

Vitor Amadeu Souza

Introdução ao

Kotlin

Parte III

© 2025 by Cerne Tecnologia e Treinamento Ltda.

© 2025 by Vitor Amadeu Souza

Nenhuma parte desta publicação poderá ser reproduzida sem autorização prévia e escrita de **Cerne Tecnologia e Treinamento Ltda.** Este livro publica nomes comerciais e marcas registradas de produtos pertencentes a diversas companhias. O editor utiliza as marcas somente para fins editoriais e em benefício dos proprietários das marcas, sem nenhuma intenção de atingir seus direitos.

Fevereiro de 2025

Direitos reservados por:

Cerne Tecnologia e Treinamento Ltda

Produção: Cerne Tecnologia e Treinamento

E-mail da Empresa: cerne@cerne-tec.com.br

Home Page: www.cerne-tec.com.br.com.br

Atendimento ao Consumidor: sac@cerne-tec.com.br

Contato com o Autor: vitor@cerne-tec.com.br



FEITO NO BRASIL

“Se quer ir rápido, vá sozinho. Se quer ir longe, vá acompanhado.”

Provérbio africano

Cerne Tecnologia

A Cerne Tecnologia tem uma equipe preparada para desenvolvimento de projetos eletrônicos em diversas áreas: Médica, Entretenimento, Industrial, Robótica, Científica, Automobilística, Aeronáutica, etc. Trabalhamos com tecnologia microcontrolada usando o PIC, ARM, AVR, 8051, dsPIC, PIC24, PIC32 além do Arduino, Raspberry, Beaglebone etc. Desenvolvemos o projeto desde sua concepção até a entrega do produto final, passando pelas etapas de esquema elétrico, protótipo e desenvolvimento de circuito impresso.

Desenvolvemos aplicativos para smartphones/tablets Android, iOS, Blackberry, Windows Phone e no desenvolvimento de softwares a nível PC para plataforma Windows, usando ferramentas como o Visual Basic, C# e C++.

Atuamos na parte de montagem de placas, onde podemos fornecer ambos os serviços de desenvolvimento de projetos e produção ou apenas um destes.

Desenvolvemos esquemas elétricos e layout de PCI, tanto em tecnologia convencional como SMD.

Temos a flexibilidade de customizar um de nossos produtos, de modo a atender a uma necessidade específica do cliente, tornando o custo de desenvolvimento menor se comparado a construção de um projeto desde a sua fase inicial.

Desenvolvemos e fornecemos kits didáticos para diversos microcontroladores além de apostilas, livros e e-books.

Na hora de desenvolver um projeto ou equipar seu laboratório não hesite em nos contatar. Entre em contato conosco através do endereço cerne-tec.com.br para obter mais informações.



Sumário

Capítulo I – Metodologia de desenvolvimento	6
1. Introdução	6
 Capítulo II – Programação em Kotlin	 7
1. Classe Simples	7
2. Herança em Classes	8
3. Data Class	10
4. Try-Catch (Tratamento de Exceção)	11
5. Nullable Types e Elvis Operator	13
6. Corrotinas Simples	14
7. Função de Extensão	16
8. Operador Spread	17
9. Funções de Coleções	18
10. flatMap	20

Capítulo I

Metodologia de desenvolvimento

1. Introdução

Este livro tem como objetivo ensinar a programação utilizando Kotlin, explorando seus recursos de forma prática e aplicada. Ao longo desta série, diversas aplicações serão desenvolvidas, permitindo uma compreensão aprofundada da linguagem e de suas principais funcionalidades. Cada volume abordará temas específicos, conforme detalhado no sumário, proporcionando um aprendizado progressivo e estruturado.

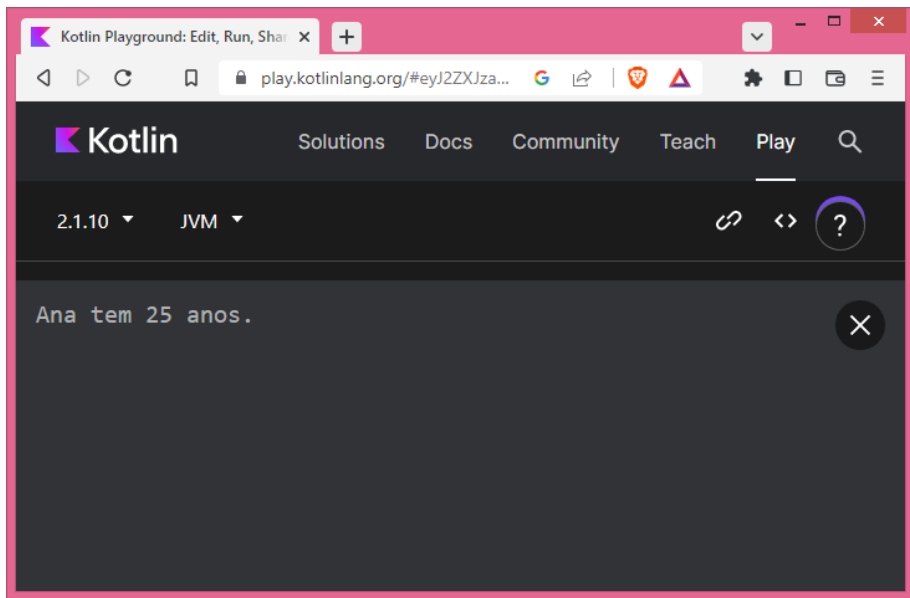
Capítulo II

Programação em Kotlin

1. Classe Simples

O código define uma classe em Kotlin chamada Pessoa, que possui duas propriedades: nome, do tipo String e imutável (val), e idade, do tipo Int e mutável (var). Dentro da função main(), é criada uma instância da classe Pessoa com o nome "Ana" e idade 25. Em seguida, a saída println("\${p.nome} tem \${p.idade} anos.") exibe "Ana tem 25 anos.", utilizando interpolação de strings para acessar os atributos do objeto. Esse exemplo demonstra como definir classes e instanciar objetos em Kotlin de forma simples e concisa.

```
fun main() {  
  
    class Pessoa(val nome: String, var idade: Int)  
  
    val p = Pessoa("Ana", 25)  
  
    println("${p.nome} tem ${p.idade} anos.")  
  
}
```

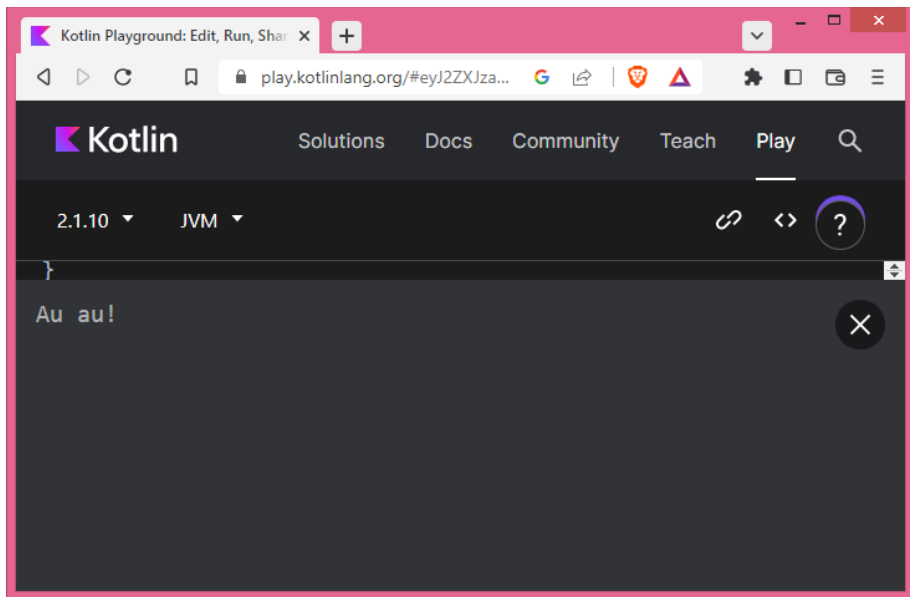


2. Herança em Classes

O código demonstra o conceito de herança e polimorfismo em Kotlin. A classe `Animal` é definida como `open`, permitindo que outras classes a herdem. Ela possui um atributo `nome` e um método `fazerSom()`, também marcado como `open`, permitindo sua sobrescrita (`override`). A classe `Cachorro` herda de `Animal` e sobrescreve o método `fazerSom()`, retornando "Au au!" em vez de "Som genérico". Dentro da `main()`, um objeto `dog` da classe `Cachorro` é criado e seu método `fazerSom()` é chamado, imprimindo "Au au!". Esse código

ilustra como Kotlin permite a reutilização e personalização de comportamentos em classes por meio de herança e sobrescrita de métodos.

```
fun main() {  
    open class Animal(val nome: String)  
    {  
        open fun fazerSom() = "Som genérico"  
    }  
  
    class Cachorro : Animal("Cachorro")  
    {  
        override fun fazerSom() = "Au au!"  
    }  
  
    val dog = Cachorro()  
    println(dog.fazerSom())  
}
```



3. Data Class

O código define uma data class chamada `Usuario` em Kotlin, que possui duas propriedades imutáveis: `nome`, do tipo `String`, e `idade`, do tipo `Int`. As data classes são úteis para representar dados e fornecem automaticamente métodos como `toString()`, `equals()`, `hashCode()` e `copy()`. Em seguida, um objeto `u` é criado com os valores `"Carlos"` e `30`. Quando `println(u)` é chamado, o método `toString()` gerado automaticamente exibe `"Usuario(nome=Carlos, idade=30)"`.