



**BKPM**  
**( BUKU KERJA PRAKTEK MAHASISWA )**

APLIKASI MOBILE  
( SEMESTER 4 )

OLEH :  
FAISAL LUTFI AFRIANSYAH

**PROGRAM STUDI MANAJEMEN INFORMATIKA**  
**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI JEMBER**  
**TAHUN 2018**

**Praktikum ke : 1**

**Judul praktikum : Pengenalan Android Studio**

**Alokasi waktu : 2 x 50 menit**

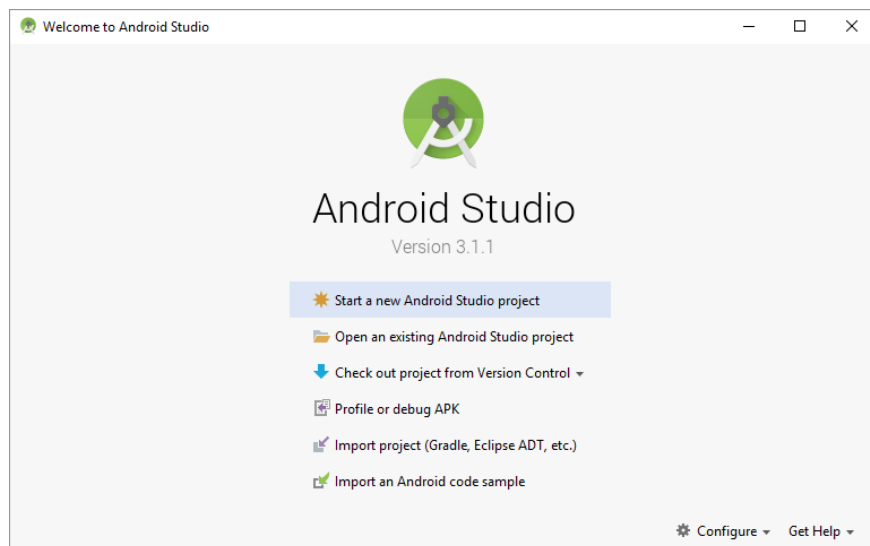
## **1. Tujuan Instruksional Khusus**

- a. Mahasiswa dapat memahami bahasa pemrograman yang digunakan untuk membangun aplikasi berbasis android
- b. Mahasiswa dapat memahami pengertian android studio dan fungsinya
- c. Mahasiswa dapat mengetahui project wizard, antar muka android studio, tools, navigasi, project explorer dan Editor, Tool Window Bar.

## **2. Teori**

### **Android Studio**

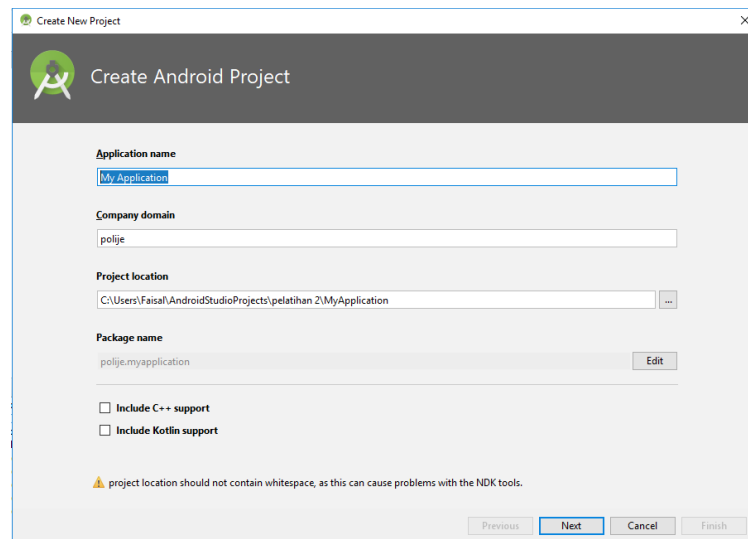
Kali pertama menjalankan Android Studio, akan melihat tampilan seperti berikut ini.



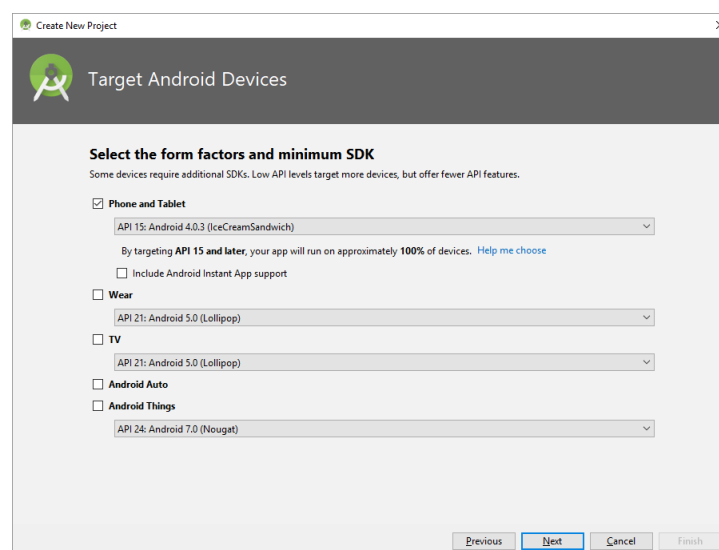
Untuk mulai proyek baru pilihlah “Start a new Android Project”.

## Project Wizard

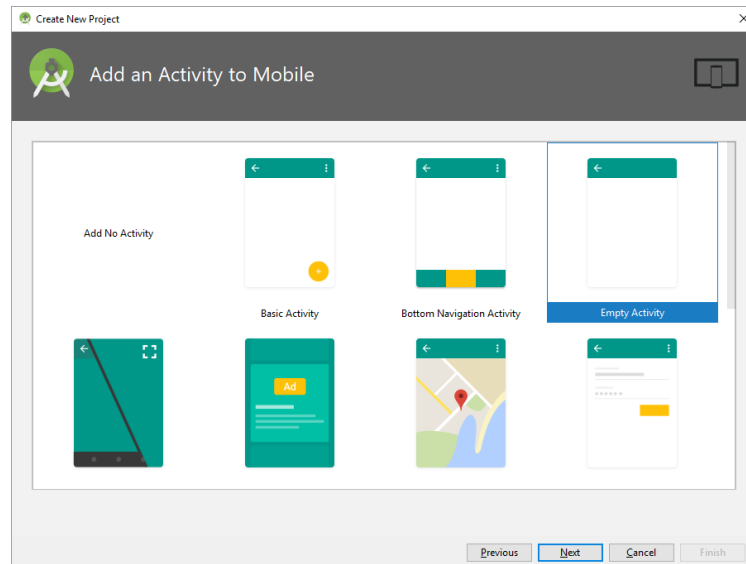
1. Dalam dialog ini kita bisa memberi nama aplikasi yang hendak dibuat, dan company domain. Company domain akan digunakan sebagai alat identifikasi ketika aplikasi akan dipublikasikan. Kita juga dapat mengganti lokasi di mana proyek akan disimpan.



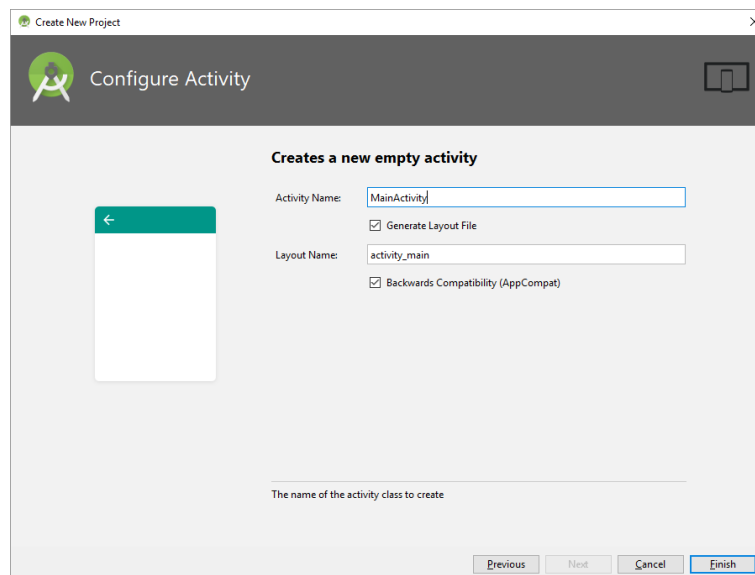
2. Dialog berikut ini adalah *target devices*, di mana kita bisa memilih peranti target dari aplikasi yang akan kita buat. Kita juga bisa menentukan nilai minimum SDK yang akan didukung oleh aplikasi.



3. Dialog di bawah ini adalah *default template*. Di dalamnya terdapat beberapa *template* yang bisa kita gunakan seperti Empty Activity, Login Activity, Navigation Drawer Activity dan lain-lain.

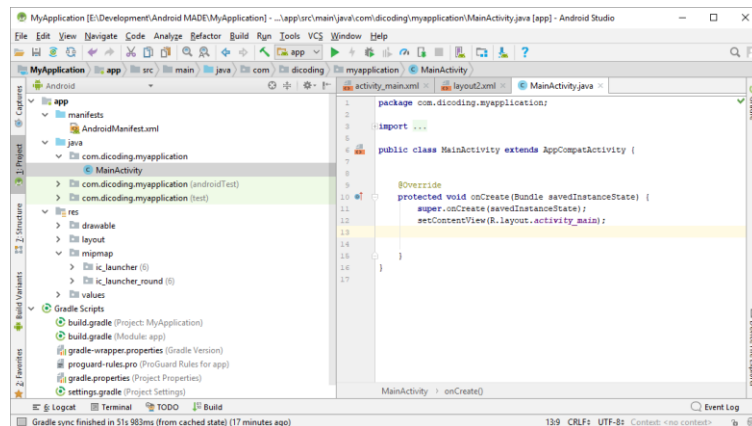


4. Dialog berikut ini adalah penamaan activity yang pertama kali kita buat. Usai memberi nama, tekan Finish.



## 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 karena perbedaan konfigurasi dan versi Android Studio.

## Tools



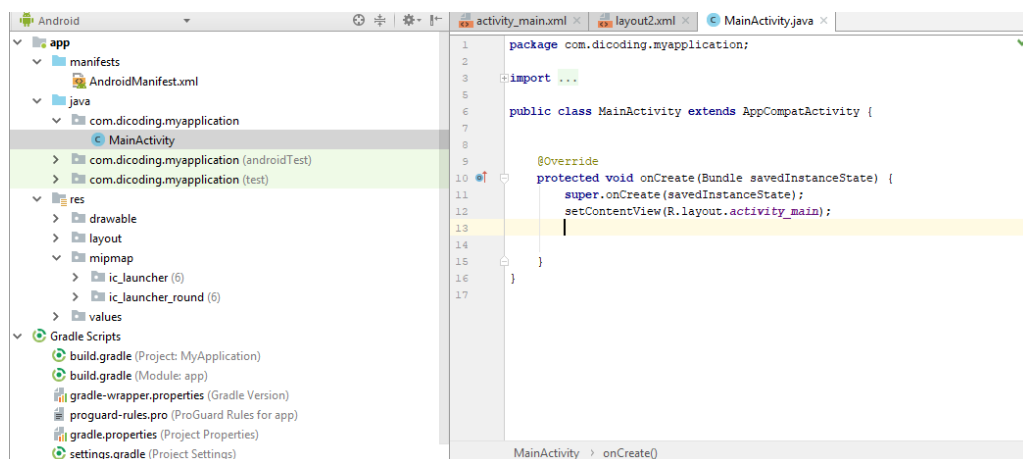
Merupakan Tools yang sering digunakan dalam development seperti copy/paste, build, menjalankan aplikasi, hingga menjalankan emulator.

## Navigasi



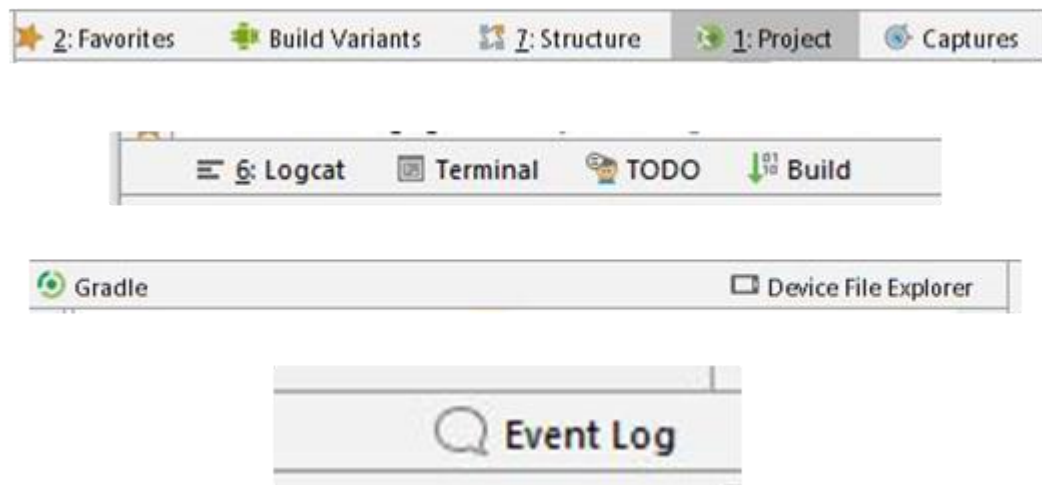
Membantu melihat struktur dari kedalaman (depth) dan posisi proyek yang sedang kita buka sekarang.

## Project Explorer dan Editor



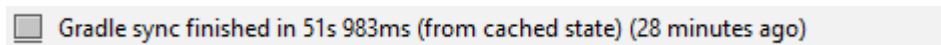
Merupakan bagian utama dari IDE Android Studio di mana kita menuliskan kode. Pada tampilan di atas, sebelah kiri adalah struktur proyek kita dan sebelah kanan adalah editor. Bagian ini akan dibahas lebih detail di poin selanjutnya.

### **Tool window bar**



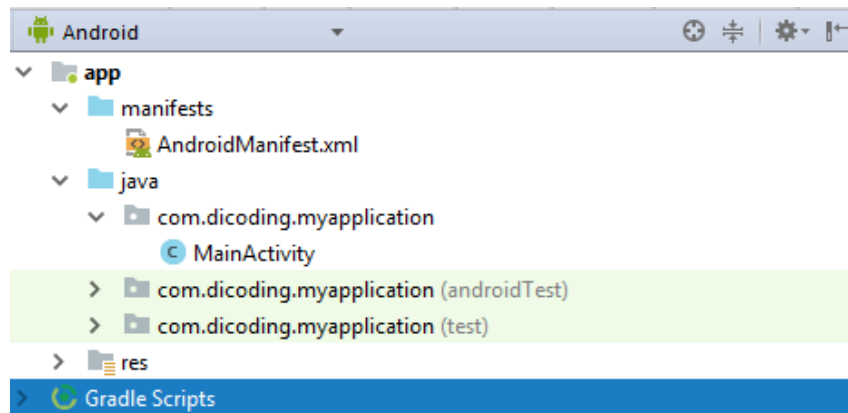
Tools menu yang mengelilingi editor ini merupakan button yang dapat di-expand ataupun untuk menampilkan Tools secara detail dan individual.

### **Status Bar**



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

## Project Structure

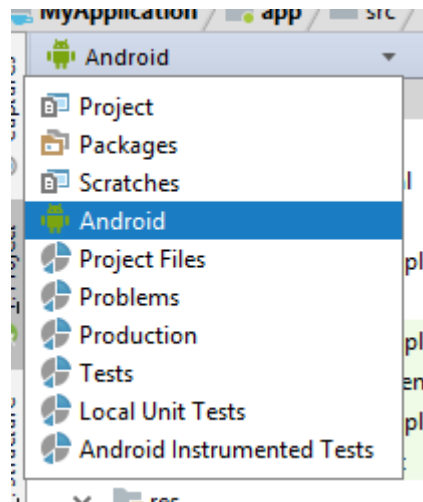


Setiap proyek di Android Studio setidaknya terdiri dari 1 modul atau lebih, dengan source code dan resource-nya. Jenis modul di antaranya:

- Android App Module
- Library Modul
- Google App Engine Module

Perhatikan pada Screenshot Project Structure di atas. Root project yang bernama app merupakan Android App Module.

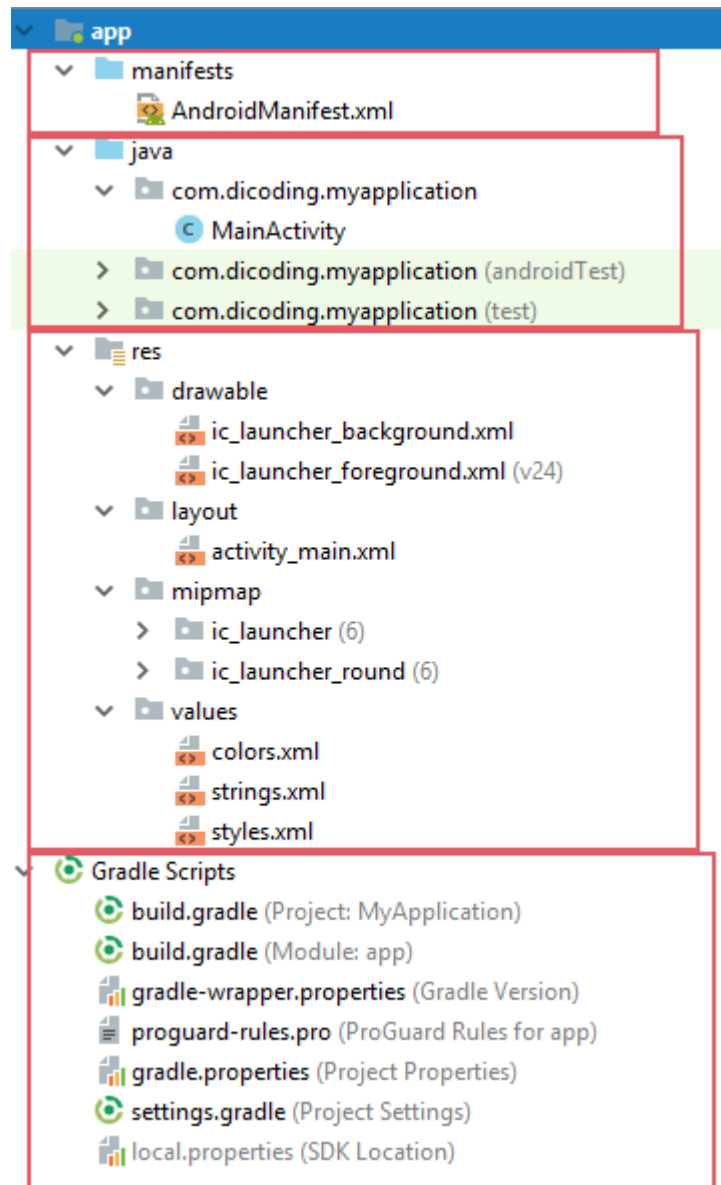
Secara default ketika kita membuat proyek baru, Android Studio akan menampilkan struktur yang lebih ringkas dan cepat sesuai dengan kebutuhan pengembangan Android. Bila ingin melihat struktur proyek dalam bentuk selain str Android, kita dapat mengubahnya melalui tombol dropdown yang terdapat di atas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



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

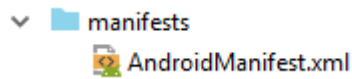
Mari kita bahas lebih detail tentang proyek yang baru saja kita buat.





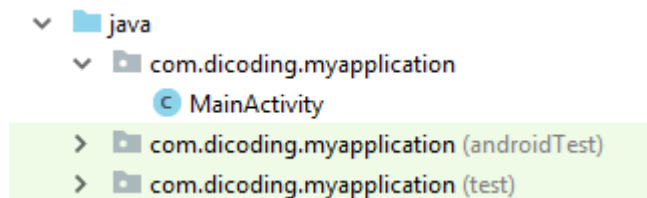
**Struktur proyek Android terdiri dari 4 bagian utama, antara lain Manifest, Java, Res, dan Gradle. Berikut penjelasan masing-masing.**

## Manifest



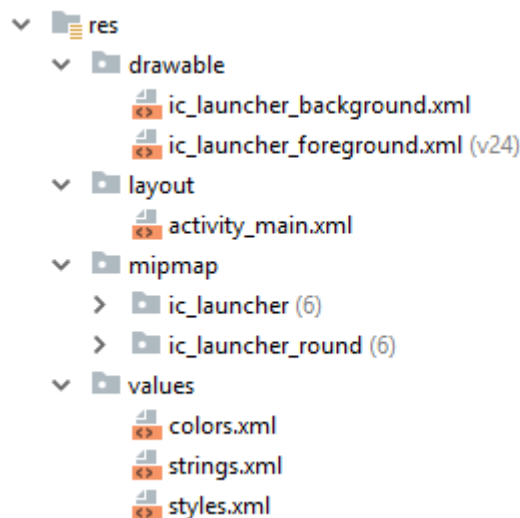
Manifest adalah salah satu berkas yang harus ada di dalam sebuah proyek Android. Manifest akan memberikan beragam informasi penting kepada sistem Android. Sistem perlu mengetahui apa yang akan digunakan oleh aplikasi sebelum dijalankan.

## Java



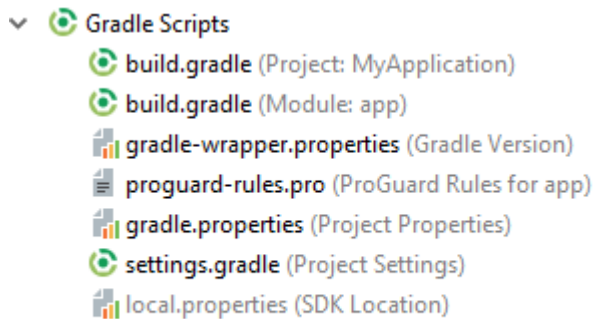
Berisi berkas source code kita yang ditulis dalam bahasa Java, termasuk juga kode Unit Test dan androidTest (Instrumentation Test).

## Res



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.

## Gradle



Secara default Gradle merupakan build tools yang digunakan oleh Android Studio. Fungsinya adalah untuk membantu kita mengkompilasi dan menjalankan source code aplikasi yang kita kembangkan berdasarkan konfigurasi di Gradle. Gradle sendiri juga mendukung manajemen proyek dalam hal penambahan library di luar framework Android.

## 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+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  
Menggkan 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 .
- Ctrl+Shift+Space  
*Smart code completion*, menampilkan saran kode untuk melengkapi kode dengan lebih pintar (menampilkan apa yang benar-benar terkait dengan kode ).
- 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.

## Run

- Ctrl + F9  
*Make project, build project.*
- Ctrl + Shift + F9  
Melakukan kompilasi pada berkas, package atau modul.
- Shift + F10  
*Run.*
- Shift + F9  
*Debug.*

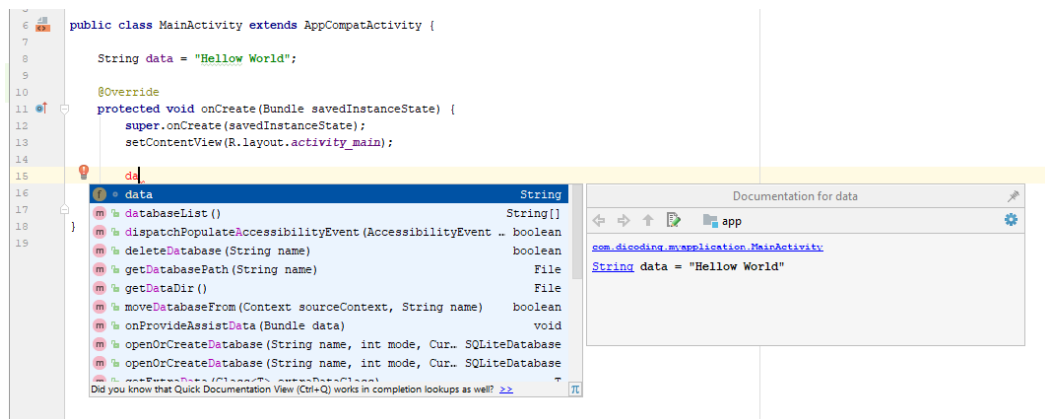
## Code Completion

Untuk meminimalisir salah ketik (typo) dalam pemanggilan class, method hingga variabel sebaiknya kita memanfaatkan Code Completion di Android Studio. Terdapat dua (2) jenis Code Completion yang sering digunakan di Android Studio:

### Basic Code Completion

Ctrl + Space

Pemanggilan Code Completion str untuk membantu kita melengkapi kode.



Ketika kita akan memanggil sebuah variabel, cukup ketikkan code completion di atas maka saran akan diberikan.

### Statement Completion

Ctrl + Shift + Enter

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

Kode di bawah ini ditulis sebelum menggunakan shortcut

```

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

    data = "mar"
}

```

Kemudian kita menggunakan Statement Completion. Lihat apa yang terjadi!

```

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

    data = "mar";
}

```

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

Selengkapnya dapat pelajari di sini.

<https://www.jetbrains.com/help/idea/2016.1/code-completion-2.html>

## 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.)

```

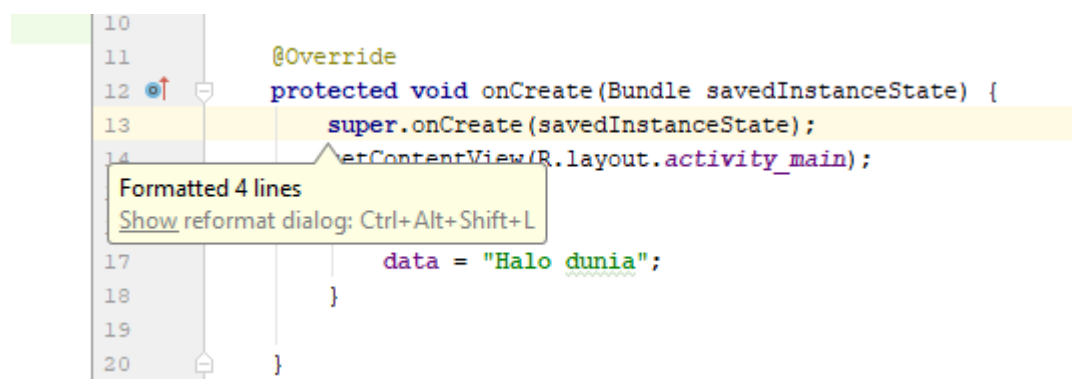
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    if(isTrue){
        data="Halo dunia";
    }
}

```

Bagaimana menurut tentang kode di atas? Ya tidak ada yang salah. Namun Code Style berantakan dan tidak indah dilihat.

Nah, kini kita akan melakukan kode formatting dengan menggunakan shortcut **Ctrl+Alt+L**.



```

@Override
protected void onCreate(Bundle savedInstanceState) {
    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.

### **3. Alat dan Bahan**

- a. BKPM
- b. Komputer
- c. LCD
- d. Alat Tulis Kantor (ATK)

### **4. Pelaksanaan Praktikum**

#### **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.  
<https://developer.android.com/studio/write/sample-code.html>

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

**Praktikum ke : 2**

**Judul praktikum : Run Emulator dan Device**

**Alokasi waktu : 2 x 50 menit**

## **1. Tujuan Instruksional Khusus**

- a. Mahasiswa dapat menjalankan emulator dan device
- b. Mahasiswa dapat membuat virtualization menggunakan emulator maupun device smartphone
- c. Mahasiswa dapat memahami fungsi dari emulator dan cara kerja emulator dan device

## **2. Teori**

### **Persiapan Running Menggunakan Emulator**

Sebelum menggunakan emulator, perlu memastikan beberapa hal berikut ini:

#### **Virtualization**

Untuk menjalankan emulator di dalam Android Studio, pastikan aspek virtualization. Sistem harus memenuhi persyaratannya, yakni ketentuan prosesor dan sistem operasi dari laptop / PC yang gunakan.

#### **Processor**

- Prosesor Intel: Jika laptop/pc menggunakan prosesor Intel, maka pastikan ia mendukung Intel VT-x, Intel EM64T (Intel 64), dan Execute Disable (XD) Bit functionality.
- Prosesor AMD Jika laptop/pc menggunakan AMD, maka pastikan bahwa ia support dengan AMD Virtualization (AMD-V) dan Supplemental Streaming SIMD Extensions 3 (SSSE3).

#### **Sistem Operasi**

- Intel : Jika menggunakan processor Intel maka dapat menjalankannya di sistem operasi Windows, Linux, maupun Mac
- AMD : Untuk prosesor AMD maka hanya bisa menjalankannya di sistem operasi Linux.

### **Menginstal Hardware Accelerated Execution Manager (HAXM)**

Setelah memenuhi persyaratan di atas, langkah selanjutnya adalah menginstal HAXM. HAXM adalah *hardware-assisted virtualization engine* yang menggunakan teknologi VT dari Intel untuk mempercepat aplikasi Android yang diemulasi di mesin host. HAXM diperlukan untuk menjalankan emulator di Android Studio.

HAXM diperlukan jika sistem operasi yang digunakan adalah Windows atau Mac. Untuk menginstalnya, ikuti petunjuk berikut ini.

1. Buka SDK Manager.
2. Pilih SDK Update Sites, kemudian hidupkan Intel HAXM.
3. Tekan OK.
4. Cari berkas installer-nya di directory folder sdk komputer ,  
~sdk\extras\intel\Hardware\_Accelerated\_Execution\_Manager\intelhaxm-android.exe.
5. Jalankan installer dan ikuti petunjuknya sampai selesai.

### **Menginstal Kernel-based Virtual Machine (KVM) untuk Pengguna Linux**

Karena HAXM hanya untuk Windows dan Mac, bagaimana dengan sistem operasi Linux? Untuk Linux, harus menginstal KVM. Sistem operasi Linux dapat support accelerated virtual machine dengan menggunakan KVM. Untuk instal KVM, bisa menggunakan perintah berikut ini.

```
$ sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils ia32-libs-multiarch
```

Selengkapnya dapat baca pada halaman

ini <https://developer.android.com/studio/run/emulator.html>

<https://developer.android.com/studio/run/emulator-acceleration.html>


### 3. Alat dan Bahan

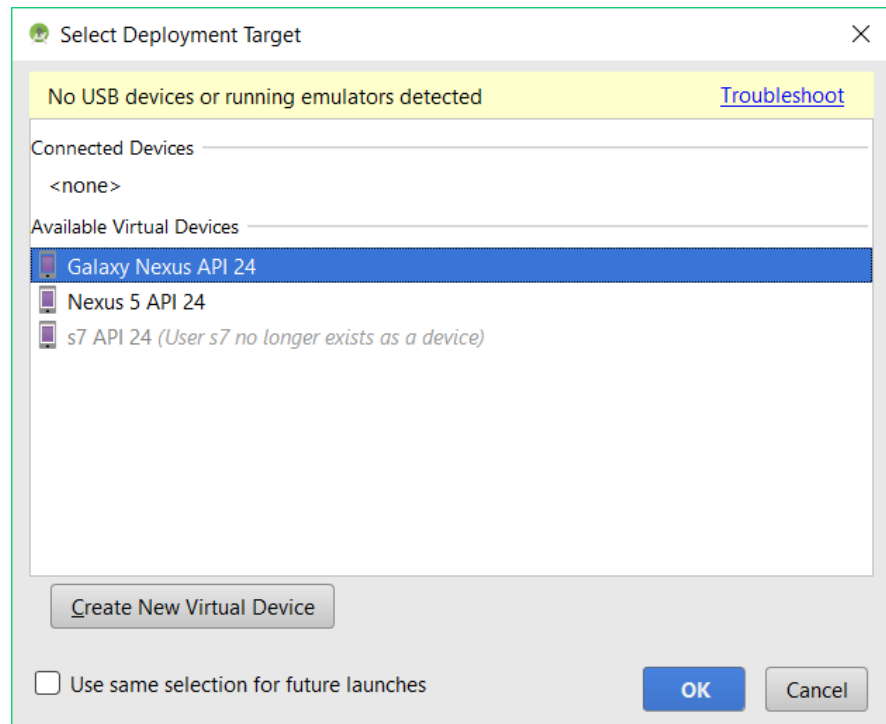
- a. BKPM
- b. Komputer
- c. LCD
- d. Alat Tulis Kantor (ATK)

### 4. Pelaksanaan Praktikum

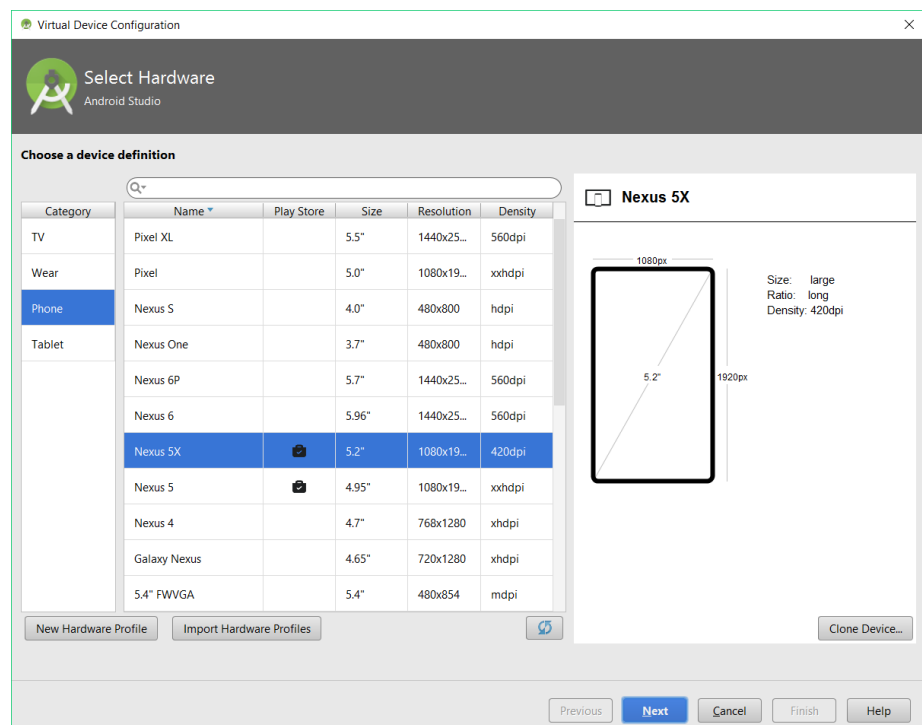
#### Menggunakan Emulator

Setelah memastikan bahwa virtualization bisa berjalan di komputer, ikuti langkah-langkah berikut untuk menjalankan aplikasi kita dengan menggunakan emulator built-in dari Android Studio.

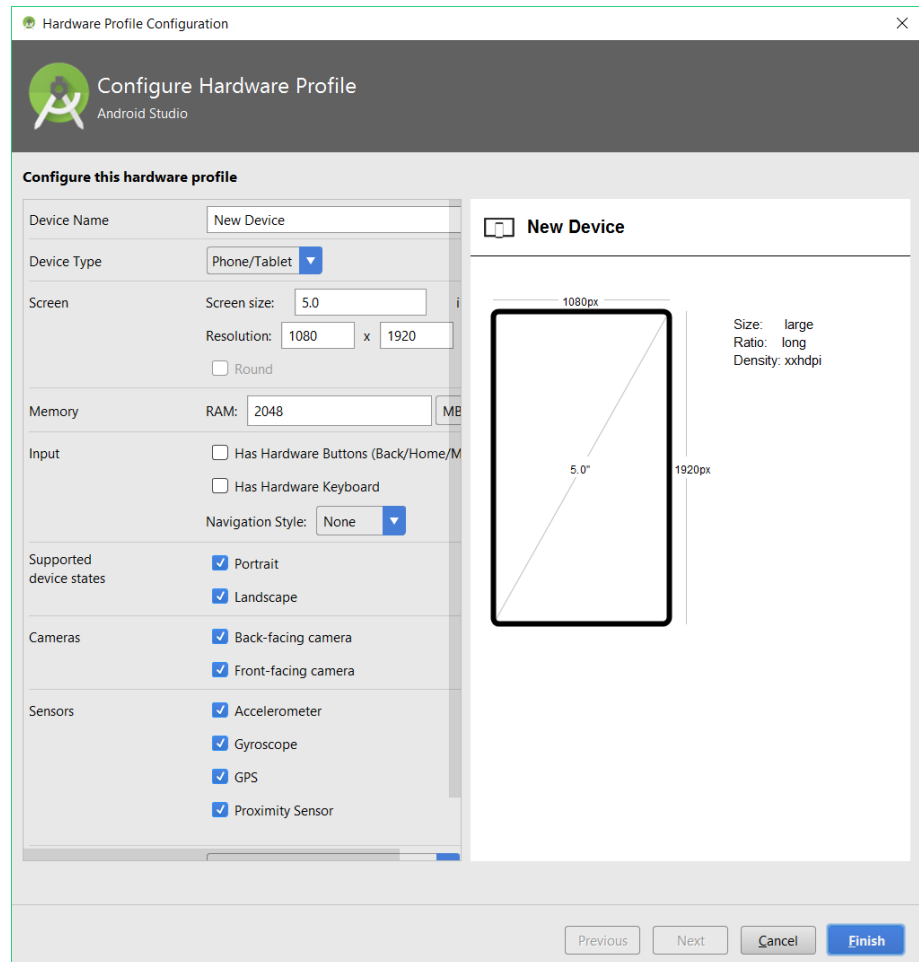
1. Jalankan ikon , kemudian akan muncul dialog seperti ini. Mari kita coba buat emulator baru dengan memilih **Create New Virtual Device**.



2. Akan muncul dialog dengan pilihan beberapa emulator yang bisa langsung gunakan.

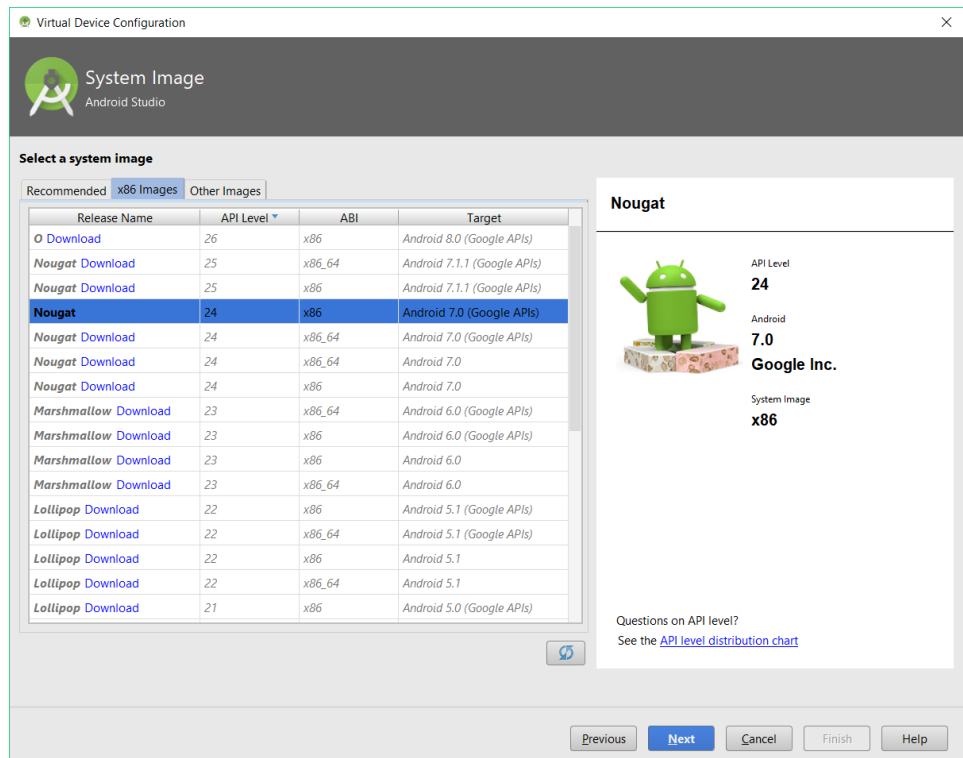


3. Jika ingin membuat spesifikasi hardware (perangkat keras) sendiri, bisa memilihnya pada pilihan **New Hardware Profile**. Akan muncul dialog seperti di bawah ini.



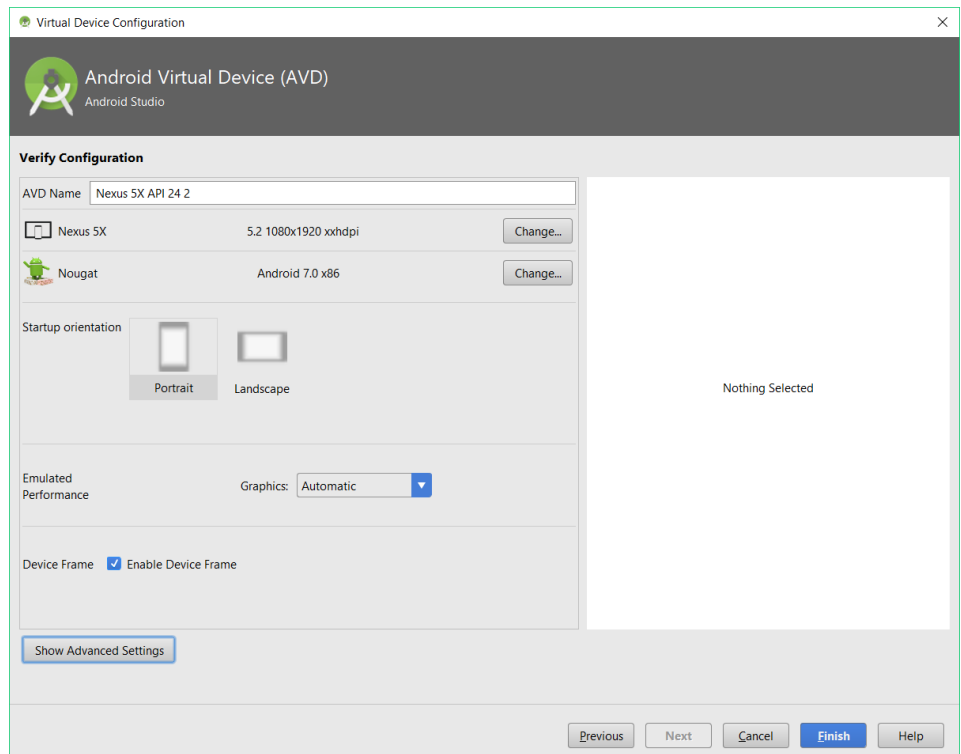
dapat menentukan konfigurasi hardware sesuai dengan kebutuhan . Yang perlu diingat adalah untuk menggunakan konfigurasi emulator yang sesuai dengan kemampuan laptop atau komputer yang digunakan.

4. dapat membuat hardware emulator baru atau memilih hardware emulator yang sudah ada. Setelah memilih hardware emulator, akan muncul dialog seperti ini.

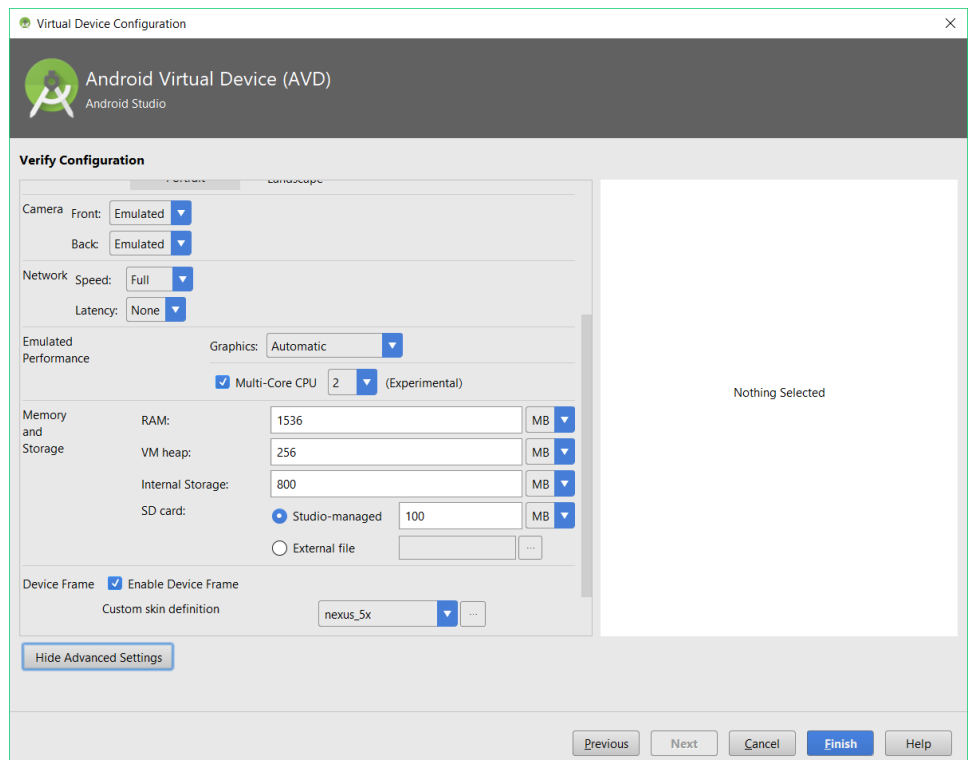


Pada dialog ini akan memilih versi android dari emulator yang akan buat. Pada dialog tersebut, perlu memilih versi yang sudah diunduh yaitu Nougat. Tombol download di sebelah kanan versi menunjukkan bahwa perlu mengunduhnya terlebih dahulu jika ingin menggunakannya.

5. Selanjutnya klik **Next**. Akan muncul dialog verify configuration. Pada dialog ini, bisa memeriksa konfigurasi dari emulator yang pilih.



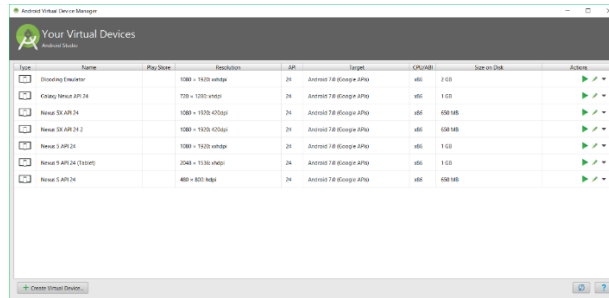
Pada bagian kiri bawah, terdapat tombol **Show Advanced Settings**. Bila menekan tombol ini, maka akan muncul tampilan dialog baru seperti gambar di bawah ini.





Pada bagian advanced setting, bisa mengubah konfigurasi hardware yang telah ditentukan sebelumnya.

6. Jika sudah selesai, dapat menekan tombol **Finish**. dapat membuka emulatornya dengan menekan tombol hijau yang ada di sebelah kanan.



7. Pengaturan emulator sudah selesai dan bisa langsung dijalankan.

## Run dengan device

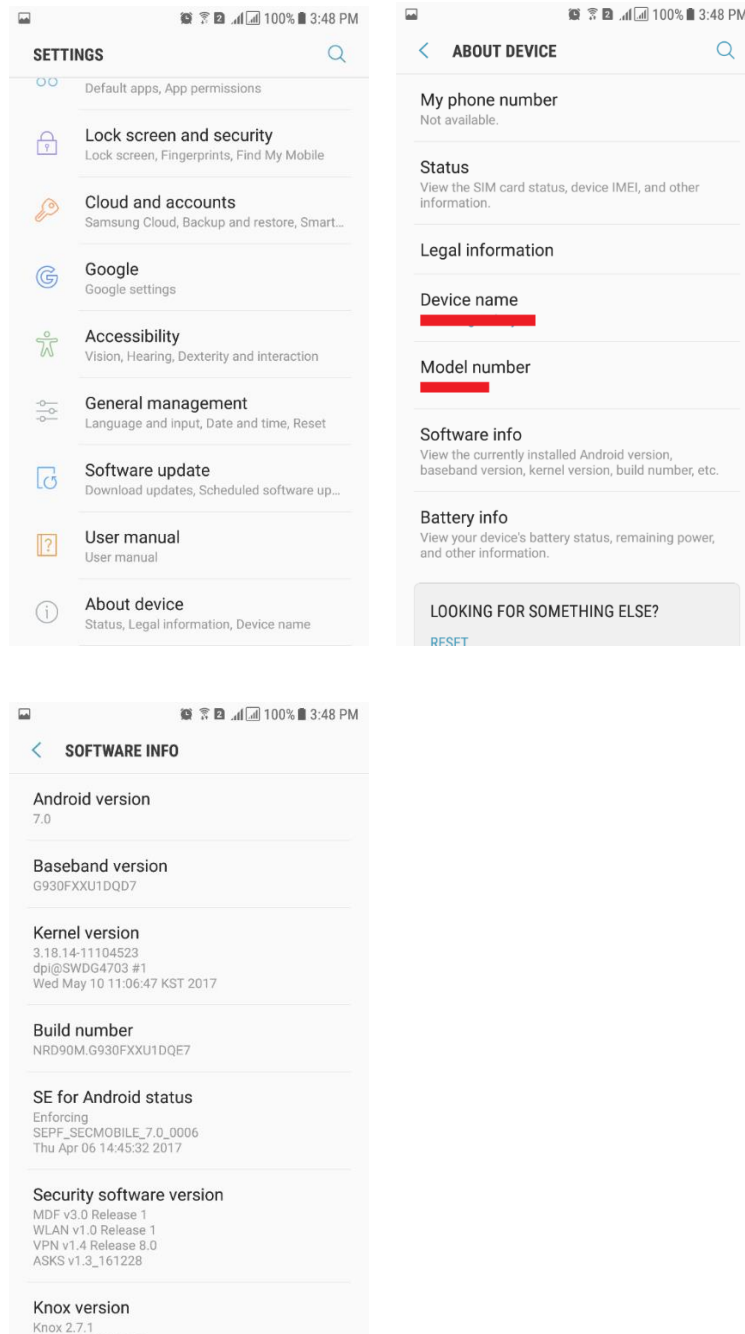
Bila hendak melakukan *run* atau *debugging*, lebih baik menjalankannya pada peranti *smartphone* asli. *Running* dengan menggunakan peranti memiliki beberapa kelebihan jika dibandingkan dengan emulator yaitu :

- Lebih cepat;
- Fitur seperti geo-location, push notif bisa digunakan;
- Bisa mengetahui daya serap baterai terhadap aplikasi.
- Lebih mudah

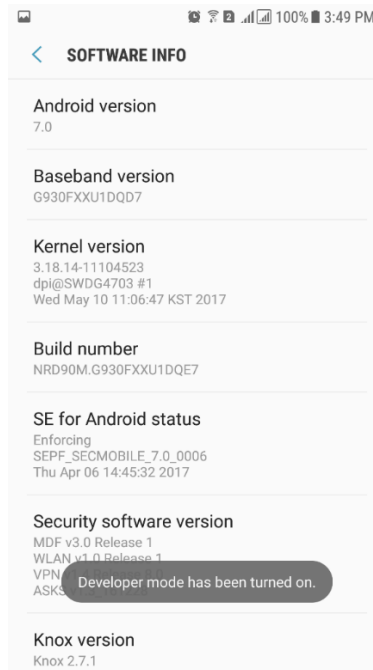
Dengan menggunakan peranti *smartphone* asli, kita dapat memastikan bahwa aplikasi kita berjalan dengan wajar ketika sudah sampai di tangan pengguna. Kendala dari pendekatan ini adalah beragamnya model peranti yang ada di pasaran. Namun, pembahasan mengenai hal tersebut berada diluar cakupan kelas ini.

Mari ikuti langkah-langkah untuk menjalankan proses *run* atau *debugging*. Tampilan dari langkah berikut bisa dipastikan akan berbeda dengan peranti yang pakai. Akan tetapi secara garis besar langkahnya akan sama.

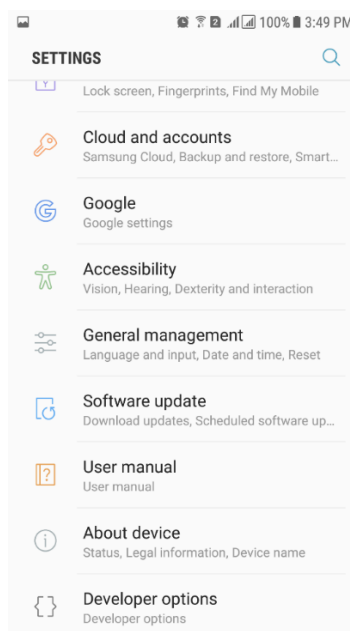
1. Pastikan peranti yang akan dipakai sesuai dengan target SDK atau paling tidak mendukung versi SDK terendah yang digunakan aplikasi.
2. Buka *setting* dan masuk ke dalam menu **About**. Pada halaman menu ini, perlu menemukan informasi tentang **Build number**.



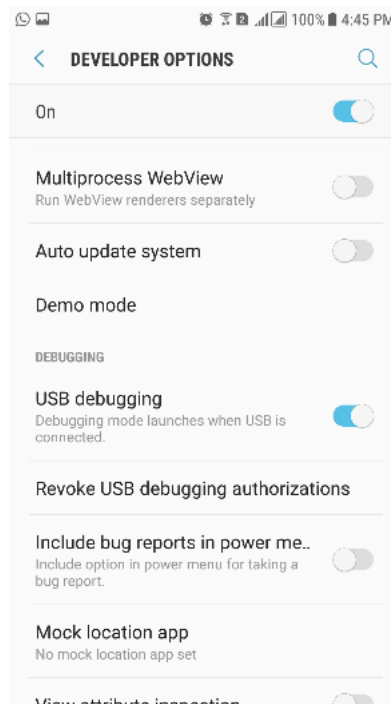
3. Kemudian tekan **Build number** sebanyak 7 kali, hingga muncul tulisan dalam bentuk toast seperti pada gambar dibawah ini.



4. Kembali ke menu setting di awal dan akan muncul menu baru di bawah about yaitu **Developer Options**.



5. Masuk ke dalam menu **Developer Options** dan pastikan opsi **USB Debugging Mode** sudah nyala.



Setelah menyelesaikan pengaturan pada peranti, maka peranti dapat dihubungkan dengan laptop atau komputer yang pakai.

**Catatan :** Beberapa vendor smartphone memiliki sistem operasi yang unik. Tampilan setting dan letak opsi bisa jadi tidak sama dengan gambar di atas.

Beberapa vendor juga mengharuskan untuk mengunduh *driver* khusus sebelum bisa menghubungkannya ke Android Studio. Kami sarankan untuk mengunjungi website / membaca petunjuk yang sesuai dengan vendor dari peranti .

## 5. Tugas

Import sample code kemudian running program dari sample code

**Praktikum ke : 3**

**Judul Praktikum : Dasar Pengenalan Aplikasi Mobile**

**Alokasi Waktu : 2 x 50 menit**

### **1. Tujuan Instruksional Khusus**

- a. Mahasiswa dapat memahami konsep dasar dari ekosistem android
- b. Mahasiswa dapat memahami Activity, Intent, Fragment, Threads dan Receiver

### **2. Teori**

Modul pertama yang akan pelajari adalah komponen-komponen dasar dari sebuah aplikasi Android. Beberapa komponen dasar dari Android seperti berikut :

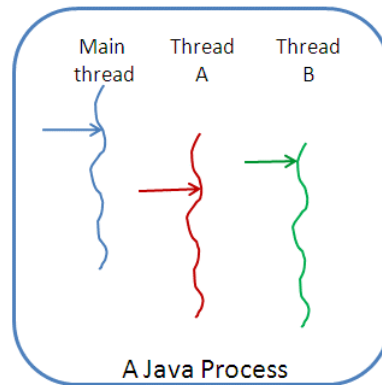
#### **1. Activity**

Merupakan satu komponen yang berhubungan dengan pengguna. Activity menangani window (tampilan) mana yang akan di tampilkan ke dalam interface (antarmuka).

Activity memiliki daur hidup (life cycle) tersendiri yang dimulai dari **onCreate** hingga **onDestroy**.



Bukan hanya main thread saja yang bisa kita gunakan, kita bisa membuat thread baru yang terpisah dari main thread agar tidak mengganggu proses rendering tampilan di layar. Beberapa komponen yang bisa digunakan adalah seperti handler dan async task.



### 5. Service

Service merupakan komponen tidak terlihat yang dapat digunakan untuk menjalankan suatu proses di dalam aplikasi. Service biasanya digunakan untuk menjalankan proses yang memakan waktu lama atau yang membutuhkan komputasi intensif. Contohnya adalah pemutar musik dan blocking operation untuk networking.

### 6. Receiver

Receiver menggunakan pola *publish-subscribe*. Ketika terjadi sebuah *event* dibangkitkan (di-*publish*), komponen lain yang telah mendaftar untuk mendengarkan *event* tersebut (*subscribed*) dapat menjalankan perintah-perintah tertentu.

## 3. Alat dan Bahan

- a. BKPM
- b. Komputer
- c. LCD
- d. Alat Tulis Kantor (ATK)

#### **4. Pelaksanaan Praktikum**

Mempelajari fungsi dari Activity, Intent, Fragment, Threads dan Receiver



**Praktikum ke : 4 dan 5**

**Judul Praktikum : Activity pada pemrograman android**

**Alokasi Waktu : 2 x 50 menit**

## **5. Tujuan Instruksional Khusus**

- a. Mahasiswa dapat memahami konsep dasar dari activity pada android
- b. Mahasiswa dapat mengimplementasikan activity pada pemrograman berbasis android

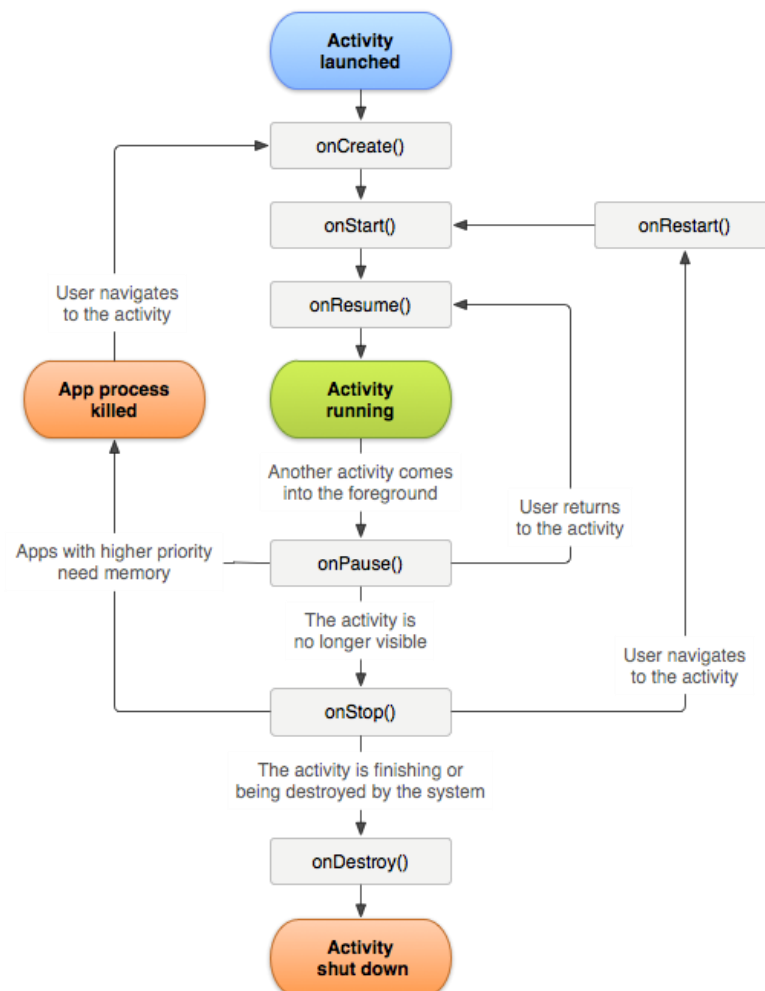
## **6. Teori**

Activity merupakan sebuah komponen di Android yang berfungsi untuk menampilkan user *interface* ke layar handset Android pengguna. Ini seperti pada saat melihat daftar percakapan pada aplikasi *chat* atau daftar *email* pada aplikasi Gmail di ponsel Android .

Umumnya dalam sebuah aplikasi terdapat lebih dari satu activity yang saling terhubung dengan tugas yang berbeda-beda. Activity merupakan salah satu komponen penting Android yang memiliki daur hidup (life cycle) dalam sebuah *stack* pada *virtual sandbox* yang disiapkan oleh *Dalvik Virtual Machine* (DVM) atau *Android Runtime* (ART) yang bersifat *last in first out*. Pada implementasinya, activity selalu memiliki satu layout *user interface* dalam bentuk berkas xml. Suatu aplikasi Android bisa memiliki lebih dari satu activity dan harus terdaftar di berkas **AndroidManifest.xml** sebagai sub aplikasi. Sebuah class Java dinyatakan sebuah activity jika mewarisi (*extends*) superclass **Activity** atau turunannya seperti **AppCompatActivity** dan **FragmentActivity**.

Untuk lebih mendalami activity, kami menyarankan untuk membaca referensi berikut :

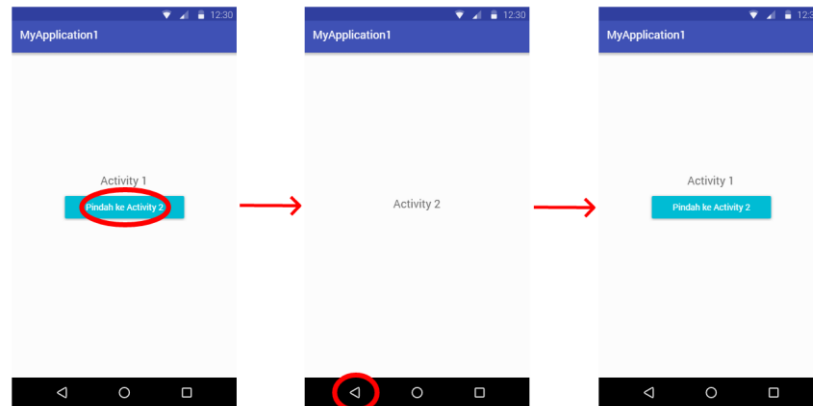
<https://developer.android.com/guide/components/activities.html>



Developer yang baik harus mengetahui secara detail tentang *life cycle* sebuah activity. Terutama untuk melakukan aksi yang tepat, saat terjadi perubahan *state* activity. *Callback methods* yang ada dapat digunakan untuk melakukan beragam proses terkait state dari activity. Misalnya melakukan semua inisialisasi komponen di `onCreate()`, melakukan *disconnect* terhadap koneksi ke *server* pada `onStop()` atau `onDestroy()` dan lain sebagainya.

Pemahaman yang baik tentang daur hidup activity akan membuat implementasi rancangan aplikasi menjadi lebih baik. Hal ini juga akan meminimalisir terjadinya *error/bug/force close* yang tidak diinginkan.

## Last In, First Out (LIFO)



Gambar 1	Gambar 2	Gambar 3
Aktif: Activity 1 <code>onCreate()</code> → <code>onStart()</code> → <code>onResume()</code>	Aktif: Activity 2 Stack append: Activity 2 [ <code>onResume()</code> ]	Activity 1 <code>onStop()</code> → <code>onRestart()</code> → <code>onStart()</code> → <code>onResume()</code>
Aksi: Klik Button1 (Pindah)	Aksi: Klik Hardware Back Button	Aktif: Activity 1
Stack append: Activity 1 [ <code>onStop()</code> ]	Activity 2 [ <code>finish()</code> ] Stack pop: Activity 2 [ <code>onDestroy()</code> ]	

### Penjelasan Gambar 1

Jika memiliki sebuah aplikasi yang terdiri dari 2 activity, maka activity pertama akan dijalankan setelah pengguna meluncurkan aplikasi melalui ikon aplikasi di layar device. Activity yang ada saat ini berada pada posisi activity running setelah melalui beberapa state `onCreate` (created) → `onStart` (started) → `onResume` (resumed) dan masuk ke dalam sebuah stack activity. Bila pada activity pertama menekan sebuah tombol untuk menjalankan activity kedua, maka posisi state dari activity pertama berada pada posisi stop. Saat itu, callback `onStop()` pada activity pertama akan dipanggil. Ini terjadi karena activity pertama sudah tidak berada pada layar foreground / tidak lagi ditampilkan. Semua informasi terakhir pada activity pertama akan disimpan secara otomatis. Sementara itu, activity kedua masuk ke dalam stack dan menjadi activity terakhir yang masuk.

### Penjelasan Gambar 2

Activity kedua sudah muncul di layar sekarang. Ketika menekan tombol back pada physical button menu utama atau menjalankan metode `finish()`, maka activity kedua akan dikeluarkan dari stack. Pada kondisi di atas, state activity kedua akan berada pada destroy. Oleh karenanya, metode `onDestroy()` akan dipanggil. Kejadian keluar dan masuk stack pada proses di atas menkan sebuah model Last In, First Out. Activity kedua menjadi yang terakhir masuk stack (Last In) dan yang paling pertama keluar dari stack (First Out).

### Penjelasan Gambar 3

Activity Pertama akan dimunculkan kembali di layar setelah melalui beberapa *state* dengan rangkaian *callback method* yang terpanggil, `onStop` → `onRestart` → `onStart` → `onResume`.

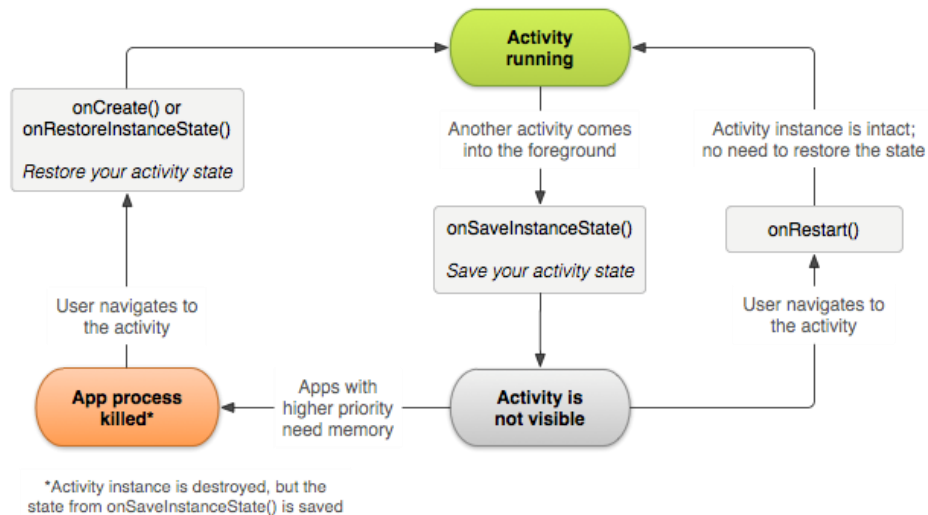
Detailnya dapat baca disini :

<https://developer.android.com/guide/components/activities/#Lifecycle>

### Saving Activity State

Ketika sebuah activity mengalami pause kemudian resume, maka state dari sebuah activity tersebut dapat terjaga. Sebabnya, obyek activity masih tersimpan di memory sehingga dapat dikembalikan state-nya. Dengan menjaga state dari activity, maka ketika activity tersebut ditampilkan, kondisinya akan tetap sama dengan kondisi sebelumnya. Akan tetapi ketika sistem menghancurkan activity untuk keperluan memori misalnya karena memori habis, maka obyek activity dihancurkan. Alhasil, ketika activity ingin ditampilkan kembali diperlukan proses re-create activity yang dihancurkan tadi. Kejadian di atas adalah hal yang lumrah terjadi. Oleh karena itu, perubahan yang terjadi pada activity perlu disimpan terlebih dahulu sebelum ia dihancurkan. Disinilah metode `onSaveInstanceState()` digunakan. Dalam `onSaveInstanceState` terdapat bundle yang dapat digunakan untuk menyimpan informasi. Informasi dapat disimpan dengan memanfaatkan fungsi seperti `putString()` dan `putInt()`. Ketika activity di-restart, bundle akan diberikan kepada metode `onCreate` dan `onRestoreInstanceState`. Bundle tersebut

akan dimanfaatkan untuk mengembalikan kembali perubahan yang telah terjadi sebelumnya.



Proses penghancuran activity dapat juga terjadi ketika terdapat perubahan konfigurasi seperti perubahan orientasi layar (portrait-landscape), keyboard availability, dan perubahan bahasa. Penghancuran ini akan menjalankan callback method onDestroy dan kemudian menjalankan onCreate. Penghancuran ini dimaksudkan agar activity dapat menyesuaikan diri dengan konfigurasi baru yang muncul pada kejadian-kejadian sebelumnya. Hal yang perlu diingat ketika menggunakan onSaveInstanceState adalah untuk tidak menyimpan data yang besar pada bundle. Contohnya, hindari penyimpanan data bitmap pada bundle. Bila data pada bundle berukuran besar, proses serialisasi dan deserialisasi akan memakan banyak memori.

## 7. Alat dan Bahan

- a. BKPM
- b. Komputer
- c. LCD
- d. Alat Tulis Kantor (ATK)

## 8. Pelaksanaan Praktikum

### Tujuan

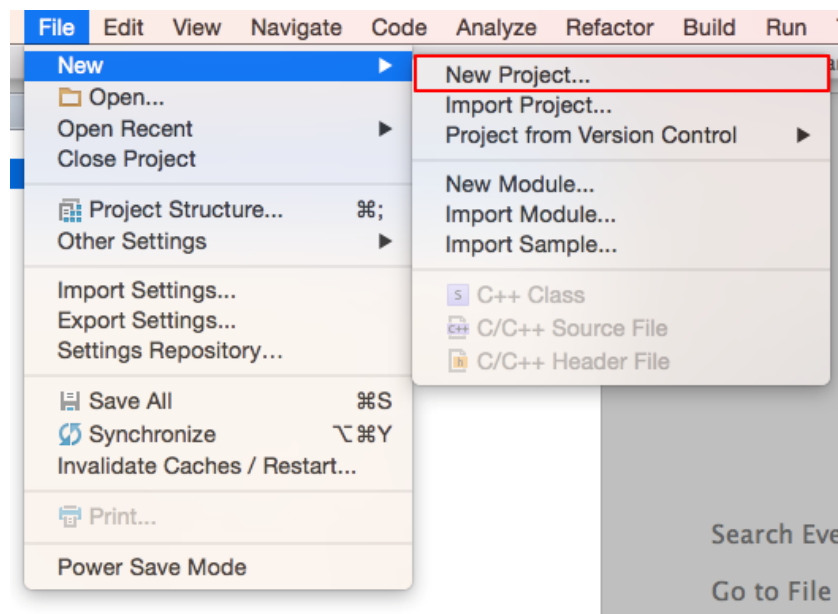
Codelab ini bertujuan untuk mengimplementasikan komponen activity pada aplikasi pertama yang bangun. Harapannya aktifitas ini dapat memberi gambaran yang jelas tentang cara kerja activity.

### Logika Dasar

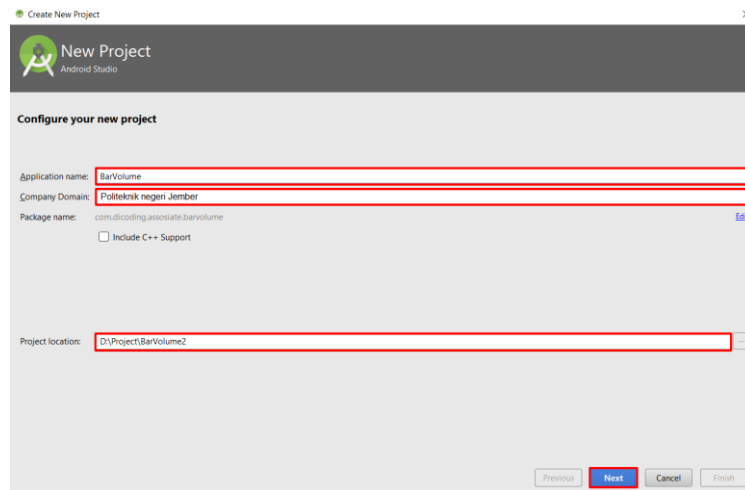
Melakukan input ke dalam obyek **TextBox** → melakukan validasi input → melakukan perhitungan volume balok ketika tombol hitung diklik.

Codelab Perhitungan Volume

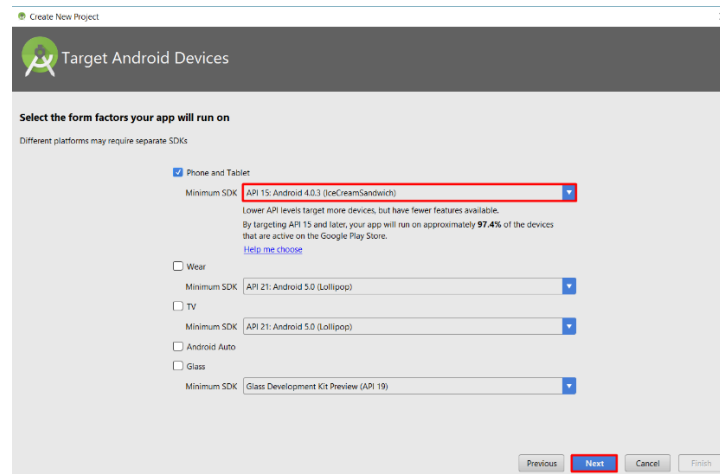
1. Buat proyek baru dengan klik **File → New → New Project** pada Android Studio .



2. Setelah muncul jendela **Create New Project**, atur nama aplikasi dan domain perusahaan / *website* . Sebaiknya jangan sama dengan apa yang ada dicontoh. Dan jangan lupa pula untuk menentukan lokasi proyek.



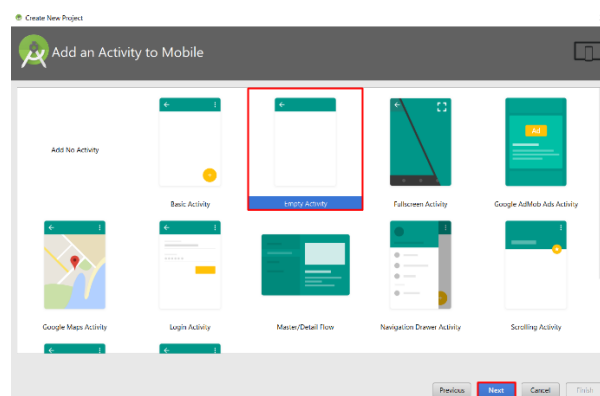
3. Kemudian pilih tipe peranti (device) untuk aplikasi beserta target minimum SDK yang akan digunakan. Pilihan target Android SDK akan mempengaruhi banyaknya peranti yang dapat menggunakan aplikasi. Di sini kita memilih tipe peranti Phone and Tablet dan nilai minimum SDK kita pasang ke Level 15 (Ice Cream Sandwich). Klik **Next** untuk melanjutkan.



4. Pada bagian ini kita akan memilih tipe activity awal dari *template* yang telah disediakan. Saat ini Android Studio sudah menyediakan berbagai macam *template* activity dari yang paling sederhana hingga yang paling kompleks seperti :

- **Add No Activity** Tidak ada activity yang ditambahkan

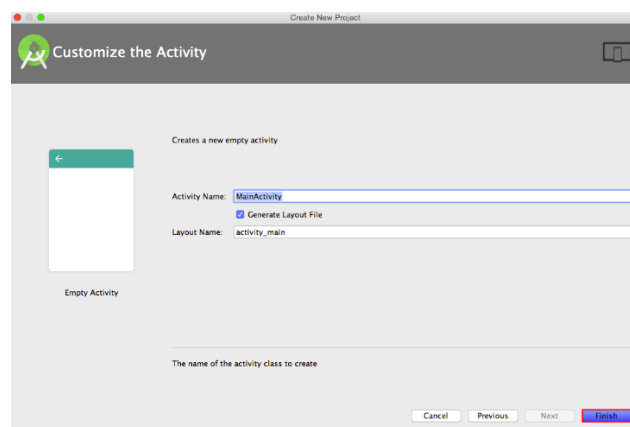
- **Basic Activity** Activity dengan template komponen material design seperti **FloatingActionButton**
- **Empty Activity** Activity dalam bentuk yang paling dasar
- **Fullscreen Activity** Activity *fullscreen* tanpa status bar
- **Google AdMob Ads Activity** Activity dengan konfigurasi *default* iklan Admob
- **Google Maps Activity** Activity dengan menyediakan konfigurasi dasar Google Maps
- **Login Activity** Activity untuk halaman login
- **Master / Detail Flow** Activity yang diperuntukan untuk alur aplikasi *master detail* pada peranti tablet
- **Navigation Drawer Activity** Activity dengan tampilan side bar menu
- **Scrolling Activity** Activity dengan kemampuan scroll konten didalamnya secara vertikal
- **Settings Activity** Activity yang diperuntukan untuk konfigurasi aplikasi
- **Tabbed Activity** Activity yang diperuntukan untuk menampilkan lebih dari satu tampilan, dapat digeser ke kanan dan ke kiri (*swipe*) dan dengan menggunakan komponen viewpager



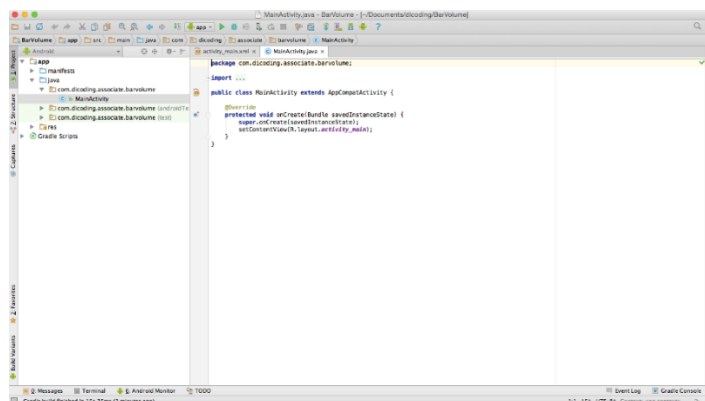


Saat ini kita pilih tipe **empty activity**, klik **Next** untuk melanjutkan.

5. Selanjutnya, tentukan nama activity pertama kita, saat ini kita biarkan pada kondisi apa adanya. Ingat, jika suatu saat nanti kita ingin melakukan penambahan activity, maka sebaiknya nama untuk activity tersebut disamakan dengan nama kelasnya dan diakhiri dengan activity. Misal: **ProfileActivity**, **SettingsActivity** dan lain sebagainya. Klik **Finish** untuk menyelesaikan.



6. Selamat! telah berhasil membuat sebuah proyek baru Android. Biasanya, tampilan layar akan seperti contoh di bawah ini:



7. Di sebelah kanan adalah *workspace* dimana Activity berada dan bernama **MainActivity.java** dengan layoutnya **activity\_main.xml**. Di sebelah kiri terdapat struktur proyek, di mana nanti kita akan banyak menambahkan berbagai komponen baru, *asset* dan *library*. Untuk lebih mengenal Android

Studio lebih dalam silakan baca materi  
ini <https://developer.android.com/studio/intro/index.html>

**Praktikum ke : 6 dan 7**

**Judul Praktikum : Intent pada pemrograman android**

**Alokasi Waktu : 2 x 50 menit**

### **1. Tujuan Instruksional Khusus**

- a. Mahasiswa dapat memahami konsep dasar dari intent pada android
- b. Mahasiswa dapat memahami explicit intent dan implicit intent
- c. Mahasiswa dapat mengimplementasikan intent pada pemrograman berbasis android

### **2. Teori**

Intent adalah mekanisme untuk melakukan sebuah action dan komunikasi antar komponen aplikasi misal activity, services, dan broadcast receiver. Ada tiga penggunaan umum intent dalam aplikasi Android yaitu: Memindahkan satu activity ke activity lain dengan atau tidak membawa data. Menjalankan background service, misalnya melakukan sinkronisasi ke server dan menjalankan proses berulang (periodic/scheduler task). Mengirimkan obyek broadcast ke aplikasi yang membutuhkan. Misal, ketika aplikasi membutuhkan proses menjalankan sebuah background service setiap kali aplikasi selesai melakukan booting. Aplikasi harus bisa menerima obyek broadcast yang dikirimkan oleh sistem Android untuk event booting tersebut.

Intent memiliki dua bentuk yaitu:

#### **Explicit Intent**

Adalah tipe Intent yang digunakan untuk menjalankan komponen dari dalam sebuah aplikasi. Explicit intent bekerja dengan menggunakan nama kelas yang dituju misal : `com.dicoding.activity.DetailActivity`. Umumnya intent ini digunakan untuk mengaktifkan komponen pada satu aplikasi.

## **Implicit Intent**

Adalah tipe intent yang tidak memerlukan detail nama kelas yang ingin diaktifkan. Model ini memungkinkan komponen dari aplikasi lain bisa merespon request intent yang dijalankan. Penggunaan tipe intent ini umumnya diperuntukan untuk menjalankan fitur/fungsi dari komponen aplikasi lain. Contohnya ketika kita membutuhkan fitur untuk mengambil foto. Daripada membuat sendiri fungsi kamera, lebih baik kita menyerahkan proses tersebut pada aplikasi kamera bawaan dari peranti atau aplikasi kamera lain yang telah terinstal sebelumnya di peranti. Hal yang sama misalnya ketika kita membutuhkan fungsi berbagi konten. Kita bisa memanfaatkan intent untuk menampilkan aplikasi mana saja yang bisa menangani fitur tersebut. Implementasi implicit intent ini akan sangat memudahkan bagi pengembang agar tetap fokus pada proses bisnis inti dari aplikasi yang dikembangkan.

Baca dan pahami materi pada link berikut ini:

<https://developer.android.com/guide/components/intents-filters.html>

### **3. Alat dan Bahan**

- a. BKPM
- b. Komputer
- c. LCD
- d. Alat Tulis Kantor (ATK)

### **4. Pelaksanaan Praktikum**

#### **Tujuan**

Codelab ini menitik beratkan pada implementasi intent untuk melakukan perpindahan dari activity ke activity lain, dengan atau tidak membawa data.

Beberapa bagian dari codelab ini akan menjawab beberapa pertanyaan umum dalam pengembangan aplikasi Android sebagai berikut:

- Bagaimana berpindah dari satu activity ke activity lain?
- Bagaimana berpindah dari satu activity ke activity lain dengan membawa data?
  - Single value dari suatu variabel.
  - Obyek model Plain Old Java Object (POJO).
- Menjalankan komponen di aplikasi lain untuk keperluan membuka *browser* atau melakukan pemanggilan melalui aplikasi telepon bawaan?
- Mengirimkan hasil nilai balik melalui Intent

### Logika dasar

Berpindah dari satu **Activity** ke **Activity** lain dengan membawa data. **Activity** asal akan mengirimkan data melalui Intent dan **Activity** tujuan akan menerima data yang dikirimkan.

Codelab Intent Sederhana

1. Buat Project baru di Android Studio dengan kriteria sebagai berikut:
  - Nama Project : **MyIntentApp**
  - Target & Minimum Target SDK : **Phone and Tablet, Api level 15**
  - Tipe Activity : **Empty Activity**
  - Activity Name : **MainActivity**
2. Selanjutnya kita akan membangun antarmuka (*interface*) seperti ini:



3. Kita akan memiliki 5 tombol dengan fungsi yang berbeda-beda dan 1 buah `TextView` untuk menampilkan data yang berasal dari intent. Baik, kita akan mulai selangkah demi selangkah dimulai dari tombol yang paling atas. Kondisikan `activity_main.xml` menjadi seperti ini:

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:tools="http://schemas.android.com/tools"
4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent"
6.     android:paddingBottom="@dimen/activity_vertical_margin"
7.     android:paddingLeft="@dimen/activity_horizontal_margin"
8.     android:paddingRight="@dimen/activity_horizontal_margin"
9.     android:paddingTop="@dimen/activity_vertical_margin"
10.    android:orientation="vertical"
11.    tools:context="com.dicoding.associate.myintentapp.MainActivity">
12.    <Button
13.        android:id="@+id/btn_move_activity"
14.        android:layout_width="match_parent"
15.        android:layout_height="wrap_content"
16.        android:text="Pindah Activity"
17.        android:layout_marginBottom="@dimen/activity_vertical_margin"/>
18. </LinearLayout>

```

Jangan lupa untuk menambahkan file `dimens.xml` secara manual di dalam `res` → `values`. Dan isikan file `dimens.xml` seperti berikut :

```

1. <resources>
2.     <!-- Default screen margins, per the Android Design guidelines. -->
3.     <dimen name="activity_horizontal_margin">16dp</dimen>
4.     <dimen name="activity_vertical_margin">16dp</dimen>
5. </resources>

```

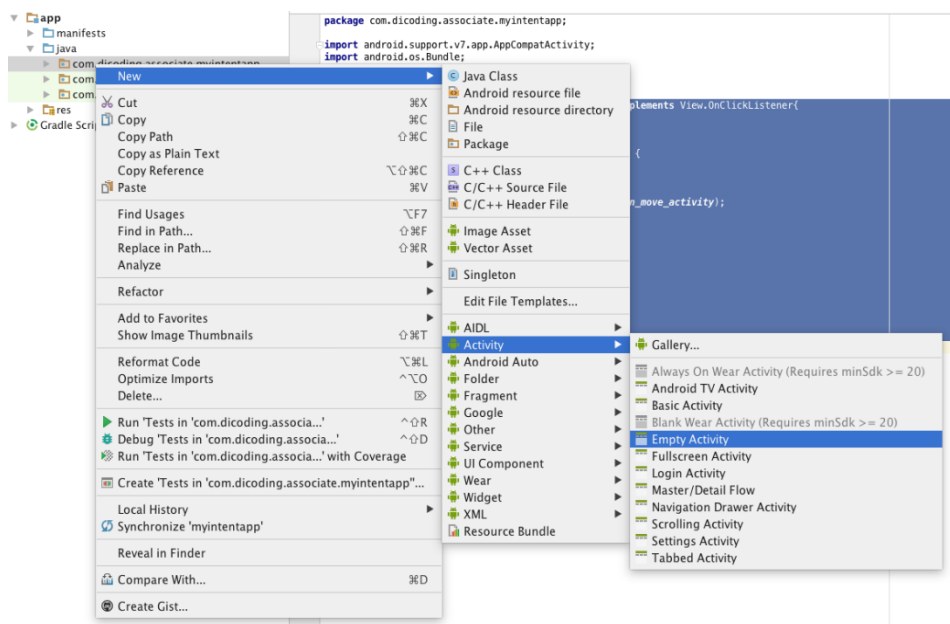
4. Lalu untuk `MainActivity.java` tambahkan beberapa baris seperti ini.

```

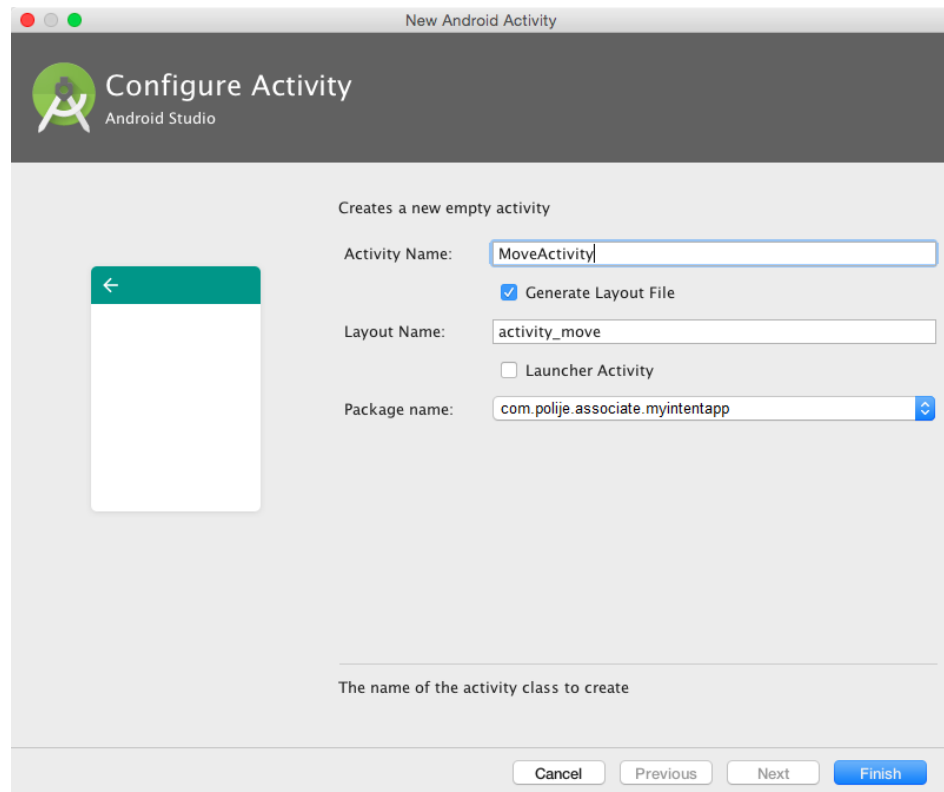
1. public class MainActivity extends AppCompatActivity implements View.OnClickListener{
2.     private Button btnMoveActivity;
3.     @Override
4.     protected void onCreate(Bundle savedInstanceState) {
5.         super.onCreate(savedInstanceState);
6.         setContentView(R.layout.activity_main);
7.
8.         btnMoveActivity = (Button)findViewById(R.id.btn_move_activity);
9.         btnMoveActivity.setOnClickListener(this);
10.    }
11.    @Override
12.    public void onClick(View v) {
13.        switch (v.getId()){
14.            case R.id.btn_move_activity:
15.                break;
16.        }
17.    }
18. }

```

5. **Button btnMoveActivity** akan memiliki fungsi untuk berpindah activity ke activity lain. Sekarang kita buat Activity baru dengan cara sebagai berikut: **Klik kanan** di *package* utama aplikasi **package name** → **New** → **Activity** → **Empty Activity**.



Lalu isikan **MoveActivity** pada dialog. Ketika sudah klik **Finish**.



6. Untuk menandakan bahwa perpindahan activity berhasil, silakan tambahkan satu **TextView** dan kondisikan **activity\_move.xml** menjadi seperti berikut.

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:tools="http://schemas.android.com/tools"
4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent"
6.     android:paddingBottom="@dimen/activity_vertical_margin"
7.     android:paddingLeft="@dimen/activity_horizontal_margin"
8.     android:paddingRight="@dimen/activity_horizontal_margin"
9.     android:paddingTop="@dimen/activity_vertical_margin"
10.    tools:context="com.dicoding.associate.myintentapp.MoveActivity">
11.    <TextView
12.        android:layout_width="match_parent"
13.        android:layout_height="wrap_content"
14.        android:text="Ini MoveActivity"/>
15. </RelativeLayout>

```

7. Setelah activity tujuan sudah berhasil diciptakan, sekarang saatnya menambahkan suatu intent pada method **onClick()** di **MainActivity.java** menjadi sebagai berikut.



```

1. public class MainActivity extends AppCompatActivity implements View.OnClickListener{
2.     private Button btnMoveActivity;
3.     @Override
4.     protected void onCreate(Bundle savedInstanceState) {
5.         super.onCreate(savedInstanceState);
6.         setContentView(R.layout.activity_main);
7.         btnMoveActivity = (Button)findViewById(R.id.btn_move_activity);
8.         btnMoveActivity.setOnClickListener(this);
9.     }
10.    @Override
11.    public void onClick(View v) {
12.        switch (v.getId()){
13.            case R.id.btn_move_activity:
14.                Intent moveIntent = new Intent(MainActivity.this, MoveActivity.class);
15.                startActivity(moveIntent);
16.                break;
17.        }
18.    }
19. }

```

Selesai! Langkah pertama untuk memindahkan satu activity ke activity lain sudah selesai, sekarang silakan jalankan aplikasi Anda dengan mengklik tombol pada menu bar. Seharusnya sekarang anda sudah bisa memindahkan activity dengan mengklik tombol ‘Pindah Activity’.

**Praktikum ke : 9 dan 10**

**Judul Praktikum : Fragment pada pemrograman android**

**Alokasi Waktu : 2 x 50 menit**

### **1. Tujuan Instruksional Khusus**

- a. Mahasiswa dapat memahami konsep dasar dari fragment pada android
- b. Mahasiswa dapat memahami fragment life cycle
- c. Mahasiswa dapat mengimplementasikan fragment pada pemrograman berbasis android

### **2. Teori**

Fragment merupakan komponen yang memiliki fungsi untuk menampilkan antarmuka ke pengguna melalui activity dengan memiliki layout xml sendiri. Fragment memiliki daur hidup sendiri dan bergantung penuh pada daur hidup activity dimana ia ditanamkan. Fragment digunakan agar komponen tampilan aplikasi menjadi fleksibel dan dapat digunakan kembali (*reusable*). Satu activity bisa memiliki lebih dari satu fragment. Tidak seperti activity, fragment tidak perlu didaftarkan ke dalam file `AndroidManifest.xml`. Satu kelas Java dinyatakan sebagai sebuah fragment ketika kelas tersebut meng-*extends* (*inherit*) kelas `Fragment`. Melalui Android Support Library, fragment bersifat kompatibel sampai Android api level 10 Gingerbread. Analogi yang mendekati fragment pada platform lain adalah penggunaan komponen *iframe* pada aplikasi berbasis *web*.

#### **Fragment Life Cycle**

Ada beberapa state yang perlu kita ketahui sebelum menggunakan fragment.

- **Resumed**

Fragment bisa dilihat ketika activity sedang berjalan.

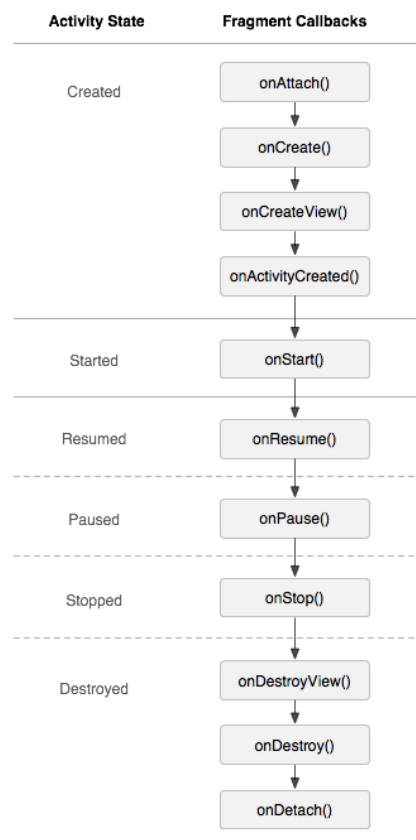
- **Paused**

Ketika ada activity lain yang menutupi sebagian dari activity dimana fragment ditambahkan. Yang dimaksud menutupi sebagian adalah ketika activity-nya tidak tertutup sepenuhnya oleh activity lain. Jadi masih ada bagian dari activity yang masih bisa dilihat di layar.

- **Stopped**

Ketika fragment tidak kelihatan di layar. Bisa jadi karena activity dimana fragment itu ditambahkan berhenti atau bahkan fragment itu sendiri sudah dihapus dari activity. Pada kondisi ini fragment masih hidup dengan semua informasinya. Akan tetapi sudah tidak kelihatan di layar dan akan dihancurkan.

Skema di bawah ini menunjukkan *callback method* apa saja yang akan dipanggil di dalam fragment ketika terjadi perubahan pada sebuah activity.



Skema di atas menunjukkan bahwa perubahan *state* dari sebuah activity akan mempengaruhi *life cycle* dari sebuah *fragment*. Ini karena fragment merupakan komponen view yang bisa ditambahkan (*embed*) ke dalam activity.

Untuk tahu lebih detail tentang fragment silakan kamu pahami baik-baik materi di link ini:

<https://developer.android.com/guide/components/fragments>

## 9. Alat dan Bahan

- e. BKPM
- f. Komputer
- g. LCD
- h. Alat Tulis Kantor (ATK)

## 10. Pelaksanaan Praktikum

### Tujuan

Agar dapat lebih memahami topik fragment, akan ada beberapa codelab yang akan di lakukan.

1. Membuat tampilan fleksibel dengan fragment.
2. Membuat tampilan yang dapat digeser-geser.
3. Membuat tampilan yang dapat disesuaikan dengan perubahan orientasi device.

### Codelab

Buat project dengan kriteria sebagai berikut :

Project Name : **MyFlexibleFragment**

Minimum Level Api : **15**

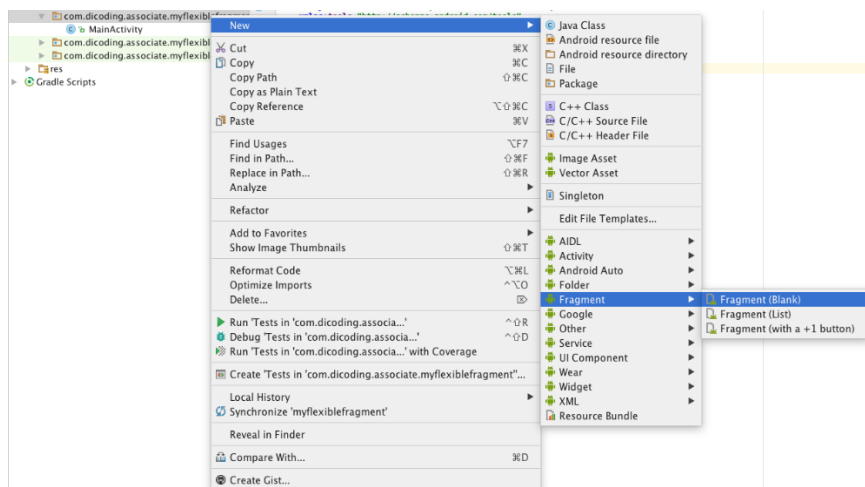
Default Activity : **Empty Activity**

Activity Name : **MainActivity**

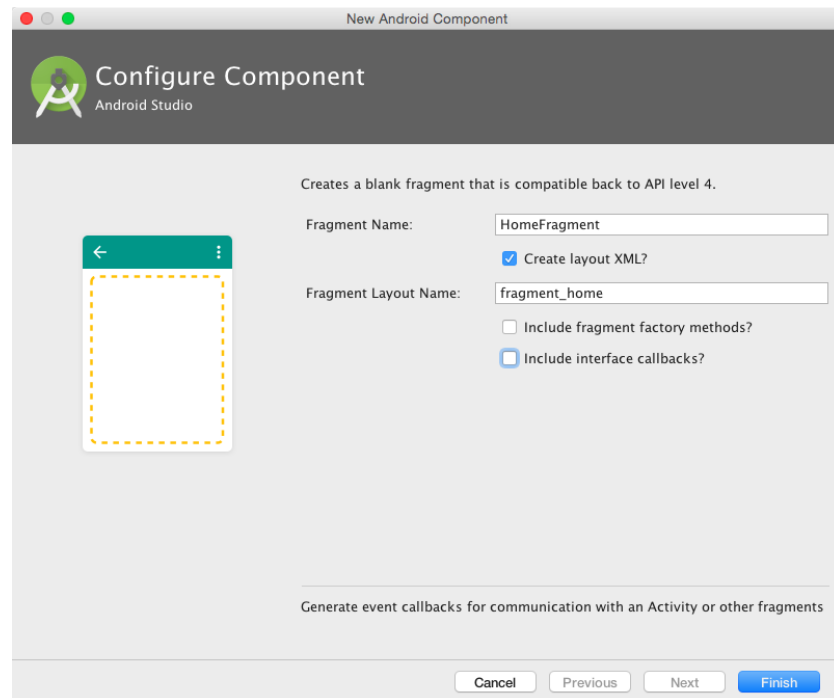
1. Pada `activity_main.xml`, silakan kondisikan kode pada berkas tersebut menjadi seperti berikut :

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:tools="http://schemas.android.com/tools"
4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent"
6.     tools:context="com.dicoding.associate.myflexiblefragment.MainActivity"
7.     android:id="@+id/frame_container">
8. </FrameLayout>
```

2. Kemudian, kita buat beberapa fragment untuk mengimplementasikan perpindahan tampilan tanpa perpindahan activity. Pertama kita buat **Fragment** dengan nama **HomeFragment**. Caranya : klik kanan pada package utama pada project aplikasi Anda → **New** → **Fragment** → **Fragment (Blank)**.



3. Setelah tampil dialog untuk fragment, isikan **HomeFragment** pada kolom **Fragment Name** dan uncheck untuk kedua pilihan (**Include fragment factory methods** dan **include interface callbacks**) seperti gambar dibawah ini. Klik **Finish** untuk melanjutkan penciptaan fragment.



4. Setelah **HomeFragment** tercipta, pada **fragment\_home.xml** kita sesuaikan tampilannya dengan menambahkan sebuah obyek **TextView** dan sebuah obyek **Button** seperti berikut :

```

1. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2.   xmlns:tools="http://schemas.android.com/tools"
3.   android:layout_width="match_parent"
4.   android:layout_height="match_parent"
5.   android:orientation="vertical"
6.   tools:context="layout.HomeFragment">
7.   <TextView
8.       android:layout_width="match_parent"
9.       android:layout_height="wrap_content"
10.      android:text="Hello Ini Home Fragment"
11.      android:layout_marginTop="@dimen/activity_vertical_margin"
12.      android:layout_marginBottom="@dimen/activity_vertical_margin"
13.      android:layout_marginLeft="@dimen/activity_horizontal_margin"
14.      android:layout_marginRight="@dimen/activity_horizontal_margin"/>
15.   <Button
16.       android:id="@+id/btn_category"
17.       android:layout_width="match_parent"
18.       android:layout_height="wrap_content"
19.       android:layout_margin="@dimen/activity_horizontal_margin"
20.       android:text="Ke Halaman Category"/>
21. </LinearLayout>

```

Akan terjadi error pada **tools:context**. Tenang, yang perlu Anda lakukan adalah menyesuaikannya dengan nama activity dan ditambahkan dengan nama *package* dari project Anda.

5. Pada **HomeFragment.java** lakukan penyesuaian kode sebagai berikut:

```

1. public class HomeFragment extends Fragment implements View.OnClickListener{
2.     public HomeFragment() {
3.         // Required empty public constructor
4.     }
5.     @Override
6.     public View onCreateView(LayoutInflater inflater, ViewGroup container,
7.         Bundle savedInstanceState) {
8.         // Inflate the layout for this fragment
9.         View view = inflater.inflate(R.layout.fragment_home, container, false);
10.        Button btnCategory = (Button)view.findViewById(R.id.btn_category);
11.        btnCategory.setOnClickListener(this);
12.        return view;
13.    }
14.    @Override
15.    public void onClick(View v) {
16.        if (v.getId() == R.id.btn_category){
17.            //todo to CategoryFragment
18.        }
19.    }
20. }

```

6. Selanjutnya, pada **MainActivity.java**, kita tanamkan **HomeFragment** kedalam activity tersebut sehingga bisa tampil ke layar pengguna dengan menambahkan beberapa baris berikut:

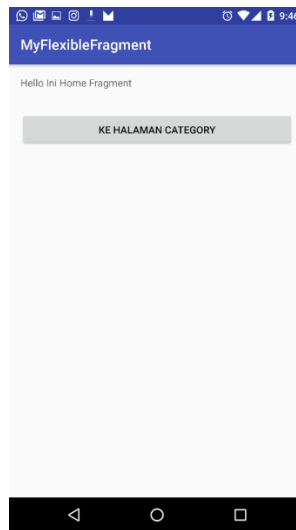
```

1. public class MainActivity extends AppCompatActivity {
2.     @Override
3.     protected void onCreate(Bundle savedInstanceState) {
4.         super.onCreate(savedInstanceState);
5.         setContentView(R.layout.activity_main);
6.
7.         FragmentManager fragmentManager = getSupportFragmentManager();
8.         FragmentTransaction mFragmentTransaction = fragmentManager.beginTransaction();
9.         HomeFragment mHomeFragment = new HomeFragment();
10.        mFragmentTransaction.add(R.id.frame_container, mHomeFragment, HomeFragment.class.getSimpleName());
11.
12.        Log.d("MyFlexibleFragment", "Fragment Name :"+HomeFragment.class.getSimpleName());
13.        mFragmentTransaction.commit();
14.    }
15. }

```

Untuk mengatasi tanda merah pada class, jangan lupa untuk tekan (**Alt + Enter**) untuk *import* kelas secara otomatis.

7. Sekarang, setelah selesai semua, silakan jalankan aplikasi Anda dan hasilnya haruslah seperti ini.



Ketika aplikasi dijalankan, aplikasi akan menampilkan satu text dan satu button yang dimana kedua komponen tersebut dimiliki oleh **HomeFragment**.



**Praktikum ke : 11 dan 12**

**Judul Praktikum : Thread, Handler dan Asynctask**

**Alokasi Waktu : 2 x 50 menit**

### **1. Tujuan Instruksional Khusus**

- a. Mahasiswa dapat memahami konsep dasar dari Thread, Handler dan Asynctask pada android
- b. Mahasiswa dapat mengimplementasikan Thread, Handler dan Asynctask pada pemrograman berbasis android

### **2. Teori**

Aplikasi dijalankan menggunakan proses dalam linux. Ketika sebuah komponen aplikasi dijalankan, maka sistem android akan menjalankan sebuah proses linux baru pada aplikasi dan dengan menggunakan suatu *thread* tunggal.

Secara umum semua komponen aplikasi di Android berjalan pada proses dan *thread* yang sama. Ini disebut *main thread* atau *ui thread*. Secara *default* ini berarti sistem Android tidak secara otomatis menciptakan *thread* lain untuk menjalankan proses secara spesifik.

**Ingat! activity, fragment, service, ui toolkit, broadcast receiver dijalankan di main thread atau ui thread**

Dalam pengembangan aplikasi, terkadang kita membutuhkan proses komputasi yang intensif. Misalnya proses memanipulasi *bitmap* dan proses menghubungi *server*.

Jika proses-proses tersebut dilakukan pada ui thread atau main thread, maka proses tersebut akan menghambat *rendering* tampilan aplikasi. Hal ini ditandai dengan komponen ui widget tidak bisa diklik. Mirip dengan kondisi *hang*.

Sistem Android akan menghitung selama 5 detik apakah aplikasi yang sedang berjalan itu responsif atau tidak. Jika tidak, maka sistem Android akan

menampilkan dialog Application Not Responding (ANR) Jelas ini akan memberi dampak yang negatif ke pengguna. Kesalahan ini dapat berujung ke pengguna mencopot aplikasi kita (*uninstall*).

Oleh karena itu kita membutuhkan *thread* lain. *Thread* lain ini bernama *worker thread* atau *async task*. Ia akan berjalan pada *thread* yang terpisah dengan *main thread*. Alhasil, aplikasi tetap terasa responsif.

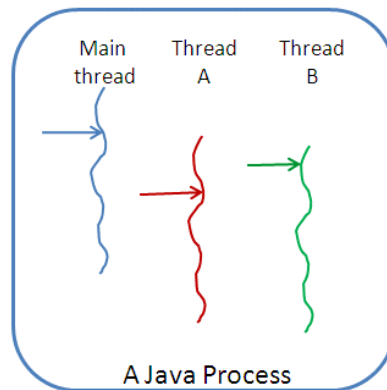
Ada dua aturan yang harus kita perhatikan agar tercipta pengalaman pengguna yang baik ketika menerapkan proses komputasi intensif yang memakan waktu.

- Jangan memblok *ui thread* atau *main thread*. Ini berarti kita harus menciptakan *worker threads* atau *async task*. Ini akan menjaga aplikasi tetap responsif.
- Jangan melakukan pemanggilan komponen ui widget (seperti *textview*, *button*, *imageview* dsb) didalam *worker thread* atau *thread* yang sedang berjalan secara *asynchronous*. Ini karena Android UI Toolkit merupakan komponen yang hanya berjalan pada *ui thread*.

Oiya, bagi yang belum tahu apa itu *thread* berikut penjelasan singkatnya.

*Thread* adalah sekumpulan perintah (instruksi) yang dapat dilaksanakan (dieksekusi) secara beriringan dengan *thread*lainnya. Hal ini dicapai dengan menggunakan mekanisme *time slice* (ketika satu CPU melakukan perpindahan antara satu *thread* ke *thread* lainnya) atau *multiprocess* (ketika *thread* tersebut dijalankan oleh CPU yang berbeda dalam satu sistem).

Gambarannya sebagai berikut:



Thread A dan Thread B masih tetap berjalan beriringan dengan *main thread*.

### Worker Thread

Dengan menggunakan *worker thread*, kita dapat menciptakan *thread* baru dan menjalankan proses di *thread* tersebut. Terciptanya *thread* baru dapat memenuhi aturan no 1 tentang pengalaman pengguna yang baik, yaitu dengan tidak memblokir *ui thread*.

Contoh bagaimana menggunakan *worker threads* yaitu.

```
1. public void onClick(View v) {  
2.     new Thread(new Runnable() {  
3.         public void run() {  
4.             final String txt = loadStringFromNetwork("http://example.com/string.json");  
5.             textView.setText(txt);  
6.         }  
7.     }).start();  
8. }
```

Sayangnya, contoh *worker thread* di atas tidak memenuhi aturan no 2, yaitu jangan melakukan pemanggilan ui di luar *ui thread*. Perhatikan metode `setText`. Metode ini adalah bagian dari `textView`. Sementara itu `textView` adalah salah satu komponen ui. Sehingga, ia tidak diperbolehkan untuk dijalankan di *worker thread*.

Agar kode di atas memenuhi aturan no 2, kita dapat menggunakan metode post milik ui atau view. Contohnya:

```

1. public void onClick(View v) {
2.     new Thread(new Runnable() {
3.         public void run() {
4.             final String txt = loadStringFromNetwork("http://example.com/string.json");
5.             textView.post(new Runnable() {
6.
7.                 public void run() {
8.                     textView.setText(txt);
9.                 }
10.            })
11.        }
12.    }).start();
13. }
14. }

```

Kode `run()` di dalam `new Thread (new runnable)` berjalan secara *asynchronous*, sedangkan kode `run()` di dalam `textView.post()` berjalan di *ui thread*.

Ada beberapa fungsi yang dapat dimanfaatkan untuk memenuhi aturan 2 yaitu:

1. `Activity.runOnUiThread(Runnable)`
2. `View.post(Runnable)`
3. `View.postDelayed(Runnable , long)`

## Handler

Beberapa fungsi seperti `view.post` dapat menyelesaikan masalah tentang *worker thread*. Akan tetapi menggunakan fungsi `view.post` akan menjadikan kode lebih kompleks dan sulit untuk dikelola.

Solusinya adalah dengan menggunakan handler yang dapat mengirim dan memproses *message* dan objek *runnable* lainnya yang berhubungan dengan *thread*. Ketika handler diciptakan, maka dia terkait dengan *thread* dimana diciptakan.

Berikut adalah contoh perubahan yang dapat kita lakukan:

```

1. private Handler handler = new Handler(){
2.     public void handleMessage (Message msg){
3.         String message = (String) msg.obj;
4.         textView.setText(message);
5.     }
6. }
7.
8. public void onClick(View v) {
9.     new Thread(new Runnable() {
10.         public void run() {
11.
12.             final String txt = loadStringFromNetwork("http://example.com/string.json");
13.             Message msg = Message.obtain();
14.             msg.obj = txt;
15.             msg.setTarget(handler);
16.             msg.sendToTarget();
17.         }
18.     }).start();
19. }

```

Dengan mendeklarasikan obyek handler pada *ui thread* maka obyek handlers diikat ke *ui thread*. Ketika suatu *message* dikirim ke handler, maka handler akan memprosesnya. Proses dari *message* tersebut terjadi pada *ui thread* sehingga aturan no 2 dapat terpenuhi.

## AsyncTask

AsyncTask adalah komponen pada Android yang memiliki kemampuan untuk menjalankan proses secara *asynchronous*. Ia tetap bisa berkomunikasi dengan *ui thread* untuk mengirimkan hasil proses yang dilakukannya, tanpa melanggar dua aturan utama yang dijelaskan sebelumnya. Selain itu, AsyncTask sangat mudah dan sederhana untuk diterapkan.

Poin-poin penting tentang AsyncTask:

- Diperuntukan untuk proses *asynchronous* dan mampu berkomunikasi dengan *ui thread* untuk mengirimkan hasil proses.
- Kelas Java yang dibuat harus inherit kepada **AsyncTask**.
- Kasus umum yang biasa diterapkan adalah ketika mengunduh berkas dari Internet dan dengan menampilkan perkembangan pengunduhan di layar.

```

1. private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
2.     protected Long doInBackground(URL... urls) {
3.         int count = urls.length;
4.         long totalSize = 0;
5.         for (int i = 0; i < count; i++) {
6.             totalSize += Downloader.downloadFile(urls[i]);
7.             publishProgress((int) ((i / (float) count) * 100));
8.             // Escape early if cancel() is called
9.             if (isCancelled()) break;
10.        }
11.        return totalSize;
12.    }
13.    protected void onProgressUpdate(Integer... progress) {
14.        setProgressPercent(progress[0]);
15.    }
16.    protected void onPostExecute(Long result) {
17.        showDialog("Downloaded " + result + " bytes");
18.    }
19. }

```

## Params, Progress, dan Result

Kelas `AsyncTask` menggunakan 3 tipe generik.

```
1. private class DownloadFilesTask extends AsyncTask<URL, Integer, Long>
```

Yang perlu diperhatikan adalah kode `<URL, Integer, Long>`.

3 *argument* di atas menunjukkan tipe data apa yang digunakan di dalam `asyncTask`.

Argument pertama adalah **params**, kedua adalah **progress**, dan ketiga adalah **result**.

Penjelasan 3 argument di atas adalah seperti berikut:

- **Params** : Tipe parameter yang akan menjadi inputan untuk proses *asynchronous*.
- **Progress** : Tipe satuan unit untuk memberi kabar perkembangan ke *ui thread*.
- **Result** : Tipe hasil dari proses *asynchronous* yang dijalankan.

## Beberapa Metode Utama di dalam AsyncTask

Kelas `asyncTask` memiliki 4 metode utama, yaitu :

### 1. `onPreExecute()`

Metode ini akan dijalankan pertama kali sebelum proses *asynchronous* dilakukan. Metode ini masih berada pada *ui thread*. Pada umumnya, ia digunakan untuk menampilkan komponen ui seperti progress bar.

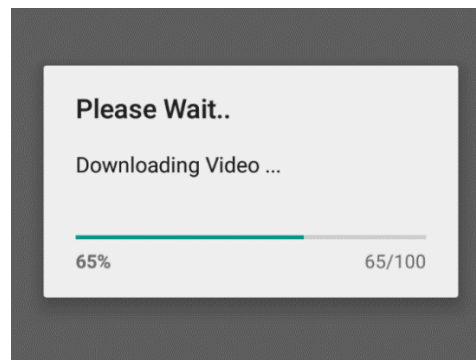
## 2. **doInBackground()**

Metode ini akan dijalankan setelah **onPreExecute()**. Disinilah proses *asynchronous* terjadi. Pada kasus diatas, metode **doInBackground()** akan melakukan pengunduhan berkas dari network melalui metode **downloadFile()** dan hasil perkembangannya dikirim melalui metode **publishProgress()**.

```
1. protected Long doInBackground(URL... urls) {  
2.     int count = urls.length;  
3.     long totalSize = 0;  
4.     for (int i = 0; i < count; i++) {  
5.         totalSize += Downloader.downloadFile(urls[i]);  
6.         publishProgress((int) ((i / (float) count) * 100));  
7.         // Escape early if cancel() is called  
8.         if (isCancelled()) break;  
9.     }  
10.    return totalSize;  
11. }
```

## 3. **onProgressUpdate()**

Metode ini yang akan menerima input dari apa yang dilakukan oleh metode **publishProgress**. Proses umum yang terjadi adalah memperbarui persentasi dari tampilan progress bar seperti ini.



## 4. **onPostExecute()**

Setelah proses di **doInBackground()** selesai, maka hasilnya akan dikirimkan ke metode **onPostExecute()**. Kemudian prosesnya akan dikembalikan lagi ke *ui thread*. Disinilah proses penampilan atau proses lain yang menunjukkan bahwa **AsyncTask** telah selesai dijalankan. Dalam contoh di atas, kita bisa menampilkan berkas yang baru saja diunduh.

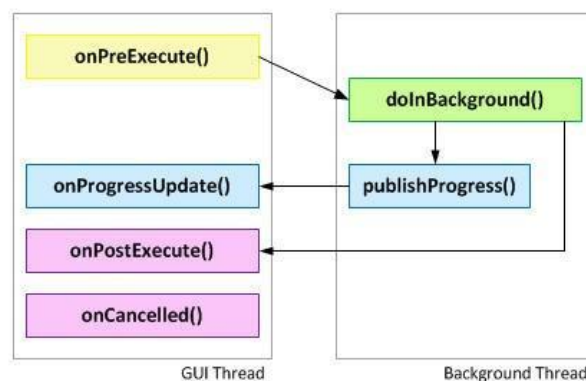
## Running AsyncTask

Untuk menjalankan AsyncTask, caranya adalah dengan menuliskan kode baris :

```
1. new DownloadFilesTask().execute(ur11,ur12,ur13);
```

Baris diatas akan membuat asynctask berjalan secara berurutan berdasarkan proses dan tanggung jawab pada setiap metode didalamnya.

Berikut proses kerja yang terjadi di asynctask.



Untuk lebih mendalami topik ini, Anda dapat membaca tautan berikut:

- <https://developer.android.com/guide/components/processes-and-threads>
- <https://developer.android.com/reference/android/os/AsyncTask>
- <https://android-developers.googleblog.com/2009/05/painless-threading.html>

## 3. Alat dan Bahan

- a. BKPM
- b. Komputer
- c. LCD
- d. Alat Tulis Kantor (ATK)

## 4. Pelaksanaan Praktikum



Kita buat sebuah proyek aplikasi bernama **MyAsyncTask** dengan **Empty Activity** (konfigurasi dasar) seperti pada sebelumnya.

1. Jika sudah, kondisikan agar **activity\_main.xml** menjadi seperti ini:

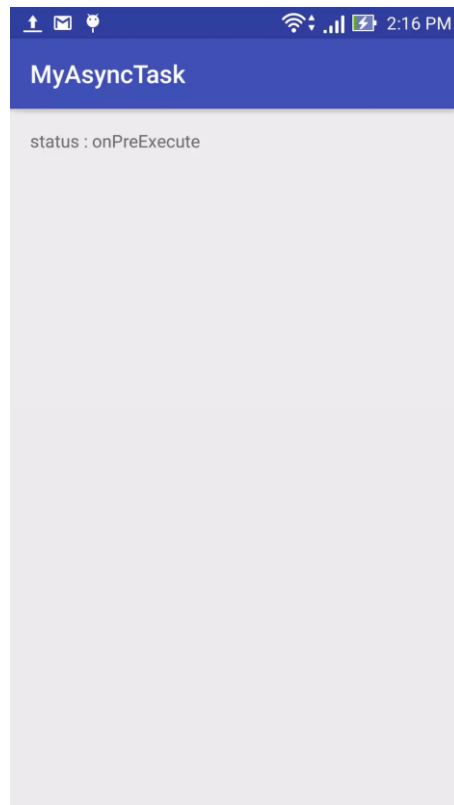
```
1. <?xml version="1.0" encoding="utf-8"?>
2. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:tools="http://schemas.android.com/tools"
4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent"
6.     android:paddingBottom="@dimen/activity_vertical_margin"
7.     android:paddingLeft="@dimen/activity_horizontal_margin"
8.     android:paddingRight="@dimen/activity_horizontal_margin"
9.     android:paddingTop="@dimen/activity_vertical_margin"
10.    tools:context="com.dicoding.myserviceapp.MainActivity">
11.    <TextView
12.        android:id="@+id/tv_status"
13.        android:layout_width="wrap_content"
14.        android:layout_height="wrap_content"
15.        android:text="Status Aplikasi"/>
16. </RelativeLayout>
```

Kita memiliki sebuah obyek **TextView** berID **tv\_status** yang akan menampilkan proses yang sedang dijalankan oleh obyek **AsyncTask**.

2. Selanjutnya lengkapi kode pada **MainActivity** sehingga menjadi seperti ini:

```
7. setContentView(R.layout.activity_main);
8. tvStatus = (TextView)findViewById(R.id.tv_status);
9. DemoAsync demoAsync = new DemoAsync();
10. demoAsync.execute("Halo Ini Demo AsyncTask");
11. }
12. private class DemoAsync extends AsyncTask<String, Void, String>{
13.     @Override
14.     protected void onPreExecute() {
15.         super.onPreExecute();
16.         tvStatus.setText("status : onPreExecute");
17.     }
18.     @Override
19.     protected String doInBackground(String... params) {
20.         Log.d(DEMO_ASYNC, "status : doInBackground");
21.         try{
22.             Thread.sleep(5000);
23.         }catch (Exception e){
24.             Log.d(DEMO_ASYNC, e.getMessage());
25.         }
26.         return params[0];
27.     }
28.     @Override
29.     protected void onPostExecute(String s) {
30.         super.onPostExecute(s);
31.         tvStatus.setText("status : onPostExecute : "+s);
32.     }
33. }
34. }
```

3. Sekarang, silakan jalankan aplikasi **MyAsyncTask** dan perhatikan obyek **TextView** akan melakukan perubahan status terhadap obyek **DemoAsyncTask** yang dijalankan.



Pada metode `doInBackground()`, kita hanya menulis *log* prosesnya saja. Bukan sebuah pendekatan yang baik untuk mengakses komponen antarmuka yang berada pada *thread* utama (*main thread*) ke dalam *background thread*.

**Praktikum ke : 13 dan 14**

**Judul Praktikum : Service**

**Alokasi Waktu : 2 x 50 menit**

### **1. Tujuan Instruksional Khusus**

- a. Mahasiswa dapat memahami konsep dasar dari service pada android
- b. Mahasiswa dapat mengimplementasikan service pada pemrograman berbasis android

### **2. Teori**

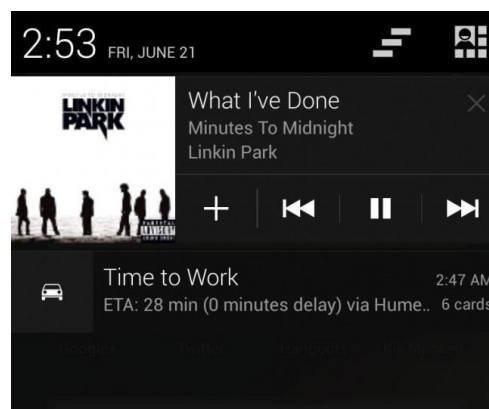
Kita telah belajar mengenai activity dan implementasinya. Activity dan fragment adalah dua komponen yang memberikan pengalaman kepada pengguna secara langsung. Pengguna dapat melihat dan berinteraksi di atasnya.

Service berada pada sisi yang lain, komponen ini tidak memiliki antarmuka dan bahkan pengguna tidak akan tahu bagaimana dia bekerja. Pengalaman yang diberikan oleh service hanya berupa proses yang tidak terlihat. Ia digunakan untuk menjalankan beragam macam proses yang memakan waktu lama.

Walaupun berjalan secara *background*, pada dasarnya service dan komponen Android lainnya berjalan pada satu proses dan *thread* yang sama yaitu *main thread* atau *ui thread*. Bekerja di *background* bukan berarti ia bekerja secara *asynchronous*. Service tetap membutuhkan *thread* terpisah jika kita ingin melakukan proses yang membutuhkan komputasi intensif atau yang memakan waktu.

Contoh pemanfaatan service sudah banyak sekali, antara lain:

- Aplikasi sosial media atau aplikasi yang memiliki kemampuan untuk menerima *push notification*. Aplikasi semacam ini pasti memiliki sebuah service yang berjalan dalam posisi *stand by* untuk selalu menerima pesan yang masuk.
- Aplikasi *chat* juga membutuhkan service untuk melakukan pengiriman dan menerima pesan yang dikirimkan oleh pengguna.
- Aplikasi pemutar musik juga melakukan hal yang sama. Untuk memberikan pengalaman yang lebih baik kepada pengguna, aplikasi pemutar musik biasanya meletakkan proses *streaming* atau memainkan musik di komponen service dengan tetap mempertahankan integrasi dengan komponen lain, misalnya notifikasi. Seperti gambar berikut :



Secara umum, terdapat dua bentuk dari service :

### 1. **Started**

Service berjenis ini adalah tipe yang dijalankan oleh komponen lain, misal activity. Sekali dijalankan, service ini akan berjalan selama belum dimatikan atau proses yang dijalankan selesai. Service akan tetap berjalan walaupun komponen yang lain dimatikan oleh sistem Android. Umumnya penggunaan service ini adalah untuk melakukan proses yang tidak memberikan nilai balik ke komponen yang memanggilnya. Contohnya adalah, mengunduh atau mengunggah berkas.

## 2. Bound

Service jenis ini merupakan tipe service yang dijalankan oleh komponen lain, namun saling mengikat. Hubungan yang terjadi antar kedua komponen tersebut seperti *client-server*. Bisa saling menerima hasil dan menerima *request* yang ada. Pada service ini dimungkinkan terjadi proses IPC (*Interprocess Communication*). Service ini akan tetap berjalan di background selama masih ada komponen lain yang mengikatnya. Jika tidak, maka Service akan dimatikan oleh sistem. Aplikasi pemutar musik merupakan salah satu jenis aplikasi yang mengimplementasikan service jenis ini.

Pada bagian ini kita akan sepenuhnya membahas service berjenis *started*. Tipe service tersebut akan dibagi menjadi dua bagian dalam implementasinya:

- **Kelas service yang inherit langsung kepada kelas Service**

Ketika sebuah kelas java *inherit* ke service ingin menjalankan proses yang memakan waktu lama, maka kelas tersebut diharuskan membuat thread terpisah agar tidak memblok ui thread yang ada. Service ini akan selalu hidup di *background* selama tidak ada komponen yang memanggil `stopService()` atau dimatikan oleh sistem.

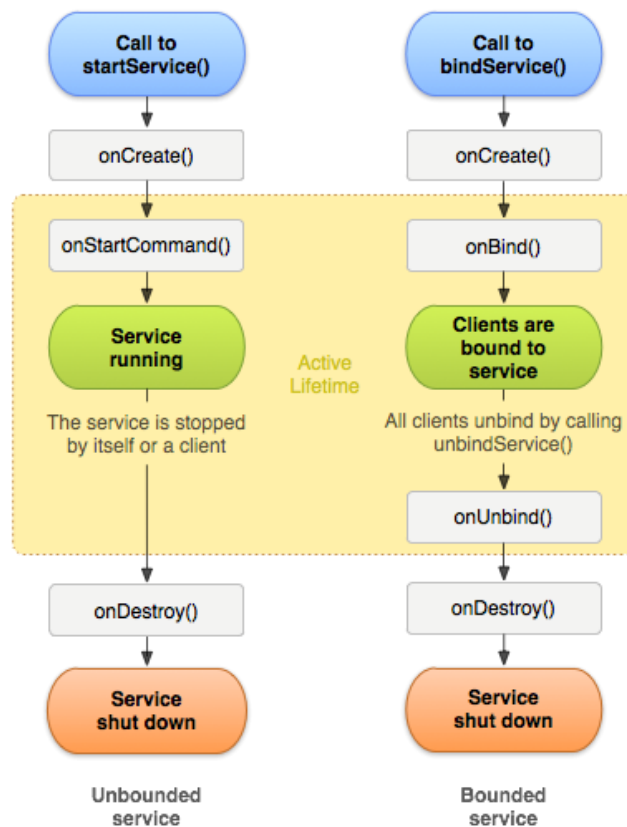
- **Kelas service yang inherit ke IntentService**

Ini adalah kelas yang sangat memudahkan hidup kita. Dia bersifat *fire and forget*, ketika ia telah menyelesaikan tugasnya, ia akan mematikan dirinya.

Poin-poin penting lain tentang service diantaranya:

- Setiap kelas Java dinyatakan sebuah service bila kelas tersebut inherit/extends ke kelas `Service` atau `IntentService`.

- Service memiliki *life cycle*-nya sendiri dan bergantung pada tipe service apa yang digunakan, started atau bound service.



- Untuk menjalankan service dari komponen lain seperti activity, cukup menggunakan `startService(Intent)`. Sebaliknya untuk mematikan/stop service terdapat dua cara. Pertama `stopService(Intent)` dijalankan dari komponen yang memanggil dan `stopSelf()` dari kelas Service itu sendiri.

Untuk mendalami topik Service lebih lanjut, Anda dapat membaca tautan berikut:

- <https://developer.android.com/reference/android/app/Service>
- <https://developer.android.com/guide/components/services>
- <https://developer.android.com/guide/topics/manifest/service-element#exported>

### 3. Alat dan Bahan

- a. BKPM
- b. Komputer
- c. LCD
- d. Alat Tulis Kantor (ATK)

### 4. Pelaksanaan Praktikum

Anda sudah paham service secara garis besar berikut pemanfaatannya. Sekarang saatnya kita menerapkannya.

1. Baik, buat proyek baru dengan nama **MyService**. Pilih **Empty Activity** dengan pilihan *default* pada *set up* proyek. Setelah proyek tercipta, lengkapi **activity\_main.xml** dengan contoh seperti ini:

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:tools="http://schemas.android.com/tools"
4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent"
6.     android:paddingBottom="@dimen/activity_vertical_margin"
7.     android:paddingLeft="@dimen/activity_horizontal_margin"
8.     android:paddingRight="@dimen/activity_horizontal_margin"
9.     android:paddingTop="@dimen/activity_vertical_margin"
10.    android:orientation="vertical"
11.    tools:context="com.dicoding.myserviceapp.MainActivity">
12.    <Button
13.        android:id="@+id/btn_start_service"
14.        android:layout_width="match_parent"
15.        android:layout_height="wrap_content"
16.        android:text="Start Service"
17.        android:layout_marginBottom="@dimen/activity_vertical_margin"/>
18.    <Button
19.        android:id="@+id/btn_start_intent_service"
20.        android:layout_width="match_parent"
21.        android:layout_height="wrap_content"
22.        android:text="Start Intent Service"/>
23. </LinearLayout>
```

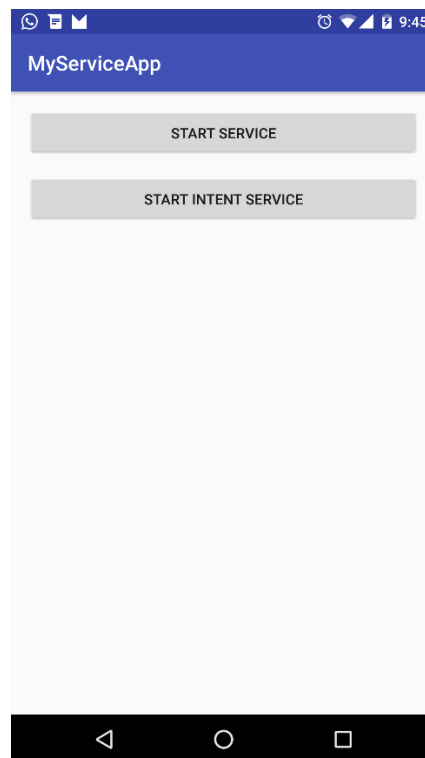
2. Pada **MainActivity.java** silakan lengkapi kode-nya menjadi sebagai berikut:

```

1. public class MainActivity extends AppCompatActivity implements View.OnClickListener {
2.     private Button btnStartService;
3.     private Button btnStartIntentService;
4.
5.     @Override
6.     protected void onCreate(Bundle savedInstanceState) {
7.         super.onCreate(savedInstanceState);
8.         setContentView(R.layout.activity_main);
9.
10.        btnStartService = (Button)findViewById(R.id.btn_start_service);
11.        btnStartService.setOnClickListener(this);
12.        btnStartIntentService = (Button)findViewById(R.id.btn_start_intent_service);
13.        btnStartIntentService.setOnClickListener(this);
14.    }
15.
16.    @Override
17.    public void onClick(View v) {
18.        switch (v.getId()){
19.            case R.id.btn_start_service:
20.                break;
21.            case R.id.btn_start_intent_service:
22.                break;
23.        }
24.    }
25. }

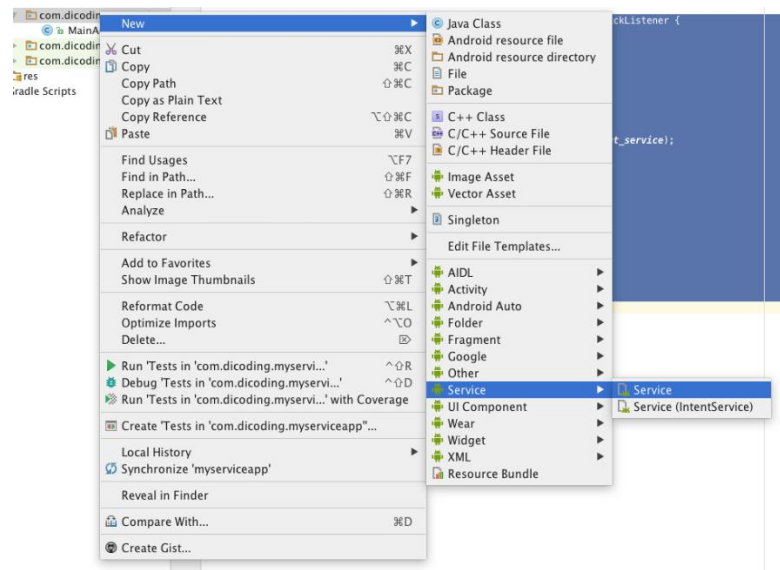
```

3. Tampilan yang seharusnya ada pada **MainActivity**, adalah seperti ini.

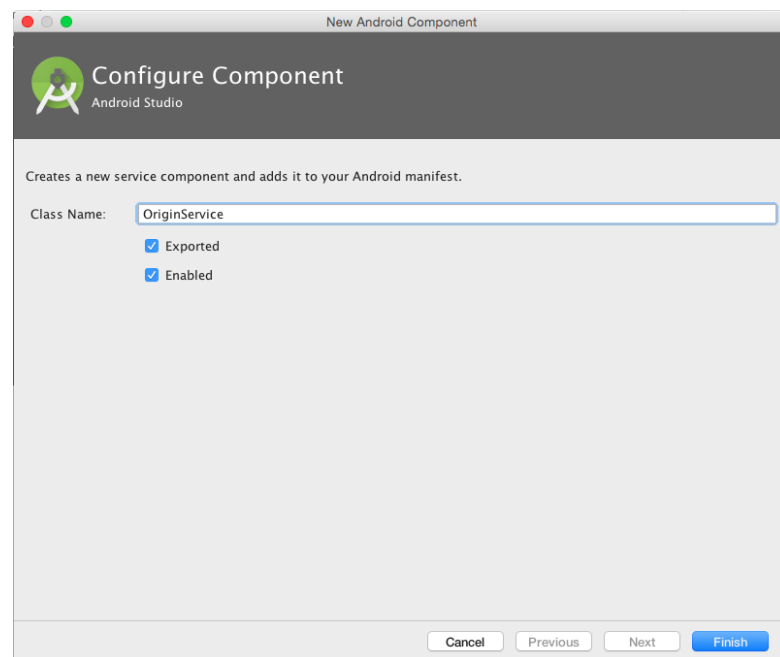


4. Lanjut, buat kelas service bernama **OriginService** dengan cara **klik kanan** pada **package > project > New > Service > Service**. **OriginService** akan *inherit (extends)* kepada kelas **Service**.





5. Selanjutnya pada dialog yang tampil, isikan nama kelas service yang diinginkan. Di sini kita menamainya sebagai OriginService dan biarkan exported dan enabled tercentang. Klik Finish untuk menyelesaikan proses.



6. Selanjutnya, buka berkas **AndroidManifest.xml** pada package manifest dan perhatikan isi berkas tersebut. Service yang baru saja kita buat sudah ada didalam tag <application> :

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.     package="com.dicoding.myseviceapp">
4.     <application
5.         android:allowBackup="true"
6.         android:icon="@mipmap/ic_launcher"
7.         android:label="@string/app_name"
8.         android:supportRtl="true"
9.         android:theme="@style/AppTheme">
10.        <activity android:name=".MainActivity">
11.            <intent-filter>
12.                <action android:name="android.intent.action.MAIN" />
13.                <category android:name="android.intent.category.LAUNCHER" />
14.            </intent-filter>
15.        </activity>
16.        <service
17.            android:name=".OriginService"
18.            android:enabled="true"
19.            android:exported="true">
20.        </service>
21.    </application>
22. </manifest>

```

7. Berkas **AndroidManifest** sudah dibuat secara otomatis. Dengan demikian kita sudah bisa menjalankan kelas service tersebut. Namun, sebelum menjalankan aplikasi, lengkapi kode pada **OriginService** menjadi seperti berikut :

```

1. public class OriginService extends Service {
2.     public static final String ORIGIN_SERVICE = "OriginService";
3.
4.     public OriginService() {
5.
6.     }
7.     @Override
8.     public IBinder onBind(Intent intent) {
9.         // TODO: Return the communication channel to the service.
10.        throw new UnsupportedOperationException("Not yet implemented");
11.    }
12.    @Override
13.    public int onStartCommand(Intent intent, int flags, int startId) {
14.        Log.d(ORIGIN_SERVICE, "OriginService dijalankan");
15.        return START_STICKY;
16.    }
17. }

```

8. Selanjutnya pada **MainActivity.java** di metode **onClick()** pada case **R.id.btn\_start\_service** tambahkan baris berikut :

```

1. Intent mStartServiceIntent = new Intent(MainActivity.this, OriginService.class);
2. startService(mStartServiceIntent);

```

9. Sehingga kode pada metode **onClick()** menjadi seperti ini :

```

1. @Override
2. public void onClick(View v) {
3.     switch (v.getId()){
4.         case R.id.btn_start_service:
5.             Intent mStartServiceIntent = new Intent(MainActivity.this, OriginService.class);
6.             startService(mStartServiceIntent);
7.             break;
8.         case R.id.btn_start_intent_service:
9.             break;
10.    }
11. }

```

10. Sekarang jalankan aplikasi. Klik tombol ‘start service’ dan perhatikan pada log-nya. **OriginService** telah dijalankan dan tidak akan pernah mati sampai dimatikan oleh sistem atau metode **stopSelf()** atau **stopService()** dijalankan.
11. Baik, sekarang kita akan menambahkan sebuah inner class **AsyncTask**. Ia seakan-akan menjalankan sebuah proses secara *asynchronous* dan mematikan/menghentikan dirinya sendiri dengan memanggil metode **stopSelf()**. Lengkapi kodenya menjadi sebagai berikut:

```
1. public class OriginService extends Service {
2.
3.     public static final String ORIGIN_SERVICE = "OriginService";
4.
5.     public OriginService() {
6.
7.     }
8.
9.     @Override
10.    public IBinder onBind(Intent intent) {
11.        // TODO: Return the communication channel to the service.
12.        throw new UnsupportedOperationException("Not yet implemented");
13.    }
14.
15.    @Override
16.    public int onStartCommand(Intent intent, int flags, int startId) {
17.        Log.d(ORIGIN_SERVICE, "OriginService dijalankan");
18.        ProcessAsync mProcessAsync = new ProcessAsync();
19.        mProcessAsync.execute();
20.        return START_STICKY;
21.    }
22.
23.    private class ProcessAsync extends AsyncTask<Void, Void, Void>{
24.        @Override
25.        protected Void doInBackground(Void... params) {
26.            try {
27.                Thread.sleep(3000);
28.            } catch (InterruptedException e) {
29.                e.printStackTrace();
30.            }
31.            return null;
32.        }
33.
34.        @Override
35.        protected void onPostExecute(Void aVoid) {
36.            super.onPostExecute(aVoid);
37.            Log.d(ORIGIN_SERVICE, "StopService");
38.            stopSelf();
39.        }
40.    }
41.
42.    @Override
43.    public void onDestroy() {
44.        super.onDestroy();
45.        Log.d(ORIGIN_SERVICE, "onDestroy()");
46.    }
47. }
```

12. Jalankan aplikasinya. Klik tombol ‘start service’ dan perhatikan *log*-nya. Service dijalankan secara *asynchronous* dan mematikan dirinya sendiri setelah proses selesai.
13. Jika berhasil dijalankan, pada log android monitor akan seperti ini :

09-22 09:52:25.028 10209-10209/com.polije.myserviceapp D/OriginService: OriginService dijalankan

09-22 09:52:28.074 10209-10209/com.polije.myserviceapp D/OriginService: StopService

09-22 09:52:28.078 10209-10209/com.polije.myserviceapp D/OriginService: onDestroy()

**Praktikum ke : 15**

**Judul praktikum : Build APK**

**Alokasi waktu : 2 x 50 menit**

## **1. Tujuan Instruksional Khusus**

- a. Mahasiswa dapat memahami tentang build APK
- b. Mahasiswa dapat membuat sebuah Build APK

## **2. Teori**

### **Build APK**

Salah satu langkah terakhir yang perlu dilakukan setelah mengembangkan aplikasi Android adalah membuat berkas *executable* dalam format APK (Android Application Package). Berkas ini yang akan didistribusikan oleh Google Play ke pengguna. Jadi, ketika hendak mempublikasikan Aplikasi ke Google Play, berkas inilah yang harus unggah.

Jika belum memahami berkas APK, maka dapat menyamakannya dengan berkas *exe* di windows atau *ipa* di iOS.

Cara membuat file APK di Android terbilang cukup mudah. dapat menggunakan sebuah *wizard* atau melalui *command line*. Pada modul ini, kita akan fokus menggunakan *wizard*.

Membuat APK dapat dibagi menjadi 2:

1. Dengan menggunakan *default keystore*
2. Dengan menggunakan *custom keystore*

*Keystore* adalah sebuah berkas biner yang berisi informasi tentang satu atau lebih *private key*. *Private key* ini digunakan untuk mencegah pemalsuan aplikasi. Konsep umumnya adalah:

- Sistem Operasi Android mewajibkan semua APK *di-sign* sebelum terpasang ke dalam device.
- Proses *signing* ini membutuhkan *Public* dan *Private Key*.
- Proses *signing* ini berlangsung selama pembuatan APK dalam mode *debug* maupun *released*.
- Sebuah sertifikat *digital public key*, atau *identity certificate*, berisi informasi mengenai sertifikat itu sendiri dan *metadata* dari pemilik sertifikat tersebut. Pemilik sertifikat ini biasanya adalah developer yang mengembangkan aplikasi.
- *Public key* yang digunakan dalam proses *signing* di atas akan dilampirkan di dalam berkas APK. Proses ini dilakukan secara otomatis oleh Android Studio.
- Ketika hendak memperbarui Aplikasi pada Google Play, maka Google Play hanya akan menerimanya bila *keystore* yang digunakan sama dengan *keystore* yang pertama kali gunakan ketika mengunggah Aplikasi tersebut ke Google Play.

Kegunaan lain dari *keystore* adalah:

- Untuk integrasi ke layanan Google seperti Google Maps dengan menggunakan nilai *hash* (digest SHA1) di dalamnya.
- Untuk integrasi ke layanan API Facebook dengan menggunakan key hash base64 yang terkandung di dalam *keystore*.

*Keystore* merupakan sebuah berkas penting yang harus jaga, terlebih ketika aplikasi memiliki jumlah unduhan pengguna yang banyak. Sebabnya, kelalaian menjaga *keystore* ini dapat menghalangi untuk memperbarui aplikasi. Akibat terburuk adalah harus melakukannya dari awal lagi.

Berikut adalah tips yang bisa gunakan untuk mengamankan *keystore* :

1. Pilih kata kunci (password) yang sulit ditebak, kombinasikan angka, alfabet dan simbol dalam membuatnya.

2. Bedakan antara *keystore password* dan *key password* ketika membuat berkas APK dengan *custom keystore*.
3. Jangan memberikan *keystore* kepada orang yang tidak dipercaya apalagi meletakkannya didalam berkas proyek aplikasi.
4. Letakan ditempat yang kamu ingat dan pastikan aman.

Untuk memahami masalah di atas lebih jauh, dapat membaca tautan berikut:

<https://developer.android.com/studio/publish/app-signing.html>

Jika ingin menemukan default keystore , maka pengguna Mac dapat menemukannya di `~/.android/debug.keystore`. Sementara itu pengguna Windows bisa menemukannya di `C:\User\YourUser\.android\debug.keystore`.

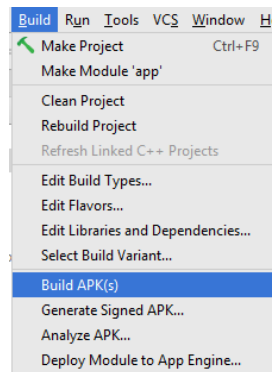
### **3. Alat dan Bahan**

- a. BKPM
- b. Komputer
- c. LCD
- d. Alat Tulis Kantor (ATK)

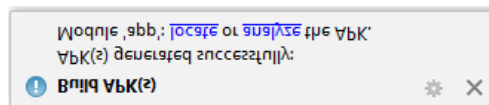
### **4. Pelaksanaan Praktikum**

Untuk mulai melakukan proses *build* APK, dapat mengikuti langkah berikut:

1. Buka kembali project kosong yang telah kita buat sebelumnya.
2. Sekarang klik menu Build → Build APK



3. Gradle akan membuat berkas APK secara otomatis. Lama proses ini bergantung pada seberapa kompleks Aplikasi yang buat, dan jumlah *dependency* yang gunakan.
4. Ketika berhasil, dapat melihat notifikasi pada sudut kanan atas Android Studio:



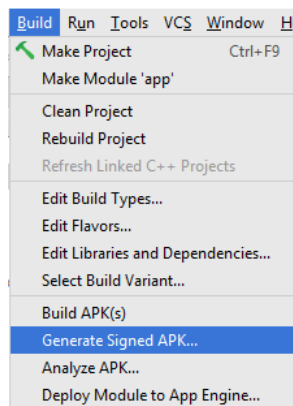
Sekarang tinggal tekan tautan yang terdapat pada notifikasi tersebut. Secara otomatis akan diarahkan ke lokasi di mana berkas APK disimpan.

Biasanya lokasinya mengikuti struktur `project-name/module-name/build/outputs/apk/`. Jika proyek bernama HelloWorld, maka lokasinya adalah **HelloWorld/app/build/outputs/apk/apk-debug.apk**

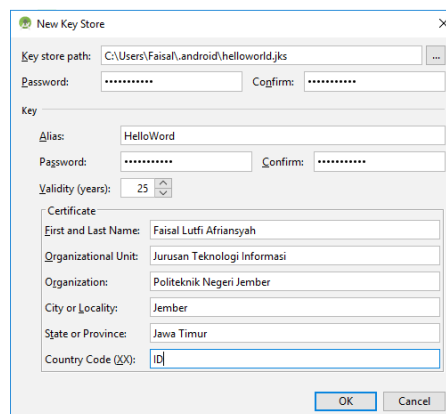
5. Sekarang sudah berhasil membuat APK dengan menggunakan default *keystore*. Ingat, APK yang baru saja buat akan ditolak oleh Google Play Store jika mencoba mengunggahnya ke Google Play Store. Agar dapat diterima, harus menjalankan proses *signing* atau generate APK tersebut dengan menggunakan *custom keystore*.
6. Sekarang coba pindahkan berkas APK yang baru dibuat ke dalam sebuah peranti (device). Buka lokasi berkas tersebut menggunakan *file explorer* pada device tersebut. Kemudian lakukan instalasi aplikasi seperti biasa.

Selamat, aplikasi Android baru sudah terpasang di peranti. Berkas APK ini bisa berikan ke pengguna lain untuk dicoba.

7. Mudah bukan? Sekarang kita lanjut membuat APK dengan *custom keystore*.
8. Kembali ke Proyek, klik Build → Generate Signed APK



9. Selanjutnya, pilih **create new**. Pada form yang tampil, lengkapi isian di dalamnya. Contoh pengisiannya adalah seperti gambar di bawah ini:



Berikut penjelasan tiap isiannya sebagai berikut:

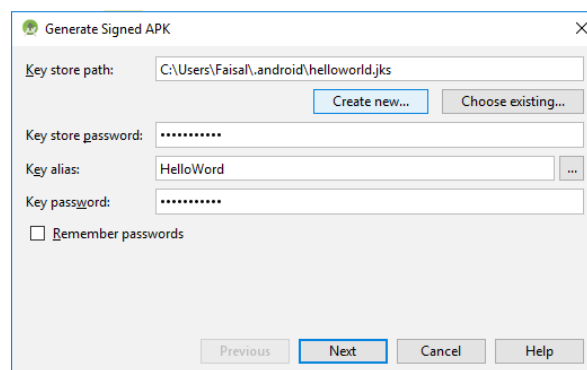
- Keystore path : perlu menentukan dimana lokasi *keystore*
- Password : Isikan keystore password minimal 6 digit dan bedakan dengan keypassword dibawahnya
- Alias : Alias dari keystore



- Password : keypassword
- Validity : Berapa lama keystore akan valid (dalam hitungan tahun)
- Firstname hingga Country Code : isikan *metadata*. Penting untuk mengisi data ini dengan benar.

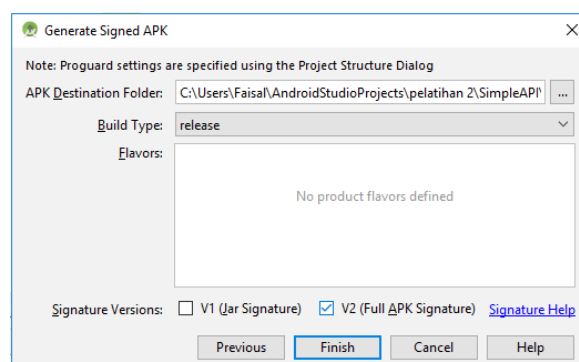
10. Setelah selesai klik **OK**.

11. Dialog yang di awal akan secara otomatis terisi ketika sudah berhasil mengisi form sebelumnya. Klik **next** untuk melanjutkan.



12. Jika ditanyakan kata kunci, masukan kata kunci yang gunakan untuk laptop atau komputer.

13. Selanjutnya tentukan di mana menyimpan APK yang dihasilkan. Di sini kami membiarkan secara *default*. Klik **finish** untuk memulai *generate signed APK*.



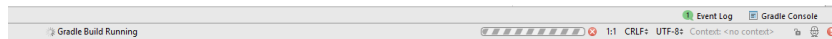
diharuskan memilih Signature V1 atau V2. Google menyarankan untuk memilih V2 karena *signature* jenis ini akan membuat instalasi APK lebih cepat. Selain itu, ia lebih aman terhadap pergantian (*alteration*) program

dengan tujuan yang tidak baik. Ikuti tautan ini untuk keterangan lebih lanjut.

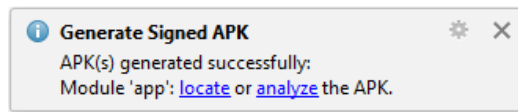
[https://developer.android.com/about/versions/nougat/android-](https://developer.android.com/about/versions/nougat/android-7.0#apk_signature_v2)

[7.0#apk\\_signature\\_v2](https://developer.android.com/about/versions/nougat/android-7.0#apk_signature_v2)

14. Perhatikan gradle process di status bar bagian bawah untuk melihat progress signed/generate APK.

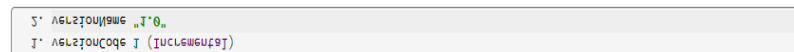


15. Ketika berhasil, notifikasi seperti berikut akan tampil :



16. Selamat APK versi *released* telah berhasil dibuat. Proses ini perlu ketika hendak mempublikasikan aplikasi di Google Play Store dan memperbaruinya di kemudian waktu.

Ketika melakukan perbaruan aplikasi, jangan lupa untuk mengubah nilai yang ada di dalam build.gradle(Module:app):



## 5. Tugas

Build APK sample code kemudian running program pada device