

# Mobile Application Development

Prof. Fernando Pinéo

# Introdução a linguagem de programação Kotlin Parte I

# Sobre a linguagem Kotlin

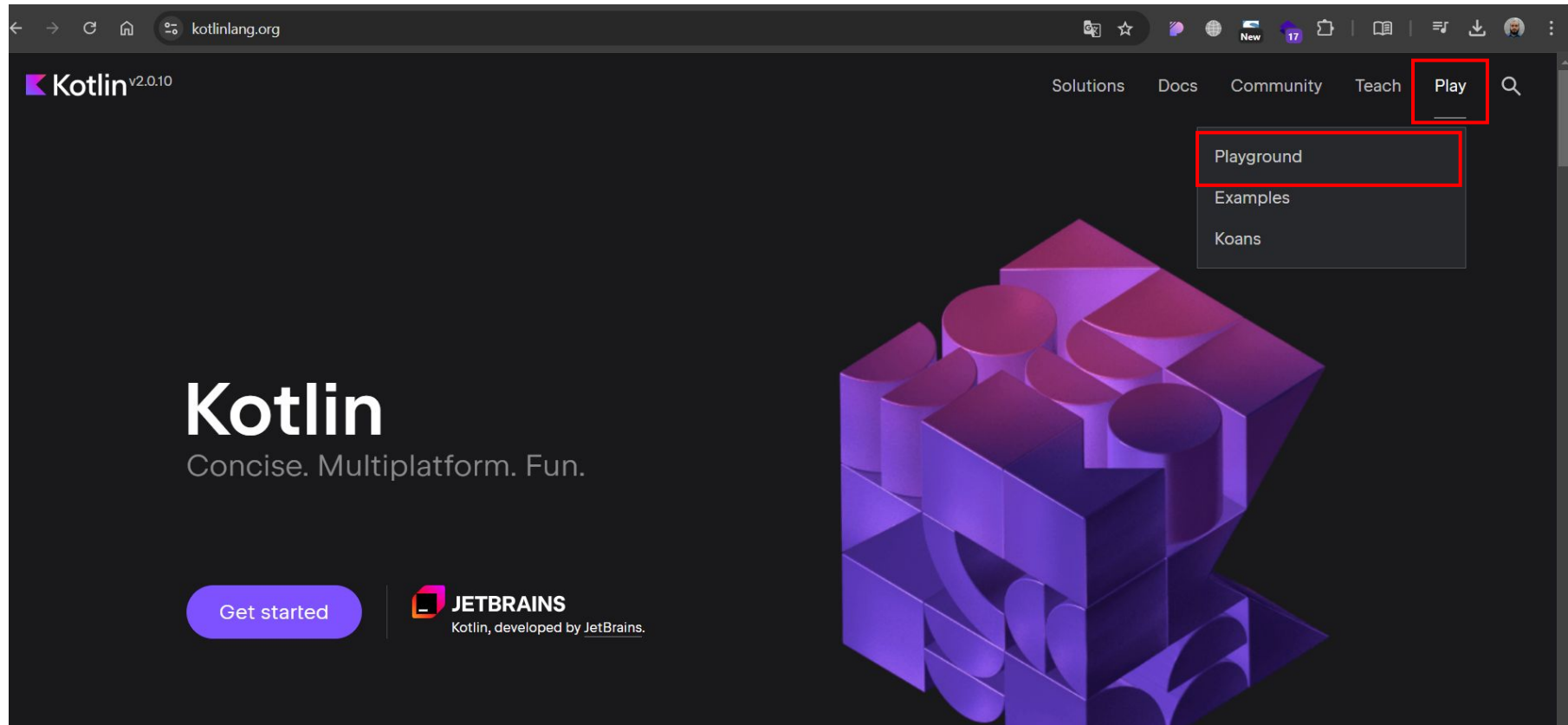
- Criada pela JetBrains
  - IntelliJ, PhpStorm, etc
  - Construída para ser usada internamente
  - Baseadas nas linguagem: Java, Scala, Groovy e C#
- Criada para focar no ecossistema da Linguagem Java
  - Android -> Utiliza JVM(Java Virtual Machine)

# Porque utilizar Kotlin

- Concisa
- Segura
- Interoperável
  - Utilize códigos Java dentro do Kotlin e vice-versa
  - Migração mais tranquila

Iniciando nossos códigos...

# IDE Online – Kotlinlang



The screenshot shows the Kotlinlang website (kotlinlang.org) with a dark theme. The browser's address bar displays 'kotlinlang.org'. The website header includes the Kotlin logo (v2.0.10) and navigation links: Solutions, Docs, Community, Teach, and Play. The 'Play' link is highlighted with a red box, and its dropdown menu is open, showing 'Playground', 'Examples', and 'Koans'. The 'Playground' option is also highlighted with a red box. The main content area features the text 'Kotlin Concise. Multiplatform. Fun.' and a 'Get started' button. The JetBrains logo and text 'Kotlin, developed by JetBrains.' are visible at the bottom left. A large, stylized 3D graphic of geometric shapes is on the right side.

Kotlin v2.0.10

Solutions Docs Community Teach **Play** 🔍

- Playground
- Examples
- Koans

**Kotlin**  
Concise. Multiplatform. Fun.

Get started

**JETBRAINS**  
Kotlin, developed by JetBrains.

# Variáveis

# O que é uma variável

- Área de memória associada a um nome, que pode armazenar valores de um determinado tipo.





# Como definir variáveis em Kotlin

- var nome (mutável)
- val pi (imutável)

Exemplo:

nome	Mariana
pi	3,14

# Exemplo, declaração de variáveis

 Kotlin

2.0.10 ▾

JVM ▾

Program arguments

```
fun main() {  
    var nome = "Fernando"  
    println("Olá " + nome)|  
}
```

```
// Comentário em uma única linha
```

```
/*
```

```
 * Comentários de múltiplas linhas
```

```
 *
```

```
 *
```

```
 * */
```

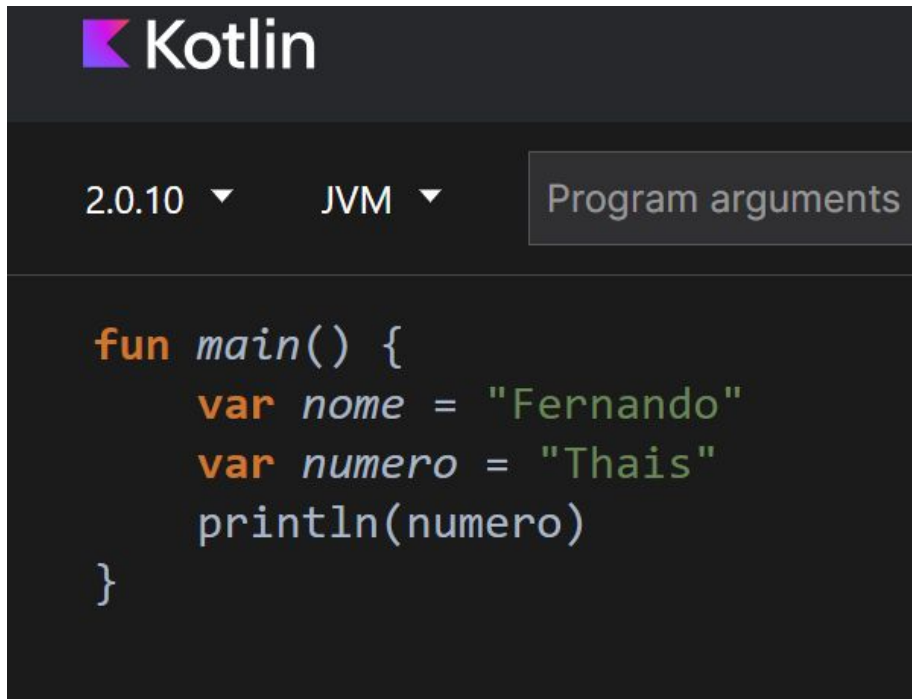
## Tipos de variáveis(números)

Tipo	Armazenamento	Exemplo
Double	1.7e-308 até 1.7e+308	val decimal = 3.5
Float	3.4e-038 até 3.4e+038	val float = 3.5f
Long	-9223372036854775808 até 9223372036854775807	val long = 652636562L
Int	-2147483648 até 2147483647	val int = 65365

# Tipos de variáveis(outros)

Tipo	Exemplo
String	<code>val string = "Fernando"</code>
Boolean	<code>val boolean = true</code>

# O Kotlin reconhece o tipo de variáveis automaticamente

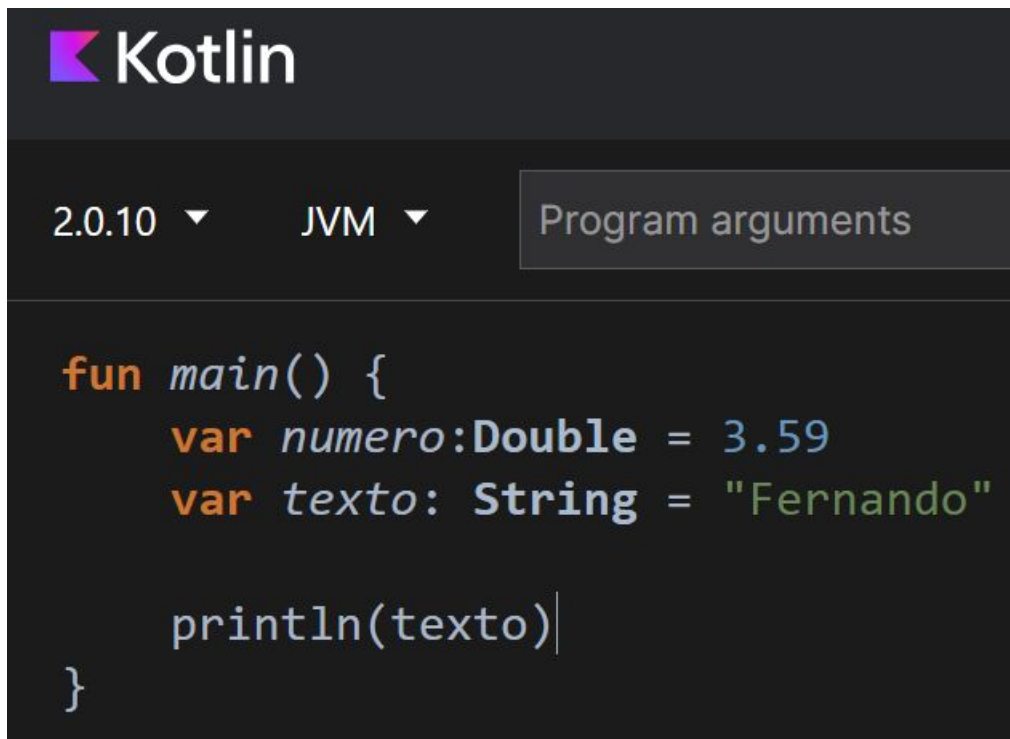


```
Kotlin

2.0.10 ▼  JVM ▼  Program arguments

fun main() {
    var nome = "Fernando"
    var numero = "Thais"
    println(numero)
}
```

Especificando o tipo de dados que a variável  
irá armazenar..

A screenshot of the Kotlin IDE interface. At the top, the Kotlin logo is displayed. Below it, there are dropdown menus for the version (2.0.10) and the target JVM (JVM). To the right of these is a button labeled 'Program arguments'. The main area shows a Kotlin code snippet for a main function. The code declares two variables: 'numero' of type 'Double' with a value of 3.59, and 'texto' of type 'String' with a value of 'Fernando'. The 'texto' variable is then printed to the console using 'println(texto)'.

# Array em Kotlin





# Criação de um array em Kotlin

```
fun main() {  
    var nomes = arrayOf("Fernando", "Ellen", "Pedro")  
    println(nomes)  
}
```

```
[Ljava.lang.String;@6e8cf4c6
```

# Criação de um array em Kotlin

```
fun main() {  
    var nomes = arrayOf("Fernando", "Ellen", "Pedro")  
  
    println(nomes[2])  
}
```

Pedro

# Subscrivendo um valor no array

```
fun main() {  
    var nomes = arrayOf("Fernando", "Ellen", "Pedro")  
  
    nomes[2] = "Maria"  
  
    println(nomes[2])  
}
```

Análise o seguinte cenário, o q ocorrerá?

```
fun main() {  
    var nomes = arrayOf("Fernando", "Ellen", "Pedro")  
  
    nomes[3] = "João"  
  
}
```

Resultado.. Será exibido um erro, devido ao tamanho do array

```
fun main() {  
    var nomes = arrayOf("Fernando", "Ellen", "Pedro")  
  
    nomes[3] = "João"  
  
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3  
    at FileKt.main (File.kt:4)  
    at FileKt.main (File.kt:-1)  
    at jdk.internal.reflect.NativeMethodAccessorImpl.invoke0 (:-2)
```

Tbm é possível armazenar vários tipos no array

```
fun main() {  
    var nomes = arrayOf(530,"Ellen","Pedro")  
  
    println(nomes[0])  
  
}
```

530

# Criando um array de números inteiros

```
fun main() {  
  
    var numero = intArrayOf(23,490,49)  
  
    println(numero[2])  
  
}
```

49



# Funções

# Funções

- Definindo funções simples

```
fun mostrarMsg(){  
    println("Exibindo mensagem")  
}  
  
fun main() {  
    mostrarMsg()  
}
```

# Funções com parâmetro

```
fun mostrarMsg(nome:String){  
    println("Exibindo mensagem para $nome")  
}  
  
fun main() {  
    mostrarMsg("Fernando")  
}
```

# Função com retorno

- No exemplo, o tipo de retorno da função somar é um retorno no tipo inteiro..

```
fun somar(valorA:Int , valorB:Int):Int{  
    var total = valorA+valorB  
    return total  
}  
  
fun main() {  
    println(somar(10,10))  
}
```

# Classes e objetos

# Classes e objetos

```
class Casa{  
    var cor:String=""  
  
}  
  
fun main() {  
    val casa01 = Casa()  
    casa01.cor = "Amarela"  
    println(casa01.cor)  
}
```

# Classes e objetos

```
class Casa{  
    //Propriedades  
    var cor:String=""  
  
    //Métodos  
    fun abrirJanela(){  
        println("Abrir janela..")  
    }  
  
}  
  
fun main() {  
    val casa01 = Casa()  
    casa01.cor = "Amarela"  
    println(casa01.cor)  
    casa01.abrirJanela()  
}
```

# Construtores



# Criando classe Casa

```
class Casa{  
    //Propriedades  
    var cor: String = ""  
    var vagasGaragem = 0  
  
    //Métodos  
    fun detalhesCasa(){  
        println("Cor da casa: $cor - Vagas: $vagasGaragem")  
    }  
}  
  
fun main() {  
  
    val casa01 = Casa()  
    casa01.cor = "Verde"  
    casa01.vagasGaragem = 2  
  
    casa01.detalhesCasa()  
}
```

# Construtor primário

```
class Casa(cor:String , vagasGaragem:Int){  
    //Propriedades  
    var cor: String  
    var vagasGaragem:Int  
  
    init{  
        this.cor = cor  
        this.vagasGaragem = vagasGaragem  
    }  
    //Métodos  
    fun detalhesCasa(){  
        println("Cor da casa: $cor - Vagas: $vagasGaragem")  
    }  
}  
fun main() {  
  
    val casa01 = Casa("Verde", 2)  
  
    casa01.detalhesCasa()  
}
```

# Outra forma de se declarar o construtor FIAP

```
class Casa{
    //Propriedades
    var cor: String
    var vagasGaragem: Int

    constructor(cor: String , vagasGaragem: Int){
        this.cor = cor
        this.vagasGaragem = vagasGaragem
    }

    //Métodos
    fun detalhesCasa(){
        println("Cor da casa: $cor - Vagas: $vagasGaragem")
    }
}

fun main() {

    val casa01 = Casa("Verde", 2)

    casa01.detalhesCasa()
}
```

# Herança

# Herança, exemplo:

```
open class Animal () {  
    fun dormir() {  
        println("dormindo...")  
    }  
}  
  
class Cao(): Animal() {  
    fun latir() {  
        println("Latindo..")  
    }  
}  
  
fun main () {  
    var cao1 = Cao()  
    cao1.latir()  
    cao1.dormir()  
}
```

## Prática...

- Desenvolva 03 Classes para o desenvolvimento do sistema de gerenciamento de alunos da FIAP.
- Umas dessas classes deve ser uma classe Pai.
- Logo teremos 03 entidades(Professor, Aluno e Coordenador)
- No mínimo 03 métodos diferente para cada entidade.

Dúvidas?