SENAI-SP

Kotlin

Funções de Escopo

Funções de Escopo

As funções de escopo permitem a execução de um bloco de código no contexto de um objeto através de uma expressão lambda fornecida. Existem 5 funções de escopo: **apply**, **also**, **let**, **run** e **with**.

Função de Escopo	Faz referência ao objeto	Retorna
apply	this	O objeto
also	it	O objeto
let	it	Última declaração
run	this	Última declaração
with	this	Última declaração

Funções de Escopo

A utilização do **apply** é normalmente para blocos de código que não retornam um valor e operam principalmente nos membros do objeto receptor.

```
data class Pessoa(
  val nome: String,
  var idade: Int = 0,
  var cargo: String = ""
)
fun main() {
  val empregado = Pessoa("João").apply {
    idade = 25
    cargo = "Desenvolvedor"
  }
  println(empregado)
}
```

Funções de Escopo

O **also** é bom para executar algumas ações que tomam o objeto de contexto como argumento e também para ações que precisam de uma referência ao objeto e não a suas propriedades e funções, ou quando não desejar sombrear essa referência de um escopo externo.

```
fun printErr(mensagem: Any?) = System.err.println(mensagem)

fun inteiroAleatorio() : Int {
  return Random.nextInt().also {
    printErr("A função inteiroAleatorio() gerou o valor $it")
  }
}

fun main() {
  val numero = inteiroAleatorio()
  println("O valor inteiro é $numero")
}
```

Funções de Escopo

SENAI-SP

O **let** pode ser usado para invocar uma ou mais funções nos resultados das cadeias de chamadas, para executar um bloco de código apenas com valores não nulos e também para a introdução de variáveis locais com um escopo limitado para melhorar a legibilidade do código.

```
fun String.prompt() : String = JOptionPane.showInputDialog(this)

fun main() {
  val nome = "Informe seu Nome".show()

  // Teste padrão que verifica se o nome é nulo
  if(nome != null) { print(nome) }

  // O bloco LET só é executado se o nome não for nulo
  nome?.let { print(it) }
  // O bloco LET também retorna o valor do objeto contido no bloco
  val tamanho : Int = nome?.let { it.length }
}
```

Funções de Escopo

A utilização de **run** é útil quando o bloco lambda contém a inicialização do objeto e o cálculo do valor de retorno. Também pode ser usado como uma função de não extensão. A execução sem extensão permite executar um bloco de várias instruções em que uma expressão é necessária.

```
data class Ponto (val x: Float, val y: Float) {
  fun distanciaEntre(a: Ponto, b: Ponto) : Float {
    val dx = a.x - b.x
    val dy = a.y - b.y
    return sqrt(dx * dx + dy * dy)
  }
}
fun main() {
  val local1 = Ponto(3f, 5f)
  println(local1.run {
    val local2 = Ponto(5f, 10f)
    distancia(local2)
  })
}
```

Funções de Escopo

SENAI-SP

Na função **with** o objeto de contexto é passado como argumento, mas dentro da expressão lambda, está disponível como um receptor (**this**). O valor de retorno é o resultado lambda.

```
fun String.prompt() : String = JOptionPane.showInputDialog(this)

fun String.show() = JOptionPane.showMessageDialog(null, this)

fun main() {
  // No bloco WITH objeto é referenciado com THIS
  with( "Informe seu Nome".show()) {
    "Bem Vindo, $this".show()
  }
}
```

Funções de Escopo

Como escolher qual função de escopo utilizar