

Aula 3 mais detalhada

Aula 03 – Variáveis e Operadores

Tipos Primitivos em Java

Tipo	Descrição	Exemplo
<code>int</code>	Números inteiros	<code>int idade = 20;</code>
<code>double</code>	Números com casas decimais	<code>double pi = 3.14;</code>
<code>float</code>	Similar ao <code>double</code> , menor precisão	<code>float f = 2.5f;</code>
<code>char</code>	Caracteres únicos	<code>char letra = 'A';</code>
<code>boolean</code>	Lógico (verdadeiro/falso)	<code>boolean ativo = true;</code>
<code>long</code>	Inteiros maiores	<code>long l = 1000000L;</code>
<code>byte</code>	Inteiro pequeno (8 bits)	<code>byte b = 127;</code>
<code>short</code>	Inteiro pequeno (16 bits)	<code>short s = 32000;</code>

+ Operadores em Java

Aritméticos:

- Soma: `+`
- Subtração:
- Multiplicação:
- Divisão: `/`
- Resto da divisão: `%`

Relacionais:

- Igualdade: `==`
- Diferença: `!=`
- Menor que: `<`

- Maior que: `>`
- Menor ou igual: `<=`
- Maior ou igual: `>=`


Lógicos:

- E: `&&`
- Ou: `||`
- Negação: `!`

Leitura com `Scanner`

```
import java.util.Scanner;

Scanner sc = new Scanner(System.in);
int idade = sc.nextInt();
String nome = sc.nextLine(); // Para ler texto com espaço
double altura = sc.nextDouble();
sc.close();
```

 Dica: Sempre feche o Scanner com `sc.close()`; quando não for mais usá-lo.

Trabalhando com `String`

- Strings são objetos, não tipos primitivos!
- **Criação:** `String nome = "Felipe";`
- **Comparação correta:**

```
if (nome.equals("Felipe")) {
    System.out.println("Olá, Felipe!");
}
```

Métodos úteis de **String** :

Método	Descrição	Exemplo
<code>nome.length()</code>	Retorna o número de caracteres	<code>nome.length()</code> → 6
<code>nome.toUpperCase()</code>	Converte para maiúsculas	<code>"felipe".toUpperCase()</code> → "FELIPE"
<code>nome.toLowerCase()</code>	Converte para minúsculas	<code>"FELIPE".toLowerCase()</code> → "felipe"
<code>nome.charAt(0)</code>	Retorna o caractere da posição indicada	<code>nome.charAt(0)</code> → 'F'
<code>nome.contains("li")</code>	Verifica se contém uma substring	<code>nome.contains("li")</code> → true
<code>nome.substring(i, j)</code>	Retorna a parte da string entre as posições <code>i</code> e <code>j</code> (exclusivo)	<code>"Felipe".substring(1, 4)</code> → "eli"
<code>nome.substring(i)</code>	Retorna da posição <code>i</code> até o final	<code>"Felipe".substring(3)</code> → "ipe"

⚠ Índices começam em 0! `substring(0, 3)` retorna os 3 primeiros caracteres.

Classe **Math** (biblioteca matemática)

Método	Descrição	Exemplo
<code>Math.sqrt(x)</code>	Raiz quadrada	<code>Math.sqrt(16)</code> → 4.0
<code>Math.pow(a, b)</code>	Potência (a^b)	<code>Math.pow(2, 3)</code> → 8.0
<code>Math.abs(x)</code>	Valor absoluto	<code>Math.abs(-5)</code> → 5
<code>Math.max(a, b)</code>	Maior entre dois valores	<code>Math.max(3, 7)</code> → 7
<code>Math.min(a, b)</code>	Menor entre dois valores	<code>Math.min(3, 7)</code> → 3
<code>Math.round(x)</code>	Arredonda para inteiro mais próximo	<code>Math.round(2.8)</code> → 3
<code>Math.floor(x)</code>	Arredonda para baixo (menor inteiro)	<code>Math.floor(2.8)</code> → 2
<code>Math.ceil(x)</code>	Arredonda para cima (maior inteiro)	<code>Math.ceil(2.1)</code> → 3
<code>Math.random()</code>	Retorna um valor entre 0 e 1	<code>Math.random()</code> → 0.0–1.0

Dicas e Boas Práticas

- Sempre **inicialize variáveis** antes de usá-las.
- Use `camelCase` para nomes de variáveis.
- Prefira nomes **claros e descritivos**.
- Cuidado ao alternar `nextLine()` e `nextInt()` com `Scanner` — pode haver quebra de linha pendente.
- Em divisões de inteiros (`int/int`), a parte decimal é descartada.

System.out – Impressão no Console

Principais métodos de saída:

Método	Descrição	Exemplo
<code>System.out.print()</code>	Imprime sem pular linha	<code>System.out.print("Olá");</code>
<code>System.out.println()</code>	Imprime com quebra de linha	<code>System.out.println("Olá");</code>
<code>System.out.printf()</code>	Imprime com formatação (placeholders)	<code>System.out.printf("Valor: %d", 10);</code>

Caracteres de escape (Escape Sequences)

Usados para inserir caracteres especiais dentro de uma `String`.

Sequência	Significado	Exemplo	Saída
<code>\n</code>	Nova linha	<code>System.out.print("Olá\nMundo");</code>	Olá Mundo
<code>\t</code>	Tabulação (tab)	<code>System.out.print("A\tB");</code>	A B
<code>\\</code>	Barra invertida	<code>System.out.print("Caminho: C:\\Docs");</code>	Caminho: C:\Docs
<code>\"</code>	Aspas duplas	<code>System.out.print("Ele disse: \"Oi\"");</code>	Ele disse: "Oi"
<code>\'</code>	Aspas simples	<code>System.out.print("Letra: \'A\'");</code>	Letra: 'A'
<code>\r</code>	Retorno de carro (carriage return)	Raramente usado	

Uso do `System.out.printf()`

Permite controlar o **formato** da saída. Usa **placeholders (especificadores)** para indicar onde inserir valores.

Especificadores comuns:

Especificador	Tipo de dado	Exemplo	Saída
<code>%d</code>	Inteiro (<code>int</code>)	<code>System.out.printf("%d", 10);</code>	10
<code>%f</code>	Ponto flutuante (<code>float</code> , <code>double</code>)	<code>System.out.printf("%.2f", 3.1415);</code>	3.14
<code>%s</code>	<code>String</code>	<code>System.out.printf("%s", "Oi");</code>	Oi
<code>%c</code>	Caractere (<code>char</code>)	<code>System.out.printf("%c", 'A');</code>	A
<code>%%</code>	Imprime o caractere <code>%</code>	<code>System.out.printf("100%%");</code>	100%

Formatação numérica com `%f` :

Formato	Descrição	Exemplo	Saída
<code>%.2f</code>	2 casas decimais	<code>System.out.printf("%.2f", 3.456);</code>	3.46
<code>%10.2f</code>	Largura 10, 2 casas decimais	<code>System.out.printf("%10.2f", 3.456);</code>	3.46
<code>%-10.2f</code>	Alinhado à esquerda	<code>System.out.printf("%-10.2f", 3.456);</code>	3.46



Você pode combinar vários especificadores:

```
java
CopiarEditar
String nome = "Ana";
int idade = 22;
System.out.printf("Nome: %s, Idade: %d\n", nome, idade);
```