



BINAR
ACADEMY

Android Studio sebagai IDE

Silver - Chapter 2 - Topic 2

**Selamat datang di Chapter 2 Topik 2 online course
Android Developer dari Binar Academy!**





Hi Teman-Teman 🙋

Pada topik sebelumnya, kita sudah belajar konsep pemrograman dengan **prinsip SOLID**.

Nah, di **Topik 2** ini kita mulai menyentuh tools andalan kita, yaitu **Android Studio**. Kita akan kenal lebih jauh dan coba praktik sedikit dengan tools andalan kita nii.

Yuk, lanjut!



Detailnya, kita bakal bahas hal-hal berikut ini :

- Pengenalan Android Studio
- Mengenal struktur dalam Android Studio
- Menggunakan Shortcuts Android Studio
- Logging dan Debugging





Belajar **Android Studio** sangat penting bagi kamu yang mau terjun ke dunia pengembangan aplikasi Android.

Mengapa? Karena **Android Studio** adalah satu-satunya aplikasi yang didukung oleh **Google** untuk tujuan tersebut.



“Emang seperti apa sih aplikasi Android Studio?”

Nah, biar makin gampang belajarnya,
mari kita *flashback* ke materi di
chapter sebelumnya!



Apa itu Android Studio?

Android Studio adalah ***Integrated Development Environment (IDE)*** resmi untuk pengembangan aplikasi **Android berdasarkan IntelliJ IDEA**.

Untuk mendukung pengembangan aplikasi dalam sistem operasi Android, Android Studio menggunakan *build system*, *emulator*, *template code*, dan integrasi ke Github berbasis Gradle.

Setiap project di Android Studio memiliki satu atau lebih modalitas dengan *source code* dan *file resources*. Modalitas ini meliputi module aplikasi Android, module Library, dan Google App Engine module.

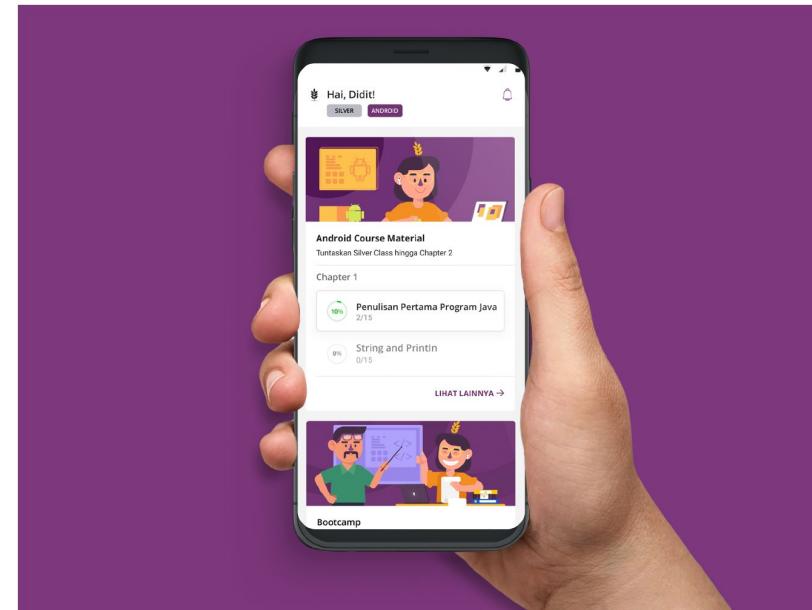




Android Studio menggunakan fitur *Push Instans* untuk mengupload perubahan kode dan *resources* ke aplikasi yang sedang berjalan.

Editor code membantu *developer* dengan menulis kode dan menawarkan penyelesaian, perubahan, serta analisis code.

Aplikasi yang dibangun di Android Studio kemudian dikompilasi ke dalam format APK untuk diserahkan ke Google Play Store.

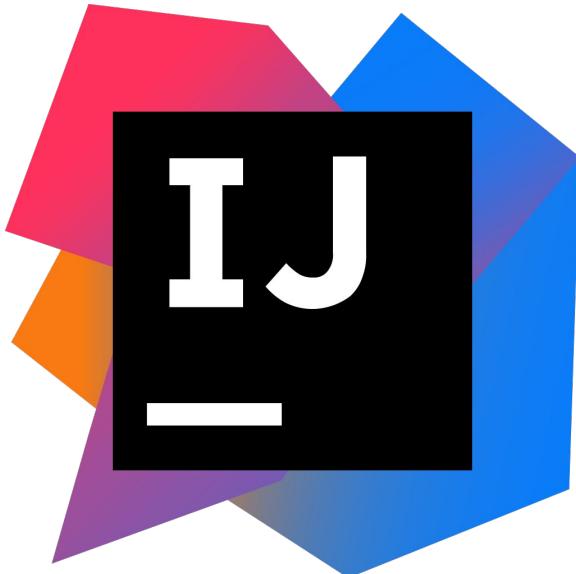




Android Studio tersedia untuk platform *desktop* Mac, Windows, dan Linux.

Android studio menggantikan *Eclipse Android Development Tools* (ADT) sebagai IDE utama untuk pengembangan aplikasi Android. Android Studio dan *Software Development Kit* (SDK) dapat diunduh langsung dari Google.

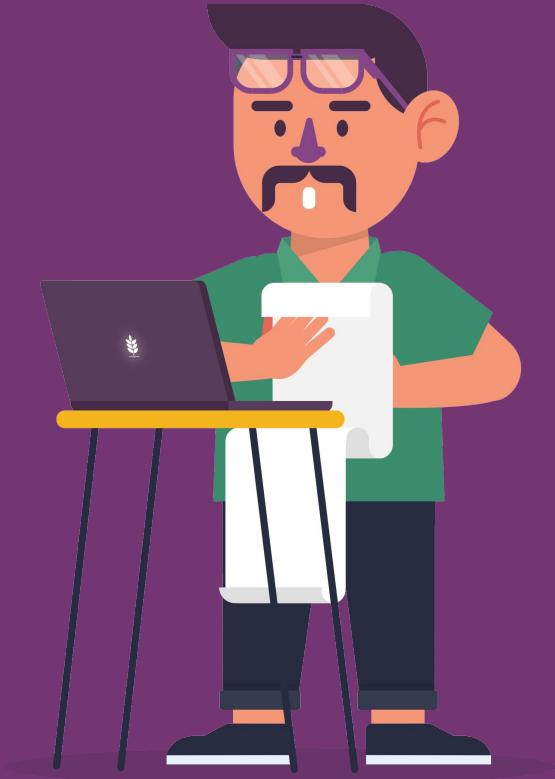




Sebelumnya, kita telah menggunakan IntelliJ IDEA sebagai IDE pada pembelajaran dasar Java dan Kotlin.

Supaya nggak nang ning nung dengan Android Studio, ada baiknya kita mengetahui strukturnya terlebih dahulu.

Tapi, jangan khawatir! Belajarnya nggak akan sesulit itu karena IntelliJ IDEA dan Android Studio nggak jauh berbeda.



**Sebagai awal petualangan,
kita buat dulu yuk **project Android
Studio** pertama kita!**



Install Android Studio

Untuk install Android Studio, silahkan ikut link di bawah ini untuk mulai install-nya~

Windows

[Click Here](#)

Mac

[Click Here](#)

Linux

[Click Here](#)

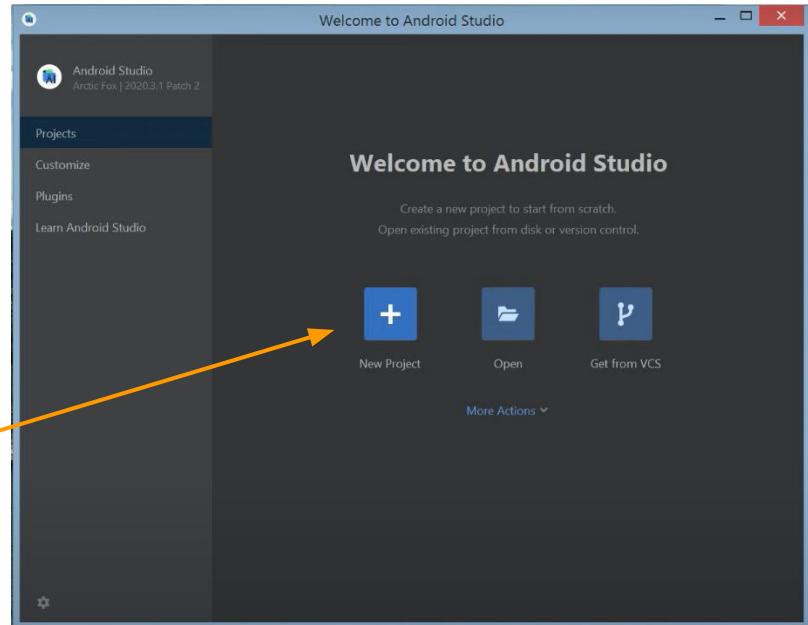


Membuat Project Android Studio

Kalau kamu belum pernah membuat *project* di Android Studio sebelumnya. Kita akan disambut dengan tampilan seperti ini:

Jika kita ingin membuat sebuah project baru, kita dapat menekan menu
"Start a new Android Studio project".

Jendela Welcome Screen





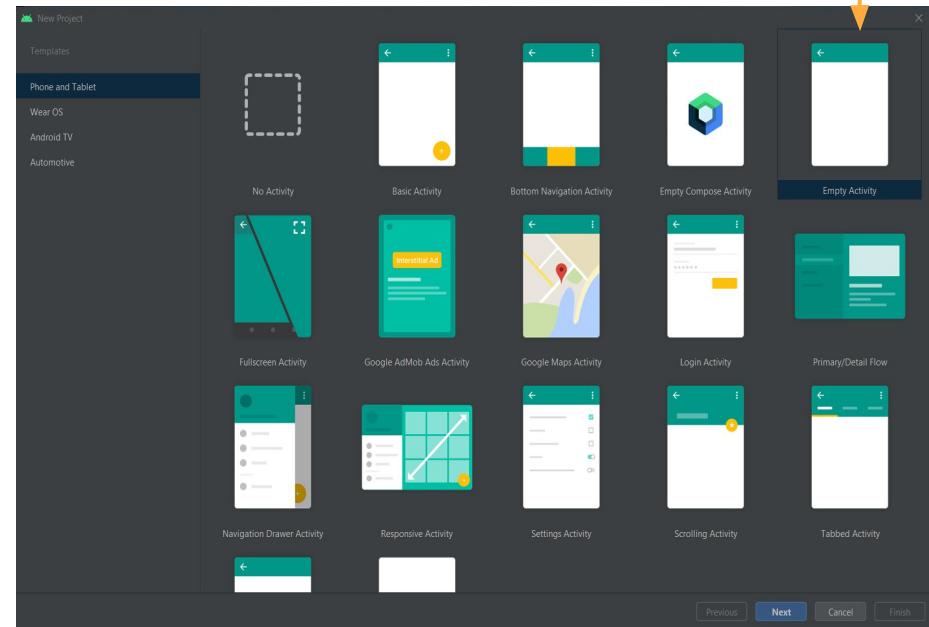
Untuk permulaan kita bisa menggunakan **Empty Activity**.

New Project

Setelah kita menekan menu "**Start a new Android Studio project**", kita akan masuk ke jendela baru untuk memilih spesifikasi dalam pembuatan project baru.

Dalam jendela ini, kita bisa memilih *template* dari Aplikasi yang nantinya akan kita buat. Terdapat beberapa template yang telah disediakan Android Studio yang bisa kita gunakan.

Setelah memilih **Empty Activity**, tekan "**Next**" untuk masuk ke jendela baru.

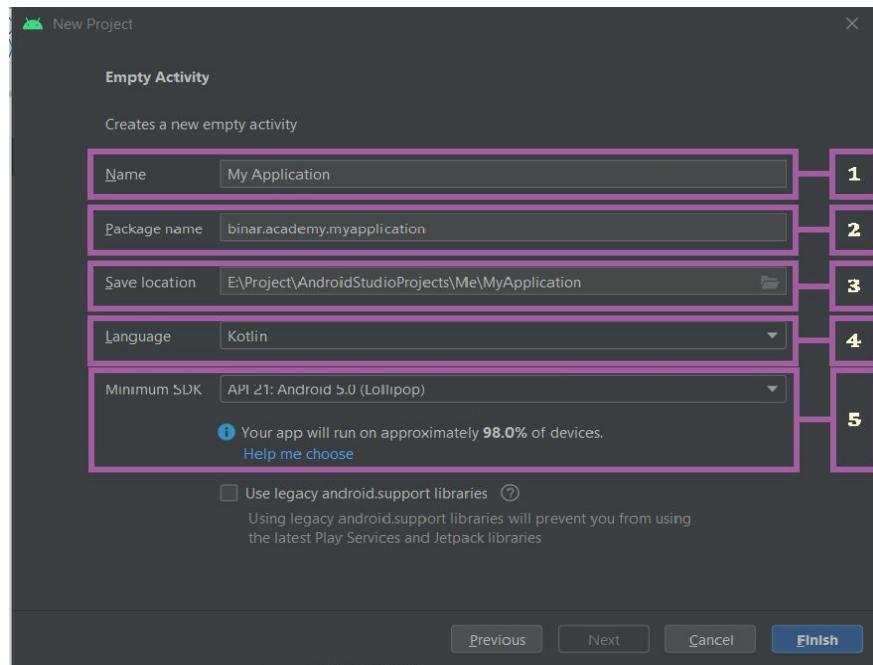




Membuat Project Android Studio - New Project Wizard

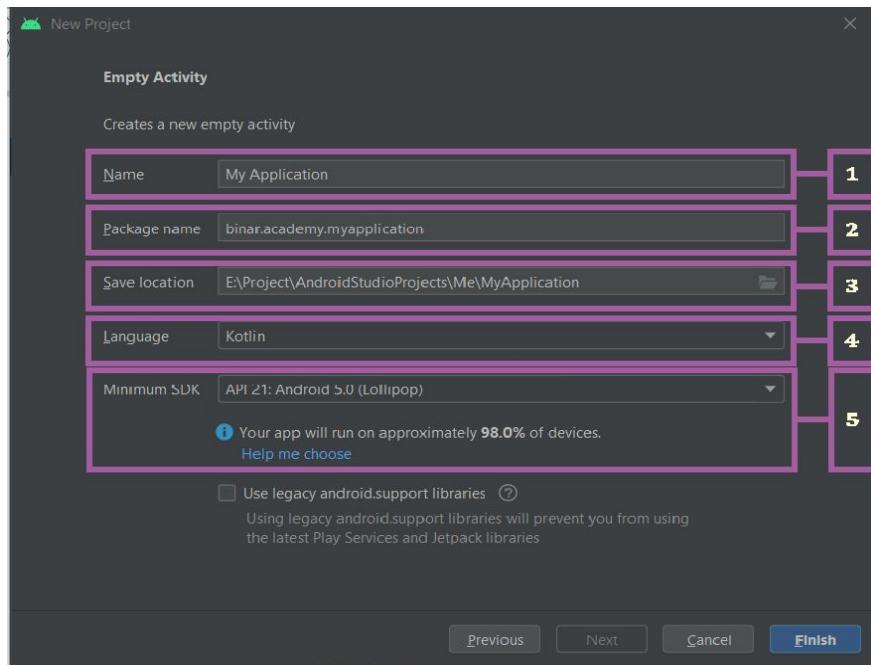
Pada jendela ini, kita ditugaskan untuk melakukan konfigurasi lebih lanjut tentang project kita. Penjelasannya sebagai berikut:

- **Name:** Nama untuk project yang akan kita susun.
- **Package name:** Nama package digunakan untuk identifikasi unik dari aplikasi kita ketika akan di-publish. (Cont.)





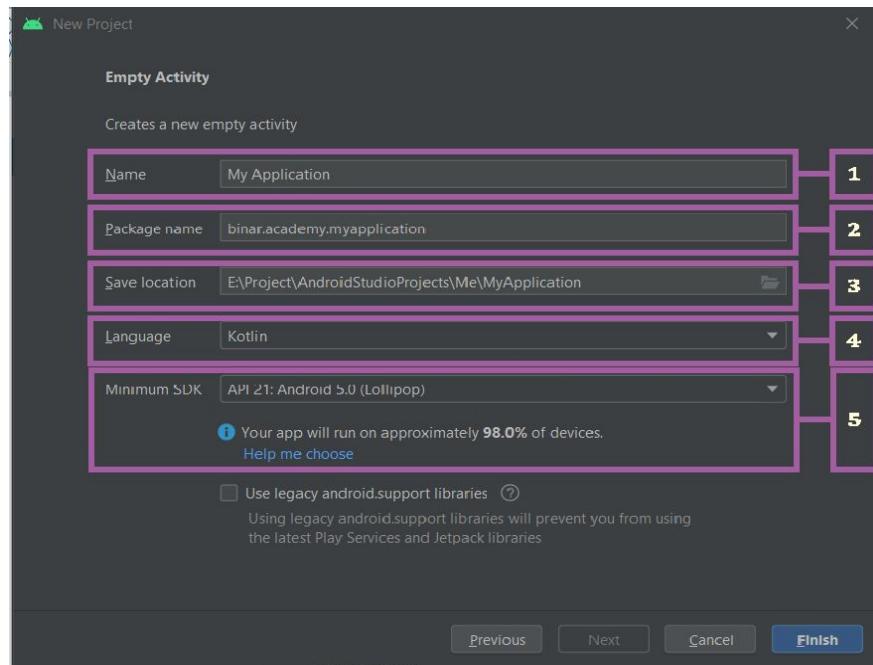
- **Save location:** Menentukan lokasi untuk menyimpan *project* kita.
- **Language:** Bahasa pemrograman yang akan kita gunakan untuk membuat *project* Android. Pilihannya ada dua: Java atau Kotlin.
- **Minimum API level:** Minimum API yang akan digunakan untuk *project* kita. Ia berfungsi untuk membatasi penggunaan API pada sebuah aplikasi.





Setelah melengkapi semuanya, klik “**Finish**”. Maka, akan berjalan proses gradle sync. Dan, project Android Studio kita sudah siap!

Dengan menekan tombol “Finish”, interface dari Android Studio akan terbuka.

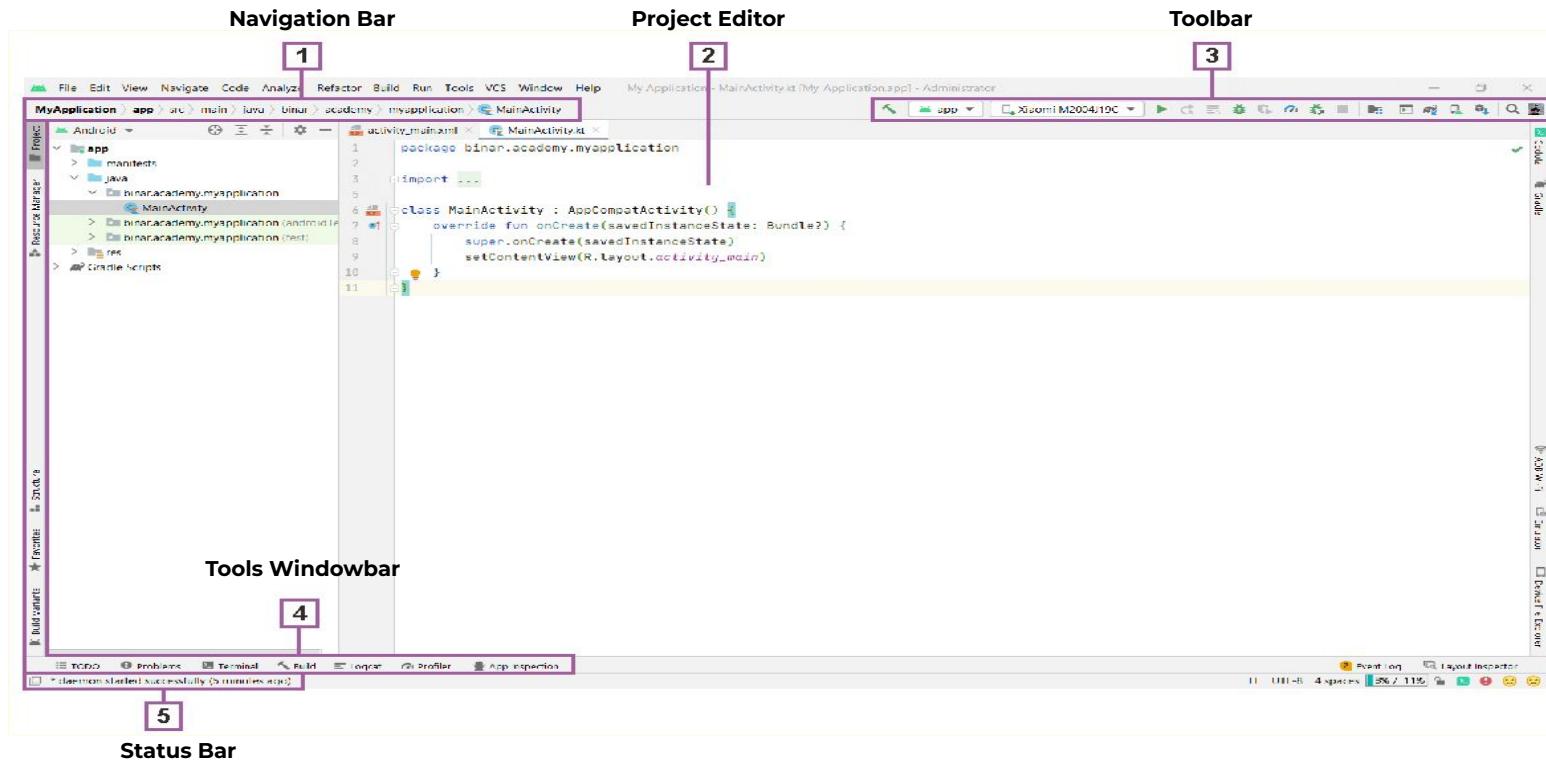




Nah, supaya kita tahu bagian-bagian dan kegunaan dari setiap komponen, sekarang kita kenalan sama **Tampilan Utama Android Studio!**



Interface Android Studio



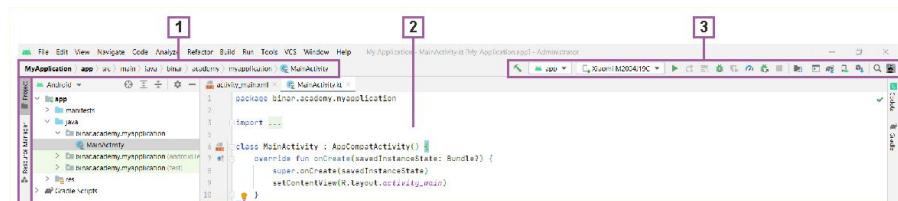


Ini lah tempat kita melakukan pembuatan program.

Tampilan ini mungkin akan berbeda di tampilan layar kita masing-masing karena perbedaan konfigurasi dan versi Android Studio.

Terdapat beberapa bagian yang terdapat pada Android Studio. Mari kita bongkar satu persatu!

Navigation Bar Project Editor Toolbar



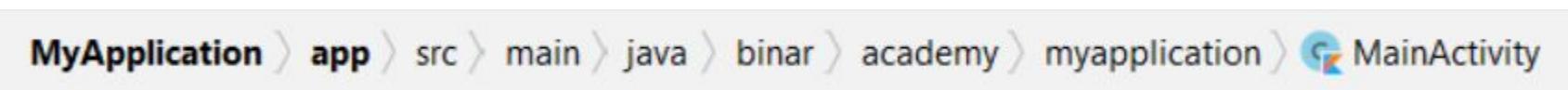
Tools Windowbar



Status Bar



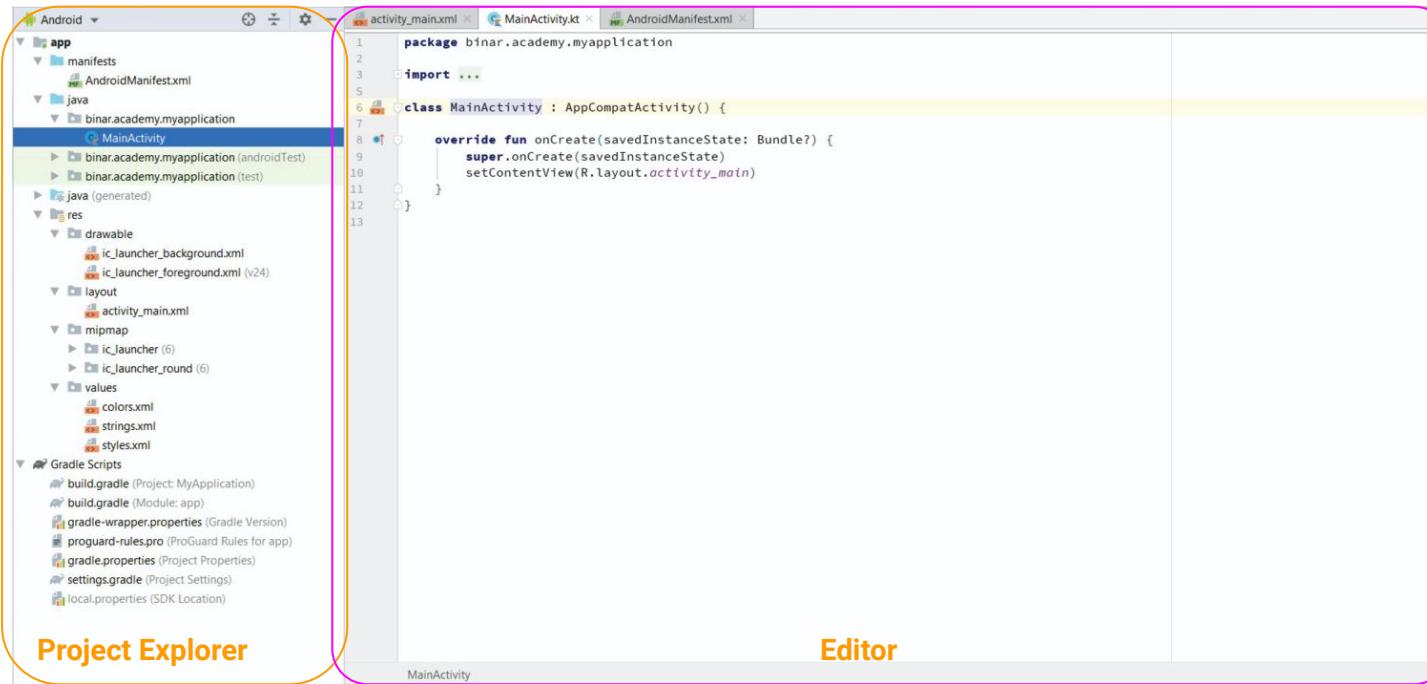
1) Navigation Bar



MyApplication > app > src > main > java > binar > academy > myapplication >  MainActivity

Navigation bar ini menunjukkan jalur *file* yang sedang kita kerjakan. Melalui Navigation Bar kita juga dapat melihat struktur dari *project* kita.

2) Project Explorer dan Editor

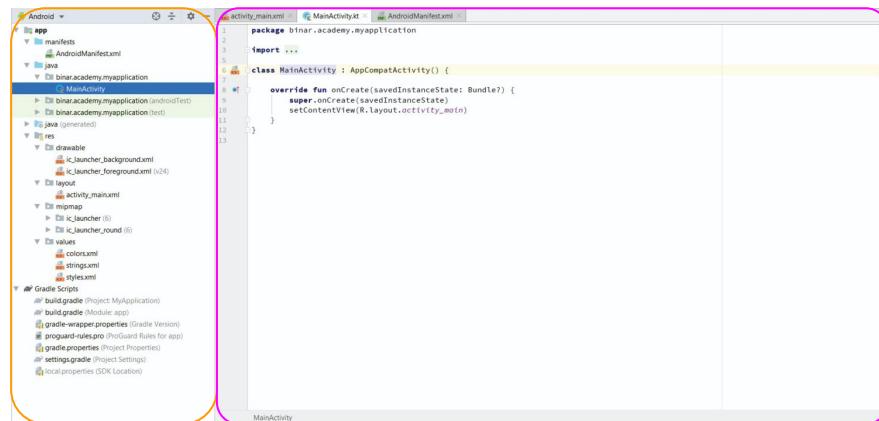


2) Project Explorer dan Editor

Bagian utama dari IDE Android Studio.

Bagian *editor* ini adalah layar tempat kita membuka semua *file*, menuliskan kode, merancang layout, dll. Tergantung pada jenis *file* yang sedang dibuka.

Dan, pada bagian paling kiri adalah struktur *project* kita. Kita akan membahas bagian ini pada bahasan selanjutnya.

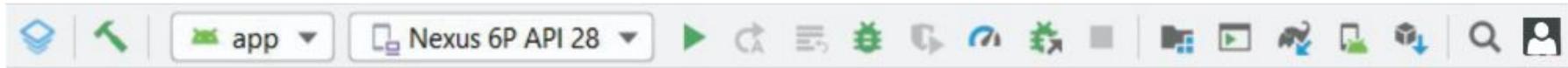


Project Explorer

Editor



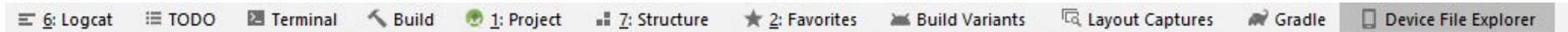
3) Tools Bar



Toolbar mencakup semua *tools action* umum yang diperlukan seperti *copy*, *paste*, *find*, dll. Toolbar juga menyediakan fitur untuk menjalankan aplikasi, menjalankan emulator, buka SDK manager, dan *tools* lainnya.



4) Tool Window Bar



Tool Window Bar merupakan menu *tools* yang mengelilingi IDE Android Studio. Ini adalah jendela yang terdiri dari beberapa opsi yang dapat kita *expand* saat di-klik.

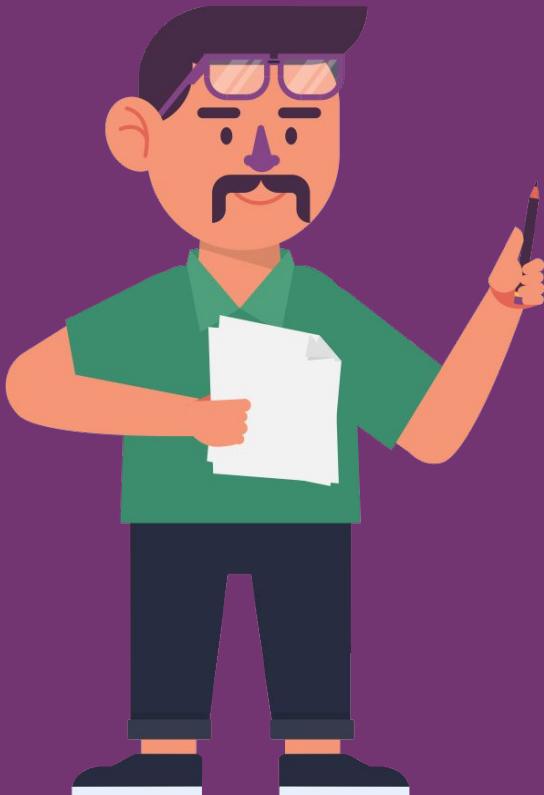


5) Status Bar



Gradle build finished in 1 s 793 ms (19 minutes ago)

Status Bar terletak di bagian terbawah di Android Studio. Ia berfungsi untuk menampilkan status project kita juga menampilkan pesan warning bila ada. Setiap proses yang sedang berjalan, tampil di bagian ini.



Gimana? Udah cukup familiar belum sama layout Android Studio?

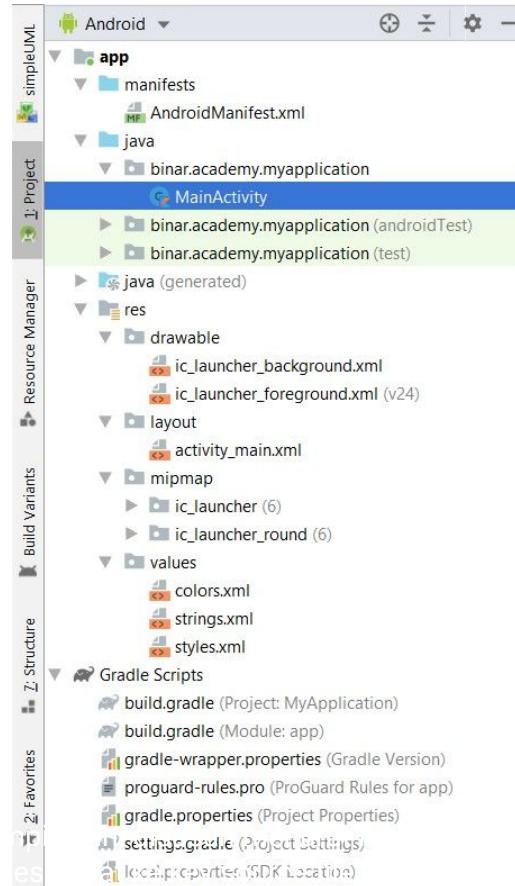
Abis ini kita bakal perdalam satu bagian dari fitur Android Studio, yaitu **Project Structure!**



Project Structure

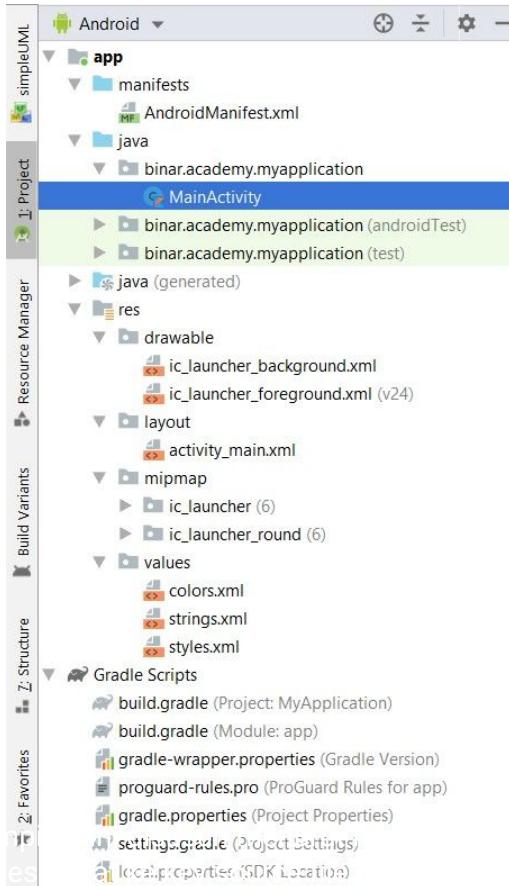
Pada Project Structures, setiap project di Android Studio berisi satu atau beberapa *module* dengan *source code file* dan *resources file*.

Ngomongin module, di Android Studio, ada 3 jenis module yakni **Android App Modules**, **Library Modules**, dan **Google App Engine Modules**





Secara default, Android Studio akan menampilkan *file project* kita dalam tampilan *project* Android, seperti yang ditampilkan seperti gambar disamping.

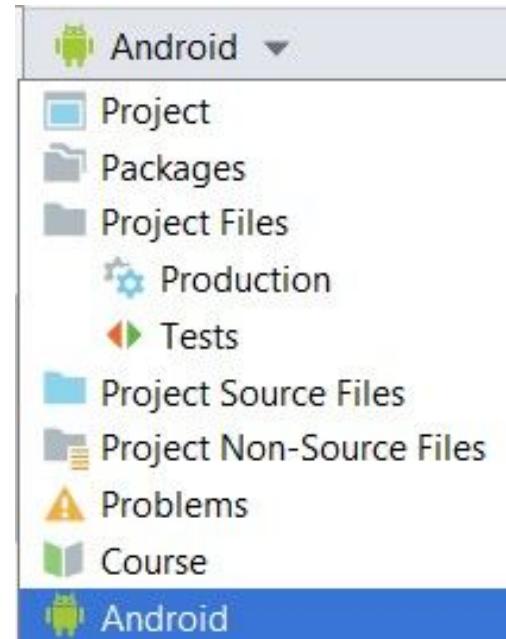




Secara default ketika kita membuat *project* baru, Android Studio akan menampilkan struktur yang lebih ringkas dan cepat sesuai dengan kebutuhan pengembangan Android.

Kita dapat saja mengubahnya dengan tombol yang terdapat di atas *project structure*.

Hal tersebut tentu saja tergantung dari kebutuhan kita. Apakah cukup di module Android saja, atau perlu pindah ke module yang lain.

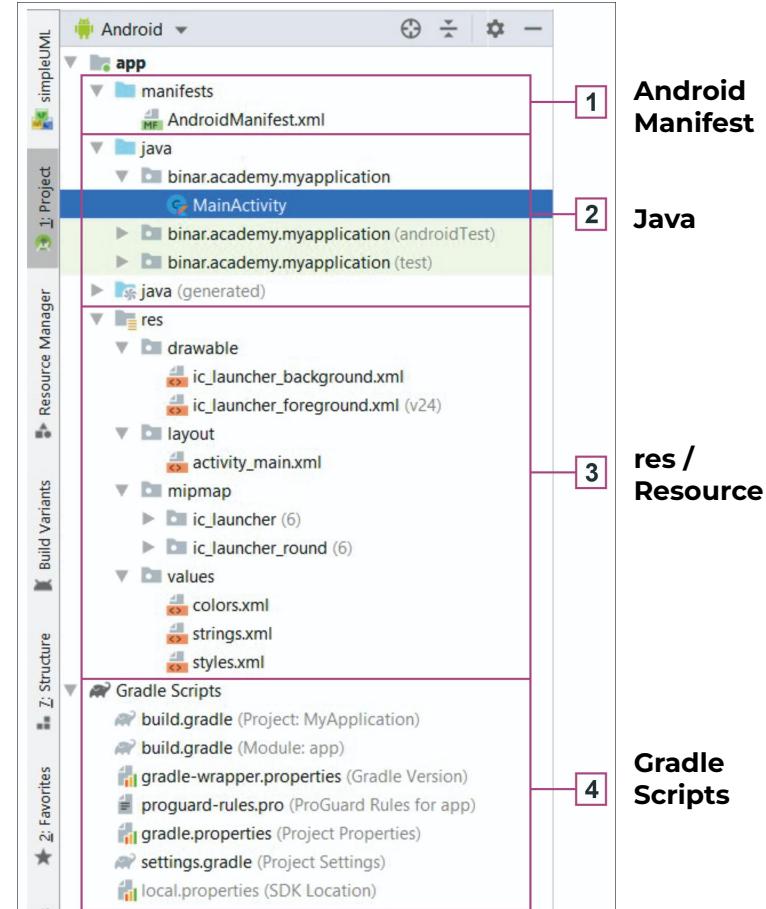




Gambar di samping menunjukkan tampilan Project Android, yang menunjukkan semua file yang terkait dengan *project* kita.

Kalau kamu lihat panah berisikan angka, kita memiliki satu module Aplikasi Android yang terdiri dari **Manifest, Java, Res, dan Gradle Scripts**.

Selanjutnya kita akan coba membongkar apa saja yang ada pada **project structure** kita.





Android Manifest

Setiap project harus memiliki file **AndroidManifest.xml** pada root project source set.

File manifest menjelaskan informasi penting tentang aplikasi kita ke Android *build tools*, Android *operating system*, dan Google Play.

Android Manifest memiliki beberapa fungsi. **Apa saja sih?**
Mari kita cari tahu!





1. Nama Package Aplikasi

Nama package merupakan *unique identity* dari sebuah aplikasi. Identitas ini juga yang akan digunakan di dalam Google Play untuk membedakan satu aplikasi dengan aplikasi lainnya.

Tapi perlu jadi catatan nih, jangan sampai kita mengganti *value* di dalam package tersebut karena bisa dikenali sebagai aplikasi lain kalo udah masuk ke Google Play.



The screenshot shows the file structure of an Android project. At the top, there is a folder icon labeled "manifests". Inside this folder, there is a file icon labeled "AndroidManifest.xml". Below this, there is a dark panel containing the XML code for the manifest file. The code is as follows:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="binar.academy.myapplication">
```



2. Komponen Aplikasi

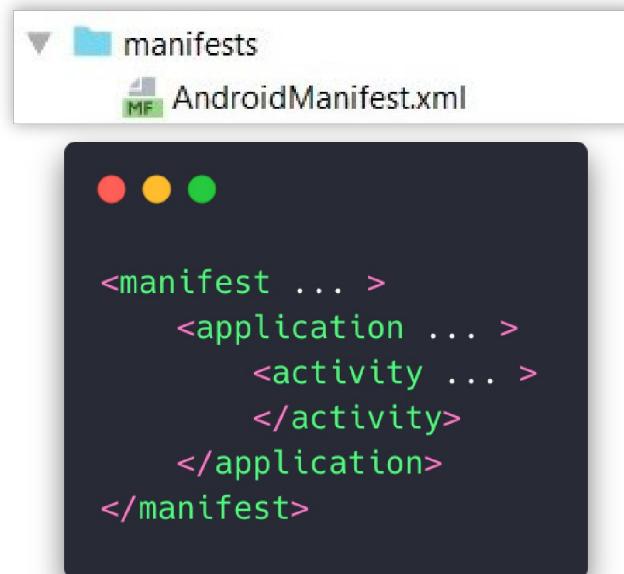
Bertujuan untuk mendeskripsikan komponen dari aplikasi, mulai dari *activity*, *services*, *broadcast receiver*, dan *content provider*. Umumnya komponen aplikasi akan terlihat kayak contoh di samping👉

```
<manifest ... >
    <application ... >
        <activity ... >
            ...
        </activity>
    </application>
</manifest>
```



Tapi kita bisa lho menambahkan beberapa komponen lain di dalam komponen aplikasi. Contoh lain dari komponen pada aplikasi adalah sebagai berikut:

- **<activity>** untuk setiap *subclass* Activity.
- **<service>** untuk setiap *subclass* Service.
- **<receiver>** untuk setiap *subclass* BroadcastReceiver.
- **<provider>** untuk setiap *subclass* ContentProvider.



The screenshot shows the 'manifests' folder containing the 'AndroidManifest.xml' file. The code within the XML file is displayed in a dark-themed code editor, showing the structure of the manifest file:

```
<manifest ... >
    <application ... >
        <activity ... >
            </activity>
        </application>
    </manifest>
```



Semua komponen aplikasi berada di dalam **tag**. Selain itu, komponen aplikasi juga berfungsi sebagai penamaan *class* yang mengimplementasikan komponen lain.

Salah satu contohnya adalah **intent-filter**, yang berfungsi untuk menandai activity mana yang akan dijalankan pertama kali ketika aplikasi dibuka.

Kalau kita masukkan semua komponen aplikasi pada manifest kita. Jadinya bakal kayak gambar disamping.

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category
                android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <service
        android:name=".MyIntentService"
        android:exported="false"/>
    <receiver
        android:name=".MyReceiver"
        android:enabled="true"
        android:exported="true">
    </receiver>
</application>
```



3. Permissions

Pada semua versi Android, elemen **<uses-permission>** perlu dimasukkan pada *manifest* aplikasi untuk mendeklarasikan bahwa aplikasi yang ingin kita gunakan memerlukan **permission**.

Nah, elemen **<uses-permission>** sendiri merupakan turunan dari elemen **<manifest>** di paling atas.

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.snazzyapp">

    <uses-permission android:name="android.permission.INTERNET"/>
    <!-- permissions lainnya -->

    <application ...>
        ...
    </application>
</manifest>
```



Contoh penerapan *permission* adalah untuk akses ke dalam komponen API, seperti *internet*, *external storage*, *contact*, serta untuk berinteraksi dengan aplikasi lainnya.

Supaya lebih kebayang, disamping ini adalah contoh kode untuk permission Internet👉

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.snazzyapp">

    <uses-permission android:name="android.permission.INTERNET"/>
    <!-- permissions lainnya -->

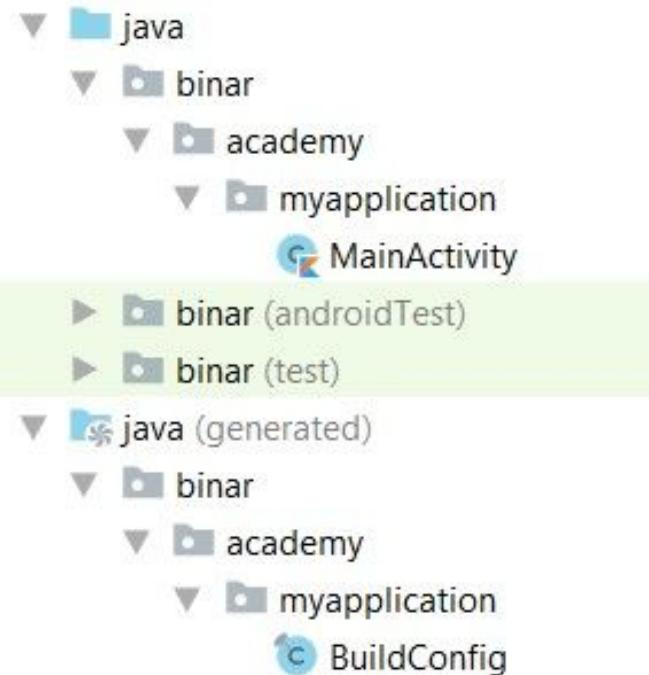
    <application ...>
        ...
    </application>
</manifest>
```



Java

Pada folder **Java**, terdapat berbagai *file source code* yang nantinya kita tulis dalam bahasa Java/Kotlin, termasuk juga **unit test** maupun **android test**.

Untuk folder **Java(generated)**, isinya adalah *file* hasil dari *generate library* atau sebuah *class* dari *project* Android kita.

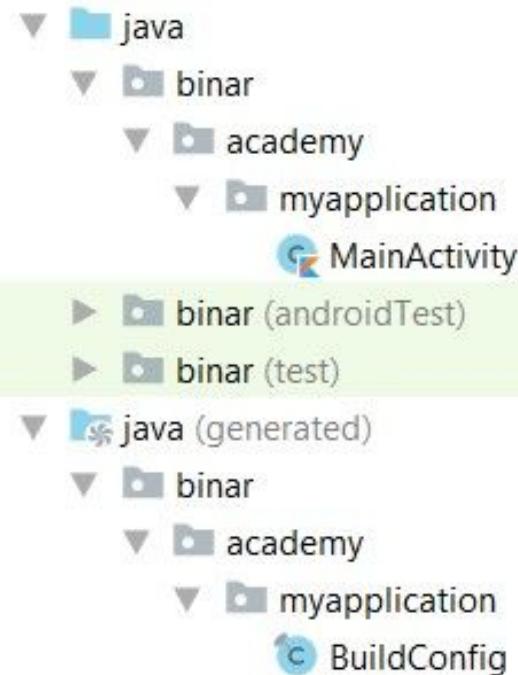




Super direkomendasikan agar kamu nggak mengotak-atik folder ini supaya project tetap bisa berjalan mulus.

Kenapa? Karena code dalam Java (generated) dibuat secara otomatis oleh compiler dan codenya sudah pasti benar.

Jadi tenang aja bestie, kita nggak perlu pusing tujuh keliling memikirkan dan mengubah kode-kode didalamnya 



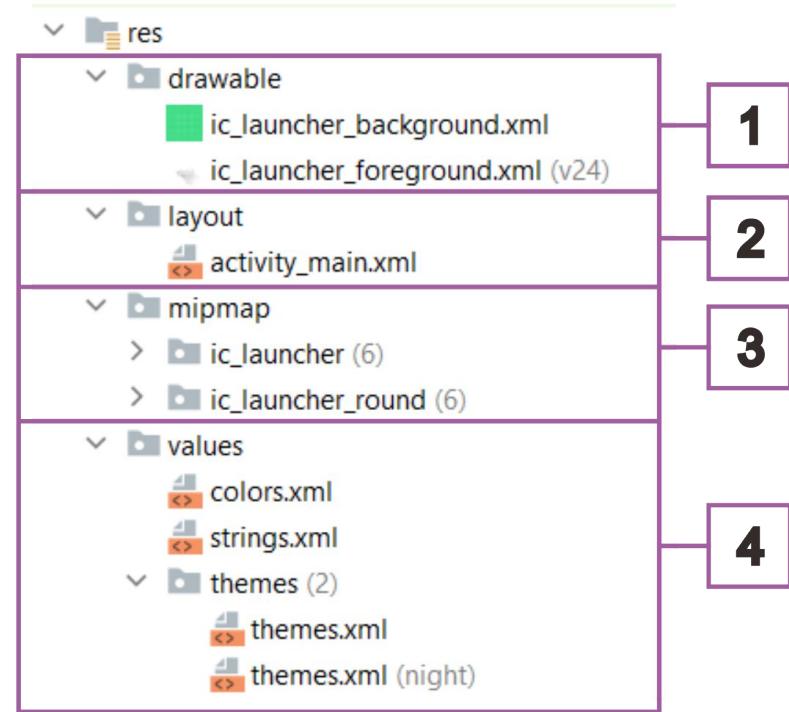


Res/resource

Folder ini terdiri dari semua media aplikasi yang kita butuhkan, seperti *layout*, *gambar*, *font*, *warna*, *string*, *animation*, dan dimensi.

Pada awal *project*, folder **res** akan memiliki susunan seperti gambar di samping.

Apa saja maksudnya? Kita cari tahu lagi, yuk!





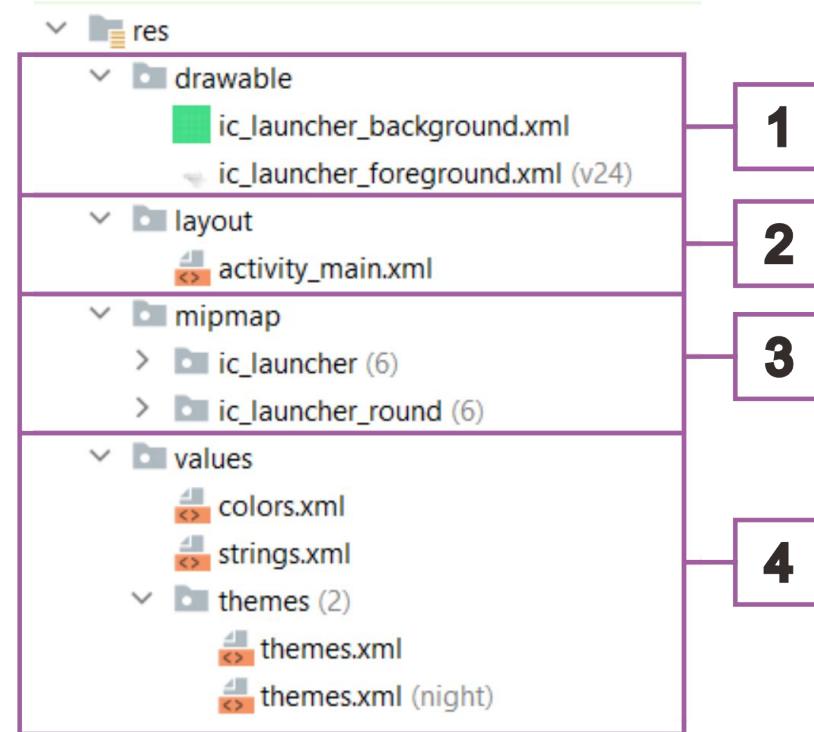
Apa saja maksudnya? Begini nih penjabaran sesuai penomorannya :

1) Folder *drawable*:

berfungsi untuk menyimpan file *gambar, icon, bitmap*, maupun *shape*.

2) Folder *layout*:

berfungsi untuk menyimpan desain layout aplikasi.
(cont)





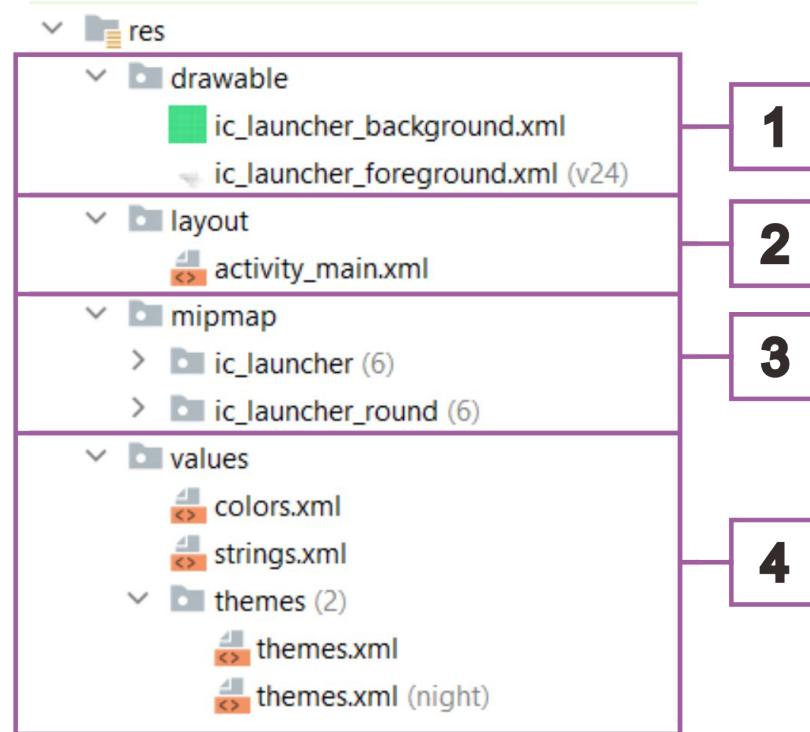
3) Folder **mipmap**:

berfungsi untuk menyimpan file logo dalam berbagai dimensi.

4) Folder **values**:

Berisiakan berbagai macam *data resources* seperti:

- **arrays.xml**: berfungsi untuk menyimpan file array.
- **colors.xml**: berfungsi untuk menyimpan data warna.
- **strings.xml**: berfungsi untuk menyimpan data teks.
- **dimens.xml**: berfungsi untuk menyimpan data ukuran.
- **themes.xml**: berfungsi untuk menyimpan data *style* atau *template*.





Selain itu, folder *res* bisa juga berisikan folder-folder lain, seperti:

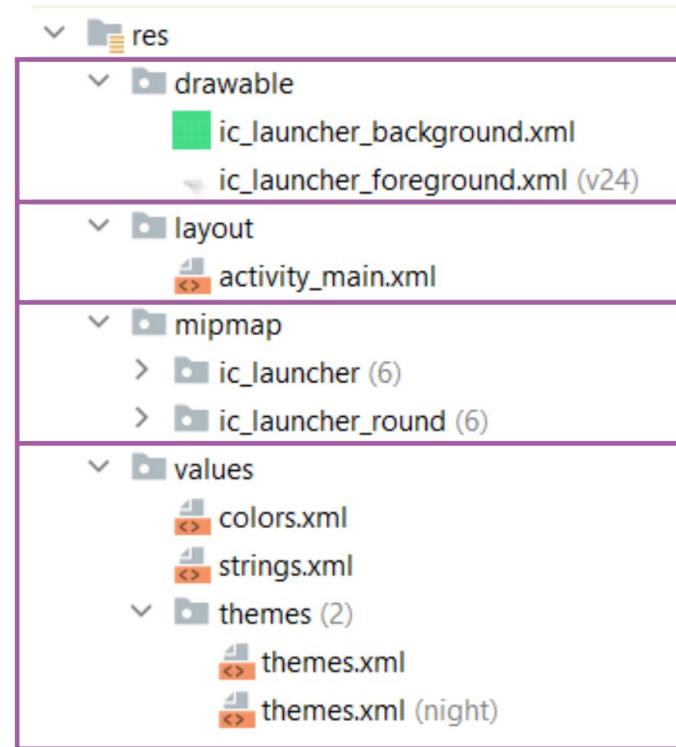
1. **Animator**

Berfungsi untuk menyimpan file XML yang menetapkan **properti animasi**.

2. **Anim**

Berfungsi untuk menyimpan file XML yang menetapkan **animasi tween**.

Properti animasi juga bisa disimpan dalam direktori ini, namun lebih baik tetap ditempatkan pada direktori animator agar kedua tipe ini dapat dibedakan.





3.

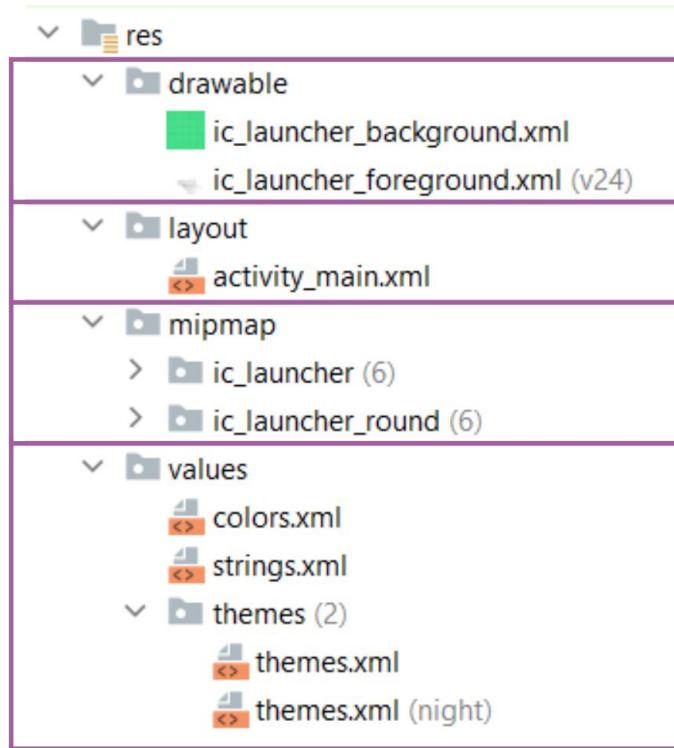
Color

Berfungsi untuk menyimpan file XML yang mendefinisikan **daftar status warna**.

4.

Menu

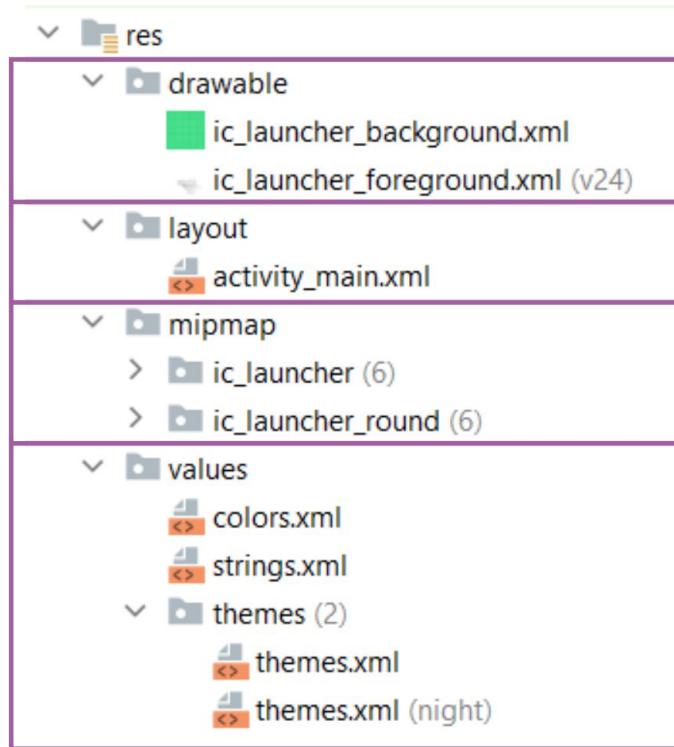
berfungsi untuk menyimpan file XML yang menetapkan **menu aplikasi**, seperti Menu Opsi, Menu Konteks, atau Sub Menu.



**5.****Raw**

Berfungsi untuk menyimpan file mentah, seperti file teks, mp3, mp4, dll.

Jika file mentahnya terdiri dari kesatuan beberapa file dan folder, kamu bisa menggunakan folder assets/ untuk menyimpan file-file itu.



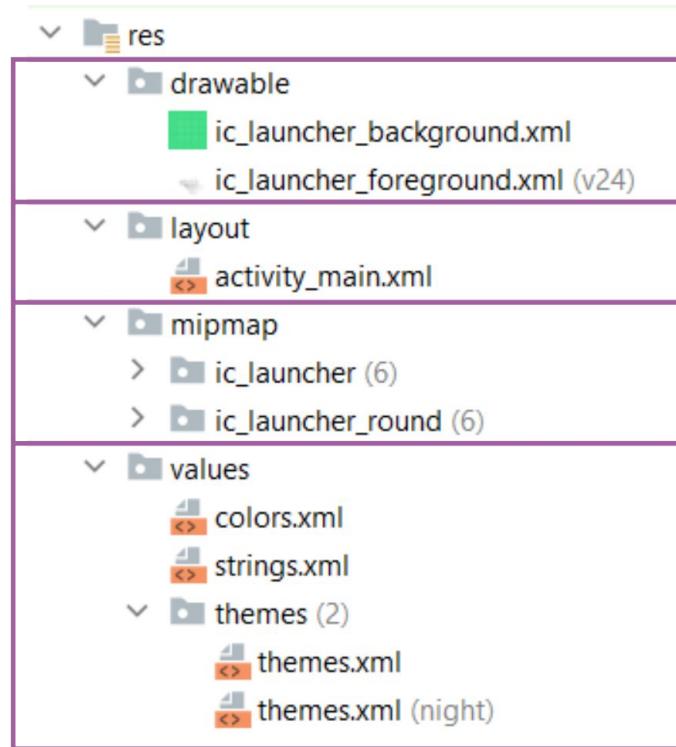


6.

Xml

Berfungsi untuk menyimpan *arbitrary file XML* yang bisa dibaca dengan memanggil **Resources.getXML()**.

Berbagai file konfigurasi XML harus disimpan di sini, seperti konfigurasi yang dapat dicari.



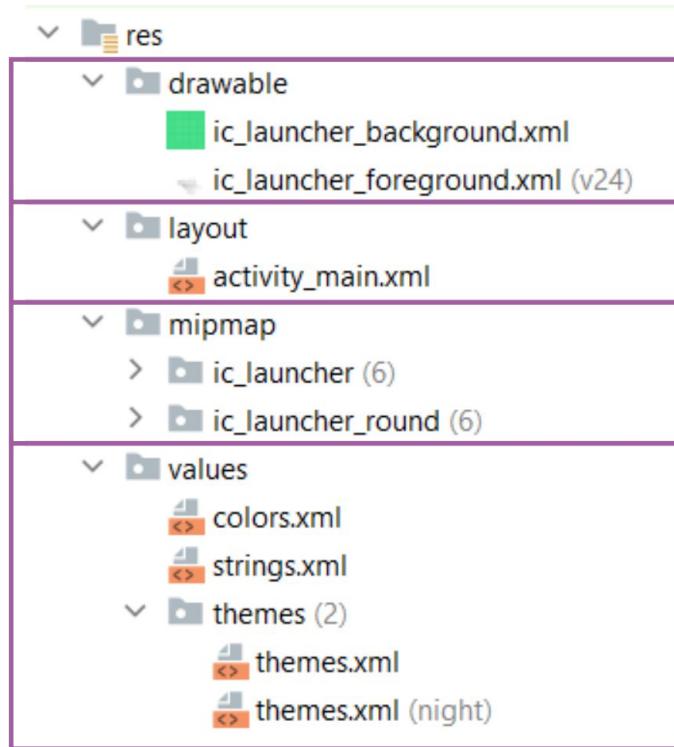


7.

Font

Berfungsi untuk menyimpan file font dengan ekstensi khusus, seperti **.ttf**, **.otf**, **.ttc**, atau file XML yang menyertakan elemen **<font-family>**.

Untuk informasi selengkapnya tentang font sebagai resource. Kita dapat mempelajari lebih lanjutnya di <https://developer.android.com/guide/topics/ui/look-and-feel/fonts-in-xml>.





Gradle Scripts

Android Studio menggunakan Gradle sebagai dasar dari *build system*. Ia memiliki lebih banyak kemampuan khusus yang disediakan oleh *plugin* Android untuk Gradle.

Sistem *build* ini berjalan sebagai fitur terintegrasi dari menu Android Studio, dan terpisah dari *command-line*.



Gradle Scripts

-  **gradle-wrapper.properties** (Gradle Version)
-  **proguard-rules.pro** (ProGuard Rules for app)
-  **gradle.properties** (Project Properties)
-  **settings.gradle** (Project Settings)
-  **local.properties** (SDK Location)



Kita dapat menggunakan fitur-fitur *build system* untuk :

- Menyesuaikan, mengkonfigurasi, dan memperluas proses pembuatan *build*.
- Membuat banyak APK untuk aplikasi kita, dengan berbagai fitur yang menggunakan *project* dan modul yang sama.
- Menggunakan kembali kode dan *resource* di seluruh *set source*.

Kita dapat mencapai semua ini tanpa mengubah *file source* inti aplikasi kita.



Gradle Scripts

-  `build.gradle` (Project: `MyApplication`)
-  `build.gradle` (Module: `app`)
-  `gradle-wrapper.properties` (Gradle Version)
-  `proguard-rules.pro` (ProGuard Rules for app)
-  `gradle.properties` (Project Properties)
-  `settings.gradle` (Project Settings)
-  `local.properties` (SDK Location)



Gimana sih, cara Gradle bekerja?

File *build* Android Studio diberi nama **build.gradle**.

File tersebut adalah file teks biasa yang menggunakan syntax **Groovy** untuk mengkonfigurasi *build* dengan elemen yang disediakan oleh plugin Android untuk Gradle.



Gradle Scripts

-  **build.gradle** (Project: MyApplication)
-  **build.gradle** (Module: app)
-  **gradle-wrapper.properties** (Gradle Version)
-  **proguard-rules.pro** (ProGuard Rules for app)
-  **gradle.properties** (Project Properties)
-  **settings.gradle** (Project Settings)
-  **local.properties** (SDK Location)



Setiap *project* memiliki satu file *build top-level* untuk seluruh project dan file build tingkat module terpisah untuk setiap module.

Jadi, saat kita mengimpor *project* yang ada, Android Studio akan otomatis menghasilkan file *build* yang diperlukan.

▼ Gradle Scripts

-  `build.gradle` (Project: `MyApplication`)
-  `build.gradle` (Module: `app`)
-  `gradle-wrapper.properties` (Gradle Version)
-  `proguard-rules.pro` (ProGuard Rules for app)
-  `gradle.properties` (Project Properties)
-  `settings.gradle` (Project Settings)
-  `local.properties` (SDK Location)



**Wah! Denger-denger makin deket nih
sama Android Studio~**

Biar makin akrab kayak keluarga
sendiri, abis ini kita menuju ke
Kumpulan *Shortcut* yang berlaku di
Android Studio!



Shortcut - Pencarian

Android Studio menyediakan shortcut yang memudahkan developer. Berikut merupakan beberapa shortcut yang biasa digunakan pada Android Studio :

Deskripsi	Windows/Linux	Mac
Menelusuri semuanya (termasuk kode dan menu)	Tekan Shift dua kali	Tekan Shift dua kali
Pencarian teks dalam salah satu file	Ctrl+F	Command+F
Pencarian teks di seluruh project	Ctrl+Shift+F	Command+Shift+F
Pencarian aksi atau perintah-perintah yang ada di Android Studio	Ctrl+Shift+A	Command+Shift+A
Mengganti teks di dalam file	Ctrl+R	Command+R
Mengganti teks di dalam scope tertentu	Ctrl+Shift+R	Command+Shift+R



Shortcut - Navigasi

Deskripsi	Windows/Linux	Mac
Menavigasi ke kelas tertentu	Ctrl+N	Command+O
Menavigasi ke berkas	Ctrl+Shift+N	Command+Shift+O
Lompat ke deklarasi yang dipilih	Ctrl+B	Command+B
Lompat ke method sebelumnya	Alt+↑	Control+↑
Lompat ke method sesudahnya	Alt+↓	Control+↓
Lompat ke baris tertentu	Ctrl+G	Command+L



Shortcut - Navigasi

Deskripsi	Windows/Linux	Mac
Membuka berkas terbaru (recent file)	Ctrl+E	Command+E
Melihat penggunaan pada variabel/objek yang diklik	Ctrl+Alt+F7, atau Ctrl+Kiri Mouse	Command+Option+F7
Melihat penggunaan variabel/objek yang dipilih di seluruh berkas proyek	Alt+F7	Option+F7
Mencari tahu implementasi dari variabel/objek yang dipilih.	Ctrl+Shift+B	Command+Shift+Delete



Shortcut - Editor

Deskripsi	Windows/Linux	Mac
Menggandakan bagian yang dipilih	Ctrl+D	Command+D
Melihat dokumentasi dengan tampilan minimal	Ctrl+Q	Control+J
Melihat isi dari parameter, penting ketika melihat method dari Android atau library lain	Ctrl+P	Command+P
Basic code completion, menampilkan saran untuk melengkapi kode Anda	Ctrl+Spasi	Control+Spasi
Smart code completion, menampilkan saran kode untuk melengkapi kode Anda dengan lebih pintar (menampilkan apa yang benar-benar terkait dengan kode Anda)	Ctrl+Shift+Spasi	Control+Shift+Spasi



Shortcut - Editor

Deskripsi	Windows/Linux	Mac
Generate code, menghasilkan (generate) kode. Perintah ini sangat memudahkan ketika membuat constructor dan setter/getter	Alt+Insert	Command+N
Memformat ulang kode, merapikan kode	Ctrl+Alt+L	Command+Option+L
Delete One Line, Menghapus satu baris kode	Ctrl+Y	Command+Delete
Create method, Membuat teks yang diblok menjadi sebuah fungsi	Ctrl+Alt+M	Command+Option+M
Rename, untuk mengganti nama suatu file atau variabel maupun fungsi	Shift+F6	Shift+F6



Shortcut - Run App

Deskripsi	Windows/Linux	Mac
Debug. Menjalankan aplikasi ke emulator atau devices dalam mode Debug, biasanya untuk keperluan testing	Shift+F9	Control+D
Run. Menjalankan aplikasi ke emulator atau devices	Shift+F10	Control+R
Menerapkan Perubahan Kode	Control+Alt+F10	Control+Shift+Command+R
Control+Shift+Command+R	Ctrl+Shift+F10	Control+Shift+R
Make project, build project	Ctrl+F9	Command+F9

Shortcut di atas merupakan shortcut yang sering digunakan, kita dapat mempelajarinya lebih lanjut [di sini](#).



Ciyee yang udah menjalin ikatan keluarga dengan Android Studio.

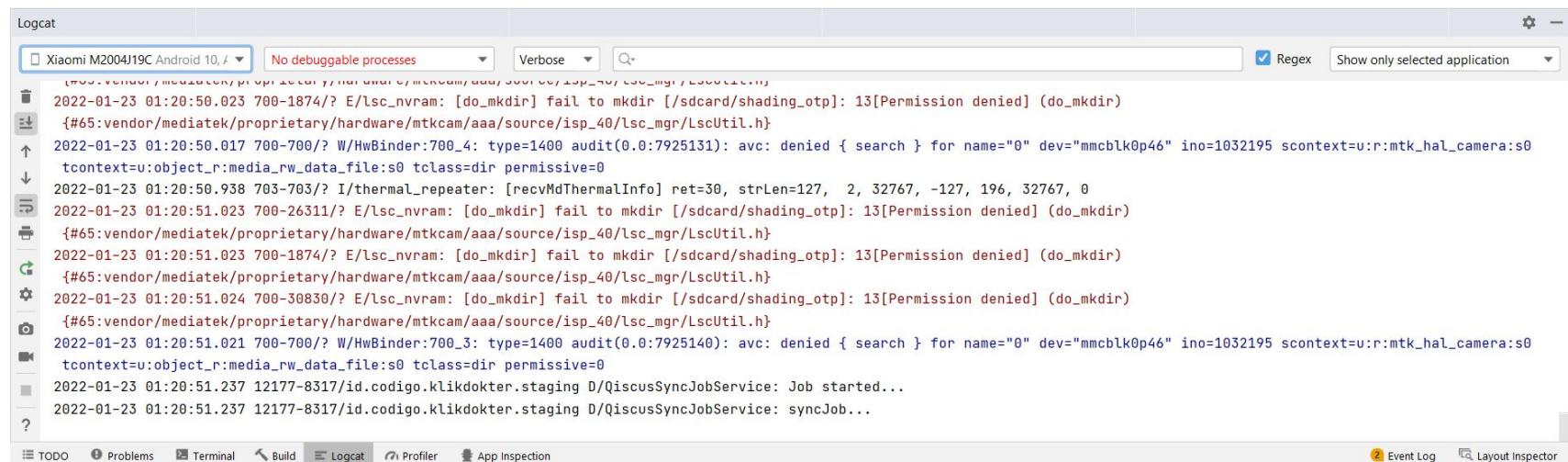
Tapi belum lengkap nih kalau kita kenalan di masa normalnya aja. Gimana kalo kita nemuin error dalam aplikasi kita?

Jawabannya ada di ***Logging & Debugging***.



Bahasan Pertama: Logging

Bagian paling bawah Android Studio yang menampilkan semua pesan bersumber dari aplikasi ini disebut **Window Logcat**. Pesan ini dapat berupa semua pesan error dari Aplikasi yang lagi kamu bikin, maupun error dari Aplikasi dan Sistem Android yang ada di Device.



The screenshot shows the Android Studio Logcat window. At the top, there are dropdown menus for device selection ('Xiaomi M2004J19C Android 10, /'), process selection ('No debuggable processes'), log level ('Verbose'), and search ('Q'). There are also checkboxes for 'Regex' and 'Show only selected application'. The main area displays a list of log messages in red text. Some messages are truncated with ellipses. The log includes entries from various system components like 'lsc_nvram', 'mtkcam/aaa/source/isp_40/lsc_mngr/LscUtil.h', and 'QiscusSyncJobService'. Most messages are related to permission denied errors for mkdir operations. The bottom of the window shows tabs for 'TODO', 'Problems', 'Terminal', 'Build', 'Logcat' (which is selected), 'Profiler', 'App Inspection', 'Event Log' (with a count of 2), and 'Layout Inspector'.

```
2022-01-23 01:20:50.023 700-1874/? E/lsc_nvram: [do_mkdir] fail to mkdir [/sdcard/shading_otp]: 13[Permission denied] (do_mkdir)
{#65:vendor/mediatek/proprietary/hardware/mtkcam/aaa/source/isp_40/lsc_mngr/LscUtil.h}
2022-01-23 01:20:50.017 700-700/? W/HwBinder:700_4: type=1400 audit(0.0:7925131): avc: denied { search } for name="0" dev="mmcblk0p46" ino=1032195 scontext=u:r:mtk_hal_camera:s0
tcontext=u:object_r:media_rw_data_file:s0 tclass=dir permissive=0
2022-01-23 01:20:50.938 703-703/? I/thermal_repeater: [recvMdThermalInfo] ret=30, strLen=127, 2, 32767, -127, 196, 32767, 0
2022-01-23 01:20:51.023 700-26311/? E/lsc_nvram: [do_mkdir] fail to mkdir [/sdcard/shading_otp]: 13[Permission denied] (do_mkdir)
{#65:vendor/mediatek/proprietary/hardware/mtkcam/aaa/source/isp_40/lsc_mngr/LscUtil.h}
2022-01-23 01:20:51.023 700-1874/? E/lsc_nvram: [do_mkdir] fail to mkdir [/sdcard/shading_otp]: 13[Permission denied] (do_mkdir)
{#65:vendor/mediatek/proprietary/hardware/mtkcam/aaa/source/isp_40/lsc_mngr/LscUtil.h}
2022-01-23 01:20:51.024 700-30830/? E/lsc_nvram: [do_mkdir] fail to mkdir [/sdcard/shading_otp]: 13[Permission denied] (do_mkdir)
{#65:vendor/mediatek/proprietary/hardware/mtkcam/aaa/source/isp_40/lsc_mngr/LscUtil.h}
2022-01-23 01:20:51.021 700-700/? W/HwBinder:700_3: type=1400 audit(0.0:7925140): avc: denied { search } for name="0" dev="mmcblk0p46" ino=1032195 scontext=u:r:mtk_hal_camera:s0
tcontext=u:object_r:media_rw_data_file:s0 tclass=dir permissive=0
2022-01-23 01:20:51.237 12177-8317/id.codigo.klikdokter.staging D/QiscusSyncJobService: Job started...
2022-01-23 01:20:51.237 12177-8317/id.codigo.klikdokter.staging D/QiscusSyncJobService: syncJob...
```

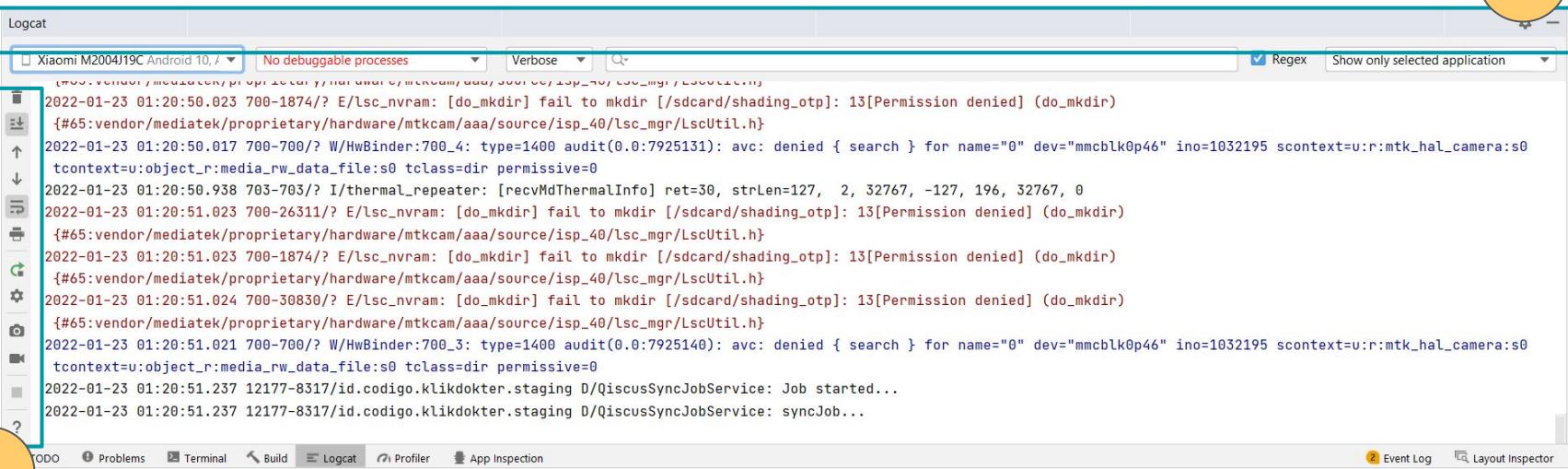


Kalau bagian Android Studio yang menampilkan semua pesan disebut Windows Logcat, pesan yang berisi informasi error disebut dengan **Logging**. Dengan adanya fitur Logging, **kamu bakal tau letak baris code yang bikin aplikasi kamu error.**

Yuk kita dalami fitur-fitur yang ada di Windows Logcat ini~



1



```
Xiaomi M2004J19C Android 10, A No debuggable processes Verbose Regex Show only selected application
2022-01-23 01:20:50.023 700-1874/? E/lsc_nvram: [do_mkdir] fail to mkdir [/sdcard/shading_otp]: 13[Permission denied] (do_mkdir)
{#65:vendor/mediatek/proprietary/hardware/mtkcam/aaa/source/isp_40/lsc_mgr/LscUtil.h}
2022-01-23 01:20:50.017 700-700/? W/HwBinder:700_4: type=1400 audit(0.0:7925131): avc: denied { search } for name="0" dev="mmcblk0p46" ino=1032195 scontext=u:r:mtk_hal_camera:s0
tcontext=u:object_r:media_rw_data_file:s0 tclass=dir permissive=0
2022-01-23 01:20:50.938 703-703/? I/thermal_repeater: [recvMdThermalInfo] ret=30, strLen=127, 2, 32767, -127, 196, 32767, 0
2022-01-23 01:20:51.023 700-26311/? E/lsc_nvram: [do_mkdir] fail to mkdir [/sdcard/shading_otp]: 13[Permission denied] (do_mkdir)
{#65:vendor/mediatek/proprietary/hardware/mtkcam/aaa/source/isp_40/lsc_mgr/LscUtil.h}
2022-01-23 01:20:51.023 700-1874/? E/lsc_nvram: [do_mkdir] fail to mkdir [/sdcard/shading_otp]: 13[Permission denied] (do_mkdir)
{#65:vendor/mediatek/proprietary/hardware/mtkcam/aaa/source/isp_40/lsc_mgr/LscUtil.h}
2022-01-23 01:20:51.024 700-30830/? E/lsc_nvram: [do_mkdir] fail to mkdir [/sdcard/shading_otp]: 13[Permission denied] (do_mkdir)
{#65:vendor/mediatek/proprietary/hardware/mtkcam/aaa/source/isp_40/lsc_mgr/LscUtil.h}
2022-01-23 01:20:51.021 700-700/? W/HwBinder:700_3: type=1400 audit(0.0:7925140): avc: denied { search } for name="0" dev="mmcblk0p46" ino=1032195 scontext=u:r:mtk_hal_camera:s0
tcontext=u:object_r:media_rw_data_file:s0 tclass=dir permissive=0
2022-01-23 01:20:51.237 12177-8317/id.codigo.klikdokter.staging D/QiscusSyncJobService: Job started...
2022-01-23 01:20:51.237 12177-8317/id.codigo.klikdokter.staging D/QiscusSyncJobService: syncJob...
```

2

Event Log

Layout Inspector



Bagiannya apa aja tuh?



Pada bagian atas window Logging, terdapat beberapa bagian komponen, yaitu :

1. Selected Device : Menampilkan semua handphone Android yang terhubung ke komputer. Termasuk Emulator dan Handphone fisik kamu bakal ada disini.

2. Selected Application : Menampilkan aplikasi debug yang berjalan pada Selected Device. Aplikasi yang berjalan pada hp kamu dalam mode debug akan tampil disini.



3. Log Priority : Untuk memfilter prioritas message yang mau ditampilkan. Priority yang tersedia adalah V: Verbose, D: Debug, I: Info, W: Warning, E: Error, A: Assert

4. Search Bar : Digunakan untuk mencari message dengan keyword tertentu.

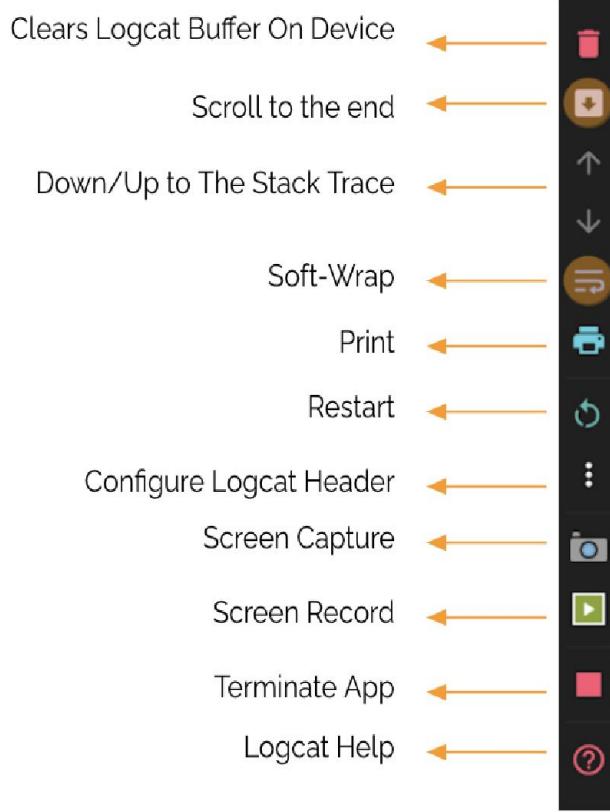
5. Regex Mode : Memungkinkan kita mencari message dengan menggunakan syntax Regex.

6. Selected Scope : Untuk memfilter message yang akan ditampilkan. Apakah hanya untuk aplikasi yang dipilih saja? Atau message dari aplikasi bahkan sistem mau ditampilkan juga?



Nah, kalo di bagian samping kiri, ada bagian yang gak kalah kerennya :

- **Clears Logcat Buffer On Device** : Menghapus semua logcat yang udah tampil. Layar logcatnya jadi bersih deh~
- **Scroll to the end** : Tekan ini kalo kamu udah cari-cari error dengan scroll ke paling atas, lalu pengen balik lagi ke yang paling bawah.





- **Down/Up to Stack Trace** : Tombol ini untuk menavigasi ke baris code yang ada pada message.
- **Print** : Perlu mencetak errornya? Tombol ini sangat membantu!

Clears Logcat Buffer On Device



Scroll to the end



Down/Up to The Stack Trace



Soft-Wrap



Print



Restart



Configure Logcat Header



Screen Capture



Screen Record



Terminate App



Logcat Help





- **Restart** : Tombol ini sangat membantu kalau layar logcat nggak nampilin apa-apa atau ketika kamu mau mulai ulang Logcat nya.
- **Configure Logcat Header** : Untuk konfigurasi logcat dengan konfigurasi lanjutan.
- **Screen Capture** : Psst! Kamu bisa screenshot HP kamu dengan fitur ini lho!

Clears Logcat Buffer On Device



Scroll to the end



Down/Up to The Stack Trace



Soft-Wrap



Print



Restart



Configure Logcat Header



Screen Capture



Screen Record



Terminate App



Logcat Help





- **Screen Record** : Wow. Kamu bisa screen record layar HP kamu juga! Tentunya dengan kualitas yang sangat bagus!
- **Terminate App** : Untuk menutup aplikasi yang dipilih.
- **Logcat Help** : Untuk membuka situs dokumentasi tentang Logcat

Wih keren!, kayak hacker-hacker gitu ya! 😎

Clears Logcat Buffer On Device



Scroll to the end



Down/Up to The Stack Trace



Soft-Wrap



Print



Restart



Configure Logcat Header



Screen Capture



Screen Record



Terminate App



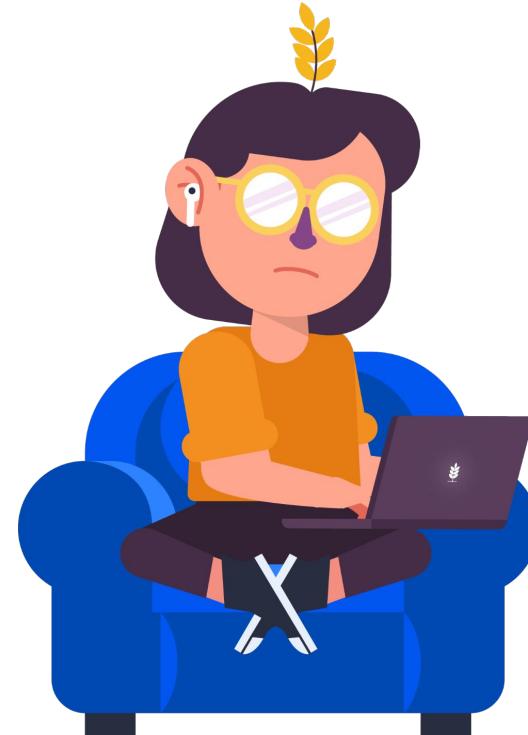
Logcat Help





Kalo kita mau nampolin pesan yang dibuat oleh kita sendiri gimana?

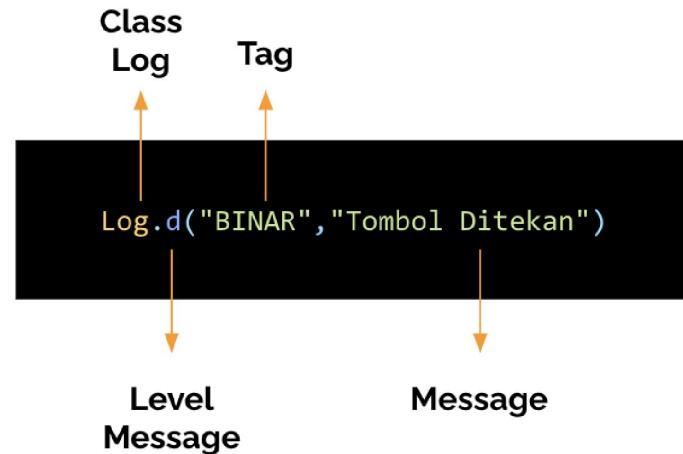
Gampang banget! Tinggal pakai class yang namanya Log!





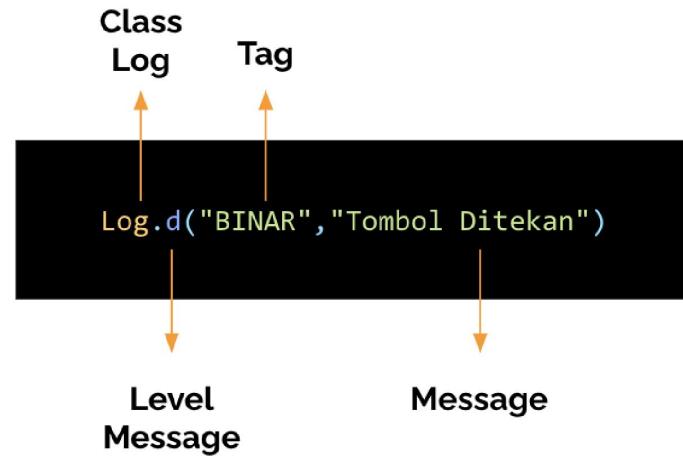
Cara pembuatan kode nya bisa cek gambar di samping yah.
Berikut bagian-bagiannya :

- **Class Log** : Class yang berfungsi untuk menampilkan Logcat
- **d** : Salah satu method sekaligus menyatakan level dari message yang akan ditampilkan. Ayoo inget-inget lagi level message ada apa aja? Nah, tinggal ambil huruf pertamanya aja ya!





- **Tag** : Untuk menandai message yang kamu buat. Tag jangan lebih dari 23 Karakter yaa..
- **Message** : Disinilah kamu menuliskan message yang ingin kamu tampilkan. Dan disini juga biasanya ada error message yang dapat dimengerti oleh kita.





Gimana cara manggil Log-nya?

Okee, gini nih caranya.

Misal kita mau menampilkan pesan logcat pada slide sebelumnya ketika tombol ditekan.

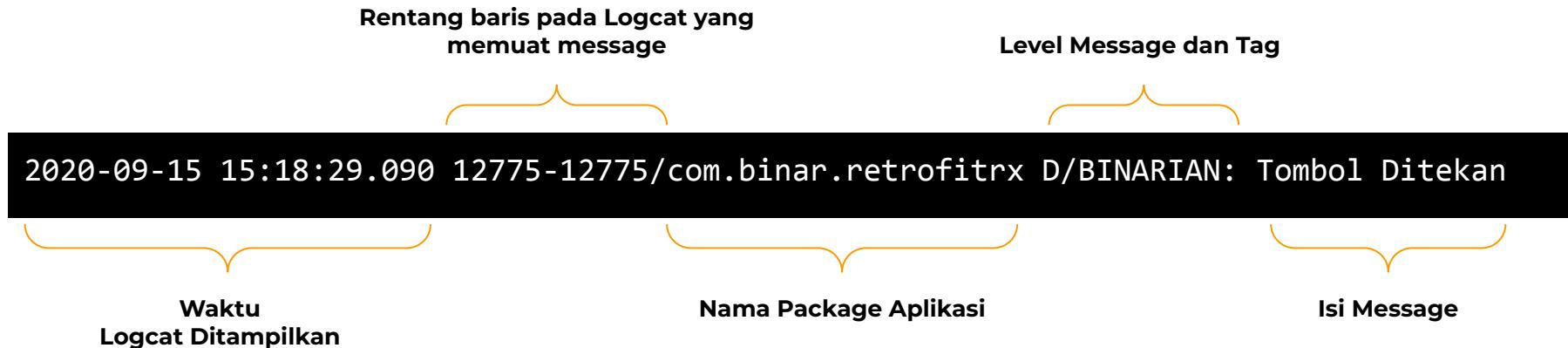
Kita konfigurasi code kita kurang lebih menjadi seperti pada Gambar.

Dengan catatan, di Layout sudah ada **Button** dengan id **btnAdd** yaa~

```
● ● ●  
  
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        btnAdd.setOnClickListener {  
            Log.d("BINARIAN", "Tombol Ditekan")  
        }  
    }  
}
```



Setelah itu, kita jalankan aplikasi kita lalu tekan tombol **logcatnya**. Maka akan muncul message seperti pada Gambar.





Bahasan Kedua: Debugging

Sebelumnya kita udah mempelajari tentang Logging dan Logcat. Dimana dengan mempelajari Logcat, kita bisa tahu **Error Message** dan baris code mana yang menyebabkan error.

Tapi gimana sih kita tahu step-step pada code yang dijalankan sebelumnya, sehingga error tersebut dapat terjadi?

Jawabannya adalah **Debugging** di Android Studio. Nggak cuma tahu step kode mula, kita juga bisa tahu isi dari setiap variable yang ada, lewat debugging.





Sebelum mulai proses Debugging, kita belajar dulu tentang **Breakpoint**.

Breakpoint adalah titik dimana proses debugging akan berhenti sejenak, dan kita sebagai developer bisa mengevaluasi hal apa yang terjadi pada breakpoint tertentu.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val name = "Sabrina"
        Log.d( tag: "BINARIAN", msg: "Logcat ini dari MainActivity")

        btnAdd.setOnClickListener { it: View!
            Log.d( tag: "BINARIAN", msg: "Hello $name")
        }
    }
}
```



Cara menambahkan breakpoint cukup mudah, kita tinggal menekan bagian yang kosong di samping kanan Nomor Baris code sehingga akan muncul bulatan icon berwarna merah.

Sebagai contoh, kita akan mendebug hal yang sederhana kayak gambar disamping.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

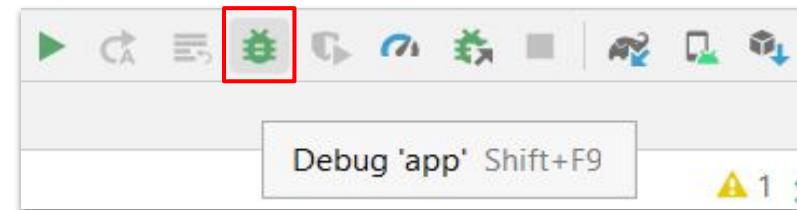
        val name = "Sabrina"
        Log.d( tag: "BINARIAN", msg: "Logcat ini dari MainActivity")

        btnAdd.setOnClickListener { it: View!
            Log.d( tag: "BINARIAN", msg: "Hello $name")
        }
    }
}
```



Kalau kamu udah siap, kita bisa memulai debugging dengan menekan tombol Debug yang memiliki icon Serangga pada Toolbar atau bisa juga dengan mengakses menu Run ⇒ Lalu Debug.

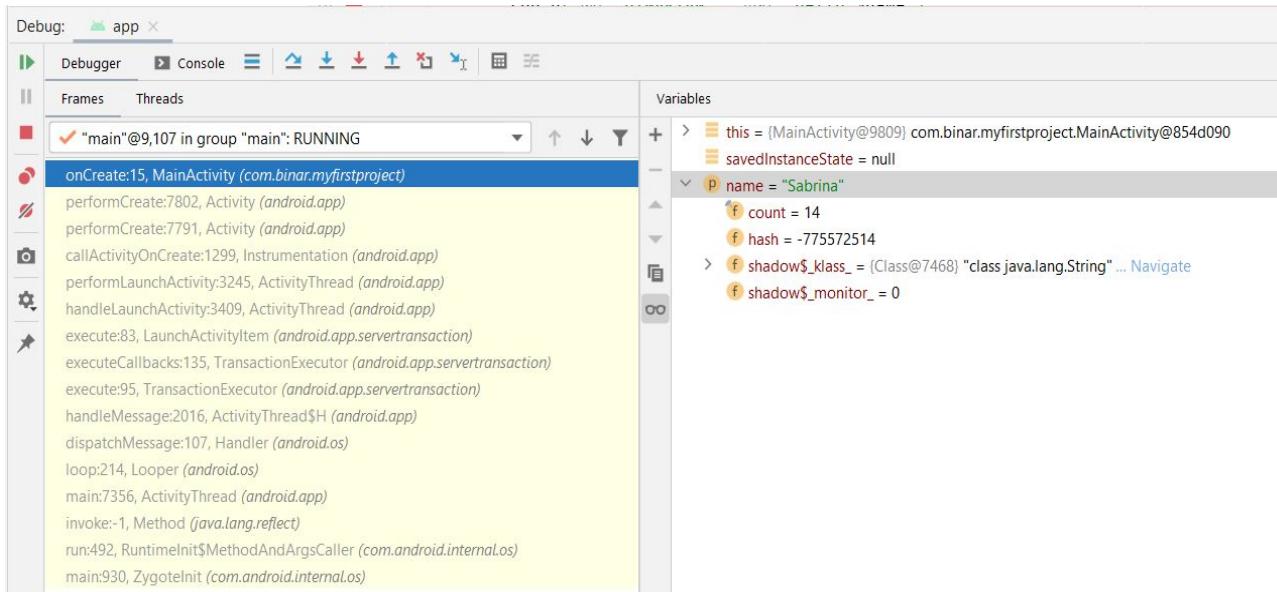
Setelah menekan tombol Debug, Android Studio akan melakukan Build APK lalu menjalankannya ke Device dengan mode Debugging.





Ketika aplikasi sudah mencapai baris code yang telah ditambahkan breakpoint, maka window debug akan otomatis muncul, dan kita sebagai developer dapat mengidentifikasi kondisi pada baris tersebut.

Sebagai contoh, pada baris tersebut kita dapat mengetahui kondisi pada saat itu. Salah satunya ada variable bernama **name** yang memiliki nilai **Sabrina**.





Pada window Debugging, terdapat beberapa tombol untuk mengatur jalannya Debugging, yaitu :

- **Resume Program** : Menjalankan lagi aplikasi jika sudah sampai di salah satu breakpoint
- **Pause Program** : Ketika aplikasi berjalan, bisa diberhentikan sejenak walaupun sedang tidak di breakpoint
- **Stop** : Menghentikan aplikasi. Aplikasi akan keluar.
- **View Breakpoints** : Menampilkan semua breakpoints yang sudah kita pasang (cont.)

Resume Program



Pause Program



Stop



View Breakpoints



Mute Breakpoints



Get Thread Dump



Settings

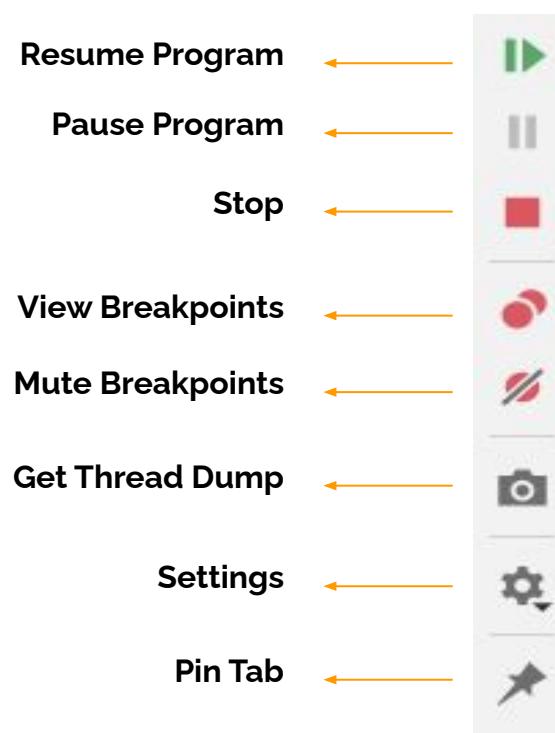


Pin Tab





- **View Breakpoints** : Menampilkan semua breakpoints yang sudah kita pasang
- **Mute Breakpoints** : Menandai breakpoint saat ini agar proses debug tidak terjeda ketika bertemu lagi breakpoint tersebut
- **Get Thread Dump** : Mendapatkan Log Thread secara teknis
- **Settings**:Menampilkan pengaturan debugging lanjutan.
- **Pin Tab** : Mempin tab file debug.



Saatnya kita Quiz!





```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.binar.myfirstproject">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MyFirstProject"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyFirstProject">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

**1. Perhatikan syntax di samping.
Terdapat pada file apakah syntax tersebut?**

- A. build.gradle
- B. AndroidManifest.xml
- C. strings.xml



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.binar.myfirstproject">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MyFirstProject"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyFirstProject">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

**1. Perhatikan syntax di samping.
Terdapat pada file apakah syntax tersebut?**

- A. build.gradle
- B. AndroidManifest.xml
- C. strings.xml

**Yap betul! Contoh syntax di atas hanya terdapat pada file AndroidManifest.xml.
Pada file tersebut, kita dapat mengatur nama package yang kita buat. Mengatur beberapa permissions yang dibutuhkan dalam project. Dan beberapa deklarasi activity, service, receiver, maupun provider.**



2. Ketika kita sudah membuka project, langkah seperti apakah yang kita lakukan untuk membuat sebuah project baru?

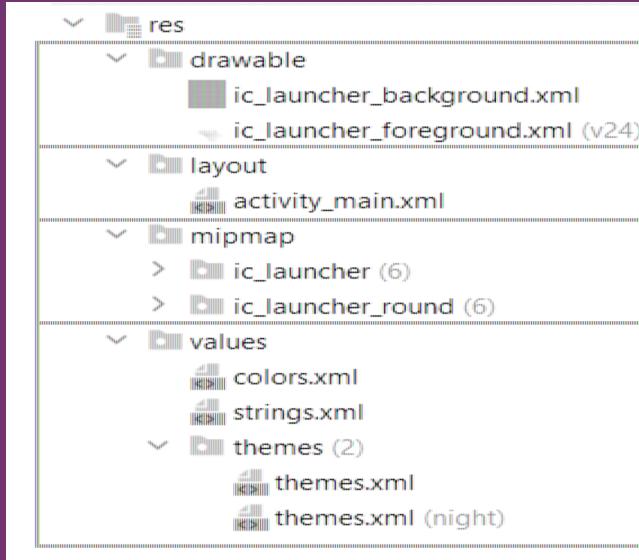
- A. File > New Project
- B. Tools > New > New Project
- C. File > New > New Project



2. Ketika kita sudah membuka project, langkah seperti apakah yang kita lakukan untuk membuat sebuah project baru?

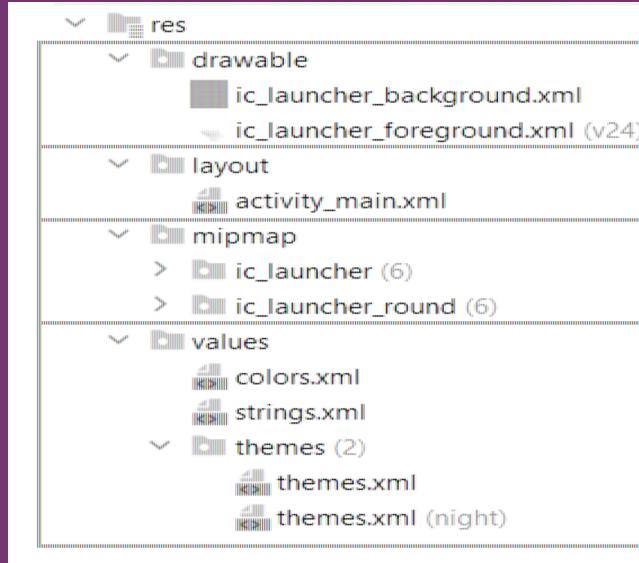
- A. File > New Project
- B. Tools > New > New Project
- C. File > New > New Project

Yap bener banget! Ketika kita sudah pernah membuka sebuah project pada Android Studio. Cara yang tepat untuk membuat sebuah project baru adalah menekan File > New > New Project. Setelahnya kita akan ditampilkan jendela baru untuk template sebuah project.



3. Apa fungsi dari nomor 3?

- A. Menyimpan file gambar, icon, bitmap, maupun bentuk.
- B. Menyimpan file logo dalam berbagai dimensi.
- C. Menyimpan desain layout aplikasi.



3. Apa fungsi dari nomor 3?

- A. Menyimpan file gambar, icon, bitmap, maupun bentuk.
- B. Menyimpan file logo dalam berbagai dimensi.
- C. Menyimpan desain layout aplikasi.

Yap betul!. Pilihan yang tepat adalah Menyimpan file logo dalam berbagai dimensi. Itu adalah fungsi dari folder mipmap. Sedangkan untuk menyimpan file gambar, icon, bitmap, maupun bentuk adalah fungsi folder drawable, dan opsi menyimpan desain layout aplikasi adalah fungsi folder layout.



4. Jika kita ingin mencari suatu potongan code, tapi kita tidak tau code itu ada di file/class mana.. Shortcut apa yang kita gunakan?

- A. Ctrl+F atau Command+F
- B. Tekan Shift Dua Kali
- C. Ctrl+Shift+F atau Command+Shift+F



4. Jika kita ingin mencari suatu potongan code, tapi kita tidak tau code itu ada di file/class mana.. Shortcut apa yang kita gunakan?

- A. Ctrl+F atau Command+F
- B. Tekan Shift Dua Kali
- C. Ctrl+Shift+F atau Command+Shift+F

Yap, benar banget!. Jawabannya Ctrl + Shift + F atau Command + Shift + F. Shortcut ini akan membuka dialog yang bisa melakukan pencarian Code dalam project yang dibuka. Jadi tidak terbatas oleh suatu file.



5. Secara default, dasar dari build system yang digunakan Android Studio dalam membuat aplikasi Android adalah ...

- A. Groovy
- B. Gradle
- C. Maven



5. Secara default, dasar dari build system yang digunakan Android Studio dalam membuat aplikasi Android adalah ...

- A. Groovy
- B. Gradle
- C. Maven

Yuhuu~ Jawabannya Gradle. Secara default, Android Studio menggunakan Gradle sebagai Build System aplikasi Android.



**Biar makin paham, kita adakan
TUGAS yuuk~**

**“Carilah cara membuat dan fungsi
dari folder assets!”**

**Pengumpulan tugas diserahkan
kepada mentor.**

**Kita akan membahas bersama-sama
di pertemuan berikutnya.**

Referensi dan bacaan lebih lanjut~

1. [Meet Android Studio | Android Developers](#)
2. [What is Android Studio? | Tech Target](#)
3. [Keyboard shortcuts | Android Developers](#)
4. [Logcat command-line tool | Android Developers](#)





Nah, selesai sudah pembahasan kita di Chapter 2 Topic 2 ini.

Selanjutnya, kita bakal bahas konsep penamaan dalam Android Studio (**Code Conventions**).

Penasaran kayak gimana? Cus langsung ke topik selanjutnya~



Terima Kasih!



Next Topic

loading...