Cábula de Java · Bases da programação

Programa · Dados · Cálculos · Texto · Fluxo · Métodos · Arrays · Ajuda

Holger Sebastião · www.sebastiao.org
Versão 1.8 · 2026-10-25

Estrutura do programa

```
// Ficheiro: HelloWorld.java
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

Nome do ficheiro = nome da classe (`HelloWorld.java`). O programa começa em `main`.

Cada instrução termina com `;`. Maiúsculas e minúsculas contam: `soma` ≠ `Soma`.

Saída

```
System.out.println("Olá"); // com Enter
System.out.print("Olá");   // sem Enter
System.out.println("Preço: " + preco);
System.out.printf("%.2f €%n", preco);
// → 3,50 €  (2 casas decimais)
```

`%.2f` decimal com 2 casas, `%d` inteiro, `%s` texto, `%n` nova linha.

! `"Soma: " + 3 + 4` → `Soma: 34`
Usa parênteses: `"Soma: " + (3 + 4)` → `Soma: 7`

Comentários e carateres especiais

```
// de uma linha
/* de várias linhas */
/** Descrição de um método (Javadoc) */
```

`\n`	nova linha	`\"`	aspas
`\t`	tabulação	`\\`	barra invertida

Variáveis e constantes

```
int idade;          // declarar
idade = 17;         // atribuir um valor
int pontos = 0;     // as duas coisas numa linha
pontos = pontos + 10;
final double IVA = 0.23;  // constante
```

`=` é a **atribuição**: calcula o lado direito e guarda à esquerda.

Constante com `final`: depois, o valor já não pode mudar. No estruturograma aparece `const`. `Math.PI` já existe.

Entrada — Scanner

```
import java.util.Scanner;  // no topo
Scanner entrada = new Scanner(System.in);

String palavra = entrada.next();   // palavra
String linha = entrada.nextLine(); // linha
int idade = entrada.nextInt();
double preco = entrada.nextDouble();
```

! Depois de `nextInt()` ou `nextDouble()` sobra o Enter. Um `nextLine()` a seguir devolve `""`. Solução: um `entrada.nextLine();` extra.

! Num sistema em português, `nextDouble()` espera **vírgula**: `3,5`

`entrada.close();` só no fim do programa; depois, nenhum Scanner lê mais.

Tipos de dados

TIPO	BITS	CONTEÚDO	EXEMPLO
`byte`	8	−128 … 127	
`short`	16	−32 768 … 32 767	
`int`	**32**	**±2,1 mil milhões**	`int total = 5;`
`long`	64	±9,2 triliões	`long n = 5000000000L;`
`float`	32	≈ 7 dígitos	`float f = 2.5f;`
`double`	**64**	**≈ 15 dígitos**	`double p = 3.99;`
`char`	16	um caráter	`char c = 'A';`
`boolean`		**true / false**	`boolean ok = true;`
`String`		**texto (classe)**	`String s = "Ada";`

A negrito = padrão nas aulas. Decimais com **ponto**: `3.99`. `'A'` é um `char`, `"A"` uma `String`. String é uma classe, por isso começa por maiúscula.

Nomes

`Bilheteira`	classes: UpperCamelCase
`precoComIva`	variáveis, métodos: lowerCamelCase
`VALOR_MAX`	constantes: TUDO_MAIUSCULAS

Sem espaços, sem algarismo no início, sem palavras-chave como `int` ou `class`. Evitar acentos e ç.

Nomes que dizem algo: `numeroAlunos` em vez de `a`.

Cálculos

`+` `-` `*` `/` `%` (resto). Multiplicar antes de somar, parênteses primeiro.

```
7 / 2        // 3    int / int: perde o resto!
7.0 / 2      // 3.5
7 % 2        // 1    resto (módulo)
num % 2 == 0 // true se num for par
```

! Se os dois números forem `int`, o resultado também é `int`: `1 / 2` → `0`.

Formas curtas — não recomendado

```
pontos = pontos + 10;  // curto: pontos += 10;
preco = preco * 2;     // curto: preco *= 2;
i = i + 1;             // curto: i++;
i = i - 1;             // curto: i--;
```

Escreve a forma longa: daqui a duas semanas ainda a percebes. A curta só precisas de a saber ler, aparece muito em código alheio. Exceção: `i++` no for.

Math — sem import

```
Math.sqrt(16)     // 4.0  raiz quadrada
Math.pow(2, 3)    // 8.0  2 elevado a 3
Math.abs(-5)      // 5    valor absoluto
Math.max(3, 8)    // 8    (min igual)
Math.round(2.5)   // 3    arredondar
```

Números aleatórios

```
double z = Math.random(); // 0.0 até <1.0
// 0 a 9:
int digito = (int) (Math.random() * 10);
// 1 a 6:
int dado = (int) (Math.random() * 6) + 1;
```

Em geral:
`(int) (Math.random() * (max - min + 1)) + min`

Conversão de tipos — casting

```
double d = 5;              // 5.0 automático
int inteiro = (int) 3.7;   // 3 cortado!
int r = (int) Math.round(3.7); // 4 arredondado
double q = (double) 7 / 2; // 3.5
char c = (char) 65;        // 'A'
int codigo = (int) 'A';    // 65
```

Pequeno → grande automaticamente, grande → pequeno só com `(tipo)`.

Texto ↔ número

```
int n = Integer.parseInt("42");
double d = Double.parseDouble("3.14");
boolean b = Boolean.parseBoolean("true");
String s = String.valueOf(42); // ou 42 + ""
```

`parseDouble` espera um **ponto**. Se o texto não for um número (`"4x"`) → `NumberFormatException`.

Cadeias de carateres — `String texto = "Bolas";`

`texto.length()`	`5`
`texto.charAt(0)`	`'B'` (conta-se a partir de **0**)
`texto.substring(1, 3)`	`"ol"` (o fim não conta)
`texto.indexOf("l")`	`2` (não existe: `-1`)
`texto.contains("ola")`	`true`
`texto.equals("Bolas")`	`true`
`texto.equalsIgnoreCase("bolas")`	`true`
`texto.toUpperCase()`	`"BOLAS"` (`toLowerCase` igual)
`texto.replace("l", "x")`	`"Boxas"` (todos)
`texto + " de sabão"`	`"Bolas de sabão"`
`" Oi ".strip()`	`"Oi"` (só nas pontas)

`trim()` como `strip()`, mas não conhece todos os espaços. `stripLeading()` só à esquerda, `stripTrailing()` só à direita.

! As strings nunca mudam sozinhas: `texto.toUpperCase();` sozinho não faz nada. Certo: `texto = texto.toUpperCase();`

! Compara textos com `equals`, nunca com `==`.

```
for (int i = 0; i < texto.length(); i++) {
  System.out.println(texto.charAt(i));
}
```

Parênteses e import

`{ }`	bloco: classe, método, if, ciclo
`( )`	condição, parâmetros, cast, cálculo
`[ ]`	arrays

`import java.util.Scanner;` permite usar uma classe de outro pacote pelo nome curto. `String`, `Math` e `System` não precisam de import.

Cábula de Java · Bases da programação

Programa Dados Cálculos Texto Fluxo Métodos Arrays Ajuda

Holger Sebastião · www.sebastiao.org
Versão 1.8 · 2026-10-25

Condições

Uma condição dá sempre `true` ou `false`.

`==`	igual	`!=`	diferente
`<`	menor	`>`	maior
`<=`	menor ou igual	`>=`	maior ou igual
`&&`	E: as duas true	`||`	OU: pelo menos uma
`!`	NÃO: inverte, `!feito`		

```
boolean jovem = idade >= 14 && idade < 18;
```

! `=` atribui, `==` compara.

! `1 <= x <= 6` não funciona. Certo: `x >= 1 && x <= 6`

! Textos: `nome.equals("Ada")`, nunca `nome == "Ada"`

Decisão `if`

```
if (idade >= 18) {
  System.out.println("maior de idade");
} else {
  System.out.println("menor de idade");
}
```

VÁRIOS CASOS

```
if (pontos >= 90) {
  nivel = "muito bom";
} else if (pontos >= 50) {
  nivel = "aprovado";
} else {
  nivel = "reprovado";
}
```

Verifica-se de cima para baixo e corre o primeiro ramo que corresponder. O `else` pode faltar.

! Nada de `;` depois de `if (…)`, senão a condição não se aplica a nada.

Pseudocódigo

```
LER numero
SE numero > 0 ENTÃO
    ESCREVER "positivo"
SENÃO
    ESCREVER "não positivo"
```

Escolha múltipla `switch`

MODERNO, A PARTIR DO JAVA 14

```
switch (dia) {
  case 1 -> texto = "segunda";
  case 2 -> texto = "terça";
  case 6, 7 -> texto = "fim de semana";
  default -> texto = "outro dia";
}
```

Não é preciso `break`. Várias instruções: `case 1 -> { … }`

CLÁSSICO

```
switch (dia) {
  case 6: case 7:
    texto = "fim de semana";
    break;
  default:
    texto = "outro dia";
}
```

! Sem `break`, o programa continua pelo `case` seguinte.

Funciona com `int`, `char`, `String`, não com `double` nem `boolean`.

Ciclo, teste no início `while`

```
int i = 0;
while (i < 5) {
  System.out.println(i);
  i = i + 1;
}
```

Verifica **antes** de cada passagem. Se a condição for logo falsa, não corre nem uma vez.

! Sem `i = i + 1;` a condição nunca fica falsa: ciclo infinito.

Ciclo, teste no fim `do-while`

```
int num;
do {
  System.out.print("Número de 1 a 6: ");
  num = entrada.nextInt();
} while (num < 1 || num > 6);
```

Verifica **depois** de cada passagem, por isso corre pelo menos uma vez. Típico: repetir a entrada até estar certa. No fim leva `;`.

Ciclo de contagem `for`

```
// for (início; condição; passo)
for (int i = 0; i < 5; i++) {
  System.out.println(i);  // 0 1 2 3 4
}
for (int i = 10; i > 0; i--) { … } // a descer
```

O contador chama-se `i`; encaixados, fora `i`, dentro `j`.

`break;` sai logo do ciclo mais interior. `continue;` salta o resto desta passagem.

Arrays

```
int[] numeros = new int[5]; // 5 casas, todas 0
int[] notas = {2, 1, 3};    // com valores
notas[0] = 4;               // primeira casa
int total = notas.length;   // 3, sem ()
```

Índice de `0` a `length - 1`. `notas[3]` → `ArrayIndexOutOfBoundsException`.

```
for (int i = 0; i < notas.length; i++) {
  System.out.println(notas[i]);
}
System.out.println(Arrays.toString(notas));
// → [4, 1, 3]
// precisa de: import java.util.Arrays;
```

! `println(notas)` não mostra os valores, só algo como `[I@1b6d3586`.

Array 2D `tabela`

```
int[][] lugares = new int[3][4];
// 3 linhas, 4 colunas
lugares[1][2] = 1;   // [linha][coluna]
lugares.length       // 3 linhas
lugares[0].length    // 4 colunas

for (int i = 0; i < lugares.length; i++) {
  for (int j = 0; j < lugares[i].length; j++) {
    System.out.print(lugares[i][j] + " ");
  }
  System.out.println();
}
```

Métodos `subprogramas`

Cabeçalho: `public static` **`tipoRetorno nome(parâmetros)`**

PROCEDIMENTO, SEM VALOR DE RETORNO

```
public static void saudar(String nome) {
  System.out.println("Olá " + nome);
}
saudar("Ada");                  // chamada
```

FUNÇÃO, COM VALOR DE RETORNO

```
public static int dobro(int num) {
  return num * 2;
}
int resultado = dobro(5);      // 10
```

`return` devolve o valor e termina o método. As variáveis locais só existem nele.

Passagem de parâmetros

Números, char, boolean: o método recebe uma cópia; quem chama não nota alterações (passagem por valor).

Arrays: o método recebe uma referência às mesmas casas; as alterações veem-se fora (passagem por referência).

```
static void zerar(int[] a) { a[0] = 0; }
int[] z = {5, 6};
zerar(z);           // z[0] agora é 0
```

Perceber as mensagens de erro

AO COMPILAR

`';' expected`	falta `;`, vê a linha de cima
`cannot find symbol`	erro de escrita ou não declarado
`incompatible types`	tipo errado: `(int)` ou `parseInt`
`missing return statement`	falta `return num` caminho
`reached end of file`	falta uma `}`

AO EXECUTAR

`…IndexOutOfBounds…`	índice grande demais ou negativo
`InputMismatchException`	o Scanner recebeu o tipo errado
`NumberFormatException`	o texto não é um número
`/ by zero`	inteiro dividido por 0