

Programação de Computadores (resumo “Java 101”)

André Kishimoto
2024

Sobre este material...

Este material é um resumo do conjunto de slides Programação de Computadores (versão “Java 101”) do prof. André Kishimoto.

Parte do conteúdo e exemplos de código Java foram adaptados do material do prof. Dr. Ivan Carlos Alcântara de Oliveira.

1

Tipos de Dados e Variáveis

Tipos de dados e variáveis

- Case-sensitive
 - Letras maiúsculas são diferenciadas de letras minúsculas
- Identificadores em Java
 - Sequência de parte dos caracteres Unicode
 - Um nome pode ser composto por letras (minúsculas e/ou maiúsculas), dígitos e os símbolos _ (underscore) e \$ (dólar)
 - Embora o símbolo \$ não seja muito usado...
 - Um nome não pode ser iniciado por um dígito (0..9)
 - Palavra-chave da linguagem Java não pode ser um identificador
 - <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/keywords.html>
 - É recomendável usar a convenção de programação (estilo de escrita) do Java
 - Mas procure sempre adotar o padrão usado pela equipe/projeto!
 - <https://www.oracle.com/java/technologies/javase/codeconventions-contents.html>

Tipos de dados e variáveis

- Fortemente tipada
 - Precisamos definir o tipo de dado a ser armazenado
 - Exceção: var
 - Java 10+
 - Somente variáveis locais
 - A variável sempre deve ser declarada e iniciada com um valor
 - O compilador decide o tipo da variável baseado no valor atribuído à variável
 - Uma vez criada, o tipo da variável não pode ser alterado

Tipos de dados e variáveis

A linguagem Java não é totalmente Orientada a Objetos, pois possui atributos (variáveis) do tipo primitivo, ou seja, são tipos de dados que não representam classes, mas sim valores básicos.

Category	Types	Size (bits)	Minimum Value	Maximum Value	Precision	Example
Integer	<code>byte</code>	8	-128	127	From +127 to -128	<code>byte b = 65;</code>
	<code>char</code>	16	0	$2^{16}-1$	All Unicode characters ^[1]	<code>char c = 'A';</code> <code>char c = 65;</code>
	<code>short</code>	16	-2^{15}	$2^{15}-1$	From +32,767 to -32,768	<code>short s = 65;</code>
	<code>int</code>	32	-2^{31}	$2^{31}-1$	From +2,147,483,647 to -2,147,483,648	<code>int i = 65;</code>
	<code>long</code>	64	-2^{63}	$2^{63}-1$	From +9,223,372,036,854,775,807 to -9,223,372,036,854,775,808	<code>long l = 65L;</code>
Floating-point	<code>float</code>	32	2^{-149}	$(2 \cdot 2^{-23}) \cdot 2^{127}$	From 3.402,823,5 E+38 to 1.4 E-45	<code>float f = 65f;</code>
	<code>double</code>	64	2^{-1074}	$(2 \cdot 2^{-52}) \cdot 2^{1023}$	From 1.797,693,134,862,315,7 E+308 to 4.9 E-324	<code>double d = 65.55;</code>
Other	<code>boolean</code>	--	--	--	false, true	<code>boolean b = true;</code>
	<code>void</code>	--	--	--	--	--

https://en.wikibooks.org/wiki/Java_Programming/Primitive_Types

Tipos de dados e variáveis

Variáveis em Java podem ser do tipo:

- **Primitivas:** podem ser de um dos tipos de dados apresentados no slide anterior.
 - Armazenadas na memória stack.
 - Cópia: uma segunda variável apenas armazena o mesmo valor, mas o endereço de memória é diferente.
 - Alteração da segunda variável: não afeta a variável original.
- **Referências:** usadas para referenciar um objeto.
 - A referência é armazenada na memória stack.
 - O objeto original é armazenado na memória heap.
 - Referência: uma segunda variável (stack) aponta para o mesmo objeto da memória heap.
 - Alteração da segunda variável: altera a variável original.

Tipos de dados e variáveis

```
<tipo> <nomeDaVariavel>;  
<tipo> <nomeDaVariavel> = <valorInicial>;
```

// Exemplo

```
String disciplina = "Estrutura de Dados I";
```



The diagram consists of a curved arrow pointing from the text 'Estrutura de Dados I' in the code line above to the text 'Estrutura de Dados I' in the explanatory text below. A vertical arrow points from the text 'Estrutura de Dados I' in the explanatory text to the variable 'disciplina' in the code line above.

Associa/atribui o valor "Estrutura de Dados I" na variável disciplina, que é do tipo String.

***** OBSERVAÇÃO: String não é um tipo de dado primitivo em Java! *****

Tipos de dados e variáveis

```
<tipo> <nomeDaVariavel>;  
<tipo> <nomeDaVariavel> = <valorInicial>;
```

// Exemplos

```
String disciplina = "Estrutura de Dados I";  
int numero1;  
int numero2 = 99;  
bool jogando;  
float nota = 7.75f;  
char letra = 'Y';
```

Tipos de dados e variáveis

```
static final <tipo> <NOME_DA_CONSTANTE> = <valorFixo>;
```

// Exemplos

```
static final double PI = 3.14159265359;
```

```
static final String CODIGO_DISCIPLINA = "ENEX50328";
```

1b

Exemplo Java

Estrutura básica de um programa Java

```
[import ...]
```

```
[public] class Identificador {
```

```
    public static void main(String[] args) {  
        // Constantes, variáveis e comandos locais.  
    }
```

```
}
```

Estrutura básica de um programa Java

Quando o código faz uso de bibliotecas adicionais, usamos o comando `import <nome da biblioteca>`.

`[import ...]`

Todo programa Java precisa ter uma classe que contém a `main()`.

`[public] class Identificador {`

Indicamos o ponto de partida (inicial) do projeto.

```
public static void main(String[] args) {  
    // Constantes, variáveis e comandos locais.  
}
```


```
}
```

Exemplo 1: Hello, World!

```
class Main {  
  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
  
}
```

Exemplo 1: Hello, World!

```
class Main {  
  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```



Imprime a mensagem "Hello, World!" (sem aspas) no terminal do sistema, pulando uma linha após a string "Hello, World!".
System.out.print(<string>); não pula uma linha.

Exemplo 2: Leitura/escrita de variáveis

Exemplo:

Entrada, saída e tipos primitivos (ou não) de dados.

Programa que solicita ao usuário alguns dados: nome, idade, peso e altura, depois imprime uma frase contendo os valores lidos.

Exemplo 2: Leitura/escrita de variáveis

A entrada de dados via console pode ser feita com a classe `java.util.Scanner`

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Scanner.html>

Tipo de dado	Método Scanner	Funcionamento
boolean	nextBoolean()	Varre o próximo token da entrada em um valor booleano e retorna esse valor.
byte	nextByte()	Verifica o próximo token da entrada como um byte.
double	nextDouble()	Verifica o próximo token da entrada como um double.
float	nextFloat()	Verifica o próximo token da entrada como um float.
int	nextInt()	Verifica o próximo token da entrada como um int.
String	nextLine()	realiza a leitura de uma string com espaço em branco e finaliza com “return” ou enter.
long	nextLong()	Verifica o próximo token da entrada como um long.
short	nextShort()	Verifica o próximo token da entrada como um short.
String	next()	Localiza e retorna o próximo token completo deste scanner.

Exemplo 2: Leitura/escrita de variáveis

```
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.print("Digite seu nome: ");
        String nome = s.nextLine();
        System.out.print("Digite sua idade: ");
        int idade = s.nextInt();
        System.out.print("Digite seu peso: ");
        double peso = s.nextDouble();
        System.out.print("Digite sua altura: ");
        float altura = s.nextFloat();
        System.out.println("\n" + nome + " tem " + idade + " anos, pesa " + peso + " Kg, mede "
+ altura + " m.");
    }
}
```

Exemplo 3: Funções matemáticas

A classe `Math` já está no pacote (package) `java.lang`.

Esse pacote já é automaticamente importado, então, não precisamos importar, como foi feito com a classe `Scanner` do exemplo anterior.

Exemplo 3: Funções matemáticas

```
class Main {  
    public static void main(String[] args) {  
        int n1 = Math.abs(80);  
        System.out.println("Valor absoluto de 80 é " + n1);  
  
        int n2 = Math.abs(-60);  
        System.out.println("Valor absoluto de -60 é " + n2);  
  
        double n3 = Math.sqrt(36.0);  
        System.out.println("Raiz quadrada de 36.0 é " + n3);  
  
        double n4 = Math.cbrt(8.0);  
        System.out.println("Raiz cúbica de 8.0 é " + n4);  
  
        int n5 = Math.max(15, 80);  
        System.out.println("Valor máximo é " + n5);  
  
        int n6 = Math.min(15, 80);  
        System.out.println("Valor mínimo é " + n6);  
  
        // Continua no próximo slide...
```

Exemplo 3: Funções matemáticas

```
// ...continuação do slide anterior.
```

```
double n7 = Math.ceil(6.34);  
System.out.println("Teto de 6.34 é " + n7);
```

```
double n8 = Math.floor(6.34);  
System.out.println("Piso de 6.34 é " + n8);
```

```
double n9 = Math.round(22.445);  
System.out.println("Valor arredondado de 22.445 é " + n9);
```

```
double n10 = Math.round(22.545);  
System.out.println("Valor arredondado de 22.545 é " + n10);
```

```
double n11 = Math.pow(2.0, 3.0);  
System.out.println("Potência de 2 elevado a 3 é " + n11);
```

```
double n12 = Math.random();  
System.out.println("Um valor aleatório a partir da data e hora atual do sistema é " + n12);
```

```
System.out.println("Valor de pi é " + Math.PI);
```

```
}  
}
```

2

Operadores

Operadores

- Operadores

- Atribuição

- =

- Aritméticos

- + - * / % ++ --

- Aritméticos com atribuição

- += -= *= /= %=

- Relacionais

- == != < <= > >=

Operadores

- Operadores lógicos
 - AND → && (dois “e comercial” juntos)
 - OR → | | (duas barras verticais juntas)
 - NOT → ! (ponto de exclamação)

Operadores

Operadores	Comentários
()	agrupamento
++, --, +, -	Incremento, decremento e operadores unários
*, /, %	Multiplicação, divisão e resto
+, -	Soma e subtração
<, <=, >, >=	Operadores relacionais: menor, menor ou igual, maior, maior ou igual
==, !=	Operadores relacionais: Igualdade, desigualdade
&&	Operador lógico E
!!	Operador lógico OU
!	Operador Lógico Negação
(tipo) variavel	Conversão da valor da variável para o tipo especificado em ()
? :	Operador condicional ternário
=, +=, -=, *=, /=, %=	Operadores de atribuição

Estruturas de Decisão

Estruturas de Decisão

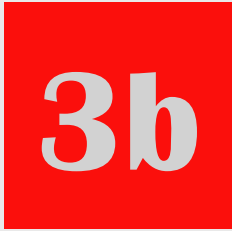
```
// Comando condicional simples:  
if (condição) {  
    // Bloco de comandos.  
}
```

```
// Comando condicional composto:  
if (condição) {  
    // Bloco de comandos A.  
} else {  
    // Bloco de comandos B.  
}
```

```
if (condição A) {  
    // Bloco de comandos A.  
} else if (condição B) {  
    // Bloco de comandos B.  
} else {  
    // Bloco de comandos C.  
}
```

Estruturas de Decisão

```
// Comando condicional de múltiplas escolhas:
switch (<identificador>) {
case <valor1>:
    // Instruções quando identificador == valor1.
    break;
case <valor2>:
    // Instruções quando identificador == valor2.
    break;
case <valorN>:
    // Instruções quando identificador == valorN.
    break;
default: // Opcional
    // Instruções quando identificador é diferente de todos os outros casos.
    break;
}
```



Exemplo Java

Exemplo 4: Função polinomial do 2º grau

Uma função polinomial do 2º grau é qualquer função $f: \mathcal{R} \rightarrow \mathcal{R}$ dada por $y = f(x) = ax^2 + bx + c$, sendo $a, b, c \in \mathcal{R}$ e $a \neq 0$.

As raízes da função polinomial do 2º grau são dadas pela fórmula de Bhaskara: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Observação:

O total de raízes reais depende do valor obtido para o discriminante $\Delta = b^2 - 4ac$, a saber:

- $\Delta > 0 \rightarrow$ há duas raízes reais e distintas;
- $\Delta = 0 \rightarrow$ há só uma raiz real;
- $\Delta < 0 \rightarrow$ não há raiz real.

O seu gráfico é uma curva chamada parábola, sendo que, se $a > 0$, a parábola tem concavidade para cima e um ponto de mínimo $P = (-b/2a, -\Delta/4a)$ e, se $a < 0$, a parábola tem concavidade para baixo e um ponto de máximo $P = (-b/2a, -\Delta/4a)$. Tendo por base isso, elabore um programa em Java que leia a, b, c , calcule o Δ , as raízes da equação e o ponto de mínimo ou máximo P .

Exemplo 4: Função polinomial do 2º grau

```
import java.util.Scanner;
class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        // Entrada dos coeficientes a, b, c para uma equação do 2o grau.
        System.out.println("Equação do 2o Grau: ax^2 + bx + c = 0");
        System.out.print("Digite o valor de a: ");
        double a = s.nextDouble();
        if (a == 0)
            System.out.println("A equação não é de 2o grau.");
        else {
            System.out.print("Digite o valor de b: ");
            double b = s.nextDouble();
            System.out.print("Digite o valor de c: ");
            double c = s.nextDouble();

            // Continua no próximo slide...
```

Exemplo 4: Função polinomial do 2º grau

```
// ...continuação do slide anterior.

// Calcula o delta e imprime.
double delta = Math.pow(b, 2) - 4 * a * c;
System.out.println("\nO valor do delta é: " + delta);

// Apresenta as raízes da equação.
if (delta > 0) {
    System.out.println("Duas raízes reais!");
    double x1 = (-b + Math.sqrt(delta)) / (2 * a);
    double x2 = (-b - Math.sqrt(delta)) / (2 * a);
    System.out.println("x1 = " + x1 + ", x2 = " + x2);
} else if (delta == 0) {
    System.out.println("Uma raiz real!");
    double x = -b / 2 * a;
    System.out.println("x = " + x);
} else
    System.out.println("Não há raízes reais!");

// Continua no próximo slide...
```


Exemplo 4: Função polinomial do 2º grau

```
// ...continuação do slide anterior.  
  
// Apresenta os pontos de máximo/mínimo.  
if (a > 0)  
    System.out.println("Ponto de mínimo:");  
else  
    System.out.println("Ponto de máximo:");  
double px = -b / 2 * a;  
double py = -delta / 4 * a;  
System.out.println("x = " + px + ", y = " + py + ".\n");  
}  
}  
}
```

Estruturas de Repetição

Java

- Possui três estruturas de repetição
 - while
 - do-while
 - for

Java

- Há também uma estrutura de repetição conhecida como for-each (*range-based loop*), usada para percorrer coleções de dados (ex. arrays).
- Algumas regras para o for-each:
 - Somente leitura (um elemento da coleção não pode ser modificado).
 - Uma única coleção de dados (não é possível percorrer duas coleções de dados de uma vez – por ex., comparar dois arrays).
 - Um único elemento por vez (acesso apenas ao elemento atual do loop).
 - Apenas para frente (a coleção de dados é percorrida do primeiro ao último elemento, um por vez).
- Um exemplo usando for-each pode ser visto na seção 5 (Vetores e Matrizes).

Java – while

Expressão que retorna true ou false, define se bloco de instruções da repetição deve ser executado. Condição é verificada **ANTES** de executar o bloco de instruções.




```
while (<condição>) {  
    <instruções quando resultado da condição é verdadeira>  
}
```

Java – do-while

do {

**<instruções quando resultado da condição é verdadeira,
a partir da segunda vez (o bloco de instruções sempre é
executado pelo menos uma vez)>**

} while (<condição>);



**Expressão que retorna true ou false, define se bloco de
instruções da repetição deve ser executado.
Condição é verificada DEPOIS de executar o bloco de
instruções pelo menos uma vez.**

Java – for

Valor inicial da
variável usada para
controlar a
repetição.

Condição para que
loop seja executado
ou encerrado.

Como a variável de
controle é
atualizada a cada
iteração do loop.



```
for (<valor inicial>; <condição>; <passo>) {  
    <instruções quando resultado da condição é verdadeira>  
}
```

Java – for-each

Tipo de dado da coleção de dados a ser percorrida.

Variável local do loop for (elemento atual da coleção de dados).

Variável que contém a coleção de dados.



```
for (<tipo> <variável> : <array/coleção de dados>) {  
    <instruções a serem executadas para o elemento atual da coleção de dados>  
}
```


4b

Exemplo Java

Exemplo 5: Repetição while

```
import java.util.Scanner;
class Main {
    static final int QTDE_NUMEROS = 5;
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        float soma = 0.0f, valor = 0.0f;
        int i = 0;
        while (i < QTDE_NUMEROS) {
            System.out.print("Digite o " + (i + 1) + "o valor: ");
            valor = s.nextFloat();
            soma = soma + valor;
            i++;
        }
        System.out.println("\nSoma = " + soma);
    }
}
```

Exemplo 6: Repetição do-while

```
import java.util.Scanner;
class Main {
    static final int QTDE_NUMEROS = 5;
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        float soma = 0.0f, valor = 0.0f;
        int i = 0;
        do {
            System.out.print ("Digite o " + (i + 1) + "o valor: ");
            valor = s.nextFloat();
            soma = soma + valor;
            i++;
        } while(i < QTDE_NUMEROS);
        System.out.println("\nSoma = " + soma);
    }
}
```

Exemplo 7: Repetição for

```
import java.util.Scanner;
class Main {
    static final int QTDE_NUMEROS = 5;
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        float soma = 0.0f, valor = 0.0f;
        int i = 0;
        for (i = 0; i < QTDE_NUMEROS; i++) {
            System.out.print("Digite o " + (i + 1) + "o valor: ");
            valor = s.nextFloat();
            soma = soma + valor;
        }
        System.out.println("\nSoma = " + soma);
    }
}
```

5

Vetores e Matrizes

Java

- Vetores em Java podem ser de tipos primitivos e objetos.
- Primeiro elemento começa no índice 0.
 - Último índice é $n - 1$.
 - n = tamanho do vetor (quantidade de elementos).
- Vetores são declarados usando [], que podem ser inseridos logo após o tipo de dado ou logo após o nome da variável.
- É possível criar um vetor em Java de algumas maneiras, conforme os próximos slides.

Java

```
// Declaramos um vetor de chars.  
char a[];  
// Depois alocamos memória para armazenar N chars.  
// Nesse caso, 5 chars, de a[0] até a[4].  
a = new char[5];  
  
// Declaramos outro vetor de chars.  
char[] b;  
// Depois alocamos memória para armazenar N chars.  
// Nesse caso, 5 chars, de b[0] até b[4].  
b = new char[5];  
  
// Atribuimos valores para os elementos do vetor de chars b.  
b[0] = '?';  
b[1] = '@';  
b[2] = 'A';  
b[3] = '9';
```

Java

```
// Declaramos um vetor de inteiros e alocamos memória para 10 elementos.  
// Vetor com capacidade 10, de vetor[0] até vetor[9].  
int[] vetor = new int[10];  
  
// Declaramos outro vetor de inteiros e alocamos memória para  
// 20 elementos. Vetor com capacidade 20.  
int outroVetor[] = new int[20];  
  
// Declaramos um vetor de inteiros e já atribuímos os valores do vetor.  
// Nesse caso, a memória é alocada conforme a quantidade de valores atribuídos.  
int[] c = { 10, 20, 30, 40, 50 };  
  
// Outro exemplo, com double.  
double[] x = { 1.2, 3.4, 5.6, 7.8 };
```


Java

Atenção!

- Em Java, vetores (arrays) são objetos.
 - Instâncias da classe `java.lang.Object`.
- Para obter o tamanho de um vetor, acessamos o atributo `length`:

```
int[] vetor = new int[10];  
System.out.println("Tamanho do vetor: " + vetor.length);
```

Java

▪ Exemplo de matrizes em Java

```
// Declaramos, alocamos memória e iniciamos uma matriz bidimensional de inteiros.  
int x[][] = {  
    { 1, 2, 3 },  
    { 4, 5, 6 }  
};
```

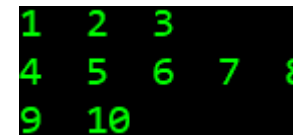
```
// Com a declaração anterior, os elementos da matriz x são:  
// c[0][0] = 1 c[0][1] = 2 c[0][2] = 3  
// c[1][0] = 4 c[1][1] = 5 c[1][2] = 6
```

Java

▪ Exemplo de matrizes em Java

// É possível declarar uma matriz com linhas de tamanhos diferentes,
// ainda que isto não seja muito usado:

```
int[][] y = {  
    { 1, 2, 3 },  
    { 4, 5, 6, 7, 8 },  
    { 9, 10 }  
};  
  
for (int row = 0; row < y.length; ++row) {  
    for (int col = 0; col < y[row].length; ++col) {  
        System.out.print(y[row][col] + " ");  
    }  
    System.out.println();  
}
```



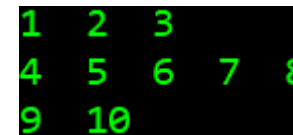
1	2	3		
4	5	6	7	8
9	10			

Java

▪ Exemplo de for-each em Java

// É possível declarar uma matriz com linhas de tamanhos diferentes,
// ainda que isto não seja muito usado:

```
int[][] y = {  
    { 1, 2, 3 },  
    { 4, 5, 6, 7, 8 },  
    { 9, 10 }  
};  
  
for (int row : y) {  
    for (int col : row) {  
        System.out.print(col + " ");  
    }  
    System.out.println();  
}
```



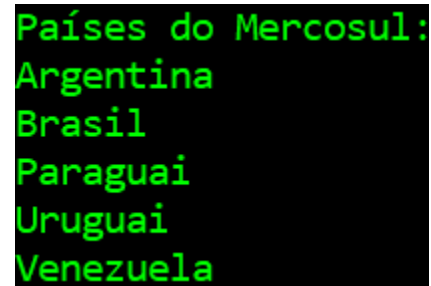
1	2	3		
4	5	6	7	8
9	10			

Java

▪ Exemplo de for-each e var em Java

```
System.out.println("Países do Mercosul:");
```

```
String mercosul[] = { "Argentina", "Brasil", "Paraguai", "Uruguai", "Venezuela" };  
for (var pais : mercosul) {  
    System.out.println(pais);  
}
```



```
Países do Mercosul:  
Argentina  
Brasil  
Paraguai  
Uruguai  
Venezuela
```

5b

Exemplo Java

Exemplo 8: Array

```
import java.util.Scanner;

class Exemplo8 {
    static final int QTDE_NUMEROS = 5;

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        float soma = 0.0f;
        int vetor[] = new int[QTDE_NUMEROS];

        // Leitura e soma dos números.
        for (int i = 0; i < QTDE_NUMEROS; ++i) {
            System.out.print("Digite o " + (i + 1) + "o valor: ");
            vetor[i] = s.nextInt();
            soma = soma + vetor[i];
        }

        // Continua no próximo slide...
```

Exemplo 8: Array

```
// ...continuação do slide anterior.
```

```
// Calcula a média.
```

```
float media = soma / QTDE_NUMEROS;
```

```
int qtdeMaiorMedia = 0;
```

```
// Encontra a quantidade maior do que a média.
```

```
for (int i = 0; i < QTDE_NUMEROS; ++i) {
```

```
    if (vetor[i] > media)
```

```
        ++qtdeMaiorMedia;
```

```
}
```

```
System.out.println("Média = " + media + "\nQtde. Maior Média = " +  
qtdeMaiorMedia);
```

```
}
```

```
}
```


Referências

Referências



Programação de Computadores
Desenvolvimento de Jogos Digitais com
GameMaker: Studio
André Kishimoto
ISBN 978-85-906129-2-6

Referências

Estrutura de Dados I – Conceitos Básicos da Linguagem de Programação Java

Notas de aula da disciplina Estrutura de Dados I, 2023.

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Estrutura de Dados I – Conceitos Básicos da Linguagem de Programação Java

Exemplos Java (código-fonte), 2023.

Prof. Dr. Ivan Carlos Alcântara de Oliveira



CC BY-SA 4.0 DEED

Atribuição-Compartilhagual 4.0 Internacional

Canonical URL : <https://creativecommons.org/licenses/by-sa/4.0/>

[See the legal code](#)


Você tem o direito de:


Compartilhar — copiar e redistribuir o material em qualquer suporte ou formato para qualquer fim, mesmo que comercial.

Adaptar — remixar, transformar, e criar a partir do material para qualquer fim, mesmo que comercial.

O licenciante não pode revogar estes direitos desde que você respeite os termos da licença.

De acordo com os termos seguintes:

 **Atribuição** — Você deve dar o [crédito apropriado](#), prover um link para a licença e [indicar se mudanças foram feitas](#). Você deve fazê-lo em qualquer circunstância razoável, mas de nenhuma maneira que sugira que o licenciante apoia você ou o seu uso.

 **Compartilhagual** — Se você remixar, transformar, ou criar a partir do material, tem de distribuir as suas contribuições sob a [mesma licença](#) que o original.

Sem restrições adicionais — Você não pode aplicar termos jurídicos ou [medidas de caráter tecnológico](#) que restrinjam legalmente outros de fazerem algo que a licença permita.



CC BY-SA 4.0 DEED

Attribution-ShareAlike 4.0 International

Canonical URL : <https://creativecommons.org/licenses/by-sa/4.0/>

[See the legal code](#)


You are free to:


Share — copy and redistribute the material in any medium or format for any purpose, even commercially.

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

 **Attribution** — You must give [appropriate credit](#), provide a link to the license, and [indicate if changes were made](#). You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

 **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the [same license](#) as the original.

No additional restrictions — You may not apply legal terms or [technological measures](#) that legally restrict others from doing anything the license permits.

Notices: