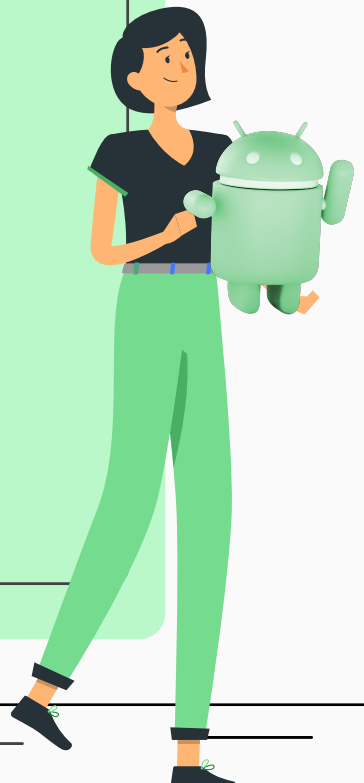
AULA 3

Kotlin



01 KOTLIN

Entendimento básico da linguagem

02 ALGORITMOS

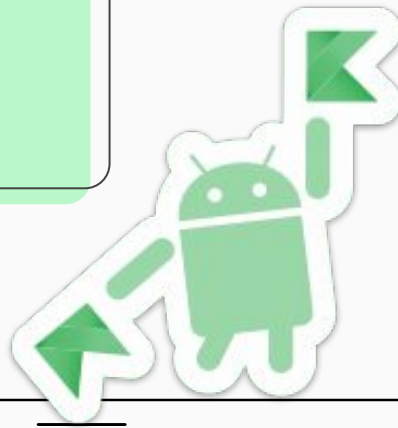
Praticar o que aprendemos de kotlin

03 LAMBDAS

Programação funcional
Métodos úteis

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

```
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

PALAVRAS CHAVE

- MULTIPLATAFORMA
- ESTATICAMENTE TIPADA
- INFERÊNCIA DE TIPO
- INTEROPERÁVEL COM JAVA
- LAMBDA

DOJO - TIPAGEM

Abrir **Kotlin REPL (read evaluate print loop)**

var idade = 24
Ctrl + enter - executar
Idade

var idade: Int = 24

val nome = "Marcella"

idade = 25

nome = "Marcella Souza"
erro

DOJO - NULL SAFE

Abrir **Kotlin REPL (read evaluate print loop)**

```
var stringNaoNula: String = null
var stringNula: String? = null
stringNula.length
stringNula!!.length
stringNulla?.legth
```

?:

DOJO - ELVIS OPERATOR

Abrir **Kotlin REPL (read evaluate print loop)**

```
var stringNaoNula: String = null
var stringNula: String? = null
stringNula.length
stringNula!!.length
stringNulla?.legth

val stringLength = stringNulla?.length :? 0
```

DOJO - CONDITIONALS

Abrir **Kotlin REPL (read evaluate print loop)**

```
val i = 21
If (i < 15) {
   print("pequeno")
} else {
   print("grande")
}

If (i < 15) {
   print("pequeno")
} else if (i >= 15 &&  i < 25) {
   print("ok")
} else {
   print("grande")
}

X = If (i < 15) {
   print("pequeno")
} else if (i >= 15 &&  i < 25) {
   print("ok")
} else {
   print("grande")
}
```

```
X
X = If (i < 15) {
   print("pequeno")
"pequeno"
} else if (i >= 15 &&  i < 25) {
   print("ok")
"ok"
} else {
   print("grande")
"grande"
}

X

val texto: String? = "Kotlin"
if(text != null) {
         String.lenght
}
```

DOJO - CONDITIONALS

Abrir **Kotlin REPL (read evaluate print loop)**

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

```
val preco = 30
when(preco) {
        0 -> print("hoje é gratis")
        25 -> print("promoção")
        10 + 20 -> print("desconto fidelidade")
        31..49 -> print("desconto funcionarias")
        else -> ("preço regular")
}
```

```
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"
 }
```

DOJO - CONDITIONALS

Abrir **Kotlin REPL (read evaluate print loop)**

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

```
val preco = 30
when(preco) {
        0 -> print("hoje é gratis")
        25 -> print("promoção")
        10 + 20 -> print("desconto fidelidade")
        31..49 -> print("desconto funcionarias")
        else -> ("preço regular")
}
```

```
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"
 }
```

DOJO - COLLECTIONS

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

MutableList<Int>

Val array = arrayOf(1, 3, 4, 11)
array.joinToString()

val list = listOf(1,2,3)
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)
```

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

DOJO - LOOPS

```
for(i in 1..10) {
     println(i)
 }

for(i in 1..10) {
     println("$i ")
 }

for(c in "Kotlin") {
     println("$c ")
 }

val list = listOf(1,2,3,4,5,6,7,8,9)
 for (i in list) {
     print(i)
 }
```

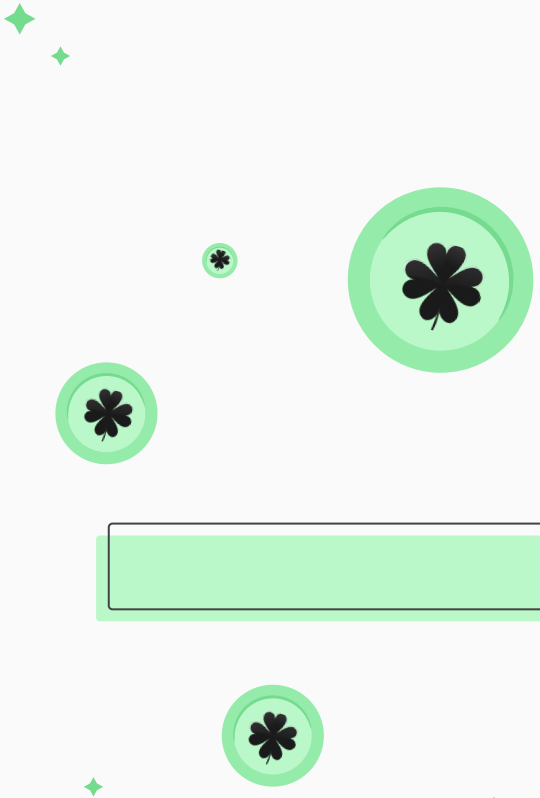
```
val listOfCities = listOf("Belo Horizonte", "São Paulo", "Rio de Janeiro")
 for (city in listOfCities) {
     print("$city é legal")
 }

val listOfCities = listOf("Belo Horizonte", "São Paulo", "Rio de Janeiro")
 for (city in listOfCities) {
     print("$city é legal\n")
 }

for(i in 10 downTo 1){
     print("$i " )
 }
```

DOJO - FUNCTIONS

```
fun permitirEntrada(idade: Int): Boolean {
    return idade >= 18
 }

 permitirEntrada(7)
res11: kotlin.Boolean = false

permitirEntrada(18)
res12: kotlin.Boolean = true

fun permitirEntrada(idade: Int): Boolean = idade >= 18

permitirEntrada(24)
res14: kotlin.Boolean = true

val permitido = permitirEntrada(12)

permitido
res16: kotlin.Boolean = false
```

```
fun permitirEntrada(vararg idades: Int): Boolean {
    return idades.any {idade -> idade >= 18}
 }

 permitirEntrada(12, 18, 13, 10)
res17: kotlin.Boolean = true
```

AQUECIMENTO



RESOLVER PROBLEMA DO TRIPLETS

KOTLIN PROJECT

- Inteli]
- Estrutura geral
- Executar main()

DESAFIO

Ler o nome do usuário no console

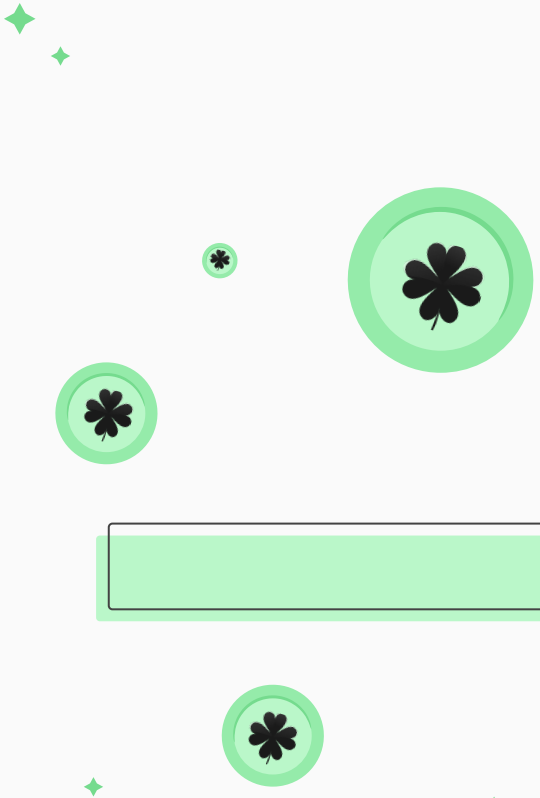
Agora que você pode criar arquivos Kotlin com uma função main (), você está pronto para alguns desafios para aplicar o que aprendeu!

- Use a função readLine () para ler uma entrada da linha de comando
- Observe o tipo de retorno de readLine () e use o que você aprendeu para trabalhar com ele
- O usuário deve inserir seu nome
- Se o usuário inserir uma string vazia, armazene um valor padrão
- Use uma expressão if para definir uma mensagem de saudação diferente com base em se o usuário inseriu um nome
- Primeiro, verifique se readLine () retorna nulo ou é uma string em branco e atribua um valor padrão.

DOJO - PARÂMETROS PADRÃO & PARÂMETROS NOMEADOS

```
fun main(args: Array<String>) {
    val together = concat(listOf("Kotlin", "Java", "Scala"), ": ")
    print(together)
}

fun concat(texts: List<String>, separator: String = ", ") =
texts.joinToString(separator)
```

```
fun main(args: Array<String>) {
    val together = concat(separator = ": ", texts = listOf("Kotlin",
"Java", "Scala"))
    print(together)
}

fun concat(texts: List<String>, separator: String = ", ") =
texts.joinToString(separator)
```



PROGRAMAÇÃO FUNCIONAL

PALAVRAS CHAVE

- IMMUTABILITY
- STATELESS
- RECURSÃO
- HIGH ORDER FUNCTIONS
- LAZY EVALUATIONS

DOJO - FILTER & MAP

```
fun main(args: Array<String>) {
    val timesTwo = { x: Int -> x * 2 }
    val add: (Int, Int) -> Int = { x: Int, y: Int -> x + y }
    val toA: (Char) -> String = {letra: Char -> "$letra OI"}

    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
```

```
fun main(args: Array<String>) {

    //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)

}
```

DOJO - TAKE, DROP & ZIP

```
fun main(args: Array<String>) {
    //take()
    val list =(1..100).toList()
    val takeFirst10 = list.take(10)
    print(takeFirst10)

    //drop()
    val drop10First = list.drop(10)
    print(drop10First)
}
```

```
fun main(args: Array<String>) {
    val list = listOf("hi", "there", "kotlin", "fans")
    val containsT = listOf(false, true, true, false)

    val zipped: List<Pair<String, Boolean>> = list.zip(containsT)
    val mapping = list.zip(list.map { it.contains("t")})

    print(zipped)
    print(mapping)

}
```

DESAFIO - Análise de dados