



# Memahami Code Convention pada Android Studio

**Silver** - Chapter 2 - Topic 3

---

**Selamat datang di Chapter 2 Topik 3 *online course*  
Android Developer Binar Academy!**





# Hi Teman Teman 🙌

Masih di Chapter 2 niih ...

Pada topik sebelumnya, kita sudah belajar mendalam tentang tools andalan kita, **Android Studio**. Kali ini, kita akan belajar **format penulisan kode yang baik** dengan Bahasa **Pemrograman Kotlin di Android Studio**.

Yuk, lanjut!





**Detailnya, kita bakal bahas hal-hal berikut ini :**

- Pengenalan Code Convention
- Aturan Main dalam Code Convention





Disini siapa yang udah pernah denger  
Code Convention? 🤓

Seperti arti dalam bahasa Indonesia,  
Code Convention itu merupakan  
kebiasaan seorang programmer dalam  
menuliskan code.



## Apa itu Code Convention?

Code conventions itu berbeda-beda tergantung bahasa pemrograman yang digunakan.

Pada topik ini, kita bakal membahas Code Convention pada bahasa pemrograman Kotlin, terutama dalam Pengembangan Aplikasi Android.

### Translations of **convention**

noun

**konvensi**

convention, agreement, congress

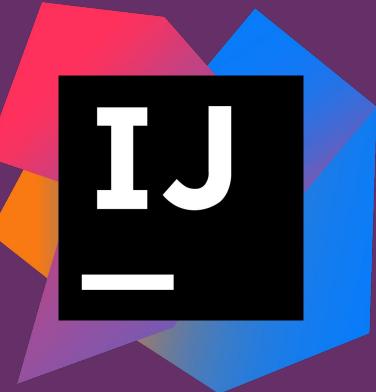
**kebiasaan**

habit, custom, practice, rut, way, convention

**rapat**

meeting, conference, assembly, convention, gathering, confabulation





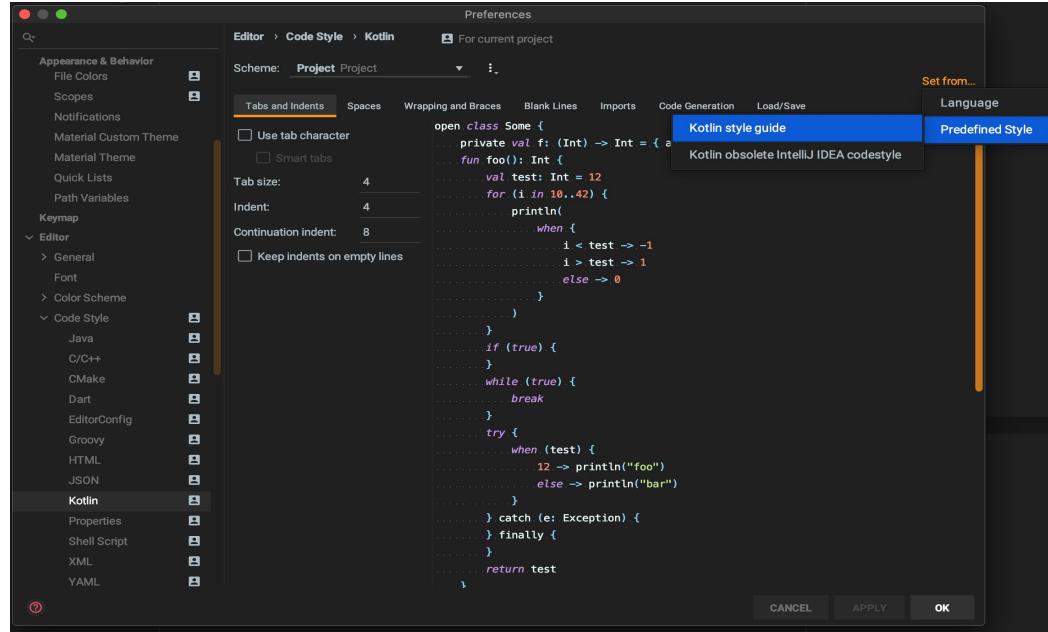
Ngomongin kebiasaan menuliskan kode, ada lho fitur yang membantu programmer untuk menampilkan kode lebih rapi.

Baik di **IntelliJ Idea** maupun **Android Studio** fitur ini bisa dipake kok.

Apa sih fitur misterius itu?



## Style Guide Kotlin pada IDE Android Studio



Kita pake fitur Code Inspection yang ada di Style Guide Kotlin pada IDE Android Studio untuk menuliskan code lebih rapi.

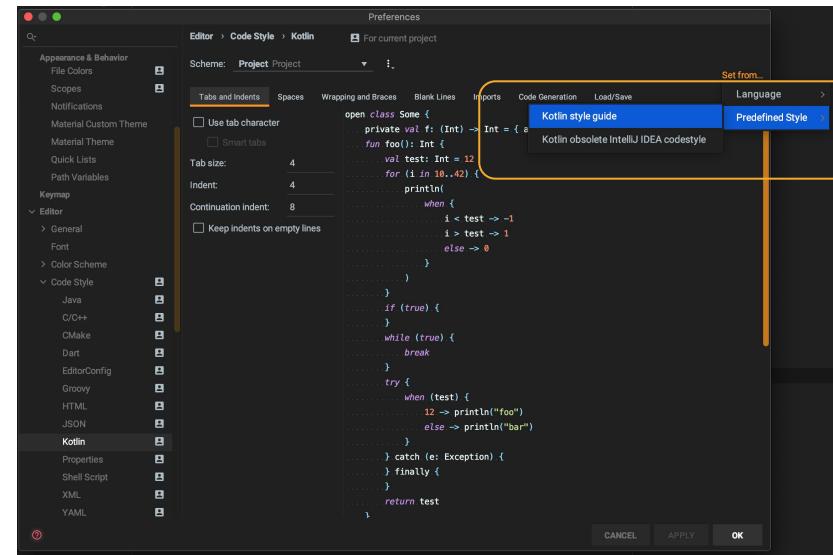


## Menerapkan Style Guide Kotlin dengan Code Inspection

Style guide Kotlin Android membantu kita menghindari kekeliruan saran format dengan aturan yang tepat dan jelas dipakai secara umum.

Pada Style Guide Kotlin ada fitur “Code Inspection” yang **membantu kita analisis penulisan kode yang bermasalah**.

Namun sebelum itu, kita perlu install dulu nih Style Guide Kotlin supaya muncul kayak gambar disamping.



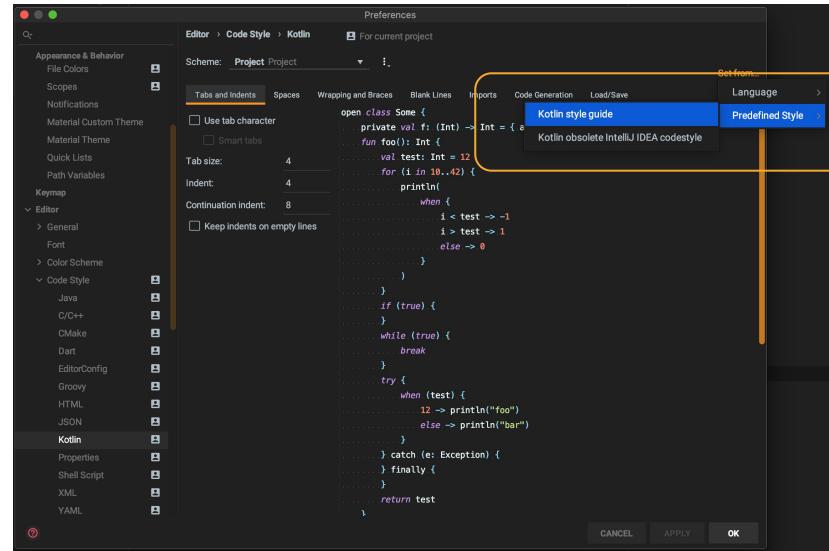


Untuk konfigurasi IntelliJ formatter sesuai dengan style guide ini, silakan install Kotlin plugin versi 1.2.20 atau yang terbaru.

Untuk menginstalnya, kamu bisa ikuti langkah berikut :

**buka File > Setting/Preferences > Editor > Code Style > Kotlin.**

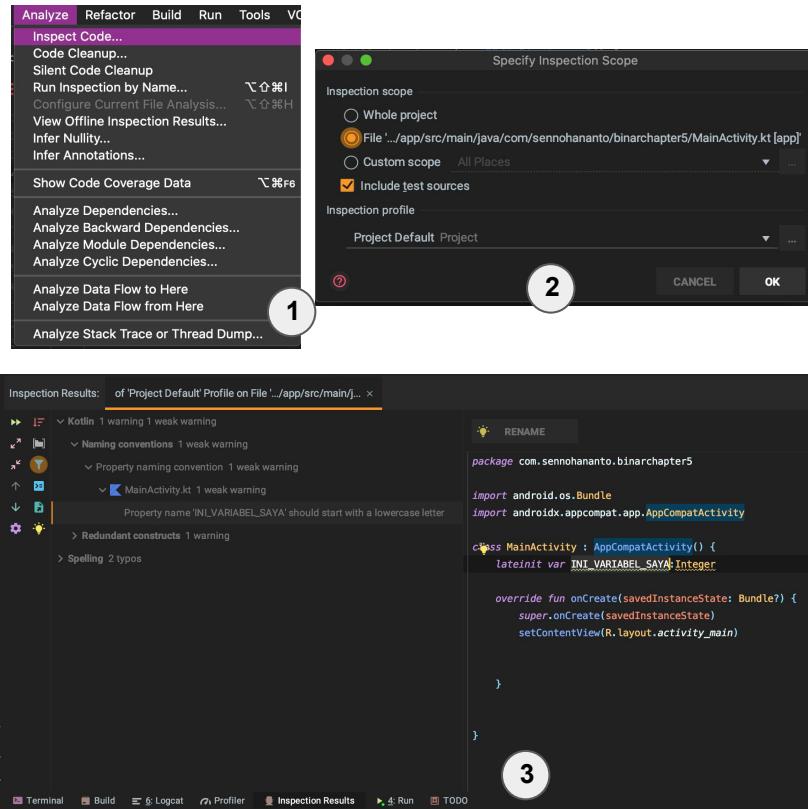
Lalu, klik **Set form...** yang berada pojok kanan dan pilih **Predefined style | Kotlin style guide** dari menu.





Setelah mengatur Style Guide Kotlin pada IDE Android Studio, maka kita akan mengetahui code mana saja yang cara penulisannya tidak sesuai Code Convention.

Pemberitahuan dan Saran Code Convention akan terlihat setelah kita melakukan Code Inspection.



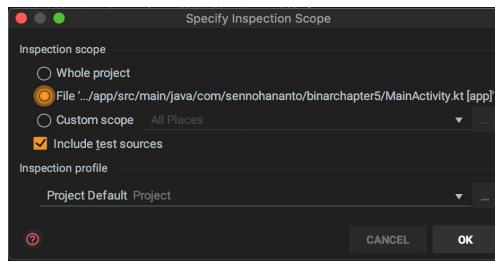
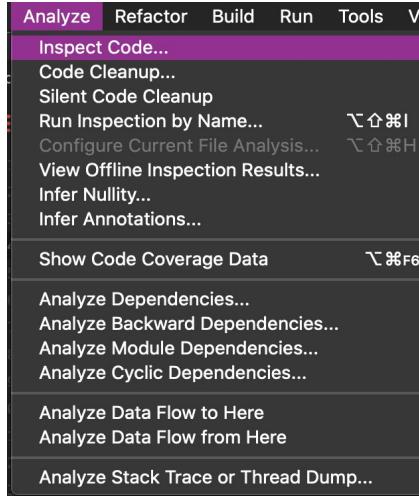
The screenshot illustrates the process of performing a code inspection in Android Studio:

- Step 1:** The "Analyze" menu is open, showing options like "Code Cleanup...", "Run Inspection by Name...", and "View Offline Inspection Results...". A circled number "1" is placed over the "Run Inspection by Name..." option.
- Step 2:** The "Specify Inspection Scope" dialog is displayed. It shows a "File" path: "/app/src/main/java/com/sennohananto/binarchapter5/MainActivity.kt [app]". The "Custom scope" radio button is selected, and the "Include test sources" checkbox is checked. A circled number "2" is placed over the "File" field.
- Step 3:** The "Inspection Results" window is shown, displaying a tree view of inspection findings under the "Project Default" profile. One finding is highlighted: "Property name 'INT\_VARIABEL\_SAYA' should start with a lowercase letter". To the right, the code editor shows the MainActivity.kt file with the problematic line: `lateinit var INT\_VARIABEL\_SAYA: Integer`. A circled number "3" is placed over the inspection result in the results window.



Cara melakukan Code Inspection cukup dengan ke menu Analyze lalu klik Inspect Code.

Setelah kamu klik Inspect Code kamu bisa memilih Specify Inspection Scope kayak di gambar di bawah ini.



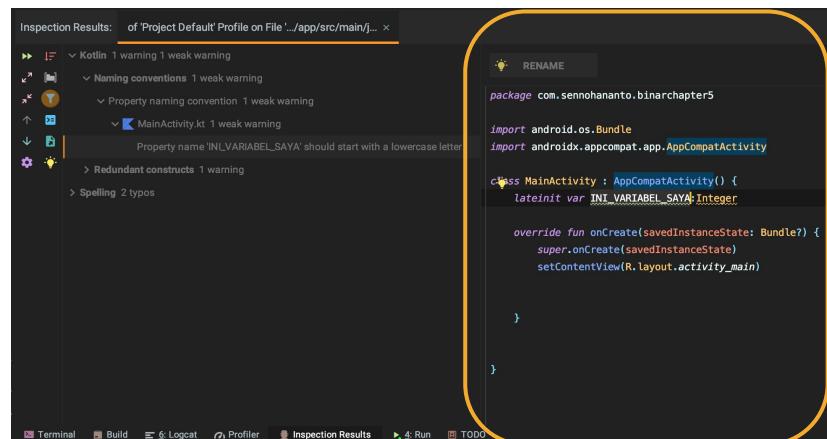


Nah kalo kamu lihat gambar disamping , terdapat sebuah variable dengan nama variable **INI\_VARIABEL\_SAYA**, yang tentunya penamaan seperti ini bukan penamaan yang baik untuk sebuah variable.

Maka code *inspection* memberitahu bahwa:

**INI\_VARIABEL\_SAYA should start with a lowercase letter**

dan disarankan untuk mengubah nama variable tersebut.



Inspection Results: of 'Project Default' Profile on File .../app/src/main/j... ×

► [F] ▾ Kotlin 1 warning 1 week warning

  ▾ Naming conventions 1 week warning

    ▾ Property naming convention 1 week warning

      >MainActivity.kt 1 week warning

      Property name 'INI\_VARIABEL\_SAYA' should start with a lowercase letter

  > Redundant constructs 1 warning

  > Spelling 2 typos

RENAME

package com.sennuhananto.binarchapter5

import android.os.Bundle

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    lateinit var INT\_VARIABEL\_SAYA: Integer

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity\_main)

    }

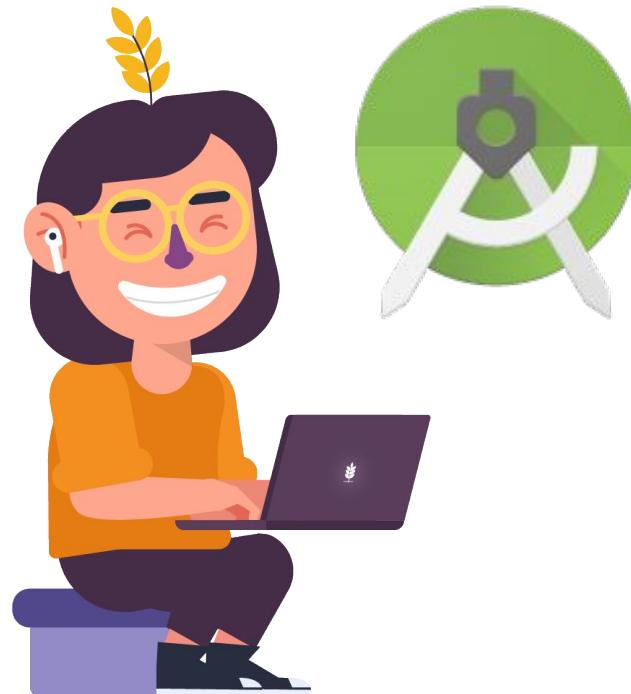
}

Terminal Build Logcat Profiler Inspection Results Run TODO



Gimana? Praktis kan? 😊

Bisa dibilang, pengecekan **udah dihafal semua sama IDE**. Sehingga yang perlu kita lakukan adalah mengikuti penamaan dan formatting dengan sebaik mungkin.





Seperti yang udah dijelasin tadi, walaupun IDE bekerja secara ajaib tapi kita tetap harus **memperhatikan formatting yang sesuai**.

Kita perlu **mengetahui aturan main dasar yang berlaku**.



# Apa saja aturan mainnya?

Berikut adalah beberapa “aturan main” dasar yang berlaku

- 1) Struktur Direktori Package
- 2) Penamaan Package
- 3) Penamaan File Source
- 4) Class Layout
- 5) Implementasi Interface Layout
- 6) Overload Layout
- 7) Penamaan Method
- 8) Penamaan Property
- 9) Penamaan File Resources





### 1) Struktur Direktori / Package

Dalam project Kotlin, struktur direktori yang disarankan adalah **mengikuti struktur package dengan package aplikasi sebagai induknya**.



The screenshot shows the Android Studio interface. On the left is the Project Navigational Bar, which displays the package structure:

```
app
  manifests
  java
    org
      example
        kotlin
          MainActivity.kt
          Person.kt
```

On the right is the code editor for `MainActivity.kt`. The code is:

```
1 package org.example.kotlin
2
3 import ...
```

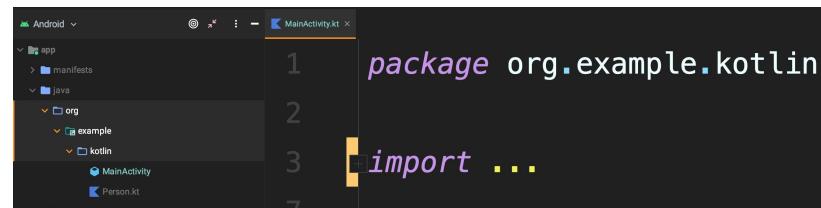
A yellow rectangular highlight is placed over the package declaration line (line 1) and the import statement line (line 3).



Sebagai contoh, kalo nama package aplikasi adalah **org.example.kotlin**, maka file dengan package itu harus ditempatkan langsung di bawah package tersebut.

Lalu misalkan ada package bernama **socket** di dalam package **network**, berarti full name packagenya **org.example.kotlin.network.socket**.

Nah file dengan package **org.example.kotlin.network.socket** harus berada di subdirektori **socket**.



The screenshot shows the Android Studio interface. On the left is the Project Navigational Drawer, which lists the project structure: app, manifests, java, org, example, and kotlin. Inside the kotlin folder, there are files named MainActivity.kt and Person.kt. On the right is the code editor for MainActivity.kt, displaying the following code:

```
1 package org.example.kotlin  
2  
3 import ...
```



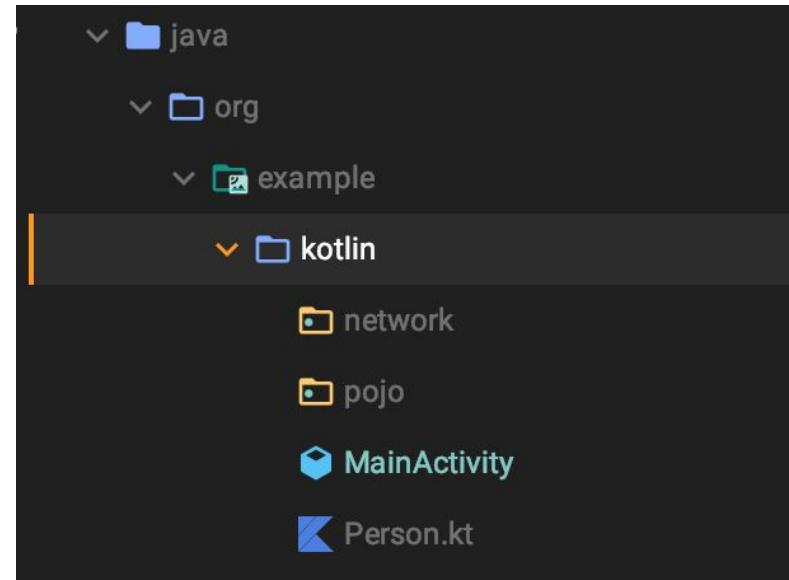
### 2) Aturan penamaan Direktori / Package

Oh ya perlu diingat juga, aturan penamaan package adalah :

- Memakai semua huruf kecil / tidak kapital,
- Tidak mengandung angka, dan
- Tidak menggunakan simbol apapun, termasuk simbol dash (-) dan underscore (\_).

Contohnya nama package seperti :

***pojo, network, ui, db, entity***, dll.





### 3) Nama File Source Code

Jika file Kotlin berisi satu class, nama filenya harus sama dengan nama class, dengan ekstensi ".kt" ditambahkan di belakangnya.

Jika file berisi beberapa class, atau hanya deklarasi top-level, pilih nama yang menjelaskan isi dari file tersebut, dan beri nama file yang sesuai.

Untuk penamaan *File Source Code* maupun Class, gunakan **PascalCase** atau **camel case dengan huruf pertama besar** (Misalnya, **ProsesDeklarasi.kt**).

The screenshot shows the Android Studio interface with the project navigation bar on the left and a code editor on the right. The code editor displays a file named 'Person.kt' containing the following Kotlin code:

```
1 package org.example.kotlin
2
3 class Person(val name: String, val age: Int)
4
5 init {
6     println("Person with name $name and age $age has been created")
7 }
8
9 fun introduce(){
10     println("Hello i am $name. I am $age Years Old")
11 }
```

The screenshot shows the Android Studio interface with the project navigation bar on the left and a code editor on the right. The code editor displays a file named 'Animals.kt' containing the following Kotlin code:

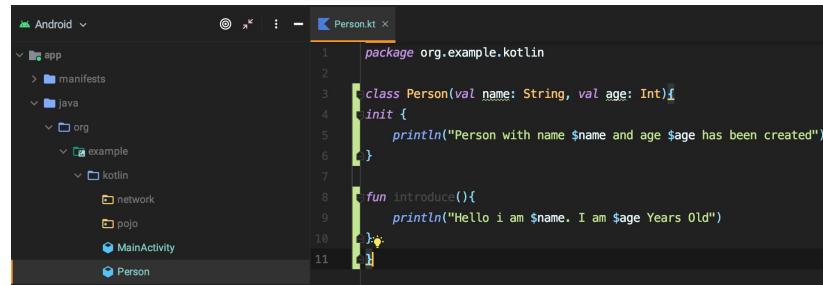
```
1 package org.example.kotlin
2
3 class Snake(val long: Int, val color: String){
4     //Some code here
5 }
6
7 class Sheep(val furColor: String, weight: Double){
8     //Some code here
9 }
```



### Hindari penamaan file yang tidak menjelaskan apapun.

Nama file harus menjelaskan apa yang dilakukan code dalam file tersebut.

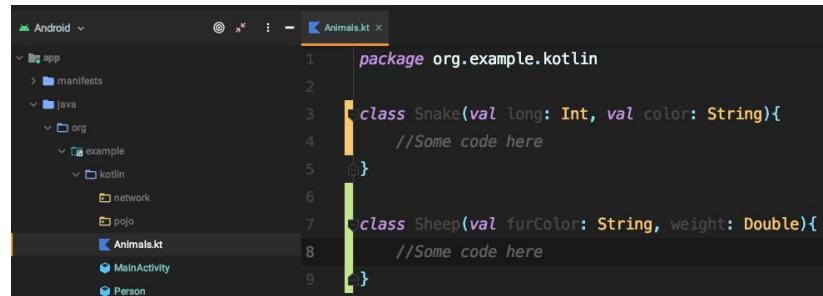
Misalnya ada nama class **ApaAjaBoleh.kt** kita nggak bisa ngebayangin class itu tujuannya dan fungsinya untuk apa.



```
package org.example.kotlin

class Person(val name: String, val age: Int) {
    init {
        println("Person with name $name and age $age has been created")
    }

    fun introduce(){
        println("Hello i am $name. I am $age Years Old")
    }
}
```



```
package org.example.kotlin

class Snake(val long: Int, val color: String){
    //Some code here
}

class Sheep(val furColor: String, weight: Double){
    //Some code here
}
```



### 4) Class Layout

Secara umum, isi class diurutkan dalam urutan berikut :

1. Deklarasi property dan blok inisialisasi
2. Constructor sekunder
3. Deklarasi method
4. Companion Object

Jangan mengurutkan deklarasi method berdasarkan abjad atau berdasarkan visibility. **Urutkan method berdasarkan keterkaitannya.**

```
package org.example.kotlin

class Person(val name: String, val age: Int) {
    //Property Declaration
    lateinit var address: String

    init {
        println("Person with name $name and age $age has been created")
    }

    constructor(name: String, age: Int, address: String): this(name, age){
        // Some code
        this.address = address
    }

    fun introduce() {
        println("Hello i am $name. I am $age Years Old")
    }

    companion object{
        //Static variables and Static methods here
    }
}
```

```
fun introduce() {
    println("Hello i am $name. I am $age Years Old")
    sayAddress()
}

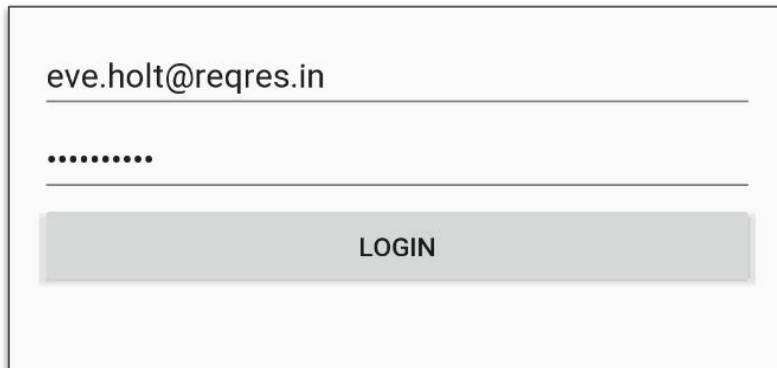
fun sayAddress(){
    println("I live in $address")
}
```



## 5) Implementasi Interface Layout

Saat mengimplementasikan interface, **pastikan setiap View dalam urutan yang sesuai dengan tampilannya.**

Sebagai contoh, dalam mendesain di layout XML, sebisa mungkin code-nya juga berurutan berdasarkan tampilan yang dilihat pengguna. Coba deh kamu lihat gambar sebelah kiri. Ketika kamu melakukan login maka urutannya akan ditampilkan kayak gambar di sebelah kanan.

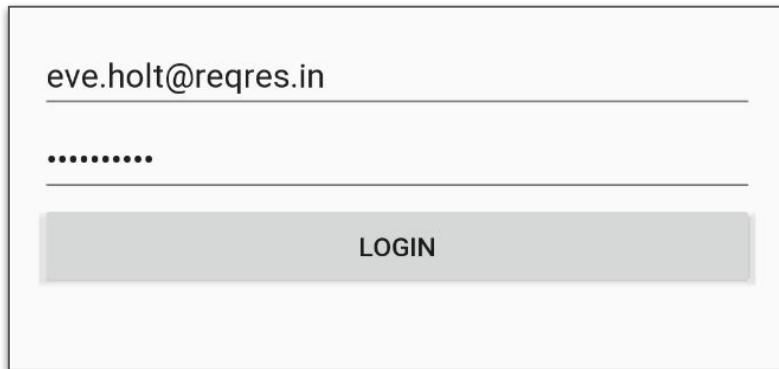


```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout ...>
    <EditText
        android:id="@+id/etUsername"
        .../>
    <EditText
        android:id="@+id/etPassword"
        .../>
    <Button
        android:id="@+id/btnLogin"
        .../>
</androidx.constraintlayout.widget.ConstraintLayout>
```



Bagi yang menerapkan **LinearLayout** sebagai **ViewGroup**, mungkin hal ini menjadi wajib.

Namun perlu diperhatikan bila menggunakan **ConstraintLayout** atau pun **RelativeLayout** sebagai **ViewGroup**, karena urutan dalam code XML jika menggunakan ViewGroup tersebut tidak terlalu berpengaruh.



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout ...>
    <EditText
        android:id="@+id/etUsername"
        .../>
    <EditText
        android:id="@+id/etPassword"
        .../>
    <Button
        android:id="@+id/btnLogin"
        .../>
</androidx.constraintlayout.widget.ConstraintLayout>
```



### 6) Overload Layout

Selalu menempatkan overloads di samping satu sama lain di class.

Kalau kita review sedikit,, overload adalah function yang namanya sama persis namun memiliki parameter yang berbeda urutan dan tipe datanya.

Jadi, sebisa mungkin **menempatkan function yang overload bersebelahan**.

```
fun eat(){
    println("Is eating...")
}

fun eat(foodName: String){
    println("Is eating $foodName")
}
```



Pada contoh disamping, terdapat function overload **eat** sehingga suatu class memiliki 2 function yang sama, namun beda parameter.

Sebisa mungkin, letakkan functionnya saling bersebelahan ya~

```
fun eat(){
    println("Is eating...")
}

fun eat(foodName: String){
    println("Is eating $foodName")
}
```



### 7) Aturan Penamaan Fungsi

Nama fungsi, property, dan variabel lokal dimulai dengan huruf kecil. Kamu juga harus menggunakan **camelCase** dan tanpa underscore.

**Pengecualian:** fungsi bawaan yang digunakan untuk membuat *instance class* dapat memiliki nama yang sama dengan class yang dibuat.

Cek contohnya di Gambar yes~



```
fun processDeclarations() { /*...*/ }
var declarationCount = 1

abstract class Foo { /*...*/ }
class FooImpl : Foo { /*...*/ }
fun FooImpl(): Foo { return FooImpl() }
```



### 8) Aturan Penamaan Property

**Konstanta** adalah property yang ditandai dengan const, atau top-level atau property val object tanpa fungsi pemanggilan khusus yang menyimpan immutable data yang tidak dapat diubah.

Konstanta harus menggunakan **nama yang dipisahkan dengan underscore dan huruf besar**.

Nama top-level atau property object yang menyimpan object dengan behavior atau data yang bisa mutable (mutable) harus menggunakan penamaan camel case.



```
//ini constant
const val MAX_COUNT = 8

//This is constant too
val USER_NAME_FIELD = "UserName"

//Mutable Property using camelCase
val mutableCollection: MutableSet<String> = HashSet()
```



## Catatan Penamaan untuk mendukung Property

Jika sebuah class memiliki dua property yang secara konseptual sama tetapi yang satunya dapat diakses secara **public**, gunakan underscore sebagai awalan untuk nama property **private**.

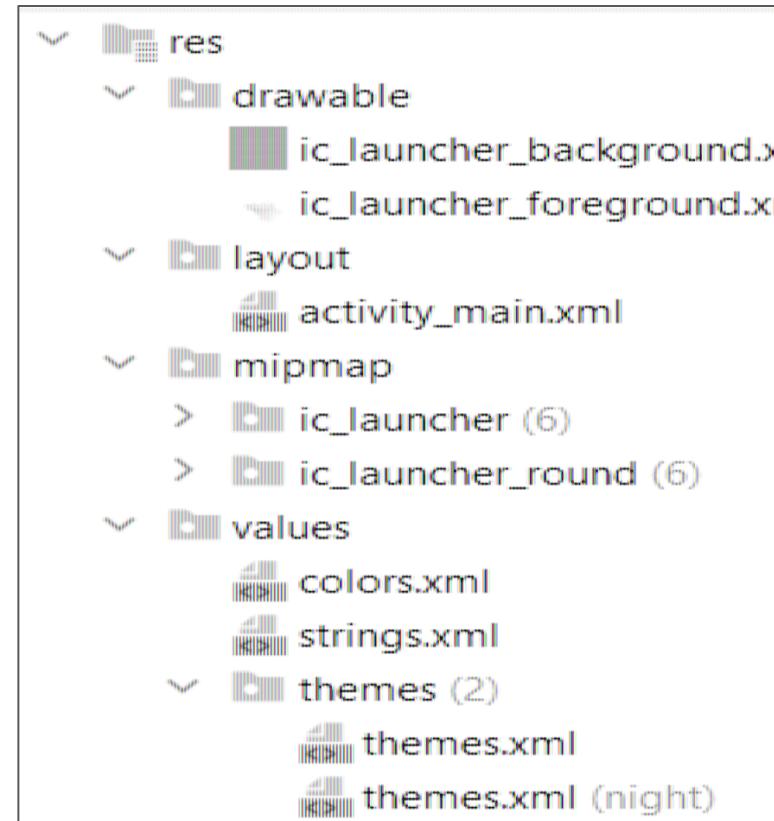
```
class C {  
    private val _elementList = mutableListOf<Element>()  
    val elementList: List<Element>  
        get() = _elementList  
}
```

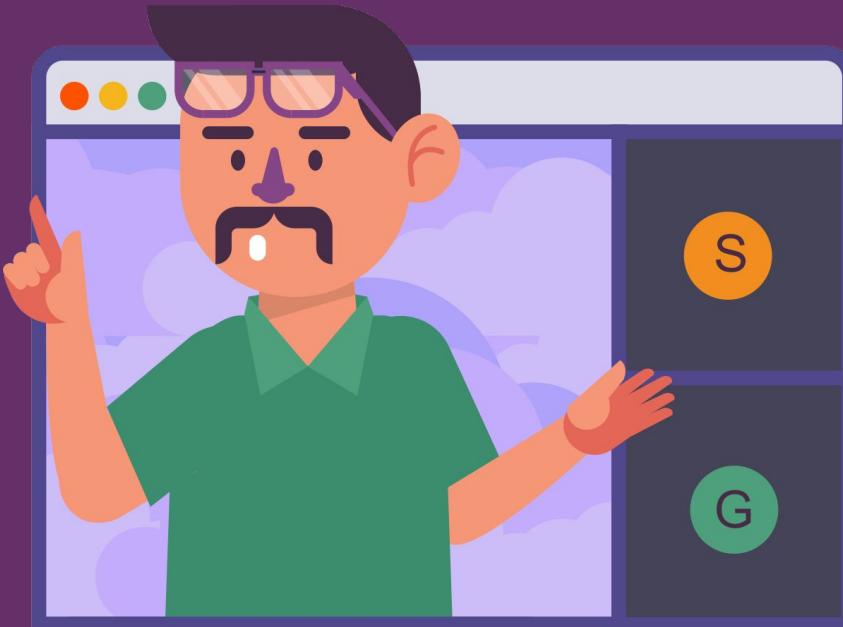


## Aturan Penamaan File Resources

File Resources adalah semua file yang ada di dalam folder **res**. Termasuk file yang ada di dalam folder drawable, layout, assets, mipmap, values, dan kawan-kawannya.

Penamaan file resources menggunakan format ***snake\_case***. Format penamaan ***snake\_case*** yaitu menggunakan huruf kecil semua dan menggunakan underscore (\_) untuk memisahkan antar kata.





**Yuk kita Recall dulu!**

**Apa saja sih aturan penulisan Case dan ciri-cirinya?**

- **Sebutkan satu-persatu!**
- **Berikan contoh!**
- **Diskusikan di Miro / Aplikasi lain [Sesuaikan dengan Fasil]**

# Saatnya kita Quiz!





## 1. Bagaimana penulisan konstanta yang sesuai Kotlin Code Convention?

- A. INIKONSTANTA
- B.INI\_KONSTANTA
- C. iniKonstanta



## 1. Bagaimana penulisan konstanta yang sesuai Kotlin Code Convention?

- A. INIKONSTANTA
- B.INI\_KONSTANTA
- C. iniKonstanta

Penulisan konstanta menggunakan `snake_case` dengan huruf kapital semuanya.



## 2. Bagaimana penulisan nama Variabel yang baik?

- A. iniVariabelSaya
- B.INI\_VARIABEL\_SAYA
- C. IniVariabelSaya



## 2. Bagaimana penulisan nama Variabel yang baik?

- A. iniVariabelSaya
- B.INI\_VARIABEL\_SAYA
- C. IniVariabelSaya

Nama variabel iniVariabelSaya menerapkan camelCase yang diterapkan untuk penamaan variabel / property



### 3. Bagaimana penulisan nama class yang baik?

- A. KelasJahat
- B. KELASJAHAT
- C. Kelas\_Jahat



### 3. Bagaimana penulisan nama class yang baik?

- A. KelasJahat
- B. KELASJAHAT
- C. Kelas\_Jahat

Nama class KelasJahat memenuhi penamaan class yang baik, yaitu camelCase dengan huruf kapital di awal, atau bisa juga disebut dengan PascalCase.



#### 4. Urutan deklarasi property dalam class ditempatkan pada urutan yang ...

- A. Pertama
- B. Kedua
- C. Ketiga



## 4. Urutan deklarasi property dalam class ditempatkan pada urutan yang ...

- A. Pertama
- B. Kedua
- C. Ketiga

Yaps, betul!. Penulisan deklarasi property / variable ditempatkan pada awal blok class dan tidak tersebar. Hal ini juga membantu programmer untuk mengetahui class yang sedang dibuka memiliki property apa saja dengan mudah.



## 5. Penulisan nama file resource di project Android Studio menggunakan ...

- A. camelCase
- B. PascalCase
- C. snake\_case



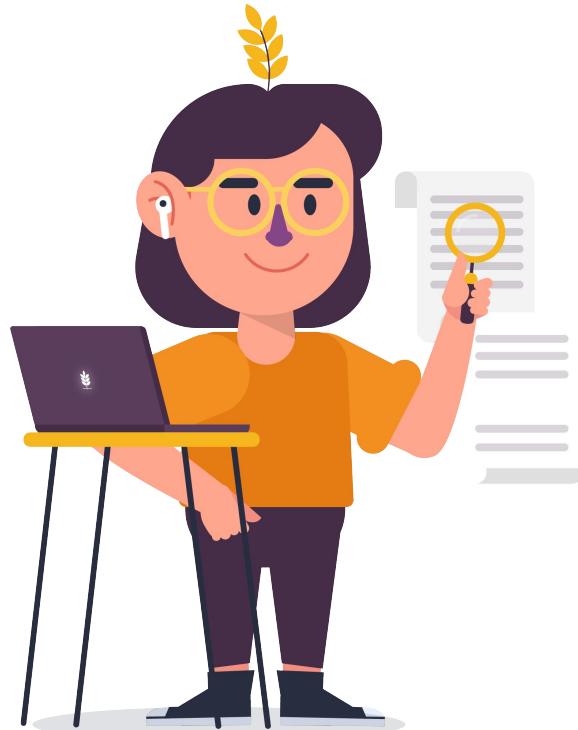
## 5. Penulisan nama file resource di project Android Studio menggunakan ...

- A. camelCase
- B. PascalCase
- C. snake\_case

Yaps, betul!. Penamaan file resource pada Android Studio project menggunakan **snake\_case**. Yaitu penamaan menggunakan huruf kecil semuanya, lalu setiap kata dipisahkan dengan underscore (\_)

## Referensi dan bacaan lebih lanjut~

1. [Coding conventions | Kotlin](#)





**Nah, selesai sudah pembahasan kita di Chapter 1 Topic 3 ini.**

**Selanjutnya, kita bakal belajar penggunaan Android Studio dalam pembuatan layout UI aplikasi.**

**Penasaran kayak gimana? Cus langsung ke topik selanjutnya~**



# Terima Kasih!



Next Topic

loading...