Estruturas Básicas de Programação em Java

Variáveis

Variáveis são espaços na memória do computador que armazenam valores. Em Java, cada variável tem um tipo que define qual tipo de dado ela pode armazenar.

Exemplo de declaração e uso de variáveis em Java:

```java
public class VariaveisExemplo {
    public static void main(String[] args) {
        // Declaração de variáveis
        int idade = 25;                 // variável do tipo inteiro
        double altura = 1.75;           // variável do tipo ponto
flutuante (número decimal)
        char inicial = 'J';         // variável do tipo caractere
        boolean estudante = true;  // variável do tipo booleano (true ou
false)
        String nome = "João";          // variável do tipo String (texto)

        // Exibindo valores
        System.out.println("Nome: " + nome);
        System.out.println("Idade: " + idade);
        System.out.println("Altura: " + altura);
        System.out.println("Inicial: " + inicial);
        System.out.println("Estudante? " + estudante);
    }
}
```

Tipos de Dados

s principais tipos de dados em Java são:

1. **int**: números inteiros (ex: 10, -5)
2. **double**: números decimais (ex: 3.14, -0.001)
3. **char**: caracteres (ex: 'a', 'Z')
4. **boolean**: valores lógicos (true ou false)
5. **String**: cadeia de caracteres (ex: "Olá mundo")

Aula 20 de agosto Professores Thiago/Paulo

Operadores

Operadores são símbolos que realizam operações sobre variáveis e valores.

Operadores Aritméticos

Operador	Descrição	Exemplo
+	**Adição**	**5 + 3 = 8**
-	**Subtração**	**5 - 3 = 2**
*	**Multiplicação**	**5 * 3 = 15**
/	**Divisão**	**10 / 2 = 5**
%	**Módulo (resto)**	**10 % 3 = 1**

Exemplo em Java:

```java
int a = 10;
int b = 3;

int soma = a + b;          // 13
int diferenca = a - b;     // 7
int produto = a * b;       // 30
int divisao = a / b;       // 3 (divisão inteira)
int resto = a % b;         // 1 (resto da divisão)
```

Operadores Relacionais

Comparam valores e retornam true ou false.

Operador	Descrição
==	Igual a
!=	Diferente de
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

```java
public class ternario {

    public static void main(String[] args) {

        int idade = 18;
        String resultado = (idade >= 18) ? "Maior de idade" : "Menor de
idade";
        System.out.println(resultado);
    }
}
```

Exemplo:

```java
int x = 5;
int y = 10;

boolean resultado = x < y;   // true
```

Digite o Código

```java
public class BasicoJava {
    public static void main(String[] args) {
        int idade = 30;
        double peso = 70.5;
        char genero = 'M';
        boolean temCarteira = true;
        String nome = "Carlos";

        System.out.println("Nome: " + nome);
        System.out.println("Idade: " + idade);
        System.out.println("Peso: " + peso);
        System.out.println("Gênero: " + genero);
        System.out.println("Tem carteira? " + temCarteira);
```

```
        // Operações
        int anosParaAposentar = 65 - idade;
        System.out.println("Anos para aposentar: " + anosParaAposentar);

        boolean podeDirigir = temCarteira && idade >= 18;
        System.out.println("Pode dirigir? " + podeDirigir);
    }
}
```

Conceitos de Programação Orientada a Objetos:

Definição

Programação Orientada a Objetos (POO) é um paradigma de programação que organiza o software em torno de **objetos**, que são instâncias de **classes**. Esses objetos combinam dados (atributos) e comportamentos (métodos) em uma única unidade.

Ao invés de pensar o problema só como uma sequência de comandos, a POO modela o sistema como um conjunto de entidades que interagem entre si, representando objetos do mundo real ou conceitual.

Objetivos da POO

1. **Modularidade**: Facilitar a divisão do programa em partes independentes (classes e objetos), melhorando a organização e manutenção do código.
2. **Reutilização**: Criar componentes reutilizáveis, por meio de herança e composição.
3. **Flexibilidade**: Permitir a adaptação e extensão do código sem grandes impactos (polimorfismo e encapsulamento).
4. **Encapsulamento**: Proteger os dados, expondo somente o que é necessário e escondendo os detalhes internos.
5. **Abstração**: Representar apenas as características essenciais dos objetos, ignorando detalhes irrelevantes para o contexto.

Conceitos Fundamentais

1. **Classe**: Molde ou projeto que define atributos e métodos comuns a todos os objetos daquele tipo.
2. **Objeto**: Instância concreta de uma classe, com valores específicos.
3. **Atributos**: Características (dados) do objeto.
4. **Métodos**: Comportamentos (funções) do objeto.
5. **Herança**: Permite que uma classe herde atributos e métodos de outra.
6. **Polimorfismo**: Objetos diferentes podem responder ao mesmo método de formas distintas.
7. **Encapsulamento**: Proteção dos dados do objeto, controlando o acesso através de métodos.

Aula Scanner

O comando **`Scanner`** em Java é uma classe da biblioteca padrão (presente no pacote `java.util`) usada para **ler dados de entrada**, como **números, strings, e outros tipos de dados**. Ela é muito usada quando queremos ler dados do teclado, de arquivos, ou de strings.

Importando a Classe Scanner

```
import java.util.Scanner;
```

Usos Comuns

A forma mais comum de usar `Scanner` é para **ler entrada do teclado** (`System.in`):

```
 Scanner scanner = new Scanner(System.in);
```

Leitura de texto

```
 String nome = scanner.next(); // Lê uma palavra (até espaço ou Enter)
 String linha = scanner.nextLine(); // Lê uma linha inteira
```

Leitura de números

```
        int idade = scanner.nextInt();     // Inteiro
        double altura = scanner.nextDouble(); // Número com ponto
        float peso = scanner.nextFloat();  // Número com ponto
        long numeroGrande = scanner.nextLong(); // Número inteiro longo
```

Leitura de booleanos

```
boolean ativo = scanner.nextBoolean(); // true ou false
```

Mistura de `nextLine()` com outros métodos

```
 int idade = scanner.nextInt();
 String nome = scanner.nextLine(); // Problema: pode pular esta linha!
```

Explicação: O `nextInt()` não consome o ENTER deixado no buffer. Então o `nextLine()` pega uma linha vazia.
Solução: Adicione um `scanner.nextLine()` logo após a leitura de números para limpar o buffer.

```
 int idade = scanner.nextInt();
 scanner.nextLine(); // Limpa o ENTER
String nome = scanner.nextLine(); // Agora lê
corretamente
```

Fechando o Scanner

Sempre que terminar de usar, **feche o `Scanner`** para liberar os recursos:

```
 scanner.close();
```

Método	Descrição
`next()`	Lê uma palavra
`nextLine()`	Lê uma linha completa
`nextInt()`	Lê um número inteiro
`nextDouble()`	Lê um número com ponto
`nextBoolean()`	Lê um valor booleano (`true/false`)
`close()`	Fecha o scanner

Scanner numérico

Scanner Caractere

Aula 20 de agosto Professores Thiago/Paulo

```
projeto > src > Aula > ConsoleCaractere.java > ConsoleCaractere > main(String[])
1   package Aula;
2
3   import java.util.Scanner;
4
5   public class ConsoleCaractere {
        Run | Debug
6       public static void main(String[] args) {
7
8
9           Scanner dados = new Scanner(System.in);
10
11          String nomeCliente;
12
13          System.out.println(x:"Digite o Nome do Cliente: ");
14          nomeCliente = dados.nextLine();
15
16
17          System.out.println("Seu nome é: " + nomeCliente);
18
19          dados.close();
20  }
21  }
```

Digite o código e passe para o SCANNER

```
Ex_01.java 1 | ConsoleCaractere.java | ConsoleNumber.java | Ternario.java
projeto > src > Ex_01.java > Ex_01
1   public class Ex_01 {
        Run | Debug
2       public static void main(String[] args) {
3        // Faça um programa que calcule a área do quadrado, triângulo, retângulo
4
5
6           double base = 10;
7           double altura = 15;
8           double lado = 2;
9
10
11          double   quadrado = lado * lado;
12          double retangulo = base*altura;
13          double triangulo =(base*altura)/2;
14
15          System.out.println("A área do triângulo é: " + triangulo + "\n" +
16                             "A área do retângulo é: " + retangulo + "\n" +
17                                "A área do quadrado é: " + quadrado
18
19                             );
20
21
22
```