

# Apostila de Java

Mariana Cavalle

1 de janeiro de 2024

# Sumário

<b>Sumário</b>	<b>2</b>
<b>1 Introdução</b>	<b>3</b>
<b>2 Um pouco da história do Java</b>	<b>3</b>
2.1 Origem do nome Java	3
2.2 A tecnologia	3
2.3 Mas por quê o Java?	3
<b>3 Ambiente de desenvolvimento</b>	<b>4</b>
3.1 Preparando o ambiente	4
3.2 Qual IDE utilizar?	4
3.3 Primeiro programa em Java	4
<b>4 O básico</b>	<b>6</b>
4.1 Classes, métodos e atributos	6
4.2 Variáveis	7
4.3 Operadores aritméticos	8
4.4 Operadores relacionais	9
4.5 Operadores lógicos	9
4.6 Operadores de incremento e decremento	10
4.7 Comentários	10
<b>5 Estruturas condicionais</b>	<b>10</b>
<b>6 Estruturas de repetição</b>	<b>10</b>

## 1 Introdução

Essa apostila tem como intuito auxiliar iniciantes em Java que, assim como eu, já estiveram perdidos nesse mundo da programação sem saber por onde iniciar. Aqui, pretendo demonstrar conceitos teóricos e exemplificações práticas da linguagem. Como funciona, como exercitar e onde aprender mais.

Se alguém algum dia ler isso, espero que possa ajudar de alguma forma, enquanto também me ajuda a aprofundar meus conhecimentos. Aceito críticas, dúvidas e sugestões para melhorar isso aqui.

## 2 Um pouco da história do Java

### 2.1 Origem do nome Java

Antes de mais nada, é importante falar um pouquinho sobre como a linguagem surgiu. Tudo começou em 1991, na Sun Microsystems que criou a linguagem chamada "Oak" (que significa carvalho, em inglês). O nome foi dado em homenagem à uma árvore de carvalho que tinha perto do escritório e que visível por James Gosling, considerado o pai do Java.

No entanto, não foi possível registrar a linguagem com esse nome, pois já existia uma chamada Oak. Para decidir o novo nome, a equipe foi consultada e decidiram por Java. A ideia do nome é referência ao café de mesmo nome que é originário da ilha de Java. Por isso também a logo do nosso querido Java é uma xícara de café.

### 2.2 A tecnologia

Originalmente, a linguagem Oak foi desenvolvida para ser executada em diferentes aparelhos eletrônicos, permitindo que um só código rodasse em diferentes plataformas, como em televisões, geladeiras, entre outros. Mas essa ideia ainda era muito a frente do seu tempo e acabou não sendo prioridade da Sun Microsystems. Mas em 1994 com o avanço da internet, eles viram uma oportunidade de revisitar a linguagem e adaptá-la para o que fosse uma linguagem que rodasse em mais de uma plataforma e permitisse integração com a Web. Oficialmente, em 1995 a linguagem foi renomeada de Oak para Java e permitia a integração dos conteúdos Web de forma interativa.

### 2.3 Mas por quê o Java?

O Java foi a primeira linguagem a permitir que você rodasse o mesmo código em plataformas diferentes. Isso porque é uma linguagem tanto compilada quanto interpretada. Mas o que isso quer dizer?

Bom, quando eu escrevo um código em Java, eu o compilo (utilizando o `javac`). Esse processo de compilação vai gerar um "bytecode", que é um arquivo com a extensão `.class` na minha máquina para ser executada.

E é aí que entra a parte do "interpretada" na história, pois esse arquivo vai ser lido pela minha Java Virtual Machine (JVM). Ela é a responsável por interpretar todos os bytecodes que gerei anteriormente no meu arquivo `.class` e rodar os comandos em meu computador.

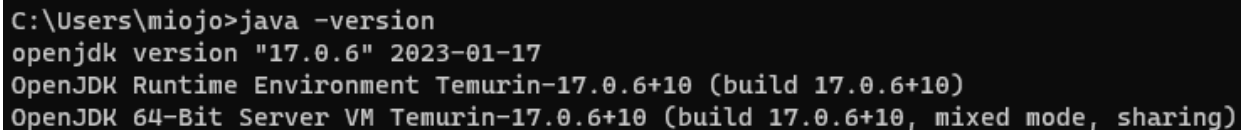
Isso foi algo revolucionário, principalmente porque agora eu poderia escrever um código só e rodar em qualquer Sistema Operacional. O mesmo código que eu escrevo no meu Windows pode ser executado no Linux ou em um MacOS, por exemplo, tendo em vista que os bytecodes gerados podem ser lidos por qualquer sistema operacional desde que ele tenha a JVM instalada. Mas óbvio que essa não é a única vantagem do Java. Além disso ele é orientado a objetos, muito robusto e permite a criação tanto de aplicativos web como aplicações

de desktop, programação de robôs, celulares, entre muitas outras coisas. Resumindo, Javinha é bom demais e serve pra tanta coisa. Vale a pena aprender.

### 3 Ambiente de desenvolvimento

#### 3.1 Preparando o ambiente

Para começar a programar em Java, primeiro precisamos instalar o Java Development Kit (JDK). O download pode ser feito pelo site da Oracle, por esse link: <https://www.oracle.com/br/java/technologies/downloads/>. Depois de realizar o processo de instalação, para confirmar que está tudo certo, basta abrir o prompt de comando e digitar "java -version". Se aparecer a versão instalada, está tudo pronto:

A screenshot of a Windows command prompt window. The text displayed is: C:\Users\miojo>java -version  
openjdk version "17.0.6" 2023-01-17  
OpenJDK Runtime Environment Temurin-17.0.6+10 (build 17.0.6+10)  
OpenJDK 64-Bit Server VM Temurin-17.0.6+10 (build 17.0.6+10, mixed mode, sharing)

```
C:\Users\miojo>java -version
openjdk version "17.0.6" 2023-01-17
OpenJDK Runtime Environment Temurin-17.0.6+10 (build 17.0.6+10)
OpenJDK 64-Bit Server VM Temurin-17.0.6+10 (build 17.0.6+10, mixed mode, sharing)
```

Figura 1 – Versão do Java

#### 3.2 Qual IDE utilizar?

Há diversas IDEs que podem ser utilizadas no desenvolvimento do Java, sendo algumas das mais famosas:

- NetBeans
- Eclipse
- IntelliJ
- VSCode

Ainda tem alguns que se arriscam a usar o Neovim, pra deixar tudo mais personalizado.

Aqui no meu caso, vou usar o IntelliJ para os exercícios e exemplos, porque foi a que mais me adaptei. Mas você pode testar e usar a que preferir.

#### 3.3 Primeiro programa em Java

Lógico que o primeiro programa tem que ser o clássico "Hello, World!". Dizem que se o seu primeiro programa em qualquer linguagem não for o famoso hello world, terá 7 anos de azar.

Dentro da IDE de sua escolha, crie um novo projeto e dentro da pasta "src" crie uma nova classe chamada "HelloWorld", conforme o exemplo abaixo:

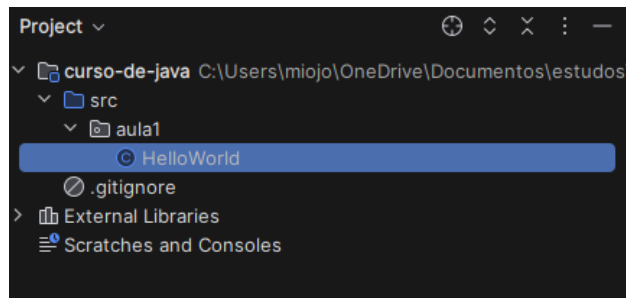


Figura 2 – Classe HelloWorld

As Classes em Java são onde você irá escrever o programa e seus nomes normalmente se iniciam em letra maiúscula, seguindo o padrão CamelCase. Portanto, sempre que eu tenho uma nova palavra, ela se inicia com letra maiúscula. Para esse programa inicial, iremos escrever esse código:

---

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

---

O método "public static void main(String[] args)" indica para o Java que aquele é o programa executável, é onde eu digo para o programa que aquelas são as linhas que devem ser seguidas quando iniciar o programa. Acredito que mais pra frente os métodos e classes ficarão mais claros.

Já para imprimir a informação na tela, utilizamos o "System.out.println();

Simples, né? Agora já posso rodar em minha IDE. No caso do IntelliJ, basta apertar "Shift + 10" ou clicar nessa setinha verde no topo, com a opção current file selecionada:

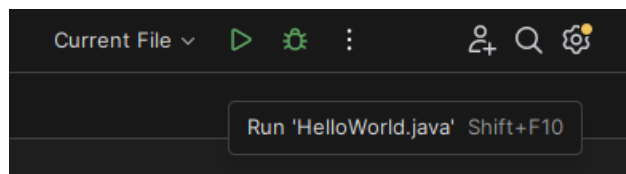


Figura 3 – Executar programa

Assim que compilar, a frase será exibida no terminal. Prontinho, temos nosso primeiro programa em Java!

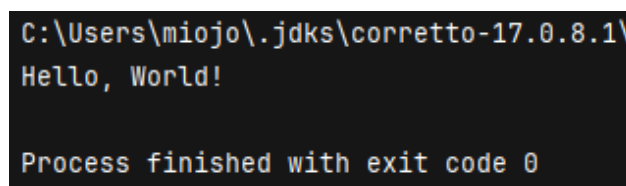


Figura 4 – Hello World!

## 4 O básico

Aqui vou repassar um pouco sobre os conceitos mais básicos do Java, como o que são classes, métodos, variáveis, tipos de dados, etc. Literalmente o básico e que é importante saber antes de começar a programar, de fato.

Claro, se o leitor já souber esses conceitos, sintam-se à vontade para pular para a próxima parte! Mas se quiser relembrar ou aprender, estamos aqui pra isso.

### 4.1 Classes, métodos e atributos

As classes são abstrações do mundo real, um tipo de dado que definimos pra representar objetos que têm uma característica em comum. Os programas feitos em Java são basicamente um monte de classes que se conversam. Toda classe é composta por atributos (características da classe) e métodos (comportamentos da classe).

Vendo assim é meio complicado de entender. Eu acho, pelo menos. Então vamos ao exemplo:

As classes servem para definir objetos do mundo real, então podemos ter uma classe chamada "Pessoa". Os atributos de uma pessoa, ou seja, as suas características podem ser: nome, idade, altura, nacionalidade, entre outros. São as informações que toda pessoa tem.

Já os métodos seriam as ações que a pessoa pode fazer: andar, ler, comer, falar... E por aí vai. Abaixo, segue um exemplo de como ficaria essa classe no Java:

---

```
public class Pessoa {  
    //Atributos  
    private String nome;  
    private int idade;  
    private double altura;  
    private String nacionalidade;  
  
    //Metodos  
    public String andar(){  
        System.out.println("Andando");  
    }  
  
    public String ler(){  
        System.out.println("Lendo");  
    }  
  
    public String comer(){  
        System.out.println("Comendo");  
    }  
  
    public String falar(){  
        System.out.println("Bla bla bla");  
    }  
}
```

---

Um outro exemplo bem comum para aprendizado é da classe "Carro". Quais seriam os atributos de um

carro? Marca, modelo, ano, cor, etc. Já os seus métodos podem ser: ligar o carro, desligar o carro, acelerar, abastecer, desacelerar, entre outros.

## 4.2 Variáveis

Variáveis são um pedacinho da memória do computador que separamos para armazenar algum tipo de informação temporária, enquanto o programa é executado.

As variáveis armazenam diferentes tipos de dados e as principais são:

- String: Armazena textos (ex: "Mariana");
- char: Armazena um caractere (ex: 'a', 'z', '2');
- boolean: Armazena se um valor é verdadeiro ou falso;
- int: Armazena números inteiros (ex: 1, 500, 35);
- double: Armazena números decimais (ex: 3,14);
- float: Armazena números decimais com menor precisão
- long: Armazena números inteiros mais longos
- short: Armazena números inteiros menores
- byte: Armazena números inteiros ainda menores

A diferença é basicamente o tamanho das variáveis e quantos bytes elas vão ocupar na memória. Por exemplo, se vou armazenar apenas um número, faz mais sentido usar "byte" do que "int" por uma questão de otimização, pois assim o sistema irá reservar um espaço menor na memória durante a execução. Os tipos de dados você pode consultar mais a fundo pela documentação aqui: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

Regras para nomeação de variáveis:

- Sempre inicia com uma letra minúscula ou com underline
- Não use caracteres especiais
- Não use palavras reservadas no nome da variável
- Siga o padrão camelCase, ou seja, coloque a primeira letra de cada palavra maiúscula (ex: minhaPrimeiraVariavel)
- Sempre opte por nomear as variáveis com nomes claros e objetivos (ex: nomeCompleto, idade, altura, etc)

Um exemplo de como ficam as variáveis em Java:

---

```
public class Variaveis{
    public static void main(String[] args){
        //Declaracao das variaveis, com tipo nome e valor
        int idade = 19;
        String nome = "Joaozinho";
        double altura = 1.86;
```

```
//Alterando o valor da variavel idade
idade = 20;
}
}
```

---

Acima, declarei a idade com o valor de 19 e depois alterei para 20. Isso serve apenas para mostrar que ao longo do programa uma mesma variável pode ter diferentes valores.

Mas se você quiser, também é possível declarar como uma **constante**, de modo a impedir que o valor mude durante o programa. Para isso, temos que colocar a palavra "final" antes de informar o tipo. Além disso, o nome deve ser colocado todo em letras maiúsculas para indicar que é um constante, conforme o exemplo abaixo:

```
public class Constante{
    public static void main(String[] args){
        final int EXEMPLO_CONSTANTE = 15;
    }
}
```

---

Depois disso, não posso alterar o valor que coloquei inicialmente.

### 4.3 Operadores aritméticos

Esses são aqueles operadores que usamos pra fazer as operações de matemática. São eles:

Operador	Função	Exemplo
+	Somar	a + b
-	Subtrair	a - b
*	Multiplicar	a * b
/	Dividir	a / b
%	Resto da divisão	a % b

Tabela 1 – Operadores aritméticos

Exemplo prático: Crie um programa que leia duas variáveis inteiras e realize a todas as operações matemáticas acima e exiba os resultados.

```
public class OperadoresAritmeticos{
    public static void main(String[] args){
        int a = 10;
        int b = 3;
        int resultado; //Variavel que vai receber os valores resultantes das operacoes

        resultado = a + b;
        System.out.println(resultado);

        resultado = a - b;
        System.out.println(resultado);
    }
}
```



```

        resultado = a * b;
        System.out.println(resultado);

        resultado = a / b;
        System.out.println(resultado);

        resultado = a % b;
        System.out.println(resultado);

    }
}

```

---

Ao rodar o programa, você deve receber todos os resultados das operações em ordem:

```

C:\Users\miojo\.jdk\corretto-17.0.8
13
7
30
3
1

Process finished with exit code 0

```

Figura 5 – Resultado operadores matemáticos

Veja que o valor da variável `resultado` foi alterado diversas vezes durante a execução do programa. E como é sequencial, sempre que seu valor é alterado foi chamado um `System.out.println` para exibir seu novo valor.

Exercício de fixação: Crie um programa que calcule a área de um triângulo e exiba o resultado na tela.

#### 4.4 Operadores relacionais

Esses operadores sempre irão nos retornar se a expressão é verdadeira ou falsa. São eles:

Operador	Função	Exemplo
>	Maior que	a > b
>=	Maior ou igual que	a >= b
<	Menor	a < b
<=	Menor ou igual que	a <= b
==	É igual a	a == b
!=	Diferente de	a != b

Tabela 2 – Operadores relacionais

#### 4.5 Operadores lógicos

Esses operadores avaliam se uma expressão retornam um valor verdadeiro ou falso. São eles:

- `&&`: Operador e ->. Vai avaliar todas as condições de são verdadeiras. Exemplo: `a && b` são maior que 5? Para dar verdadeiro, tanto o valor de `a` quanto de `b` devem ser maior que 5.
- `||`: Operador ou ->. Vai avaliar se pelo menos uma das condições é verdadeira. Exemplo `a || b` é maior que 5? Pelo menos um dos dois valores deve ser maior que 5 para retornar verdadeiro
- `!`: Operador de negação ->. Avalia se a negação da condição é verdadeira (Estranho né?). Exemplo: `a != b`. Aqui estou dizendo que **não** quero `a` sendo igual a `b`. Portanto, se os valores forem diferentes, será retornado "verdadeiro".

#### 4.6 Operadores de incremento e decremento

Esse tipo de operador aumenta ou diminui em uma unidade o valor da variável. Usaremos isso mais quando chegarmos em estruturas de repetição, principalmente no `for`. Eles são representados basicamente pelo `++` e `--`, dependendo se quero incrementar ou decrementar, mas é importante nos atentarmos na sequência que é feito, como abaixo:

- `i++`: A unidade será adicionada depois da expressão
- `++i`: A unidade será adicionada antes da expressão
- `i--`: A unidade será removida depois da expressão
- `--i`: A unidade será removida antes da expressão

#### 4.7 Comentários

Para fazer comentários em java você pode fazer da seguinte forma:

- Comentário em linha única: `// Texto comentado`
- Comentário em múltiplas linhas:
 

```
/*
 * Texto
 * Comentado
 */
```

### 5 Estruturas condicionais

### 6 Estruturas de repetição