Fundamentos do Desenvolvimento Android

Prof. Thiago Vieira de Aguiar

Roteiro

- **Introdução a Kotlin**
  - Entry Point
  - Variáveis
  - Controle de Fluxo
  - Funções
  - Classes

Entry Point

Entry Point

Função **main** é a primeira a ser chamada ao se iniciar o programa

```
fun main(args: Array<String>) {
    if (args.size == 0) {
        println("Please provide a nar
        return
    }
    println("Hello, ${args[0]}!")
}
```

Variáveis

Variáveis

- val e var

```
fun main(){
    var numero = 10 // mutavel
    val outroNumero = 20 // imutavel
    println("Numeros: ${numero} / ${outroNumero}")
}
```

Variáveis

- Tipos / Numérico

Numbers

Kotlin provides a set of built-in types that represent numbers.

For integer numbers, there are four types with different sizes and, hence, value ranges.

Type	Size (bits)	Min value	Max value
Byte	8	-128	127
Short	16	-32768	32767
Int	32	-2,147,483,648 (-2^{31})	2,147,483,647 ($2^{31} - 1$)
Long	64	-9,223,372,036,854,775,808 (-2^{63})	9,223,372,036,854,775,807 ($2^{63} - 1$)

Variáveis

- Tipos / Numérico

```
fun main(){
    var nByte: Byte = 127
    var nShort: Short = 32767
    var nInt: Int = 2147483647
    var nLong: Long = 2999999999
    println("Byte: ${nByte} \n"+
            "Short: ${nShort} \n"+
            "Int: ${nInt} \n"+
            "Long: ${nLong} \n"
        )
}
```

Variáveis

- Tipos / Numérico

Type	Size (bits)	Significant bits	Exponent bits	Decimal digits
Float	32	24	8	6-7
Double	64	53	11	15-16

Variáveis

- Tipos / Numérico

```
fun main(){
    var nDouble = 3.8
    var nFloat = 3.8f
    println("Double: ${nDouble} \n"+
        "Float: ${nFloat} \n"
    )
}
```

Variáveis

- Tipos / Numérico / Conversão

Every number type supports the following conversions:

- `toByte(): Byte`
- `toShort(): Short`
- `toInt(): Int`
- `toLong(): Long`
- `toFloat(): Float`
- `toDouble(): Double`
- `toChar(): Char`

Variáveis

- Tipos / Numérico / Operadores

```
var opndA = 13
var opndB = 3
println(opndA + opndB)   // Soma
println(opndA - opndB)   // Subtracao
println(opndA / opndB)   // divisao
println(opndA * opndB)   // Multiplicacao
println(opndA % opndB)   // Modulo
```

Variáveis

- Tipos / Numérico / Operadores

```
var opndA = 13
var opndB = 3
println(opndA == opndB) // Igualdade
println(opndA != opndB) // Diferenca
println(opndA > opndB)  // Maior que
println(opndA < opndB)  // Menor que
println(opndA >= opndB) // Maior ou igual
println(opndA <= opndB) // Menor ou igual
```

Variáveis

- Tipos / Texto

Strings

Strings are represented by the type `String`. Strings are immutable. Elements of a string are characters that can be accessed by the indexing operation: `s[i]`. A string can be iterated over with a `for`-loop:

```
for (c in str) {
    println(c)
}
```

Variáveis

- Tipos / Texto

```
fun main(){
    var tChar = 't'
    var tString= "Thiago"
    println("Char: ${tChar} \n"+
        "String: ${tString} \n"
    )
}
```

Variáveis

- Tipos / Booleano

Booleans

The type `Boolean` represents booleans, and has two values: `true` and `false`.

Variáveis

- Tipos / Booleano

```
fun main(){
    var bTrue = true
    var bFalse = false
    println("True: $bTrue \n"+
        "False: $bFalse \n"
    )
}
```

Variáveis

- Tipos / Booleano

```
var bollA = true
var bollB = true
var bollC = false
var bollD = false
println(bollA && bollB) // Conjuncao
println(bollA && bollC) // Conjuncao
println(bollA || bollC) // Disjuncao
println(bollD || bollC) // Disjuncao
println(!bollA)         // Negacao
```

Variáveis

- Tipos / Array

```
var aList = arrayOf(1,2,3)
var aFunc = arrayOf(
    "Thiago",
    "Paulo",
    "Maria")
println("$aFunc")
aList.forEach{
    println("$it")
}
```

Controle de Fluxo

Controle de Fluxo

IF

```
if (condicao) {

} else {

}
```

```
fun main(args: Array<String>) {
    var idade = 17
    if (idade > 17){
        println("Acesso permitido.")
    } else {
        println("Acesso negado.")
    }
}
```

```
val menor = if (idade < 18) true else false
```

Controle de Fluxo

IF

```
if (condicao) {

} else if (condicao) {

}
```

```
if (idade < 18){
    println("Acesso negado.")
} else if (idade < 65){
    println("Acesso permitido.")
} else {
    println("Acesso prioritario.")
}
```

Controle de Fluxo

WHEN

```
when (var) {
 X -> op1
 Y -> op2
 else -> opDefault
}
```

```
fun main(args: Array<String>) {
    var valor = 100
    var imposto = "itbi"
    when (imposto){
        "icms" ->
            println("${valor + valor*0.10}")
        "itbi" ->
            println("${valor + valor*0.15}")
        else ->
            println("${valor + valor*0.05}")
    }
}
```

Controle de Fluxo

FOR

```
for (item in col) {

}
```

```
fun main(args: Array<String>) {
    for (num in 1..10) {
        println(num)
    }
}
```

Controle de Fluxo

WHILE

while (condicao)

}

```
fun main(args: Array<String>) {
    var cont = 10
    while (cont > 0) {
        println(cont)
        cont--
    }
}
```

Controle de Fluxo

DO ... WHILE

do {

} while (condicao)

```
var cont = 10
do {
    println(cont)
    cont--
} while (cont > 0)
```

Controle de Fluxo

BREAK

Interrompe o fluxo

```
fun main(args: Array<String>) {
    for (num in 1..10) {
        if (num == 8)
            break
        println(num)
    }
}
```

Controle de Fluxo

CONTINUE

Passa para o próximo passo da iteração

```
fun main(args: Array<String>) {
    for (num in 1..10) {
        if (num == 8)
            continue
        println(num)
    }
}
```

Funções

Funções

- Declaradas com **fun**
- Nome
- Parâmetros
- Escopo
- [Retorno]

Function declarations

Functions in Kotlin are declared using the `fun` keyword:

```
fun double(x: Int): Int {
    return 2 * x
}
```

Funções

- Declaradas com **fun**
- Nome
- Parâmetros
- Escopo
- [Retorno]

```
fun imprimir() {
    println("Uma funcao")
}
```

```
fun imprimir(texto: String) {
    println(texto)
}
```

Funções

- Declaradas com **fun**
- Nome
- Parâmetros
- Escopo
- [Retorno]

```
fun imprimir(
    nome: String,
    sobrenome: String) : String{
    return "$nome $sobrenome"
}
```

```
fun imprimir(
    nome: String = "Nome",
    sobrenome: String = "Sobrenome") : String{
    return "$nome $sobrenome"
}
```

Funções

- Declaradas com **fun**
- Nome
- Parâmetros
- Escopo
- [Retorno]

```
fun imprimir(
    nome: String = "Nome",
    sobrenome: String = "Sobrenome") : String{
    return "$nome $sobrenome"
}
```

```
var nomeCompleto = imprimir(sobrenome = "Silva Santos")
println(nomeCompleto)
```

Dúvidas?

Referências

- https://kotlinlang.org/