

Ronin do Java



Um guerreiro sem mestre que
aprende por si só.

Angelo Lunardini

Um guerreiro sem mestre
que aprende por si só.

Generico aikidksakiikfaikgikikasgkidkiaiksik
alsidiladliasldiglaoildoalogsalogloagdlosolaglosadgloa

Capitulo 01

Começando

O Que é Java?

Uma breve história e a importância da linguagem no mundo da tecnologia.

Pense no Java como um idioma universal para computadores. Criado nos anos 90, seu objetivo era ser uma linguagem robusta e segura que pudesse ser usada para criar programas complexos. Hoje, ela é uma das linguagens mais populares do mundo, conhecida por sua confiabilidade e desempenho.

Java na Prática

Onde a linguagem é usada?

De aplicativos Android a sistemas de grandes empresas.

Você encontra Java em todos os lugares! A maioria dos aplicativos para Android é feita com Java. Grandes empresas, como bancos e lojas online, usam Java para construir seus sistemas seguros e de alta performance. Até mesmo jogos como Minecraft foram criados com Java.

O Lema "WORA"

Entendendo o conceito de
"Write Once, Run Anywhere"
(Escreva uma vez, execute em
qualquer lugar) e a JVM.

O grande superpoder do Java é o lema "WORA". Isso significa que você escreve seu código uma única vez e ele pode rodar em qualquer sistema operacional (Windows, Mac, Linux) sem precisar de alterações. A mágica acontece graças à JVM (Java Virtual Machine), uma espécie de "tradutor" que adapta seu código para o sistema em que ele está sendo executado.

Capitulo 02

Preparação

Kit de Ferramentas

Instalando o JDK (Java Development Kit) passo a passo.

Para começar a programar em Java, você precisa do JDK. Ele é um kit que traz todas as ferramentas necessárias para escrever e executar seus programas. A instalação é simples: basta baixar a versão mais recente do site oficial da Oracle (ou uma alternativa como o OpenJDK) e seguir as instruções do instalador.

Ambiente Configurado

Configurando as variáveis de ambiente para que o Java funcione em seu computador.

Depois de instalar o JDK, você precisa "avisar" ao seu computador onde ele foi instalado. Isso é feito configurando as variáveis de ambiente, como `JAVA_HOME` e `PATH`. Essa configuração permite que você execute comandos Java de qualquer pasta no seu terminal.

Escolhendo sua IDE

O que é um Ambiente de Desenvolvimento Integrado e como instalar o VS Code ou IntelliJ.

Uma IDE é um editor de texto turbinado para programadores. Ela ajuda a escrever código mais rápido, aponta erros e facilita a execução dos seus programas. Para iniciantes, recomendamos o Visual Studio Code (com a extensão de Java) ou o IntelliJ IDEA Community, que são gratuitos e muito poderosos.

Capitulo 03

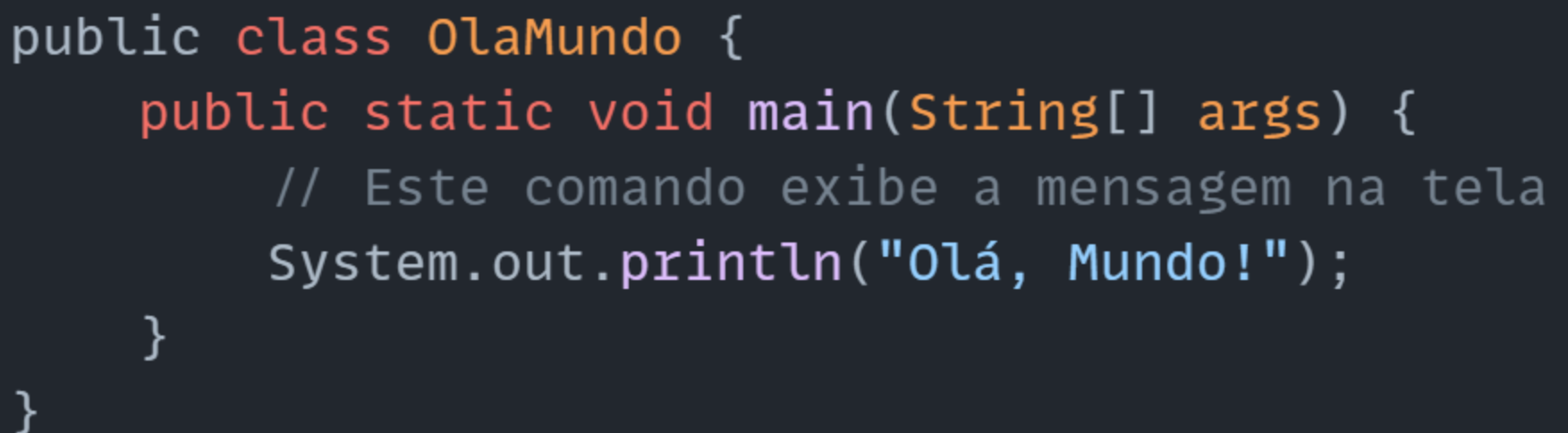
Fundamentos

Primeiro Código

Criando e executando o tradicional "Olá, Mundo!" em Java.

Vamos ao que interessa! Este é o seu primeiro programa em Java. Ele simplesmente imprime a mensagem "Olá, Mundo!" na tela.

System.out.println() é o comando que usamos para exibir algo.

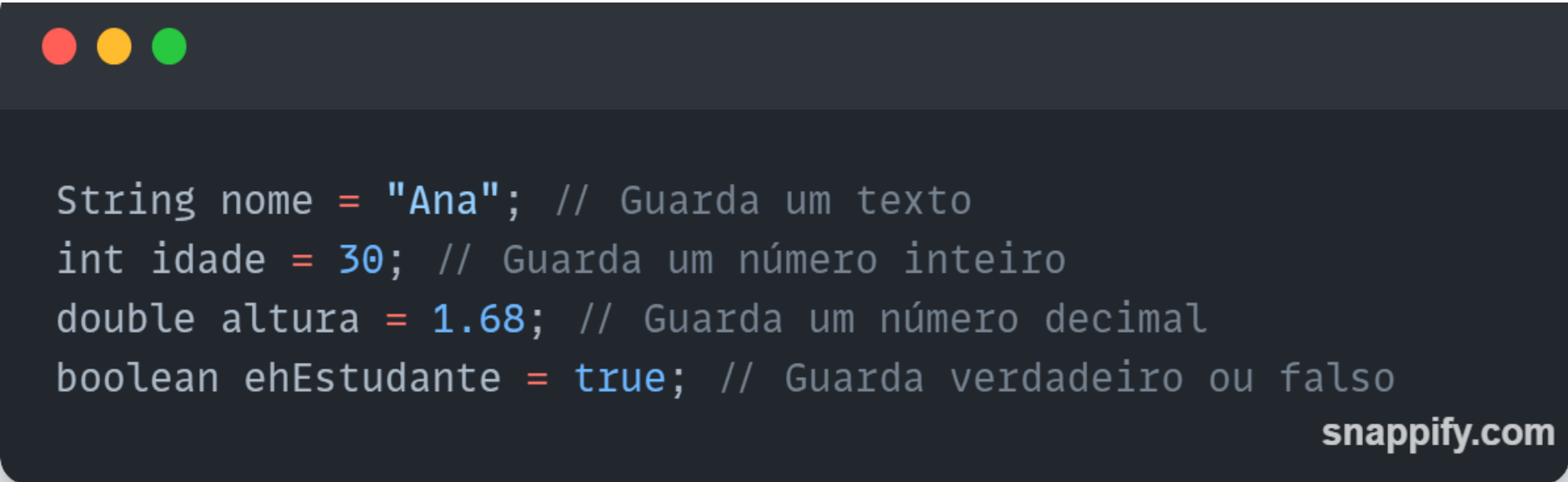


```
public class OlaMundo {  
    public static void main(String[] args) {  
        // Este comando exibe a mensagem na tela  
        System.out.println("Olá, Mundo!");  
    }  
}
```

Guardando Dados

Introdução a variáveis e os tipos de dados primitivos (texto, números, etc.).

Variáveis são como caixas onde guardamos informações. Cada caixa tem um tipo. As principais são: **int** para números inteiros, **double** para números com casas decimais, **boolean** para verdadeiro ou falso, e **String** para textos.



```
String nome = "Ana"; // Guarda um texto  
int idade = 30; // Guarda um número inteiro  
double altura = 1.68; // Guarda um número decimal  
boolean ehEstudante = true; // Guarda verdadeiro ou falso
```

snappify.com

Matemática Básica

Conhecendo os operadores aritméticos e lógicos essenciais.

Você pode fazer contas em Java usando operadores matemáticos simples, como + (soma), - (subtração), * (multiplicação) e / (divisão).

```
int a = 10;  
int b = 5;
```

```
int soma = a + b; // resultado será 15  
int multiplicacao = a * b; // resultado será 50
```

snappify.com

Entrada do Usuário

Como interagir com o usuário e ler informações digitadas no console.

Para tornar seu programa interativo, você pode ler o que o usuário digita. Para isso, usamos uma ferramenta chamada **Scanner**.

```
import java.util.Scanner; // Importa a ferramenta

public class Leitura {
    public static void main(String[] args) {
        Scanner leitor = new Scanner(System.in);

        System.out.println("Digite seu nome:");
        String nomeUsuario = leitor.nextLine(); // Lê o texto digitado

        System.out.println("Olá, " + nomeUsuario + "!");
        leitor.close(); // Fecha o leitor
    }
}
```

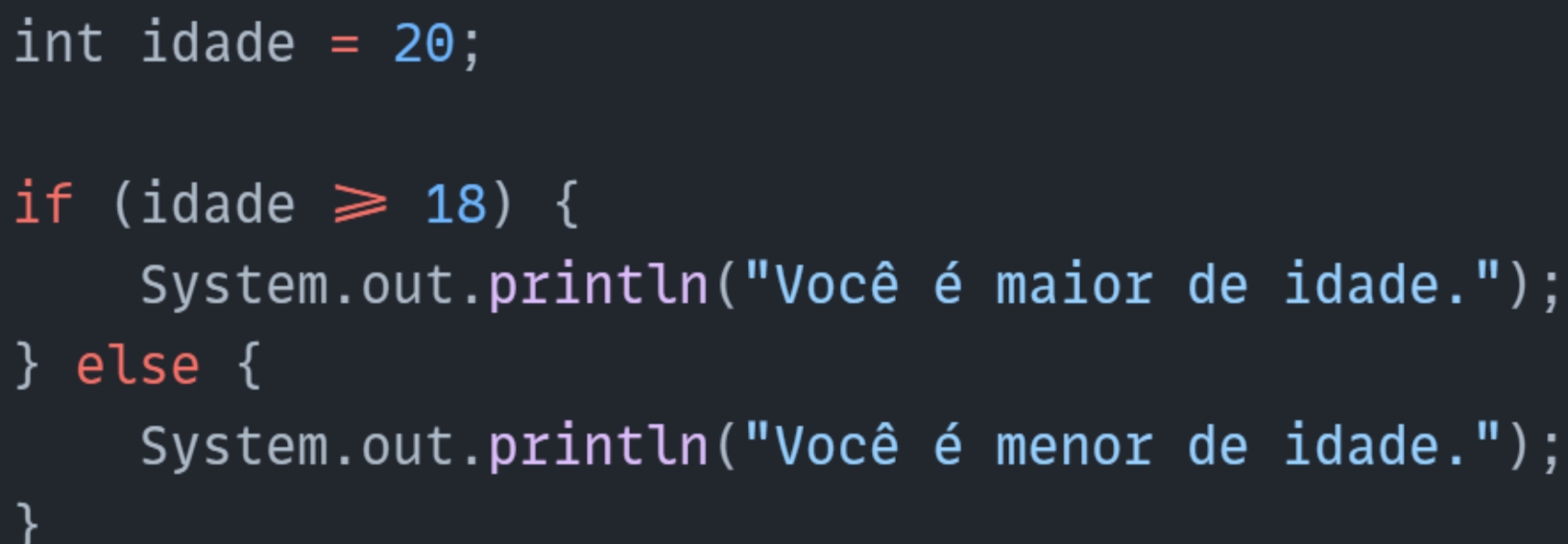
Capitulo 04

Controle

Tomando Decisões

Utilizando as estruturas condicionais **if**, **else** e **else if**.

Com o **if** (se), seu programa pode tomar decisões. Se uma condição for verdadeira, ele executa um bloco de código. Com o **else** (senão), você pode definir o que acontece se a condição for falsa. Com o **else if** você verifica se uma outra condição é verdadeira apenas se a condição anterior do **if** (ou de um **else if** anterior) for falsa.



```
int idade = 20;  
  
if (idade ≥ 18) {  
    System.out.println("Você é maior de idade.");  
} else {  
    System.out.println("Você é menor de idade.");  
}
```

Múltiplas Escolhas

Simplificando decisões complexas com a estrutura **switch**.

Quando você tem várias opções para uma mesma variável, o **switch** é mais organizado do que vários **ifs**. Ele testa o valor da variável e executa o código do caso correspondente.

```
int diaDaSemana = 2;

switch (diaDaSemana) {
    case 1:
        System.out.println("Domingo");
        break;
    case 2:
        System.out.println("Segunda-feira");
        break;
    default:
        System.out.println("Dia inválido");
}
```

Repetindo Tarefas

Criando laços de repetição com **for** para tarefas com número definido de execuções.

O laço **for** é perfeito para repetir uma ação um número específico de vezes. Por exemplo, para contar de 1 a 5.

```
// Este código vai contar de 1 até 5
for (int i = 1; i ≤ 5; i++) {
    System.out.println("Contando: " + i);
}
```

snappify.com

Repetições Inteligentes

Usando os laços **while** e **do-while** para repetir ações baseadas em condições.

O **while** (enquanto) repete um bloco de código enquanto uma condição for verdadeira. É útil quando você não sabe exatamente quantas vezes a repetição vai acontecer. Já o **do-while** (faça enquanto) é uma estrutura de repetição que executa um bloco de código pelo menos uma vez antes de verificar a condição.

```
int contador = 0;

while (contador < 3) {
    System.out.println("Repetindo...");
    contador++; // Aumenta o contador para evitar um loop infinito
}
```

Capitulo 05

Estruturas

Listas Simples

Armazenando coleções de dados com Arrays.

Um **Array** é uma estrutura que armazena vários valores do mesmo tipo em uma única variável, como uma lista de compras. Cada item tem uma posição, que começa em 0.

```
// Cria um array de 3 números inteiros
int[] numeros = {10, 20, 30};

// Acessa o primeiro item (posição 0), que seria o 10
System.out.println("O primeiro número é: " + numeros[0]);
```

Manipulando Textos

Introdução à classe String e seus métodos mais úteis.

Textos (Strings) em Java vêm com várias funções úteis. Você pode, por exemplo, deixar tudo em maiúsculas, verificar o tamanho ou ver se o texto contém uma palavra específica..

```
String frase = "Java é divertido!";

System.out.println(frase.toUpperCase()); // "JAVA É DIVERTIDO!"
System.out.println("Tamanho da frase: " + frase.length()); // 17
```

Listas Dinâmicas

Uma breve introdução ao **ArrayList** para coleções de tamanho flexível.

Diferente dos arrays, o **ArrayList** é uma lista que pode crescer e diminuir de tamanho. É muito mais flexível para quando você não sabe quantos itens precisará guardar.

```
import java.util.ArrayList;

ArrayList<String> nomes = new ArrayList<>();
nomes.add("Carlos"); // Adiciona um item
nomes.add("Beatriz");
System.out.println(nomes); // [Carlos, Beatriz]
```


Capitulo 06

Métodos

Blocos de Código

O que são métodos e como eles organizam seu programa.

Métodos (ou funções) são blocos de código que realizam uma tarefa específica. Eles ajudam a organizar o programa, evitando repetição de código. Você pode "chamar" o método sempre que precisar executar aquela tarefa.

```
public class Funcoes {  
    // Define um método chamado 'saudacao'  
    public static void saudacao() {  
        System.out.println("Seja bem-vindo!");  
    }  
  
    public static void main(String[] args) {  
        saudacao(); // Chama o método para ser executado  
    }  
}
```

Enviando Dados

Entendendo o uso de parâmetros para tornar seus métodos mais flexíveis.

Parâmetros são informações que você "envia" para um método usar. Isso torna o método reutilizável e dinâmico. Por exemplo, um método que pode saudar qualquer nome que você passar.

```
// Este método aceita um nome como parâmetro
public static void saudarPessoa(String nome) {
    System.out.println("Olá, " + nome + "!");
}

public static void main(String[] args) {
    saudarPessoa("João"); // Saída: Olá, João!
    saudarPessoa("Maria"); // Saída: Olá, Maria!
}
```

Recebendo Respostas

Como utilizar o return para que um método devolva um valor.

Um método também pode "devolver" um resultado após sua execução. Usamos a palavra-chave **return** para isso. Por exemplo, um método que soma dois números e retorna o total.

```
// Este método retorna a soma de dois números
public static int somar(int a, int b) {
    return a + b; // Devolve o resultado da soma
}

public static void main(String[] args) {
    int resultado = somar(5, 3); // Guarda o retorno na variável
    System.out.println("O resultado é: " + resultado); // Saída: O resultado é: 8
}
```

Capitulo 07

Classes

Novo Paradigma

Introdução à Programação Orientada a Objetos (POO).

A POO é uma forma de programar que organiza o código em torno de "objetos" do mundo real. Pense em um Carro: ele tem características (cor, marca) e ações (acelerar, frear). Em POO, tentamos modelar isso no código.

Moldes de Objetos

Criando sua primeira classe com atributos e métodos.

Uma classe é o "molde" ou a planta de um objeto. Nela, definimos os atributos (as características) e os métodos (as ações) que os objetos daquela classe terão.

```
// O molde para criar objetos do tipo Carro
public class Carro {
    // Atributos
    String cor;
    String modelo;

    // Método
    void ligar() {
        System.out.println("O carro " + modelo + " está ligando.");
    }
}
```

Criando Objetos

O que é instanciar e como dar vida às suas classes.

"Instanciar" é o ato de criar um objeto real a partir do molde (classe). Cada objeto criado é uma instância independente, com seus próprios valores para os atributos.

```
public static void main(String[] args) {  
    // Cria um objeto Carro a partir da classe  
    Carro meuCarro = new Carro();  
    meuCarro.cor = "Vermelho";  
    meuCarro.modelo = "Fusca";  
  
    // Cria outro objeto  
    Carro seuCarro = new Carro();  
    seuCarro.cor = "Azul";  
    seuCarro.modelo = "Onix";  
  
    meuCarro.ligar(); // Saída: O carro Fusca está ligando.  
}
```


Construtores

O método especial para inicializar seus objetos da maneira correta.

O construtor é um método especial que é chamado automaticamente quando você cria um novo objeto. Ele é usado para definir os valores iniciais dos atributos.

```
public class Produto {  
    String nome;  
    double preco;  
  
    // Construtor  
    public Produto(String nome, double preco) {  
        this.nome = nome;  
        this.preco = preco;  
    }  
}  
  
// Como usar  
Produto caneta = new Produto("Caneta BIC", 2.50);
```

Capitulo 08

Pilares POO

Encapsulamento

Protegendo e organizando seus dados com modificadores de acesso.

Encapsulamento é a ideia de "proteger" os atributos de uma classe, controlando o acesso a eles. Usamos a palavra `private` para tornar um atributo privado e criamos métodos públicos (getters e setters) para acessá-lo de forma controlada. Isso evita modificações acidentais.

Herança

Como criar novas classes a partir de outras já existentes para reutilizar código.

Herança permite que uma classe (filha) herde atributos e métodos de outra classe (mãe). Por exemplo, uma classe Gato e uma classe Cachorro podem herdar características de uma classe Animal (como ter um nome), evitando que você repita o mesmo código.

Polimorfismo

Entendendo como objetos podem se comportar de maneiras diferentes.

Polimorfismo significa "muitas formas". Em Java, isso permite que objetos de classes diferentes respondam à mesma chamada de método de maneiras específicas. Por exemplo, tanto um Gato quanto um Cachorro podem ter um método fazerSom(), mas cada um o executará de sua própria maneira (miando ou latindo).

Abstração

Focando no essencial e escondendo a complexidade.

Abstração é o conceito de esconder os detalhes complexos e mostrar apenas a funcionalidade essencial de um objeto. Quando você liga a TV, você só precisa apertar o botão "ligar", não precisa entender toda a eletrônica por trás. Em Java, criamos interfaces e classes abstratas para definir o que um objeto deve fazer, sem detalhar como ele faz.

Capitulo 09

Erros

Lidando com Imprevistos

O que são exceções e por que elas acontecem.

Uma exceção é um erro que acontece durante a execução do seu programa, como tentar dividir um número por zero ou acessar um arquivo que não existe. Se não forem tratadas, as exceções fazem seu programa parar de funcionar.

Capturando Erros

Utilizando os blocos **try-catch** para tratar exceções de forma elegante.

Para evitar que o programa quebre, podemos "capturar" as exceções. Colocamos o código que pode dar erro dentro de um bloco **try** (tentar). Se um erro acontecer, o código dentro do bloco **catch** (pegar) é executado, permitindo que você lide com o erro de forma controlada.

```
try {  
    // Código que pode causar um erro  
    int[] numeros = {1, 2, 3};  
    System.out.println(numeros[10]); // Tentando acessar posição que não existe  
} catch (Exception e) {  
    // Código a ser executado se o erro acontecer  
    System.out.println("Ocorreu um erro! Posição inválida no array.");  
}
```

Capitulo 10

Próximos Passos

Projeto Prático

Desafio: Construindo um pequeno sistema de cadastro para consolidar o aprendizado.

A melhor forma de aprender é praticando! Como desafio, tente criar um pequeno sistema de cadastro de clientes no console. Ele deve permitir adicionar um novo cliente (nome, idade), listar os clientes cadastrados e ter um menu de opções. Use tudo o que aprendeu: variáveis, laços, Scanner, ArrayList e classes.

Além do Básico

Sugestões de estudo: frameworks (Spring), desenvolvimento web e mobile com Java.

Java é um universo gigante.

Depois de dominar os fundamentos, você pode explorar áreas como:

Spring Framework: O framework mais popular para criar sistemas web e APIs robustas.

Desenvolvimento Android: Use Java ou Kotlin (uma linguagem "prima" do Java) para criar aplicativos para celulares.

Bancos de Dados: Aprenda a conectar suas aplicações Java a bancos de dados com JDBC.

Continue Aprendendo

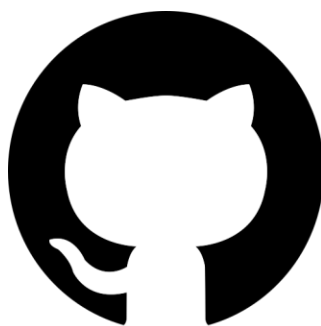
Dicas de comunidades, documentação oficial e recursos para sua jornada como dev Java.

A jornada de um programador é de aprendizado contínuo. Participe de fóruns como o Stack Overflow, leia a documentação oficial da Oracle, siga desenvolvedores Java em redes sociais e, o mais importante: nunca pare de programar e criar seus próprios projetos. Boa sorte!

OBRIGADO POR LER ATÉ AQUI

Esse ebook foi gerado por IA, e
diagramado por humano.

Esse conteúdo foi gerado com fins didáticos de
construção, não foi realizada uma validação
cuidadosa humana no conteúdo e pode conter
erros gerados por IA.



<https://github.com/AngelinTL>