08

Para saber mais: Operador invoke

Para inicializar uma variável do tipo função com um objeto, implementamos uma função na classe. Ao implementar a função, sobrescrevemos o método `invoke()` e, como foi mencionado, ele é um "operador especial". Esse operador é conhecido como sobrecarga de operador ou *operator overloading* (https://kotlinlang.org/docs/reference/operator-overloading.html).

Isso significa que não precisamos implementar um tipo função para obter um `invoke()` em uma classe, podemos também adicionar como um overloading operator:

```
class Teste {
    operator fun invoke() {
        println("executa invoke do Teste")
    }
}
```

Dessa maneira, uma variável do tipo `Teste`, pode chamar o operador `invoke` a partir do operador `()`:

```
val teste = Teste()
teste()
```

Nesta implementação não temos mais compatibilidade com o tipo função `() -> Unit`, pois só é possível a compatibilidade por meio da implementação do tipo como fizemos, porém, podemos escrever mais de um operador `invoke`:

```
class Teste : () -> Unit {

    operator fun invoke(valor: Int){
        println(valor)
    }

    override fun invoke() {
        println("executa invoke do Teste")
    }

}
```

Dessa forma, podemos inicializar variáveis do tipo `() -> Unit`, como também manter o comportamento de execução em variáveis do tipo `Teste`:

```
val teste = Teste()
teste(10)
val testeFuncao: () -> Unit = Teste()
testeFuncao()
```