



Function & Method

Silver- Chapter 1 - Topic 5

Selamat datang di **Chapter 1 Topic 5** online
course **Android Developer** dari
Binar Academy!





Haai Binarian 🙌

Masih di Chapter 1 nih~

Di Topik sebelumnya, kita sudah belajar tentang **Class & Object**.

Pada **Topik 5** ini, kita bakal belajar tentang **Function & Method**, dimana kita akan bermain lebih banyak dengan bahasa pemrograman yang kita pelajari sebelumnya.

Gimana guys, udah siaapp? 😊





Detailnya, kita bakal bahas hal-hal berikut ini :

- Memahami apa itu **function**
- Dapat membuat **function**
- Dapat **membuat function** yang lebih kompleks sesuai dengan kebutuhan





Apa itu **FUNCTION?**

Kamu bisa cek penjelasan mendalamnya
pada link [berikut](#) ini~





Apa sih, Function itu ?

Function merupakan blok kode yang ditulis untuk melakukan tugas tertentu.

Function didukung oleh semua bahasa pemrograman modern termasuk Kotlin dan juga dikenal sebagai **method** atau **subrutin**.

Pada tingkat yang lebih tinggi, suatu **function**, mengambil beberapa input tambahan yang disebut sebagai **parameter**, yang melakukan akhir tertentu pada input dan akhirnya dapat mengembalikan nilai.





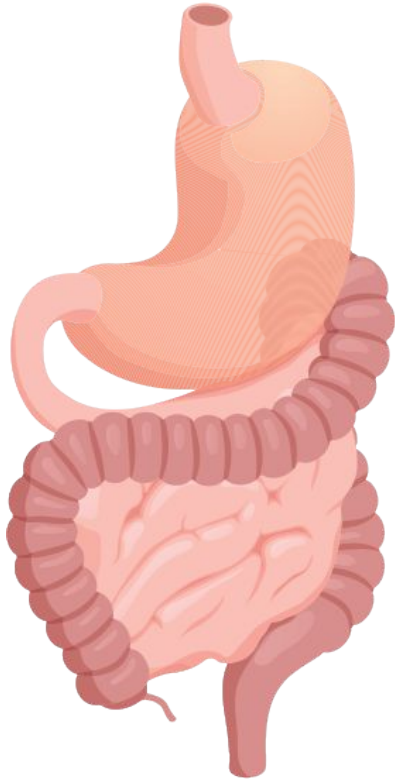
Kotlin adalah bahasa yang ditulis secara **status**.

Oleh karena itu **function** memainkan peran besar di dalamnya.

Sebenarnya, kita tuh harusnya akrab loh dengan **function**, karena kita biasa menggunakan **function** dalam kehidupan sehari-hari kita.

Biar paham, kita coba main analogi lagi yuk~

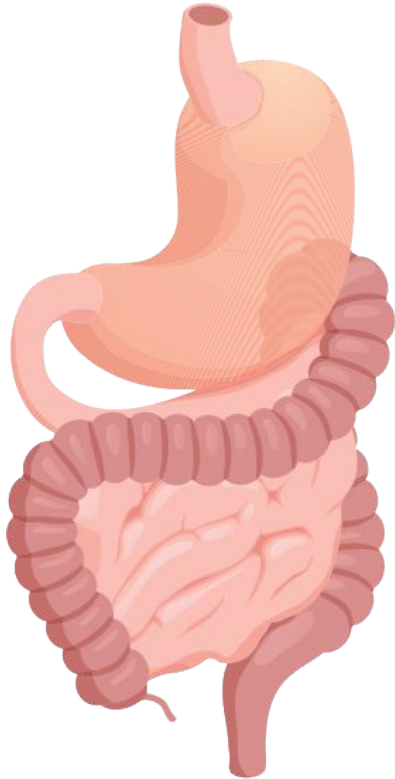




Coba Kamu Bayangkan Sistem Pencernaan dalam Diri Kamu ...

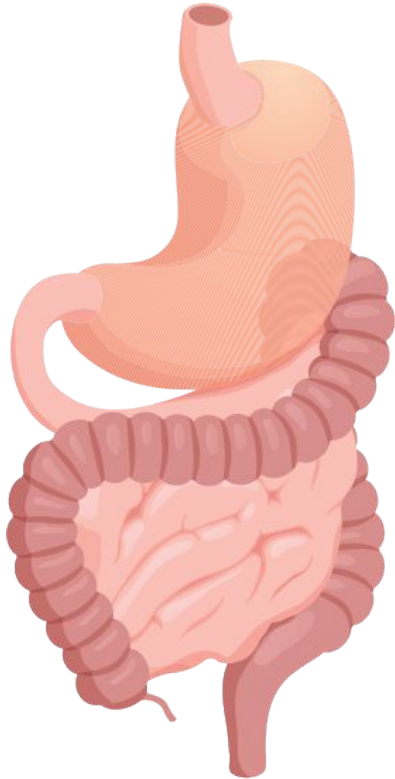
Gampangnya gini, sistem pencernaan terdiri dari berbagai organ yang berpartisipasi dalam proses pencernaan.

Setiap organ melakukan **function spesifik** dan tidak ada pertukaran **function** di antara organ-organ itu.



Misalnya, usus halus dan usus besar memiliki **function** spesifik yang harus dikerjakan.

Pekerjaan itu nggak bisa dituker, usus besar nggak bisa mengambil alih pekerjaan usus kecil dan sebaliknya.



Nah, Kita Kembali Lagi Ke Kotlin ...

Blok **function**, yang merupakan bagian dari sistem, **digunakan untuk menangani tugas tertentu** yang diperlukan untuk kelancaran function seluruh sistem.

Function/method dapat memecah program menjadi sub-sub program, sehingga kita bisa membuat program lebih efisien.

Penggunaan **function/method** juga dapat mengurangi pengetikan kode yang berulang-ulang.



So, Bagaimana Function di Kotlin?

Kotlin menyediakan sejumlah **function** bawaan, kita telah menggunakan sejumlah **function** bawaan dalam contoh.

Misalnya **print()** dan **println()** adalah **function** bawaan yang paling umum digunakan yang kita gunakan untuk mencetak output ke layar.



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



Bagaimana cara membuat dan menggunakan **Function**?

Function harus dibuat atau ditulis di dalam sebuah **Class**.



Visibility modifier

Tipe function sebuah function yang memiliki bagian spesifik dari sebuah function yang mengatur hak akses dari sebuah function. Secara default ketika membuat function akan memiliki modifier ***public***.

Nilai Kembalian

Adalah tipe data dari nilai yang dikembalikan setelah fungsi dieksekusi.

```
private fun makan(): String {  
    // blok kode  
}
```

namaFungsi()

Adalah nama fungsinya. Biasanya ditulis dengan huruf kecil di awalnya. Lalu, kalau terdapat lebih dari satu suku kata, huruf awal di kata kedua ditulis kapital.



Contoh membuat dan memanggil function dalam sebuah function

Memanggil function
makan()

```
fun main() {  
    makan()  
}  
  
fun makan() {  
    println("Ayo makan")  
}
```

Nama function
makan()

Function body



Membuat function dengan tambahan parameter

Informasi dapat diteruskan ke **function** sebagai **parameter**.

Parameter adalah pilihan dan dapat digunakan berdasarkan kebutuhan user.

Parameter ditentukan setelah nama **function**, di dalam tanda kurung. Kita dapat menambahkan **parameter** sebanyak mungkin, cukup dipisah dengan tanda koma (,).





Ketika parameter yang dikirimkan ke sebuah function, hal itu disebut argumen.

Jadi contoh di samping dapat diartikan: fName adalah parameter, sedangkan John, dan Ken adalah argumen.

Nama function Nama parameter Jenis tipe data

```
fun myFunction(fName: String) {  
    println("$fName Dor")  
}
```

```
fun main() {  
    myFunction("John")  
    myFunction("Ken")  
}
```

Memanggil
function

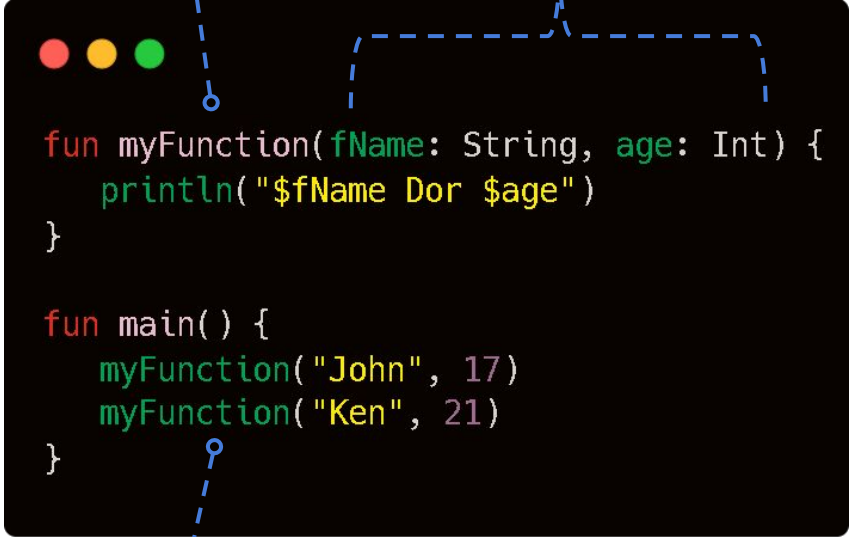


Saat menggunakan beberapa **parameter**, pemanggilan **function** harus :

- memiliki jumlah argumen yang sama dengan parameter yang ada, dan
- argumen harus dimasukkan ke dalam urutan yang sama.

Nama function

Multiple parameter



```
fun myFunction(fName: String, age: Int) {  
    println("$fName Dor $age")  
}  
  
fun main() {  
    myFunction("John", 17)  
    myFunction("Ken", 21)  
}
```

Memanggil
function



Untuk mengatasinya kita dapat menggunakan named argument yang sudah disediakan oleh Kotlin.

Dengan ini, kita tidak perlu lagi menghafal posisi dari sebuah parameter yang ada pada function.

Nama function

Multiple parameter

```
fun myFunction(fName: String, age: Int) {  
    println("$fName Dor $age")  
}  
  
fun main() {  
    myFunction("John", 17)  
    myFunction("Ken", 21)  
}
```

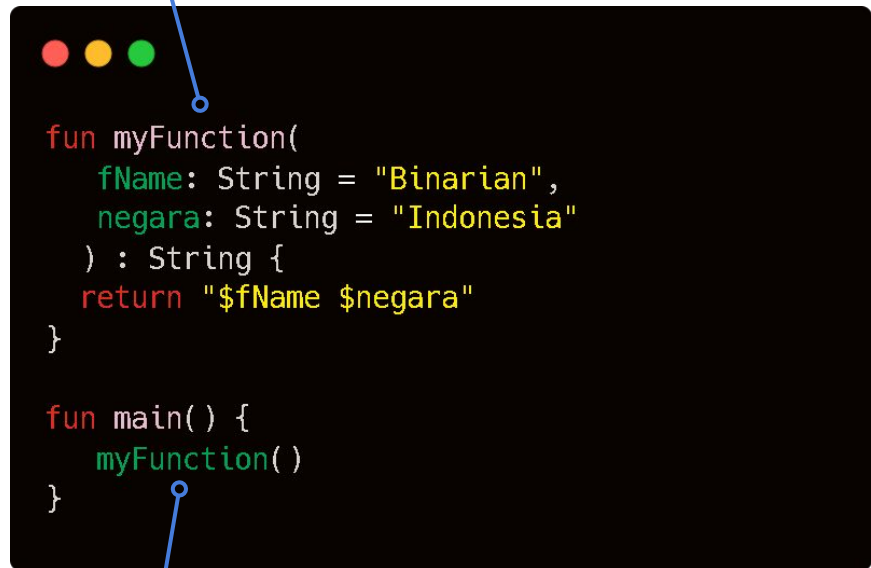
Memanggil
function



Membuat function dengan default argument

Pada Kotlin, kita juga bisa menentukan nilai default dari sebuah parameter yang ada di Kotlin. Jika kita lupa memasukkan argumen pada sebuah function, maka nilai default inilah yang akan ditampilkan.

Nama function



```
fun myFunction(  
    fName: String = "Binarian",  
    negara: String = "Indonesia"  
): String {  
    return "$fName $negara"  
}  
  
fun main() {  
    myFunction()  
}
```

Memanggil
function



Untuk membuatnya cukup mudah, perhatikan syntax di sebelah. Pada parameter yang telah kita buat, kita masukkan juga nilai default yang ingin kita keluarkan.

Dan untuk menampilkannya cukup mudah, kita hanya perlu memanggil nama functionnya tanpa harus mengisi argumen-nya.

Nama function

```
fun myFunction(  
    fName: String = "Binarian",  
    negara: String = "Indonesia"  
): String {  
    return "$fName $negara"  
}  
  
fun main() {  
    myFunction()  
}
```

Memanggil
function

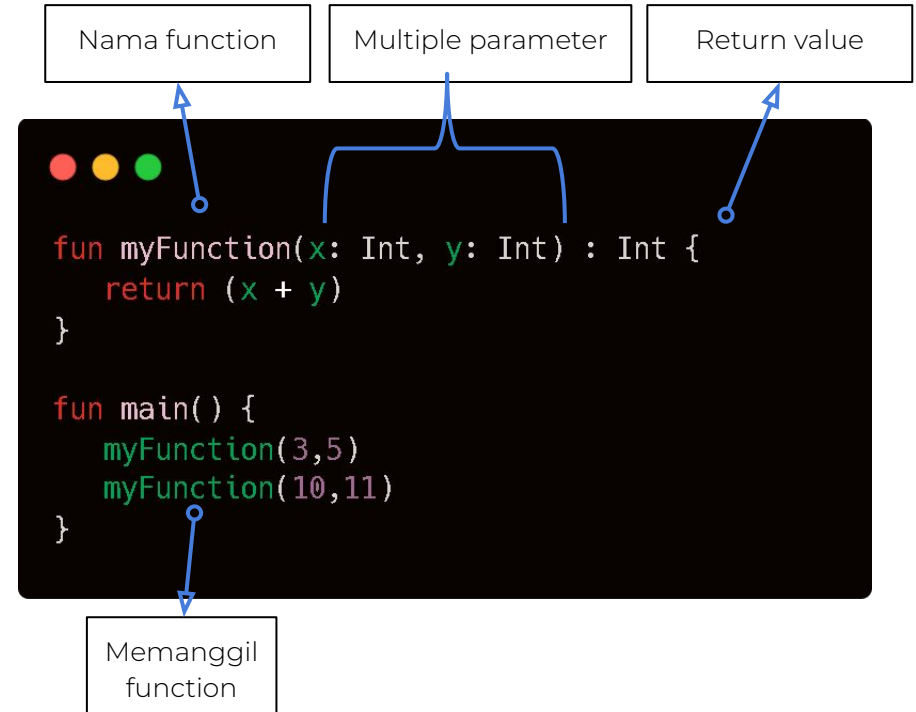


Return Values pada Function

Pada contoh sebelumnya, kita menggunakan function untuk menampilkan sebuah nilai yang terdapat dalam blok kode sebuah function.

Nah pada **return values** ini kita bakal menggunakan function untuk mengembalikan nilai dan menetakannya ke dalam variabel.

Untuk mengembalikan nilai, gunakan keyword **return** dan tentukan spesifik tipe data return setelah tanda kurung pada function





Challenges : Function penentu bilangan Genap-Ganjil

Buat lah program yang dapat menentukan bilangan ganjil genap dengan menggunakan bahasa pemrograman Kotlin.

Tips: Yang dimaksud dengan bilangan ganjil adalah **bilangan-bilangan yang tidak habis dibagi 2** atau dengan kata lain bilangan tersebut akan memiliki sisa jika dibagi dengan 2.

Sementara bilangan genap merupakan kebalikannya dimana bilangan-bilangan yang termasuk ke dalam bilangan genap ini akan habis dibagi 2 (sisa = 0).

Sebagai contoh untuk pengecekan bilangan ganjil yaitu bilangan 5.

$5 : 2 = 2$, sisa 1





Challenges : Function Memunculkan Fibonacci Series

Fibonacci Sequence adalah list angka dimana angka berikutnya merupakan penjumlahan dari dua angka di belakangnya.

Dimulai dari 0 dan 1, list angkanya akan menjadi seperti ini:

0, 1, 1, 2, 3, 5, 13, 21, 34, ...

Nah, buatlah Program / Function yang mampu membuat Fibonacci Series, berdasarkan limit yang dapat ditentukan. (Misal limit hingga angka 100)

Tips: Gunakan Loop **While** atau **For**





Break....

Continued to Next Day...



Function sendiri ternyata punya beberapa fitur looh~ Apa ajaa tuuh?

Sekarang kita pelajari tentang **Extension Function**.



Apa itu Extension Functions ?

Extension function merupakan fitur yang diberikan oleh Kotlin yang memungkinkan kita untuk menambahkan property dan function pada suatu kelas tanpa perlu mengubah atau meng-extend class tersebut.

Fitur ini sangat berguna jika kita menggunakan library yang dibuat oleh orang lain dan menemukan kekurangan di dalamnya.





Misalnya kita pingin buat extension function untuk handle **tipe data String** dan untuk handle **sebuah judul yang memiliki huruf kapital di setiap awal karakternya**.

Kita dapat membuat sebuah extension function untuk menangani hal tersebut.

Contohnya seperti kode disamping :

```
fun String.toTitleCase(): String {  
    return this.split(" ").joinToString(" ") {  
        it.capitalize()  
    }  
}  
  
fun main() {  
    val judul = "android developer Binar is the best!!!"  
    println(judul.toTitleCase())  
}
```



Pada kode di samping, kita membuat sebuah **extension function** untuk tipe data **String** dan bertujuan untuk menghandle judul.

Pada **extension function** tersebut, bertujuan untuk merubah sebuah normal text menjadi text yang memiliki huruf kapital di setiap awal karakternya.

```
fun String.toTitleCase(): String {  
    return this.split(" ").joinToString(" ") {  
        it.capitalize()  
    }  
}  
  
fun main() {  
    val judul = "android developer Binar is the best!!!"  
    println(judul.toTitleCase())  
}
```



Gimana nih? Pemahaman tentang **Extension Function** udah meningkan kan~

Next, kita akan masuk ke fitur berikutnya : **Lambda Expression**.



Kenalan dengan Lambda Expression

Fitur lainnya yang disediakan oleh Kotlin adalah **Lambda Expression**.

Lambda expression atau juga sering disebut sebagai **anonymous function** adalah sebuah **function** yang tidak perlu dideklarasikan tetapi dapat langsung digunakan dalam bentuk **expressions**.

Karena merupakan sebuah **function**, lambda juga dapat memiliki **parameter**, **body**, dan **return** type.





Karakteristik Lambda Expression

Sebelum bisa dilanjut, kita kenalan dulu yuk sama karakteristik dari lambda ini.

- Ketika menggunakan **lambda**, kita tidak perlu mendeklarasikan tipe spesifik untuk **return value**-nya. Tipe tersebut akan ditentukan oleh compiler secara otomatis
- Meskipun sebuah function, **lambda** tidak memerlukan deklarasi **fun** dan **modifier** saat pembuatannya. Hal tersebut karena lambda bersifat anonymous.





- Parameter yang akan ditetapkan berada di dalam kurung kurawal {}.
- Ketika ingin mengembalikan nilai, keyword **return** tidak diperlukan lagi karena compiler secara otomatis mengembalikan nilai dari dalam body.
- **Lambda** dapat digunakan sebagai argumen untuk sebuah **parameter** dan dapat disimpan ke dalam sebuah variabel.





Cara Deklarasi Lambda Expression

Untuk membuat lambda expression kita dapat mengikut format seperti dibawah ini :

```
val lambda_name : Data_type = { argument_List -> code_body }
```

Lambda expressions selalu dikelilingi oleh kurung kurawal “{}”, deklarasi argumen masuk ke dalam kurung kurawal dan memiliki anotasi tipe opsional, **code_body** dieksekusi setelah tanda panah.





Jika return value yang didapatkan bukanlah Unit, maka ekspresi terakhir di dalam badan lambda diperlakukan sebagai return value.

Contohnya:

```
val sum = {a: Int , b: Int -> a + b}
```





Di Kotlin, lambda expression berisi bagian opsional kecuali `code_body`.

Di bawah ini adalah lambda expression setelah menghilangkan bagian opsional.

```
val sum:(Int,Int) -> Int = { a, b -> a + b }
```

Catatan: Kita tidak selalu memerlukan variable karena variable dapat diteruskan secara langsung sebagai argumen ke suatu function.





Yuk Coba Praktek~

Berikut contoh deklarasi kode Lambda :



```
// dengan type anotasi dalam lambda expression
val sum1 = { a: Int, b: Int -> a + b }

// tanpa type anotasi dalam lambda expression
val sum2:(Int,Int)-> Int = { a , b -> a + b}

fun main() {
    val result1 = sum1(2,3)
    val result2 = sum2(3,4)
    println("The sum of two numbers is: $result1")
    println("The sum of two numbers is: $result2")

    // langsung mencetak return value dari lambda
    // tanpa perlu menyimpan ke dalam variable.
    println(sum1(5,7))
}
```



Mudahkan konsep deklarasi **Lambda Expression?**

Selanjutnya, kita akan masuk ke fitur berikutnya, **Higher-Order Function**.



Apa itu Higher-Order Functions?

Bahasa pemrograman Kotlin memiliki dukungan luar biasa untuk functional programming. Function pada Kotlin dapat disimpan dalam **variabel** dan **struktur data**, diteruskan sebagai argumen dan dikembalikan dari **higher-order function** lainnya.





Pada Kotlin, sebuah **function** dapat menerima **function** sebagai parameter atau dapat mengembalikan **function** disebut **higher-order function**.

Sebagai pengganti **Integer**, **String** atau **Array** sebagai parameter untuk **function**. Seringkali, **lambdas** dilewatkan sebagai parameter dalam **function** Kotlin untuk kenyamanan.





Kegunaan Higher-Order Function

Dengan menerapkan **higher-order function**, kita akan mendapatkan keuntungan sebagai berikut:

- Membantu mengurangi redundansi (kode yang berulang) kode dengan mengizinkan fungsionalitas kode untuk diteruskan sebagai function ke function lain.
- Memudahkan developer untuk membaca kode.





Mengirimkan Lambda Expression sebagai Parameter Higher-Order Function

Kita dapat mengirimkan **lambda expression** sebagai parameter **higher-order function**.

Terdapat 2 jenis function yang dapat dikirimkan:

- Function dengan kembalian Unit
- Function dengan kembalian salah satu nilai Integer, String, dll.





Contoh syntax higher-order function dengan kembalian Unit:

```
// regular function definition
fun printMe(s:String): Unit{
    println(s)
}
// higher-order function definition
fun higherfunc( str : String, myfunc: (String) -> Unit){
    // invoke regular function using local name
    myfunc(str)
}
fun main(args: Array<String>) {
    // invoke higher-order function
    higherfunc("Android Developer Binar luar biasa",::printMe)
}
```

Outputnya:

```
Android Developer Binar luar biasa
```





Penjelasannya Begini ...

```
// regular function definition
fun printMe(s:String): Unit{
    println(s)
}
// higher-order function definition
fun higherfunc( str : String, myfunc: (String) -> Unit){
    // invoke regular function using local name
    myfunc(str)
}
fun main(args: Array<String>) {
    // invoke higher-order function
    higherfunc("Android Developer Binar luar biasa",::println)
}
```

Kita mendefinisikan regular function **printMe()** yang menerima parameter tipe **String** dan mengembalikan **Unit**.

Outputnya:

```
Android Developer Binar luar biasa
```



Penjelasannya Begini ...

```
// regular function definition
fun printMe(s:String): Unit{
    println(s)
}
// higher-order function definition
fun higherfunc( str : String, myfunc: (String) -> Unit){
    // invoke regular function using local name
    myfunc(str)
}
fun main(args: Array<String>) {
    // invoke higher-order function
    higherfunc("Android Developer Binar luar biasa",::printMe)
}
```

(s: String) adalah satu-satunya parameter.
Unit mewakili **return** type.

Kemudian, kita definisikan **higher-order function** dengan dua parameter yang pertama dengan tipe **String** dan yang lainnya adalah tipe **function**.

Outputnya:

```
Android Developer Binar luar biasa
```



Penjelasannya Begini ...

```
// regular function definition
fun printMe(s:String): Unit{
    println(s)
}
// higher-order function definition
fun higherfunc( str : String, myfunc: (String) -> Unit){
    // invoke regular function using local name
    myfunc(str)
}
fun main(args: Array<String>) {
    // invoke higher-order function
    higherfunc("Android Developer Binar luar biasa",::printMe)
}
```

(s: String) mewakili parameter String.

myfunc : (String) -> Unit mewakili bahwa ia menerima function sebagai parameter yang mengembalikan Unit. Dari function main, higher-order function dipanggil dengan mengirimkan data String dan function sebagai argumen

Outputnya:

```
Android Developer Binar luar biasa
```

Saatnya kita
Quiz!





1. Berikut adalah pernyataan yang benar tentang Function, kecuali

- A. Function digunakan oleh semua bahasa pemrograman, tidak hanya kotlin.
- B. Function adalah Sebuah fungsi atau kumpulan perintah yang dikelompokkan bersama untuk melakukan tugas tertentu.
- C. Function dalam Kotlin berbeda dengan istilah Method yang dipakai di pemrograman lain.



1. Berikut adalah pernyataan yang benar tentang Function, kecuali

- A. Function digunakan oleh semua bahasa pemrograman, tidak hanya kotlin.
- B. Function adalah Sebuah fungsi atau kumpulan perintah yang dikelompokkan bersama untuk melakukan tugas tertentu.
- C. Function dalam Kotlin berbeda dengan istilah Method yang dipakai di pemrograman lain.

Function atau **method** merupakan sebuah kumpulan baris kode atau perintah yang berguna untuk menjalankan suatu tugas tertentu dan dapat kita panggil dengan mudah. Function dikenal juga di bahasa pemrograman lain selain Kotlin.

Nah, Istilah method itu juga sebenarnya sama dengan **Function** dalam Kotlin 😊



2. Apa yang dimaksud dengan parameter?

- A. Sebuah data yang dibutuhkan oleh function yang nantinya dapat diolah oleh function
- B. Sebuah data yang dikirimkan ketika memanggil sebuah function
- C. Sebuah nilai pada function



2. Apa yang dimaksud dengan parameter?

- A. Sebuah data yang dibutuhkan oleh function yang nantinya dapat diolah oleh function
- B. Sebuah data yang dikirimkan ketika memanggil sebuah function
- C. Sebuah nilai pada function

Parameter merupakan sebuah data yang dibutuhkan oleh function yang nantinya dapat diolah function sesuai dengan kebutuhannya. Data yang dikirimkan pada function saat pemanggilan function disebut dengan argumen.



3. Bagaimanakah cara penulisan format parameter pada function yang benar?

A

```
fun myFunction (valueA: String, valueB: Int)
```

B

```
fun myFunction (String: valueA, Int: valueB)
```

C

```
fun myFunction (valueA String, valueB Int)
```



3. Bagaimanakah cara penulisan format parameter pada function yang benar?

A

```
fun myFunction (valueA: String, valueB: Int)
```

B

```
fun myFunction (String: valueA, Int: valueB)
```

C

```
fun myFunction (valueA String, valueB Int)
```

Untuk membuat **parameter** yang terdapat pada function, kita dapat membuatnya dengan memasukkan nama variabel, diikuti tanda **titik dua “:”**, dan diakhiri dengan **tipe data**.



4. Perhatikan syntax berikut, function manakah yang paling tepat agar function itu berjalan?



```
fun main(){  
    myFunction(umur = 25, fName = "Binarian Sejati", sekolah = "SMP Al-Azhar", kelas = "12C")  
}  
  
// Output  
// Nama saya : Binarian Sejati, sekolah di SMP Al-Azhar, kelas: 12C, dan berumur 25
```

A

```
fun myFunction(fName: String, sekolah: String, kelas: String, umur: Int) {  
    println("Nama saya : $fName, sekolah di $sekolah, kelas: $kelas, dan berumur $umur")  
}
```

B

```
fun myFunction(fName: String, sekolah: String, kelas: Int, umur: Int) {  
    println("Nama saya : $fName, sekolah di $sekolah, kelas: $kelas, dan berumur $umur")  
}
```

C

```
fun myFunction(fName: String, sekolah: String, kelas: Double, umur: Float) {  
    println("Nama saya : $fName, sekolah di $sekolah, kelas: $kelas, dan berumur $umur")  
}
```



4. Perhatikan syntax berikut, function manakah yang paling tepat agar function itu berjalan?



```
fun main(){  
    myFunction(umur = 25, fName = "Binarian Sejati", sekolah = "SMP Al-Azhar", kelas = "12C")  
}  
  
// Output  
// Nama saya : Binarian Sejati, sekolah di SMP Al-Azhar, kelas: 12C, dan berumur 25
```

A

```
fun myFunction (fName: String, sekolah: String, kelas: String, umur: Int) {  
    println("Nama saya :$fName, sekolah di $sekolah, kelas: $kelas, dan berumur $umur")  
}
```

Untuk dapat menampilkan sesuai output yang diinginkan, kita dapat membuat sebuah parameter yang dapat diisi secara **dinamis**. Dan kita harus memperhatikan **tipe data** pada parameter yang kita buat, jika terdapat **perbedaan** antara **argumen** yang dikirim dan **parameter**. Hal tersebut akan menimbulkan error



5. Jika kita membutuhkan suatu function yang dapat menerima suatu function sebagai parameter atau dapat mengembalikan suatu function, maka jenis function yang digunakan adalah

- A. Extensions Function
- B. Return Value
- C. Higher-Order Function



5. Jika kita membutuhkan suatu function yang dapat menerima suatu function sebagai parameter atau dapat mengembalikan suatu function, maka jenis function yang digunakan adalah

- A. Extensions Function
- B. Return Value
- C. Higher-Order Function**

Sebuah fungsi yang menggunakan fungsi lainnya sebagai **parameter**, menjadikan tipe kembalian atau keduanya. **Higher-Order Function** merupakan fitur yang memanfaatkan **lambda**.



Referensi dan bacaan lebih lanjut~

- Definisi function





Nah, selesai sudah pembahasan kita di Chapter 1 Topic 5 ini.

Selanjutnya, kita bakal masuk ke topik 6, dimana kita akan memperkaya konsep pemrograman kita lagi dengan **OOP**

Stroberi dimakan ngengat, semangat~



Terima Kasih!



Next Topic

loading...