TEKNOLOGI MULTIMEDIA

Modul 1

PENGENALAN ANDROID STUDIO

Oleh:
Dr. Ing. Melvi, S.T., M. T.
Aryanto, S.T., M.T.

Introduction Modul 1

Apa itu Android

Android adalah sistem operasi yang dikeluarkan oleh Google. Android dibuat khusus untuk *smartphone* dan *tablet*. Berbagai macam produsen telah menggunakan Android sebagai sistem operasi untuk peranti (*device*) yang mereka produksi. Android juga mempunyai *store* dengan lebih dari 2 miliar pengguna aktif per bulannya, per Januari 2018.

Mengapa Android

Kenapa menggunakan Android ?

Android memanjakan penggunanya dengan fitur yang sangat canggih dan tampilan yang bagus. Sistem Android dapat digunakan sebagai alat multimedia seperti pemutar musik dan video. Ia juga memiliki perangkat keras seperti *accelerometer*, *gyroscope* dan sensor lainnya.

Di samping itu ada beberapa hal yang membuat Android menjadi sistem operasi yang memang layak digunakan oleh pengguna atau dikembangkan para developer, seperti yang akan diuraikan berikut ini.

Sistem Operasi Smartphone Terpopuler

Pada tahun 2013, Android menjadi *operation system* (OS) terlaris pada *tablet* dan *smartphone*. Kini *market share* Android sedikitnya 80 % dari total penjualan *smartphone* di tingkat global (statista.com). Tercatat pada tahun 2016 Android *store* memiliki lebih dari 2.8 juta aplikasi.

Android menarik bagi perusahaan teknologi yang membutuhkan barang siap jadi, biaya rendah dan kustomisasi OS untuk perangkat teknologi tinggi mereka. Hal ini menjadi daya tarik bagi banyak perusahaan, sehingga mereka memilih Android.

Source code dari Android bersifat *open source*. Ini adalah hal menarik bagi komunitas developer, karena lisensi *open source* sangat mendukung untuk mengembangkan produknya dengan aman.

Store

Aplikasi Android bisa didistribusikan menggunakan *web*, *copy APK*, dan *store*. Android *store* , yaitu Google Play, merupakan cara termudah bagi para *developer* untuk mendistribusikan aplikasinya ke pasar dengan miliaran pengguna.

Google play merupakan *store* resmi Android yang dikelola oleh Google. Pengguna bisa mencari dan mengunduh aplikasi yang dikembangkan dengan menggunakan Android Software Development Kit.

Google Play tak hanya menawarkan aplikasi. Ada beragam konten lainnya yang dapat dinikmati pengguna, misalnya media digital, musik, buku, majalah, film dan program televisi.

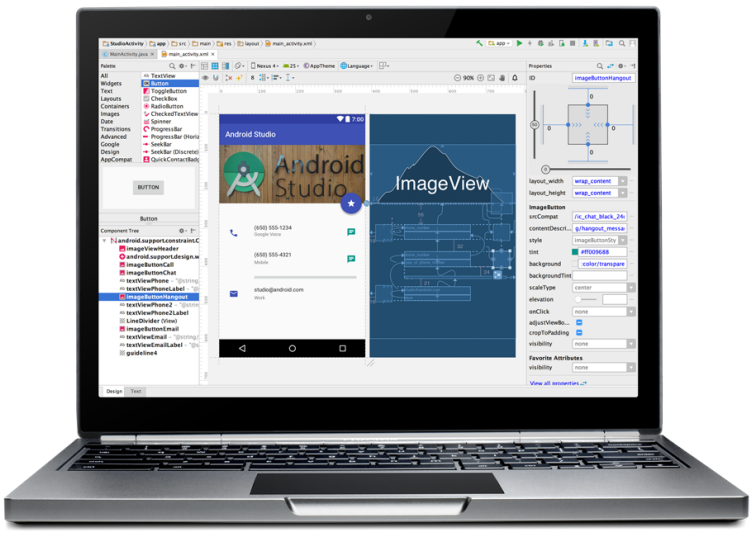
Bagaimana para developer memonetisasi aplikasi yang ada di dalam Google Play?

Strategi monetisasi aplikasi yang ditawarkan Google Play ada bermacam-macam. Dimulai dari *app* berbayar (*paid distribution*), pembelian dalam aplikasi (*in-app purchase*), langganan (*subscriptions*), dan iklan (*ads*). Tentunya *developer* harus mengikuti aturan yang ada untuk memastikan bahwa pengguna mendapatkan pengalaman (*user experience)* terbaik.

Development Kit untuk Developer

Android Software Development Kit (SDK) merupakan *kit* yang bisa digunakan oleh para *developer* untuk mengembangkan aplikasi berbasis Android. Di dalamnya, terdapat beberapa *tools* seperti *debugger*, *software libraries*, *emulator*, dokumentasi, *sample code* dan tutorial.

Bahasa pemrograman yang sering digunakan untuk mengembangkan aplikasi Android adalah Java. Namun ada beberapa bahasa lainnya yang dapat digunakan, seperti C++ dan Go. Pada IO 2017 Google juga menetapkan Kotlin sebagai tambahan bahasa resmi.



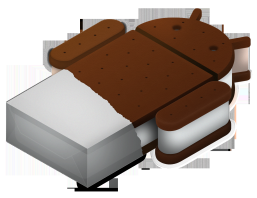
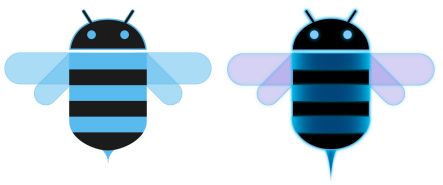
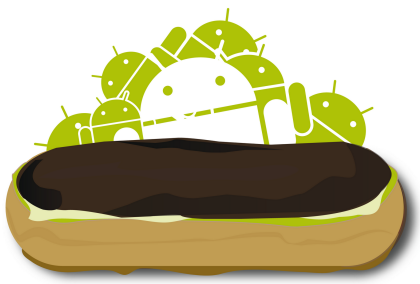
Berbicara tentang pemrograman tentunya tak lepas dari *Integrated Development Environment* (IDE). Pada 2014 Google mengeluarkan IDE yang bernama Android Studio yang berbasiskan Intellij IDEA.

Dengan menggunakan Android Studio, para *developer* dapat membuat aplikasi dari nol hingga dipublikasikan ke dalam *store*. Android Studio juga mempunyai beberapa fitur *built-in* yang sangat membantu para *developer* untuk memaksimalkan proses pembuatan aplikasi. Fitur-fitur ini misalnya Gradle, Code Completion, dan beragam integrasi dengan layanan dari Google, seperti Firebase.

Sejarah Perkembangan Android

Berikut adalah rangkaian sejarah perkembangan Android yang resmi diluncurkan oleh Google dari waktu ke waktu.

Version	Code name	Release date	API level	DVM/ART	New features	Icon
11	11	September 8, 2020	30	ART	• Chat Bubbles • Screen Recorder • Device Control • Predictive Tool • One-time permission	
10	10	September 3, 2019	29	ART	• Live Caption • Smart Reply • Sound Amplifier • Dark Theme • Privacy & Security • Digital Wellbeing	
9	Pie	August 6, 2018	28	ART	• Adaptive Battery • Adaptive Brightness	
8.0-8.1	Oreo	October 25, 2017	26 - 27	ART	• Picture-in-Picture	
7.1 - 7.1.2	Nougat	August 22, 2016	24 - 25	ART	• Multi window • GIF Keyboard	
6.0 - 6.0.1	Marshmallow	October 5, 2015	23	ART	• Now On Tap • Permissions • Battery (Doze & App Standy)	
5.1 - 5.1.1	Lollipop	November 12, 2014	21 - 22	ART	• Material Design • Multiscreen • Notifications	

Version	Code name	Release date	API level	DVM/ART	New features	Icon
4.4 - 4.4.4	KitKat	October 31, 2013	19 - 20	DVM (and ART 1.6.0)	• Voice : Ok Google • Immersive Design • Smart Dialer	
4.1 - 4.3.1	Jelly Bean	July 9, 2012	18	DVM	• Google Now • Actionable Notifications • Account Switching	
4.0 - 4.6	Ice Cream Sandwich	October 19, 2011	15	DVM	• Custom Home Screen • Data Usage Control • Android Beam	
3.0 - 3.2.6	HoneyComb	February 22, 2011	11 - 13	DVM	• Tablet-Friendly Design • System Bar • Quick Settings	
2.3 - 2.3.7	Gingerbread	February 9, 2011	9 - 10	DVM	• Gaming APIs • NFC • Battery Management	
2.2 - 2.23	Froyo	May 20, 2010	8	DVM	• Voice Action • Portable Hotspot • Dalvik JIT	
2.0 - 2.1	Eclair	October 26, 2009	5	-	• Google Maps Navigation • Home Screen Customization • Speech-to-Text	
1.6	Donut	September 15, 2009	4	-	• Quick Search Box • Screen Size Diversity • Android Market	
1.5	Cupcake	April 27, 2009	3	-	-	

Sources : https://en.wikipedia.org/wiki/Android_(operating_system)

Dari tabel sejarah perkembangan di atas dapat kita lihat ada kolom DVM / ART. Kolom ini menunjukkan eksekusi kompilasi ketika menjalankan aplikasi Android. Pada API KitKat dan sebelumnya Android menggunakan DVM (Dalvik Virtual Machine). DVM menerapkan pendekatan JIT (*Just-In-Time*), di mana kompilasi dijalankan ketika ada permintaan untuk menjalankan aplikasi.

Sedangkan ART (*Android Runtime*) menerapkan pendekatan berbeda yaitu AOT (*Ahead-Of-Time*). AOT melakukan kompilasi pada saat proses instalasi aplikasi.

Dari versi Lollipop hingga sekarang, Android sepenuhnya mengadopsi ART. Mengapa demikian?

DVM menggunakan JIT yang berarti kompilasi dilakukan setiap kali aplikasi dijalankan. Hal ini sangat mempengaruhi kecepatan respon aplikasi. Setiap kali kita menyentuh ikon aplikasi maka kompilasi akan dilakukan. Tentu proses ini menghabiskan CPU dan berimbas pada relatif lebih borosnya penggunaan baterai.

Beda dengan DVM, ART melakukan proses kompilasi pada saat proses instalasi. Jadi setiap kali aplikasi dijalankan, sudah tidak ada lagi proses kompilasi. Hal ini meningkatkan performa dalam menjalankan aplikasi. Selain itu karena penggunaan sumber daya CPU bisa dikurangi, pemakaian baterai jadi lebih hemat. Akan tetapi ART membutuhkan *space* (ukuran berkas) yang lebih besar jika dibandingkan dengan DVM.

Jika ingin mendalami proses *run-time* yang ada di Android, silakan klik tautan berikut ini:

- https://source.android.com/devices/tech/dalvik/

Beberapa bacaan dasar yang dapat menambah wawasan Anda, antara lain:

- https://www.android.com/history/
- https://en.wikipedia.org/wiki/Android_version_history

Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment* (IDE) untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA . Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih demi meningkatkan produktifitas Anda saat membuat aplikasi Android, misalnya:

- Sistem versi berbasis gradle yang fleksibel.
- Emulator yang cepat dan kaya fitur.
- Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android.
- Instant Run untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru.
- Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh.
- Alat pengujian dan kerangka kerja yang ekstensif.
- Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain.
- Dukungan C++ dan NDK.
- Dukungan bawaan untuk Google Cloud Platform, mempermudah pengintegrasian Google Cloud Messaging dan App Engine.

Persyaratan Sistem

Windows	Mac

Windows	Mac
• Microsoft® Windows® 7/8/10 (64-bit) • RAM minimum 4 GB, RAM yang disarankan 8 GB; tambah 1 GB untuk Android Emulator • Ruang disk minimum yang tersedia 2 GB, Disarankan 4 GB (500 MB untuk IDE + 1,5 GB untuk Android SDK dan gambar sistem emulator) • Resolusi layar minimum 1280 x 800	• Mac® OS X® 10.10 (Yosemite) atau lebih baru, hingga 10.14 (macOS Mojave) • RAM minimum 4 GB, RAM yang disarankan 8 GB; tambah 1 GB untuk Android Emulator • Ruang disk minimum yang tersedia 2 GB, Disarankan 4 GB (500 MB untuk IDE + 1,5 GB untuk Android SDK dan gambar sistem emulator) • Resolusi layar minimum 1280 x 800
Linux	**Chrome OS**
• Desktop GNOME atau KDE *Telah diuji pada gLinux yang berbasis Debian* • Distribusi 64-bit yang mampu menjalankan aplikasi 32-bit • GNU C Library (glibc) 2.19 atau lebih baru • RAM minimum 4 GB, RAM yang disarankan 8 GB; tambah 1 GB untuk Android Emulator • Ruang disk minimum yang tersedia 2 GB, Disarankan 4 GB (500 MB untuk IDE + 1,5 GB untuk Android SDK dan gambar sistem emulator) • Resolusi layar minimum 1280 x 800	• 8 GB RAM atau lebih yang disarankan • Ruang disk minimum yang tersedia 4 GB • Resolusi layar minimum 1280 x 800 • Disarankan Intel i5 atau lebih (U series atau lebih) Perangkat yang direkomendasikan: • **Acer:** Chromebook 13/Spin 13, Chromebox CXI3 • **Lenovo:** Yoga C630 Chromebook • **HP:** Chromebook x360 14, Chromebox G2 • **Dell:** Inspiron Chromebook 14 • **ASUS:** Chromebox 3 • **ViewSonic:** NMP660 Chromebox • **CTL:** Chromebox CBx1

How To Install

Java

Salah satu Bahasa yang bisa digunakan untuk development Android adalah Java. Selain Java ada beberapa Bahasa lain yang bisa digunakan seperti C/C++, Go, dan Kotlin (per Mei 2017).

Pada akademi ini kita hanya akan fokus menggunakan Kotlin dan Java sebagai bahasa pemrograman. Oleh karena itu mari instal dulu software yang harus kita gunakan untuk coding (menuliskan baris code). Siapkan senjata Anda sebelum berperang.

Instal Java Development Kit yang bisa kita dapatkan pada tautan berikut:

- https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html

Biasanya muncul pertanyaan, "Apakah JRE cukup?" Tidak, JRE adalah Java Runtime Environment yang berfungsi sebagai Virtual Machine untuk menjalankan program Java. Sedangkan JDK merupakan Java SE Development Kit, di mana JRE juga terdapat di dalamnya. Dan yang lebih penting adalah di dalamnya terdapat *compiler* dan *tools* untuk membuat dan *compile* program.

Sederhananya JRE untuk menjalankan program, sedangkan JDK untuk membuat program. Kabar baiknya, mulai dari Android Studio 2.2 ke atas, sudah tersedia OpenJDK yang bisa menjalankan program dasar dari Java.

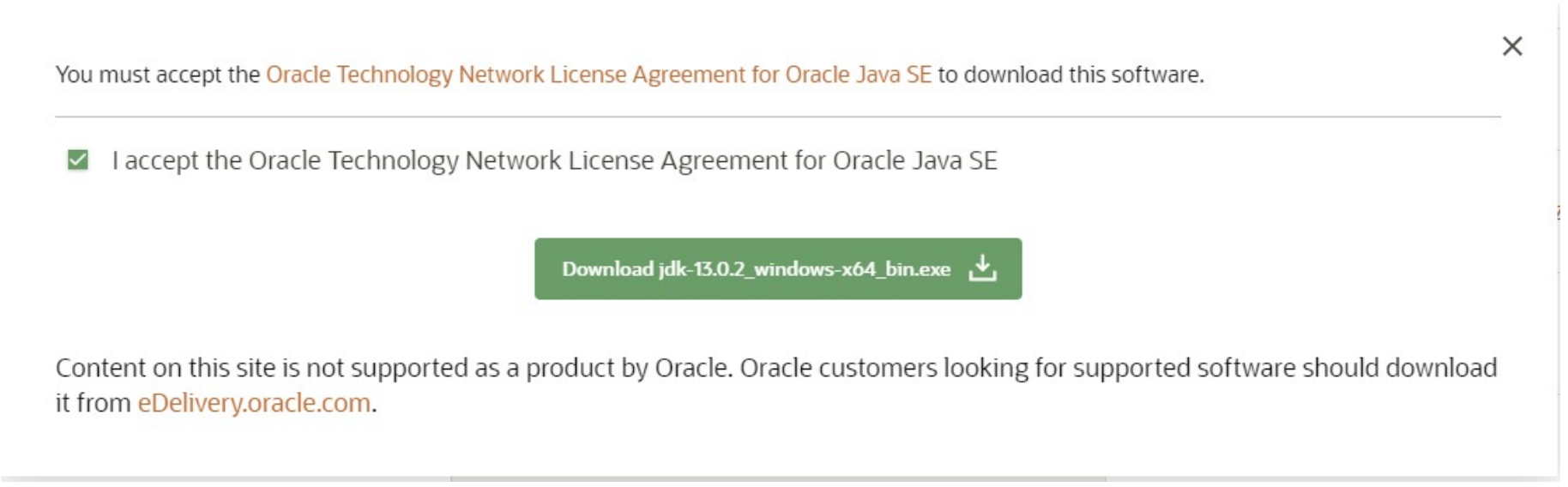
> **Catatan:**
> Saat ini ketika menginstall Android Studio sudah terdapat JDK yang bawaan yang bisa digunakan. Sehingga langkah di bawah ini bersifat opsional jika Anda mengalami kendala JDK..

Mari kita mulai dengan proses instalasi dari JDK dari Oracle.

1. Langsung saja buka tautan di atas menggunakan browser Anda. Pilihlah *link download* yang sesuai dengan OS yang Anda pakai.

Linux x86 Compressed Archive	186.6 MB	jdk-8u…ux-i586.tar.gz
Linux x64 RPM Package	171.16 MB	jdk-8u251-linux-x64.rpm
Linux x64 Compressed Archive	186.09 MB	jdk-8u251-linux-x64.tar.gz
macOS x64	254.78 MB	jdk-8u251-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	125.19 MB	jdk-8u251-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	88.16 MB	jdk-8u251-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	133.64 MB	jdk-8u251-solaris-x64.tar.Z
Solaris x64	91.9 MB	jdk-8u251-solaris-x64.tar.gz
Windows x86	201.17 MB	jdk-8u251-windows-i586.exe
Windows x64	211.54 MB	jdk-8u251-windows-x64.exe

2. Jangan lupa untuk mencentang *Accept License Agreement* dan klik tombol *Download*.



3. Anda akan diminta untuk login terlebih dahulu, silakan mendaftar dulu jika Anda belum memiliki akun.
4. Setelah proses mengunduh selesai, langsung *install* ke gawai Anda dan ikuti petunjuknya sampai selesai.

Android Studio

Pada akademi kali ini kita akan menggunakan Android Studio sebagai IDE (*Integrated Development Environment*). Android Studio dirilis 16 Mei 2013 saat Google IO berlangsung. Android Studio berbasiskan JetBrains Intellij IDEA, dan dikhususkan untuk pengembangan *software* berbasis Android.

Semua versi dari Android Studio dapat Anda unduh pada tautan ini:

- https://developer.android.com/studio/archive.html

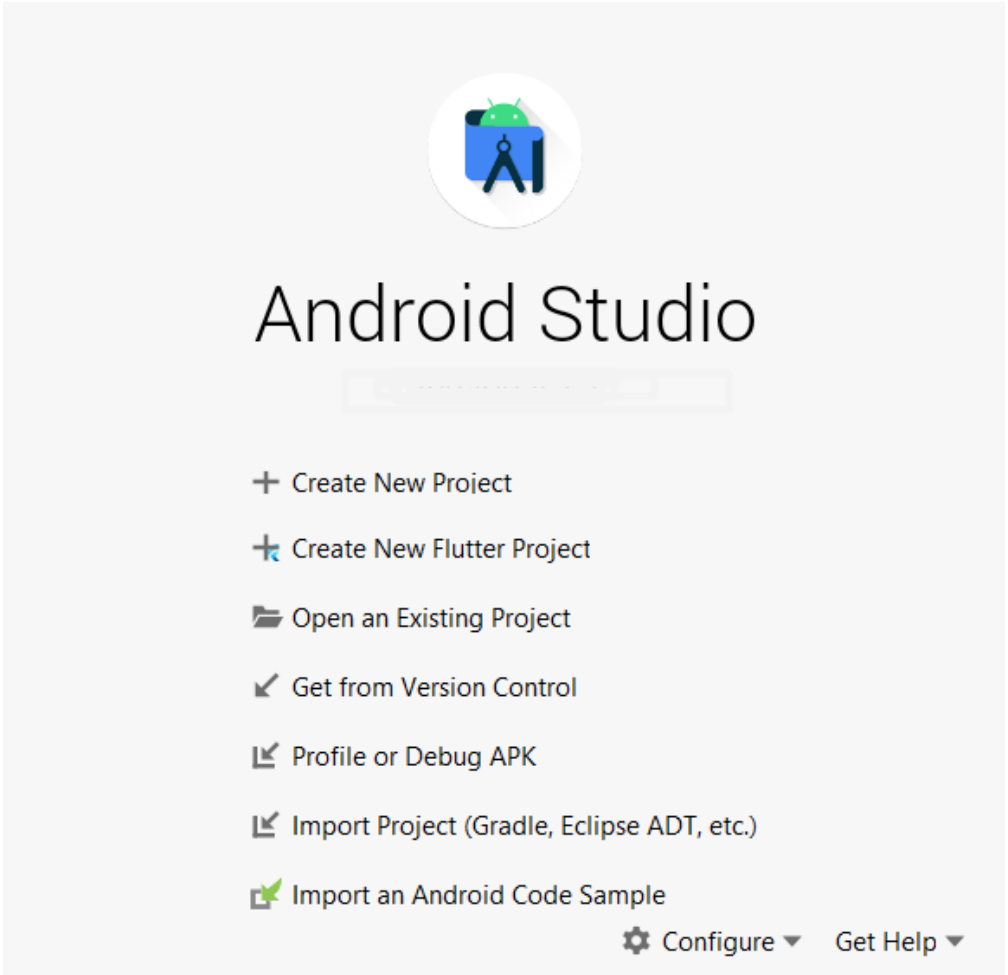
Atau versi terbarunya:

- https://developer.android.com/studio/index.html#downloads

Mari langsung saja kita mulai instalasi Android Studio.

1. Tampilan dari laman unduh Android Studio.

2. Kemudian instal Android Studio ke gawai Anda dan ikuti petunjuknya sampai selesai. Saat instalasi berlangsung Anda diminta untuk menginstal SDK(Software Development Kit) yang membutuhkan koneksi internet. Maka pastikan saat menginstal Anda terhubung dengan internet.



3. Setelah selesai melakukan instalasi Android Studio, akan muncul seperti gambar di bawah ini.

   Gambar di atas berarti aplikasi Android Studio sudah bisa digunakan.

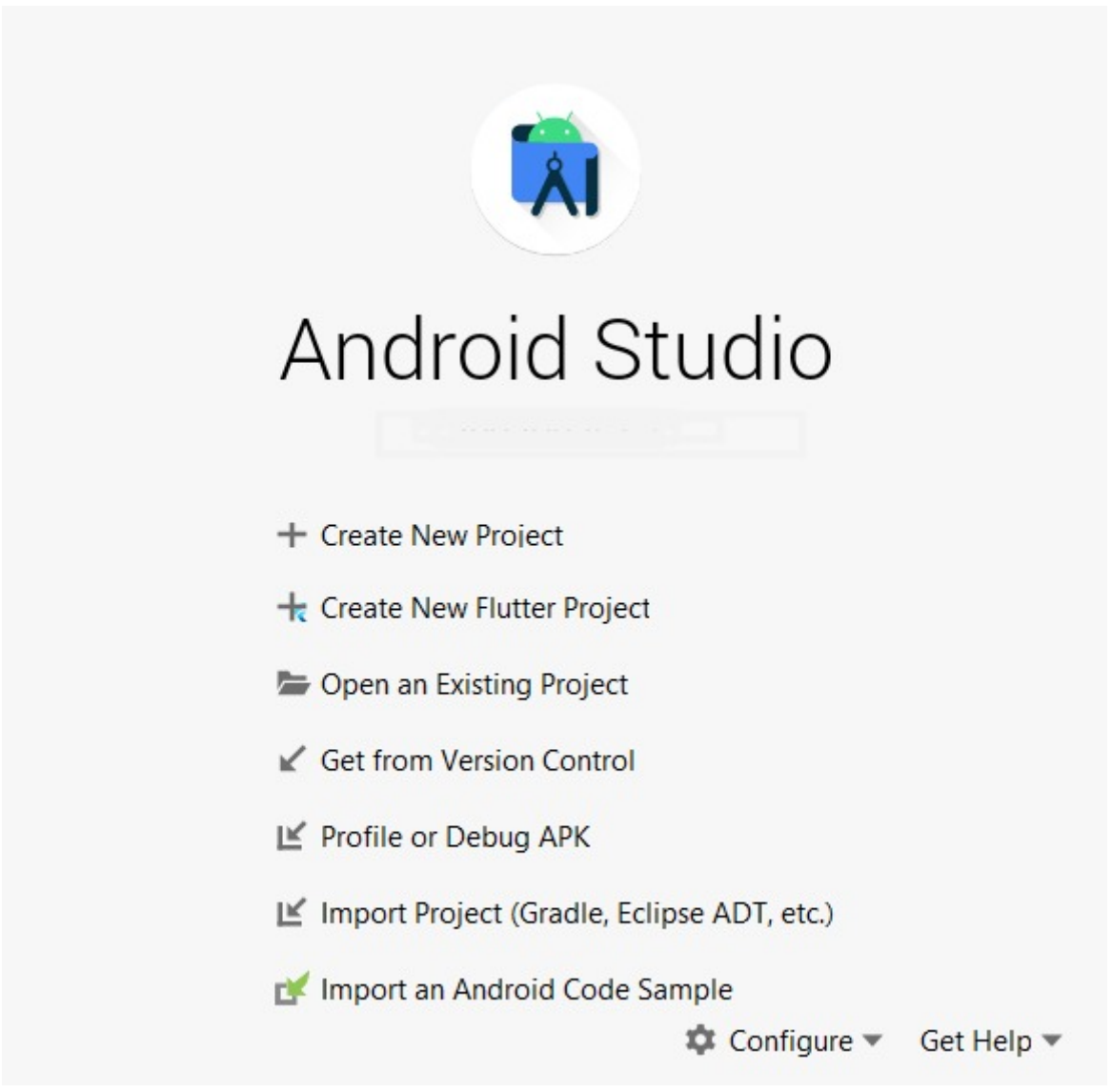
> Catatan: Untuk migrasi dari Android Studio versi 2.x ke versi 3.4 ke atas, terkadang perlu menyesuaikan beberapa kode terutama pada berkas *gradle*-nya. Silakan kunjungi tautan ini untuk keterangan lebih lanjut.

Introduction to Android Studio

Agar mulus menggunakan Android Studio, sebaiknya Anda tahu bagaimana strukturnya terlebih dahulu. Bila terbiasa menggunakan produk IntelliJ IDEA lainnya, maka akan mudah bagi Anda untuk menavigasi tata letak dan struktur Android Studio ini. Yang berbeda adalah komponen tambahan pendukung yang membantu pengembangan dan pembuatan aplikasi Android. *Yuk*, kita kenali Android Studio, *right from the git-go!!*

Project Android Pertama

Kali pertama menjalankan Android Studio, Anda akan melihat tampilan seperti berikut ini.



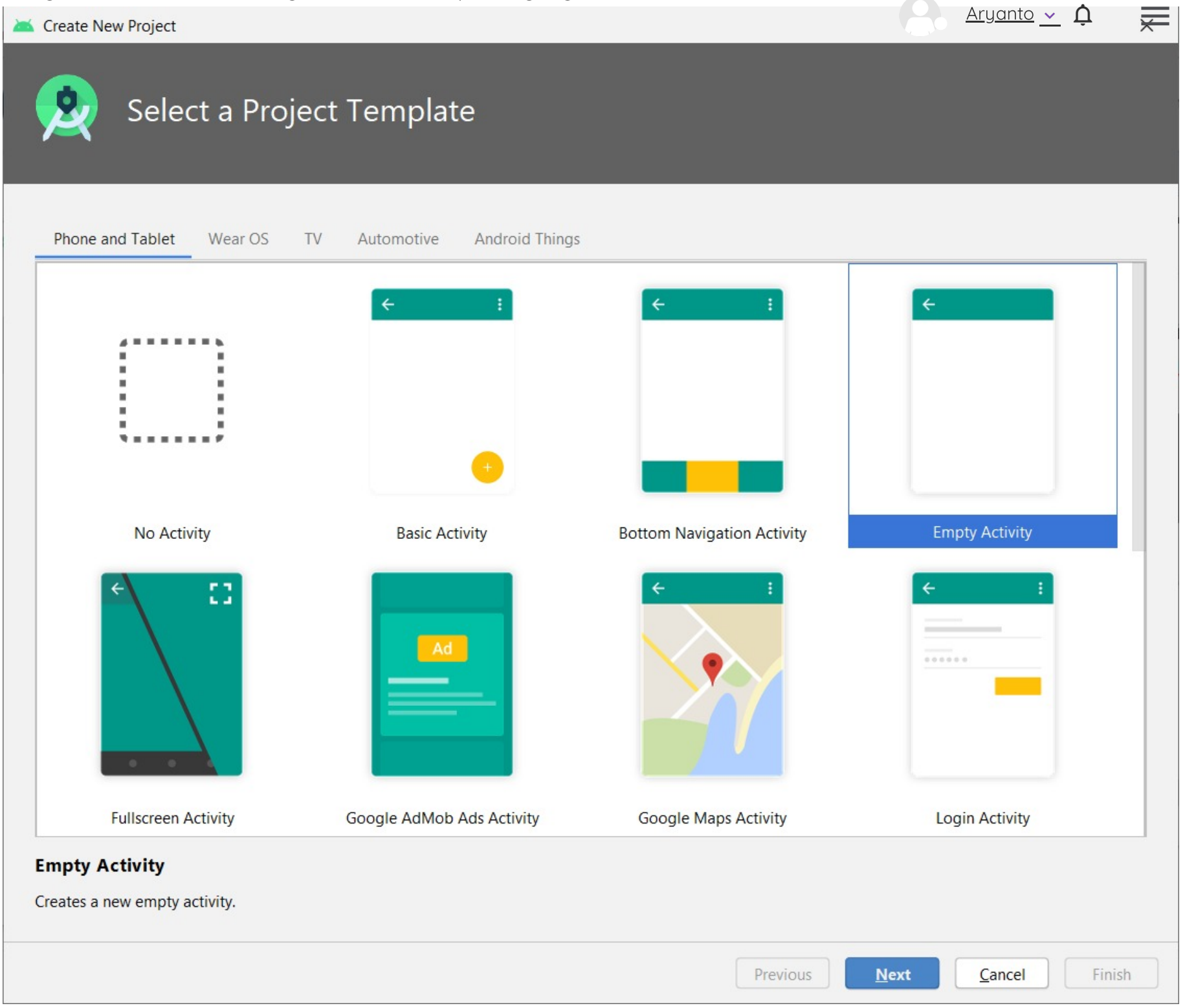
Untuk memulai proyek baru pilihlah **“Start a new Android Studio project”**.

Project Wizard

Setelah proses instalasi Android Studio, mari kita mulai membuat sebuah aplikasi *Hallo World*, aplikasi pertama Anda.

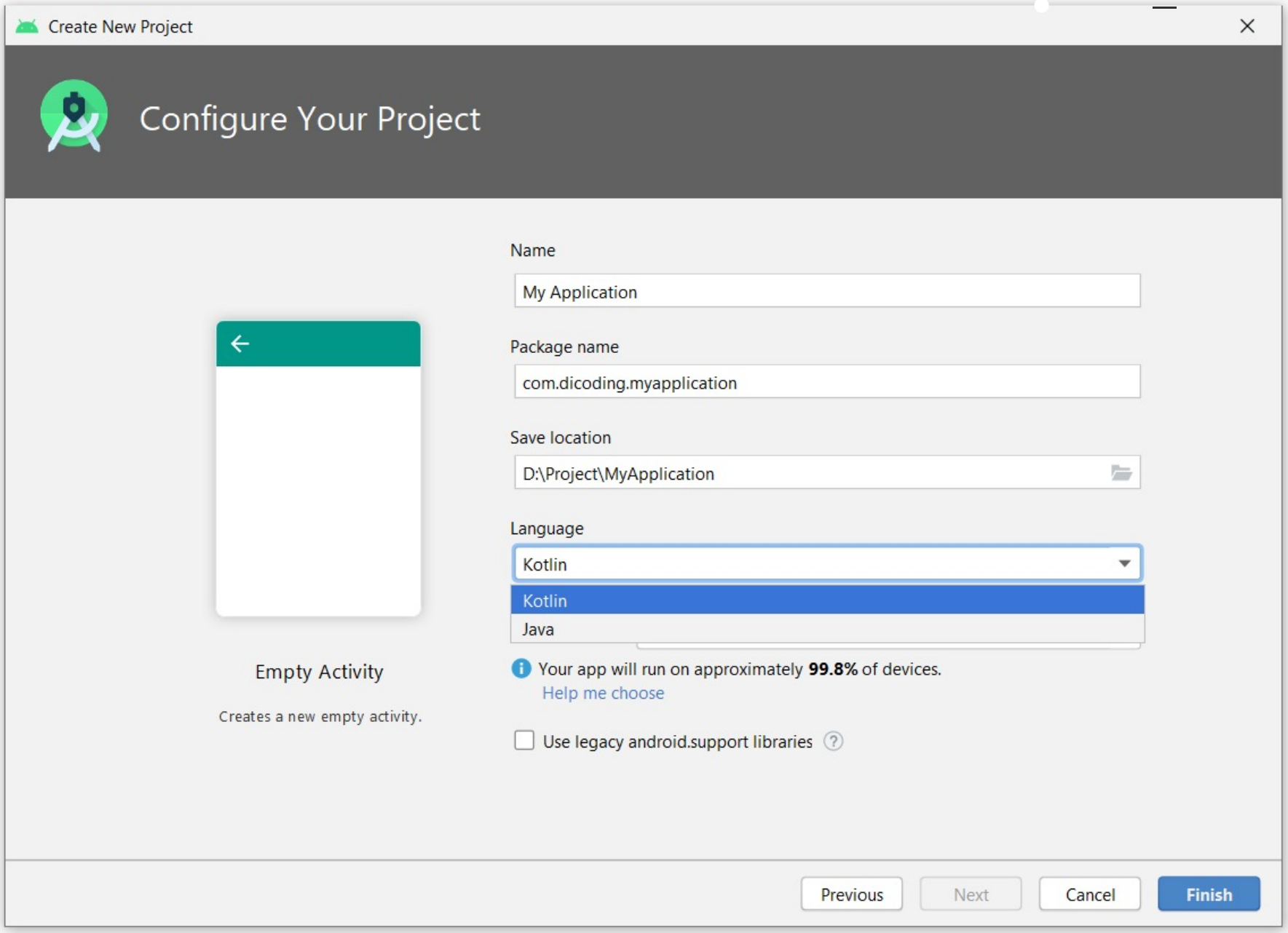
1. Setelah melakukan **Start a new Android Studio project**, Anda diminta untuk melakukan konfigurasi dalam pembuatan proyek baru Anda. Dalam dialog ini Anda bisa memilih *template* dari Aplikasi yang akan Anda buat. Terdapat beberapa *template* yang bisa kita gunakan seperti Empty Activity, Login Activity, Navigation Drawer Activity dan lain-lain. Di dalam

dialog ini kita bisa memilih target *devices* dari aplikasi yang akan kita buat.



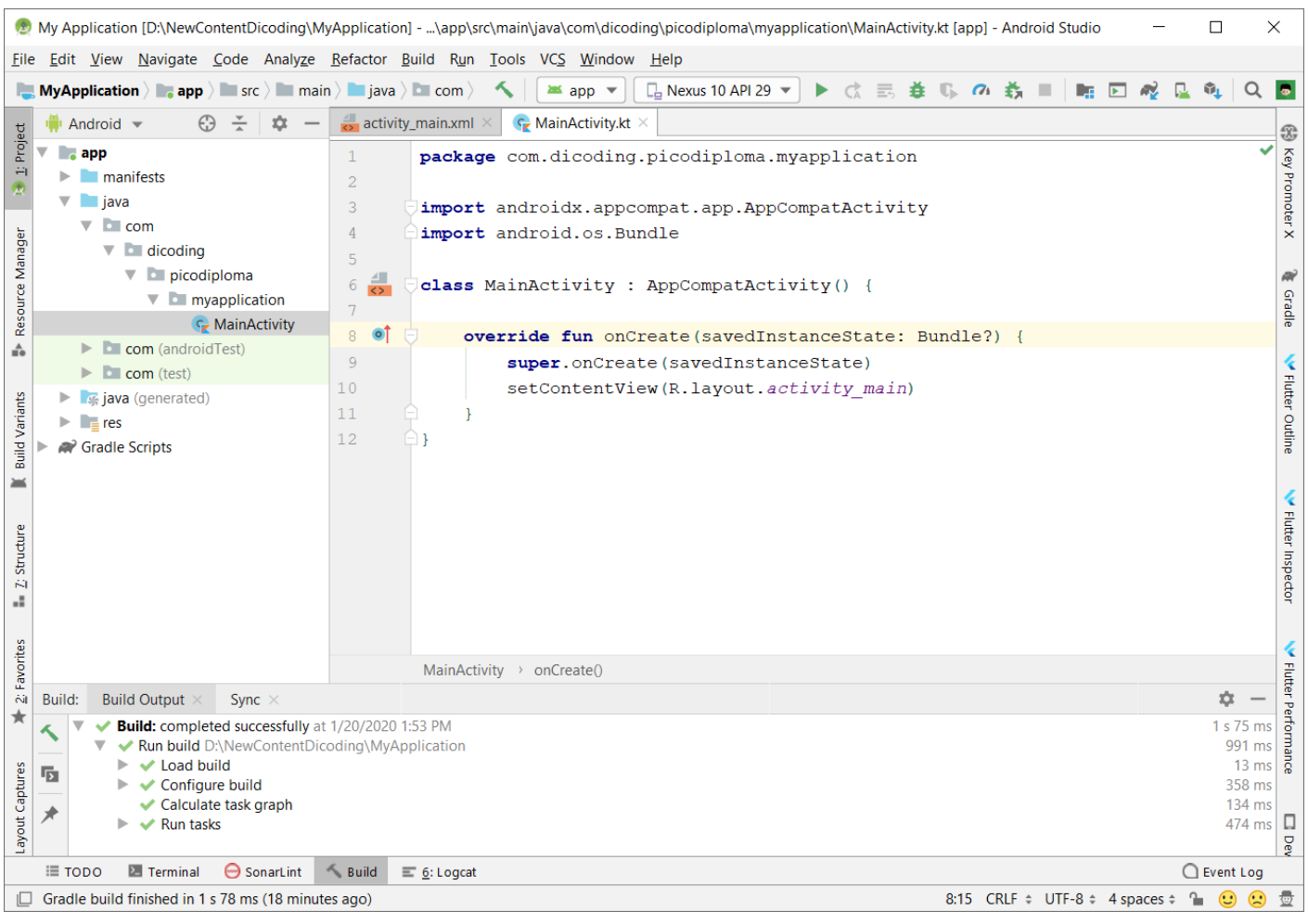
2. Dalam dialog ini Anda bisa memberi *nama dari aplikasi*, *lokasi proyek* dan *nama package*. Nama package akan digunakan dalam identifikasi unik dari aplikasi kita ketika sudah di-*publish*. Kita juga dapat mengganti dari direktori di mana proyek kita akan disimpan. Kita juga bisa mengganti nilai minimum SDK, yang berfungsi untuk membatasi penggunaan API pada sebuah aplikasi. Di dalam dialog ini juga Anda bisa memberi nama dari Activity yang pertama kali yang Anda buat. Selain itu Anda juga bisa mengganti bahasa *default* pada aplikasi tersebut menjadi bahasa Java atau

Kotlin.

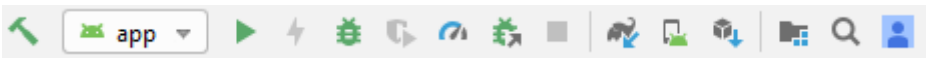


Antarmuka Android Studio

OK, membuat proyek pertama kali di Project Wizard, *done*! Kali ini kita akan menemui tampilan penuh Android Studio. Untuk meningkatkan produktivitas, mari kita bahas lebih jauh tentang antarmuka (*interface*) dari Android Studio ini.

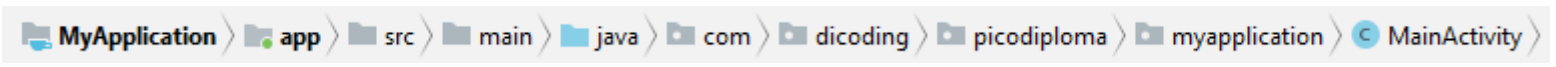


Di atas adalah *screenshot* tampilan penuh IDE Android Studio berbasis IntelliJ IDEA. Mungkin tampilan tersebut akan berbeda dengan tampilan di layar Anda karena perbedaan konfigurasi dan versi Android Studio.



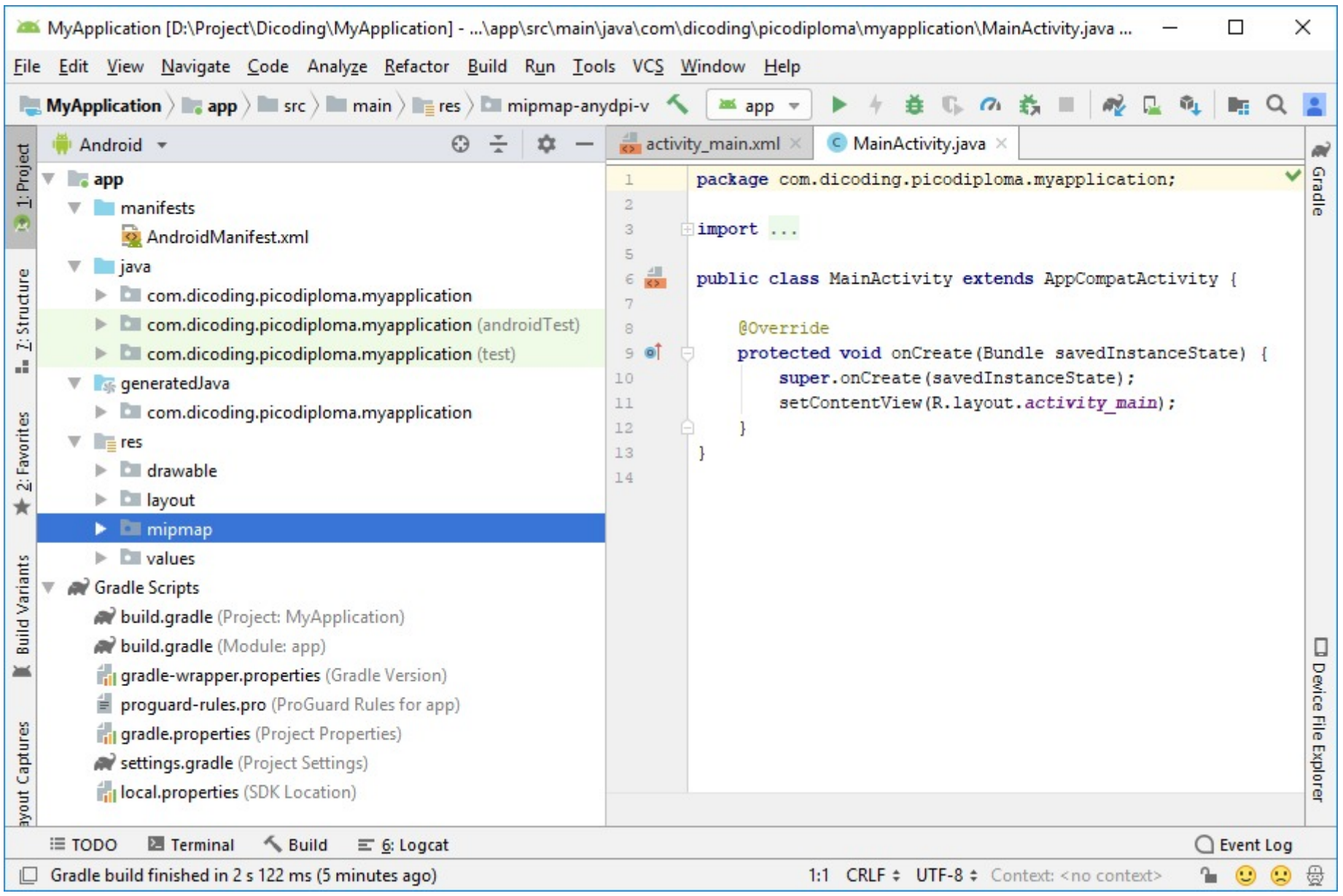
Merupakan *tools* yang sering digunakan dalam *development* seperti *copy/paste, build,* menjalankan aplikasi, hingga menjalankan emulator.

Navigasi



Membantu melihat struktur dari kedalaman (*depth*) dan posisi proyek yang saat ini sedang dibuka.

Project Explorer dan Editor

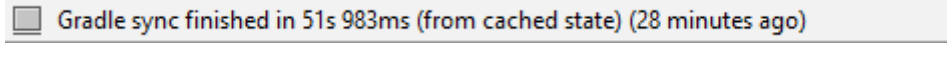


Merupakan bagian utama dari IDE Android Studio di mana kita menuliskan kode. Pada tampilan di atas, sebelah kiri adalah struktur proyek kita dan sebelah kanan adalah editor. Bagian ini akan dibahas lebih detail di poin selanjutnya.

Tool window bar

Tools menu yang mengelilingi editor ini merupakan *button* yang dapat di-*expand* ataupun untuk menampilkan tools secara detail dan individual.

Status Bar



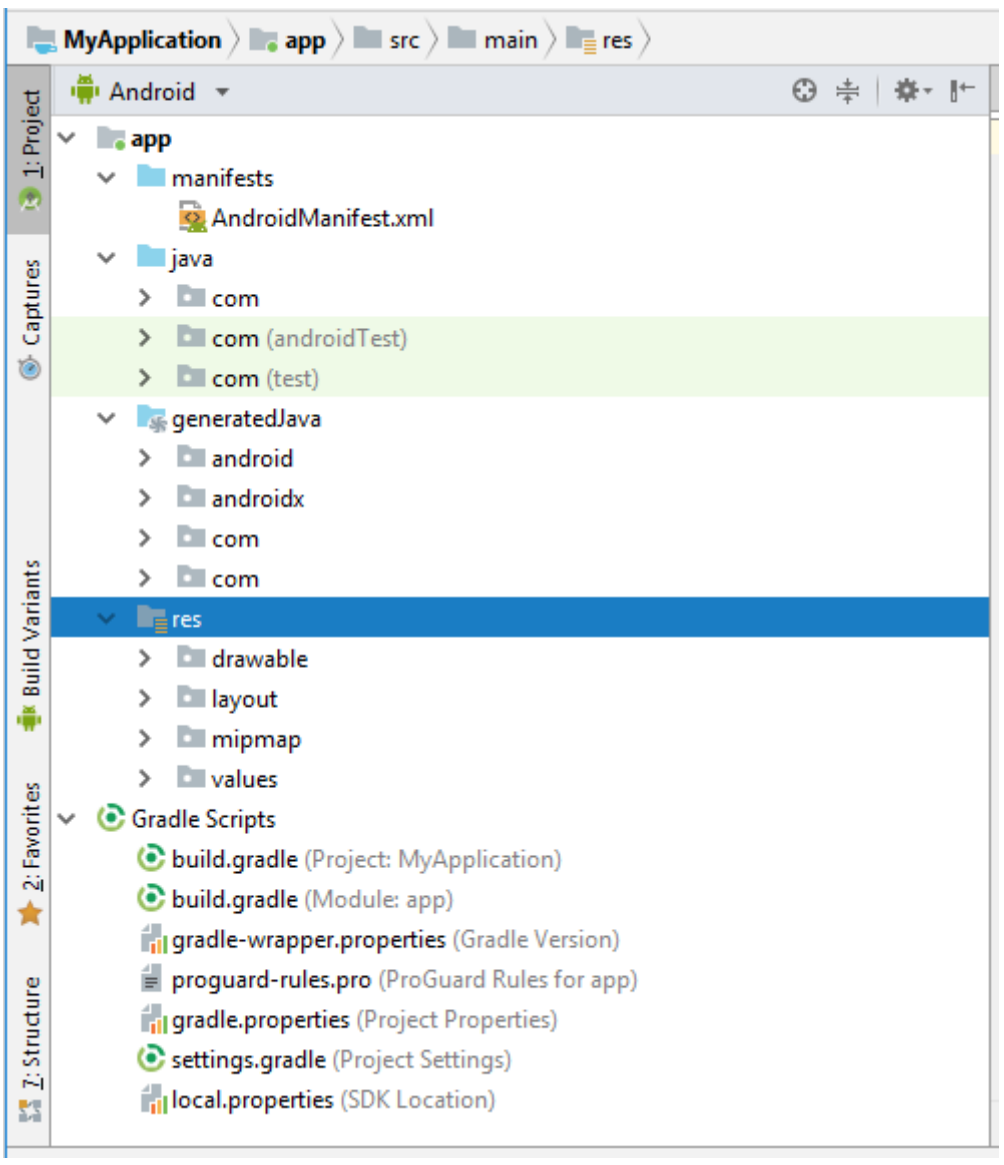
Terletak di bagian paling bawah dalam Android Studio, ia berfungsi untuk menampilkan status proyek kita dan pesan peringatan (*warning message*), bila ada.

Struktur Proyek

Setiap proyek di Android Studio berisi satu atau beberapa modul dengan berkas kode sumber dan berkas sumber daya. Jenis-jenis modul mencakup:

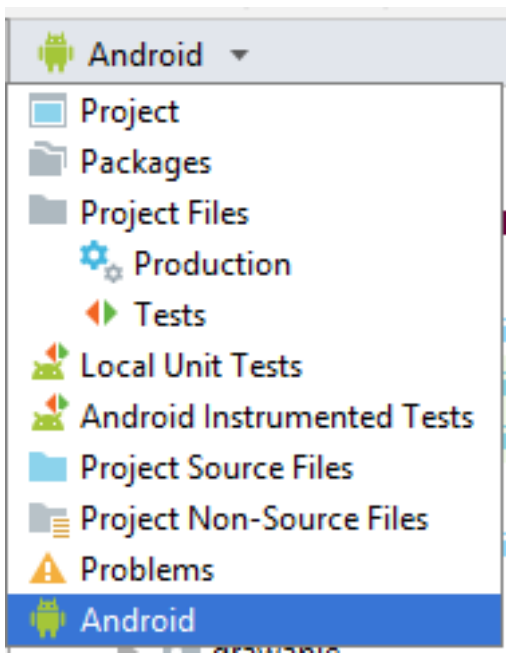
- Modul Aplikasi Android,
- Modul Pustaka, dan
- Modul Google App Engine.

Secara *default*, Android Studio akan menampilkan berkas proyek Anda dalam tampilan proyek Android, seperti yang ditampilkan dalam gambar berikut:



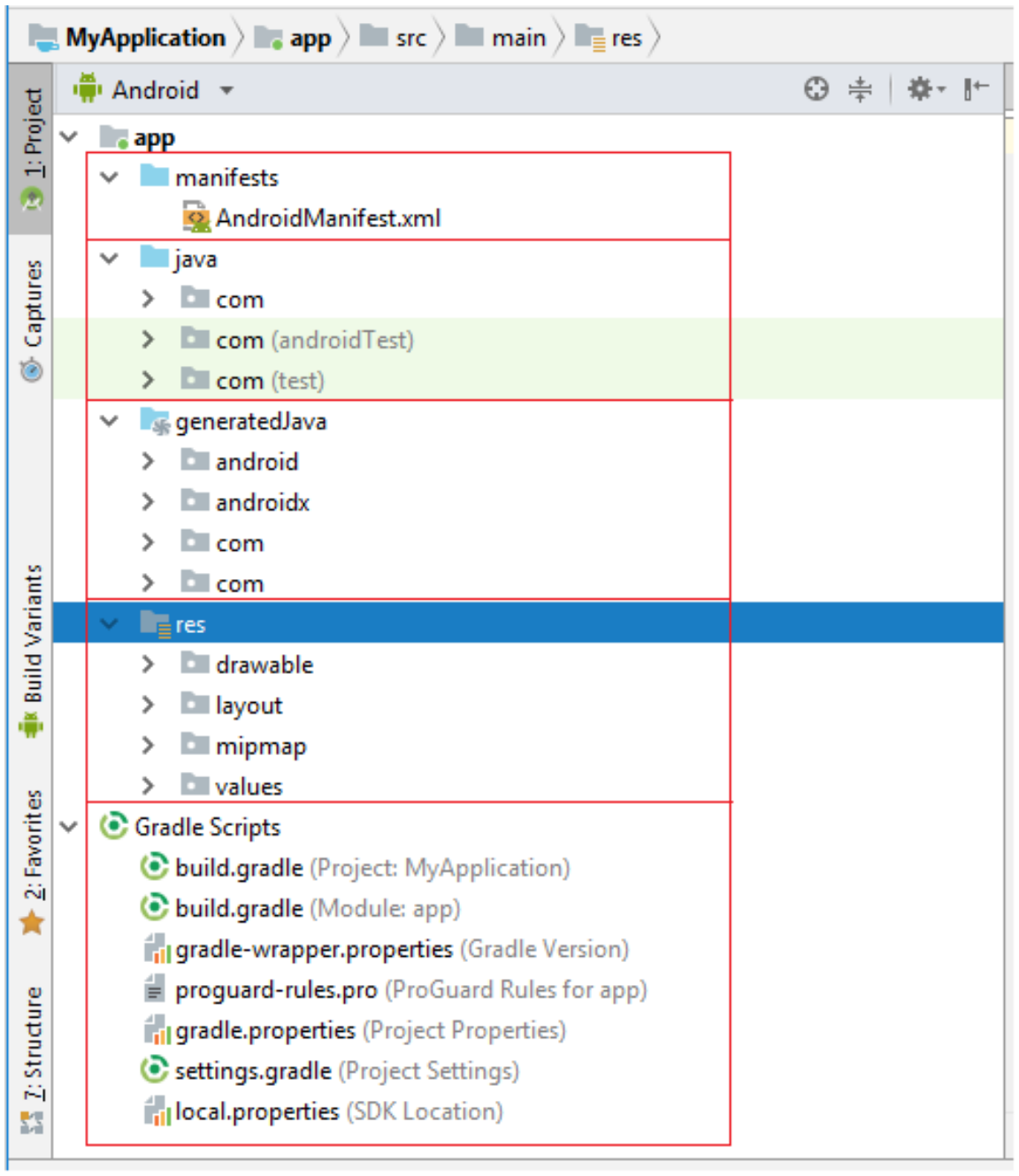
Tampilan disusun berdasarkan modul untuk memberikan akses cepat ke berkas sumber utama proyek Anda.

Secara *default* ketika kita membuat proyek baru, Android Studio akan menampilkan struktur yang lebih ringkas dan cepat sesuai dengan kebutuhan pengembangan Android. Bila ingin melihat struktur proyek dalam bentuk selain standar Android, kita dapat mengubahnya dengan tombol *dropdown* yang terdapat di atas *project structure*.



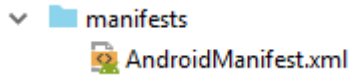
Pada bagian ini kita dapat mengganti tampilan *project structure* sesuai kebutuhan.

Mari kita bahas lebih detail tentang proyek yang baru saja kita buat.



Masing-masing modul aplikasi berisi folder berikut

Android Manifest



Manifest adalah salah satu berkas yang harus ada di dalam sebuah proyek Android. Manifest akan memberikan beragam informasi penting kepada sistem Android. Sistem perlu mengetahui apa yang akan digunakan oleh aplikasi sebelum dijalankan.

Beberapa fungsi yang ada di dalam Android Manifest adalah sebagai berikut:

- **Nama Package**

```
1.  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2.      package="com.dicoding.picodiploma.myapplication">
```

  Nama dari sebuah package bertindak sebagai identitas unik dari sebuah aplikasi. Identitas ini juga yang akan digunakan di dalam *store* untuk membedakan satu aplikasi dengan aplikasi lainnya. Jangan pernah mengganti *value* di dalam package karena nantinya akan dikenali sebagai aplikasi yang lain jika sudah masuk ke dalam *store*.
- **Komponen Aplikasi**
  Berfungsi untuk mendeskripsikan komponen dari aplikasi mulai dari activity, services, broadcast receiver, dan content provider.

```
<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" />
```

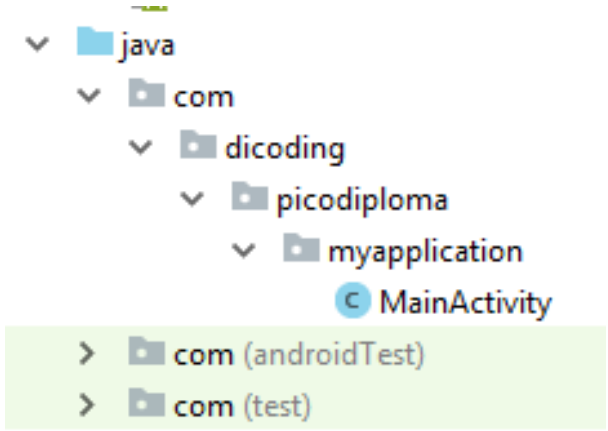
Komponen aplikasi semuanya berada di antara tag **<application>**. Ia juga berfungsi sebagai penamaan kelas yang mengimplementasi komponen dan mendeskripsikan kemampuannya seperti intent-filter, di mana fungsinya mendeskripsikan bahwa komponen itu adalah yang pertama kali dijalankan.

- **Permission**
  Mendeklarasikan *permission* apa saja yang harus dimiliki oleh aplikasi untuk akses ke dalam komponen API seperti internet, *external storage*, *contact*, dan juga untuk berinteraksi dengan aplikasi lainnya. Sebagai contoh ini adalah kode untuk *permission* Internet

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Kode ini biasanya diletakkan di atas tag <**application**> dan masih di dalam tag <**manifest**>

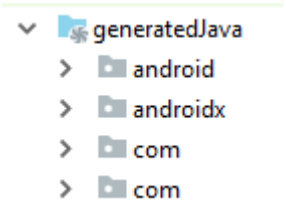
Java



Merupakan salah satu folder yang sering dipakai, berisi berkas *source code* kita yang ditulis dalam bahasa Java/Kotlin, termasuk juga kode Unit Test dan androidTest (Instrumentation Test).

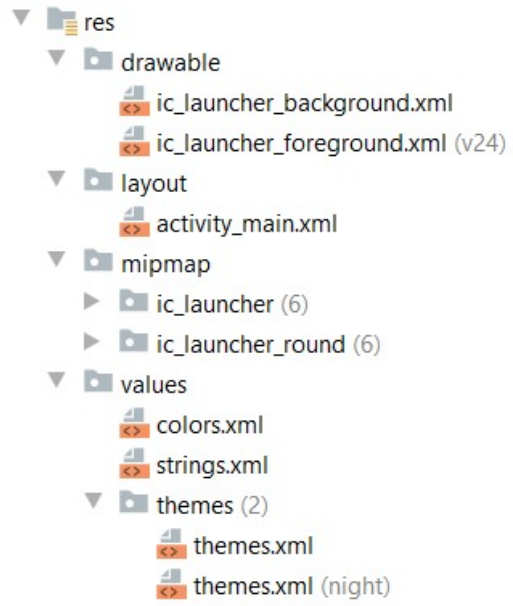
> Catatan: Penamaan folder "java" bisa juga diganti sesuai dengan bahasa yang dipakai, misal "kotlin."

GeneratedJava



Berisi file hasil dari *generate* library atau sebuah kelas dari proyek Android.

Res / Resource

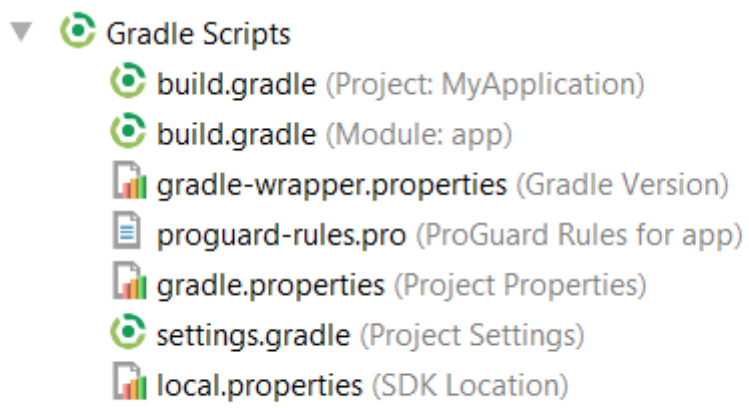


Mengatur *resource* di dalamnya, yang mana bukan berupa kode, melainkan *layout* aplikasi, sumber gambar, ikon, hingga *style*. Di dalam folder **res** ini juga terdapat sejumlah folder yang sudah diatur dan dikategorikan sesuai kebutuhan, seperti

- **Drawable**
  Untuk menyimpan berkas gambar maupun ikon.
- **Layout**
  Salah satu folder yang sering dipakai untuk berkas desain aplikasi.
- **Mipmap**
  Untuk menyimpan logo dalam berbagai dimensi.
- **Values**
  Berisi berbagai macam sumber data, seperti colors.xml untuk warna, strings.xml untuk teks, dimens.xml untuk ukuran, dan styles.xml untuk membuat *style* atau *template*.

Gradle

Gradle merupakan *open source build automation system*. *Automation system* berguna untuk mengotomatisasi proses pembuatan dari *software build* dan proses-proses terkait lainnya termasuk *compile source code* menjadi *binary code, packaging binary code*, dan menjalankan *automated test*.



build.gradle (Project: MyApplication)

Merupakan *software build* tingkat teratas yang meliputi keseluruhan dari proyek dari sebuah aplikasi. Di dalamnya berisi konfigurasi semua modul yang ada di dalam proyek.

build.gradle (Module: app)

Merupakan *software build* yang ada pada setiap modul di dalam proyek sebuah aplikasi. Beberapa konfigurasi yang diedit di antaranya adalah android settings, defaultConfig dan productFlavors, buildTypes, dan dependencies.

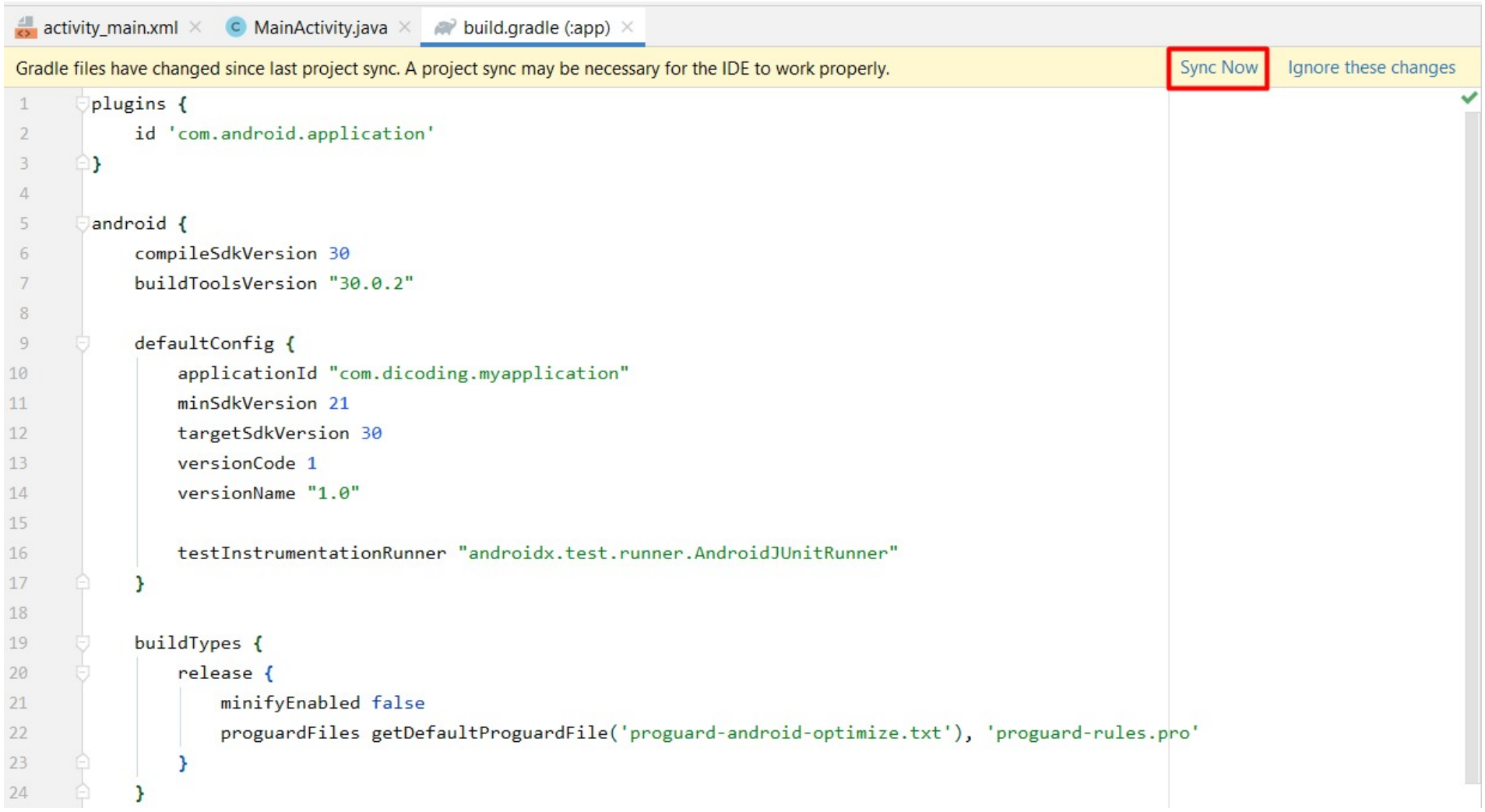
```
plugins {
    id 'com.android.application'
}

android {
    compileSdkVersion ...
    buildToolsVersion "..."

    defaultConfig {
        applicationId "com.dicoding.picodiploma.myapplication"
        minSdkVersion ...
        targetSdkVersion ...
        versionCode 1 //incremental
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
```

- **Android Settings**
  Di dalam *block* android kita bisa menetapkan compileSDKVersion dan buildToolsVersion.
- **Default Config**
  Di dalamnya terdapat *properties* seperti applicationID, minSdkVersion, targetSdkVersion dan test information.
- **Build Types**
  Di dalamnya terdapat *properties* dari debuggable*, ProGuard enabling, debug signing, version name suffix* dan *test information*.
- **Dependencies**
  Di dalamnya terdapat informasi tentang *library* yang digunakan oleh aplikasi.

Yang perlu diperhatikan di sini yaitu di bagian versionCode , misal kita sudah *publish* aplikasi ke PlayStore dengan *version code* 1, maka ketika kita ingin meng-*update* aplikasinya lagi ke Playstore, kita perlu merubah *version code*-nya menjadi 2 (*incremental*), kalau tidak Anda ubah maka PlayStore akan menolak APK yang di-*upload*.

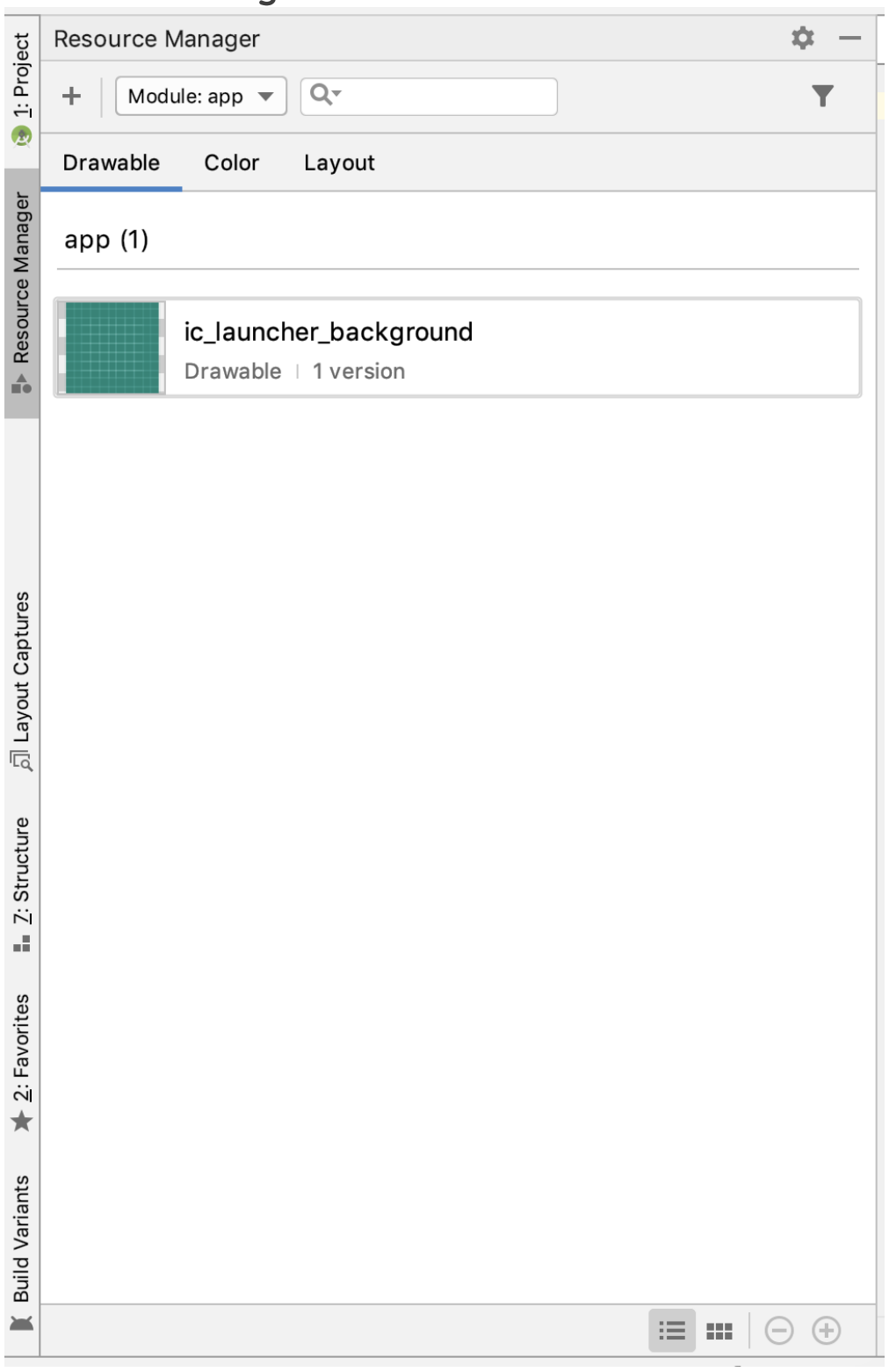
Sync Project

Setiap kali terjadi perubahan informasi di dalam **build.gradle** kita harus melakukan sinkronisasi terlebih dahulu. Tombol **sync now** akan muncul pada sebelah kanan atas ketika terjadi perubahan.



Setelah proses sinkronisasi selesai maka akan muncul informasi pada log apakah proses sinkronisasi berhasil atau tidak.

Resource Manager



Resource Manager, berfungsi untuk memanajemen segala *resource* yang ada di proyek Anda seperti gambar, warna, layout dll. Jika Anda perhatikan struktur dari **res**, dengan menggunakan Resource Manager semua *resource* akan tampil di sini dan Anda bisa memanipulasinya sesuai kebutuhan Anda.

Useful Tools pada Android Studio

Android Studio menyediakan fasilitas yang *powerful* di bawah IntelliJ IDEA ini. Banyak *tools* milik Android yang membantu kita saat mengembangkan Aplikasi. Mari kita bahas *tools* yang sering digunakan dan manfaatnya.

Shortcut

Pencarian:

- **Shift+Shift**
  Search Everywhere, atau dapat dikatakan pencarian semua jenis berkas yang masih dalam 1 proyek.
- **Ctrl+F**
  Find, pencarian teks dalam salah satu berkas.
- **Ctrl+Shift+F**
  Find in path, pencarian teks di seluruh berkas proyek.
- **Ctrl+Shift+A**
  Find action, pencarian aksi atau perintah-perintah yang ada di Android Studio.
- **Ctrl+R**
  Replace, mengganti teks di dalam berkas.

Navigasi:

- **Ctrl+N**
  Find Class, navigasi ke kelas tertentu.
- **Ctrl+Shift+N**
  Find file, navigasi ke berkas.
- **Ctrl+B**
  Go to declaration, lompat ke deklarasi yang dipilih.
- **Alt+↑**
  Lompat ke method sebelumnya.
- **Alt+↓**
  Lompat ke method sesudahnya.
- **Ctrl+G**
  Go to line, lompat ke baris tertentu.
- **Ctrl+E**
  Membuka berkas teranyar (*recent file*).
- **Ctrl+Left Mouse (or) Ctrl+Alt+F7**
  Melihat penggunaan pada variabel/objek yang diklik.
- **Alt+F7 / Ctrl+F7**
  Melihat penggunaan variabel/objek yang dipilih di seluruh berkas proyek.
- **Ctrl+Shift+B**
  Mencari tahu implementasi dari variabel/objek yang dipilih.

Redaksi:

- **Ctrl+D**
  Menggandakan bagian yang dipilih.
- **Ctrl+Q**
  Melihat dokumentasi dengan tampilan minimal.
- **Ctrl+P**
  Melihat isi dari parameter, penting ketika melihat method dari Android atau library lain.
- **Ctrl+Space**
  Basic code completion, menampilkan saran untuk melengkapi kode Anda.
- **Ctrl+Shift+Space**
  Smart code completion, menampilkan saran kode untuk melengkapi kode Anda dengan lebih pintar (menampilkan apa yang benar-benar terkait dengan kode Anda).
- **Alt+Insert**
  Generate code, menghasilkan (*generate*) kode. Perintah ini sangat memudahkan ketika membuat constructor dan setter/getter.
- **Ctrl+Alt+L**
  Memformat ulang kode, merapikan kode.
- **Ctrl+Y**
  Delete One Line, Menghapus satu baris kode.
- **Ctrl+Alt+V**
  Create variable, Membuat teks yang diblok menjadi sebuah variabel.
- **Ctrl+Alt+M**
  Create method, Membuat teks yang diblok menjadi sebuah fungsi.
- **Shift+F6**
  Rename, untuk mengganti nama suatu file atau variabel maupun fungsi.

Run

- **Ctrl+F9**
  Make project, build project.

- **Ctrl+Shift+F9**

  Melakukan kompilasi pada berkas, package atau modul.
- **Shift+F10**

  Run. Menjalankan aplikasi ke *emulator* atau *devices.*
- **Shift+F9**

  Debug. Menjalankan aplikasi ke *emulator* atau *devices* dalam mode **Debug**, biasanya untuk keperluan *testing.*

Code Completion

Untuk meminimalisir salah ketik (*typo*) dalam pemanggilan *class*, *method* hingga variabel sebaiknya kita memanfaatkan Code Completion di Android Studio. Terdapat dua jenis *code completion* yang sering digunakan di Android Studio:

- ### Basic Code Completion

  Ctrl+Space

  Pemanggilan *code completion* standar untuk membantu kita melengkapi kode.

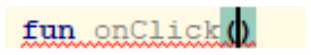
  Ketika kita akan memanggil sebuah variabel, cukup ketikkan *code completion* di atas. Saran pun akan diberikan.

- ### Statement Completion

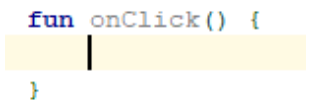
  Ctrl+Shift+Enter

  Perintah ini sangat membantu karena kita bisa menyelesaikan kode tanpa harus mengetik lengkap dan tanpa tanda kurung, kurung siku, kurung kurawal, dan banyak macam pemformatan lainnya.

  Kode di bawah ini ditulis sebelum menggunakan *shortcut*:



  Kemudian kita menggunakan Statement Completion. Lihat apa yang terjadi!



  Statement kita yang belum tuntas akan diselesaikan oleh Android Studio. Tentu hal ini akan mempercepat waktu kita dalam menggarap aplikasi.

  Selengkapnya Anda dapat mempelajarinya di sini.

Style dan Formatting

Gaya penulisan kode adalah seni dalam pemrograman. Kita memiliki *signature style* masing-masing, Semua tergantung pilihan kita sendiri. Tetapi kita tetap perlu memperhatikan bagaimana tata letak kode, apalagi bila suatu saat nanti kita membuat aplikasi bersama orang lain. Kode yang rapi itu enak dilihat dan memudahkan, baik kita maupun orang lain untuk membacanya. Secara *default* Android Studio memberikan *code style formatting* untuk tata letak kode yang kita miliki. Untuk menyesuaikan setelan *code style*, klik **File > Settings > Editor > Code Style** (**Android Studio > Preferences > Editor > Code Style** pada Mac.)

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

    if(isTrue){
    data = "Halo Dunia"
    }
}
```

Bagaimana menurut Anda tentang kode di atas? Ya tidak ada yang salah. Namun *code style* berantakan dan tidak indah untuk dilihat.
Nah, kini kita akan melakukan kode *formatting* dengan menggunakan shortcut ***Ctrl+Alt+L***.

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

    if (isTrue) {
        data = "Halo Dunia"
    }
}
```

Hasilnya lebih baik, bukan?

Mungkin bila kode yang kita miliki sedikit, tidak terlalu berpengaruh. Tapi bila baris kode sudah mulai kompleks, *formatting code* seperti ini akan sangat membantu.

Sample Code

Android Studio juga membantu kita menemukan kode yang berkualitas dan *best practice*-nya. Melalui Google, Android Studio memiliki *sample code* yang bebas kita gunakan dan manfaatkan untuk kebutuhan kita belajar atau membuat aplikasi Android. Dengan mengakses **File > New > Import Sample**, kita punya banyak pilihan contoh kode yang bisa dipakai.
Selengkapnya dapat kita jumpai di sini.

Keren kan? Jadi biasakan diri Anda menggunakan alat bantu dari Android Studio ini. Tak lain agar membantu dan mempercepat pembuatan aplikasi kita!
