



Kotlin

Rafael Vieira Coelho
rafaelvc2@gmail.com

<https://kotlinlang.org>

Funções

- fun: palavra reservada que indica início de função
- max: nome da função
- a e b: parametros de entrada
- Int: tipo de retorno

```
fun max(a: Int, b: Int): Int {  
    return if (a > b) a else b  
}
```

Podemos Melhorar a Função

- Converter em uma expressão

```
fun max(a: Int, b: Int): Int {  
    return if (a > b) a else b  
}
```

Convert to expression body



```
fun max(a: Int, b: Int): Int = if (a > b) a else b
```

Podemos Melhorar a Função

- Omitir o tipo de retorno

```
fun max(a: Int, b: Int): Int {  
    return if (a > b) a else b  
}
```

Convert to expression body



```
fun max(a: Int, b: Int) = if (a > b) a else b
```

Tipo de Retorno Unit

```
fun displayMax(a: Int, b: Int) {  
    println(max(a, b))  
}
```

: Unit

Unit é equivalente a void em Java

Escopo das Funções

Top-level function:

```
fun topLevel() = 1
```

Member function:

```
class A {  
    fun member() = 2  
}
```

Local function:

```
fun other() {  
    fun local() = 3  
}
```

Pergunta: Podemos chamar uma função top-level de Java?

- Você não pode fazer isto
- Você chama como uma função estática a partir do nome do arquivo java
- Você chama como uma função membro da classe java

Pergunta: Podemos chamar uma função top-level de Java?

- Você não pode fazer isto
- Você chama como uma função estática a partir do nome do arquivo java**
- Você chama como uma função membro da classe java

Exemplo

MyFile.kt

```
package intro
```

```
fun foo() = 0
```



UsingFoo.java

```
package other;
```

```
import intro.MyFileKt;
```

```
public class UsingFoo {
```

```
    public static void main(String[] args) {
```

```
        MyFileKt.foo();
```

```
}
```

```
}
```



Podemos Fazer a Importação Estática

MyFile.kt

```
package intro
```

```
fun foo() = 0
```

UsingFoo.java

```
package other;
```

```
import static intro.MyFileKt.*;
```

```
public class UsingFoo {
```

```
    public static void main(String[] args) {
```

```
        foo();
```

```
}
```

```
}
```



Podemos Dar um Apelido via Notação

Extensions.kt

```
@file:JvmName("Util")  
package intro
```

```
fun foo() = 0
```



JavaUsage.java

```
package other;
```

```
import intro.Util;
```

```
public class JavaUsage {  
    public static void main(String[] args) {  
        int i = Util.foo();  
    }  
}
```

Podemos dar um apelido ao arquivo com funções estáticas de auxílio através da notação @JvmName.



Valores Padrão para Parâmetros

```
fun displaySeparator(character: Char = '*', size: Int = 10) {  
    repeat(size) {  
        print(character)  
    }  
}
```

Valores Padrão para Parâmetros

```
fun displaySeparator(character: Char = '*', size: Int = 10) {  
    repeat(size) {  
        print(character)  
    }  
}
```

```
displaySeparator('#', 5)          // #####  
displaySeparator('#')             // #####  
displaySeparator()               // *****
```

Argumentos com Nome Explícito

```
fun displaySeparator(character: Char = '*', size: Int = 10) {  
    repeat(size) {  
        print(character)  
    }  
}
```

```
displaySeparator(size = 5) // *****
```

Aula01_Kotlin > src > Ex2Funcoes.kt

Project + - Ex1OlaMundo.kt Ex2Funcoes.kt

1: Project

Aula01_Kotlin ~/Dropbox/Livros
 |.idea
 | out
 | src
 | Ex1OlaMundo.kt
 | Ex2Funcoes.kt
 | Aula01_Kotlin.iml
External Libraries
Scratches and Consoles

2: Structure

3: Favorites

Run: Ex2FuncoesKt

4: Messages 5: Run 6: TODO 7: Terminal 8: Event Log

Build completed successfully in 3 s 981 ms (moments ago)

14:1 LF UTF-8 4 spaces

```
1 fun printMessage(message: String): Unit { // 1
2     println(message)
3 }
4
5 fun printMessageWithPrefix(message: String, prefix: String = "Info") { // 2
6     println("[prefix] $message")
7 }
8
9 fun sum(x: Int, y: Int): Int { // 3
10    return x + y
11 }
12
13 fun multiply(x: Int, y: Int) = x * y // 4
14
15 fun main() { // 5
16     printMessage("Hello")
17     printMessageWithPrefix(message: "Hello", prefix: "Log") // 6
18     printMessageWithPrefix(message: "Hello") // 7
19     printMessageWithPrefix(prefix = "Log", message = "Hello") // 8
20     println(sum( x: 1, y: 2)) // 9
21 }
```

Output:
Hello
[Log] Hello
[Info] Hello
[Log] Hello
3

Pergunta: O que ocorre ao rodar o código?

```
1 fun displaySeparator(character: Char = '*', size: Int = 10) {  
2     repeat(size) {  
3         print(character)  
4     }  
5 }  
6  
7 displaySeparator(3, '5')
```

- 33333
- 555
- O código não compila

Pergunta: O que ocorre ao rodar o código?

```
1 fun displaySeparator(character: Char = '*', size: Int = 10) {  
2     repeat(size) {  
3         print(character)  
4     }  
5 }  
6  
7 displaySeparator(3, '5')
```

- 33333
- 555
- O código não compila

Podemos Usar o Nome dos Parâmetros para Contornar o Problema

```
fun displaySeparator(character: Char = '*', size: Int = 10) {  
    repeat(size) {  
        print(character)  
    }  
}
```

```
displaySeparator(3, '5')
```

```
displaySeparator(size = 3, character = '5') // 555
```

Solução Java: Sobrecarga de Métodos

```
public void displaySeparator(char character, int size) {  
    /* ... */  
}  
  
public void displaySeparator(char character) {  
    displaySeparator(character, 10);  
}  
  
public void displaySepatator() {  
    displaySeparator('*');  
}
```



Sobrecarga de Métodos em Kotlin via Notação

```
@JvmOverloads  
fun sum(a: Int = 0, b: Int = 0, c: Int = 0) { }
```

- Gera automaticamente 4 Sobrecargas:

```
public static final int sum(int a, int b, int c)  
public static final int sum(int a, int b)  
public static final int sum(int a)  
public static final int sum()
```

Modificador vararg

- Permite transformar um parâmetro em vários
- No momento de execução, o vararg é apenas um Array
- Veja o exemplo a seguir

Aula01_Kotlin > src > Ex5Varargs.kt

Project Aula01_Kotlin ~/Dropbox/Livros .idea out src Ex1OlaMundo.kt Ex2Funcoes.kt Ex3FuncoesInfix.kt Ex4Operadores.kt Ex5Varargs.kt Aula01_Kotlin.iml External Libraries Scratches and Consoles

```

1 fun main() {
2     fun printAll(vararg messages: String) { // 1
3         for (m in messages) println(m)
4     }
5     printAll( ...messages: "Hello", "Hallo", "Salut", "Hola", "你好") // 2
6
7     fun printAllWithPrefix(vararg messages: String, prefix: String) { // 3
8         for (m in messages) println(prefix + m)
9     }
10    printAllWithPrefix( ...messages: // 4
11        "Hello", "Hallo", "Salut", "Hola", "你好",
12        prefix = "Greeting: "
13    )
14
15    fun log(vararg entries: String) { // 5
16        printAll(*entries)
17    }
18
19 main()

```

Run: Ex5VarargsKt

Z: Favorites

2: Structure

Modificador vararg

Notação Infix

- A notação infix permite não usar parênteses no momento da chamada
- Só pode ter uma parâmetro (não ser vararg e não ter valor default)
- Tem que ser função membro de classe

```
infix fun Int.shl(x: Int): Int { ... }

// calling the function using the infix notation
1 shl 2

// is the same as
1.shl(2)
```

```
Ex3FuncoesInfix.kt X
1 ► ⌂ fun main() {
2
3     infix fun Int.times(str: String) = str.repeat( n: this)           // 1
4     println(2 times "Bye ")                                            // 2
5
6     val pair = "Ferrari" to "Katrina"                                     // 3
7     println(pair)
8
9     infix fun String.onto(other: String) = Pair(this, other)           // 4
10    val myPair = "McLaren" onto "Lucas"
11    println(myPair)
```

Run: Ex3FuncoesInfixKt X

```
"/Applications/IntelliJ IDE
Bye Bye
(Ferrari, Katrina)
(McLaren, Lucas)
```

Ex3FuncoesInfix.kt

```
12
13     val sophia = Person( name: "Sophia")
14     val claudia = Person( name: "Claudia")
15     sophia likes claudia                                // 5
16     println(sophia.name + " likes " + sophia.toString())
17     println(claudia.name + " likes " + claudia.toString())
18 }
19
20 class Person(val name: String) {
21     val likedPeople = mutableListOf<Person>()
22     infix fun likes(other: Person) { likedPeople.add(other) } // 6
23     override fun toString(): String {
24         var t = ""
25         for (p in likedPeople) {
26             t += p.name
27         }
28         return t
29     }
}
```

Run: Ex3FuncoesInfixKt

McLaren, Lucas,
Sophia likes Claudia
Claudia likes