

Training Kotlin

...

Basic Programming with Kotlin

Topic

- Kotlin Introduction
- Output
- Variable
- Data Types
- Operator
- Array
- Selection
- Looping
- Looping Array
- Function & Lambda
- Class & Object
- Inheritance
- Interface
- Collection - List, Set & Map

Kotlin Introduction

- Bahasa pemrograman Kotlin dikenalkan oleh Perusahaan JetBrains pada tahun 2011
- Kotlin dapat berjalan di atas JVM (Java Virtual Machine), jadi jika sudah menggunakan Java Kotlin bisa langsung berjalan tanpa harus install-dependensi tambahan
- Kotlin di desain agar terintegrasi dengan Java, bahkan dapat di convert dari bahasa Kotlin ke dalam bahasa Java
- Tahun 2017, Google mengumumkan bahwa Kotlin adalah bahasa pemrograman yang direkomendasikan untuk pengembangan aplikasi Android, walau saat ini bahasa Java masih banyak digunakan dalam pembuatan aplikasi Android
- Jika Java membutuhkan Maven sebagai Dependensi, maka Kotlin menggunakan Gradle sebagai Dependensi nya
- Pada bahasa Kotlin saat ini sudah dapat membuat Restful API menggunakan Spring
- Berjalan di SDK versi 8 keatas
- Editor yang disarankan adalah IntelliJ IDEA

Output



```
1 fun main() {  
2  
3     println("Enigmacamp IT Bootcamp")  
4     print("Halo saat ini saya sedang belajar bahasa kotlin")  
5 }
```

***Print** digunakan untuk mencetak tanpa membuat baris baru*

***Println** digunakan untuk mencetak dengan membuat baris baru*

Variable

```
1 fun main() {  
2  
3     //mutable variable  
4     var firstName = "Jution"  
5     var lastName = "Kirana"  
6  
7     println(firstName)  
8     println(lastName)  
9  
10    lastName = "Candra Kirana"  
11    print(lastName)  
12 }
```

```
1 fun main() {  
2  
3     //immutable variable  
4     val firstName = "Jution"  
5     val lastName = "Kirana"  
6  
7     println(firstName)  
8     println(lastName)  
9  
10    //akan error  
11    lastName = "Candra Kirana"  
12    print(lastName)  
13 }
```

```
1 package basic  
2  
3 const val APP_CONFIG = "localhost"  
4 const val PORT = "8080"  
5  
6 fun main() {  
7  
8     println(APP_CONFIG)  
9     println(PORT)  
10 }
```

Disarankan untuk menggunakan immutable, jika variabel yang dibuat tidak di assign ulang


Data Types

```
fun main(){  
    var namaLengkap : String  
        namaLengkap = "Doni Octavian"  
        println(namaLengkap)  
}
```

```
fun main(){  
    var namaLengkap : "Doni Octavian"  
        println(namaLengkap)  
}
```

1. Untuk Variabel yang belum diberi nilai wajib diberi tipe datanya
2. Tipe Data Diawali dengan huruf kapital
3. Tipe data lainnya sama dengan bahasa pemrograman yang lain, seperti tipe data string, number, boolean dan character

Operator



```
fun main(){  
    var angka1 = 1  
    var angka2 = 2  
    var hasil = angka1 + angka2  
    println(hasil)  
}
```

Operator merupakan suatu simbol yang sering dan biasa untuk melakukan operasi perhitungan :

+ - % * /

Beberapa operator yang dapat digunakan: operator matematika, operator perbandingan dan operasi boolean

Array

```
fun main(){  
    var angka = arrayOf(1, 2, 3, 4, 5)  
    var buah = arrayOf("mangga","Apel","Pisang")  
    var kendaraan : Array<String> = arrayOf("motor","mobil")  
  
    println(angka.joinToString (""))  
    println(buah.joinToString ("\n"))  
    println(kendaraan.joinToString (""))  
}
```

Penggunaan Array Terjadi Saat kita ingin menyimpan element dengan urutan tertentu dan kita bisa mengaksesnya melalui indexnya

Selection

```
fun main() {  
    val nilai = 60  
  
    if (nilai > 90) {  
        println("A")  
    } else if (nilai > 75) {  
        println("B")  
    } else if (nilai > 60) {  
        println("C")  
    } else if (nilai > 50) {  
        println("D")  
    } else {  
        println("E")  
    }  
}
```

```
fun main() {  
    val finalExam = "A"  
    when (finalExam) {  
        "A" -> {println("Luar Biasa")}  
        "B" -> {println("Bagus")}  
        "C" -> {println("Cukup")}  
        "D" -> {println("Kurang")}  
        "E" -> {println("Coba lagi tahun depan")}  
        else -> {println("Nilai tidak diketahui")}  
    }  
}
```

- Dalam Kotlin, if adalah salah satu kata kunci yang digunakan untuk percabangan
- Percabangan artinya kita bisa mengeksekusi kode program tertentu ketika suatu kondisi terpenuhi
- Hampir di semua bahasa pemrograman mendukung if expression

Selection - When Else & In Expression

```
1 fun main() {  
2  
3     val finalExam = 70  
4     when {  
5         finalExam > 85 -> print("PASS")  
6         finalExam > 75 -> print("GOOD")  
7         finalExam > 65 -> print("VERY GOOD")  
8         else -> print("TRY AGAIN!")  
9     }  
10 }
```

```
1 fun main() {  
2  
3     val finalExam = 'D'  
4     val nilai = arrayOf('A', 'B', 'C')  
5     when(finalExam) {  
6         in nilai -> print("PASS")  
7         else -> print("TRY AGAIN!")  
8     }  
9 }
```

Looping

Looping di kotlin hampir sama seperti di bahasa pemrograman lain, yaitu ada for, while dan do while

```
1 fun main() {  
2  
3     //Perulangan menggunakan range  
4     for (number in 0..10) {  
5         println(number)  
6     }  
7  
8     //perulangan menggunakan downTo, dengan perpindahan kelipatan dua  
9     for (value in 10 downTo 0 step 2) {  
10        println(value)  
11    }  
12 }
```

```
1 fun main() {  
2  
3     var number = 0;  
4  
5     while (number < 10) {  
6         println(number)  
7         number++  
8     }  
9 }
```

```
1 fun main() {  
2  
3     var number = 0  
4  
5     do {  
6         println(number)  
7         number++  
8     } while (number < 10)  
9 }
```

Looping Array

```
1 fun main() {  
2  
3     //Perulang Array  
4     val languageProgrammings = arrayOf("Java", "Javascript", "ReactJs", "NodeJs", "Go", "Kotlin")  
5     for (languageProgramming in languageProgrammings) {  
6         println(languageProgramming)  
7     }  
8  
9     // declaring an array using arrayOf<Int>  
10    // Direkomendasikan menggunakan range ( in )  
11    val arrayNumber = arrayOf<Int>(10, 20, 30, 40, 50)  
12    for (i in 0..arrayNumber.size-1)  
13    {  
14        print(" "+arrayNumber[i])  
15    }  
16 }
```

Function

```
1 fun main() {  
2  
3     // Pemanggilan function  
4     greetingMessage()  
5 }  
6  
7 //Pembuatan function tanpa ada parameter  
8 fun greetingMessage() {  
9     println("Hello! My Name is Jution")  
10    println("Enigmacamp IT Bootcamp")  
11 }
```

Function with Parameter

```
1 fun main() {  
2  
3     greetingMessage("Jution Candra Kirana", "Enigmacamp IT Bootcamp")  
4 }  
5  
6 fun greetingMessage(name: String, office: String) {  
7     println("Hello $name, now I work in $office")  
8 }
```

```
1 fun main() {  
2  
3     greetingMessage("Jution Candra Kirana", "Enigmacamp IT Bootcamp")  
4     val printFun = greetingMessage2("Jution Candra Kirana", "Enigmacamp IT Bootcamp")  
5     println(printFun)  
6 }  
7  
8 fun greetingMessage(name: String, office: String) {  
9     //Template variable  
10    println("Hello $name, now I work in $office")  
11 }  
12  
13 //Function dengan fungsi balikan  
14 fun greetingMessage2(name: String, office: String): String {  
15     return "Hello $name, now I work in $office"  
16 }
```

```
1 fun main() {  
2  
3     val add = multiple(10, 10)  
4     println(add)  
5 }  
6  
7 fun multiple(x: Int, y: Int): Int {  
8     return x + y;  
9 }
```

Function with Default Parameter

```
1 fun main() {  
2  
3     greeting("Jution")  
4     greeting("Jution", "Kirana")  
5 }  
6  
7 fun greeting(firstName: String, lastName: String = "") {  
8     println("Hello $firstName $lastName")  
9 }
```

Single Expression Function

```
1 fun main() {  
2  
3     val divideAssign = divide(10,5)  
4     println(divideAssign);  
5     greetMessage("Jution Candra Kirana")  
6 }  
7  
8 //Return  
9 fun divide(number1: Int, number2: Int): Int = number1 / number2  
10  
11 //No Return / Unit  
12 fun greetMessage(fullName: String): Unit = println(fullName)
```


More about Function

- Function Varargs Parameter
- Extension Function
- Function Infix Notation
- Function Scope
- Function with return If and When
- Recursive Function
- Tail Recursive Function

Hands On

Terdapat sebuah deret bambu dengan Panjang yang sudah ditentukan. Untuk setiap cycle masing masing bamboo dipotong 1 satuan. Buatlah program dengan kotlin untuk merepresentasikan setiap cycle nya.

Input

^ input jumlah bamboo : 5
 > Panjang bamboo ke - 1 : 4
 > Panjang bamboo ke - 2 : 1
 > Panjang bamboo ke - 3 : 2
 > Panjang bamboo ke-4 : 7
 > Panjang bamboo ke-5 : 3
^ Input jumlah cycle : 3

>initial

--

>cycle ke 1 :

--

> cycle ke 2 :

--

-

>cycle ke 3 :

--

Lambda

```
1 fun main() {  
2  
3     // Lambda  
4     val sum: (Int, Int) -> Int = {  
5         x: Int, y: Int -> x + y  
6     }  
7  
8     // Eksekusi Lambda  
9     println(sum(10,10))  
10 }
```


Lambda adalah sebuah function tanpa nama.

Penggunaan lambda sangat berguna pada saat pembuatan aplikasi mobile atau pembuatan project besar seperti REST API.

Lambda bisa mereferensi sebuah function yang ada. Dslam kotlin disebut sebagai method reference.

Kata kuncinya adalah dengan menuliskan `::namafunc`

Lambda: Method Reference



```
1 fun main() {  
2     val uppercase: (String) -> String = ::ToUpperCase  
3  
4     val name = uppercase("Jution")  
5     println(name)  
6 }  
7  
8 fun ToUpperCase(name: String): String = name.toUpperCase()
```

Class & Object

- Pembuatan class pada kotlin dengan kata kunci **class**
- Usahakan pemberian nama class sama dengan nama file nya
- Dalam pembuatan sebuah **Object** di kotlin sama hal nya dengan pemanggilan sebuah function dengan memanggil nama class nya
- Jika di java membutuhkan kata kunci **new**, di Kotlin kita tidak memerlukannya

```
1 package oop
2
3 class Animal {}
```

```
1 package oop
2
3 class Animal {}
4
5 fun main() {
6
7     val cat = Animal()
8     println(cat)
9 }
```

```
1 package oop
2
3 class Animal {
4     var name: String = ""
5     var type: String = ""
6     var feet: Int? = null
7 }
8
9 fun main() {
10     val cat = Animal()
11     cat.name = "Pussy"
12     cat.type = "Omnivora"
13     cat.feet = 4
14     println(cat.name)
15     println(cat.type)
16     println(cat.feet)
17 }
```

Constructor

```
1 package oop
2
3 class Hero(var name: String, var hp: Double, var damage: Double) {
4
5     fun printInfo() {
6         println(this)
7         println("My Hero is a name: ${this.name}, hp: ${this.hp}, damage: ${this.damage}")
8     }
9 }
10
11 fun main() {
12     val gundala = Hero("Gundala Jagoan Petir", 10.0, 100.0)
13     gundala.printInfo()
14 }
```

Bagaimana jika kita membuat sebuah function yang bersifat overloading ?

Inheritance

- Inheritance atau pewarisan sifat.
- Child class hanya dapat mempunyai 1 parent class, tetapi parent class bisa memiliki lebih dari satu child
- Di Kotlin class bersifat final, artinya tidak dapat diturunkan/diwariskan, untuk itu untuk melakukan pewarisan dapat menggunakan kata kunci **open** di depan sebuah class

```
1 package oop
2
3 open class Employee(val name: String) {
4
5     fun sayHello(name: String) {
6         println("Hello $name, my name is ${this.name}")
7     }
8 }
9
10 class Manager(name: String): Employee(name)
11 class VicePresident(name: String): Employee(name)
12
13 fun main() {
14     val jution = Manager("Jution")
15     jution.sayHello("Arif")
16
17     val doni = VicePresident("Doni")
18     doni.sayHello("Arif")
19 }
```

```
1 package oop
2
3 open class Employee(val name: String) {
4     open fun sayHello(name: String) {
5         println("Hello $name, my name is ${this.name}")
6     }
7 }
8
9 class Manager(name: String): Employee(name) {
10     override fun sayHello(name: String) {
11         println("Hello $name, my name is manager ${this.name}")
12     }
13 }
14
15 fun main() {
16     val jution = Manager("Jution")
17     jution.sayHello("Arif")
18 }
```

Interface

```
1 package oop
2
3 interface Engine {
4
5     fun starEngine()
6 }
```

```
1 package oop
2
3 class ElectricEngine: Engine {
4
5     override fun starEngine() {
6         println("Mesin listrik menyala...")
7     }
8 }
```

```
1 package oop
2
3 class GasolineEngine: Engine {
4     override fun starEngine() {
5         println("Mesin gasoline menyala...")
6     }
7 }
```

```
1 package oop
2
3 fun main() {
4
5     val electricEngine = ElectricEngine()
6     electricEngine.starEngine()
7     val gasolineEngine = GasolineEngine()
8     gasolineEngine.starEngine()
9
10 }
```


Collection - List, Set & Map

- **List**, adalah collection yang datanya seperti Array dan memiliki akses data menggunakan index
- Data di List boleh duplicate, artinya kita boleh memasukkan data yang sama berkali-kali ke dalam sebuah List yang sama
- Set, adalah collection yang datanya harus unik dan tidak pasti berurut
- Saat kita memasukkan data duplicate ke Set, maka data hanya akan disimpan satu, data duplicatenya tidak akan ditambahkan ke dalam Set
- Set sangat cocok untuk menyimpan data yang emang unik, tidak boleh sama
- Set menggunakan function *hashCode()* dan *equals()* untuk membandingkan apakah sebuah object sama atau tidak, jika *hashCode()* dan *equals()* nya sama, maka dianggap data tersebut duplicate, dan tidak akan diterima oleh Set
- Map (atau dictionary), adalah collection yang berbentuk key-value, dimana key berperan sebagai index.
- Key harus unik, jika kita menggunakan Key yang sama, maka data sebelumnya akan di replace oleh data yang baru
- Tipe data key bisa menggunakan tipe data apapun sesuai yang kita inginkan

Collection - List

```
1 package collection
2
3 fun main() {
4
5     val lists: List<String> = listOf("Java", "Golang", "Kotlin")
6     for (list in lists) {
7         println(list)
8     }
9
10    println(lists.isEmpty())
11    println(lists.indexOf("Java"))
12    println(lists.contains("Java"))
13    println(lists.joinToString())
14 }
```

Ada beberapa function di List yang dapat digunakan, dan function di mutableList yang dapat digunakan

```
1 package collection
2
3 data class Student(val name: String, val age: Int, val gender: String)
```

```
1 package collection
2
3 fun main() {
4
5     val jution = Student("Jution", 25, "L")
6     val studentList = mutableListOf<Student>()
7
8     // Add to list
9     studentList.add(jution)
10    println(studentList.joinToString())
11
12 }
```

Collection - Set

```
1 package collection
2
3 fun main() {
4
5     val integerSet = setOf(1, 7, 2, 4, 2, 1, 5)
6     println("A: $integerSet")
7
8     val setA = setOf(1, 2, 4, 2, 1, 5)
9     val setB = setOf(1, 2, 4, 5)
10    println(setA == setB)
11
12    val setC = setOf(1, 5, 7)
13    val union = setA.union(setC)
14    val intersect = setA.intersect(setC)
15    val complement = setA.subtract(setC)
16    println("UNION: $union")
17    println("INTERSECT: $intersect")
18    println("COMPLEMENT: $complement")
19
20 }
```

```
1 package collection
2
3 fun main() {
4
5     val mutableSet: MutableSet<String> = mutableSetOf()
6     mutableSet.add("Java")
7     mutableSet.add("Go")
8     mutableSet.add("Kotlin")
9     mutableSet.add("Go")
10
11    println(mutableSet.size)
12    println(mutableSet.joinToString())
13    println(mutableSet.contains("Go"))
14
15    mutableSet.remove("Go")
16    println(mutableSet.size)
17    println(mutableSet.joinToString())
18 }
```

Collection - Map

```
1 package collection
2
3 fun main() {
4
5     val map: Map<String, String> = mapOf(
6         "a" to "Java",
7         "b" to "Golang",
8         "c" to "Kotlin"
9     )
10    println(map.size)
11    println(map["a"])
12    println(map["b"])
13    println(map["c"])
14
15 }
```

```
1 package collection
2
3 fun main() {
4
5     val pairs = mutableMapOf<Int, String>()
6
7     pairs.put(1, "Januari")
8     pairs.put(2, "Februari")
9     pairs.put(3, "Maret")
10    pairs.put(4, "April")
11    pairs.put(5, "Mei")
12    pairs.put(6, "Juni")
13    pairs.put(7, "Juli")
14
15    pairs[1] = "Agustus"
16
17    for (pair in pairs) {
18        println("${pair.key}, ${pair.value}")
19    }
20
21    println(pairs.get(4))
22    println(pairs[7])
23 }
```