STRUKTUR DATA

Chapter 1 - Topic 2

Selamat datang di Chapter 1 Topic 2 online course Android Developer dari Binar Academy!

Haaii Binarian 👋

Masih di Chapter 1 niih~

Pada **Topik 1** sebelumnya, kita sudah berkenalan dan mencoba terjun ke dunianya Android Developer.

Nah, untuk **Topik 2** ini kita akan kenal lebih lanjut tentang Bahasa Android, Kotlin. Kita akan kenalan dengan struktur data, yang erat kaitannya dengan variabel dan tipe data, dan lainnya.

Sesi ini akan praktik lewat aplikasi **IntelliJ Idea**, yang sudah kita instalasi kemarin. Buat kamu yang belum menginstall Intellij Idea pada sesi ini, bisa menggunakan online compiler. Contohnya seperti **play kotlin.lang** Compiler.

So, let's start~

Detailnya, kita bakal bahas hal-hal berikut ini:

- Struktur data dalam Kotlin
- Variable data dalam Kotlin
- Tipe-tipe data dalam Kotlin
- Elemen Array dan implementasinya
- Elemen list dan implementasinya

Pada topik sebelumnya kita PDKT sama Android dan bahasa yang mereka pakai, kali ini kita akan coba akrabin diri sama **Struktur Data.**

Apakah **Struktur Data** itu?

Penjelasan lebih lengkap bisa dipelajarin disini~

Apa sih Struktur Data itu..

Struktur data adalah cara khusus untuk **menyimpan** dan **mengatur** seluruh data yang ada di komputer kamu supaya bisa dipake secara efisien.

Sama kaya lemari sebagai tempat kita menyimpan dan mengatur seluruh baju, kalau lemarinya rapi tentu aja kita bisa ambil baju dengan efisien.

Nah, struktur data kurang lebih mirip kayak kita menata baju di lemari itu 😆

Kalau Kita Analogikan, Begini Nih...

Coba deh bayangin kalo kamu nyimpen baju kayak gambar di samping ini.

Nah, gambar di samping tuh **gambaran kalo kamu nyimpen data tanpa pake Struktur data**.

Berantakan gak karuan. Kamu akan bingung di laci mana kamu menyimpan kaos kaki, atau bingung juga mencari pakaian untuk datang ke pesta 😅

Oke, sekarang coba kalo kamu nyimpen baju pake **kategorisasi** kayak gambar di samping.

Ini sama kayak kalo kamu nyimpen data pake struktur data.

Terus, di dalam Struktur Data, ada Tipe Data nya sendiri ...

Ngomongin lemari, tentu aja ada kategori baju yang perlu dirapikan. Ada baju atasan dan bawahan. Struktur data juga punya elemen datanya sendiri.

Misalnya, kita punya bestie bernama Sabrina yang umurnya 25 tahun. Kalo dibedah berarti ...

- **“Sabrina”** ini disebut dengan tipe data **string** (tipe data yang berisi kumpulan karakter) dan
- **25 tahun** adalah tipe data **integer** (tipe data numerik)

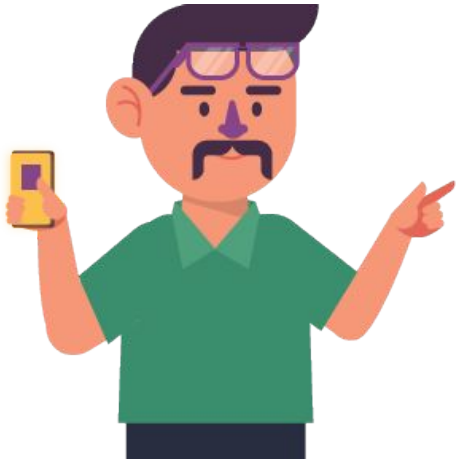
Pada struktur data, kita perlu tahu **caranya ngasih elemen data** dalam beberapa sisi yang berhubungan sama pengaturan dan penyimpanan yang lebih baik.

Seperti yang udah dijelasin sebelumnya, Struktur data punya tipe datanya sendiri.

Tapi, nggak cuma itu~ Struktur Data juga berkaitan erat dengan Variable.

Apa itu VARIABLE?

VARIABLE adalah **tempat** yang bakal kita pake buat menyimpan sebuah **nilai.**



Kenal lebih dekat dengan Variable

Pembuatan variable pada Kotlin lebih mudah dibandingkan pada Java. Kita cukup pakai keyword **var** atau **val** untuk membuat sebuah variable pada Kotlin.

Variable pada Kotlin, dibedakan berdasarkan sifatnya:

1. Imutable: read-only **(val)**
2. Mutable: read and write **(var)**

Immutable berarti cuma bisa sekali pakai, variable ini nggak bisa diisi ulang lagi nilainya atau read only.

Contoh:

```
var namaKamu = "Binarian"

// isi ulang nilainya
namaKamu = "Binarian Brother"

println(namaKamu) // outputnya akan menjadi Binarian Brother
```

Mutable adalah variable yang bisa diisi lagi nilainya. Jadi kita bakal bisa mengisinya berulang-ulang sesuai dengan kebutuhan kita.

Contoh:

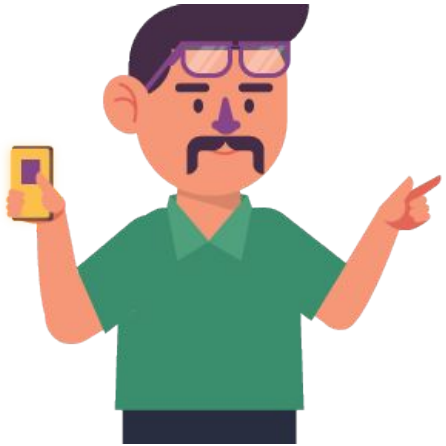
```
val namaKamu = "Binarian"
val umur = 23

// jika kita coba isi ulang nilainya, maka akan terjadi error
// karena variabel ini bersifat imutable
namaKamu = "Binarian Brother"
```

Binarian, kita harus banyak deketin hal baru nih!

Setelah berhasil deketin Variabel, selanjutnya kita akan deketin **TIPE DATA.**

Apa itu **TIPE DATA?**

TIPE DATA adalah tipe dan ukuran data yang sesuai dengan nilai yang kita simpan.



Gampangnya begini..

Sebuah tempat penyimpanan dengan label **lemari** dan **rakSepatu** di samping ini kita sebut sebagai **Variable.**

Variable **lemari** punya tipe data **pakaian** dan **rakSepatu** punya tipe data **sepatu**.

Jadii, kalau disimpulkan :

- Variable = lemari dan rakSepatu
- Tipe data = pakaian dan sepatu

Nangkep kaan 😆

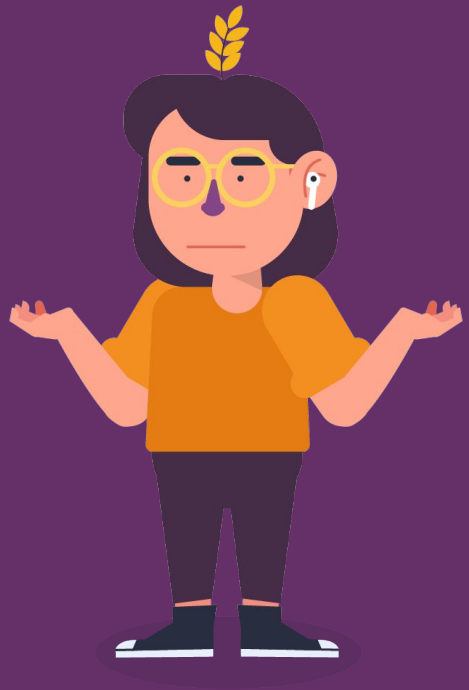
lemari

rakSepatu

Tadi kan kita bahas ada tipe data pakaian dan sepatu.

Eits.. Binarian ini bukan lagi belanja yah! Kembali ke laptop 😁

TIPE DATA itu ada apa aja? Yuk kenalan~



Tipe Data pada Kotlin

1. Bilangan
- Byte
- Short
- Integer
- Long
- Float
- Double

2. Logika
- Boolean

3. Karakter
- Char

- **Menyimpan Data**
  - Array

Binarian, bisa dibilang Variable dan Tipe Data itu udah kayak Bestfriend Forever alias saling melengkapi~

Kalau kita ngomongin Variable di Kotlin, pasti nggak bisa dipisahkan dari Tipe Data. Itu dia yang aku maksud Bestfriend banget!

Tapi apakah dua teman sejati ini bisa langsung digabungkan menjadi satu?

Tentunya kita perlu melakukan deklarasi **Variable** dengan **Tipe Data** dulu.

Caranya mudah kok!

Cara deklarasi Variable dengan Tipe Data

Format dari pendeklarasian variabel di Kotlin

```
<tipe variable> namaVariabel : <tipe data>
```

Cara deklarasi Variable dengan Tipe Data

Mendeklarasikan variable kosong dengan tipe data

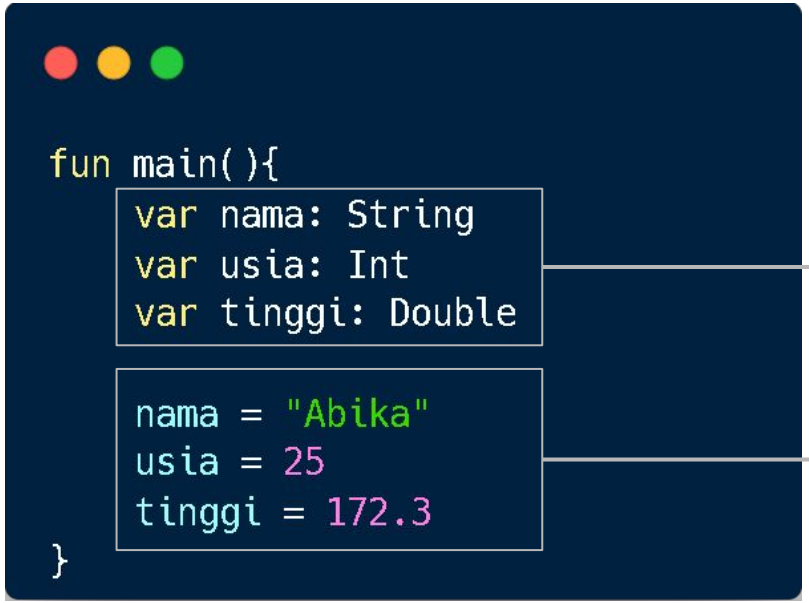
```
val panjang: Int
```

Cara deklarasi Variable dengan Tipe Data

Mendeklarasikan variabel salah satu tipe data yaitu **integer** dan langsung diisi nilainya

```
val panjang: Int = 40
```

Cara deklarasi Variable dengan Tipe Data



Mendeklarasikan variable dengan tipe data

Memberikan value pada variable

*Pemberian value pada variable harus sesuai dengan tipe data yang kita deklarasikan di atas.

Binarian, masih inget nggak Tipe Data yang ada di Kotlin tadi?

Siap-siap yaa~ sekarang kita akan kenalan sama Tipe Data-nya satu per satu.

Byte

Tipe data **bilangan** dengan ukuran 8 bit yang punya nilai minimum -128 dan nilai maksimum 127, nilai *default*-nya adalah 0.

Contoh :

```
val x: Byte = 127
val y: Byte = -128
```

Short

Tipe data **bilangan** dengan ukuran 16 bit yang punya nilai minimum -32768 dan nilai maksimum 32767, nilai default-nya adalah 0.

Contoh :

```
val a: Short = 32767
val b: Short = -32768
```

Int

Tipe **bilangan** dengan ukuran 32 bit yang punya nilai minimum -2147483648 dan nilai maksimum 2147483647, nilai *default*-nya adalah 0.

Contoh :

```
val a: Int = 2147483647
val b: Int  = -2147483648
```

Long

Tipe data **bilangan** ukuran 64 bit yang punya nilai minimum -9223372036854775808 dan nilai maksimum 9223372036854775807, nilai default-nya adalah 0L.

Contoh :

```
val a: Long = 9223372036854775807
val b: Long =
-9223372036854775808
```

Float

Tipe data yang menandai nilai desimal dengan ruang penyimpanan ukuran 32 bit. Karenanya, Tipe data **float** nggak cocok dipake buat nilai-nilai mata uang.

Nilai default buat tipe data **float** ini adalah 0.0f.

Contoh :

```
val suhu: Float =
3.14f
val f1: Float = 234.5f
```

Double

Tipe data yang menandai nilai desimal dengan ruang penyimpanan ukuran 64 bit.

Nilai default buat tipe data **float** ini adalah 0.0d.

Contoh :

```
val pi: Double = 3.14
val r: Double = 21
```

Boolean

Tipe data **boolean** dipakai untuk menghasilkan nilai logika yang cuma punya 2 nilai, yaitu true dan false.

Tipe data **boolean** banyak dipakai jadi penanda di kondisi true atau false dan sering dipakai bersamaan dengan if else. Nah, kalo nilai default dari tipe data **boolean** ini adalah false.

Contoh:

```
val a: Boolean = true
val b: Boolean = false
```

Char

Tipe data **char** kita pake buat menyatakan suatu karakter Unicode yang butuh ukuran 16 bit.

Unicode ini tuh sebenernya kumpulan karakter yang ada di semua bahasa seperti Bahasa Yunani, Bahasa Arab, dan Bahasa Latin.

Tipe data **char** juga dipake buat menampung karakter apa aja atau cuma 1 karakter aja.

Contoh :

```
val golonganDarah: Char = 'A'
```

String

Tipe data **String** pada Kotlin digunakan untuk menyimpan urutan karakter. Nilai string harus diapit oleh tanda kutip ganda (“”) atau tanda kutip tiga (“”” “”””) .

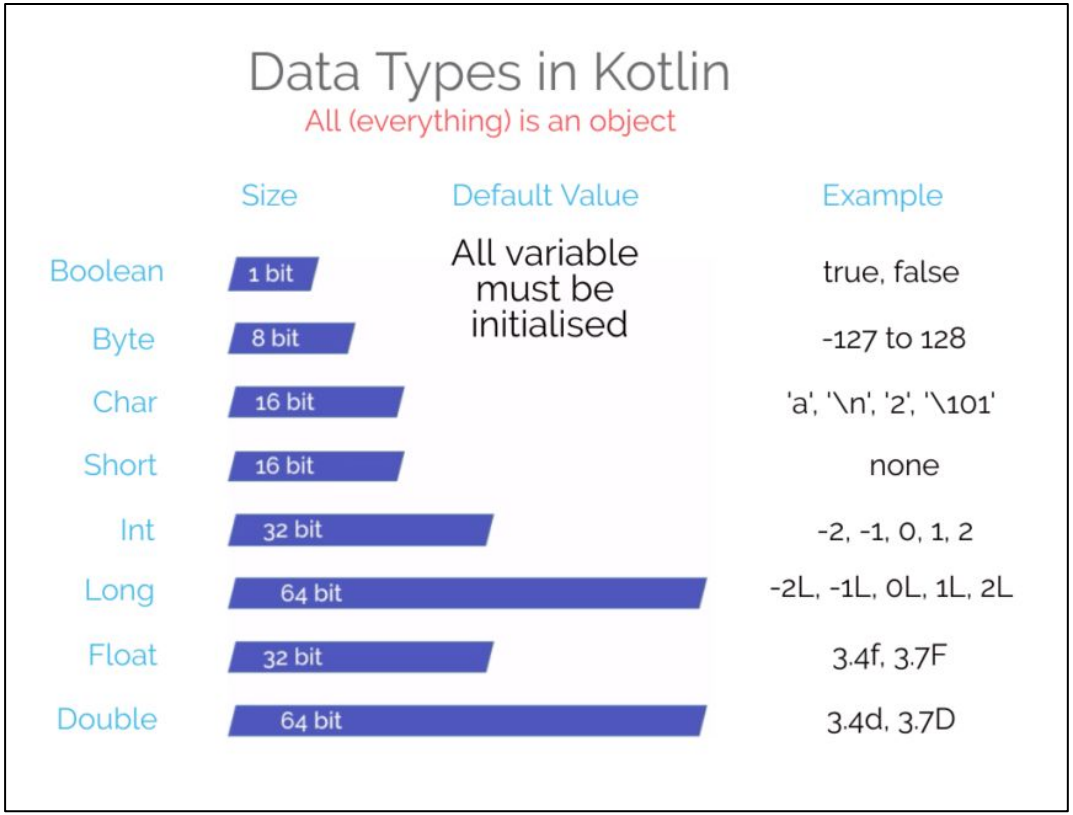
String memiliki dua jenis yang tersedia pada Kotlin. Pertama disebut **Escaped String** dan satu lagi **Raw String**.

- **Escaped String** dideklarasikan dalam tanda kutip ganda dan dapat berisikan karakter escape seperti “\n”, “\t”, “\b”, dll.
- **Raw String** dideklarasikan dalam tanda kutip tiga dan dapat berisi beberapa baris teks tanpa karakter escaped.

```
val escapedString: String = ”Naik delman istimewa!\n”
val namaBelakang: String = ”””Ku duduk di muka, Ku duduk samping pak kusir yang sedang bekerja ”””
```

Sudah paham tentang tipe-tipe data di Kotlin?

Kurang lebih, bisa disimpulkan begini nih..



Saatnya kita

Quiz!

BINAR

1. Pada potongan kode di dibawah, isilah titik-titik tersebut agar kode dapat berjalan!

```
fun main(){

    ....

    namaRobot = "Transformer"
    kodeRobot = 'L'
    tinggiRobot = 500
}
```

A.
```
val namaRobot: String

var kodeRobot: Char

val tinggiRobot: Float
```

B.
```
var namaRobot: Char

val kodeRobot: Int

var harga Robot: Boolean
```

C.
```
var namaRobot: String

var kodeRobot: Char

var hargaRobot: Int
```

BINAR

1. Pada potongan kode di dibawah, isilah titik-titik tersebut agar kode dapat berjalan!

```
fun main(){

    ....

    namaRobot = "Transformer"
    kodeRobot = 'L'
    tinggiRobot = 500
}
```

A.
```
val namaRobot: String

var kodeRobot: Char

val tinggiRobot: Float
```

Tipe variable yang digunakan harus bersifat mutable (var) karena datanya dapat berubah atau diisi ulang.

namaRobot merupakan data **String** karena dapat mengimplementasikan

lebih dari satu karakter.

kodeRobot cocok banget sama tipe data **char** karena cuma bisa mengimplementasikan **satu karakter** aja.

tinggiRobot lebih cocok pakai tipe data **integer** karena variable tinggiRobot dilihat dari potongan code akan diisi oleh **bilangan bulat.**

2. Bayangkan kamu memiliki sebuah robot. ketika robot dinyalakan selama 1 jam suhu robot tersebut akan naik menjadi 27,8 derajat, jika dideklarasikan menjadi tipe data, maka syntaxnya seperti apa?

A. val suhuRobot: Float = 27.8f

B. val suhu Robot: Int = 27.8

C. val suhuRobot: Short = 27.8

BINAR

2. Bayangkan kamu memiliki sebuah robot. ketika robot dinyalakan selama 1 jam suhu robot tersebut akan naik menjadi 27,8 derajat, jika dideklarasikan menjadi tipe data, maka syntaxnya seperti apa?

A. **val suhuRobot: Float = 27.8f**

B. val suhu Robot: Int = 27.8

C. val suhuRobot: Short = 27.8

Floating point merupakan bilangan pecahan yang bisa kita gunakan untuk mempresentasikan nilai-nilai yang mengandung **pecahan** maupun **decimal** yang memiliki **angka di belakang koma.** Cara penulisannya sangatlah berbeda dalam kehidupan sehari-hari kita yang membuat angka dibelakang koma.
Di dalam pemrograman Kotlin sendiri **koma akan diganti dengan titik sebagai komanya**, misalnya nilai ujian kita rata-rata **3,7** pada Kotlin akan menjadi **3.7** untuk cara penulisannya.

Binarian, sekarang kita mulai masuk ke Array.

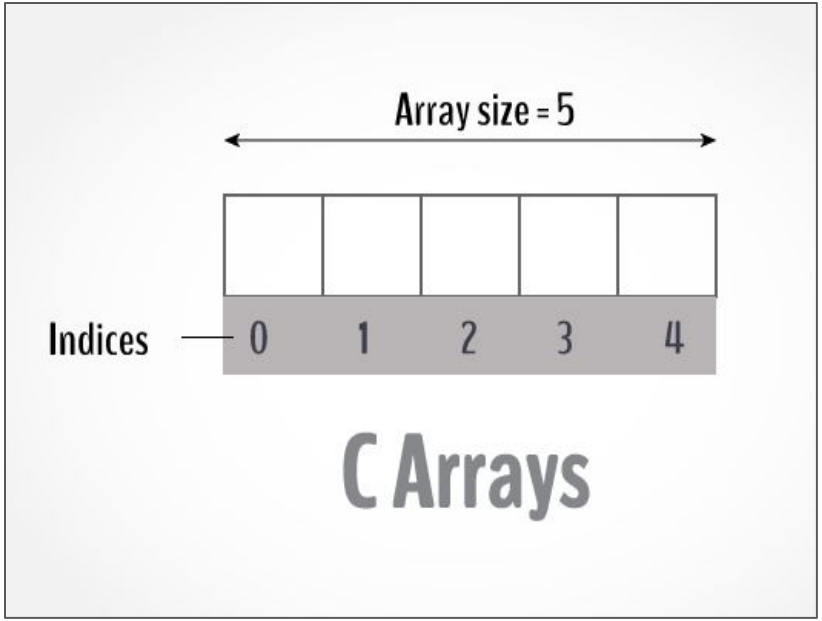
Apa itu array?

Butuh bahan bacaan? Bisa cek disini yaah~

Kenalan yuk dengan Array ...

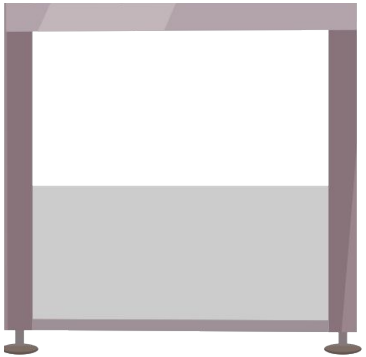
Kotlin menyediakan struktur data **array** yang dapat digunakan untuk **menyimpan kumpulan elemen secara urut dengan ukuran tetap dengan tipe data yang sama.**

Array biasa dipakai untuk menyimpan kumpulan data, namun sangat direkomendasikan memanfaatkan **array** sebagai kumpulan variabel dari tipe yang sama.

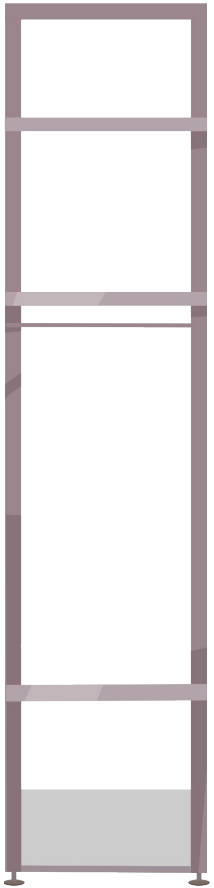


Untuk membantu kita semua lebih familiar dengan Array, sekarang coba pake analogi dulu yuk!

Kita ambil salah satu contoh dari tipe data yaitu int sebagai perbandingan ya.



Kalo kita mendeklarasikan variabel dengan tipe data **int**, kita itu seperti menyediakan lemari dengan 1 rak, seperti rak di samping ini.

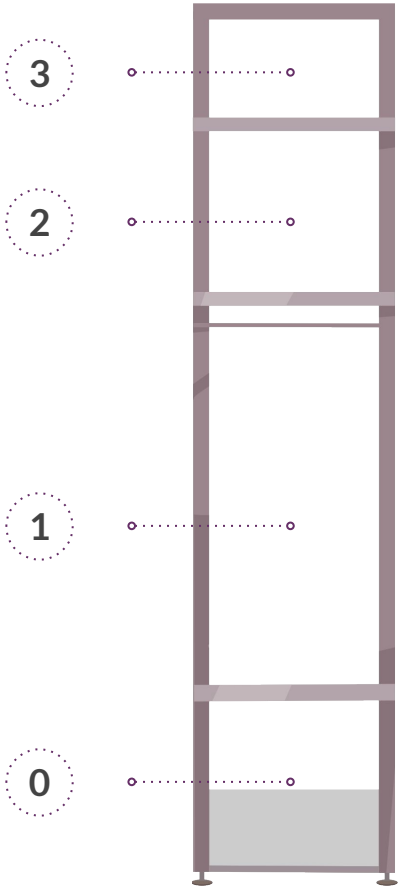


Kalau kita mendeklarasikan variabel menggunakan tipe data **array**, kita tuh kayak beli lemari yang punya beberapa rak seperti gambar di samping ini.

Array di samping ini punya 4 rak. Rak ini nantinya akan kita beri label.

Kalau kita mau kasih label angka ke **array**, harus inget kata mas-mas di pom bensin.

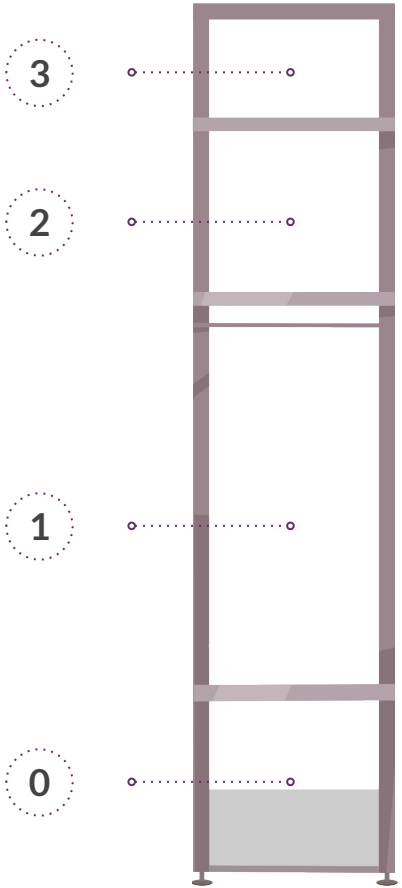
Dimulai dari 0 ya kak, bukan angka 1 seperti label pada umumnya.



Supaya lebih mudah, kita lihat ilustrasi rak di samping ini :

- Label 0 untuk rak ke 1
- Label 1 untuk rak ke 2
- Label 2 untuk rak ke 3
- Label 3 untuk rak ke 4

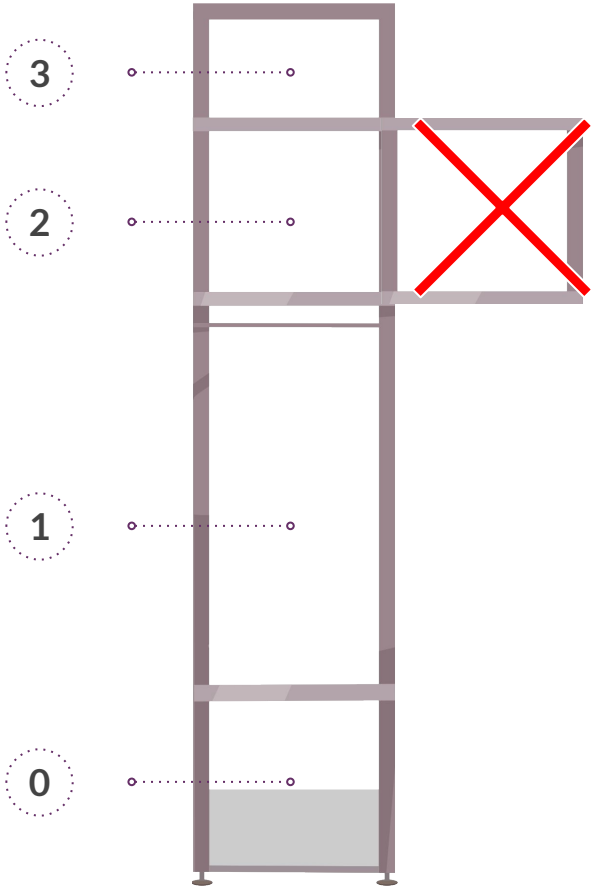
Nah, untuk penggunaan keempat rak ini ada aturannya nih ...



Pertama, Di lemari array ini kita cuma boleh menyimpan satu jenis barang aja.

Misalnya**;** Kalo kita mau nyimpen kaos di dalam lemari ini, kita cuma boleh nyimpen kaos aja, nggak boleh nyimpen sepatu atau celana.

Artinya, pas kita mau mendeklarasikan variabel dengan tipe data **array of int,** tiap elemen cuma boleh diisi sama data **int,** ga boleh lagi pake data **char, double,** dll.



Kedua, lemari array ini cuma bisa dibuat dengan desain rak yang jumlahnya 4 dan nggak bisa lebih dari itu.

Jadi, kalo kaos kita kebanyakan, nggak bisa nambah satu rak lagi buat nyimpen kaos-kaos itu.

Artinya, kalo kita mendeklarasikan variabel menggunakan tipe data **array of int** dan **length 4,** maka kita cuma punya 4 elemen:

- **Element index 0** ← untuk data **int.**
- **Element index 1** ← untuk data **int.**
- **Element index 2** ← untuk data **int.**
- **Element index 3** ← untuk data **int.**

Kita nggak bisa bikin elemen baru, misal **Element index 4.**

Kenal Array dan ArrayList yuk~

Kamu tahu nggak sih? Pada Kotlin, Array memiliki banyak saudara. Salah satunya **ArrayList** dan **Array<DataType>**

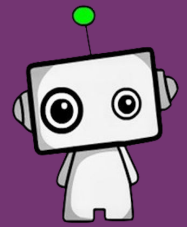
	Array	ArrayList
Definisi	**Array** adalah objek yang dibuat secara dinamis. Berfungsi sebagai wadah yang menampung jumlah nilai konstan dari jenis yang sama serta memiliki lokasi memori yang berdekatan.	ArrayList adalah kelas kerangka kerja **Collection** dari Java. Ini berisi kelas populer seperti **Vector**, **HashTable**, dan **HashMap**.
Resizeable	Array adalah struktur data dengan panjang tetap.	ArrayList adalah struktur data dengan **panjang variabel** yang bisa diubah ukurannya sendiri saat dibutuhkan.
Primitive/ Generic type	Array dapat menyimpan **objek** dan **tipe primitif**.	Kami tidak dapat menyimpan tipe **primitif** di ArrayList. Secara otomatis mengubah tipe primitif menjadi objek.
Static/Dynamic Size	Array berukuran **statis**.	ArrayList berukuran **dinamis**.

Saatnya kita

Quiz!

BINAR

3. **Kode nama robot yang di produksi oleh pabrik kita adalah `["BARBATOS", "RX-78-2", "BANSHEE", "STRIKE FREEDOM"]` kode nama robot pada index element ke 2 adalah ....**



A. "BANSHEE"

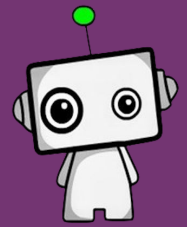
B. "RX-78-2"

C. “BARBATOS”

D. “STRIKE FREEDOM”

BINAR

3. **Kode nama robot yang di produksi oleh pabrik kita adalah `["BARBATOS", "RX-78-2", "BANSHEE", "STRIKE FREEDOM"]` kode nama robot pada index element ke 2 adalah ....**



A. **"BANSHEE"**

B. **"RX-78-2"**

C. **“BARBATOS”**

D. **“STRIKE FREEDOM”**

Integer ini pas banget dipakai buat tipe data **bilangan bulat** karena ukuran (bit) dari integer sendiri adalah **32**.

Element index dibaca dari yang ke 0

- element index 0 = BARBATOS
- element index 1 = RX-78-2
- element index 2 = BANSHEE
- element index 3 = STRIKE FREEDOM

Kita sudah kenalan jauh nih dengan **Array**.

Terus, cara deklarasi **Array** di Kotlinnya bagaimana kah?

Begini nih cara deklarasi Array ...

Untuk membuat array kosong dengan deklarasi ukuran :

```
tipeVariable namaVariable = Array(arrayLength)
```

atau

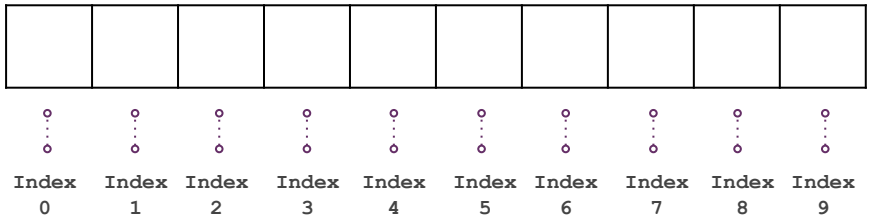
Kita juga bisa menggunakan **library function** untuk membuat array dengan lebih mudah:

```
tipeVariable namaVariable = arrayOf(arrayValue)
```

Kita juga dapat mendeklarasikan tipe data Array secara global tanpa tipe data seperti berikut:

```
val mArray = Array(10)
```

Deklarasi di atas berarti :
Kita punya variabel dengan tipe data **array** dan memiliki **length of element** 10.



Selain cara tadi, kita juga bisa mendeklarasikan Array dengan mudah menggunakan **Library Function**.

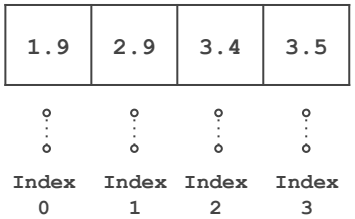
Sebelumnya, apa itu **Library function**?

Apa sih Library Function itu?

Library function merupakan kumpulan fungsi bawaan dari Kotlin.

Dengan menggunakan library function pada array, kita dapat membuat array dengan cara yang sederhana. Secara general, kita dapat membuat Array menggunakan **arrayOf** seperti berikut:

```
val mArray = arrayOf(1.9, 2.9, 3.4, 3.5)
```



Deklarasi di atas berarti:

Kita punya variabel dengan tipe **array of** dan punya **length of element 4** yang setiap elemennya udah diisi sama **value**.

Sebelumnya kita mendeklarasikan array menggunakan library function secara global.

Kita juga dapat mendeklarasikannya menggunakan tipe data primitif dengan memanfaatkan beberapa fungsi berikut:

- `intArrayOf()` : `IntArray`
- `booleanArrayOf()` : `BooleanArray`
- `charArrayOf()` : `CharArray`
- `longArrayOf()` : `LongArray`
- `shortArrayOf()` : `ShortArray`
- `byteArrayOf()` : `ByteArray`

Jika kita ingin membuat Array yang hanya bisa dimasukkan nilai dengan tipe data Integer, kita dapat gunakan **intArrayOf()**.

Untuk tipe data lain, bisa coba deklarasi sendiri yaa....

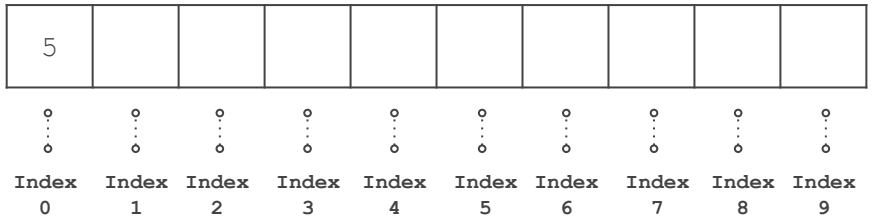
```
val mArray = intArrayOf(1.9, 2.9, 3.4, 3.5)
val mArray = booleanArrayOf(.....)
val mArray = charArrayOf(.....)
val mArray = longArrayOf(.....)
val mArray = shortArrayOf(.....)
val mArray = byteArrayOf(.....)
```

Gampang kan, cara deklarasi Array?

Nah, kalau sudah dideklarasikan, gimana caranya kita isi array tersebut dengan elemen?

Begini nih, cara mengisi Elemen dengan Array

Cara mengisi array kosong yang sudah ditentukan ukurannya, sangat mudah dengan Kotlin. Caranya bisa dilihat pada gambar disamping :




```kotlin
fun main() {
    val vArray = Array(10)
    vArray[0] = 5

    println("Array: ${vArray[0]}")
}
```

Mengisi elemen index ke 0 dengan value 5

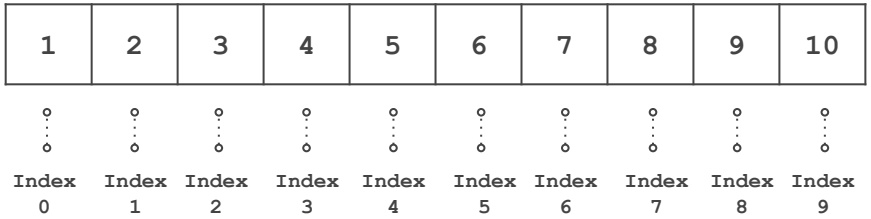
Untuk mengisi array sesuai dengan ukurannya secara cepat, kita dapat membuatnya seperti di bawah ini.

Caranya bisa cek gambar disamping

Mengisi elemen index ke i dengan value i

```
fun main() {
    val vArray = Array(10){ 1 * (it + 1) }

    for (i in vArray.indices){
        println("Array index ke $i : ${vArray[i]}")
    }
}
```

For merupakan ekspresi loop yang akan kita pelajari di topik berikutnya

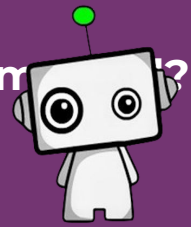


Saatnya kita

Quiz!

BINAR

4. Dari syntax berikut ini, kira-kira apakah output yang akan m[illegible]?



```
fun manggilArray() {
    val modelRobot = arrayOf("Transformer", "Gundam", "Autobot", "Mega Zord")
    modelRobot[1] = "Genesis"
    println(modelRobot[1])
}
```

A. Genesis

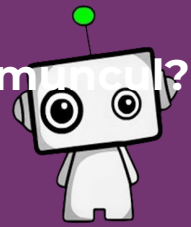
B. Gundam

C. Autobot

D. Mega Zord

BINAR

4. Dari syntax berikut ini, kira-kira apakah output yang akan muncul?



```
fun manggilArray() {
    val modelRobot = arrayOf("Transformer", "Gundam", "Autobot", "Mega Zord")
    modelRobot[1] = "Genesis"
    println(modelRobot[1])
}
```

A. Genesis
B. Gundam
C. Autobot
D. Mega Zord

Jika kita urut dengan benar, maka posisi awalnya akan seperti berikut:

- index 0 = Transformer
- index 1 = Gundam
- index 2 = Autobot
- index 3 = Mega Zord

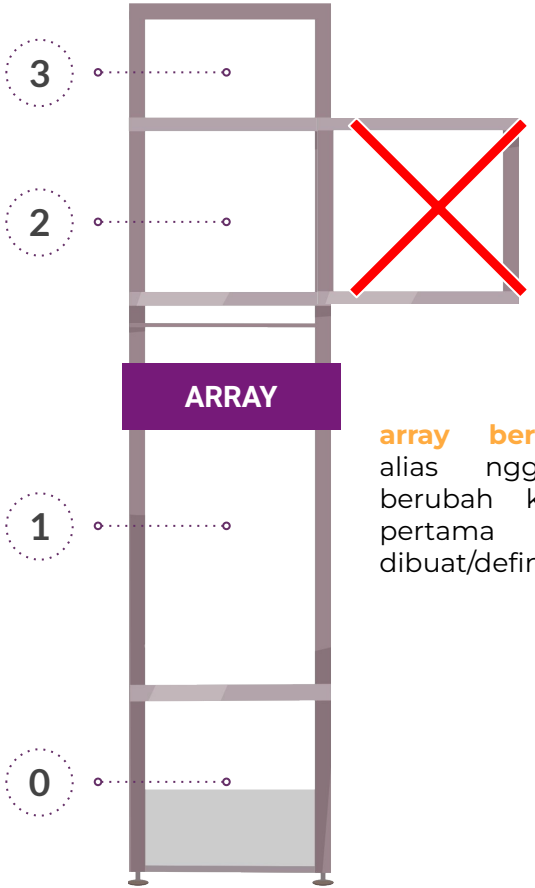
Setelah deklarasi `modelRobot[1] = "Genesis"` dimasukkan

- maka hasilnya akan seperti berikut:
- index 0 = Transformer
- index 1 = Genesis
- index 2 = Autobot
- index 3 = Mega Zord

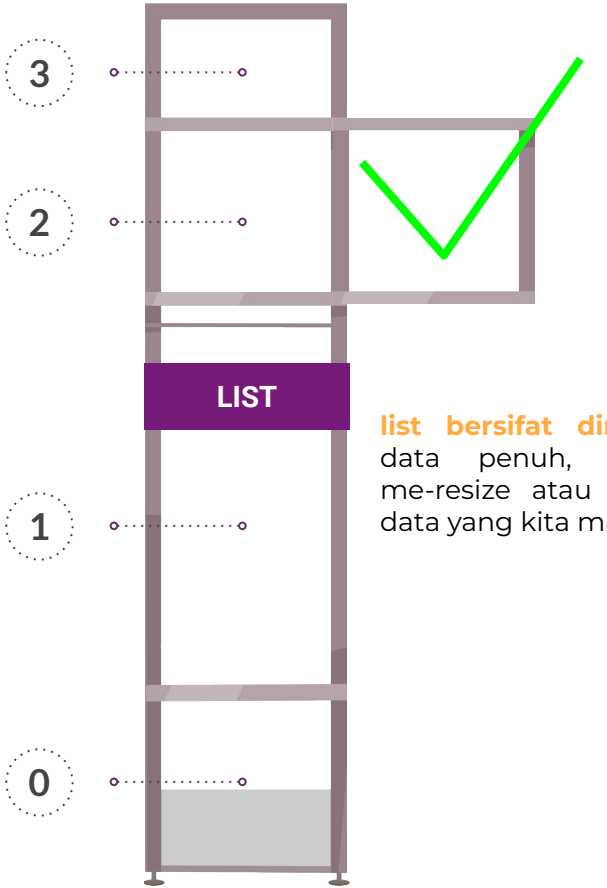
Nice! Sekarang kita bakal belajar tentang list dalam Kotlin!

Apa ya bedanya array sama list?

Kamu boleh cek link berikut untuk referensi lengkapnya~



array bersifat statis, alias nggak bisa berubah kayak waktu pertama kali dibuat/definisikan.



list bersifat dinamis, ketika data penuh, list ini bisa me-resize atau menyesuaikan data yang kita masukin.

Pada **array**, kita **nggak bisa** menambah, apalagi menghapus elemen.

Pada **list**, kita **bisa** menambah, atau menghapus elemen.

So, Gimana cara Deklarasinya di Kotlin?

Kita bahas sedikit tentang jenis list nih...

Pada penjelasan sebelumnya list dan array memiliki konsep yang mirip.

Di dalam list, terdapat 2 jenis list yang memiliki sifat yang berbeda. Yang bisa dilihat di sebelah kanan

Tentunya kita menggunakan tipe list yang sesuai dengan kebutuhan kita.

Immutable list : read-only, seperti namanya kita hanya dapat membaca datanya saja kita tidak bisa menambah dan menghapuskan datanya

```
val mList = listOf(1, 'x', false, "Binarian")
```

Mutable list : dapat merubah data

```
val mList = mutableListOf(1, 'x', false, "Binarian")
```

Memasukkan elemen baru ke dalam mutable list di Kotlin

Untuk memasukkan data pada elemen, kita tidak wajib memasukkan index-nya lagi.

Data yang tampil akan seperti berikut:

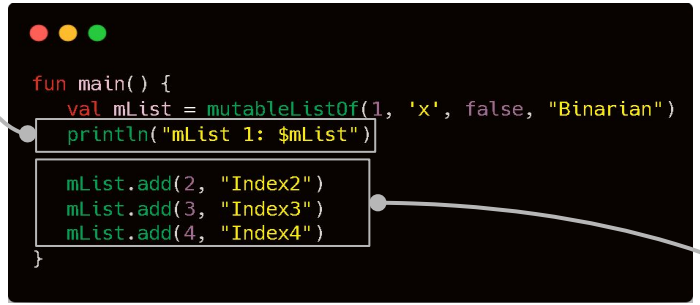
mList 1: [1, x, false, Binarian]

Syntax berikut akan menambahkan data di akhir List:

mList 2: [1, x, false, Binarian, Vijay]

Memasukkan elemen pada index tertentu ke dalam list di Kotlin

Untuk memasukkan data pada index tertentu caranya seperti pada kode disamping.



Data yang tampil akan seperti berikut:

mList 1: [1, x, false, Binarian]

Syntax berikut akan menambahkan data sesuai dengan index yang diinginkan:

mList 3: [1, x, Index2, Index3, Index4, false, Binarian, Vijay]

Mendapatkan elemen pada index tertentu dari list di Kotlin

```
fun main() {
    val mList = mutableListOf(1, 'x', false, "Binarian")
    println("mList 1: $mList")

    mList.add(2, "Index2")
    mList.add(3, "Index3")
    mList.add(4, "Index4")

    println("mList 4: ${mList[3]}")
}
```

Data yang tampil akan seperti berikut:

mList 1: [1, x, false, Binarian]

Data yang ditampilkan adalah:

Index3

Menghapus elemen dengan spesifik value pada list di Kotlin

```
fun main() {
    val mList = mutableListOf(1, 'x', false, "Binarian")
    println("mList 1: $mList")

    mList.remove(false)
}
```

Data yang tampil akan seperti berikut:
mList 1: [1, x, false, Binarian]

Syntax berikut akan menghapuskan data dengan value false:
mList 2: [1, x, Binarian]

Menghapus elemen dengan spesifik index pada list di Kotlin

```
fun main() {
    val mList = mutableListOf(1, 'x', false, "Binarian")
    println("mList 1: $mList")

    mList.removeAt(1)
}
```

Data yang tampil akan seperti berikut:
mList 1: [1, x, false, Binarian]

Syntax berikut akan menghapuskan data dengan index tertentu:
mList 2:[1, false, Binarian]

Menghapus semua elemen pada list di Kotlin

```
fun main() {
    val mList = mutableListOf(1, 'x', false, "Binarian")
    println("mList 1: $mList")

    mList.removeAll { true }
}
```

Data yang tampil akan seperti berikut:
mList 1: [1, x, false, Binarian]

Syntax berikut akan menghapuskan semua data pada list

Mendapatkan ukuran pada list di Kotlin

```
fun main() {
    val mList = mutableListOf(1, 'x', false, "Binarian")
    println("mList 1: $mList")

    println(mList.size)
}
```

Data yang tampil akan seperti berikut:

mList 1: [1, x, false, Binarian]

Syntax berikut akan menampilkan ukuran pada list

Mengecek apakah ada sebuah elemen tertentu pada list di Kotlin

```
fun main() {
    val mList = mutableListOf(1, 'x', false, "Binarian")
    println("mList 1: $mList")

    println(mList.contains('x'))
}
```

Data yang tampil akan seperti berikut:
mList 1: [1, x, false, Binarian]

Syntax berikut akan menampilkan value yang cocok dengan yang ada di list

Melihat index dari sebuah elemen pada list di Kotlin

```
fun main() {
    val mList = mutableListOf(1, 'x', false, "Binarian")
    println("mList 1: $mList")

    println(mList.indexOf('x'))
}
```

Data yang tampil akan seperti berikut:
mList 1: [1, x, false, Binarian]

Syntax berikut akan menampilkan index dengan sebuah elemen ‘x’ pada list

Melihat index terakhir sebuah elemen pada list di Kotlin

```
fun main() {
  val mList = mutableListOf(
     1, 'x', false, "Binarian",'x'
   )

   println("mList 1: $mList")

   println(mList.lastIndexOf('x'))
}
```

Data yang tampil akan seperti berikut:
mList 1: [1, x, false, Binarian, x]

Syntax berikut akan menampilkan index terakhir untuk elemen ‘x’ pada list

Melihat index pertama pada list di Kotlin

```
fun main() {
    val mList = mutableListOf("Adam", "Idris", "Nuh")
    println("mList 1: $mList")

    println(mList.first())
}
```

Data yang tampil akan seperti berikut:
mList 1: [1, x, false, Binarian]

Syntax berikut akan menampilkan index pertama pada list

Melihat index terakhir pada list di Kotlin

```
fun main() {
    val mList = mutableListOf(1, 'x', false, "Binarian")
    println("mList 1: $mList")

    println(mList.last())
}
```

Data yang tampil akan seperti berikut:
mList 1: [1, x, false, Binarian]

Syntax berikut akan menampilkan index terakhir pada list

Saatnya kita

Quiz!

BINAR

5. **Kode berikut bertujuan agar data list dapat diisi ulang. Lengkapilah syntax yang hilang tersebut pada titik-titik!**

```
fun quiz() {
    val robot = ……<String>()
    robot.add("Barbatos")
    robot.add("RX 78-2")
    robot.add("Strike Freedom")

    println(robot)
}
```

A. mutableListOf
B. add()
C. List
D. Var

BINAR

5. **Kode berikut bertujuan agar data list dapat diisi ulang. Lengkapilah syntax yang hilang tersebut pada titik-titik!**

```
fun quiz() {
    val robot = ……<String>()
    robot.add("Barbatos")
    robot.add("RX 78-2")
    robot.add("Strike Freedom")

    println(robot)
}
```

A. **mutableListOf**

B. add()

C. List

D. Var

Dari aturan deklarasi **List** yang dapat berulang kali diisi pada Kotlin adalah

`tipeVariable namaVariable = tipeList<tipeData>()`

dan juga model -model baru dapat dimasukan ke dalam list tersebut.

Referensi dan bacaan lebih lanjut~

1. Android Developer should know these Data Structures for Next Interview
2. Kotlin Variables and Basic Types
3. Basic types
4. Array - Kotlin Programming

Penutup

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

Selanjutnya, kita bakal berkenalan lebih lanjut dengan Bahasa Si Android, yaitu Kotlin.

Penasaran kayak gimana? Cus langsung ke topik selanjutnya~

Terima Kasih!

Next Topic

loading...

BINAR