



# **Kotlin Dasar**

Disusun oleh: Andi Hasan Ashari

Lisensi:

~ Modul ini boleh digunakan secara pribadi/ disebar luaskan secara non-komersial

~ Dilarang mengubah/ menghapus sumber & nama penyusun modul ini

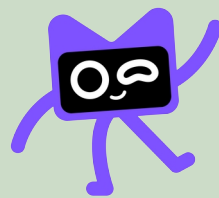
Contact:

Email: [andihasanelfalakiy@gmail.com](mailto:andihasanelfalakiy@gmail.com)

Telegram: [t.me/moonelfalakiy](https://t.me/moonelfalakiy)

Instagram: [@andihasan\\_kotlin](https://www.instagram.com/andihasan_kotlin)

YouTube: [@moonlightstudio](https://youtube.com/@moonlightstudio)



Publikasi: Pertama Mei 2023

## Pengenalan Bahasa Kotlin

- Proyek mulai 2010, nama Kotlin -> pulau Rusia dekat tim JetBrains, v 1.8.21
- Statically type, berjalan di atas JVM
- Rilis 2016 v 1.0, Google I/O 2017 Official Language for Android
- Open Source, Repository: <https://github.com/jetbrains/kotlin>
- Multiplatform, desktop app, web, and iOS, data science, machine learning
- Multiparadigma (Functional Programming, OOP) & mudah dipelajari



## Keuntungan menggunakan bahasa Kotlin untuk pengembangan aplikasi Android:

- Compatibility: JDK 6, android older
- Performance: bytecode = java, inline function -> kode lambda lebih cepat
- Interoperability: java & kotlin, java from kotlin, kotlin from java
- Compilation Time: mendukung inkremental -> efisien, sama/ lebih cepat dari java
- Concise: ringkas

Java:

```
public class SomeClasses{  
    public static void main(String[] args){  
        System.out.println("Hello, World!");  
    }  
}
```

Kotlin:

```
fun main() {  
    println("Hello, World!")  
}
```

- Tools friendly: dukungan tools yang memadai
- Tidak perlu titik koma/ semicolon
- Otomatis deteksi tipe data
- Lebih aman dari NPE (NullPointerException)

## Program Hello World



```
fun main() {  
    println("Hello, World!")  
}
```

- Kebiasaan/ adat programmer Kotlin, menggunakan camelCase saat menulis nama function dan variabel. Format camelCase: camelCaseFormat

\*akan terasa manfaatnya ketika menggunakan viewBinding

- Sedangkan nama class biasanya diawali huruf Kapital untuk setiap kata.

Contoh: NamaClass

Komentar di Kotlin:

- // (Double garis miring) untuk komentar per baris

- /\*

komentar

Ini juga komentar

\*/

Untuk komentar multi baris

# Variabel

## - Struktur variabel

```
val namaVariabel: Tipe = inisialisasi
```

- **val** untuk variabel yang nilainya tidak bisa diubah/ immutable
- **var** untuk variabel yang nilainya bisa diubah/ mutable
- Variabel di kotlin bersifat case sensitive

val:

```
fun main() {  
    val nilai: Int = 3  
    nilai = 7  
    println(nilai)  
}  
  
// ERROR: main/kotlin/Main.kt (5:5) Val cannot be  
reassigned
```

var:

```
fun main() {  
    var nilai: Int = 3  
    nilai = 7  
    println(nilai)  
}  
  
// 7
```



## Tipe Data String

- Penulisan String diapit tanda petik ganda " " (double quote) untuk satu baris
- Keyword String

```
fun main() {  
    val salam: String = "selamat datang"  
  
    println(salam)  
}  
// selamat datang
```

- String dengan tiga petik ganda """" """" untuk multi baris

```
fun main() {  
    val salam: String = """  
    hello  
    selamat  
    datang  
    """  
  
    println(salam)  
}  
/*  
hello  
selamat  
datang  
*/
```

## Fitur Pintar Kotlin

- Tanpa menulis tipe data/ otomatis deteksi tipe data

```
fun main() {  
    val salam = "selamat datang"  
  
    println(salam)  
}  
// selamat datang
```

- Mengecek tipe data dengan operator `is` & `::class.simpleName`

```
fun main() {  
  
    val salam = "Selamat datang"  
    println(salam is String)  
    println(salam::class.simpleName)  
  
}  
// true  
// String
```

## Tipe Data Char (Karakter)

- Penulisan karakter diapit tanda petik tunggal ' ' (single quote)
- Tipe data Char hanya menerima 1 karakter tunggal
- Keyword Char

```
fun main() {  
    val karakter: Char = 'A'  
  
    println(karakter)  
}  
// A
```

## Tipe Data Boolean

- Tipe data yang bernilai true & false
- Keyword Boolean

```
fun main() {  
    val benar: Boolean = 1 < 7  
    val salah: Boolean = 1 > 7  
  
    println(benar)  
    println(salah)  
}  
// true  
// false
```

# Tipe Data Number

## Bilangan Bulat:

1. Byte (8 bit)
2. Short (16 bit)
3. Int (32 bit)
4. Long (64 bit)

- penambahan L diakhir nilai, ketika nilai Long masuk di rentang nilai Integer

```
fun main() {  
  
    val umur: Byte = 25  
    val jumlah: Short = 1000  
    val total: Int = 10000  
    val semua: Long = 1000000L  
  
    println(umur)  
    println(jumlah)  
    println(total)  
    println(semua)  
}  
/*  
25  
1000  
10000  
10000000  
*/
```

## Bilangan Pecahan Desimal:

### 1. Float (32 bit)

- diakhir nilai harus ditambah F atau f, jika tidak maka akan dianggap tipe data Double

### 2. Double (64 bit)

```
fun main() {  
    val kecil: Float = 127.9281F  
    val besar: Double = 19282910.1101  
  
    println(kecil)  
    println(besar)  
}  
/*  
127.9281  
1.92829101101E7  
*/
```

## Rentang nilai tipe data number:

- mengecek nilai minimal & maksimal tipe data dengan `Type.MIN_VALUE` untuk mengecek nilai minimal, `Type.MAX_VALUE` untuk mengecek nilai maksimal tipe data

### 1. Byte

```
fun main() {  
    val nilaiMin: Byte = Byte.MIN_VALUE  
    val nilaiMax: Byte = Byte.MAX_VALUE  
  
    println(nilaiMin)  
    println(nilaiMax)  
}  
// -128  
// 127
```

### 2. Short

```
fun main() {  
    val nilaiMin: Short = Short.MIN_VALUE  
    val nilaiMax: Short = Short.MAX_VALUE  
  
    println(nilaiMin)  
    println(nilaiMax)  
}  
// -32768  
// 32767
```

### 3. Int

```
fun main() {  
  
    val nilaiMin: Int = Int.MIN_VALUE  
    val nilaiMax: Int = Int.MAX_VALUE  
  
    println(nilaiMin)  
    println(nilaiMax)  
}  
// -2147483648  
// 2147483647
```

### 4. Long

```
fun main() {  
  
    val nilaiMin: Long = Long.MIN_VALUE  
    val nilaiMax: Long = Long.MAX_VALUE  
  
    println(nilaiMin)  
    println(nilaiMax)  
}  
// -9223372036854775808  
// 9223372036854775807
```



## 5. Float

```
fun main() {  
  
    val nilaiMin: Float = Float.MIN_VALUE  
    val nilaiMax: Float = Float.MAX_VALUE  
  
    println(nilaiMin)  
    println(nilaiMax)  
}  
// 1.4E-45  
// 3.4028235E38
```

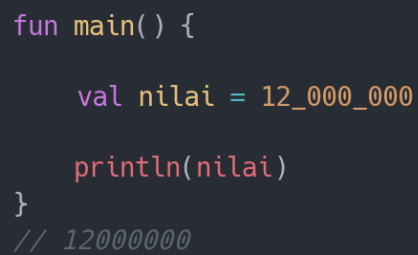
## 6. Double

```
fun main() {  
  
    val nilaiMin: Double = Double.MIN_VALUE  
    val nilaiMax: Double = Double.MAX_VALUE  
  
    println(nilaiMin)  
    println(nilaiMax)  
}  
// 4.9E-324  
// 1.7976931348623157E308
```

## Underscore & Conversion

### Underscore:

- untuk memudahkan pembacaan bilangan



```
fun main() {  
    val nilai = 12_000_000  
    println(nilai)  
}  
// 12000000
```

### Conversion:

- mengubah tipe data ke tipe data lain

toByte(): Byte

toShort(): Short

toInt(): Int

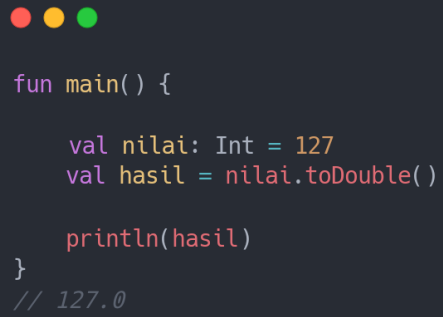
toLong(): Long

toFloat(): Float

toDouble(): Double

toChar(): Char

toString(): String



```
fun main() {  
    val nilai: Int = 127  
    val hasil = nilai.toDouble()  
  
    println(hasil)  
}  
// 127.0
```

## Tipe Data Array (Larik)

- Kumpulan objek/ data dengan tipe data yang sama atau berbeda, tergantung penulisannya
- Data dalam array dapat diakses dengan index
- Index di array dimulai dari 0
- ada 3 cara mendeklarasikan array

```
fun main() {  
    // array 1 explicit type  
    val array1: Array<String> = arrayOf("hai", "good", "mornig")  
  
    // array 2 implicit type  
    val array2 = arrayOf(12, 17.3, "hello", true)  
    // variabel array2 tipe data Array saja  
    // beda saat mengakses isi arraynya tetap sesuai tipe data masing-masing  
  
    // array 3  
    val array3 = intArrayOf(1, 2, 3, 4, 5)  
  
    // mendapatkan data isi array  
    println(array2[1]) // 17.3  
  
    // atau  
    println(array2.get(1)) // 17.3  
  
    // mengubah data isi array  
    array2[1] = 11.7  
  
    // atau  
    array2.set(1, 11.7)  
  
    println(array2[1]) // 11.7  
  
    // mencetak isi array sekaligus  
    array3.forEach {println(it)}  
    /*  
    1  
    2  
    3  
    4  
    5  
    */  
}
```

## Tipe Data Range

- Range termasuk tipe data yang unik, karena kita bisa menentukan nilai awal & akhir
- Ada beberapa cara mendeklarasikan range

### Range dengan operator ..

```
fun main() {  
    // mendeklarasikan range dengan operator ..  
    val range = 0..10  
  
    range.forEach {  
        println(it)  
    }  
}  
/*  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
*/
```

## Range dengan step

```
fun main() {  
  
    // mendeklarasikan range dengan step  
  
    val range = 0..10 step 2  
  
    range.forEach {  
        println(it)  
    }  
}  
/*  
    0  
    2  
    4  
    6  
    8  
    10  
*/
```

## Range dengan rangeTo()

```
fun main() {  
  
    // mendeklarasikan range dengan rangeTo()  
    val range = 0.rangeTo(10)  
  
    range.forEach {  
        println(it)  
    }  
}  
/*  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
*/
```

## Range dengan downTo()

```
fun main() {  
  
    // mendeklarasikan range dengan downTo()  
    val range = 10.downTo(0)  
  
    range.forEach {  
        println(it)  
    }  
}  
/*  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1  
0  
*/
```

## Range karakter

```
fun main() {  
    // mendeklarasikan range karakter  
    val range = 'A'..'F'  
  
    range.forEach {  
        println(it)  
    }  
}  
/*  
    A  
    B  
    C  
    D  
    E  
    F  
*/
```



## Operator Matematika

- Kotlin mendukung operasi matematika, diantaranya:

+ (Pertambahan)

- (Pengurangan)

\* (Perkalian)

/ (Pembagian)

% (Sisa bagi)

```
fun main() {  
  
    // pertambahan  
    val tambah = 2 + 10  
  
    // pengurangan  
    val kurang = 10 - 3  
  
    // perkalian  
    val kali = 5 * 2  
  
    // pembagian  
    /*  
        perlu diingat saat nilai Integer dibagi,  
        maka sisa pembagian nya akan diabaikan/ dihilangkan  
        seperti 10 / 3 = 3  
        jika ingin operasi normal, kita perlu menetapkan nya  
        ke tipe data Double, cara termudah .0  
        seperti 10.0 / 3.0 = 3.3333333333333335  
    */  
    val bagi = 10 / 3 // 3  
  
    // sisa bagi  
    val sisa = 11 % 2  
  
    println(tambah)  
    println(kurang)  
    println(kali)  
    println(bagi)  
    println(sisa)  
  
    /*  
        12  
        7  
        10  
        3  
        1  
    */  
  
}
```

# Operator Augmented Assignment

Augmented Assignment Operator		
Biasa	A Assignment	Fungsi
a = a + b	a += b	a.plusAssign(b)
a = a - b	a -= b	a.minusAssign(b)
a = a * b	a *= b	a.timesAssign(b)
a = a / b	a /= b	a.divAssign(b)
a = a % b	a %= b	a.modAssign(b)

= merupakan Assignment Operator

```
fun main() {  
    // augmented assignment  
    var total = 0  
  
    val sampo = 100  
    val sikat = 150  
    val sabun = 200  
  
    total += sampo // total = total + sampo  
    total += sikat // total = total + sikat  
    total += sabun // total = total + sabun  
  
    println(total)  
  
    /*  
       450  
    */  
}
```

# Unary Prefix, Increment & Decrement

Unary prefix & Increment/decrement

Operator	Arti
+	Unary Plus
-	Unary Minus
!	Not
++	Increment
--	Decrement

```
fun main() {  
    // not  
    var maju = true  
    println(!maju) // false  
  
    // increment & decrement  
    var age = 10  
    age++  
    println(age) // 11  
  
    var suhu = 0  
    suhu--  
    println(suhu) // -1  
}
```

## Comparison & Equality Operator

Comparison & Equality Operator		
Operator	Arti	Expresi
>	Lebih dari	$a > b$
<	Kurang dari	$a < b$
>=	Lebih atau sama dengan dari	$a \geq b$
<=	Kurang atau sama dengan dari	$a \leq b$
==	Sama dengan	$a == b$
!=	Tidak sama dengan	$a != b$

```
fun main() {  
  
    val a = 7  
    val b = 17  
  
    val c = 7  
  
    if (c >= a) {  
        println("benar")  
    } else {  
        println("salah")  
    }  
  
}
```

# Logical Operator

Logical Operator	
Operator	Arti
&&	AND
	OR
!	NOT

- && akan mengembalikan nilai true, jika kedua ekspresi bernilai benar
- || akan mengembalikan nilai true, jika salah satu ekspresi bernilai benar
- ! berbeda dengan kedua operator diatas, ! akan melakukan negasi/ membalik nilai hasil evaluasi

```
fun main() {  
    // &&  
  
    val buka = 7  
    val tutup = 17  
  
    val sekarang = 9  
  
    val status = if (sekarang >= buka && sekarang <= tutup) {  
        true  
    } else {  
        false  
    }  
  
    println("apakah toko buka? $status")  
    // apakah toko buka? true  
}
```

```
fun main() {  
  
    // !!  
  
    val buka = 7  
    val tutup = 17  
  
    val sekarang = 3  
  
    val status = if (sekarang < buka || sekarang > tutup) {  
        true  
    } else {  
        false  
    }  
  
    println("apakah toko tutup? $status")  
    // apakah toko tutup? true  
  
}
```

```
fun main() {  
  
    // !  
  
    val buka = 7  
    val sekarang = 9  
  
    val status = sekarang > buka  
  
    if (!status) {  
        println("toko buka")  
    } else {  
        println("toko tutup")  
    }  
    // toko tutup  
  
}
```

## Operator in

- in digunakan untuk mengecek apakah suatu elemen data ada di dalam suatu koleksi/ collection misal array, range dll

```
fun main() {  
    // in  
    val a = 7..17  
    val c = 7  
    if (c in a) {  
        println("benar")  
    } else {  
        println("salah")  
    }  
    // benar  
  
}
```

## IF Expression

- If else digunakan untuk percabangan alur program dengan menetapkan kondisi tertentu
- If else dieksekusi secara berurutan, jika kondisi if bernilai true, maka blok if akan dieksekusi, jika bernilai false maka blok else yang akan dieksekusi

```
fun main() {  
  
    val nilai = 93  
  
    if (nilai > 90) {  
        println("Hebat")  
    } else {  
        println("Lumayan")  
    }  
}  
  
// Hebat
```

```
fun main() {  
  
    val nilai = 93  
  
    var hasil =  
        if (nilai > 90) {  
            "Hebat"  
        } else {  
            "Lumayan"  
        }  
  
    println(hasil)  
}  
  
// Hebat
```



## Else if

- Else If digunakan untuk membuat percabangan lebih banyak (kondisi lebih dari 2)
- Diletakkan di antara If dan else

```
fun main() {  
  
    val nilai = 87  
  
    if (nilai > 90) {  
        println("Hebat")  
    } else if (nilai > 70) {  
        println("Baik")  
    } else {  
        println("Lumayan")  
    }  
}  
  
// Baik
```

## Nullability

- NPE NullPointerException: sebuah kesalahan saat mengakses/ mengelola nilai dari variable yang bernilai null/ belum diinisialisasi
- Secara default Kotlin tidak membolehkan data/ variabel bernilai null

```
fun main() {  
    // awas! bedakan nol dengan null  
    val nilai1: Int = 0  
    val nilai2: Int = null // compile error  
  
    println(nilai1)  
    println(nilai2)  
  
    /*  
        ERROR: Null can not be a value of  
        a non-null type Int  
    */  
}
```

- Jika ingin menjadikan variabel bisa bernilai null, kita bisa menggunakan tanda ? (Tanya) dibelakang tipe datanya

```
fun main() {  
    // penanganan null dengan ? setelah Tipe data  
    val nilai2: Int? = null  
    println(nilai2)  
    /*  
       null  
    */  
}
```

- saat kita menjadikan variabel sebagai nullability, kita tidak bisa langsung mengakses/ mengelola properti/ metod nya, contoh

```
fun main() {  
    val nilai: String? = "kotlin"  
    println(nilai.length) // error  
}
```

- Untuk bisa mengakses/ mengelola properti/ metod nya, kita bisa mengecek nya dengan if

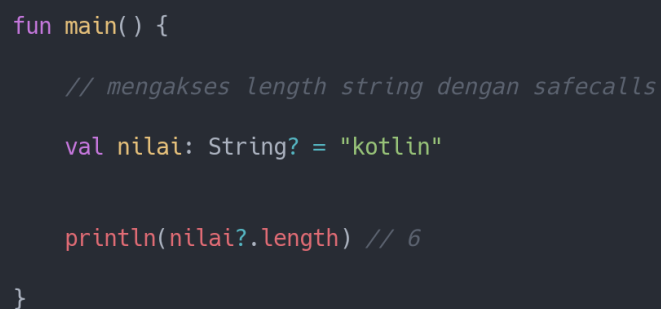
```
fun main() {  
    // mengakses length string dengan if  
    val nilai: String? = "kotlin"  
  
    if (nilai != null) {  
        println(nilai.length) // 6  
    }  
}
```

## Safe calls Operator

Safe call atau disebut juga null-safe operator adalah salah satu fitur penting di Kotlin yang memungkinkan kita untuk melakukan akses atau pemanggilan terhadap objek atau variabel yang mungkin bernilai null (nullability) tanpa menimbulkan exception atau kesalahan saat runtime.

Dalam Kotlin, operator safe call ditandai dengan simbol tanda tanya (?) yang ditempatkan setelah nama objek atau variabel yang ingin diakses.

Contoh:



```
fun main() {  
    // mengakses length string dengan safecalls  
    val nilai: String? = "kotlin"  
  
    println(nilai?.length) // 6  
}
```

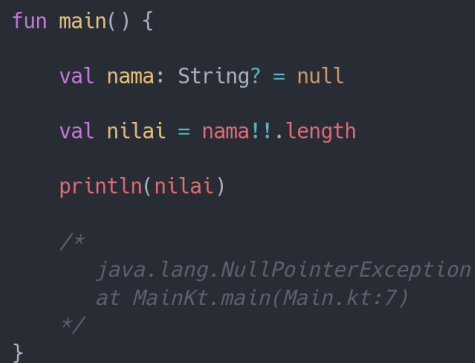
## Elvis operator

Elvis operator adalah operator yang digunakan untuk mengganti nilai null dengan nilai default jika suatu objek bernilai null. Ditandai dengan tanda ?: (Tanya Semicolon).

Operator ini memungkinkan kita untuk mengecek apakah sebuah variabel null dan memberikan nilai default jika null.

```
fun main() {  
  
    val nama: String? = null  
  
    // mengakses properti length variabel nama  
    val nilai = nama?.length ?: 9  
  
    println(nilai) // 9  
}
```

- Perhatikan penggunaan operator non-null assertion (!!). Menggunakan non-null assertion, compiler mengizinkan kita mengakses/ mengelola nilai sebuah objek nullable.



```
fun main() {  
    val nama: String? = null  
    val nilai = nama!!.length  
    println(nilai)  
  
    /*  
    java.lang.NullPointerException  
    at MainKt.main(Main.kt:7)  
    */  
}
```

- Namun penggunaan operator non-null assertion SANGAT TIDAK DISARANKAN karena akan memaksa sebuah objek menjadi non-null, sehingga saat objek tersebut bernilai null, kita tetap akan berjumpa error NullPointerException.

## String Template

Merupakan fitur pada Kotlin yang memungkinkan penggabungan nilai variabel atau ekspresi ke dalam sebuah string dengan cara yang lebih mudah dan efektif.

Dalam Kotlin, string template menggunakan tanda dollar (\$) untuk menandai ekspresi atau variabel yang ingin dimasukkan ke dalam string.

```
fun main() {  
    // concatenation (rangkaian)  
    val hari = "Senin"  
    val jam = 17  
    println("Sekarang hari " + hari + " pukul: " + jam)  
  
    // Sekarang hari Senin pukul: 17  
}
```

```
fun main() {  
    // string template  
    val hari = "Senin"  
    val jam = 17  
    println("Sekarang hari $hari pukul: $jam")  
  
    // Sekarang hari Senin pukul: 17  
}
```



- untuk menyisipkan expression/ function, gunakan `${}`, letakkan expression/ function didalam kurung kurawal.

```
fun main() {  
    // string template dengan expression  
    val hari = "Senin"  
    val jam = 17  
    println("Sekarang hari $hari pukul: $jam, toko sedang ${if (jam < 18) "buka" else  
    "tutup}")  
    // Sekarang hari Senin pukul: 17, toko sedang buka  
}
```

## **Apa yang selanjutnya harus dipelajari?**

- Kotlin Control Flow
- Kotlin Data Classes & Collection
- Kotlin Functional Programming
- Kotlin Object Oriented Programming
- Kotlin Generic
- Kotlin Coroutines

Terimakasih

**Andi Hasan Ashari**