
# Mobile Programming

## Catatan Kuliah #5

Alauddin Maulana Hirzan, M. Kom
0607069401

# Pemrograman Android

# Pemrograman Android

## Pemrograman Awal Android #1

Untuk dapat membuat aplikasi berbasiskan Android, mahasiswa memahami tahapan-tahapan yang harus dilakukan. Hal ini untuk memastikan bahwa mahasiswa memahami betul proses-proses yang terjadi di Android Studio.

Karena dalam proses pembuatan aplikasi Android sudah dalam bentuk satu proyek terintegrasi. Sehingga mahasiswa tidak perlu membuat folder projek terpisah.

# Pemrograman Android

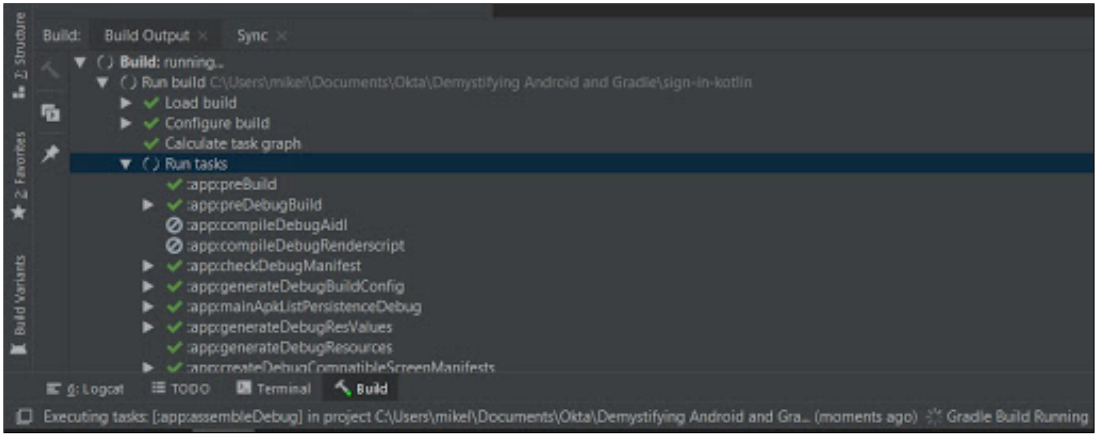
## Pemrograman Awal Android #2

Ketika membuat projek untuk pertama kali maupun di saat-saat tertentu, Android Studio akan melakukan **Gradle Building** yang berfungsi membangun projek untuk siap dijalankan di media virtual maupun perangkat sungguhan.

Proses ini juga akan melakukan pengunduhan secara otomatis pertama kali projek dibuat. Sehingga dalam prosesnya membutuhkan waktu.

# Pemrograman Android

## Pemrograman Awal Android #3



# Pemrograman Android

Pemrograman Awal Android #4

Ketika proses building berhasil, maka Android Studio akan menampilkan **Layout** pertama kali yang bisa programmer edit sesuai dengan kebutuhan. Langkah yang bisa dilakukan berikutnya adalah melakukan pemrograman antarmuka dan fungsional.

**Info**

Pemrograman Visual berupa antarmuka merupakan hal vital dari pengembangan aplikasi Android. Karena pemrograman visual sangat bergantung kepada layout yang dibuat. Dengan pengecualian aplikasi Map.

# Pemrograman Antarmuka / Visual

# Pemrograman Antarmuka / Visual

## Pemrograman Antarmuka #1

Sesuai dengan namanya, ditahap ini programmer perlu menyiapkan tampilan dari aplikasi yang dibuat. Karena pemrograman fungsional sangat bergantung dari visual/antarmuka yang dibuat, maka pemrograman tersebut dilakukan setelah antarmuka dibuat.

Dianjurkan untuk membuat dengan langkah seperti berikut:

1. Antarmuka 1 $\rightarrow$ Fungsional 1
2. Antarmuka 2 $\rightarrow$ Fungsional 2
3. Antarmuka 3 $\rightarrow$ Fungsional 3
4. dst

# Pemrograman Antarmuka / Visual

## Pemrograman Antarmuka #2

Pembuatan antarmuka/visual dapat dilakukan dengan dua langkah sederhana:

- Dengan Kode XML
- Dengan Design Editor

**Info**

Programmer dapat membuat **Activity Pertama** dengan menggunakan **Template** maupun dengan cara manual.

# Pemrograman Antarmuka / Visual

## Pemrograman Antarmuka #2 - Pemrograman XML #1

Android Studio mendukung pemrograman antarmuka dengan menggunakan bahasa pemrograman **eXtensible Markup Language (XML)**. Namun dalam implementasinya cukup menimbulkan kompleksitas dikarenakan tata bahasa yang unik hanya untuk Android Studio.

Sama seperti bahasa HTML, penggunaan XML harus diapit diantara dua **tag Tanda Kurung Siku (<> </>)**

| **Info** |
| --- |
| Contoh: <Button /> atau <ConstraintLayout> .... </ConstraintLayout> |

# Pemrograman Antarmuka / Visual

## Pemrograman Antarmuka #2 - Pemrograman XML #2

Perhatikan contoh berikut:

```
<EditText
    android:id="@+id/inputPanjang"
    android:layout_width="0dp"
    android:layout_height="50dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="50dp"
    android:layout_marginEnd="16dp"
    android:ems="10"
    android:hint="Masukkan Panjang"
    android:inputType="number"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    tools:ignore="SpeakableTextPresentCheck" />
```

# Pemrograman Antarmuka / Visual

Pemrograman Antarmuka #2 - Pemrograman XML #3

Contoh yang diberikan merupakan objek View berupa **EditText** atau field teks yang bisa di-isi. Selain itu objek ini juga memiliki **attributes** tambahan yang bisa diubah melalui Android Studio di panel kanan setelah objek di klik. Objek ini memiliki **attribut** standar berupa:

- id : Identifikasi Objek untuk Kotlin/Java, bersifat unik
- layout_width : Ukuran Lebar (0dp = match constraint, manual, atau wrap_content)
- layout_height : Ukuran Tinggi (0dp = match constraint, manual, atau wrap_content)
- layout_margin : Batasan
- dst

# Pemrograman Antarmuka / Visual

Pemrograman Antarmuka #2 - Designer Mode #1

Namun karena penggunaan XML tidak ramah untuk pemula, maka Android Studio mengusulkan mode **Designer** untuk mempermudah pembuatan antarmuka aplikasi dengan mode **Drag-n-Drop**.

Apa yang diperlukan oleh programmer hanya perlu ditarik ke kanvas Activity dan secara otomatis objek akan muncul di halaman tersebut.

# Pemrograman Antarmuka / Visual

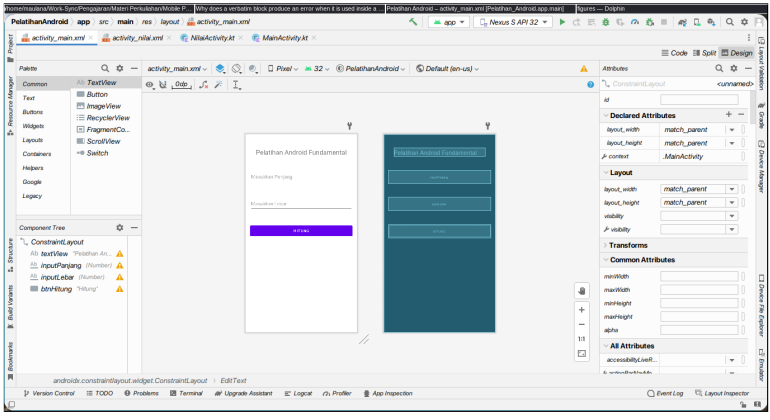
## Pemrograman Antarmuka #2 - Designer Mode #2

**Designer Mode** ini memiliki beberapa komponen utama berupa:

- **Project Tree** : Folder Projek (Panel Paling Kiri)
- **Palette** : Tempat memilih jenis objek untuk ditarik (Panel Kiri Sebelah Kanvas)
- **Attributes** : Tempat mengubah attribut masing-masing objek (Panel Paling Kanan)
- **Debugging Panel** : Tempat melihat status building/debugging/error (Terletak di bawah, namun tersembunyi)
- **Toolbar Projek** : Tempat melakukan Building, Running, maupun Stopping Aplikasi

# Pemrograman Antarmuka / Visual

## Pemrograman Antarmuka #2 - Designer Mode #3



# Pemrograman Fungsional

# Pemrograman Fungsional

## Pemrograman Fungsional #1

Pemrograman Fungsional merupakan proses memberikan fungsi, pemicu, dan respon kepada suatu objek yang ada di Activity / antarmuka. Pemrograman ini memiliki fungsi untuk menginisialisasi masing-masing objek yang ada di antarmuka, dan memberikan fungsi di dalamnya agar terlihat interaktif dengan user.

Untuk bisa dipanggil oleh kode fungsional, objek-objek view **wajib** memiliki **ID** sebagai identitas masing-masing objek.

# Pemrograman Fungsional

## Pemrograman Fungsional #2

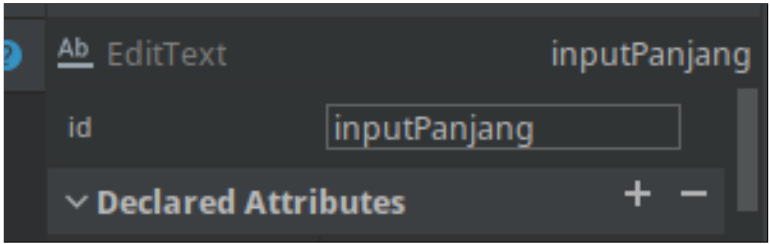
Pemrograman Fungsional dapat dilakukan secara mudah dengan cara:

1. Memberi ID kepada masing-masing objek antarmuka
2. Menginisialisasi objek ke dalam bahasa pemrograman
3. Melakukan manipulasi (menarik, menampilkan, mengubah attribut) objek
4. dan Aksi-Aksi Lainnya.

# Pemrograman Fungsional

## Pemrograman Fungsional #3 - Memulai Pemrograman #1

Cek ID objek sebelum Inisalisasi. Lihat contoh berikut:




Objek **EditText** ini memiliki id = **inputPanjang**

# Pemrograman Fungsional

## Pemrograman Fungsional #3 - Memulai Pemrograman #2

Agar **EditText** itu bisa ditarik maupun dipanggil datanya, maka dalam kode fungsional gunakan contoh kode berikut.

```
val inputPanjang = findViewById<EditText>(R.id.inputPanjang)
```

Untuk memanggil objek tersebut, bisa menggunakan **findViewById** yang ditujukan ke **R.id.<id_objek>** tersebut

# Pemrograman Fungsional

## Pemrograman Fungsional #3 - Memulai Pemrograman #3

Jika sudah diinisialisasi objek bisa dimanipulasi sesuai dengan kebutuhan user dengan menggunakan fungsi sederhana seperti:

- (variabel_objek).text.toString() = Ambil Teks dan Ubah ke String
- (variabel_objek).setText("<isi_teks>") = Set Teks
- dan lain-lain

# Pemrograman Fungsional

## Pemrograman Fungsional #3 - Memulai Pemrograman #4

Contoh kode untuk input data

```
var panjang = inputPanjang.text.toString()
var lebar = inputLebar.text.toString()
```

Dengan kode tersebut data akan diambil dan dimasukkan ke dalam variabel yang sudah dibuat.

# Pemrograman Fungsional

## Pemrograman Fungsional #3 - Memulai Pemrograman #5

Ada beberapa hal yang harus diperhatikan ketika membuat kode fungsional:

- Semua objek yang akan dimanipulasi wajib punya **ID**
- Tidak ada **ID kembar** di satu Activity/Antarmuka (Sebaiknya unik)
- Kode fungsional harus menginisialisasi **objek** agar bisa diakses
- Segala bentuk manipulasi (menarik, menampilkan data) dilakukan dalam aksi Event suatu **Tombol**


THANK YOU