



Tem algumas maneiras de declarar *lambda expressions* em Kotlin, um jeito é deixá-lo inferir o tipo do retorno da *lambda*, mas em contra partida devemos explicitar o que ela recebe como parâmetro:

```
val funcaoLambdadComCorpoExplicito: (String, Int) -> Boolean = { senha, numero ->
    senha == "Senha" && numero == 1
}

val funcaoLambdaComCorpoInferido= {senha : String, numero: Int ->
    senha == "Senha" && numero == 1
}
```

*****Função anônima é exatamente o mesmo esquema da lambda expression, só que mais verbosa e com necessidade de explicitar com "fun" a declaração, os tipos e o retorno*****

Expressões lambda também podem ter mais de um retorno. Por default a *lambda expression* sempre retorna a última instrução, mas para que ela retorne em um determinado ponto podemos usar o recurso de "label". Labels são basicamente um nome que pode ser definido pra uma lambda para explicitar um retorno em algum ponto diferente da última instrução:

```
val calculaBonus : (salario : BigDecimal) -> BigDecimal = bonus@ { it: BigDecimal
    if(it.compareTo(BigDecimal( val: "1600")) == 1){
        return@bonus it.plus(BigDecimal( val: "100"))
    }

    return@bonus it.plus(BigDecimal( val: "178"))
}
```