

The background features a complex network of thin grey lines and dots, forming a web-like structure. Scattered throughout are various triangles of different sizes and orientations, some with solid dots at their vertices. The overall aesthetic is modern and technical.

PROGRAMA DEV VENTURE

Desenvolvimento Android

KOTLIN
Esquece Java, só que
não

01

NULL SAFETY
Evitando NPE

02

CONDICIONAIS
When, if

03

COLLECTIONS
Lists, maps, pairs

04

LAMBDA
O que são lambdas, filter
& map

05

Aula 05 AGENDA



01


KOTLIN





KOTLIN

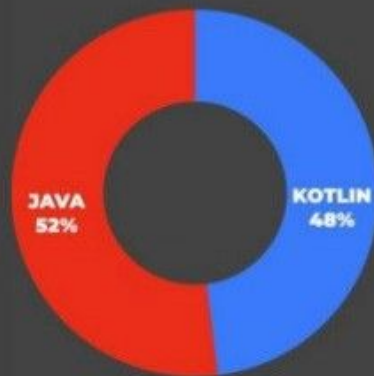
Kotlin é uma linguagem de programação multi-plataforma, estaticamente tipada e de uso geral com inferência de tipo. Kotlin é projetado para interoperar totalmente com Java, e a versão JVM da biblioteca padrão de Kotlin depende da Biblioteca de classes Java, mas a inferência de tipo permite que sua sintaxe seja mais concisa. **Em 2017 Google anunciou o suporte à Kotlin no Android**



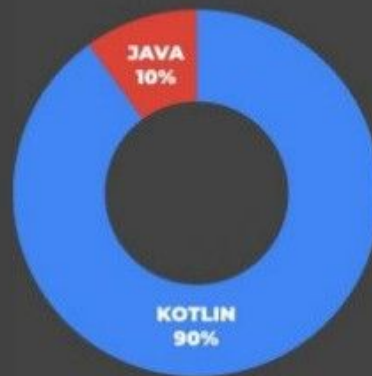
KOTLIN

Number of Android Apps in Kotlin

2018



2019



Stats from JetBrains.com

KOTLIN

KOTLIN

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        ...  
        fab.setOnClickListener { view ->  
            Snackbar.make(view, "Hello $name", Snackbar.LENGTH_LONG).show()  
        }  
    }  
}
```

Tipo Nullables e NonNull ajudam a reduzir as NullPointerExceptions

Uso de expressões lambda para criação de código conciso no trato com eventos

Uso templates para simplificar a concatenação de strings

Ponto e vírgula é opcional

```
public class Person {  
  
    private String firstName;  
    private String lastName;  
    // ...  
  
    public String getFirstName() {  
        return firstName;  
    }  
  
    public void setFirstName(String firstName) {  
        this.firstName = firstName;  
    }  
  
    public String getLastName() {  
        return lastName;  
    }  
  
    public void setLastName(String lastName) {  
        this.lastName = lastName;  
    }  
  
    @Override  
    public boolean equals(Object o) {  
        if (this == o) return true;  
        if (o == null || getClass() != o.getClass()) return false;  
        Person person = (Person) o;  
        return Objects.equals(firstName, person.firstName) &&  
            Objects.equals(lastName, person.lastName);  
    }  
  
    @Override  
    public int hashCode() {  
        return Objects.hash(firstName, lastName);  
    }  
  
    @Override  
    public String toString() {  
        return "Person{" +  
            "firstName='" + firstName + '\'' +  
            ", lastName='" + lastName + '\'' +  
            '}'  
        }  
    }  
}
```



```
data class Person(var firstName: String?, var lastName: String?)
```



dojo | VARIÁVEIS

Kotlin playground

```
var animal = gato  
val mamífero = cão  
println(animal)  
println(mamífero)
```

```
animal = "porquinho da índia"  
mamífero = "baleia"  
println(animal)  
println(mamífero)
```

O que aconteceu?





TIPAGEM ESTÁTICA

val -> valor que não pode ser atualizado
var -> variável que pode ser atualizada






dojo | VARIÁVEIS

Kotlin playground

```
var idade: Int = 1  
var qteLivros = 1  
val nome: String = "Tita"  
val especie = "Canis Familiaris"
```

```
println(nome)  
println(especie)
```





dojo | VARIÁVEIS

Kotlin playground

```
var idade = 24  
print(lidade)  
Idade = 1  
print(idade)  
Idade = "Marcella"
```

O que aconteceu?





TIPAGEM ESTÁTICA

Kotlin é uma linguagem estaticamente tipada. Isso significa que o tipo de uma variável, depois de determinado, não muda durante a execução do programa.

Kotlin oferece inferência de tipo. Por isso podemos declarar variáveis sem explicitamente escrever o tipo delas.

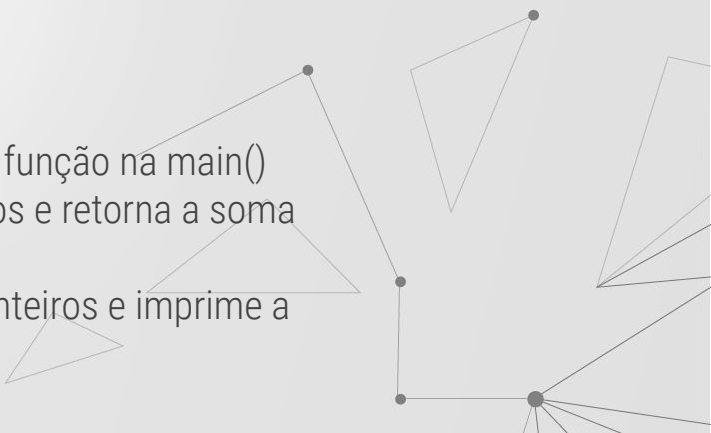




dojo | FUNÇÕES

Kotlin playground

fun start(): String =

-
1. Faça a função start retornar "OK". Chame esta função na main()
 2. Implemente a função sum, que recebe 2 inteiros e retorna a soma deles.
 3. Implemente a função printSum, que recebe 2 inteiros e imprime a soma deles na tela.
- 

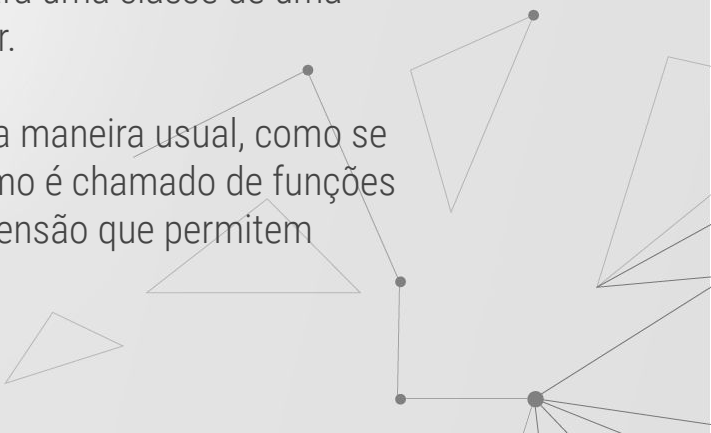


EXTENSÕES

O Kotlin oferece a capacidade de estender uma classe com novas funcionalidades sem ter que herdar da classe. Isso é feito por meio de declarações especiais chamadas extensões.

Por exemplo, você pode escrever novas funções para uma classe de uma biblioteca de terceiros que você não pode modificar.

Essas funções estão disponíveis para chamadas da maneira usual, como se fossem métodos da classe original. Esse mecanismo é chamado de funções de extensão. Existem também propriedades de extensão que permitem definir novas propriedades para classes existentes.





DESAFIO

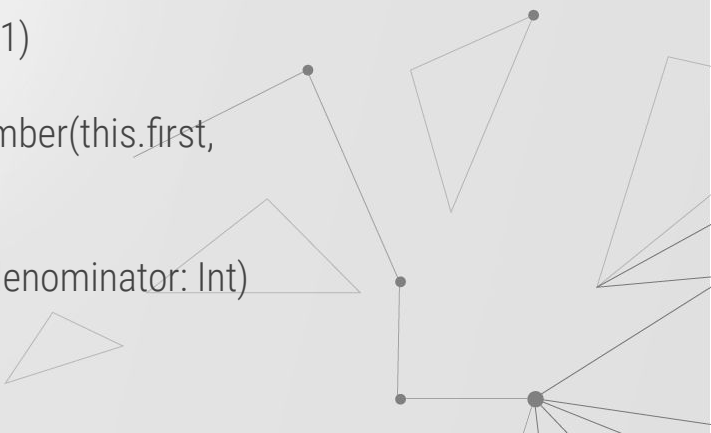
Implemente as extension functions `Int.r()` e `Pair.r()`. Elas irão converter um `Int` e um `Pair` num número racional

<https://play.kotlinlang.org/koans/Classes/Extension%20functions/Task.kt>

```
fun Int.r(): RationalNumber = RationalNumber(this, 1)
```

```
fun Pair<Int, Int>.r(): RationalNumber = RationalNumber(this.first,  
this.second)
```

```
data class RationalNumber(val numerator: Int, val denominator: Int)
```



02

NULL SAFETY





dojo | NULL SAFETY

Kotlin playground

```
var stringNaoNula: String = null  
var stringNula: String? = null
```

O que aconteceu?





dojo | NULL SAFETY

Kotlin playground

```
stringNaoNula = "Marcella"  
println(stringNaoNula.length)
```

```
stringNula.length
```

```
stringNula!!.length  
stringNulla?.legth
```

O que aconteceu?






NULL SAFETY

Uma das armadilhas mais comuns em muitas linguagens de programação, é possibilitar o acesso a um membro de uma referência nula. Esse acesso resultará em uma exceção de referência nula. Em Java, isso seria o equivalente a um `NullPointerException`.

Kotlin distingue tipos entre: referências que podem ser nulas (referências anuláveis) e aquelas que não podem (referências não nulas).

```
stringNaoNula: String  
stringNula: String?
```





dojo / ELVIS OPERATOR

```
var stringNula: String? = null  
var stringLength = stringNula?.length :? 0
```

```
println(stringLength)
```

```
stringNula = "Oi"  
stringLength = stringNula?.length :? 0  
println(stringLength)
```

O que a expressão está fazendo? `stringNula?.length :? 0`





DESAFIO

Complete o código usando apenas uma chamada if

<https://play.kotlinlang.org/koans/Introduction/Nullable%20types/Task.kt>



03

CONDICIONAIS





dojo / WHEN

```
val preco = 50
when(preco) {
    0 -> print("hoje é gratis")
    25 -> print("promoção")
    26..30 -> print("desconto fidelidade")
    31..49 -> print("desconto funcionarios")
    else -> ("preço regular")
}
```






dojo / WHEN

```
val preco = 30
when(preco) {
    0 -> print("hoje é gratis")
    !25 -> print("promoção")
    else -> ("preço regular")
}
```

```
val preco = 10
when {
    preco < 5 -> "promoção"
    preco >= 6 && preco < 8 -> "desconto"
    else -> "regular"
}
```



04

COLLECTIONS






dojo / COLLECTIONS

List<Int>
Set<Int>
Map<Int>

```
val list = listOf(1,2,3, 1,3)  
val mutableList = mutableListOf(1, 2, 3)  
mutableList[0] = 99
```

```
val set = setOf(1,1,2,3,4,5)
```

```
val mutableSet = mutableSetOf(1,2,2,3,4,5,5)
```





dojo / COLLECTIONS

```
val map = mapOf(Pair(1, "Android"), Pair(2, "Kotlin"))
```

```
val mutableMap = mutableMapOf(1 to "Android", 2 to "Kotlin", 3 to "Java")
```





DESAFIO

Resolva o desafio:

<https://www.hackerrank.com/challenges/simple-array-sum/problem>



05

LAMBDDAS





dojo / FILTER & MAP

```
val timesTwo = { x: Int -> x * 2 }  
val add: (Int, Int) -> Int = { x: Int, y: Int -> x + y }  
val toA: (Char) -> String = { letra: Char -> "$letra 01" }
```





dojo / FILTER & MAP

```
val list =(1..100).toList()
```

```
print(list.filter { element ->
    element % 2 == 0
})
```

```
list.filter {
    it % 2 == 0
}
```

```
list.filter(::isEven)
```

```
fun isEven(i: Int) = i % 2 == 0
```





dojo / FILTER & MAP


```
//map()  
val list = (1..100).toList()
```

```
val doubled = list.map { element -> element * 2 }  
list.map { it * 2 }
```

```
print(doubled)
```

```
val average = list.average()  
val shifted = list.map { it - average }
```

```
print(shifted)
```



KOTLIN

KOANS

