# Android Application Development

## Resources System

# What is Resources?


It takes more than just code to build a great app. Resources are the additional files and static content that your code uses, such as bitmaps, layout definitions, user interface strings, animation instructions, and more.

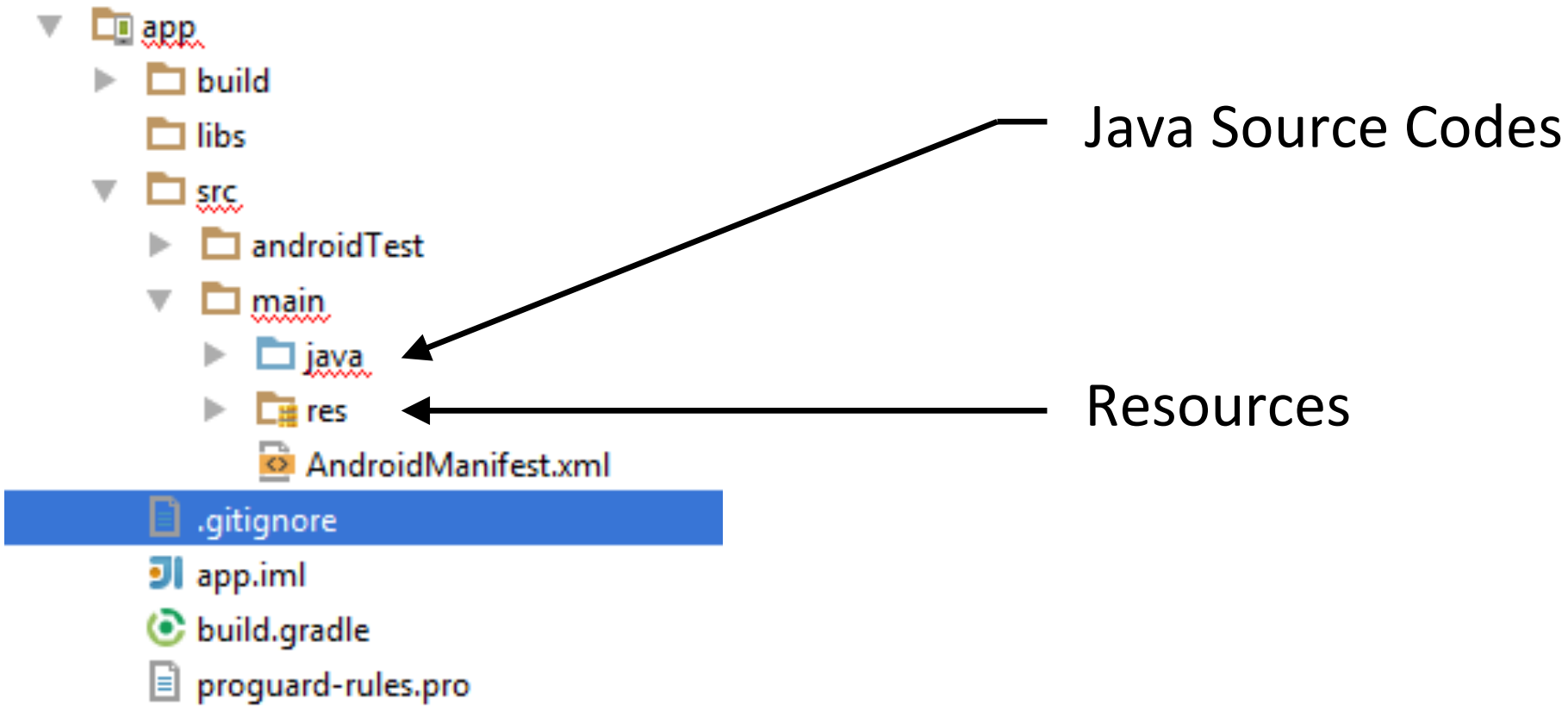
# What is Resources?


## A Storeroom

contains pieces of data needed in application

- Image (jpg, png)
- Layout (xml)
- etc.

# What is Resources?



# What is Resources?

```
package com.example.helloworld;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;


public class MainActivity extends Activity {

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

        TextView tvHello = (TextView) findViewById(R.id.tvHello);
        tvHello.setText("Yeah");
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
```

**Source Code**

+

```
<TableLayout xmlns:android="http://schemas.android.com/apk/
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1"
    tools:context=".MainActivity">

    <TableRow
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        >
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_launcher"
            />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/hello_world"
            android:padding="16dp"
            android:background="#ffaaaa"
            />
```

+

+

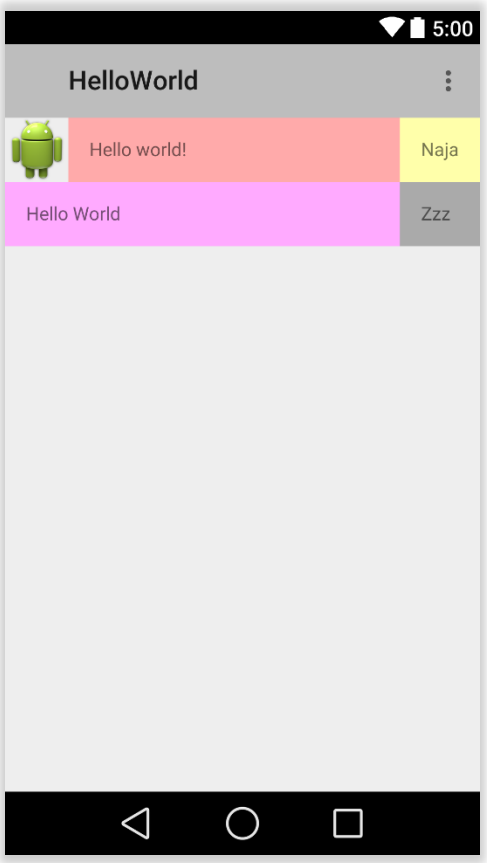
```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="yeah">Yeah</string>

</resources>
```

**Resources**

=




**Running Application**

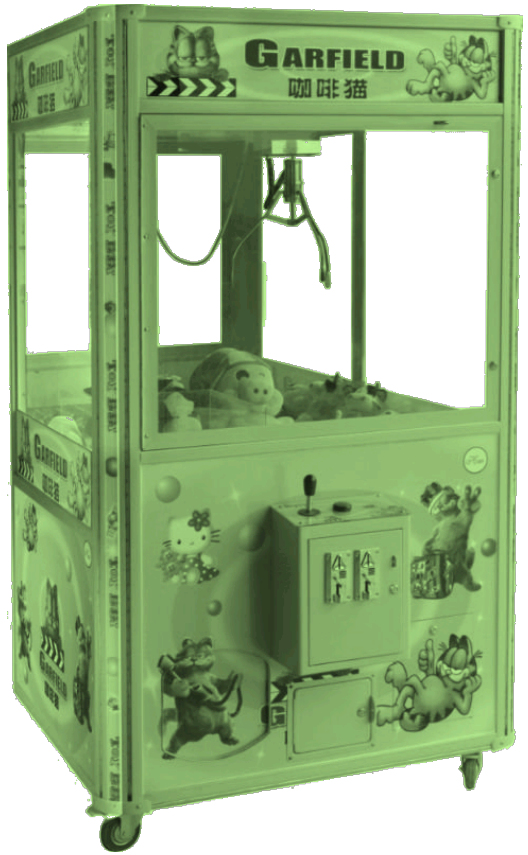
# What is Resources?


## How does **Resources System** look like?

You should always externalize resources such as images and strings from your application code, so that you can maintain them independently

# Resources Sources


App-Specific Resources




System Resources

# What is Resources?

```
public class MainActivity extends Activity {

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

        TextView tvHello = (TextView) findViewById(R.id.tvHello);
        tvHello.setText("Yeah");
    }
```

**It works**

```
public class MainActivity extends Activity {

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

        TextView tvHello = (TextView) findViewById(R.id.tvHello);
        tvHello.setText(R.string.yeah);
    }
```

+

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

    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="yeah">Yeah</string>

</resources>
```

**It’s awesome**

# Resources Type

## Drawable
Something that can be drawn on screen

## Layout
XML contains Layout hierarchy/ structure

## Menu
Context menu, Popup menu

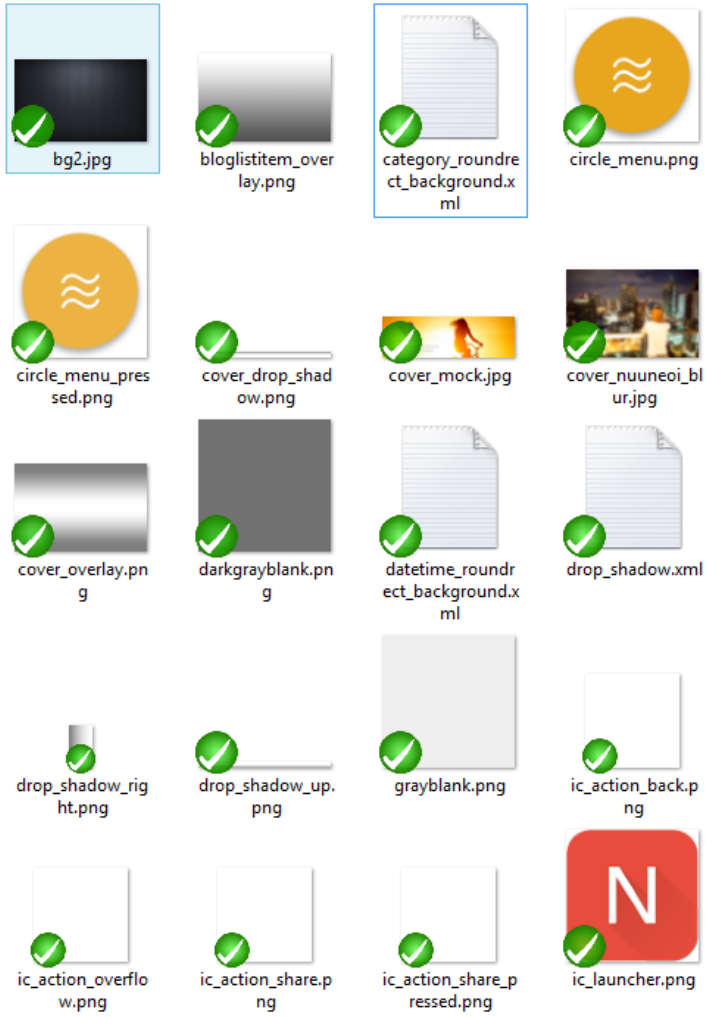
## Values
Strings, Colors, Dimension, etc.

## Animation
XML contains Property animation or View animation

## etc.
Almost everything but Java source codes

# Resources Type: Drawable

- A drawable resource is a general concept for a graphic that can be drawn to the screen
- folder **`res/drawable`**
- Have A LOT to be learned, be prepared …



# Playaround: ImageView

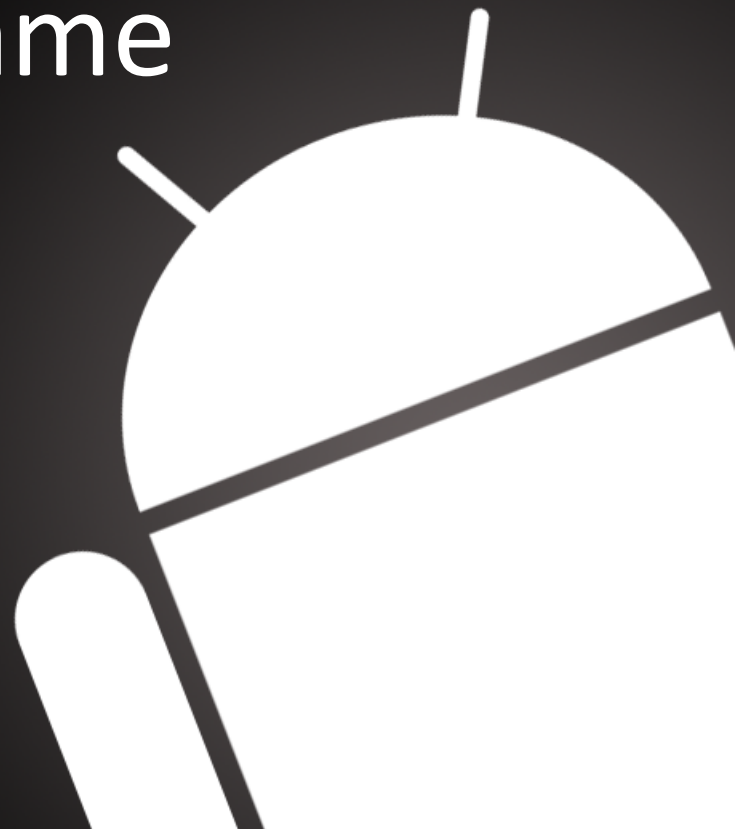
… Play Play Play …

# How to refer resource from another resource?

… Play Play Play …

## Rule 11:

Each resource folder couldn’t have any child folder. Manage filename wisely.



# The R class: How are Resources compiled?

Filename will be simply automatically turned into a int variable name under

<package_name>.R.<resource_type>

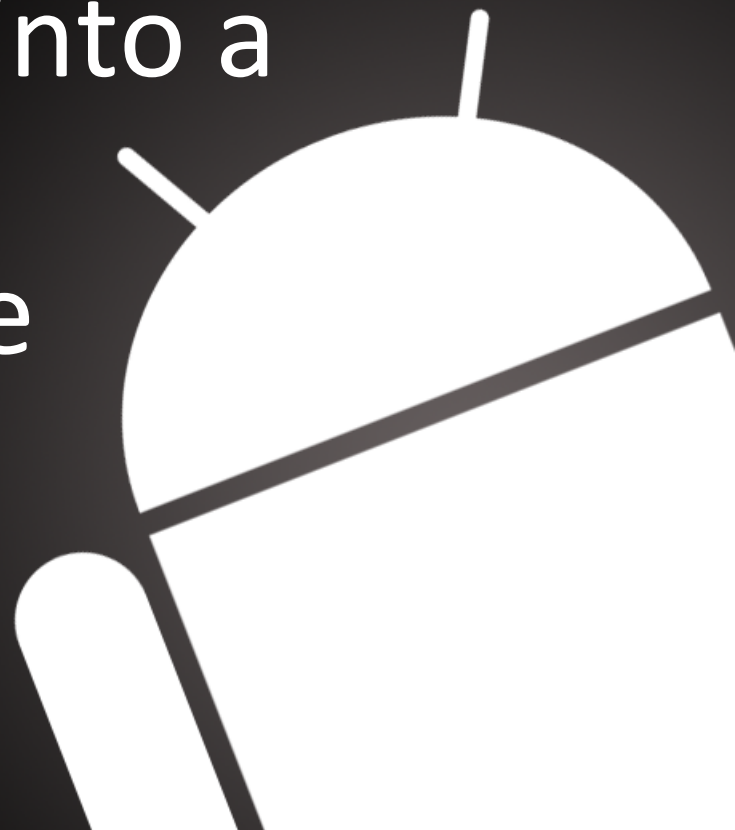
We call it **“Resource ID”**

Remember !

R.drawable.ic_launcher

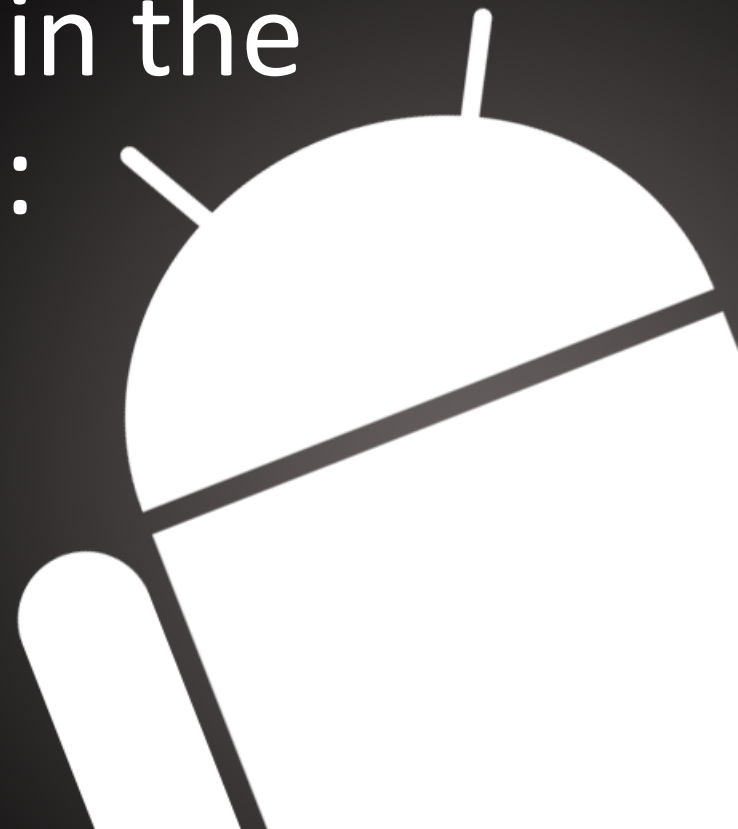
## Rule 12:

Filename (ext not included) in resource folder will be turned into a variable name in Java. So you can’t use some character in the filename (for example, - +)



## Rule 13:

Never has the same filename in the same folder although they are in the different file type (for example: icon1.jpg, icon1.png)



# Resources Type: Layout

- A layout resource defines the architecture for the UI in an Activity or a component of a UI.
- Folder **`res/layout`**

```
<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:stretchColumns="1"
    tools:context=".MainActivity">

    <Button
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:text="Hello world!"
        />

</RelativeLayout>
```

# The R class: How are Resources compiled? #2

Every single resource will be put in the same place with different assigned Resource ID automatically set by Android compiler

# android:id

- Inside the layout xml file, you will find something like

`android:id="@+id/xxxx"` or `android:id="@id/xxxx"`

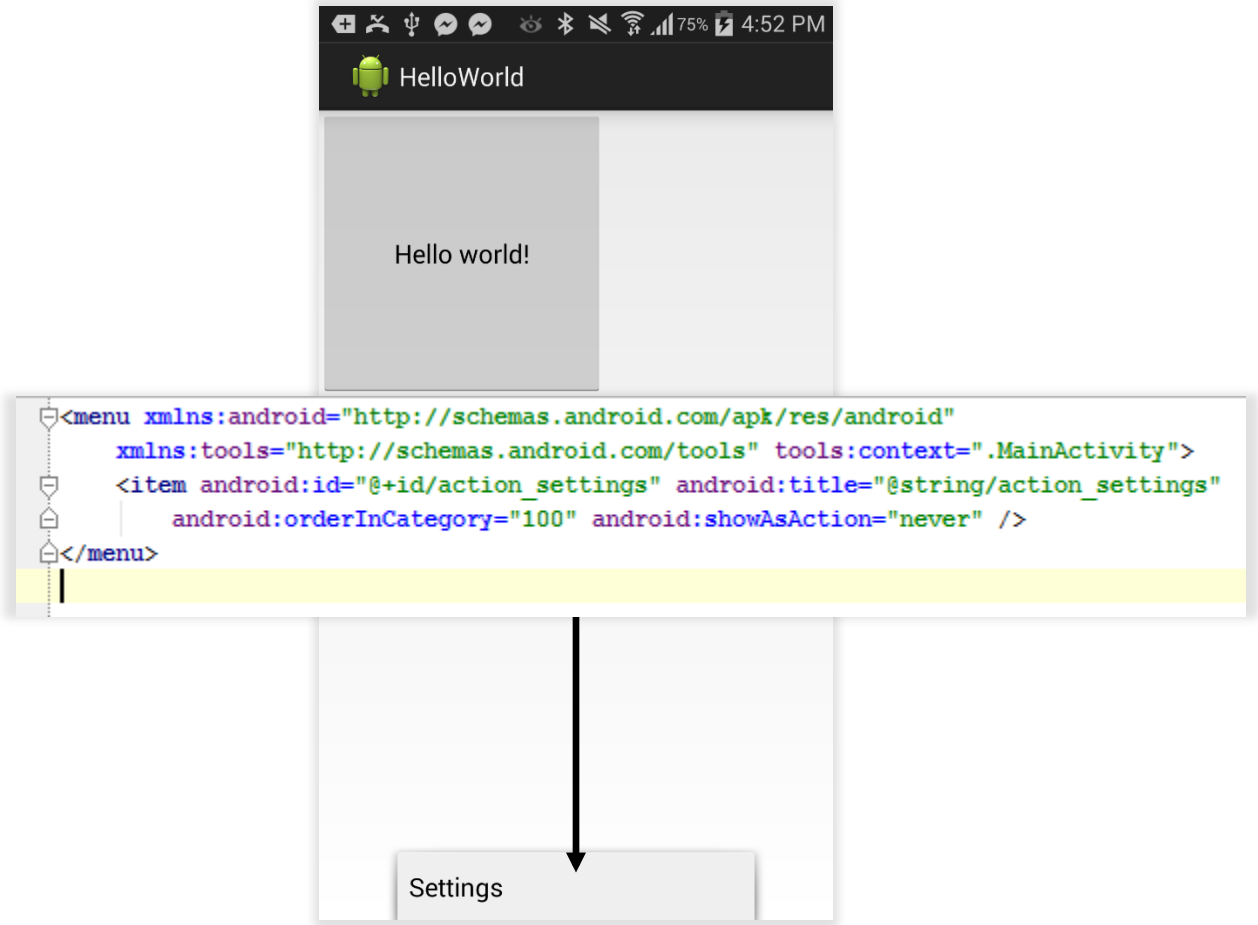
- It is simply a Resource ID referring to a View inside a Layout
- You have no need to assign it for every single element, just one you need to reference

## Rule 14:

Don’t use the same ID in the single file, but feel free to use in any other file

# Resources Type: Menu

- A menu resource defines an application menu (Options Menu, Context Menu, or submenu) that can be inflated with MenuInflater.
- Folder **`res/menu`**
- Learn more about Menus in Part 3



# Resources Type: Values

- Folder **`res/values`**
- Be careful, Resource ID is not R.values.xxxx
- Look into each one by one
  - Dimension
  - String
  - Colors
- You will learn more by practice all along this class

# Resources Type: Anim

- View Animation
- Creates an animation by performing a series of transformations on a single image or creates an animation by showing a sequence of images in order
- Folder **`res/anim`**
- Too complicated to talk about this now. See you later.

# Resources Type: Animator

- Property Animation
- Creates an animation by modifying an object's property values over a set period of time
- Folder **`res/animator`**
- Too complicated to talk about this now. See you later.

# Resources Type: The Rest

- Not so important
- But once you know the concept of Resource, you will be able to learn more about another type of resource by yourself in a minute

# Configuration Qualifier

- **ALERT: VERY IMPORTANT !!!**
- Very smart resource selection mechanic in Android
- Folder format

```
<resources_name>-<qualifier>
```

- Can be applied on any resource type
- For example

```
res/drawable-xhdpi
res/layout-xxhdpi
```

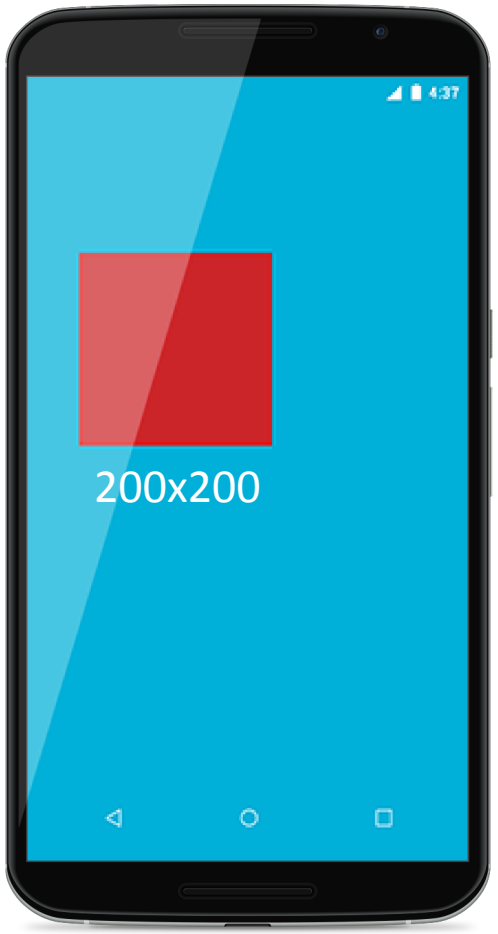
# Configuration Qualifier Type

- **dpi** – ldpi, mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi, nodpi, tvdpi
- **Screen size** – small, normal, large, xlarge
- **Width** – w<N>dp (for example, w820dp)
- **Smallest Width** – sw<N>dp (for example, sw600dp)
- **Screen aspect** – long, not long
- **Platform version** – v<N> (for example, v21)
- And a lot more
- So you might see something like `drawable-notlong-land-xxxhdpi-xlarge-v21` …
- Have to do it by order, cannot shuffle the position
- See more at http://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources

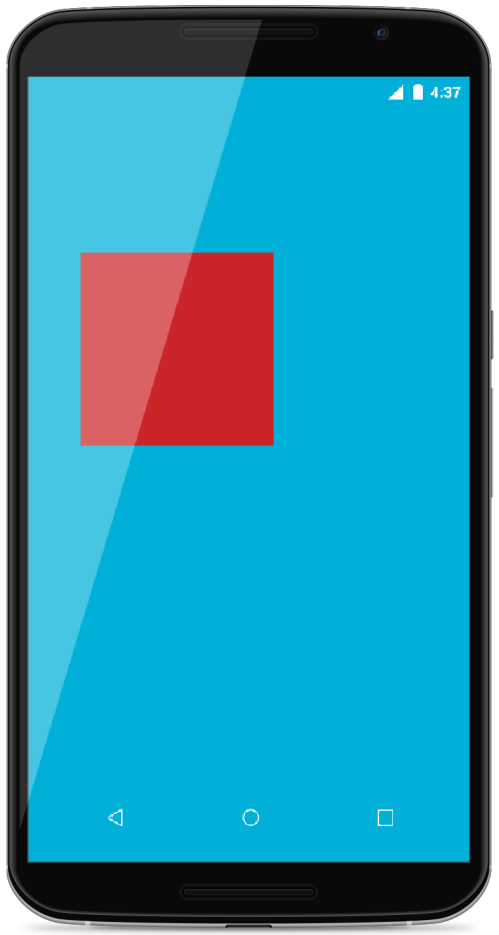
# Configuration Qualifier: Select by qualifier

- If hardware's configuration matches the qualifier, resource inside that folder will be used
- See an example

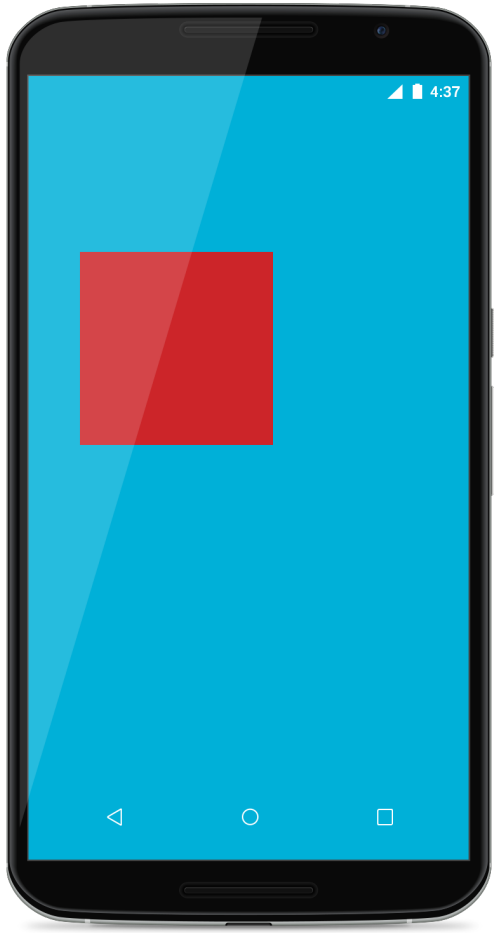
# Configuration Qualifier




480x800




720x1200




960x1600

# Configuration Qualifier: Auto Scale

- But if the folder doesn't existed, it will look into the nearest configuration and scale in the same way as dp !

- For example, if image is decoded in **xhdpi** (x2) has 100 pixels width, when it is decoded in **mdpi** (x1), its width will be at 50 pixels

- In the other words, if you put 50x50 pixels image in drawable-mdpi, you will get the same size (but not quality) as you put 100x100 pixels image in drawable-xhdpi

- See an example

# Configuration Qualifier: List of Qualifier

- Very smart resource selection mechanic in Android
- Folder format

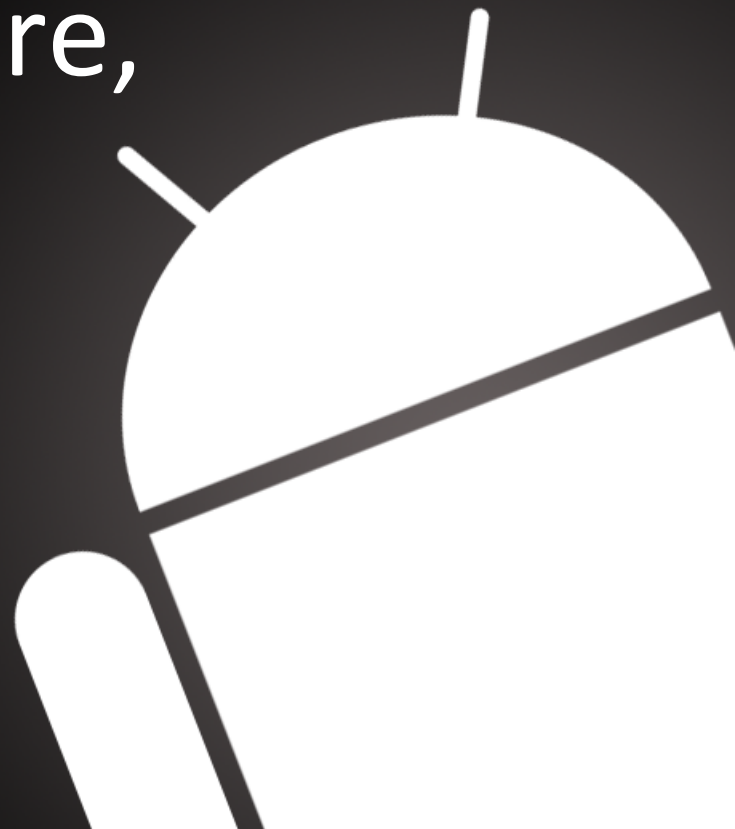
```
<resources_name>-<qualifier>
```

- For example

```
drawable-xhdpi
drawable-xxhdpi
```

## Rule 15:

With knowledge about dp and Configuration Qualifier (and sure, with some practicing), you will 80% win over Fragmentation !



# Do we need to define every single qualifier?

**NO and DON’T !**

# Best Practices: Drawable

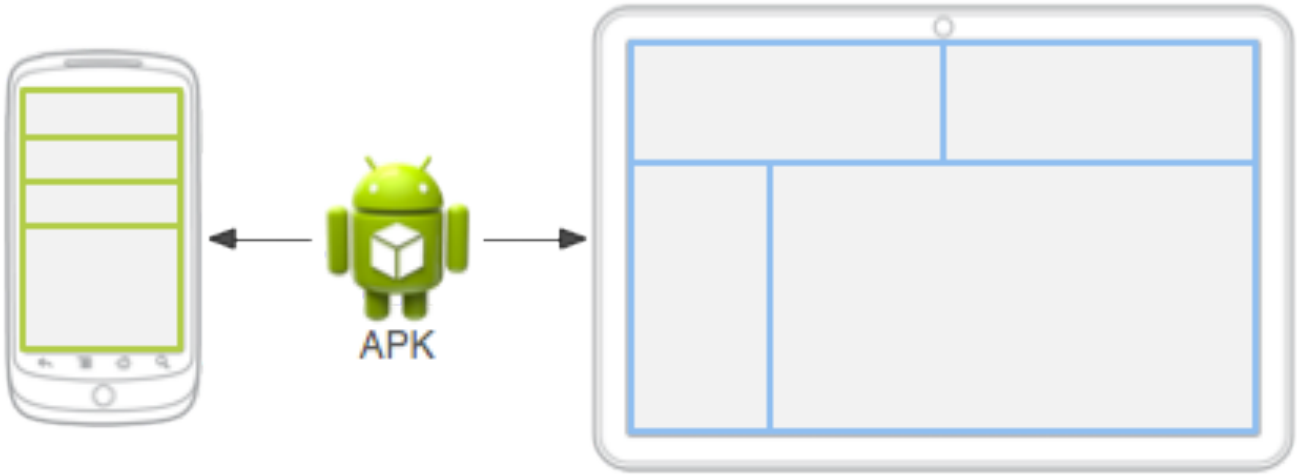
- Put icon inside every single dpi qualifier in **mipmap**
- Don't put *image files* inside `res/drawable` and/or `res/drawable-nodpi` *unless* you know what are you doing
  - `res/drawable` is treated like `res/drawable-mdpi`
  - Image put inside `res/drawable-nodpi` will be not scaled
- Put Drawable Resources inside just a single folder and let it be scaled automatically
  - `res/drawble-xhdpi` is recommended (just by me)
- (Cont.) *unless* you really need to customize for some screen (and you will need it)
  - Mobile/Tablet (because I will let you do)
  - Support for some high resolution screen (2K for example)

# Best Practices: Layout

- Since dp is all the same in any single dpi, so you have no need to put anything after `layout` folder
  - Just use **`res/layout`**
- Unless you need to customize the screen for Landscape Handset or for Tablet
  - Use **`res/layout-land`** for Landscape
  - Use **`res/layout-sw600dp`** for Tablet screen
  - Use **`res/layout-sw600dp-land`** for Landscape Tablet screen
  - And it is mostly what we do for layout already …

# Best Practices: Layout

- And you will need to do that as well … (separated Layout for Tablet)



## Rule 16:

Be lean, don’t have too many qualifiers. It causes more problem rather than helping.

# How to access resource from Java?: The Context

"Interface to global information about an application environment"

If you want to access to anything

- Application's Resources
- Launch Activity
- Send Broadcast
- Receive Intent
- Get Screen Resolution
- A lot !!

* Context is one of the thing you will see the most in Android App Development, be used to it

# How to access resource from Java?

- `context.findViewById(…)`
- `context.getResources(…)`
- Curious why do we can call those methods without calling through context?
- And what's about `setText(R.string.xxx)` ?

… Play Play Play …

## Rule 16:

Context is always be used everywhere. Have a date with it everyday from now on.