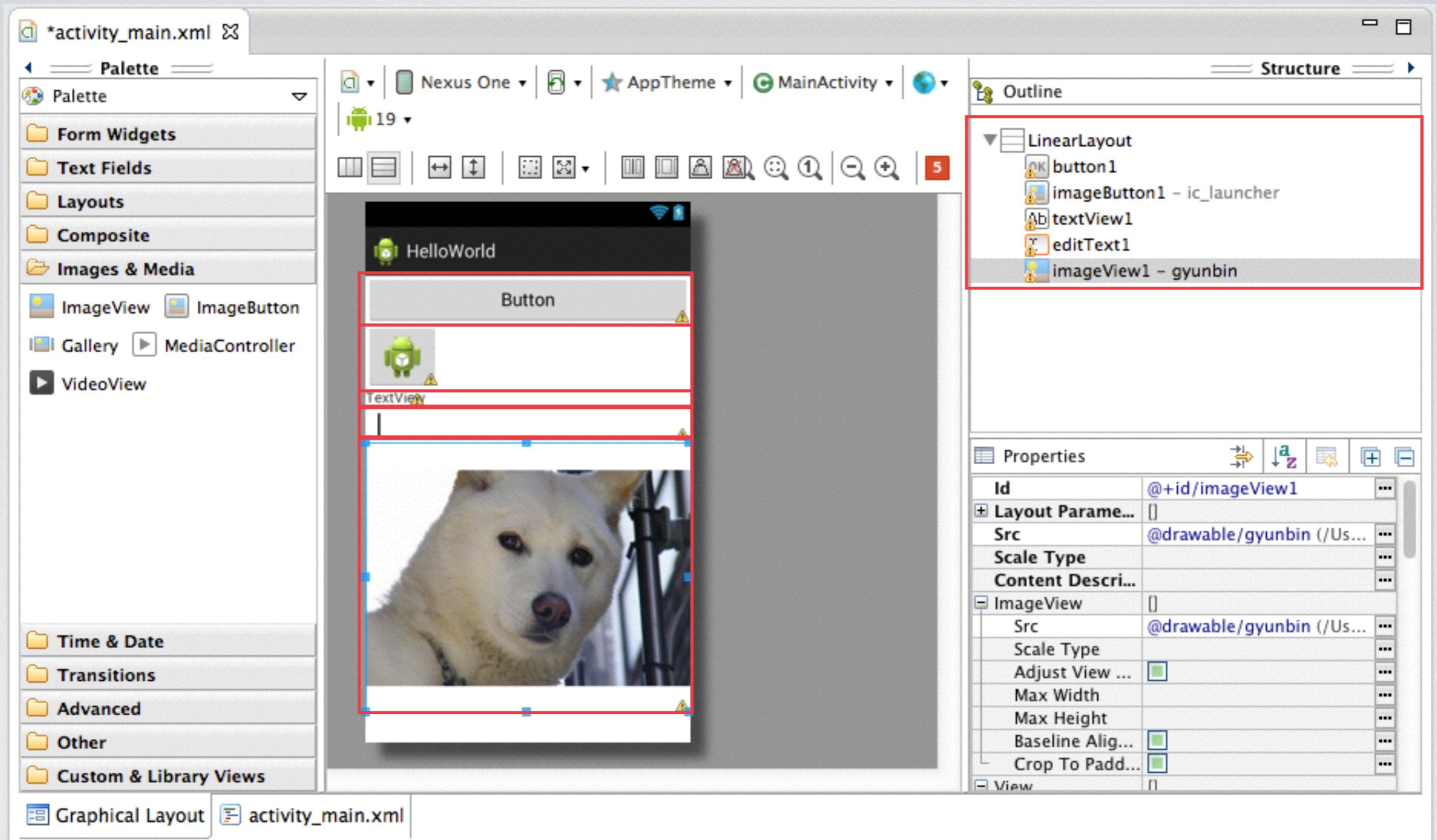
}

@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.button1:
            Toast.makeText(getApplicationContext(), "버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();
            textView1.setText( editText1.getText().toString() + "이 입력되었습니다." );
            imageView1.setImageResource(R.drawable.gyunbin);
            break;
        case R.id.imageButton1:
            Toast.makeText(getApplicationContext(), "이미지버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();
            textView1.setText( editText1.getText().toString() + "이 입력되었습니다." );
            imageView1.setImageResource(R.drawable.singer);
            break;
    }
}
}
```

setImageResource()를 사용하여
R.drawable에 있는 (drawable폴더에 있는) 이미지를 표시합니다.



리니어 레이아웃(LinearLayout)



지금까지 `LinearLayout`을 사용해서
위젯을 세로로 하나씩 배치를 하였습니다.

레이아웃이란?



레이아웃

화면 크기, 해상도(세로, 가로)의 변화에 맞추어 화면을 구성하는 것.

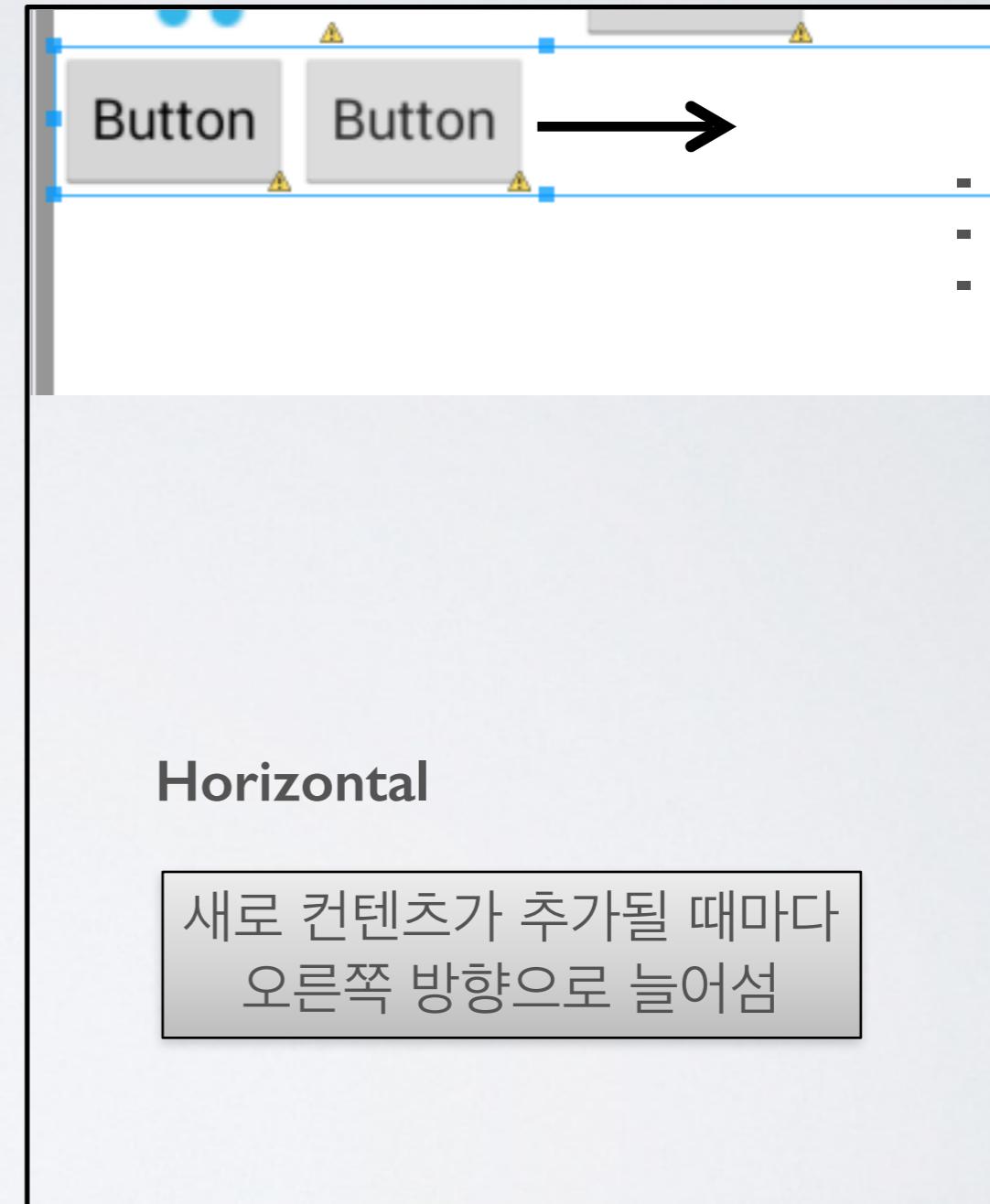
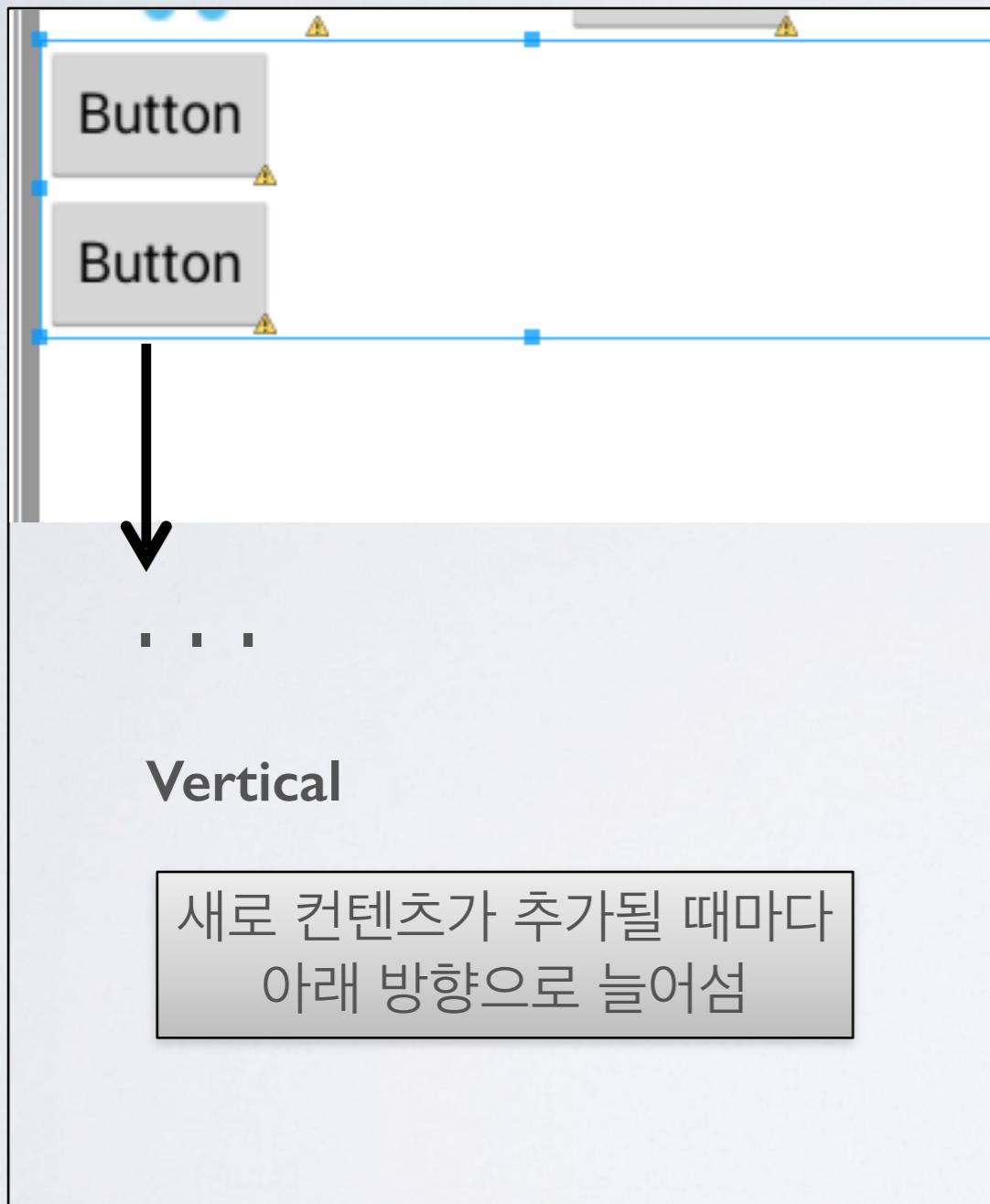


LinearLayout

Orientation

LinearLayout

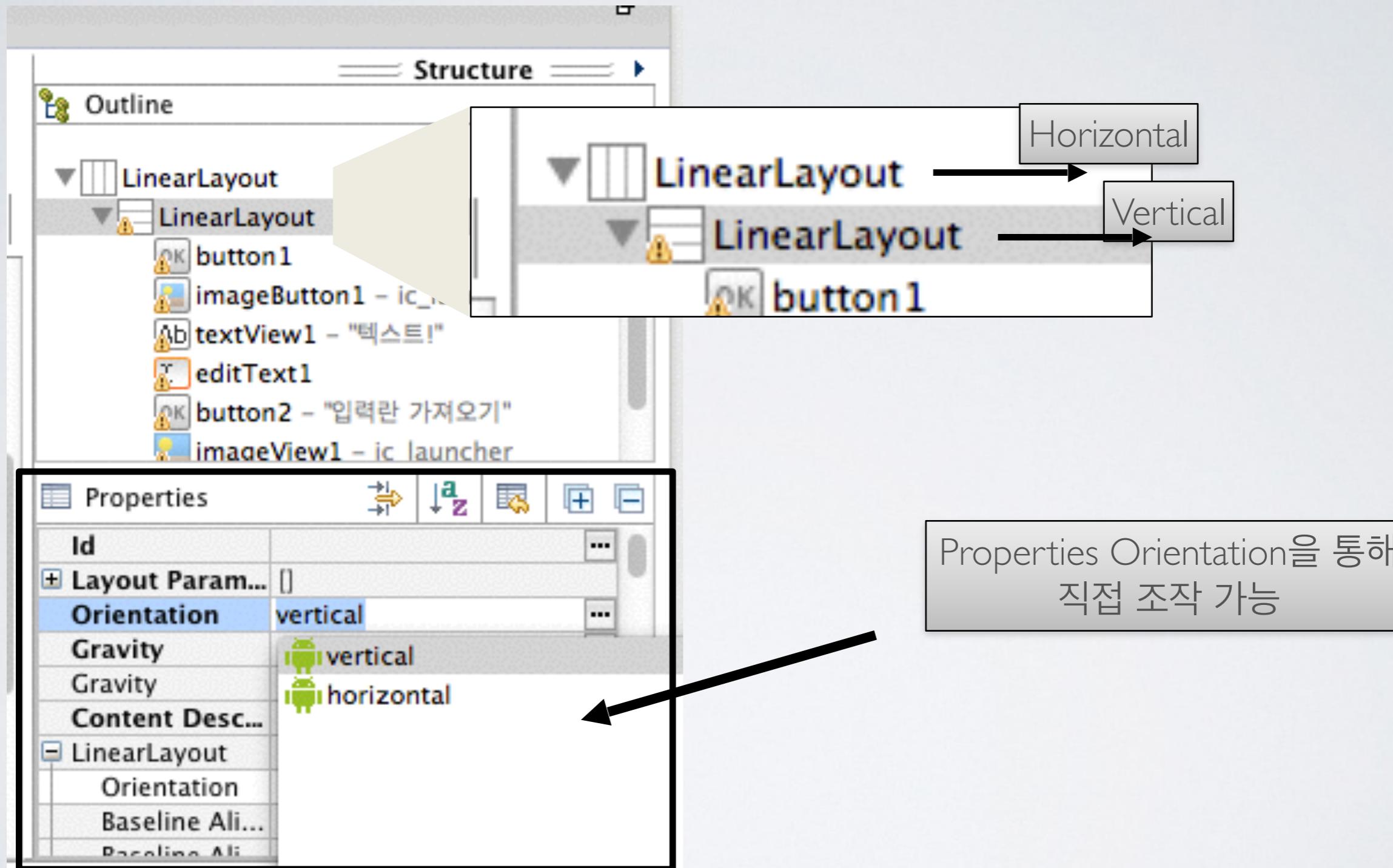
Orientation



LinearLayout은 2가지 Orientation(방향)을 이용해 세로, 가로 방향으로 레이아웃을 구성합니다.

LinearLayout

Orientation



LinearLayout

Orientation

android:orientation="horizontal" or "vertical"
뷰들을 수직으로(vertical) 배치하느냐,
수평으로(horizontal) 배치하느냐를 속성으로 줌.

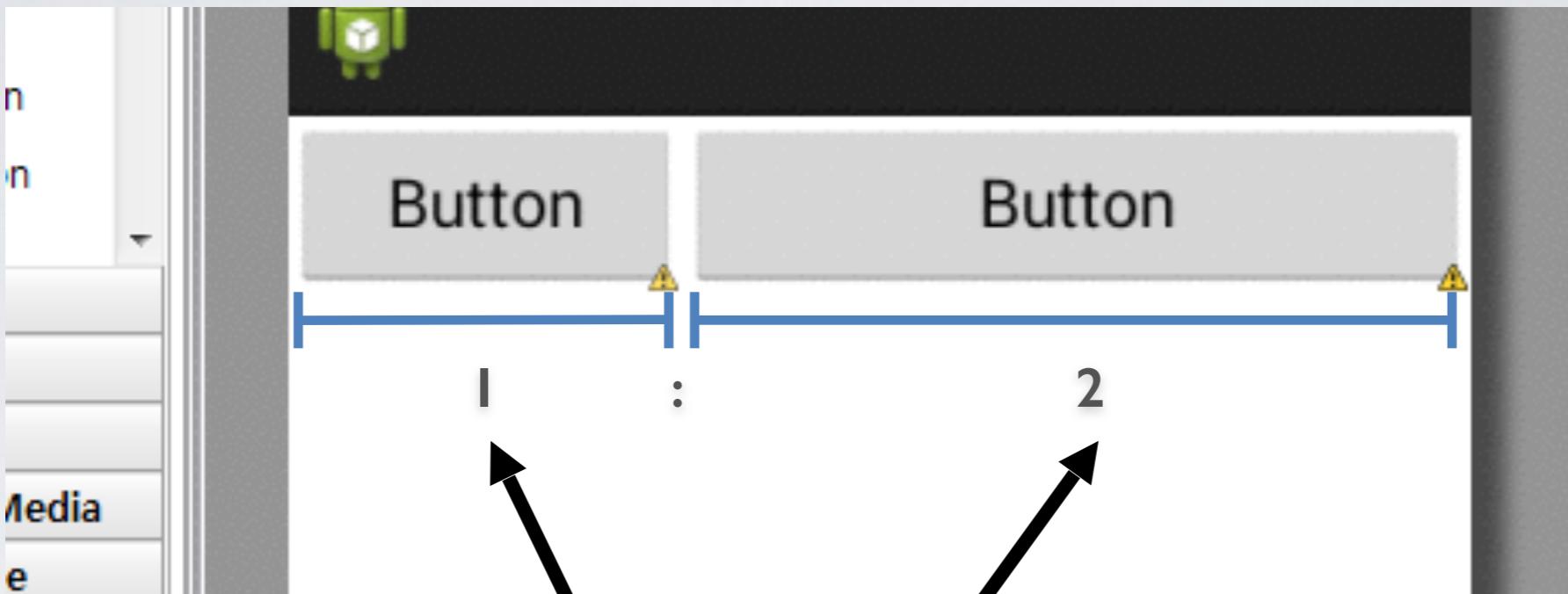
```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:orientation="vertical" >  
        "horizontal"  
  
    <Button  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Button" />  
  
    <Button  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Button" />  
    </LinearLayout>  
</LinearLayout>
```

LinearLayout

Weight

LinearLayout

weight



android:layout_weight= “1”, “2”, Etc...
뷰들간의 배치 비중을 정해주는 것으로,
서로의 비중을 바탕으로 레이아웃을 배치합니다.

LinearLayout

weight



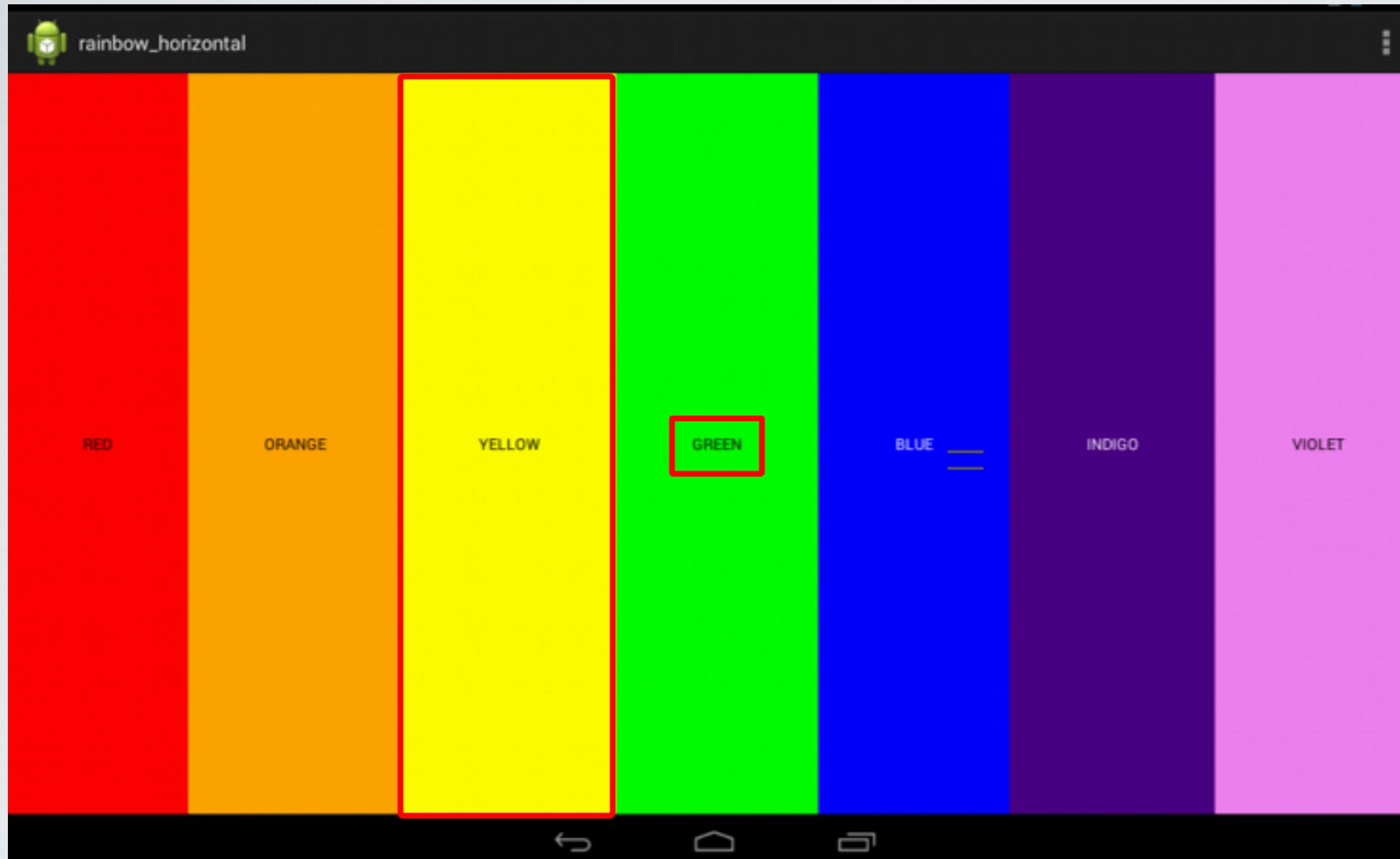
주의!

배치하는 컨텐츠의 길이는
0(dp)으로 설정해주어야 합니다.
(width나 height)

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" >  
  
    <Button  
        android:id="@+id/button1"  
        android:layout_width="0dp"  
        android:layout_height="wrap_content"  
        android:layout_weight="1"  
        android:text="Button" />  
  
    <Button  
        android:id="@+id/button2"  
        android:layout_width="0dp"  
        android:layout_height="wrap_content"  
        android:layout_weight="2"  
        android:text="Button" />  
  
</LinearLayout>
```

LinearLayout

weight



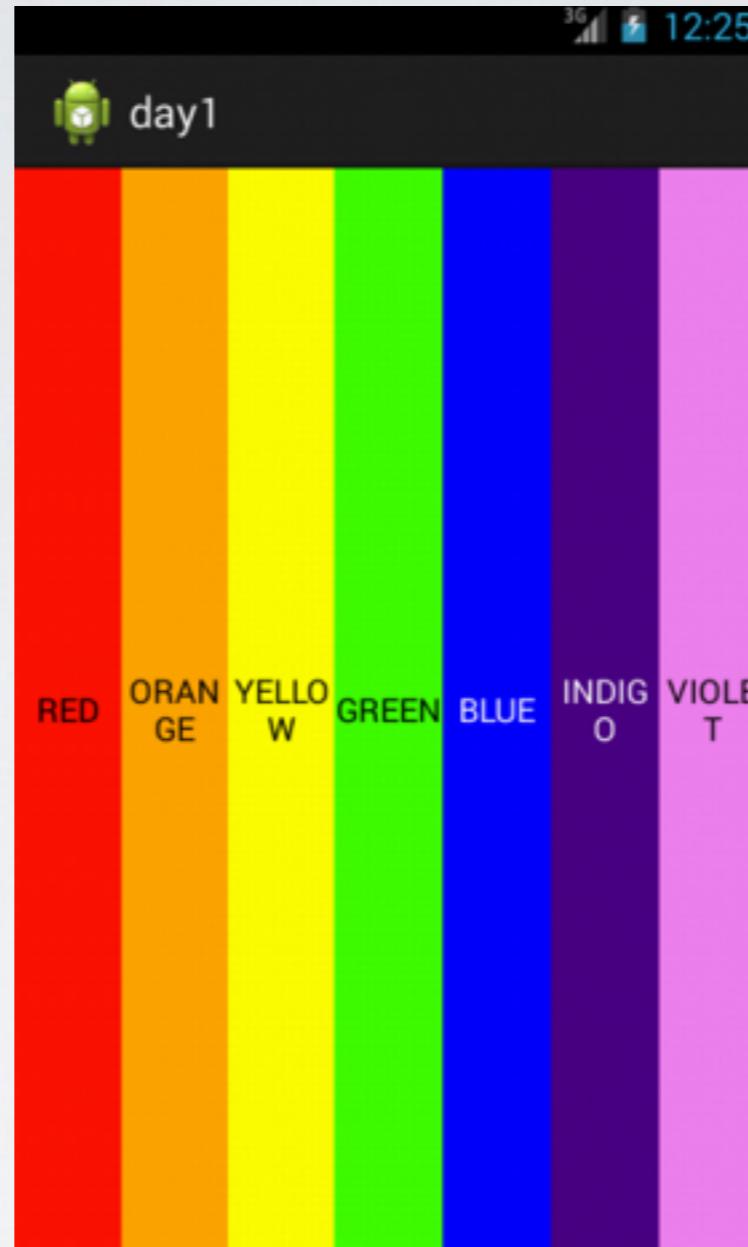
Width - wrap_content
Height - match_parent
Weight - 1로 주고
7개의 컨텐츠를
위와 같이 설정하고
텍스트를 넣었습니다.

화면의 크기가 충분히 크면 상관이 없이 잘 보이지만...



화면의 크기가 컨텐츠가 모두 들어가지 못할 정도로 작을 경우...
(Width, Height, Weight는 앞과 동일)

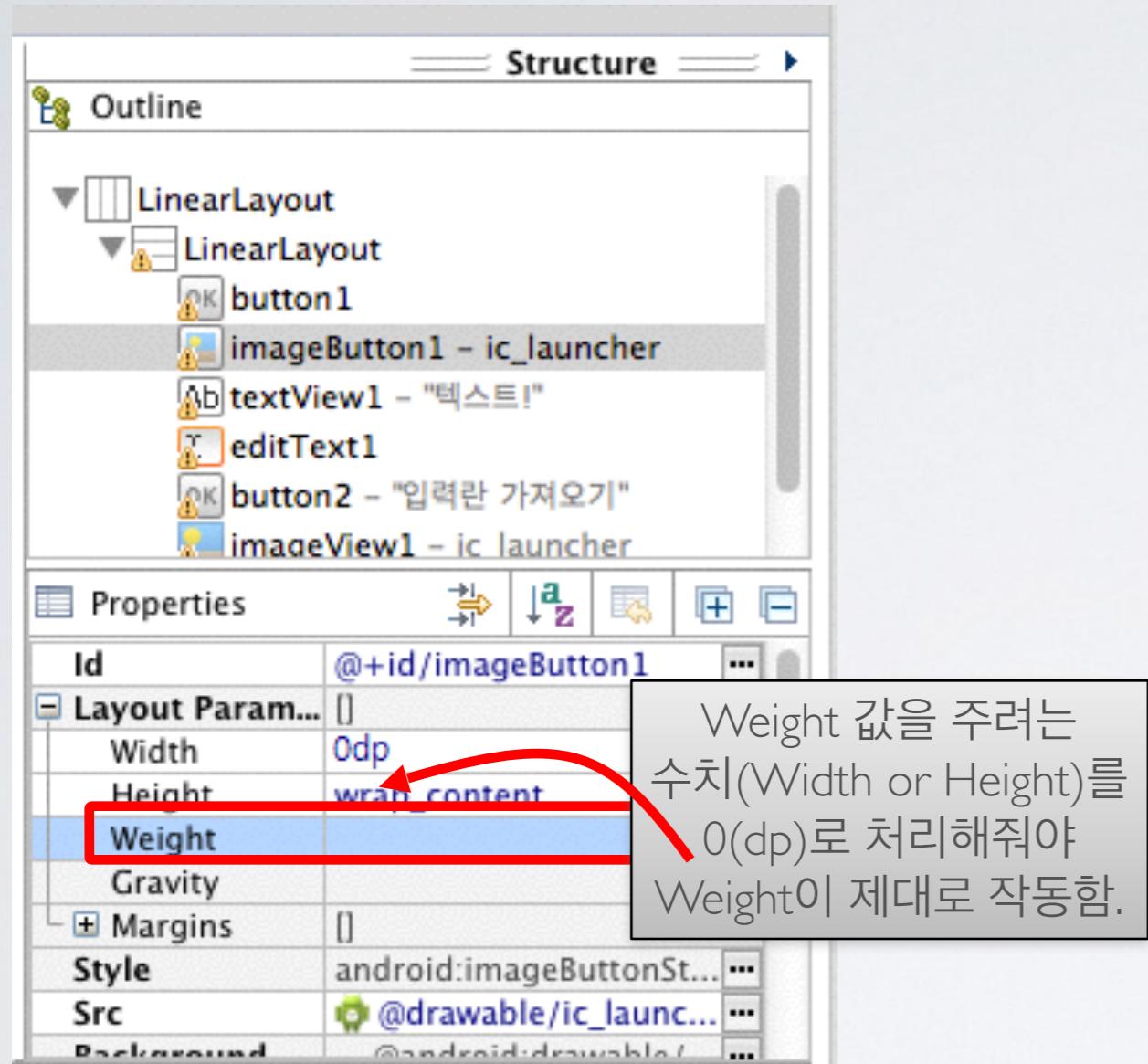
width나 height의 wrap_content를 먼저 적용한 후,
weight값으로 레이아웃 크기를 조정되기 때문에
컨텐츠 크기에 따라 레이아웃이 구성됩니다.



하지만 width나 height를 0dp로 설정을 한 후에
weight를 적용하면
컨텐츠의 크기와 상관없이 정확한 비중으로
레이아웃을 배치합니다.

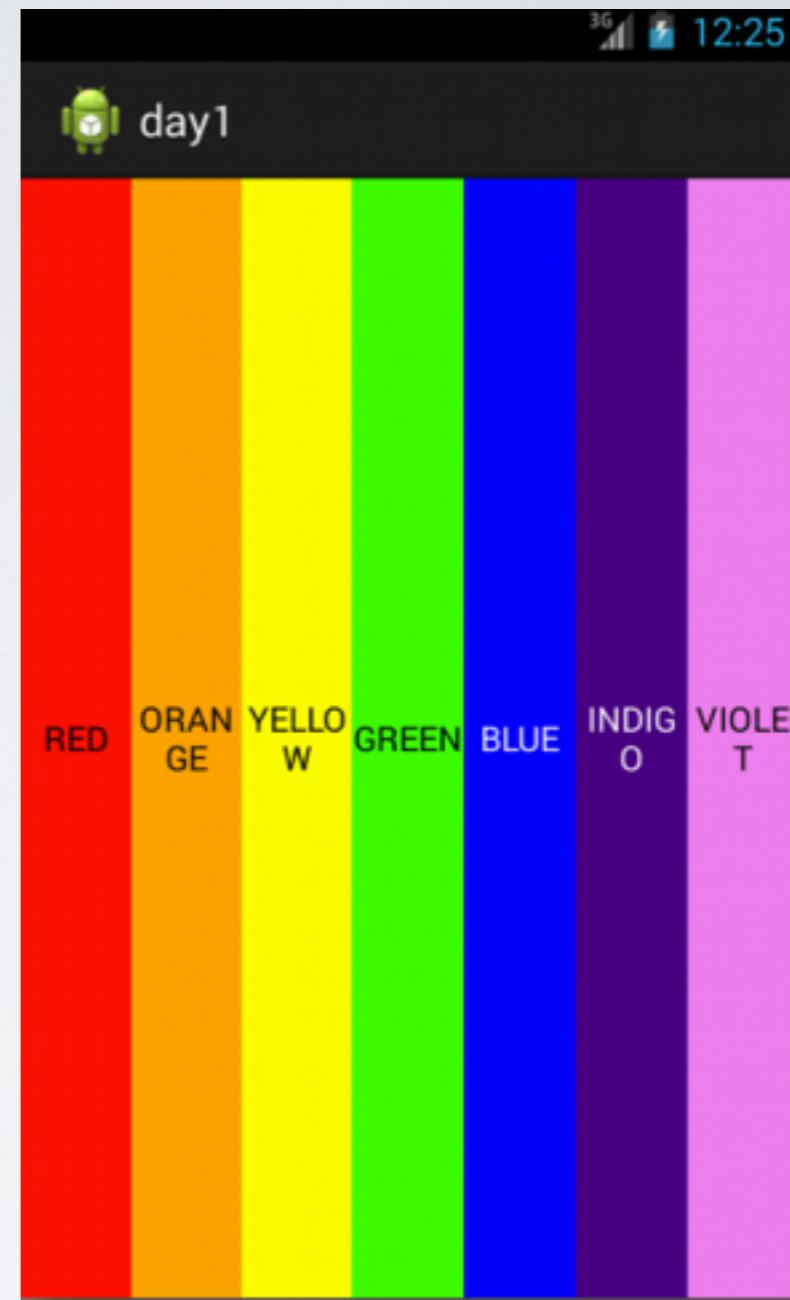
(위 예제는 width를 0dp로 적용하였습니다.)

weight 정리



```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" >  
  
<Button  
    android:id="@+id/button1"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="Button" />  
  
<Button  
    android:id="@+id/button2"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_weight="2"  
    android:text="Button" />  
  
</LinearLayout>
```

width나 height를 0dp로 주고
weight을 통해 비중을 입력해 주면...



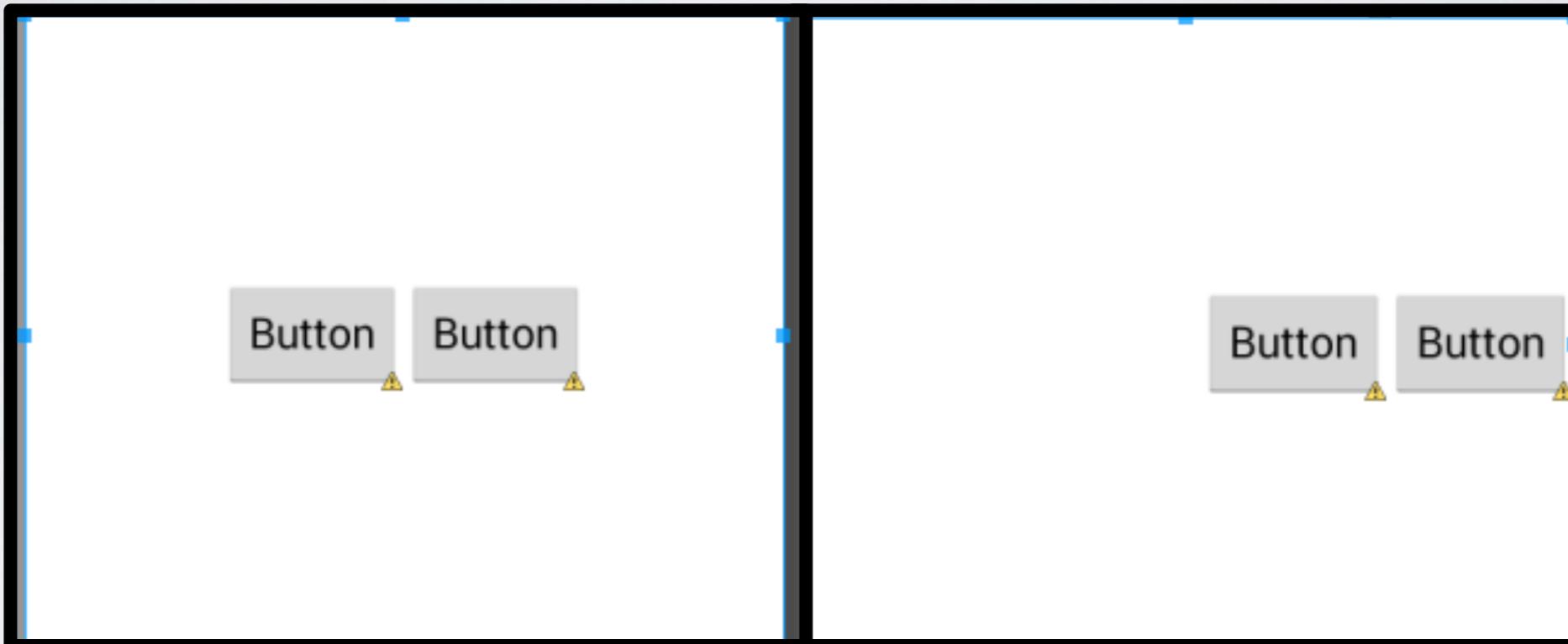
컨텐츠의 크기와 관계없이
정확한 비중을 가지는 레이아웃을 만들 수 있습니다.

LinearLayout

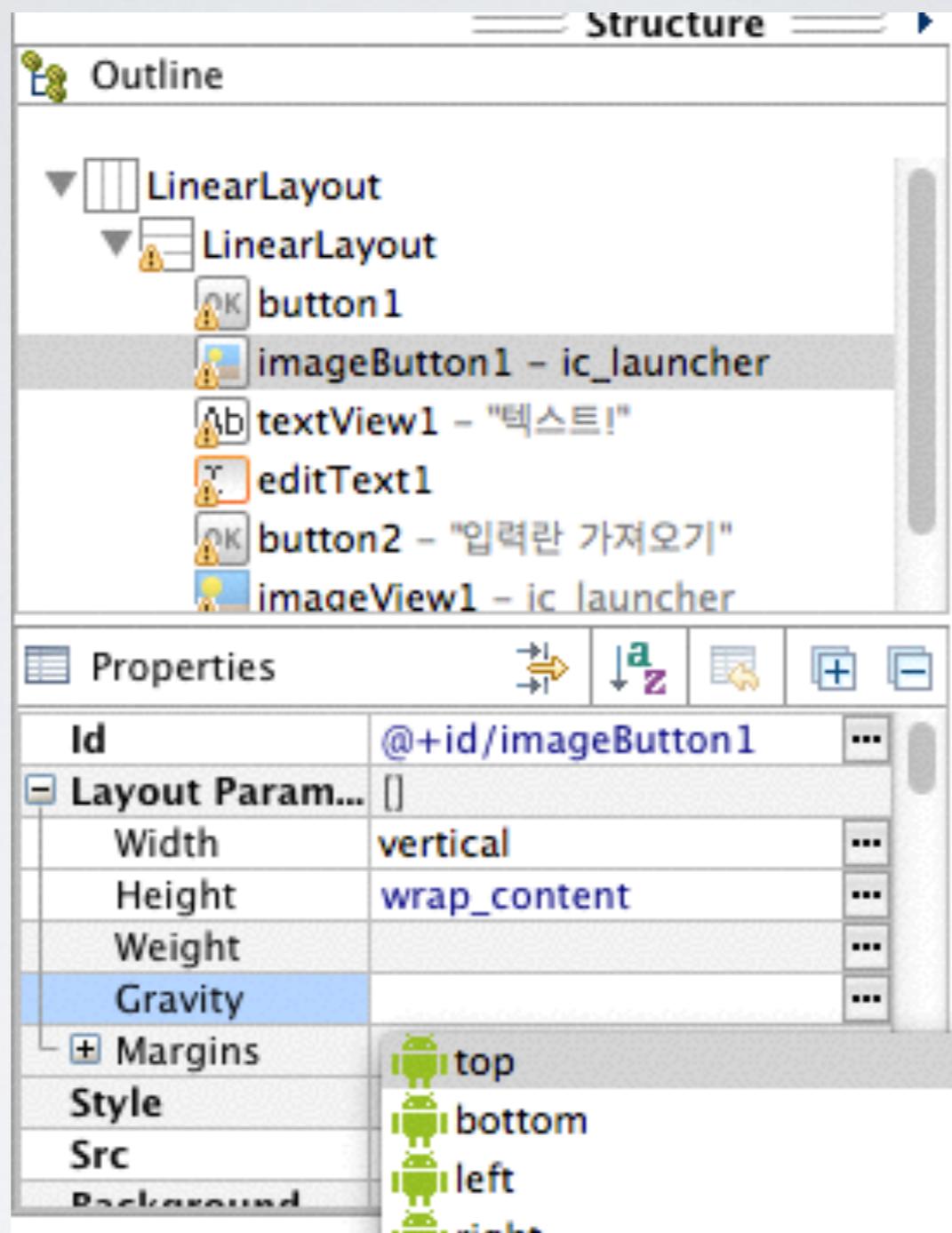
Gravity

LinearLayout

gravity



gravity를 사용하여 레이아웃 안에서
컨텐츠들의 기본 위치를 설정해줄 수 있습니다.

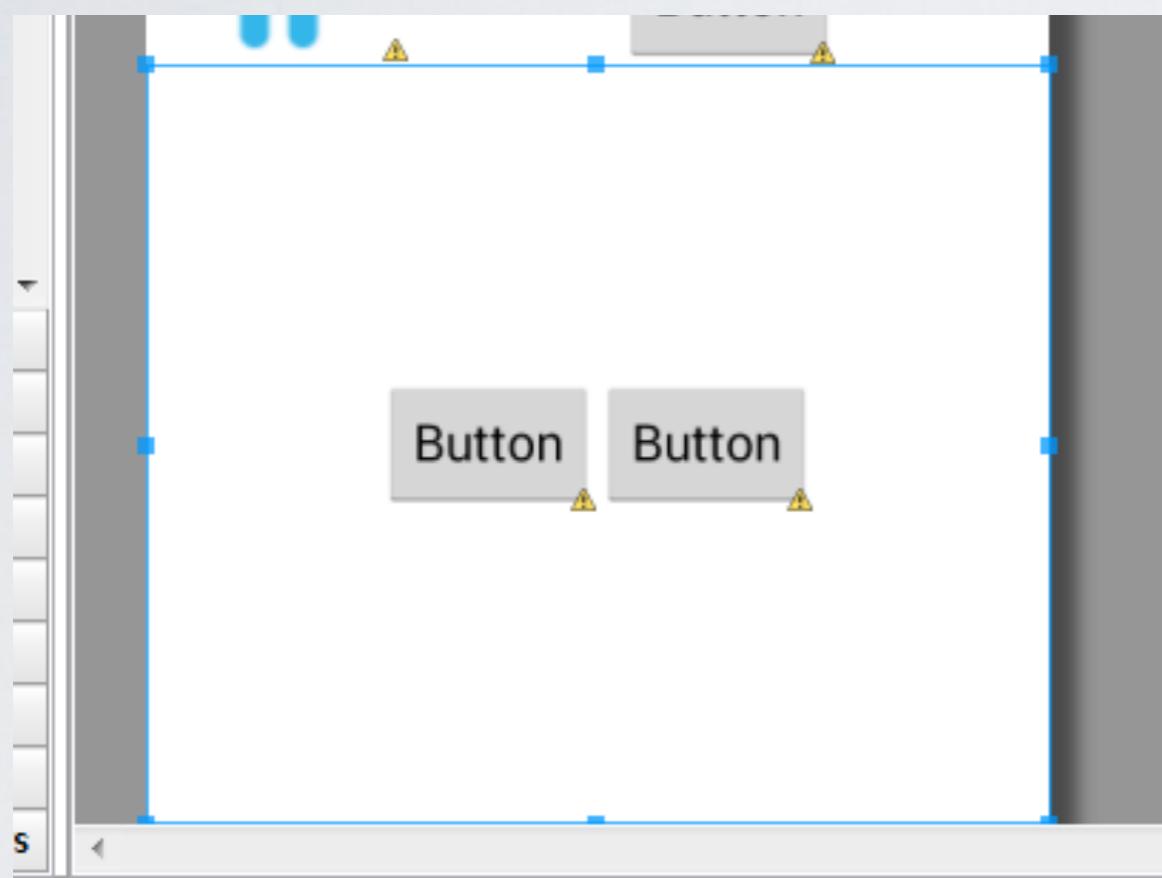


레이아웃 에디터의
Properties에서 컨텐츠 Gravity 설정하거나...

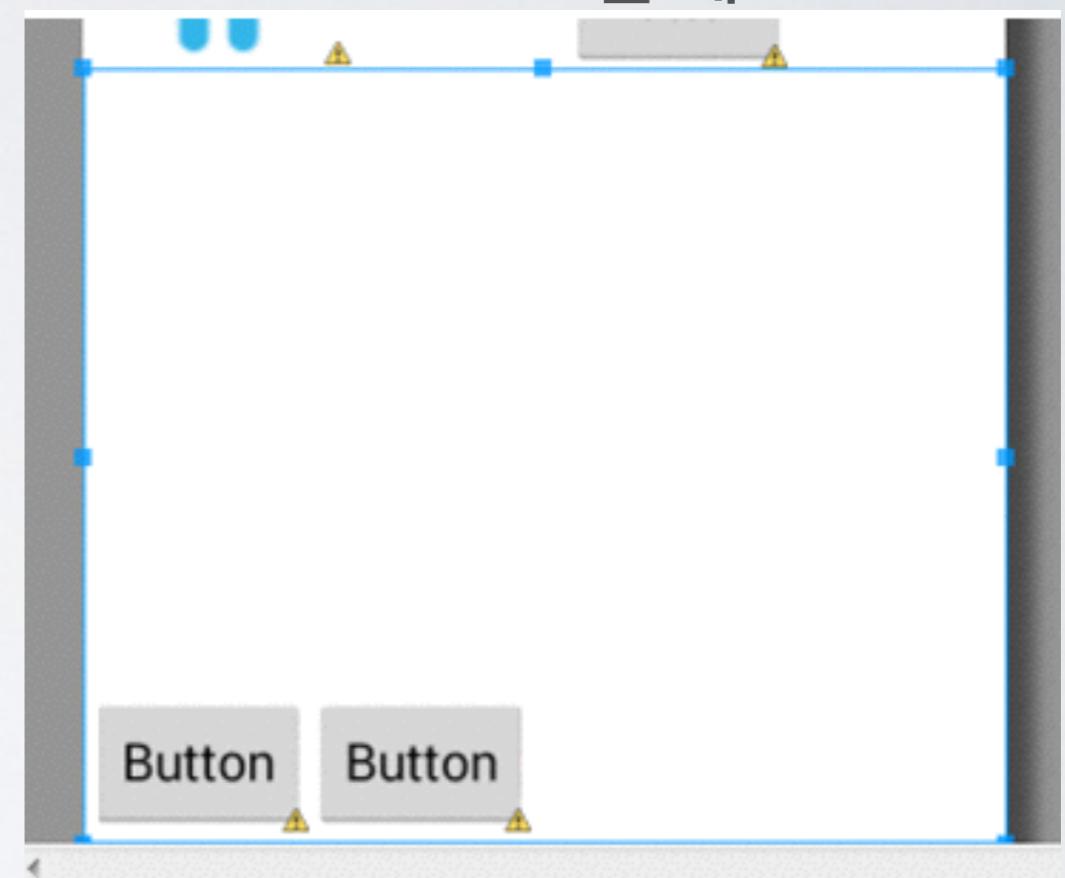
```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:gravity="center"  
    android:orientation="horizontal" >  
  
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Button" />  
  
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Button" />  
/</LinearLayout>  
</LinearLayout>
```

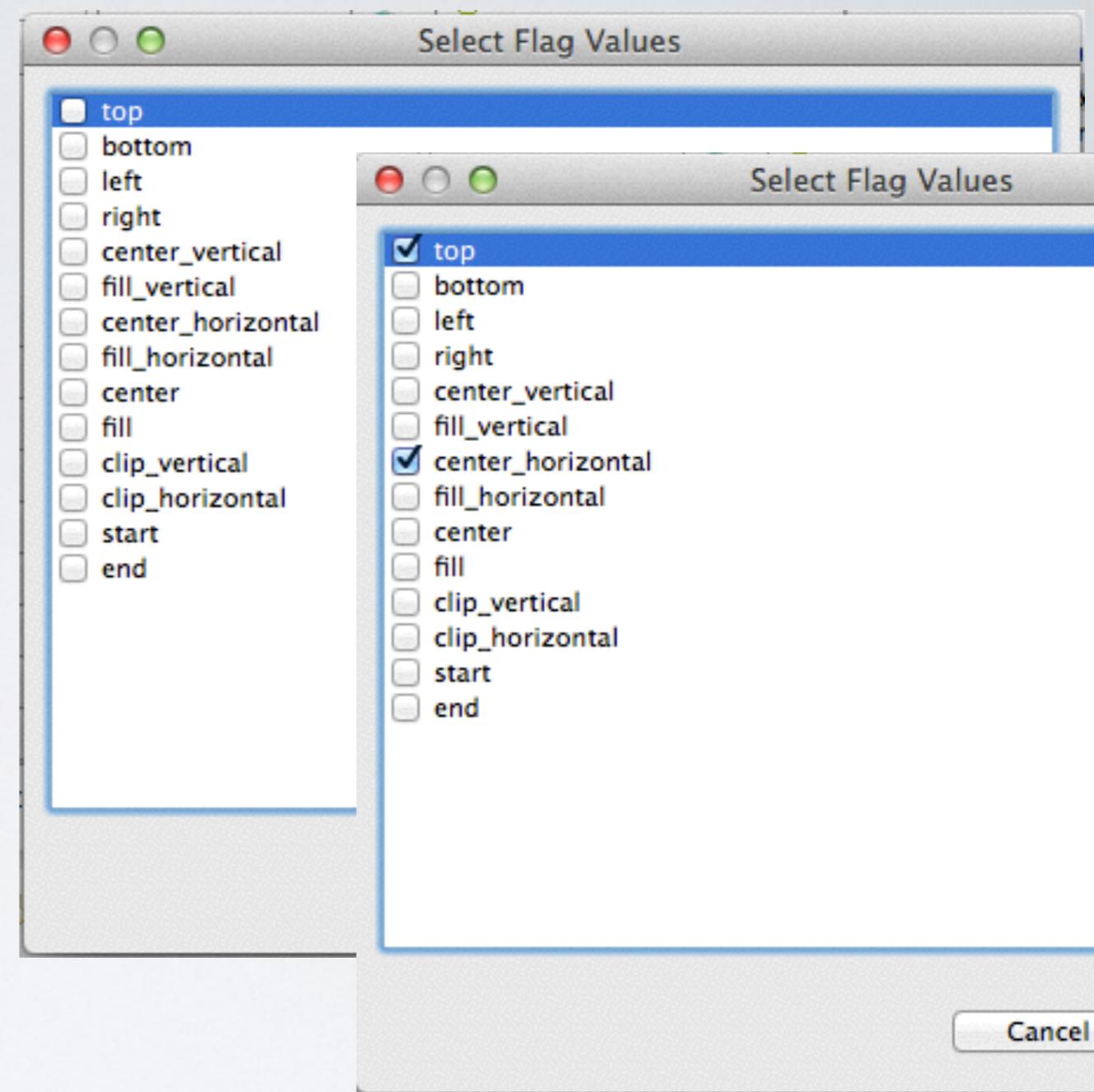
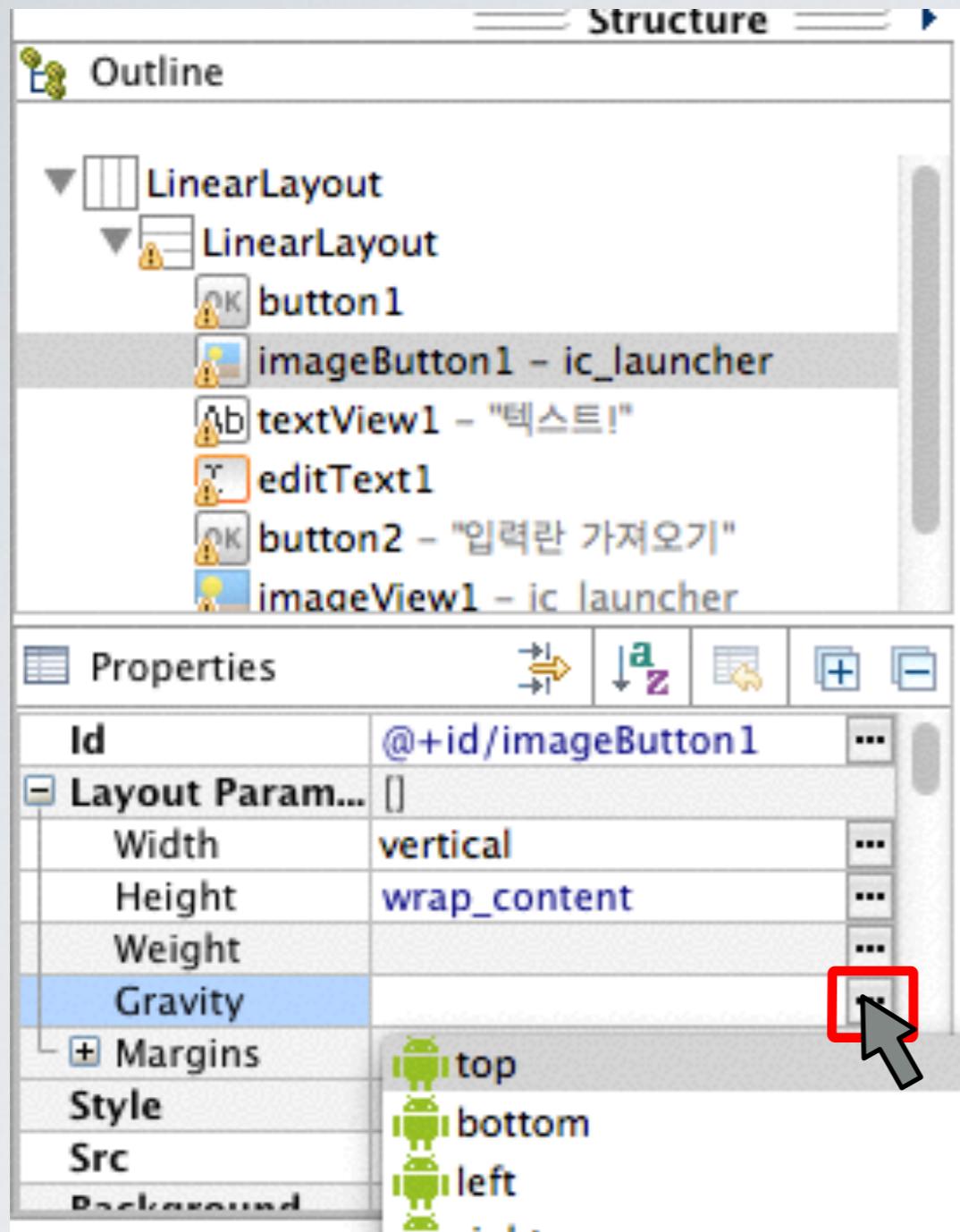
xml에서 android:gravity로 설정 할 수 있습니다.

center일 때



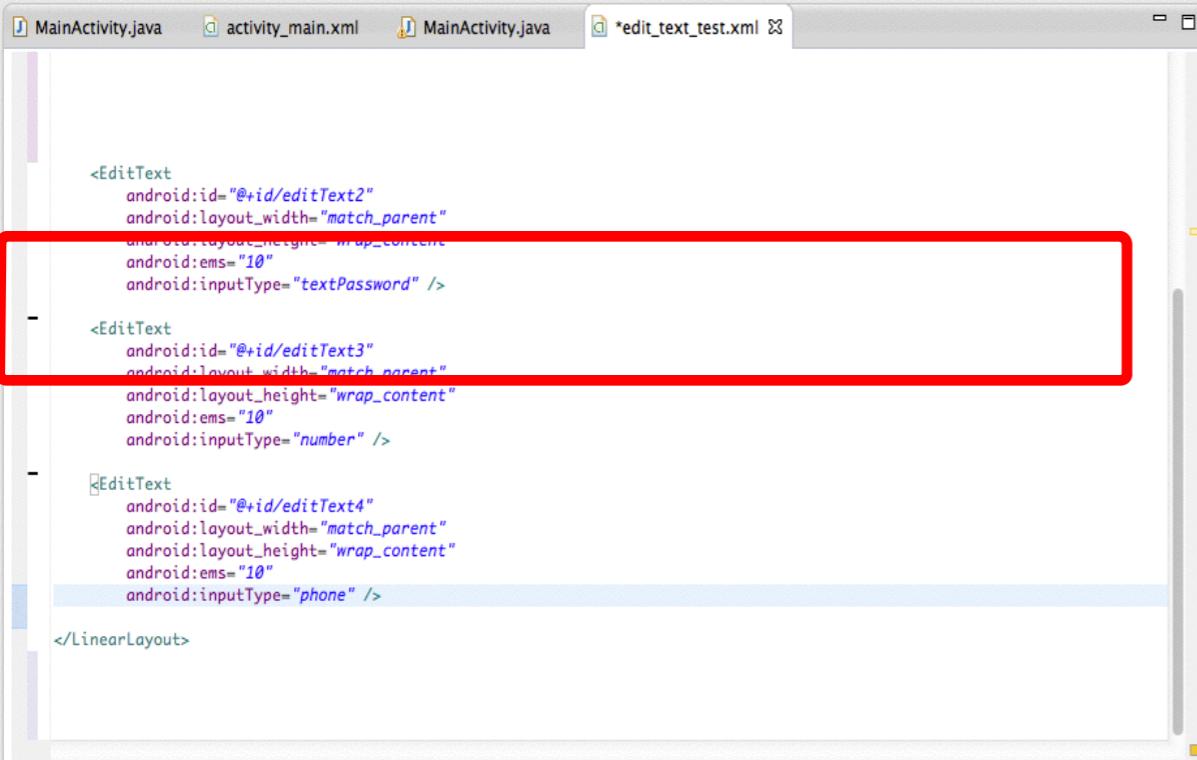
bottom일 때



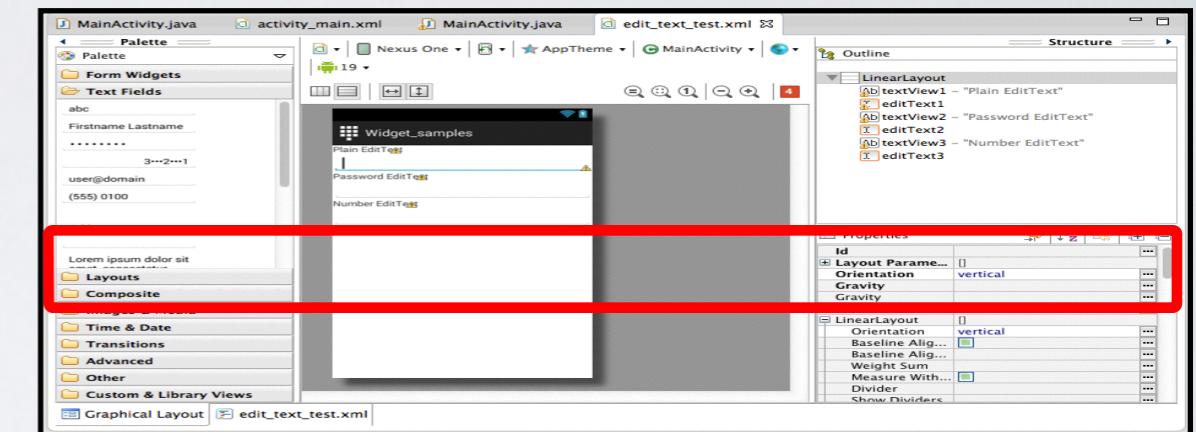


gravity는 여러 조건을 동시에 사용 할 수 있습니다.

레이아웃 에디터의
Properties에서 여러 조건으로 Gravity 설정하거나...

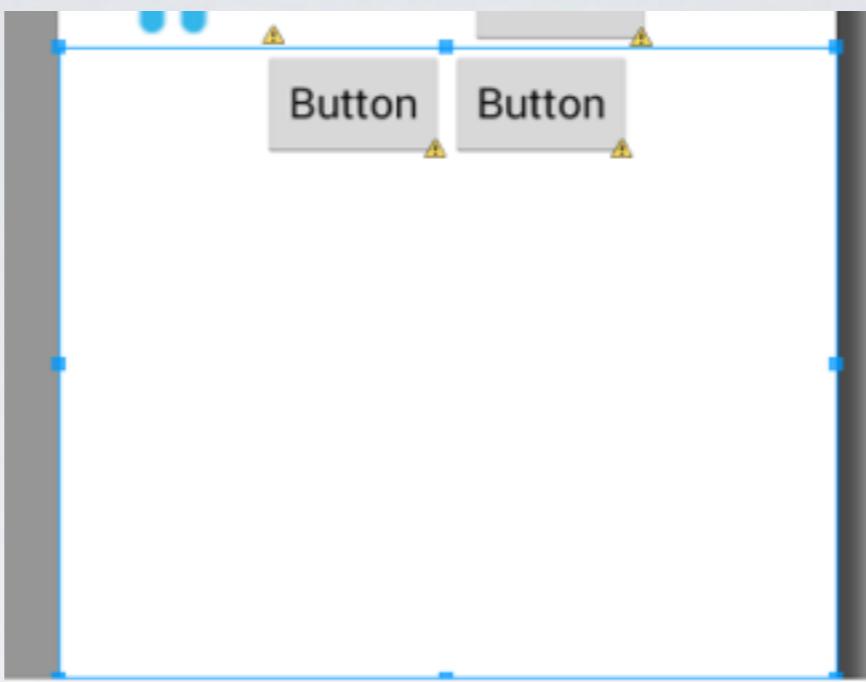


```
<EditText  
    android:id="@+id/editText2"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:ems="10"  
    android:inputType="textPassword" />  
  
<EditText  
    android:id="@+id/editText3"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:ems="10"  
    android:inputType="number" />  
  
<EditText  
    android:id="@+id/editText4"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:ems="10"  
    android:inputType="phone" />  
  
</LinearLayout>
```

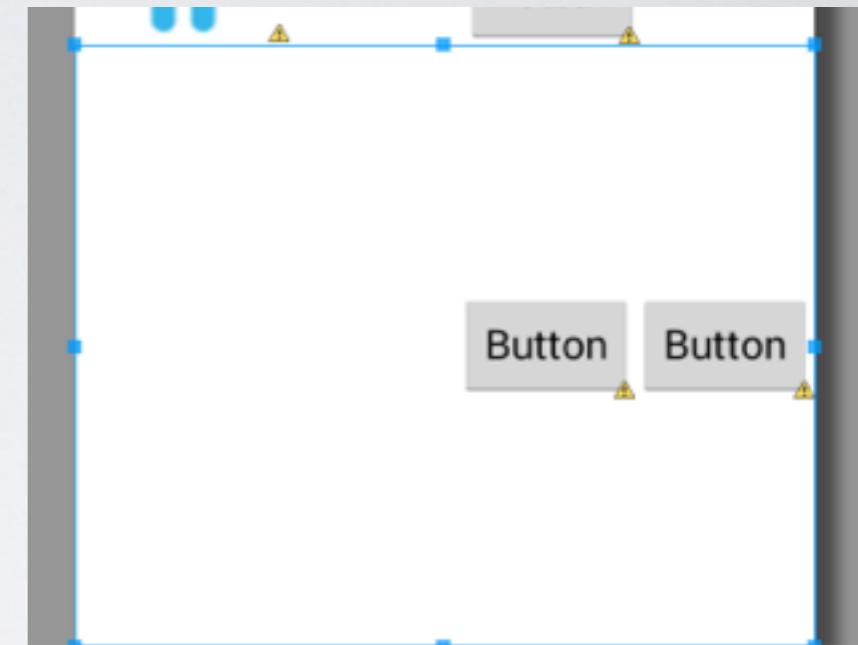


xml에서 ']'를 이용해 여러가지 속성을 사용 할 수 있습니다.

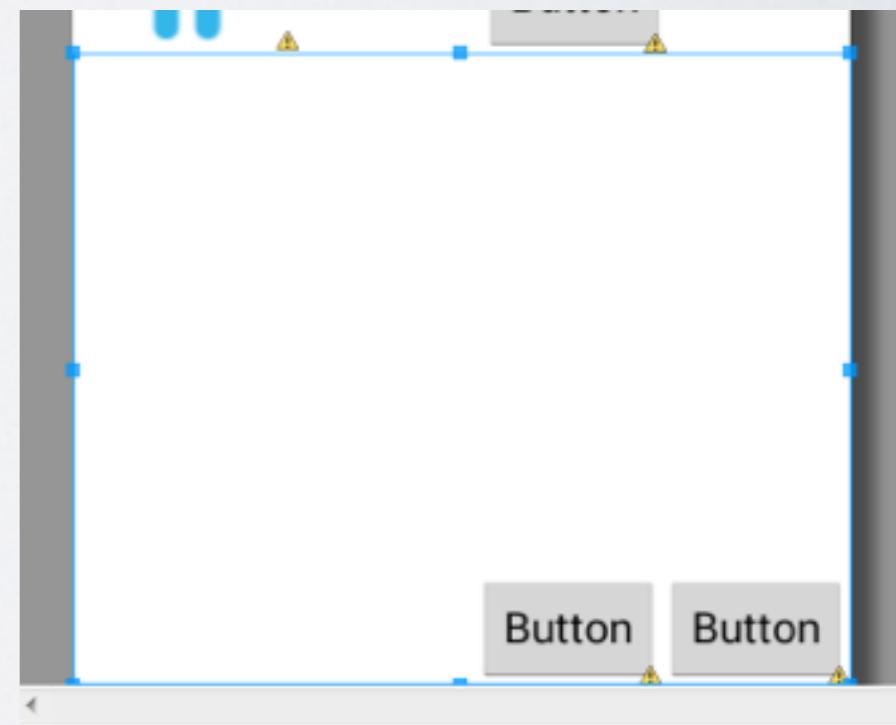
center_horizontal | top일 때



center_vertical | right일 때

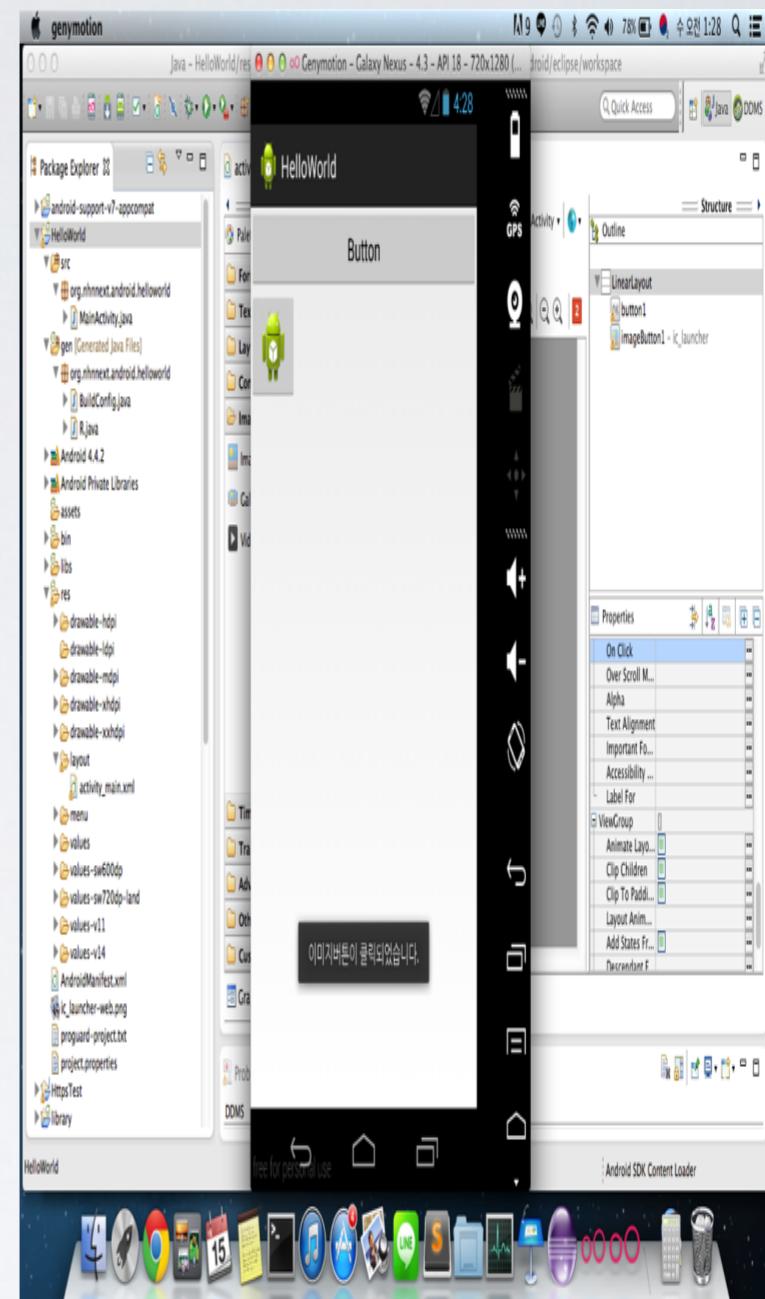
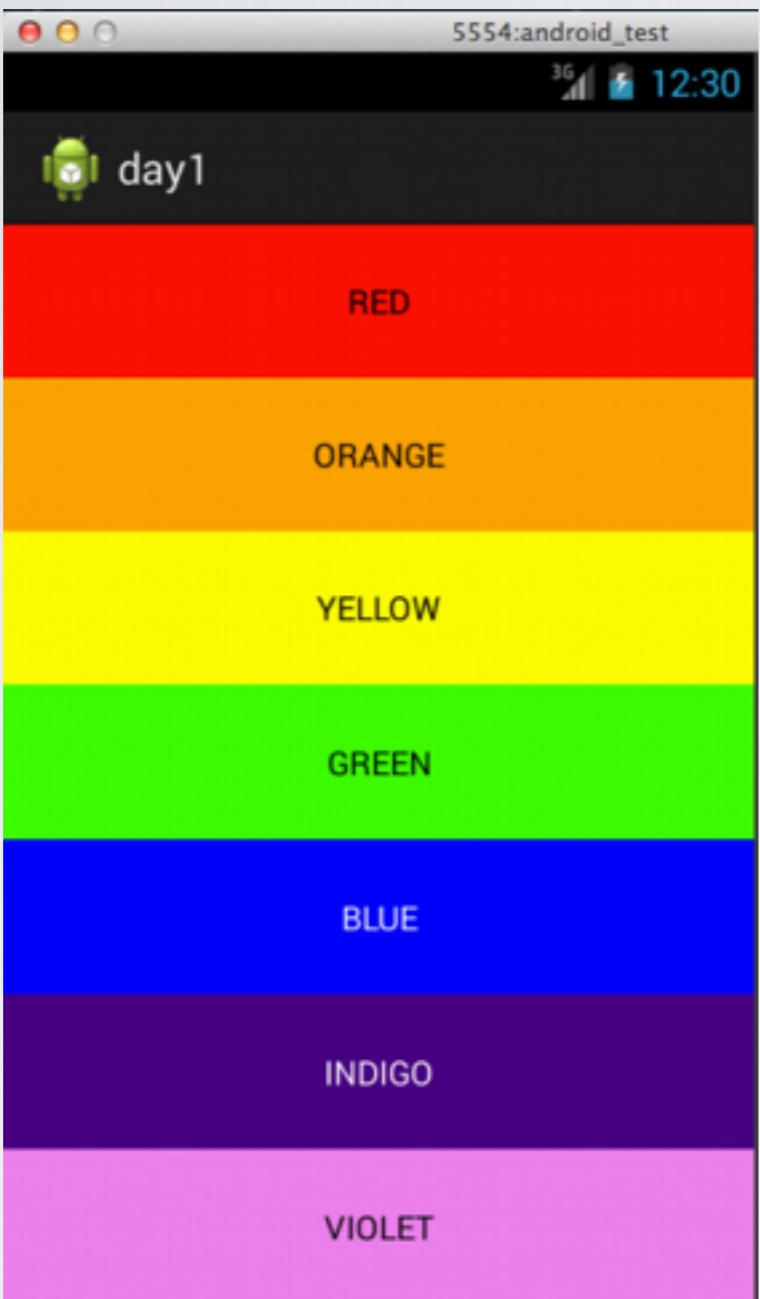


bottom | right일 때



LinearLayout

연습하기



LinearLayout을 사용하여
위와 같은 레이아웃을 만들어보겠습니다.

The screenshot shows the Android Studio code editor with the tab bar at the top containing 'activity_main.xml', '*MainActivity.java' (which is the active tab), and 'R.java'. The code in the editor is as follows:

```
package org.nhnnext.android.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

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

    }

}
```

연습용 프로젝트를 생성하고, layout 폴더의
메인 액티비티 xml 파일을 열어줍니다.

The screenshot shows the Android Studio interface. The top tab bar has two tabs: `*activity_main.xml` (selected) and `MainActivity.java`. The main content area displays the XML code for `activity_main.xml`:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context=".MainActivity" >  
  
    <TextView  
        android:id="@+id/textView1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/hello_world" />  
  
    </RelativeLayout>
```

The XML code defines a `RelativeLayout` with various padding and margin settings. Inside it, there is a `TextView` with the ID `@+id/textView1`, text `@string/hello_world`, and dimensions set to `wrap_content`.

먼저 xml 편집으로 들어가
기존에 있는 레이아웃을 삭제합니다.

The screenshot shows the Android Studio interface. The top bar has tabs for 'activity_main.xml' and 'MainActivity.java'. The main area contains two panes: one for the Java code and one for the graphical layout. The Java code pane displays the following code:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Button button1 = (Button) findViewById(R.id.button1);
    button1.setOnClickListener(this);

    ImageButton imageButton1 = (ImageButton) findViewById(R.id.imageButton1);
    imageButton1.setOnClickListener(this);

    textView1 = (TextView) findViewById(R.id.textView1);

    editText1 = (EditText) findViewById(R.id.editText1);

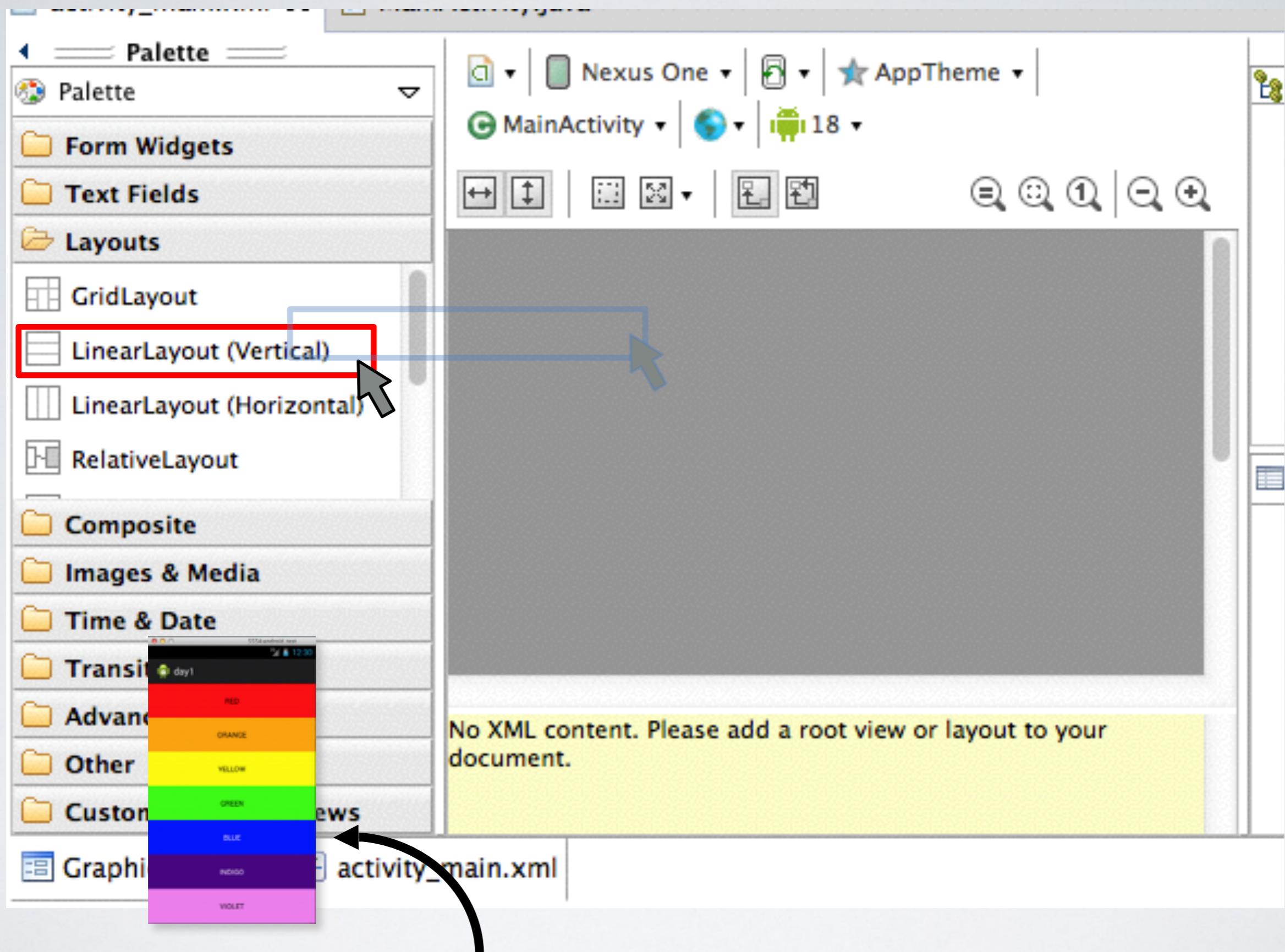
    imageView1 = (ImageView) findViewById(R.id.imageView1);

}

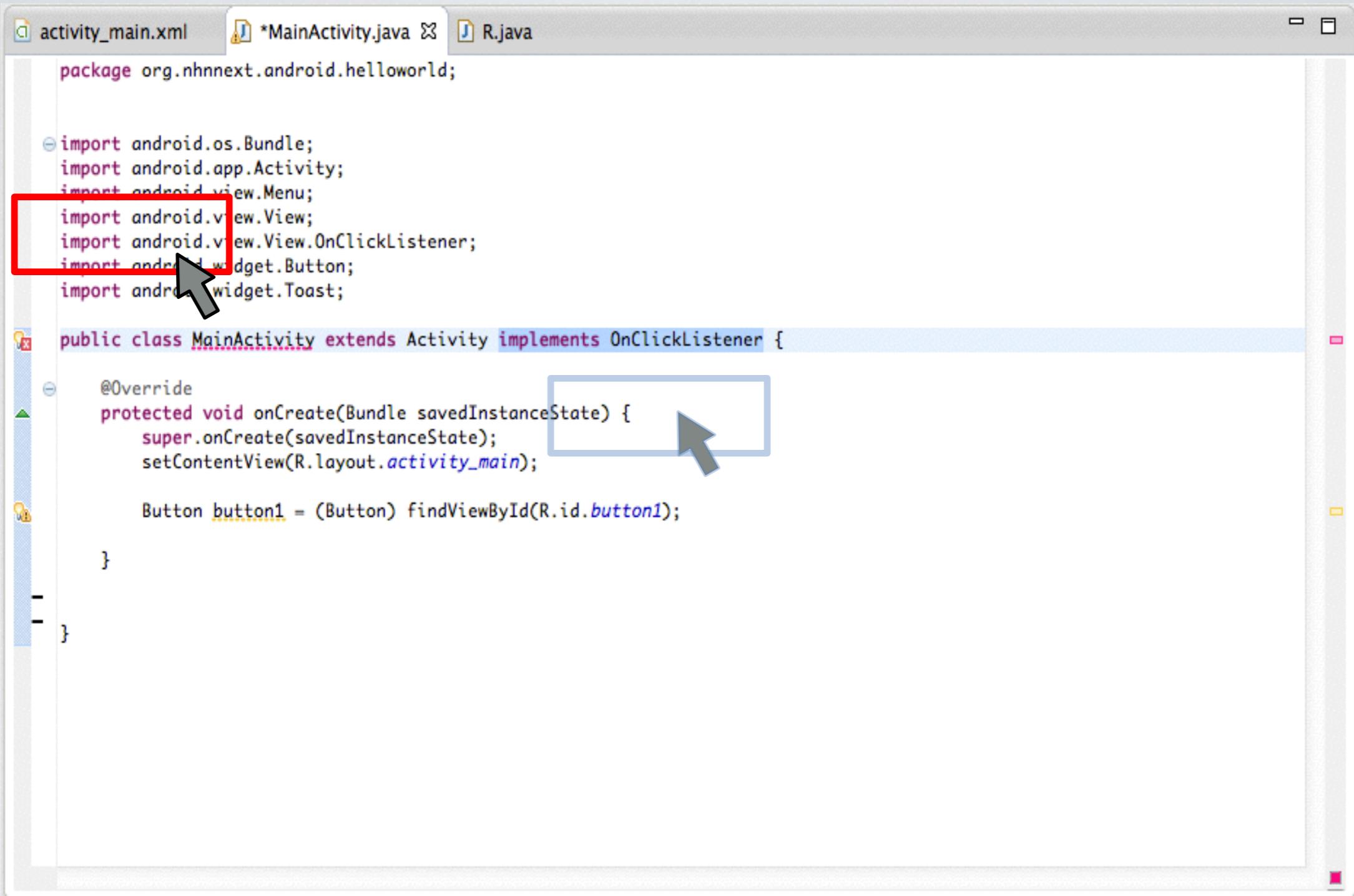
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.button1:
            Toast.makeText(getApplicationContext(), "버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();
            textView1.setText( editText1.getText().toString() + "이 입력되었습니다." );
            imageView1.setImageResource(R.drawable.gyunbin);
            break;
        case R.id.imageButton1:
            Toast.makeText(getApplicationContext(), "이미지버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();
            textView1.setText( editText1.getText().toString() + "이 입력되었습니다." );
            imageView1.setImageResource(R.drawable.singer);
            break;
    }
}
```

The graphical layout editor pane is visible on the left, showing a simple UI with a button, an image button, a text view, and an edit text field.

레이아웃 에디터로 돌아옵니다.



만들 레이아웃에 해당하는 레이아웃을 드래그합니다.



```
activity_main.xml *MainActivity.java R.java

package org.nhnnext.android.helloworld;

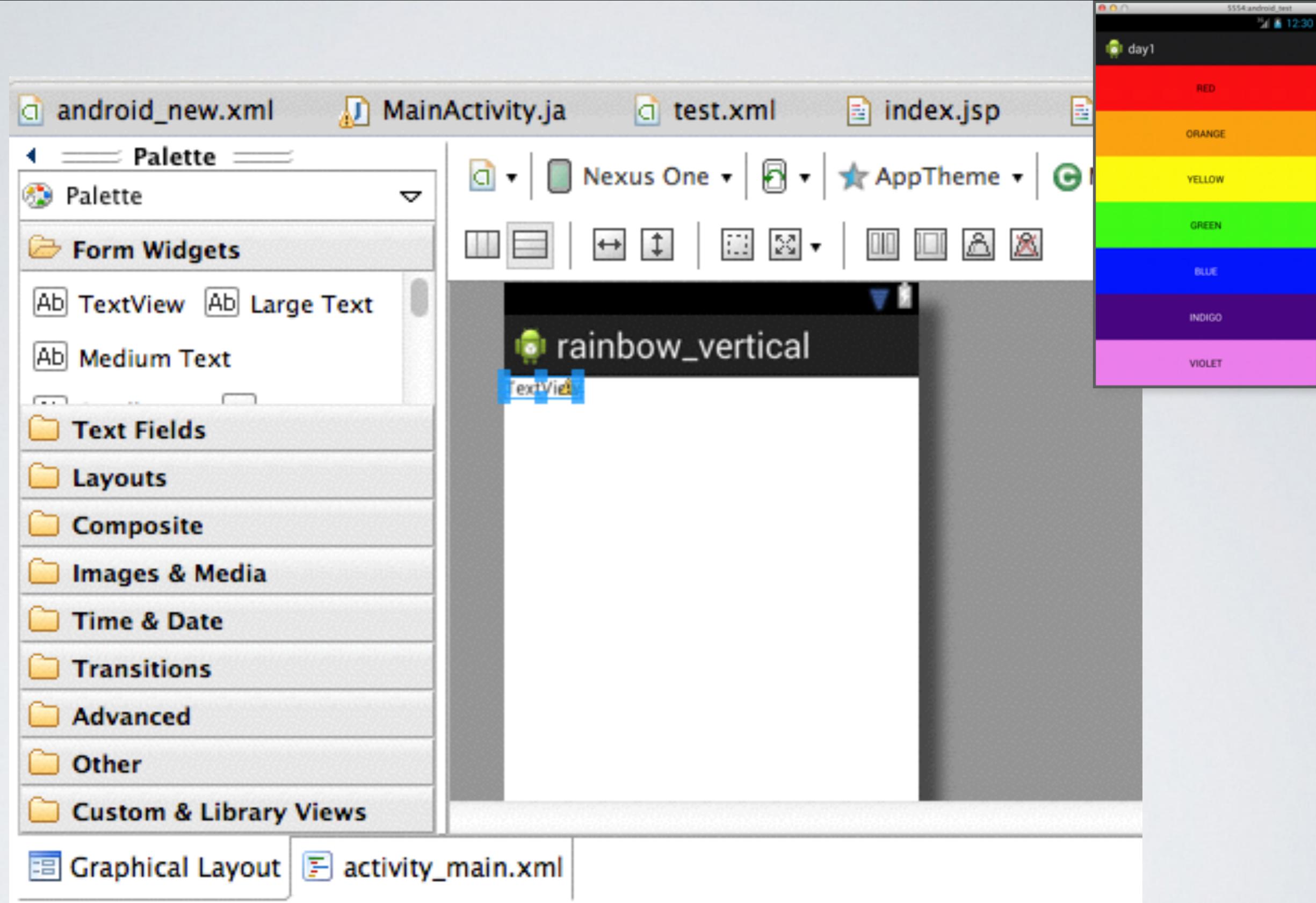
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener {

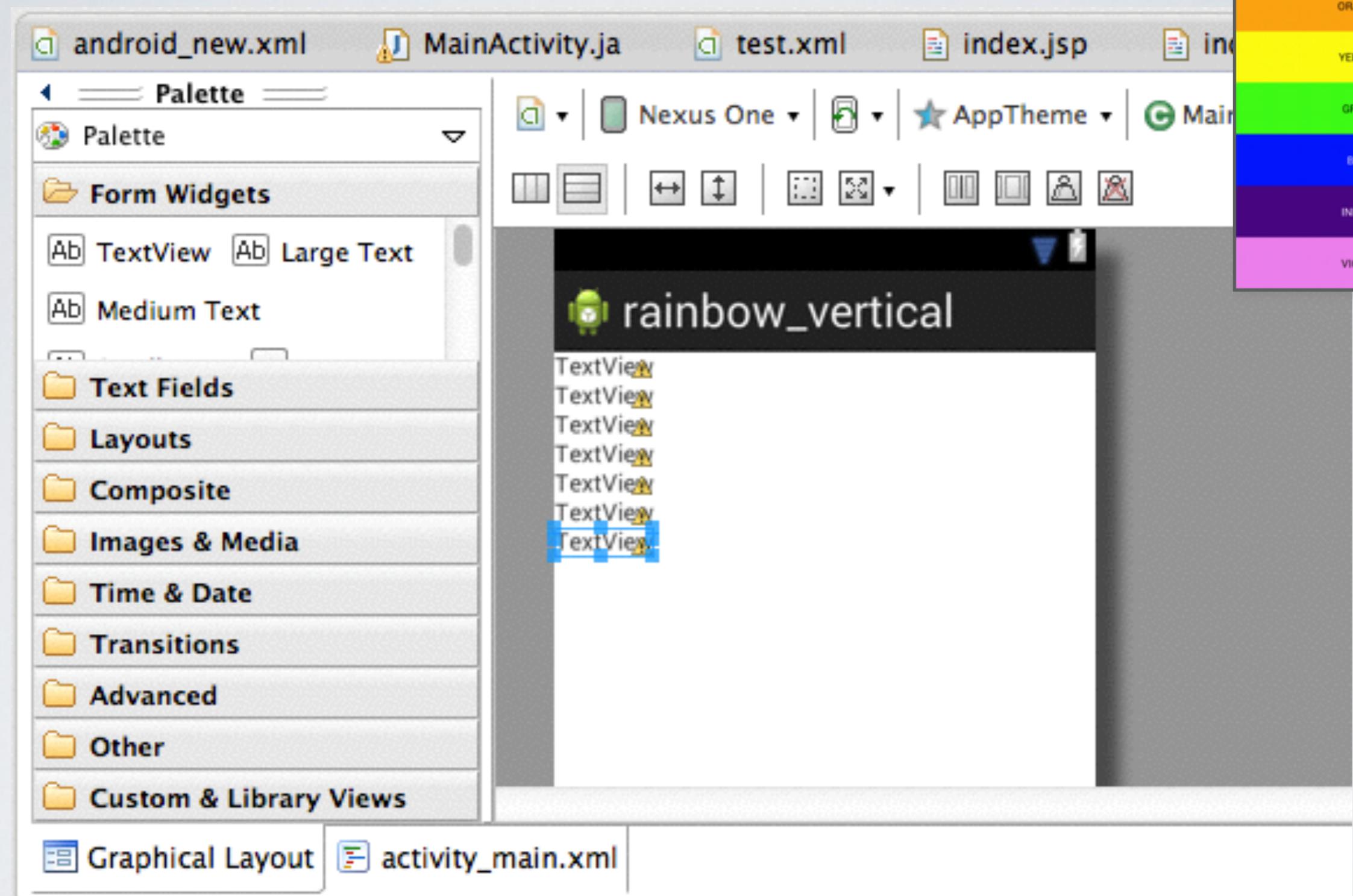
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

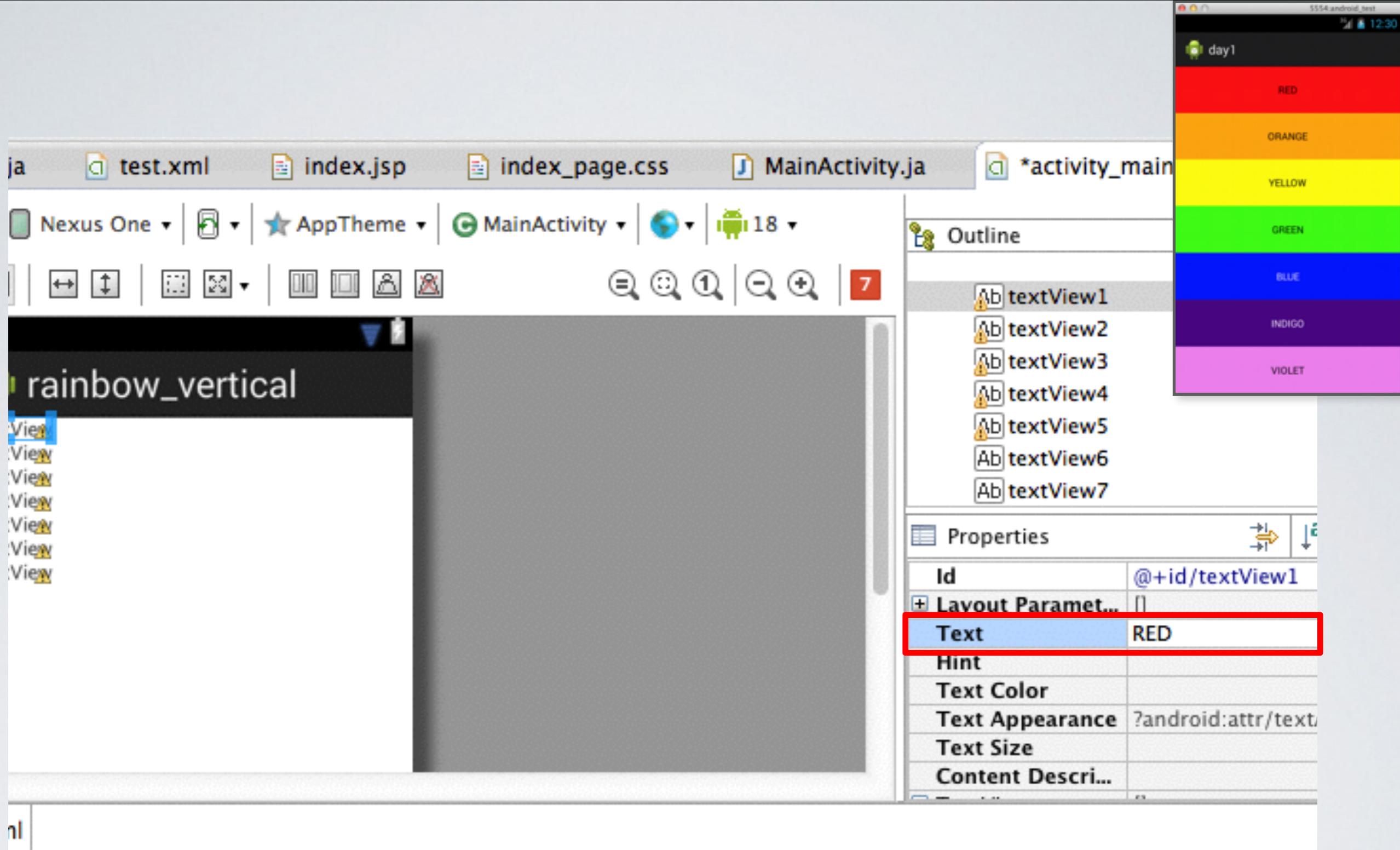
        Button button1 = (Button) findViewById(R.id.button1);
    }
}
```

텍스트뷰를 생성합니다.

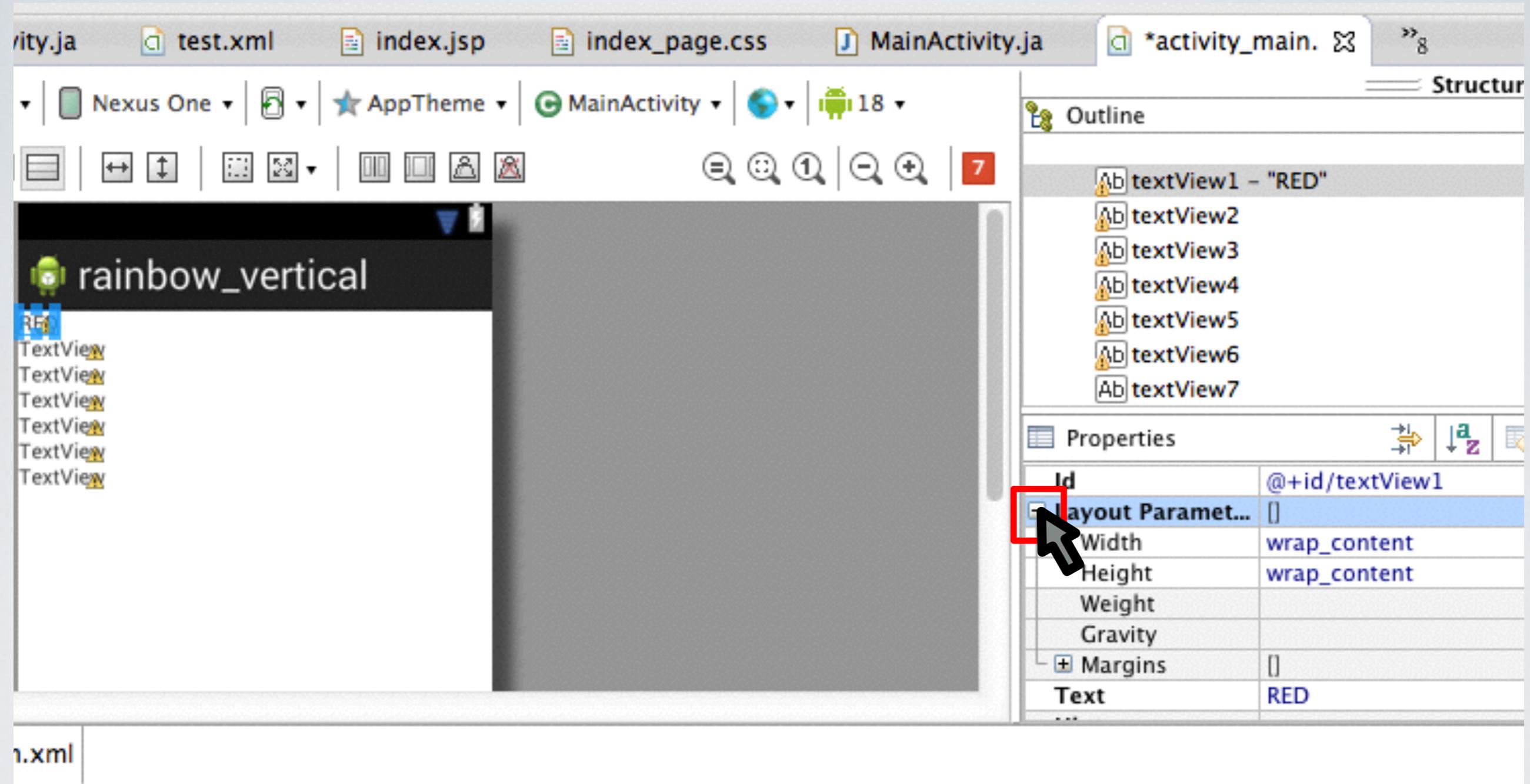


7칸에 해당하는 텍스트뷰 7개를 만듭니다.

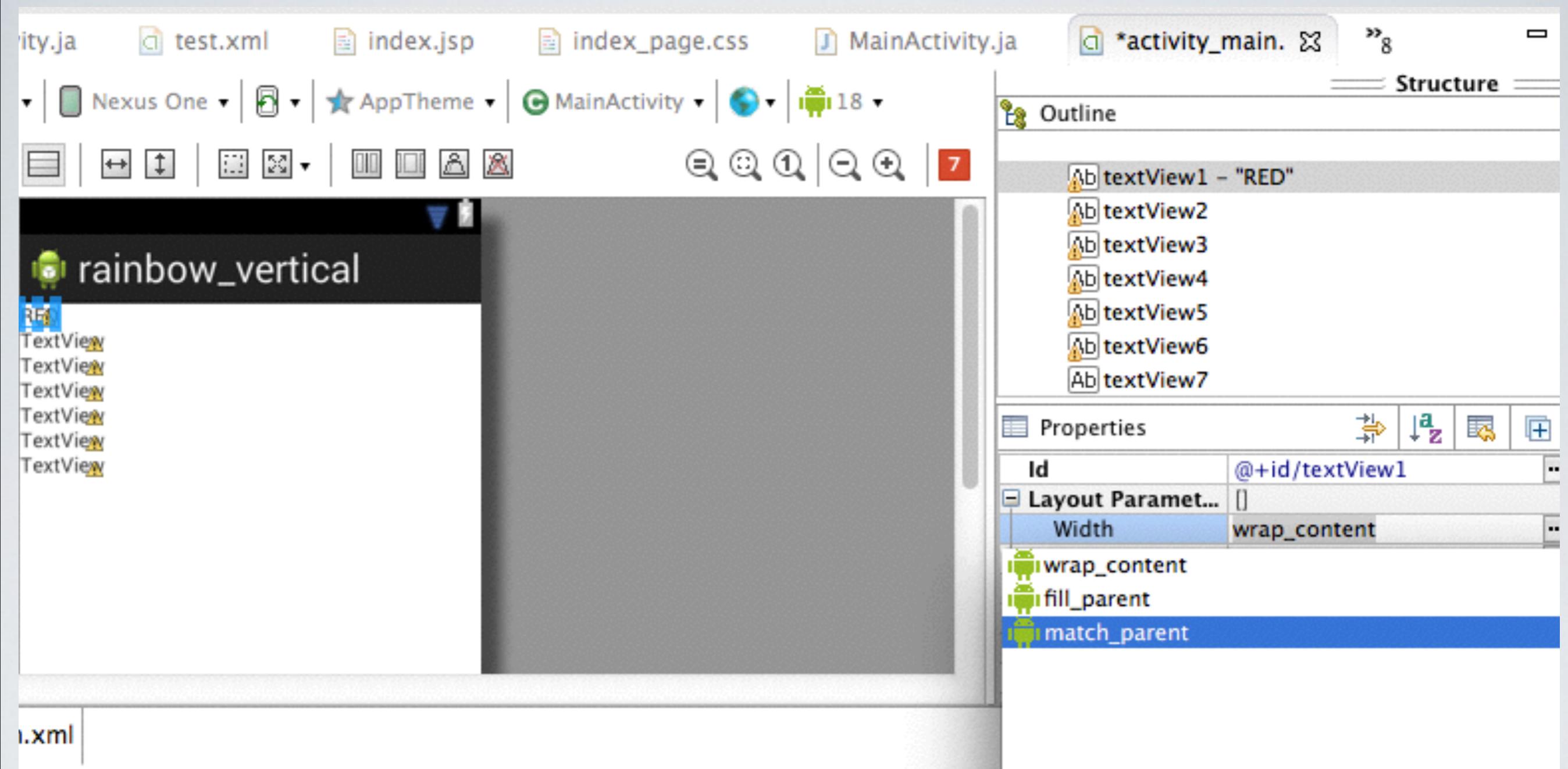




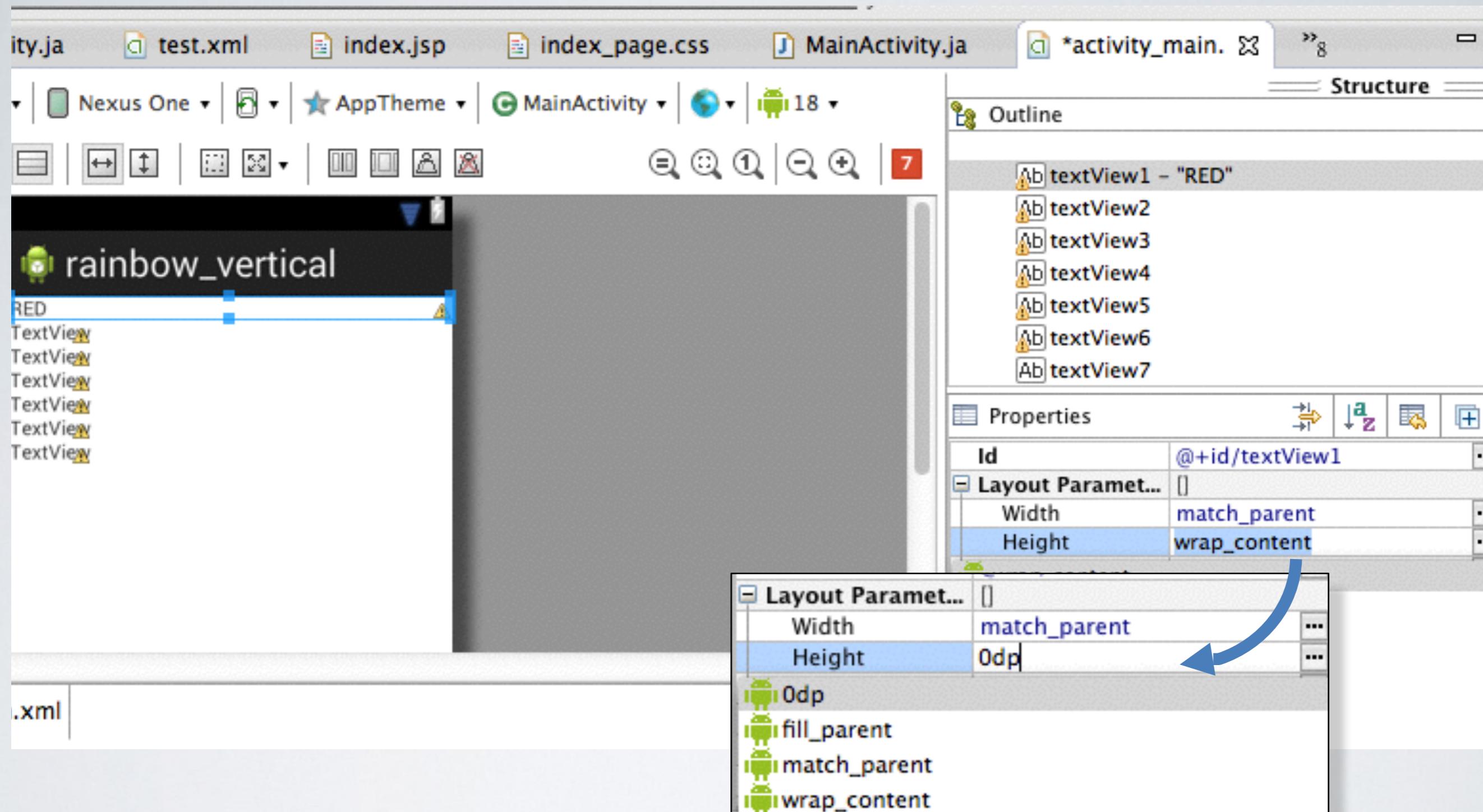
Text 항목을 원하는 글자로 수정합니다.



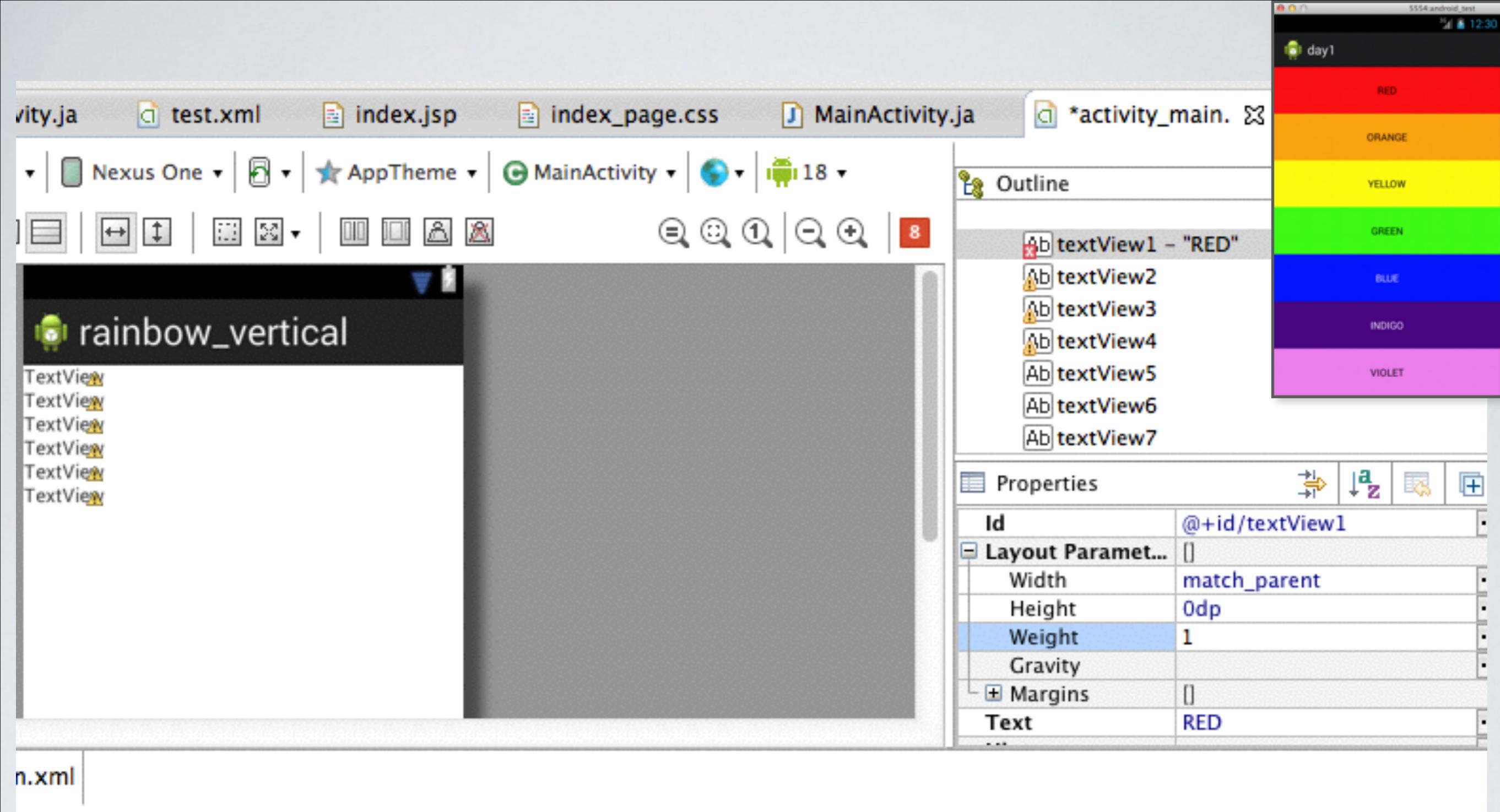
Properties의 Layout Parameter항목을 열어줍니다.



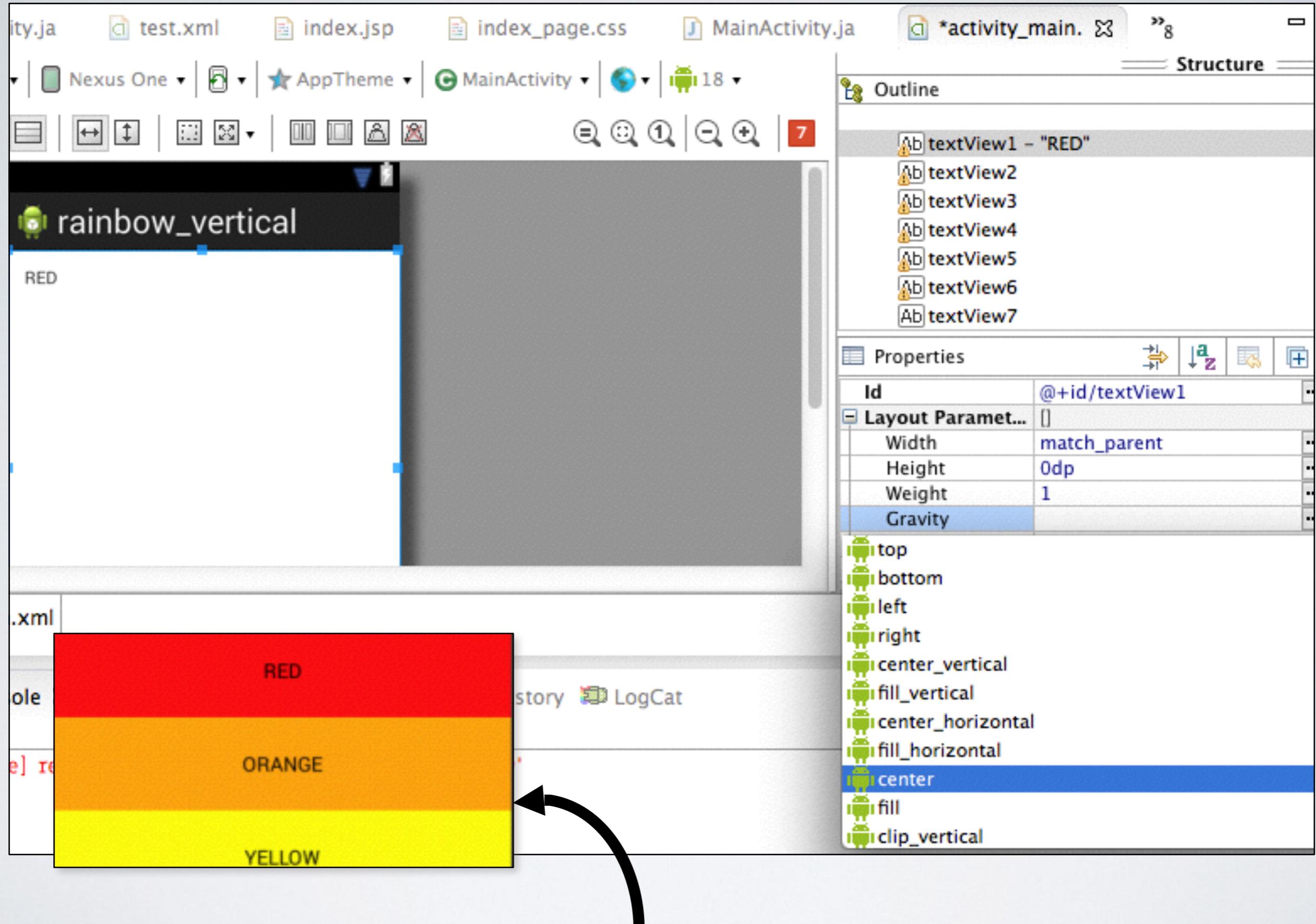
Width항목을 match_parent로 입력합니다.
(가로 길이는 상위 뷰와 같게)



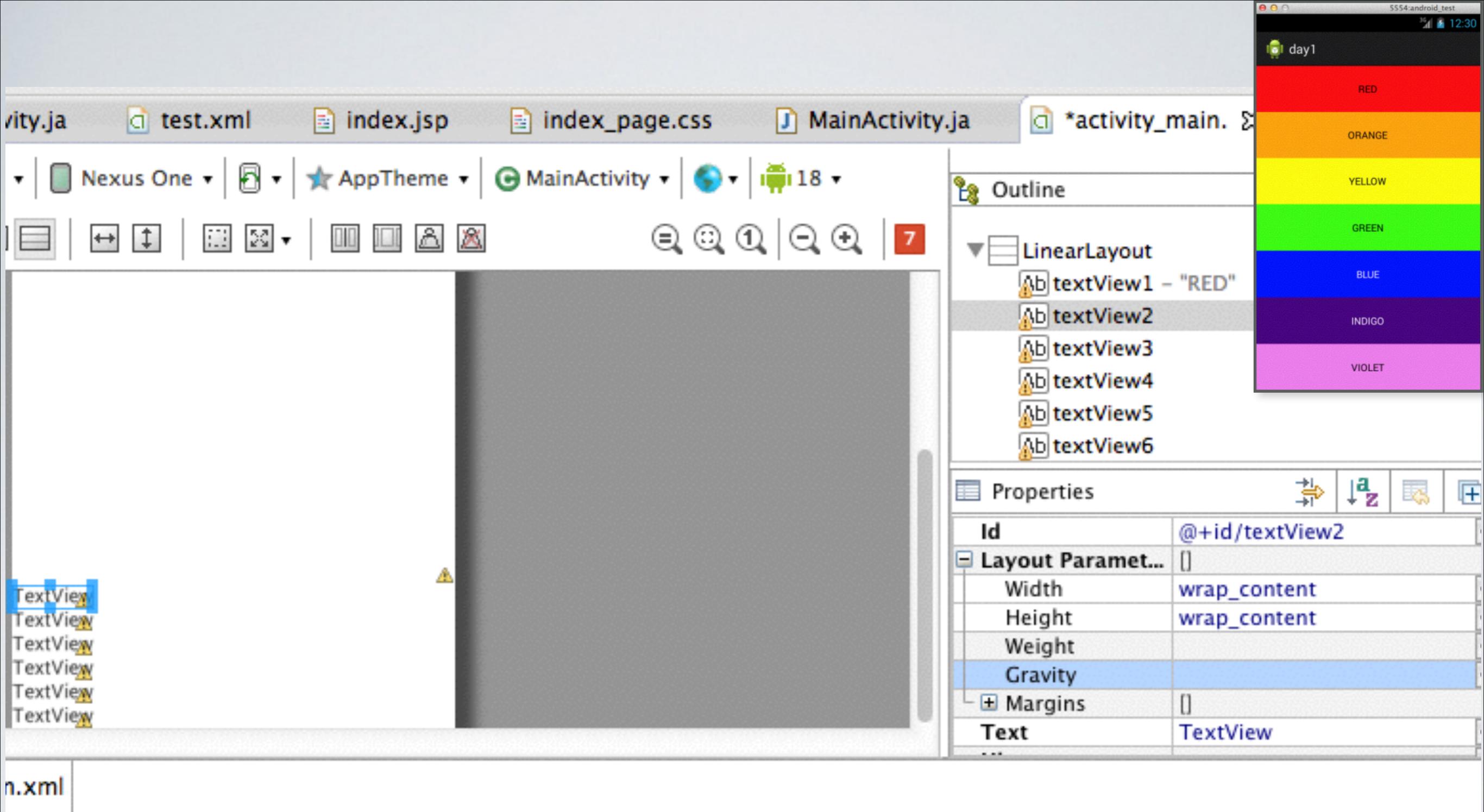
Height 항목을 0dp로 입력합니다.
(wrap_content 아니라 0dp를 주고 weight값에 적용받게 합니다.)



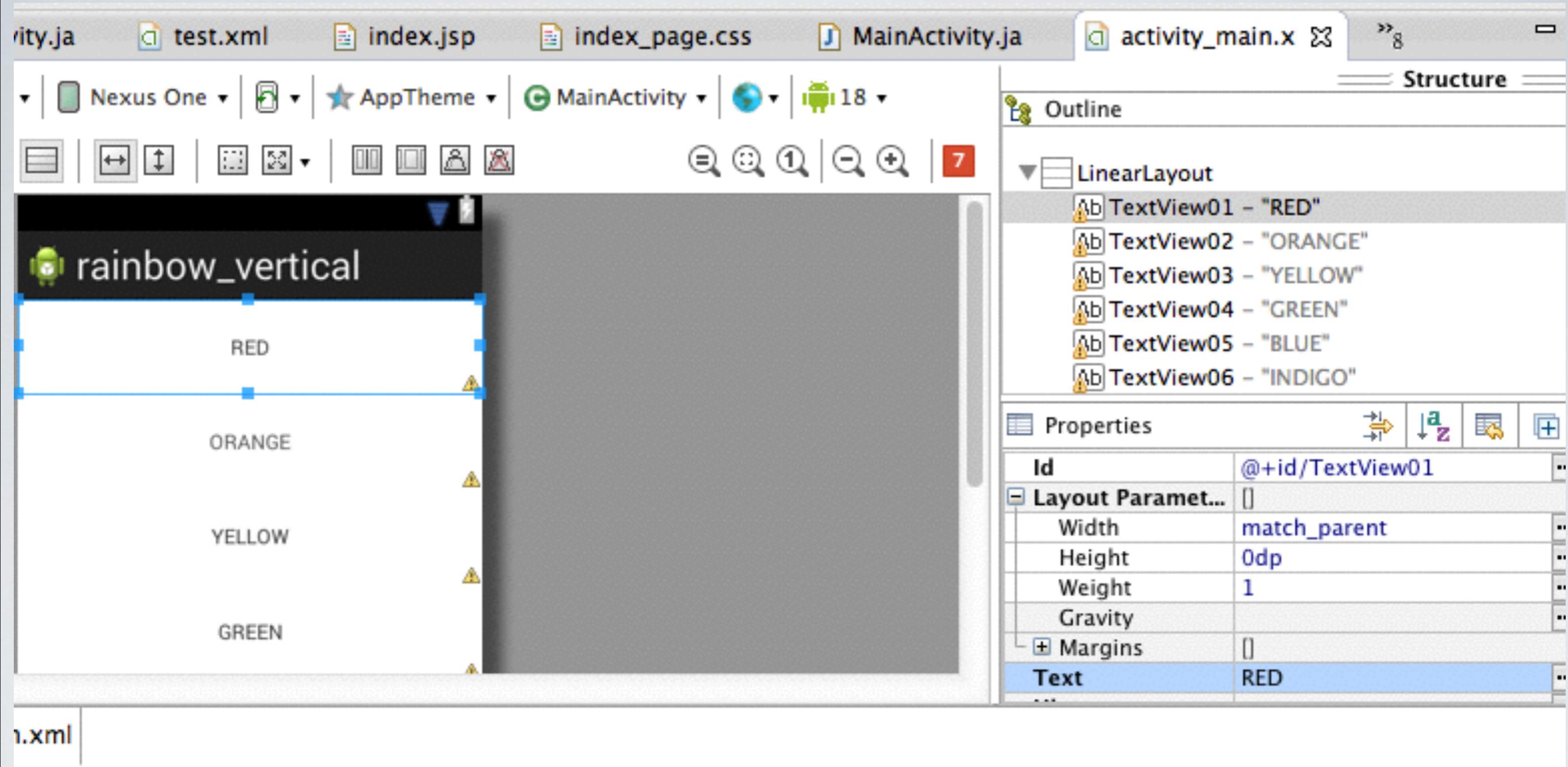
weight값을 1로 입력해줍니다.



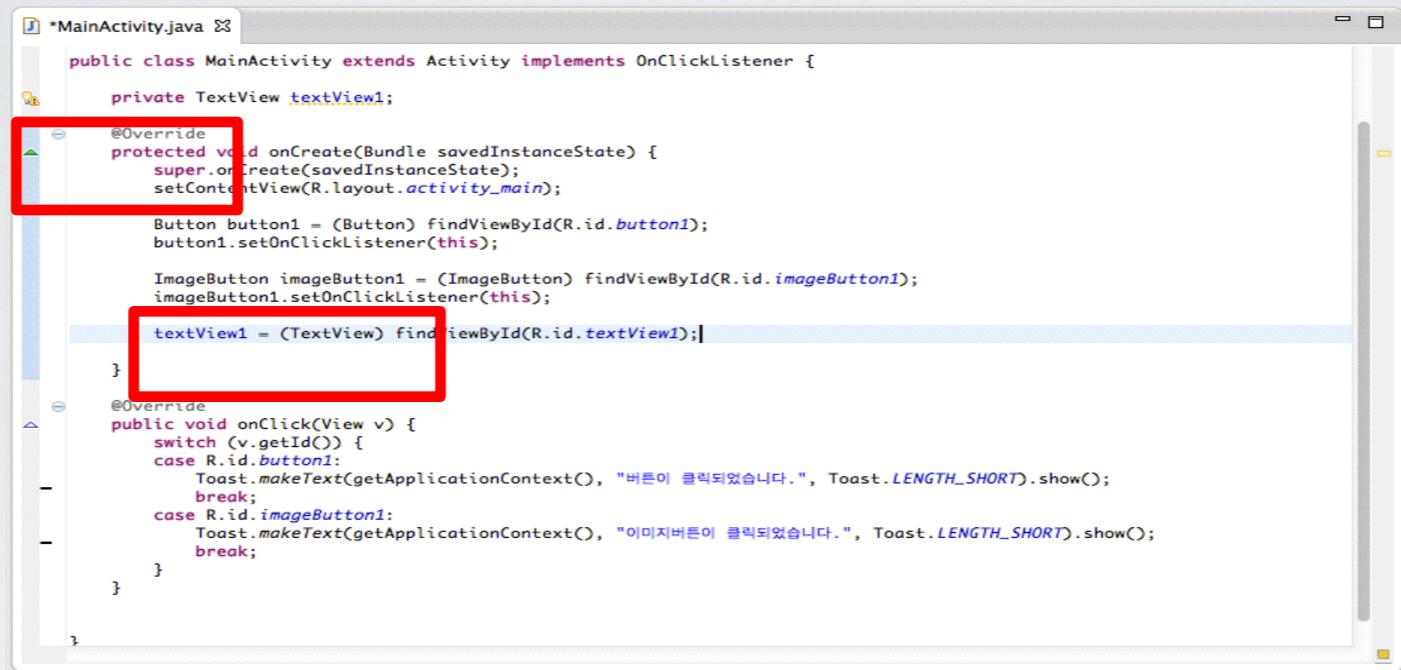
텍스트뷰 안의 글자 정렬을 위해
Gravity를 center로 맞춰줍니다.



다른 6개의 텍스트뷰도 동일하게
Properties 값을 변경해줍니다.
(Text, height, weight, gravity)



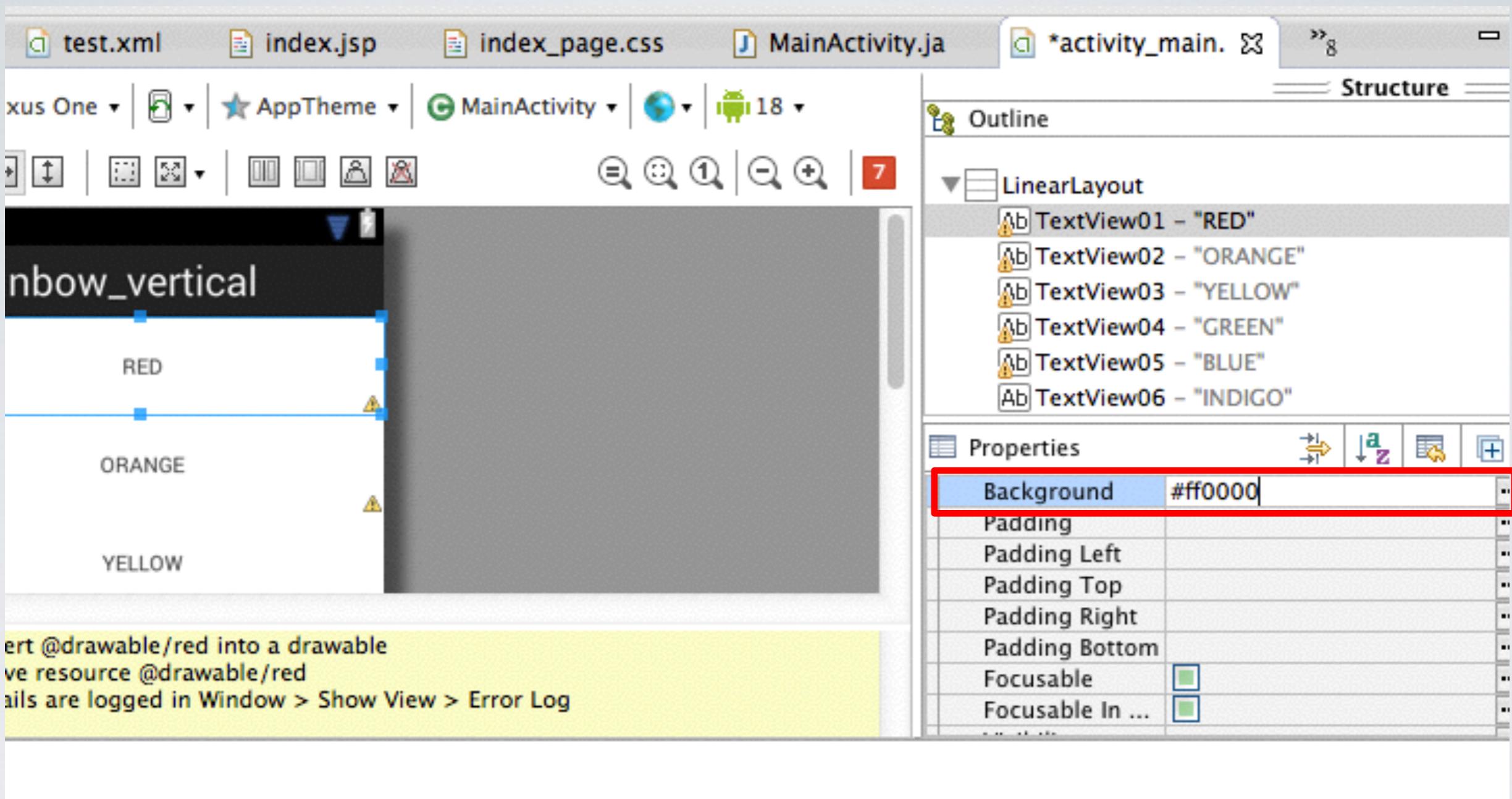
7개의 설정을 다 하였습니다.



```
*MainActivity.java
public class MainActivity extends Activity implements OnClickListener {
    private TextView textView1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button1 = (Button) findViewById(R.id.button1);
        button1.setOnClickListener(this);
        ImageButton imageView1 = (ImageButton) findViewById(R.id.imageView1);
        imageView1.setOnClickListener(this);
        textView1 = (TextView) findViewById(R.id.textView1);
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button1:
                Toast.makeText(getApplicationContext(), "버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();
                break;
            case R.id.imageView1:
                Toast.makeText(getApplicationContext(), "이미지버튼이 클릭되었습니다.", Toast.LENGTH_SHORT).show();
                break;
        }
    }
}
```

이제 색을 넣어보겠습니다.

Properties를 아래로 내리면
View 항목 아래의 Background가 보입니다.



색상 코드값으로 배경으로 넣을 색을 지정해줍니다.

사용할 색들의 코드표

RED

- **#FF0000** (= #F00)

ORANGE

- **#FFA500**

YELLOW

- **#FFFF00** (= #FF0)

GREEN

- **#00FF00** (= #0F0)

BLUE

- **#0000FF** (= #00F)

INDIGO

- **#4B0082**

VIOLET

- **#EE82EE**

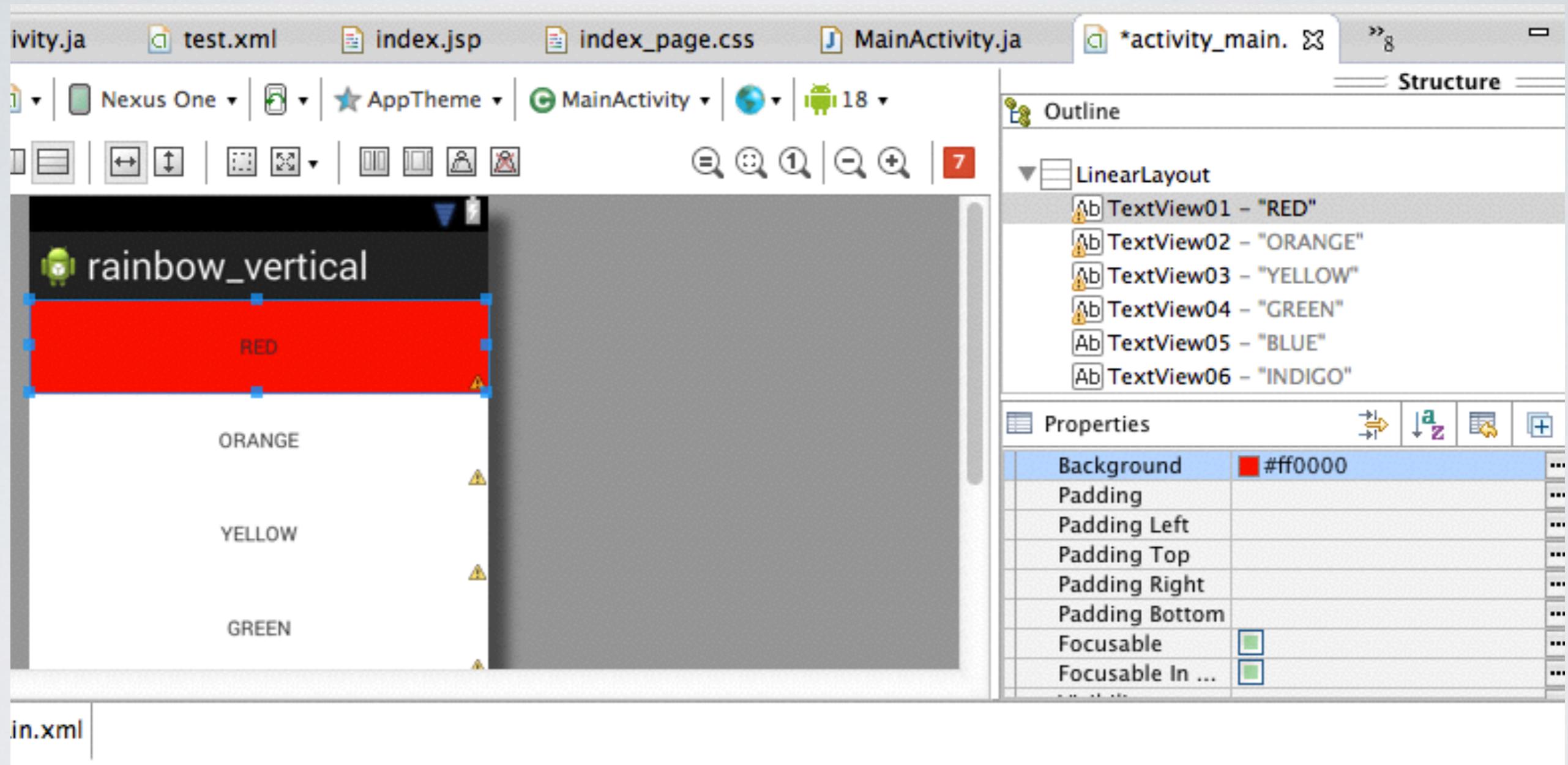
Etc

BLACK

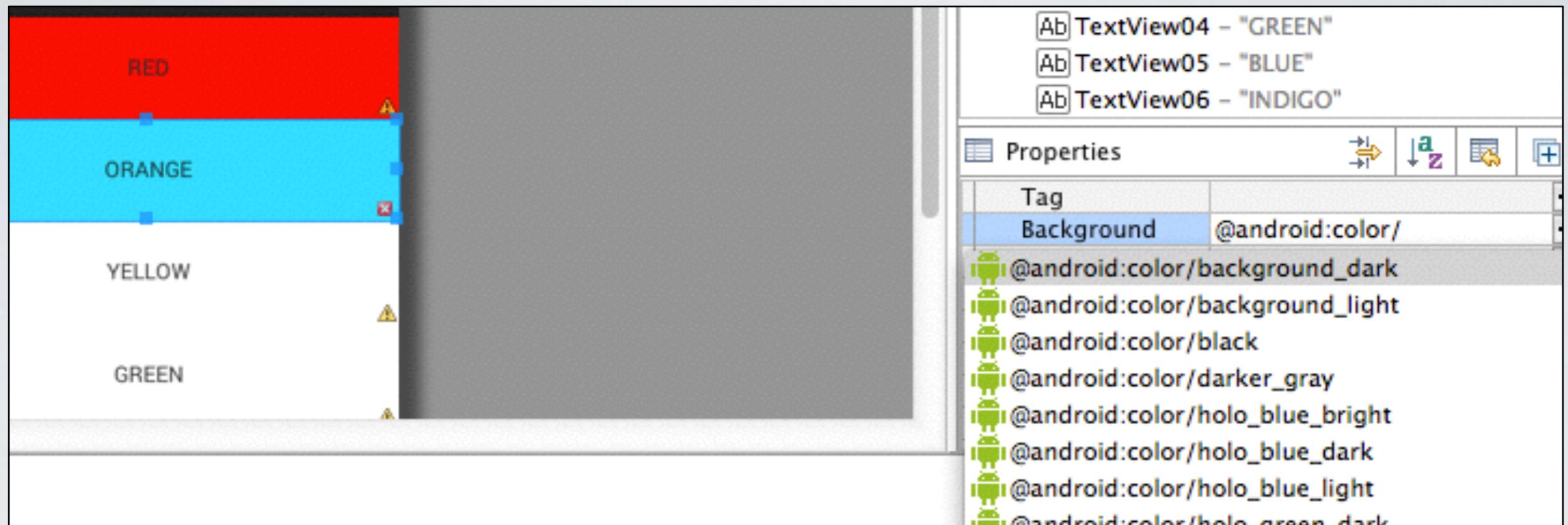
- **#000000** (= #000)

WHITE

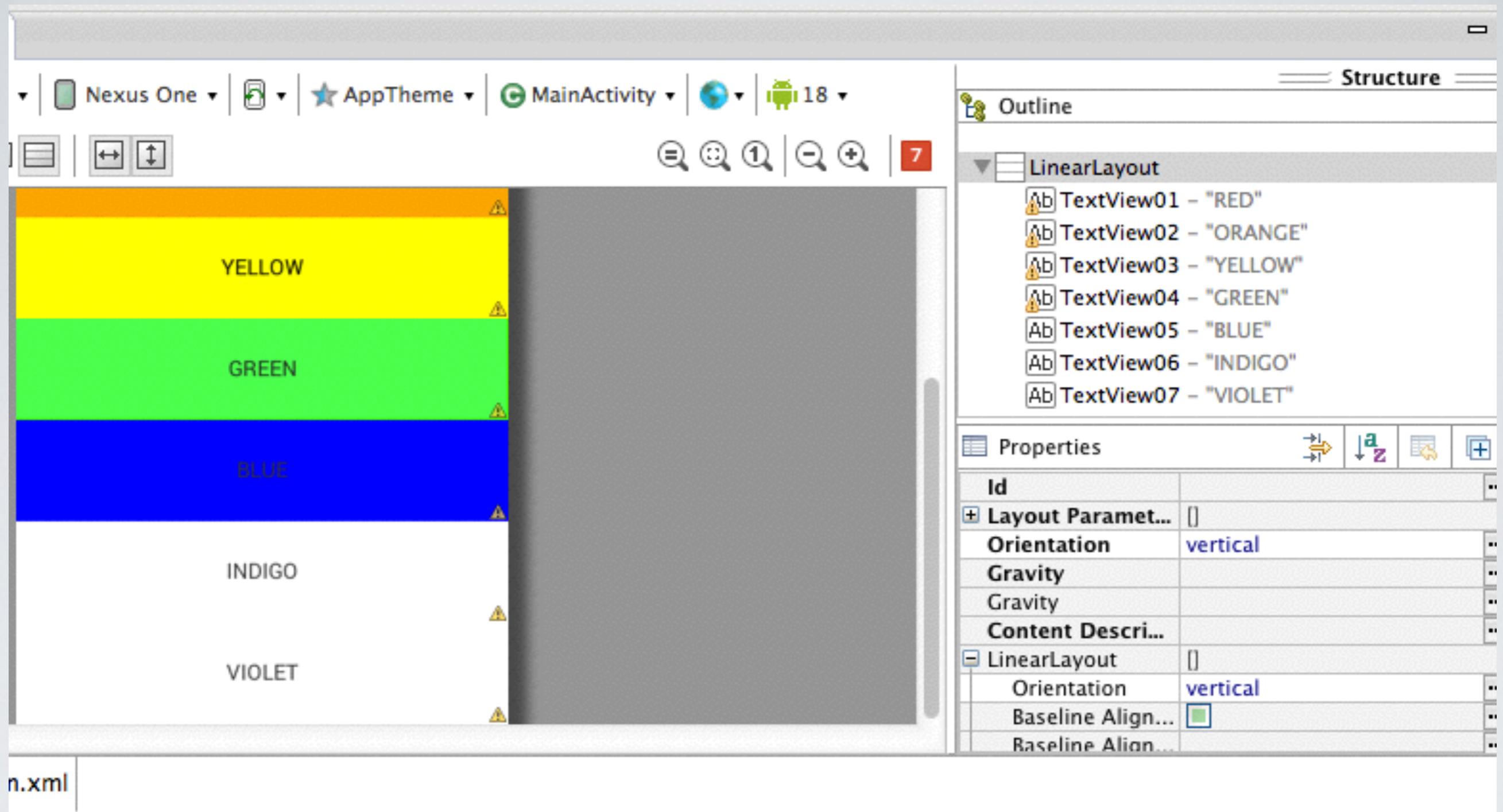
- **#FFFFFF** (= #FFF)



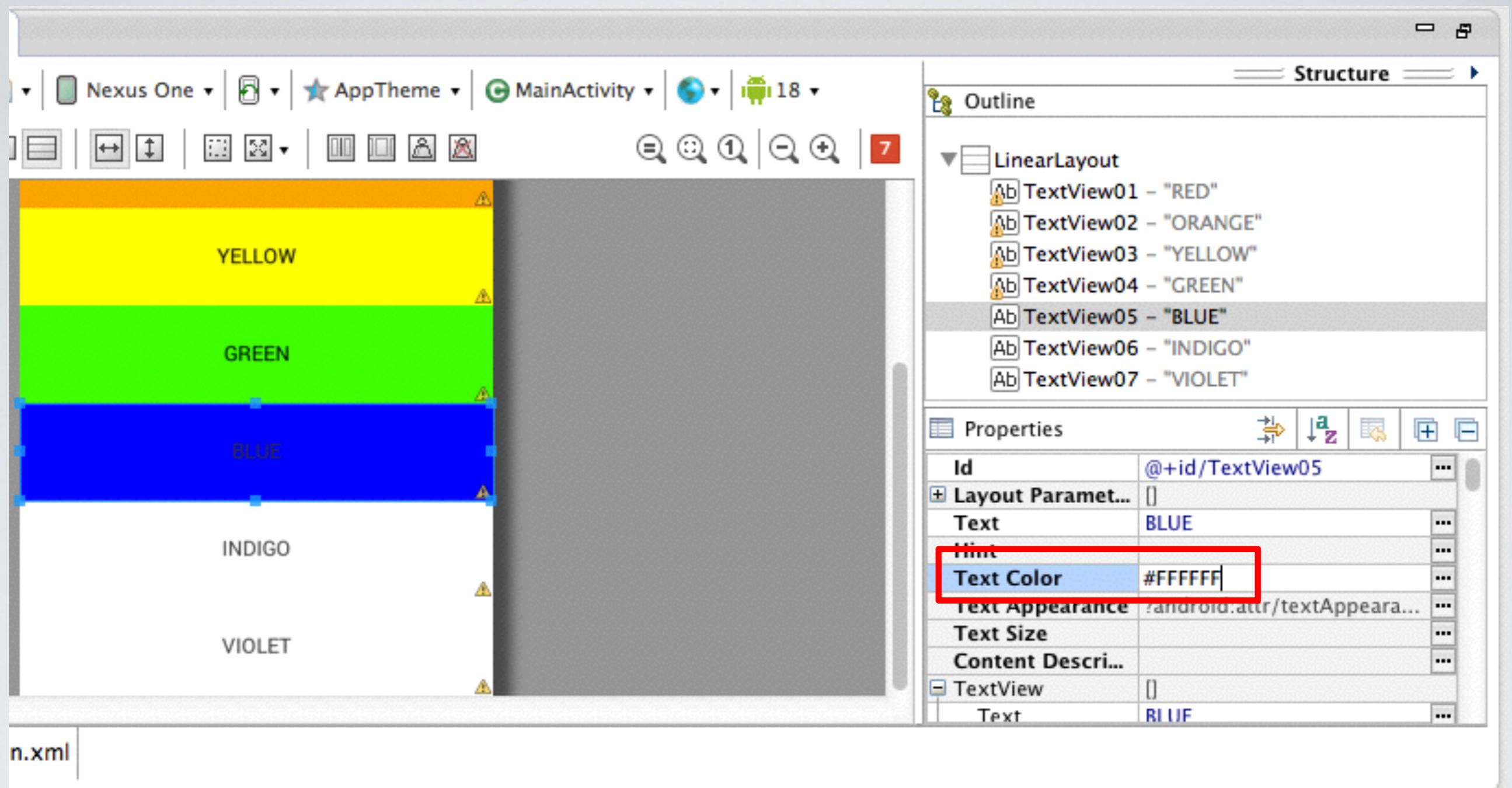
색상 코드로 값을 입력하였습니다.



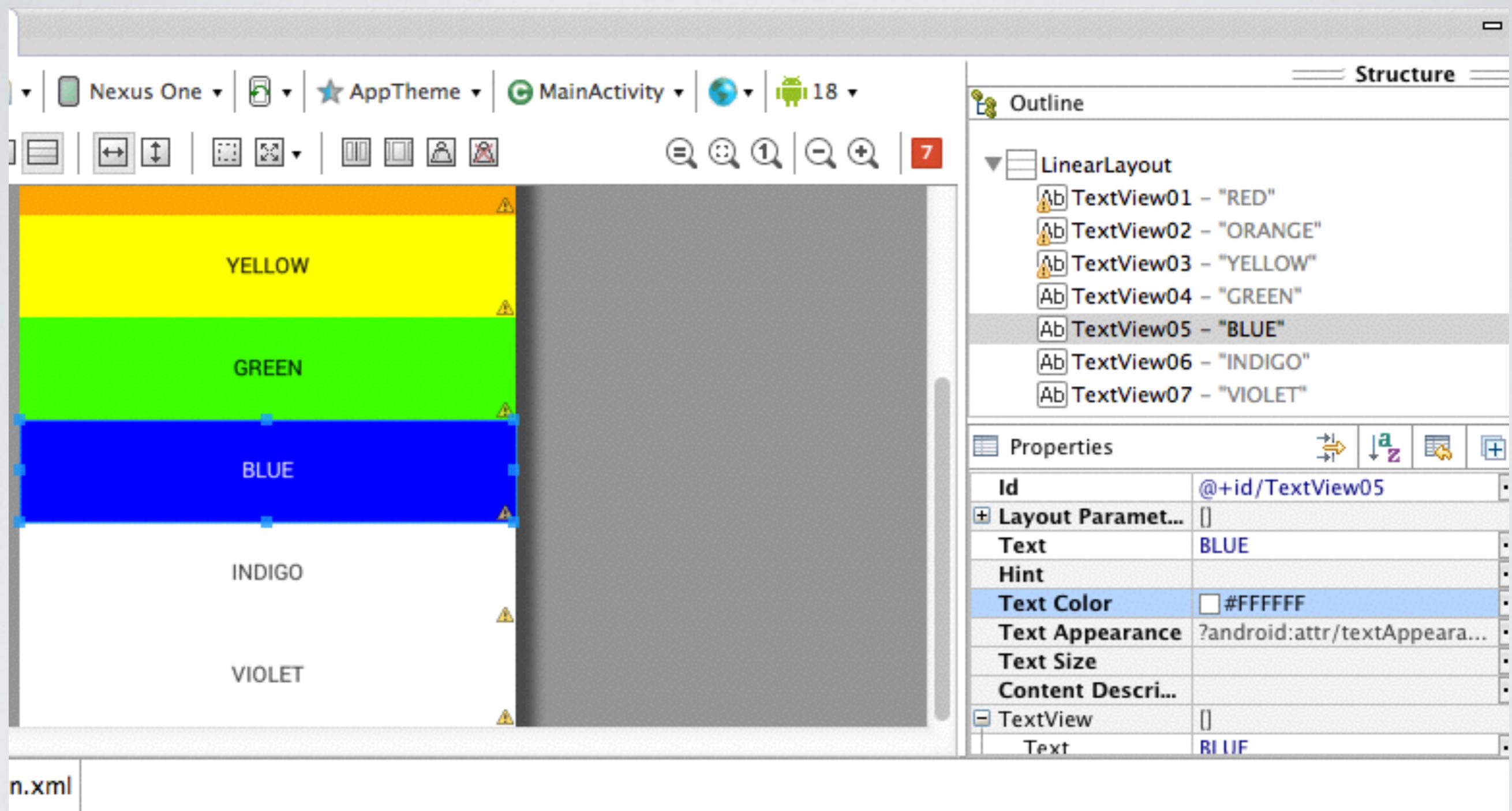
@android:color에 미리 정해진 색을 사용할 수도 있습니다.



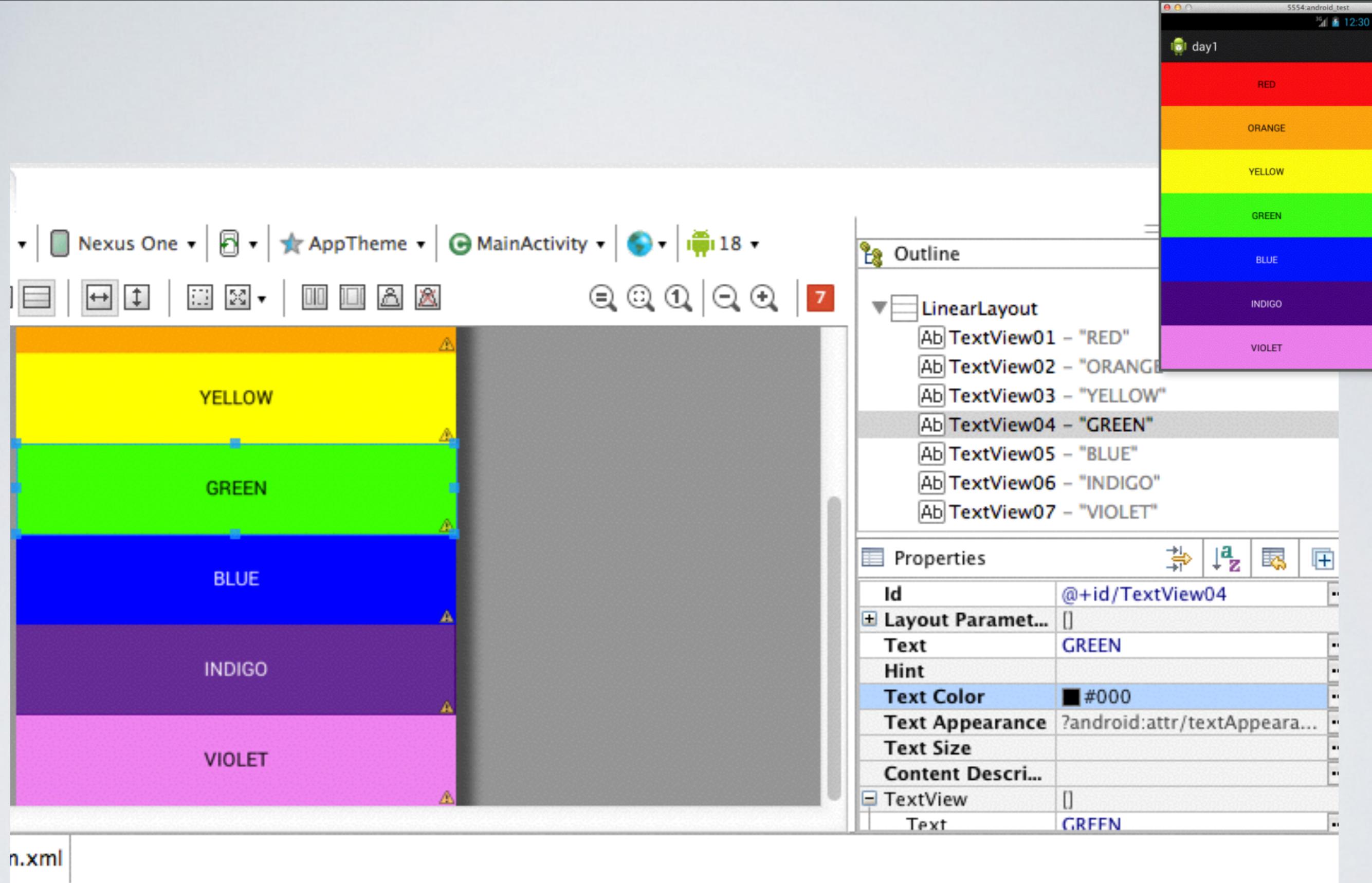
글자의 색을 변경하고 싶을 경우에는...



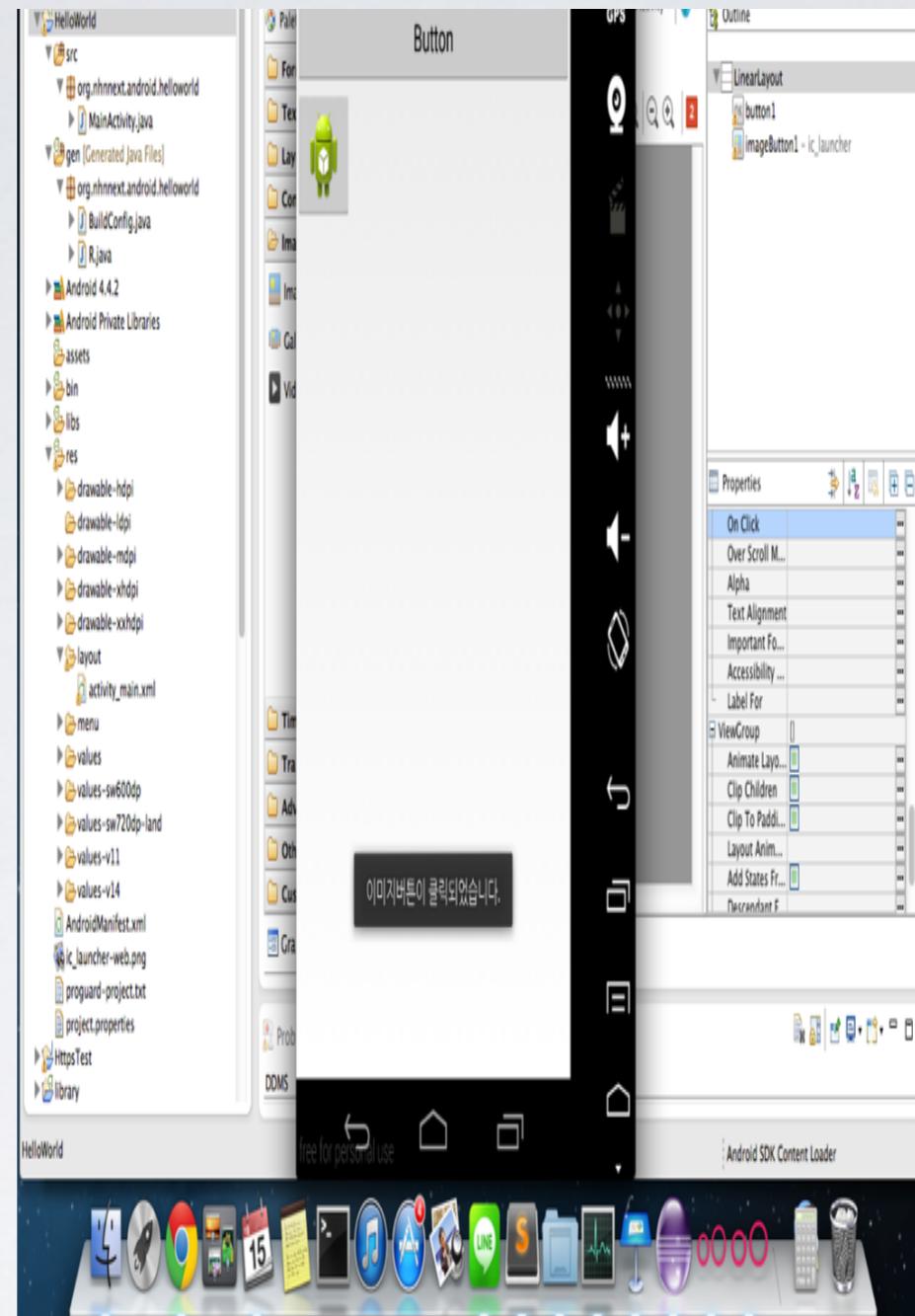
Text Color 항목에 background와 같은 방법으로
색상을 기입 해주시면 됩니다.



작업중...

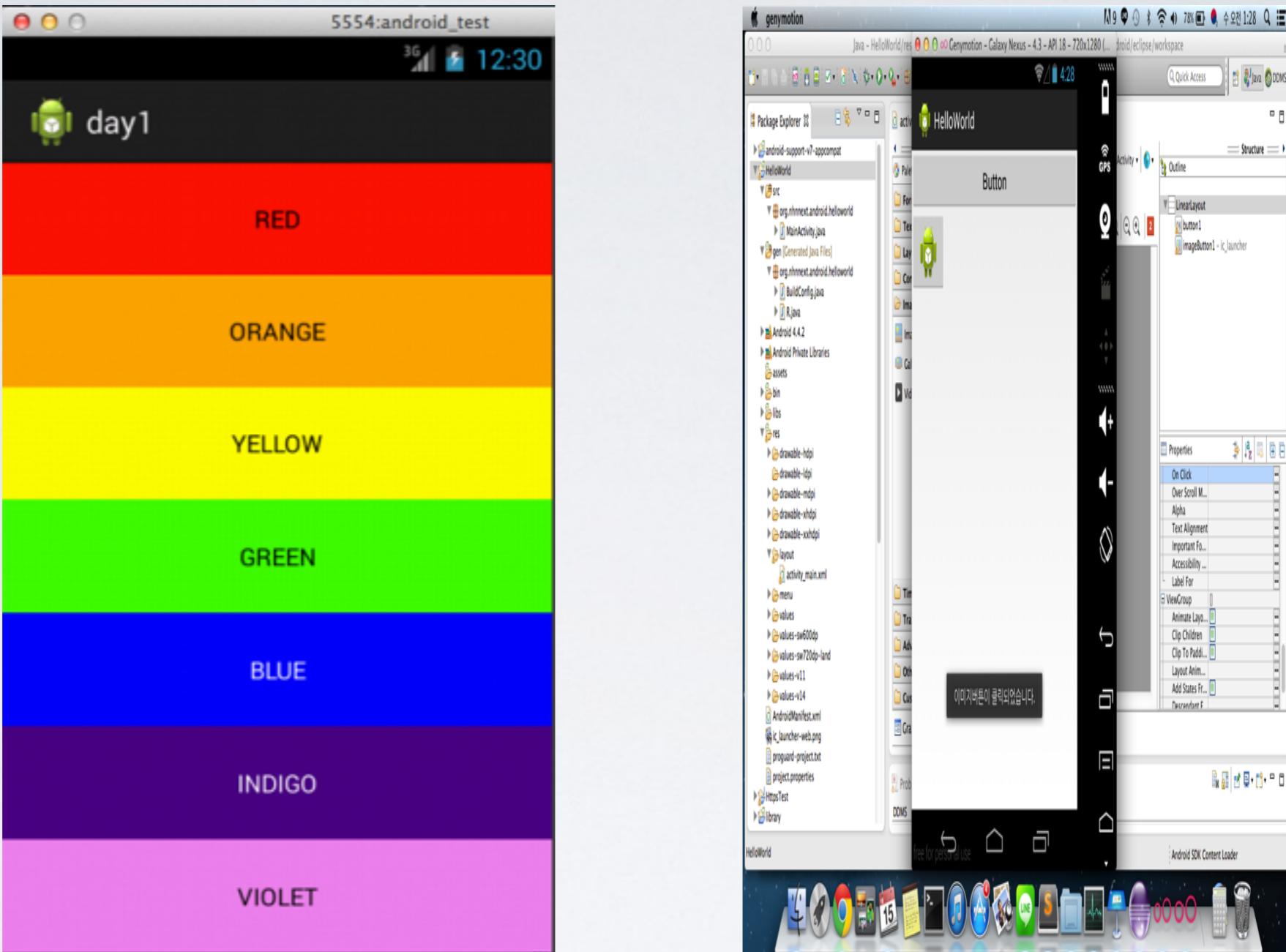


이렇게 7개 항목 전부 색을 입혀보았습니다.



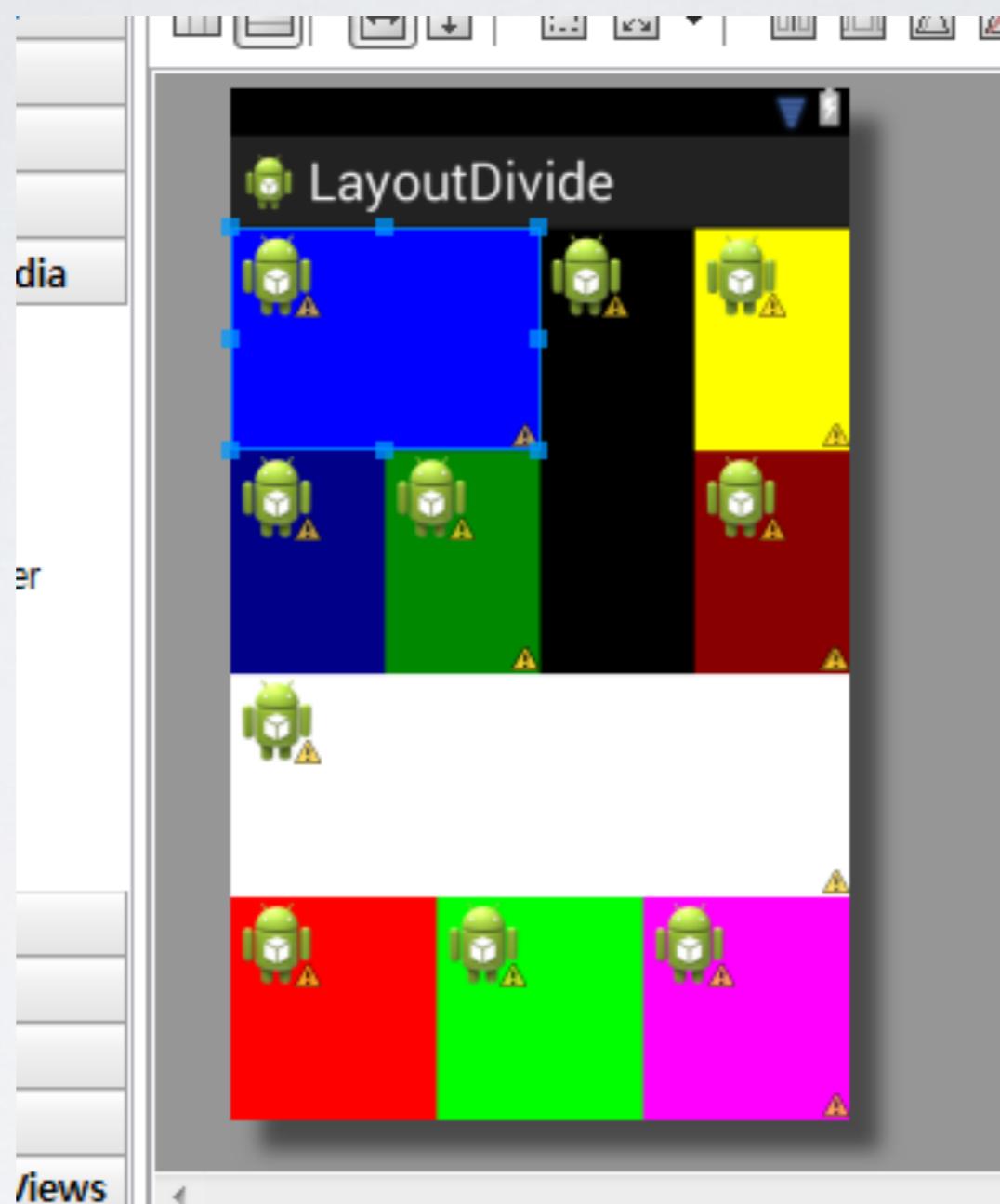
Horizontal 형식으로도 레이아웃을 만들어봅시다.

레이아웃 설정에서
어떤 부분은 같게하고, 어떤 부분을 변경해야 할까요?



https://github.com/yangjeewoong/android_rainbow

과제

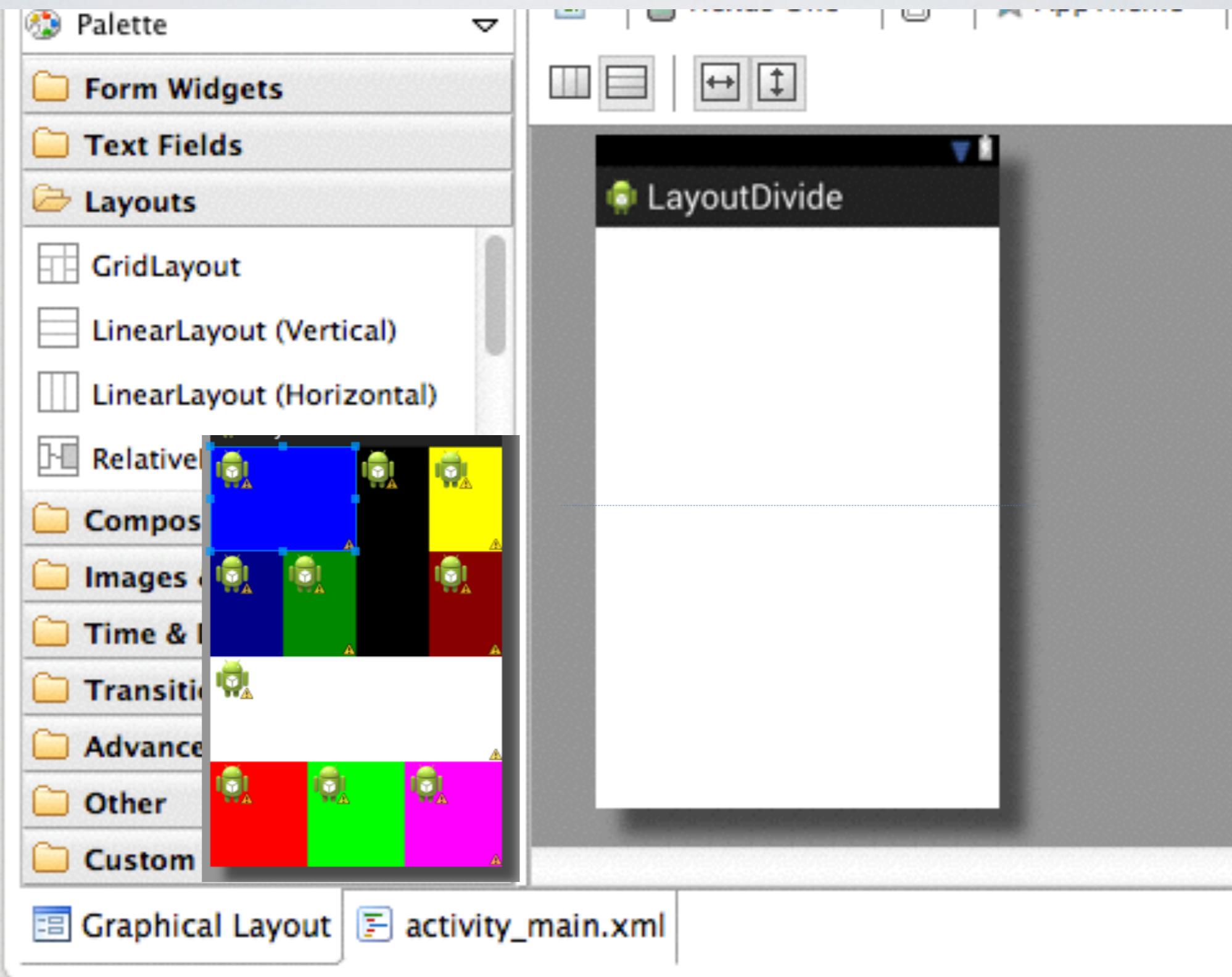


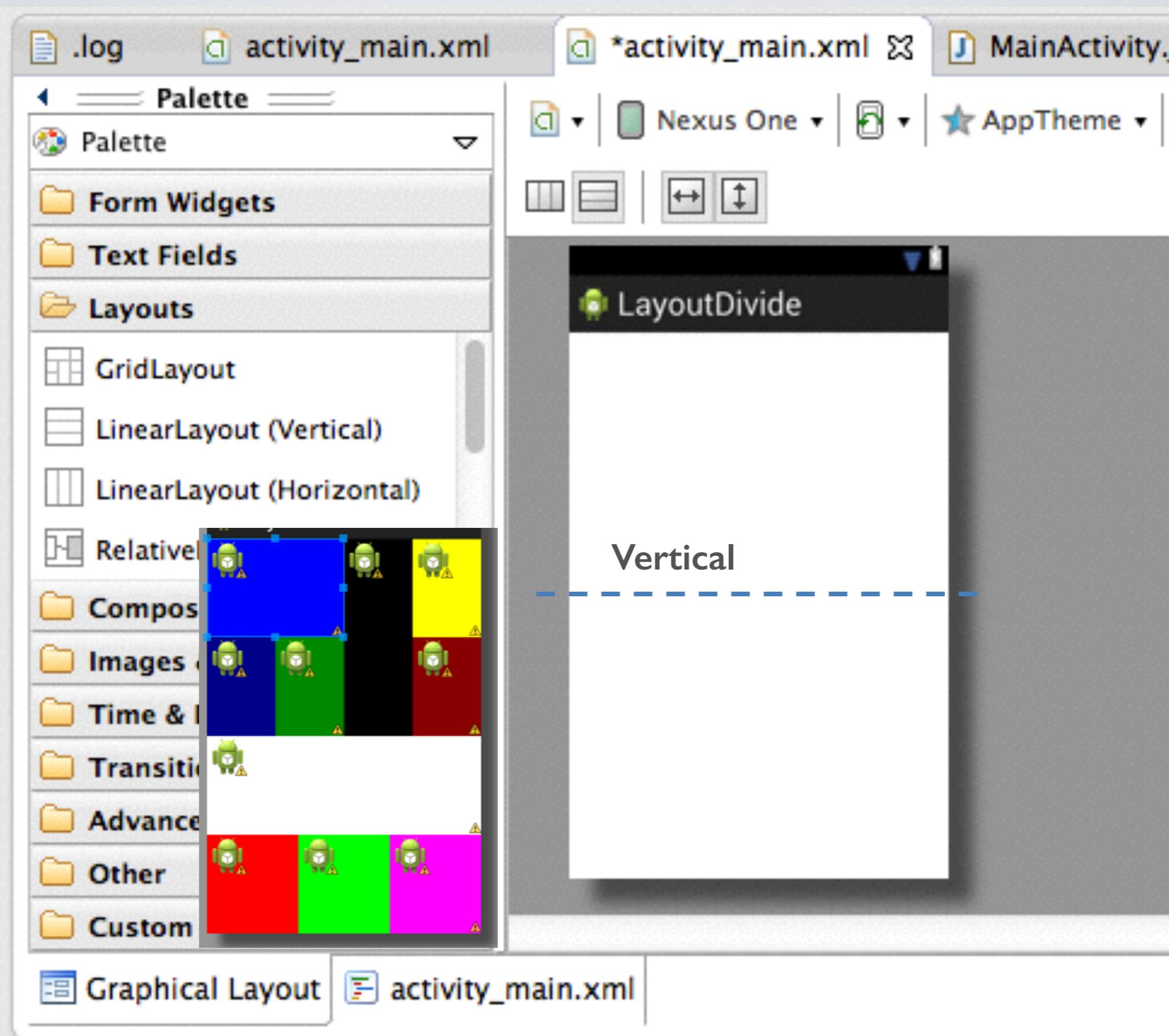
LinearLayout으로 완성해보기

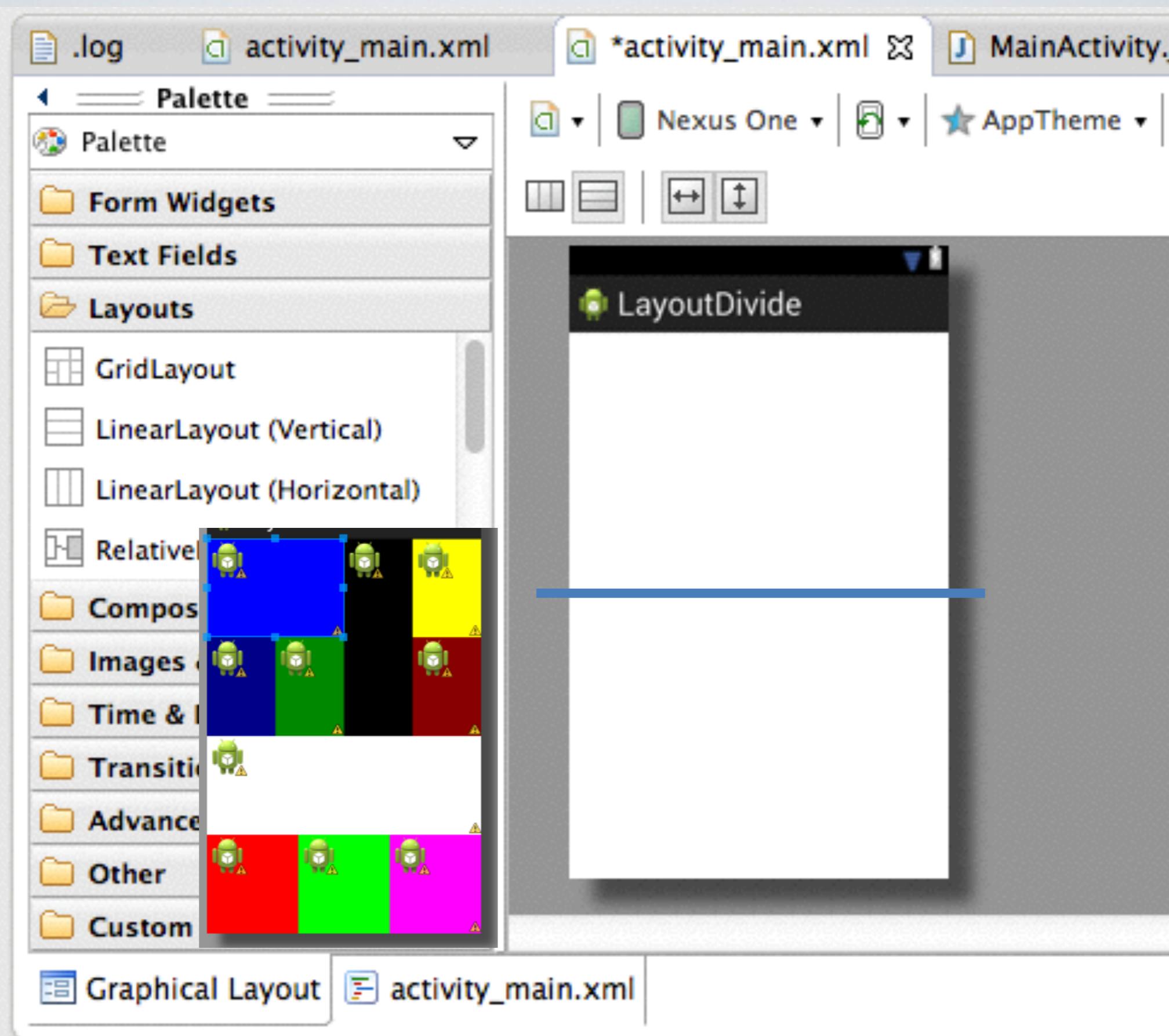


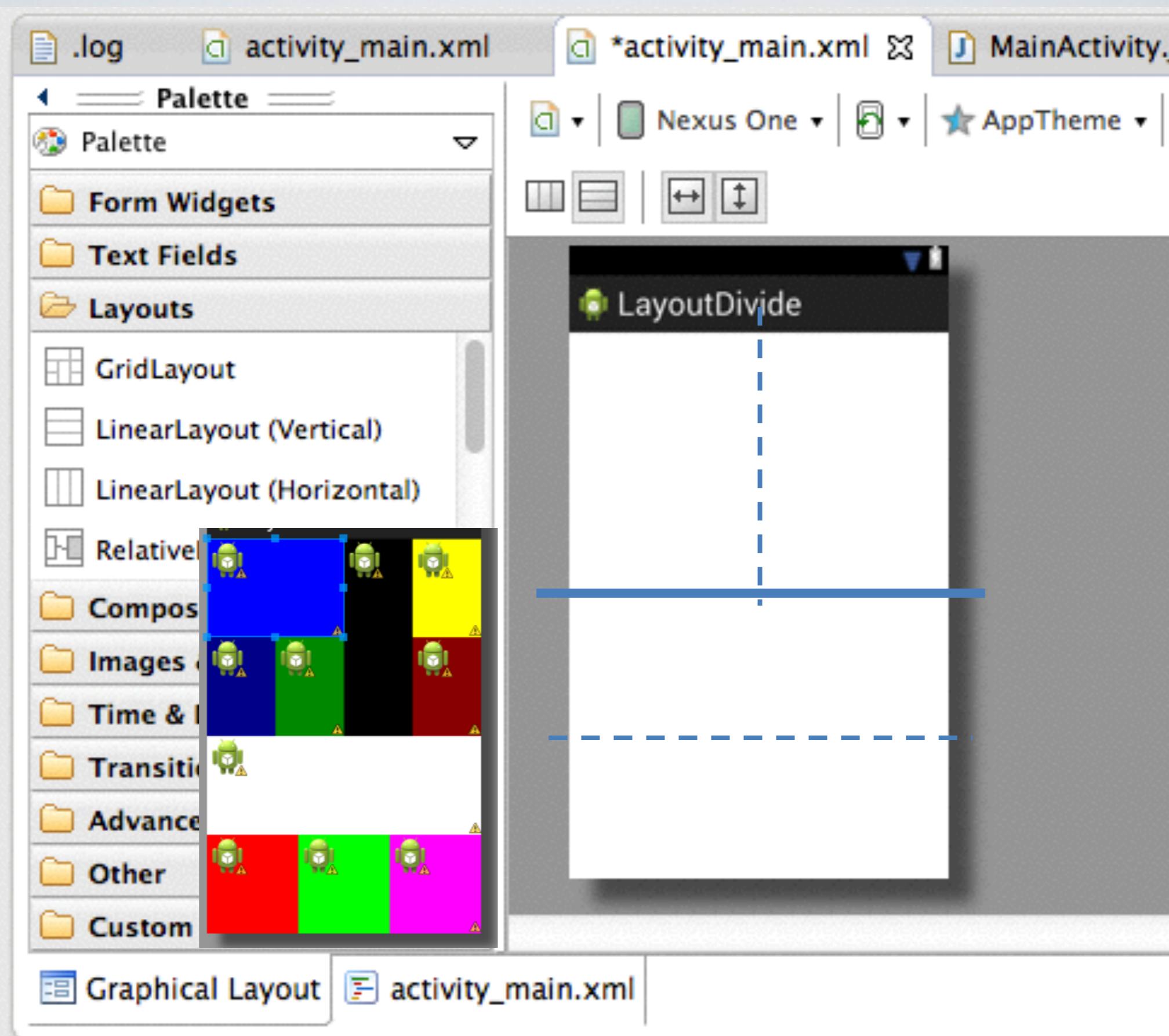
위와 같이 복잡한 레이아웃은
Vertical(가로) LinearLayout과 Horizontal(세로) LinearLayout을
레이아웃 안에 레이아웃이 들어가는 식으로
여러번 중첩해서 만들어야합니다.

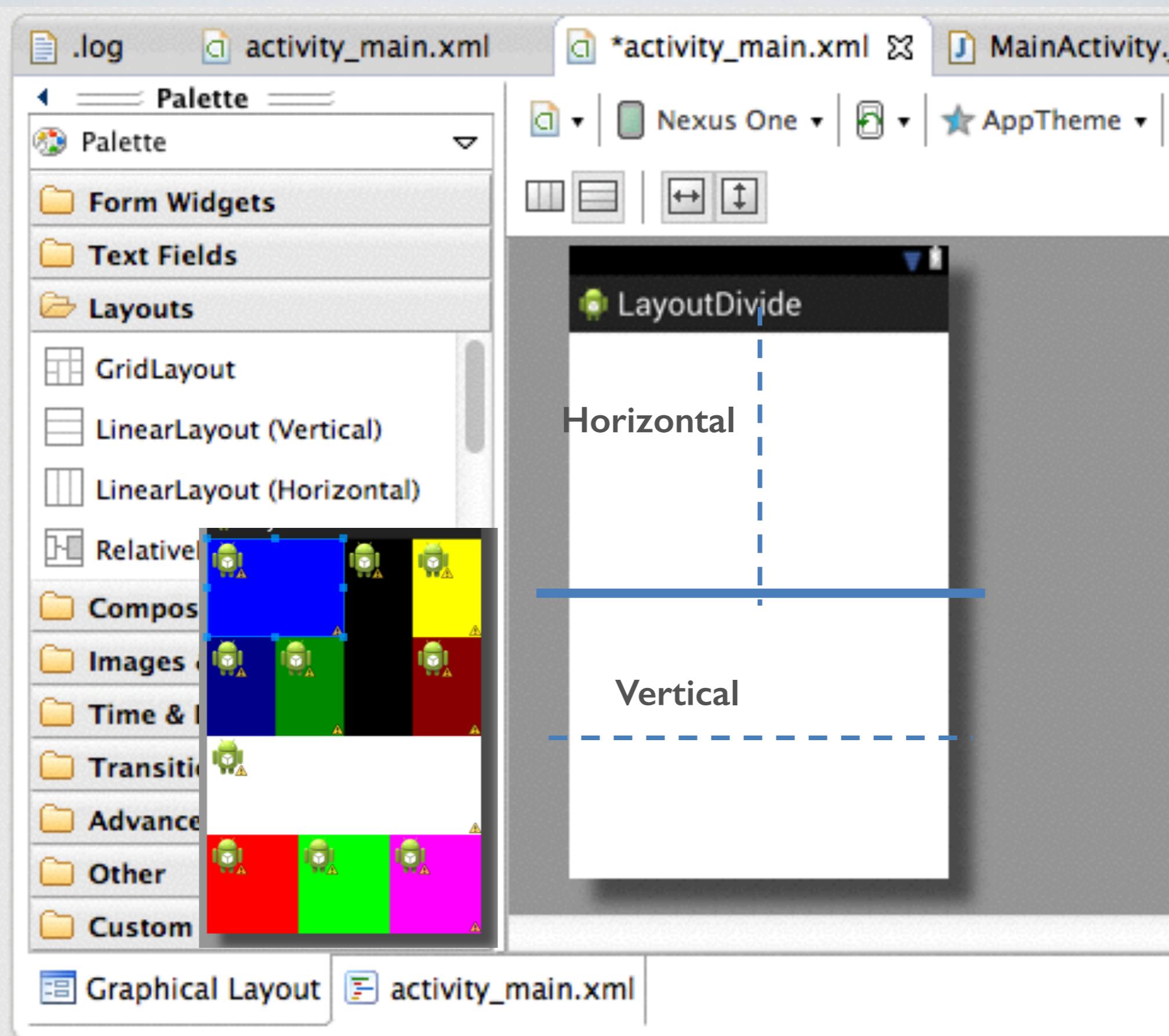
구조 잡기

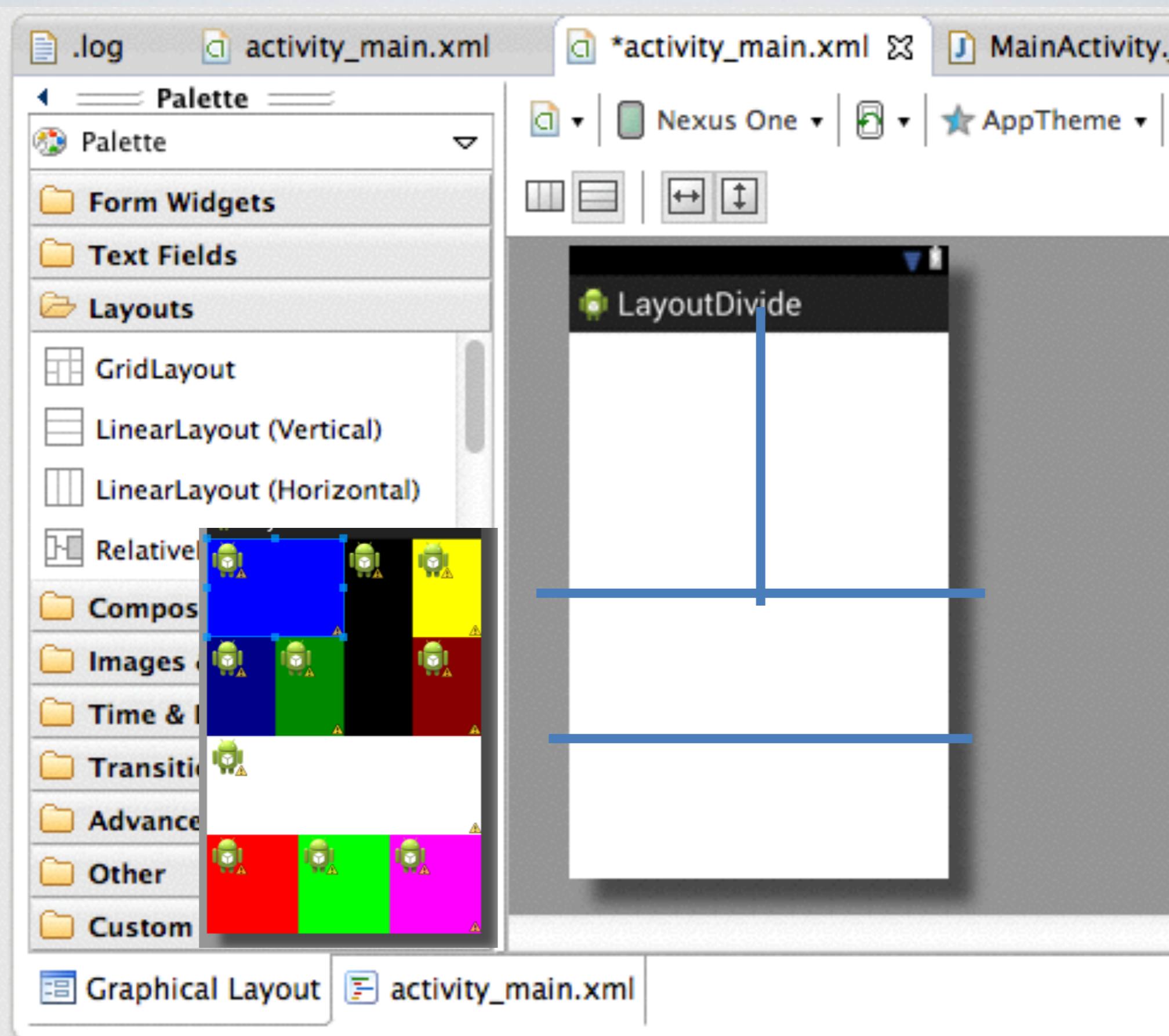


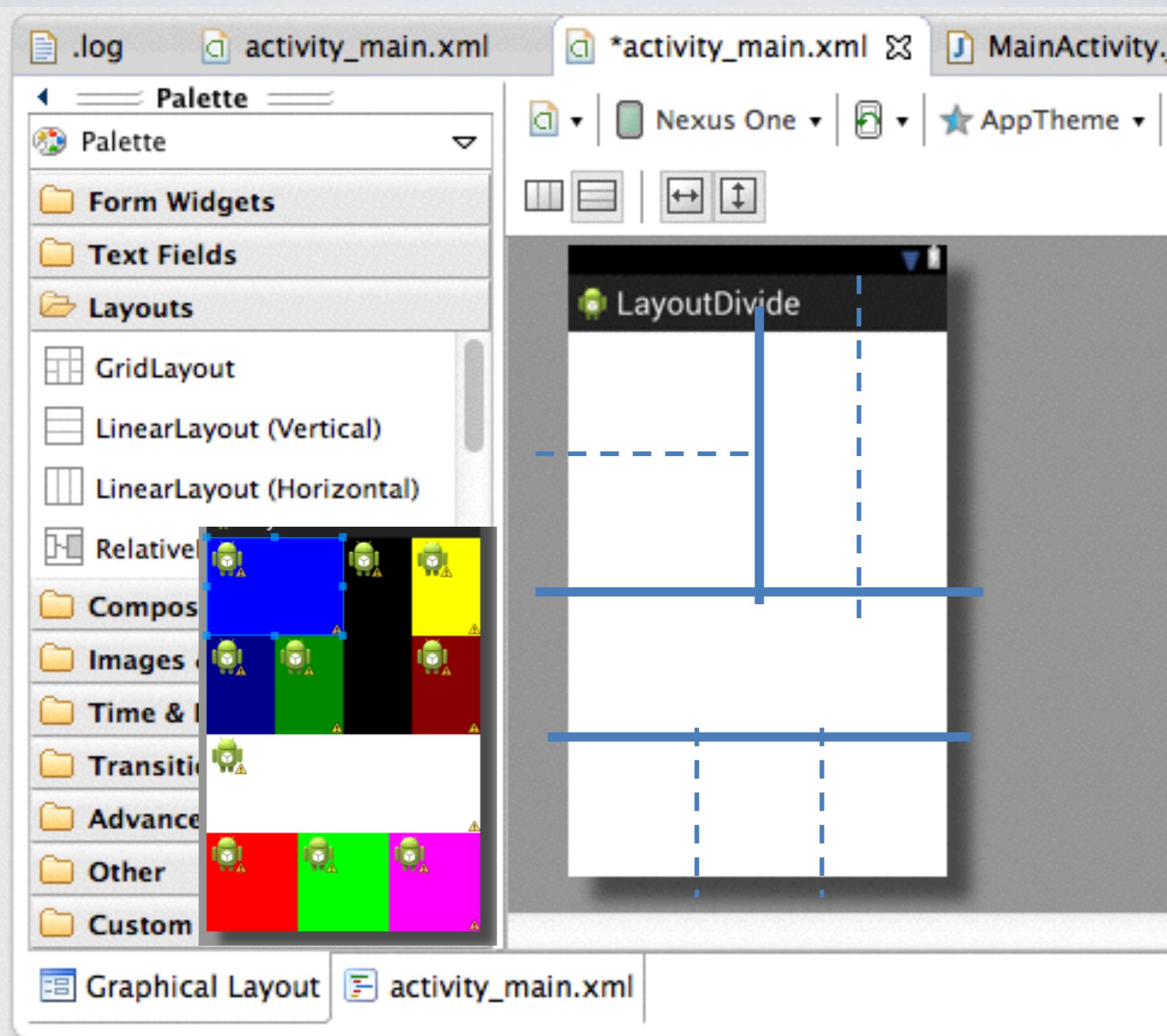


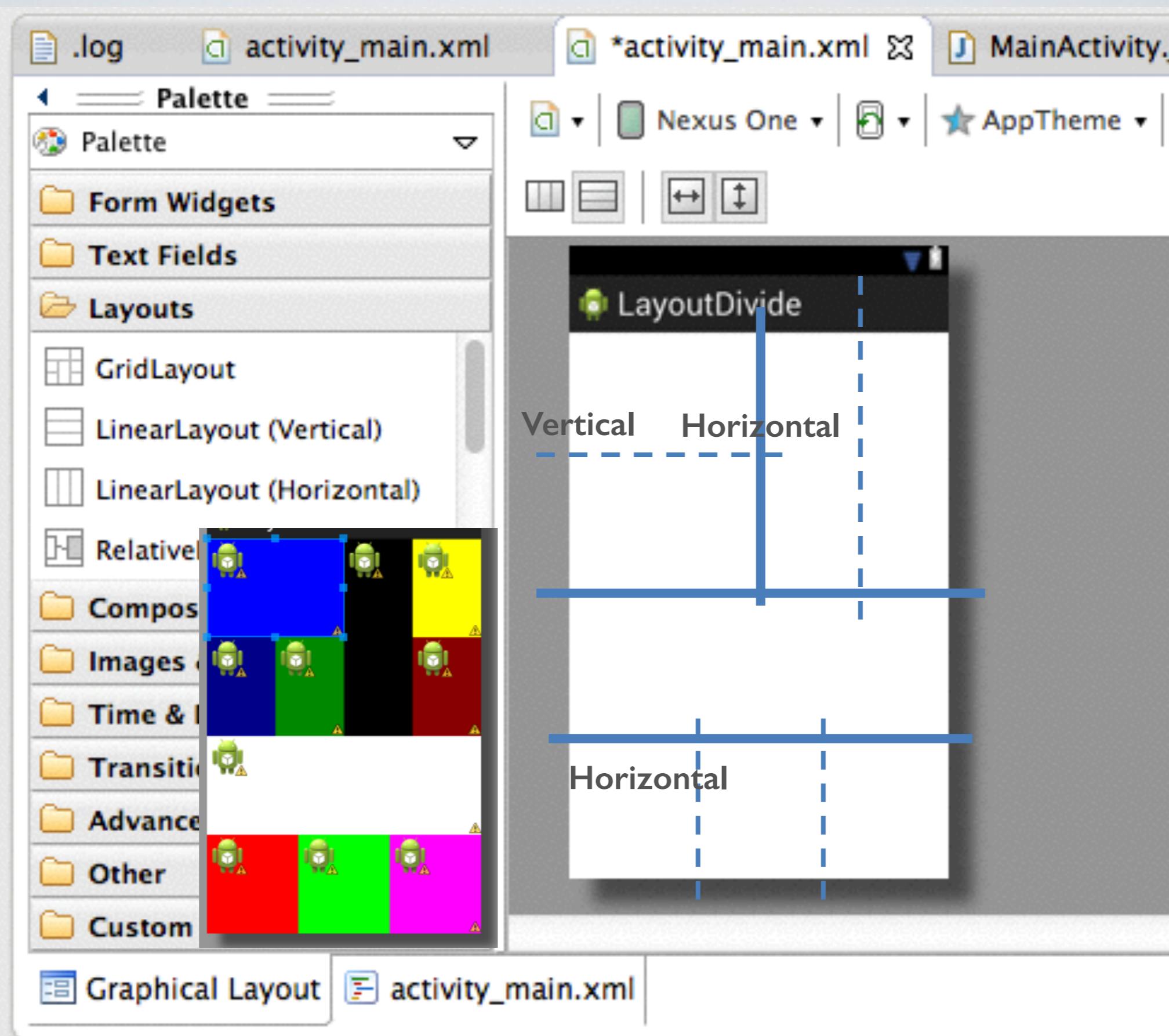


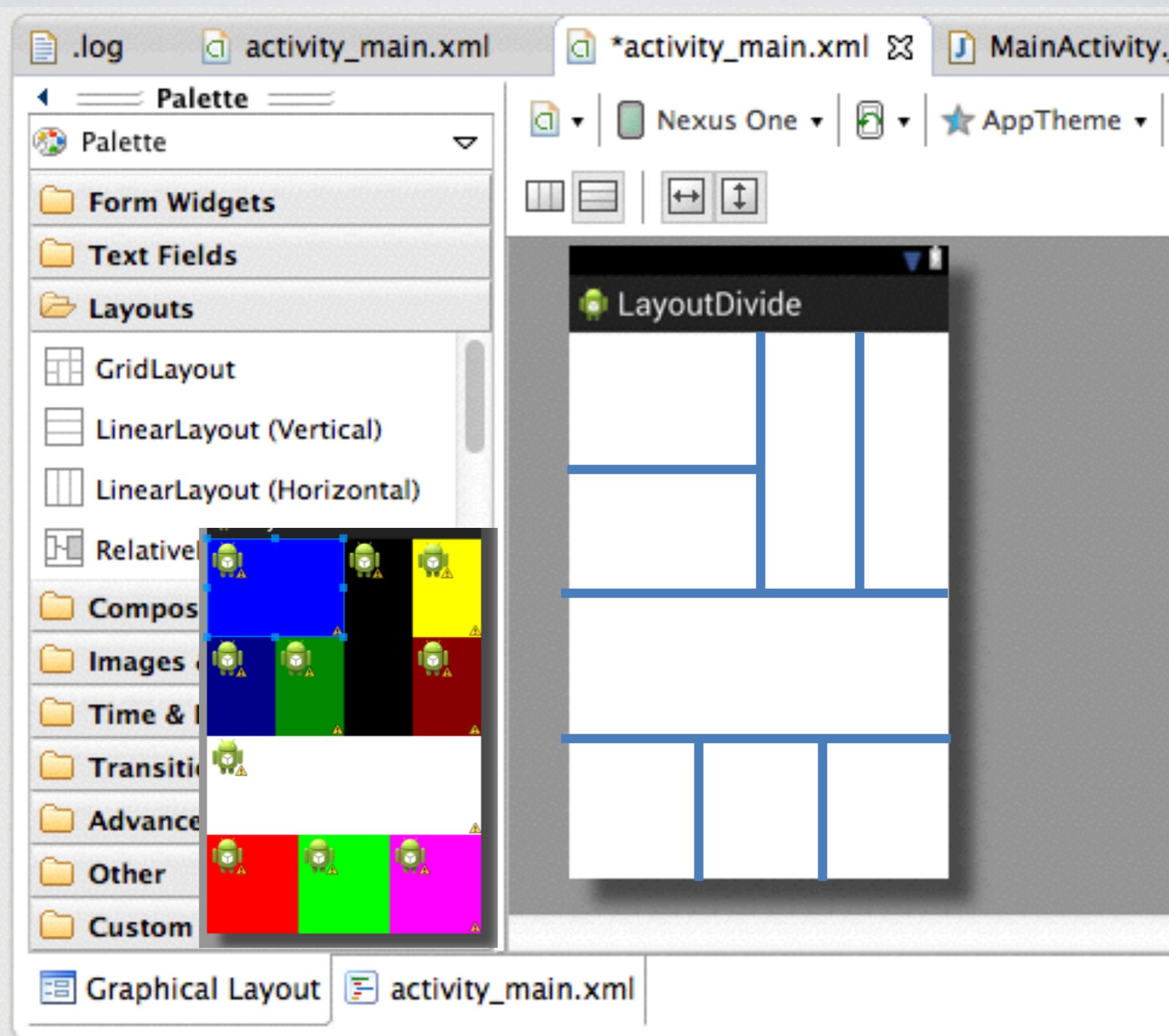


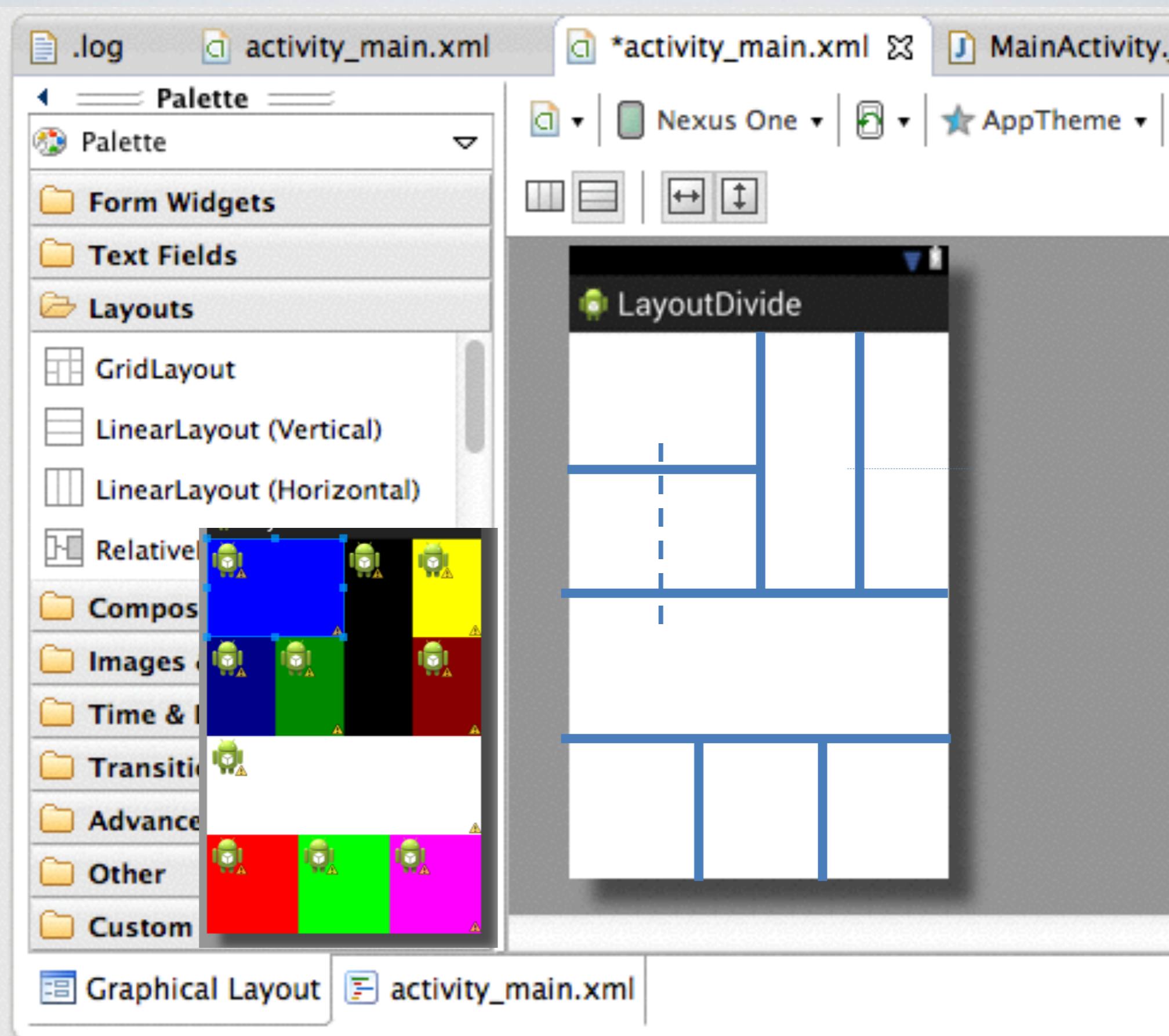


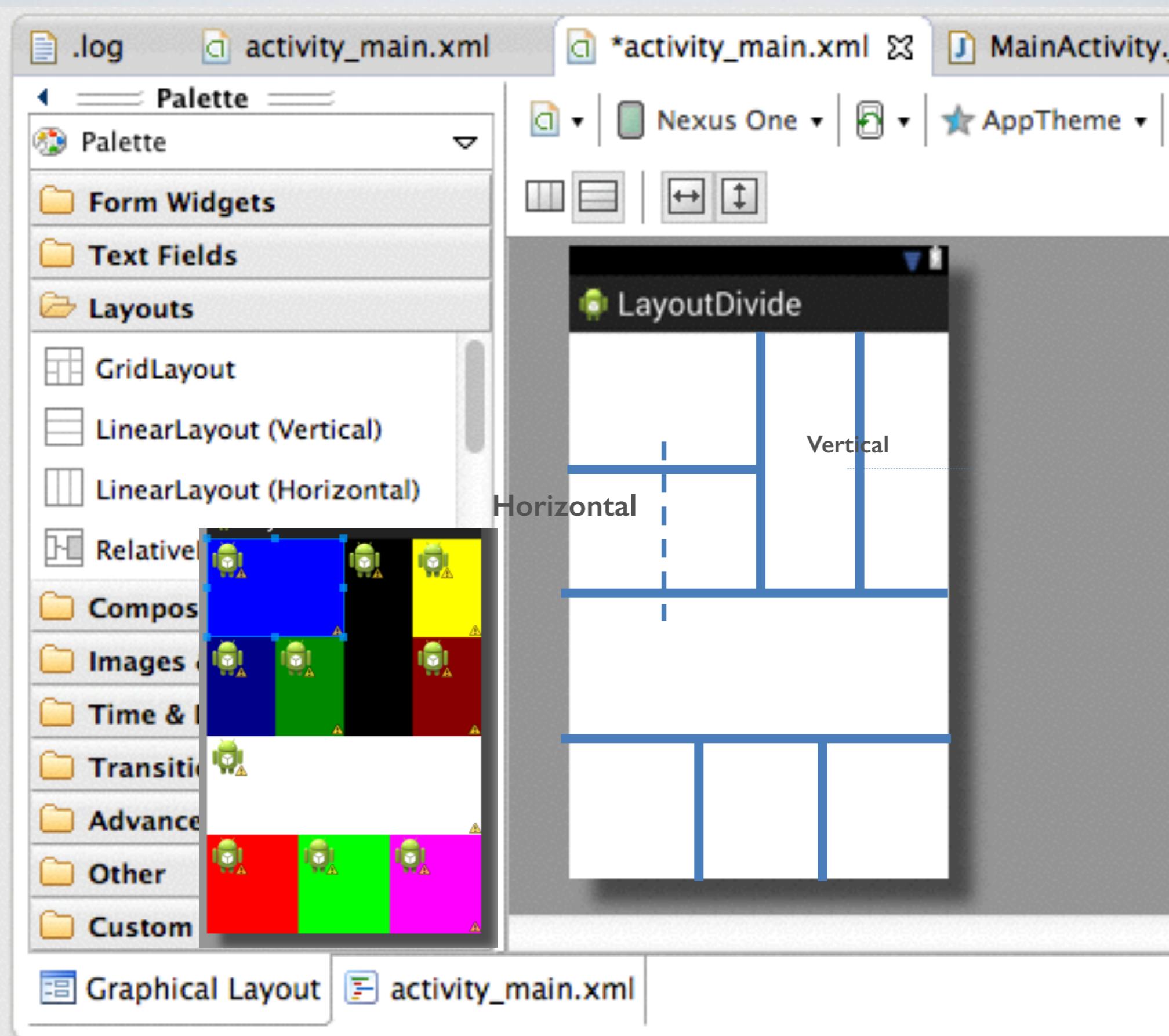


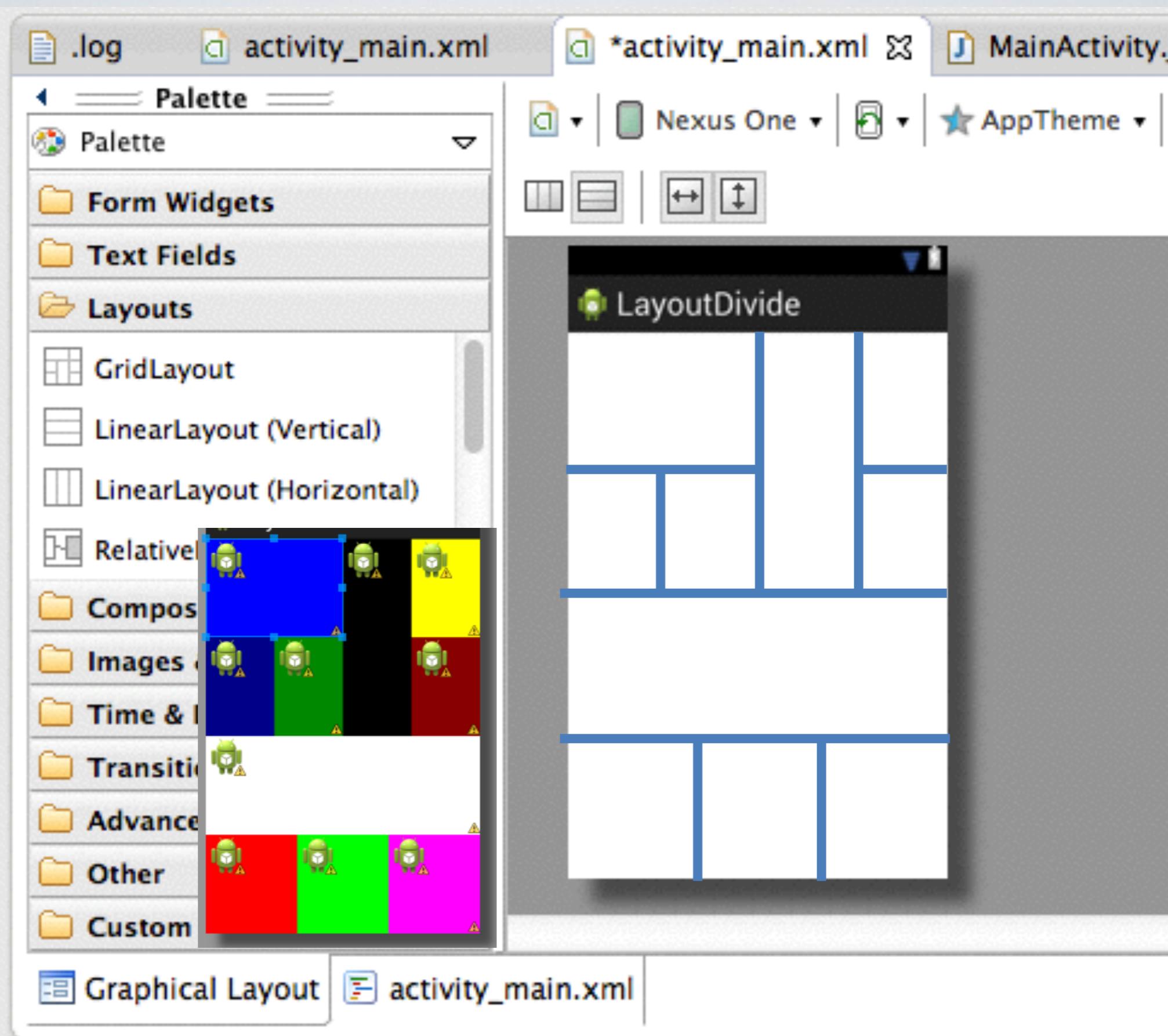














https://github.com/yangjeewoong/android_LayoutDivide/

문서 버전

1.0 초안

2.0 DP/PX 추가

양지웅
정문철