

IOS – Instituto de
Oportunidade Social

Java 02 - Tipos de Dados e Operadores



- > Sintaxe Java
- > Fluxograma
- > Tipos de Dados e Operadores

IOS – Instituto de
Oportunidade Social

Sintaxe Java



Por que estudar Java?

O Java está entre as linguagens de programação mais populares e utilizadas no mundo atual. Hoje, em 2021, apenas o **Python** é mais utilizado do que o Java de acordo com o ranking de **Top Programming Languages 2021** do IEEE Spectrum. A popularidade do Java pode ser vista pela quantidade de **vagas disponíveis** para profissionais que possuem esse conhecimento.

Por que estudar Java nesse curso?

O ensino de Java na forma de programação estruturada pode ser um grande passo para uma **introdução** a essa linguagem e com estruturas mais simples de programação. No contexto de programação web, o Java está presente no **backend** de serviços de internet. E para o aluno que deseja continuar a sua formação como programador web, esse conteúdo é uma das formas.

Primeiro programa em Java:

```
public class OlaMundo {  
    // método main inicia a execução do aplicativo  
  
    Java  
  
    public static void main(String[] args)  
    {  
        System.out.println("Bem vindo ao IOS!");  
    } // fim do método main  
  
}
```

Classes em Java

O Java é uma linguagem baseada em **classes**, portanto todo programa Java consiste em pelo menos uma classe que o programador define. Assim, a primeira instrução é a declaração de uma classe:

```
public class OlaMundo
```

A palavra-chave **class** introduz uma declaração de classe e é imediatamente seguida pelo nome dela.

Corpo da classe

O corpo de uma classe, que pode também ser chamado de **bloco de comandos** da classe, é definido pela abertura e fechamento de chaves logo após a declaração da classe. Tudo que está dentro dessas chaves faz parte do código da classe.

```
public class OlaMundo {
```

```
// Corpo da classe
```

```
}
```

Abertura e fechamento de
chaves que delimitam o
corpo da classe.

Indentação



Dica: Faça do **recuo de parágrafo** para dar **legibilidade** a seu código e ficar bem claro a **hierarquia de níveis do programa**. É comum dois ou quatro espaços. Qualquer que seja o estilo que você escolher, utilize-o de modo consistente.

```
public class OlaMundo {  
    // método main inicia a execução do aplicativo Java  
    public static void main(String[] args) {  
        System.out.println("Bem vindo ao IOS!");  
    } // fim do método main  
}
```

Recuo no nível do método main

Recuo no nível da classe

Palavras reservadas

Palavras-chave do Java

<code>abstract</code>	<code>assert</code>	<code>boolean</code>	<code>break</code>	<code>byte</code>
<code>case</code>	<code>catch</code>	<code>char</code>	<code>class</code>	<code>continue</code>
<code>default</code>	<code>do</code>	<code>double</code>	<code>else</code>	<code>enum</code>
<code>extends</code>	<code>final</code>	<code>finally</code>	<code>float</code>	<code>for</code>
<code>if</code>	<code>implements</code>	<code>import</code>	<code>instanceof</code>	<code>int</code>
<code>interface</code>	<code>long</code>	<code>native</code>	<code>new</code>	<code>package</code>
<code>private</code>	<code>protected</code>	<code>public</code>	<code>return</code>	<code>short</code>
<code>static</code>	<code>strictfp</code>	<code>super</code>	<code>switch</code>	<code>synchronized</code>
<code>this</code>	<code>throw</code>	<code>throws</code>	<code>transient</code>	<code>try</code>
<code>void</code>	<code>volatile</code>	<code>while</code>		

Palavras-chave que não são atualmente utilizadas

<code>const</code>	<code>goto</code>
--------------------	-------------------

Comentários

Comentários são **anotações** inseridas no **código fonte** com o objetivo de descrever alguma **lógica**, instrução, **lembrar algo** importante quando o código foi desenvolvido, **criar secções** para organização ou **cabeçalho** do código. Comentários são **ignorados pelo compilador** na verificação de sintaxe do fonte.

- Comentário de **linha**, que é iniciado por **//**
- Comentário **bloco**, abre com **/*** e fecha com ***/**

Declarando um método

Um **método** em Java é equivalente a uma **função**, sub-rotina ou procedimento em outras linguagens.

A instrução

```
public static void main(String[] args)
```

é o ponto de partida de cada aplicativo Java. Os **parênteses** depois do **main** indicam que ele é um bloco de **construção** do programa chamado método.

Comando de saída

A saída de um programa é o resultado do seu processamento, pode ser **armazenando** em uma **variável**, um **banco de dados** ou **exibição na tela**. Em Java, **System.out.println()** é uma instrução para imprimir o argumento passado dentro dos parênteses. O método **println()** exibe o resultado na saída padrão (monitor) e realiza a quebra de linha.





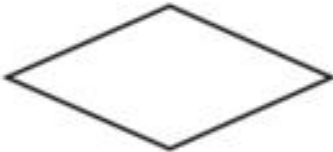
IOS – Instituto de
Oportunidade Social

Fluxograma







O Fluxograma é um tipo de **diagrama**, que representa esquematicamente um processo ou um **algoritmo**. Muitas vezes, ele é construído através de gráficos que ilustram de forma descomplicada a transição de informações entre os elementos que o compõem. O fluxograma utiliza alguns **símbolos padrões** que possuem um significado dentro do diagrama.

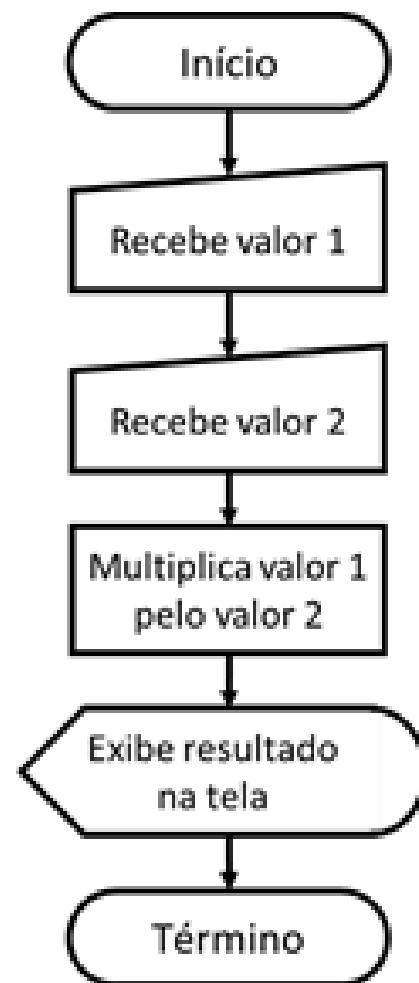
Fluxograma

Símbolo	Significado	Descrição
	Terminal	Representa o início ou o fim de um fluxo lógico. Em alguns casos definem as sub-rotinas.
	Entrada manual	Determina a entrada manual dos dados, geralmente através de um teclado.
	Processamento	Representa a execução de ações de processamento.
	Exibição	Mostra o resultado de uma ação, geralmente através da tela de um computador.
	Decisão	Representa os desvios condicionais nas operações de tomada de decisão e laços condicionais para repetição de alguns trechos do programa.

Fluxograma

Simbolo	Significado	Descrição
	Preparação	Representa a execução de um laço incondicional que permite a modificação de instruções do laço.
	Processo predefinido	Define um grupo de operações relacionadas a uma sub-rotina.
	Conector	Representa pontos de conexões entre trechos de programas, que podem ser apontados para outras partes do diagrama de bloco.
	Linha	Representa os vínculos existentes entre os símbolos de um diagrama de blocos.

A figura ao lado mostra o fluxograma para o programa de multiplicação de dois números. Inicialmente, temos a entrada dos dois valores, em seguida o processamento da multiplicação e no final a exibição na tela.



A figura abaixo mostra o pseudocódigo para o programa de multiplicação de dois números. Inicialmente, temos a entrada dos dois valores, em seguida o processamento da multiplicação e no final a exibição na tela.

ALGORITMO

DECLARE N1,N2,M NUMÉRICO

ESCREVA "Digite dois Digite dois
números:"

LEIA N1, N2

$M \leftarrow N1 * N2$

ESCREVA "Multiplicação = ", M

FIM Do ALGORITMO

IOS – Instituto de
Oportunidade Social

Tipos de Dados e Operadores



Constantes e tipos de dados

Constantes são **valores fixos** em um **algoritmo** e podem ser classificados como **literal** (caractere ou sequência de caracteres), **numérico** (inteiro ou real) ou **lógico** (verdadeiro ou falso). Já o **tipo de dado** indica a capacidade e o **tipo de conteúdo** de um valor ou constante. Veja a seguir os principais tipos de dados do Java, bem como a sua descrição:

Tipos de Variável

Primitivos		Específicos para Linguagem Java	
Tipos de dados	Definição	Tipos de dados	Capacidade
literal → também conhecido como texto ou caractere	Poderá receber letras, números e símbolos	char/String	16 bits (2 bytes)
inteiro	poderá receber números inteiros positivos ou negativos	int	32 bits (4 bytes) - 2.147.438.648 a 2.147.483.647

Primitivos		Específicos para Linguagem Java	
Tipos de dados	Definição	Tipos de dados	Capacidade
real → também conhecido como ponto flutuante	poderá receber números reais, isto é, com casas decimais, positivos ou negativos	float/double	32 bits (de -3,4E-38 até +3,4E+38) 64 bits (8 bytes) (de -1,7E-308 até +1,7E+308)
Lógico → também conhecido como booleano	poderá receber verdadeiro (1) ou falso (0)	boolean	8 bits (true ou false)

Exemplos tipos de dados:

false → lógico (boolean)	“O resultado é:” → literal (String)	“true” → literal (String)
21 → inteiro (int)	3.1415 → real (double)	‘h’ → literal (char)

Operadores

Os operadores especificam uma **avaliação/ação** a ser executada em um ou mais operandos e podem ser **aritméticos**, **relacionais** ou **lógicos**. É através de operandos que os programas em Java executam **cálculos aritméticos, lógicos e relacionais**.

Operadores aritméticos:

Operação	Operador	Expressão	Resultado
Adição	+	$6 + 4$	10
Subtração	-	$7 - 9$	-2
Multiplicação	*	$12 * 3$	36
Divisão	/	$44/2$	22
Módulo (Resto da Divisão)	%	$10 \% 3$	1

Ordem de precedência de operadores:

Desse modo a operação:

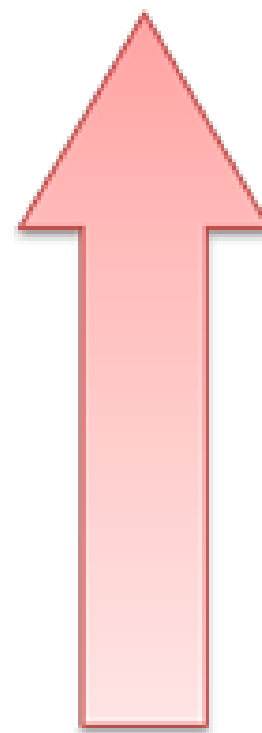
$$3 * 5 + 2$$

O programa primeiro faz a multiplicação e com o resultado dessa multiplicação realiza a soma. Os parênteses têm a maior precedência, portanto a operação:

$$3 * (5 + 2)$$

Primeiro é realizado a soma dentro dos parênteses e com o resultado dessa soma realiza a multiplicação.

Maior



()

/ *

+ -

Operadores relacionais:

Operador	Representação	Exemplo
Maior que	>	$a > b$: Se o valor de a for maior do que o valor de b, retornará true . Senão, retornará false .
Maior ou igual a	>=	$a >= b$: Se o valor de a for maior ou igual ao valor de b, retornará true . Senão, retornará false .
Menor que	<	$a < b$: Se o valor de a for menor que o valor de b, retornará true . Senão, retornará false .

Operador	Representação	Exemplo
Menor ou igual a	<=	$a <= b$: Se o valor de a for menor ou igual ao valor de, retornará true . Senão, retornará false .
igual a	==	$a == b$: se o valor de a for igual ao valor de b, retornará true . Senão, retornará false .
diferente de	!=	$a != b$: se o valor de a for diferente do valor de b, retornará true . Senão, retornará false .

Operadores lógicos:

- Operador **E (&&)**
 - somente resulta em verdadeiro (true), se **todas** as expressões condicionais forem true.
- Operador **Ou (||)**
 - se pelo menos **uma expressão** condicional for true, o resultado é verdadeiro (true).
- Operador **Não (!)**
 - se a expressão for false, o resultado é true.
 - se a expressão for true, o resultado é false.

Tabelas verdade:

A	B	A && B	A B
V	V	V	V
V	F	F	V
F	V	F	V
F	F	F	F

A	!A
V	F
F	V

IOS – Instituto de
Oportunidade Social

Vamos Praticar



Apostila de Java:

01.Apostila-Java

Páginas 38 a 43

OBS: Acompanhar o passo a passo com o instrutor

IOS – Instituto de
Oportunidade Social

Exercícios



- 1) Faça uma classe para **calcular a média de notas** (utilizar 4 notas diferentes para realizar o cálculo).
- 2) Faça uma classe para **calcular a área de um triângulo retângulo** ($\text{Base} \times \text{Altura} / 2$).