



JAVA BÁSICO



Desvendando os Fundamentos da Linguagem

Thierry Marinho

Tópicos

- 1. Introdução ao Java**
- 2. Configurando Seu Ambiente de Desenvolvimento**
- 3. Seu Primeiro Programa Java: "Olá, Mundo!"**
- 4. Sintaxe Básica e Comentários**
- 5. Variáveis e Tipos de Dados**
- 6. Operadores**
- 7. Estruturas de Controle de Fluxo**



Introdução ao Java

Java é uma das linguagens de programação mais populares e poderosas do mundo. Criada pela Sun Microsystems (hoje parte da Oracle) em meados dos anos 90, ela é conhecida pela sua capacidade de rodar em diferentes plataformas, ou seja, um código Java escrito em um computador pode ser executado em outro, independentemente do sistema operacional. Isso é possível graças à Java Virtual Machine (JVM).

Por que aprender Java?

- Versatilidade: Usada em desenvolvimento web, mobile (Android), sistemas corporativos, Big Data e muito mais.
- Plataforma Independente: "Write once, run anywhere" (Escreva uma vez, execute em qualquer lugar).
- Comunidade Grande: Milhões de desenvolvedores ao redor do mundo, o que significa muitos recursos e ajuda disponíveis.
- Orientação a Objetos: Um paradigma de programação que ajuda a organizar e estruturar seu código de forma eficiente.



Configurando Seu Ambiente de Desenvolvimento

Antes de começar a programar em Java, você precisa instalar algumas ferramentas essenciais.

Java Development Kit (JDK)

O JDK é a espinha dorsal do desenvolvimento Java. Ele inclui:

- JRE (Java Runtime Environment): Para executar aplicações Java.
- Compilador (javac): Transforma seu código Java (que você escreve) em bytecode (que a JVM entende).
- Ferramentas: Outras ferramentas para depuração e documentação.

Você pode baixar a versão mais recente do JDK no site oficial da Oracle.

Ambiente de Desenvolvimento Integrado (IDE)

Uma IDE é um software que oferece um ambiente completo para programar, com recursos como auto-completar, depuração de código e realce de sintaxe. As mais populares para Java são:

- IntelliJ IDEA (Comunidade Edition é gratuita e excelente)
- Eclipse
- NetBeans



Seu Primeiro Programa Java: "Olá, Mundo!"

Tradicionalmente, o primeiro programa que todo desenvolvedor escreve é o "Olá, Mundo!". Ele serve para verificar se seu ambiente está configurado corretamente.

Java

```
public class OlaMundo {  
    public static void main(String[] args) {  
        System.out.println("Olá, Mundo!"); // Isso imprime a mensagem no console  
    }  
}
```

O que significa?

- `public class OlaMundo`: Declara uma classe chamada OlaMundo. Em Java, todo código reside dentro de classes.
- `public static void main(String[] args)`: É o método principal (ponto de entrada) da sua aplicação. Quando você executa um programa Java, a JVM procura e executa esse método.
- `System.out.println("Olá, Mundo!");`: Esta linha imprime a string "Olá, Mundo!" no console. `System.out` representa a saída padrão do sistema, e `println` imprime a mensagem e pula para a próxima linha.



Sintaxe Básica e Comentários

A sintaxe Java é a gramática da linguagem. Seguir as regras de sintaxe é crucial para que seu código seja compilado e executado corretamente.

Elementos Chave:

- Ponto e Vírgula (;): A maioria das instruções em Java deve terminar com um ponto e vírgula.
- Blocos de Código ({ }): Chaves são usadas para definir blocos de código, como os corpos de classes, métodos, laços e condicionais.
- Case Sensitive: Java diferencia maiúsculas de minúsculas (variavel é diferente de Variavel).

Comentários

Comentários são anotações no código que o compilador ignora. Eles servem para explicar o que seu código faz, tornando-o mais fácil de entender para você e para outros desenvolvedores.

Comentário de Linha Única: Começa com //. Java

```
// Isso é um comentário de linha única  
int idade = 30; // Declara uma variável inteira
```

Comentário de Bloco (Multi-linha): Começa com /* e termina com */. Java

```
/*  
 * Este é um comentário de bloco.  
 * Ele pode ocupar várias linhas.  
 */
```

Comentário Javadoc: Começa com /** e termina com */. Usado para gerar documentação. Java

```
/**  
 * Este método calcula a soma de dois números.  
 * @param a O primeiro número.  
 * @param b O segundo número.  
 * @return A soma de a e b.  
 */  
public int somar(int a, int b) {  
    return a + b;  
}
```



Variáveis e Tipos de Dados

Variáveis são "contêineres" para armazenar dados em um programa. Cada variável tem um nome, um tipo de dado e um valor. O tipo de dado define que tipo de informação a variável pode guardar.

Declarando Variáveis

Para declarar uma variável, você especifica o tipo de dado, seguido pelo nome da variável.

```
tipo nomeDaVariavel = valorInicial; // O valor inicial é opcional
```

Tipos de Dados Primitivos (os mais comuns):

- int: Para números inteiros (ex: 10, -500, 0).
- double: Para números de ponto flutuante (decimais) (ex: 3.14, -0.5).
- boolean: Para valores verdadeiros ou falsos (ex: true, false).
- char: Para um único caractere (ex: 'A', 'z', '7').

Exemplo:

```
int idade = 25;
double preco = 19.99;
boolean isAtivo = true;
char letraInicial = 'J';
String nome = "Maria"; // String não é um tipo primitivo, mas é muito usado
```

Nota: String é uma classe em Java, não um tipo primitivo, mas é tão fundamental que é ensinada logo no início. Ela representa sequências de caracteres (texto).

6. Operadores

Operadores são símbolos que realizam operações em um ou mais valores (operandos).

Operadores Aritméticos:

- + (adição)
- - (subtração)
- * (multiplicação)
- / (divisão)
- % (módulo - resto da divisão)

```
int a = 10;
int b = 3;
int soma = a + b;      // 13
int divisao = a / b;   // 3 (divisão de inteiros)
int resto = a % b;     // 1
```



Operadores Relacionais (de Comparação):

Usados para comparar valores; o resultado é sempre um boolean (true ou false).

- == (igual a)
- != (diferente de)
- > (maior que)
- < (menor que)
- >= (maior ou igual a)
- <= (menor ou igual a)

```
int x = 5;
int y = 10;
boolean igual = (x == y);    // false
boolean maior = (y > x);    // true
```

Operadores Lógicos:

Usados para combinar expressões booleanas.

- && (AND lógico - true se ambos forem true)
- || (OR lógico - true se pelo menos um for true)
- !(NOT lógico - inverte o valor booleano)

```
boolean condicao1 = true;
boolean condicao2 = false;
boolean resultadoAnd = condicao1 && condicao2; // false
boolean resultadoOr = condicao1 || condicao2;   // true
boolean resultadoNot = !condicao1;               // false
```

Operadores de Atribuição:

Usados para atribuir valores a variáveis.

- = (atribuição simples)
- +=, -=, *=, /=, %= (atribuição com operação)

```
int numero = 5;
numero += 3; // Equivalente a: numero = numero + 3; (numero agora é 8)
```



Estruturas de Controle de Fluxo

As estruturas de controle de fluxo determinam a ordem em que as instruções do seu programa são executadas.

Condicionais (if, else if, else):

Executam blocos de código diferentes com base em condições.

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

// Exemplo com else if
String tempo = "sol";
if (tempo.equals("chuva")) {
    System.out.println("Leve um guarda-chuva.");
} else if (tempo.equals("sol")) {
    System.out.println("Use protetor solar.");
} else {
    System.out.println("Verifique a previsão do tempo.");
}
```

switch:

Uma alternativa para múltiplas condições if-else if-else quando você está comparando uma única variável com vários valores.

```
int diaSemana = 3; // 1 para Domingo, 2 para Segunda, etc.
String nomeDia;

switch (diaSemana) {
    case 1:
        nomeDia = "Domingo";
        break; // Sai do switch
    case 2:
        nomeDia = "Segunda-feira";
        break;
    case 3:
        nomeDia = "Terça-feira";
        break;
    default: // Executado se nenhum case for correspondente
        nomeDia = "Dia inválido";
        break;
}
System.out.println("Hoje é " + nomeDia);
```



Laços (Loops):

Repetem um bloco de código várias vezes.

for: Usado quando você sabe o número de repetições.

```
for (int i = 0; i < 5; i++) {  
    System.out.println("Contagem: " + i); // Imprime 0, 1, 2, 3, 4  
}
```

while: Repete um bloco de código enquanto uma condição for verdadeira.

Java

```
int contador = 0;  
while (contador < 3) {  
    System.out.println("Loop While: " + contador);  
    contador++;  
}
```

do-while: Semelhante ao while, mas garante que o bloco de código seja executado pelo menos uma vez.

Java

```
int num = 0;  
do {  
    System.out.println("Loop Do-While: " + num);  
    num++;  
} while (num < 1); // Executa uma vez (num=0), depois a condição é falsa
```

