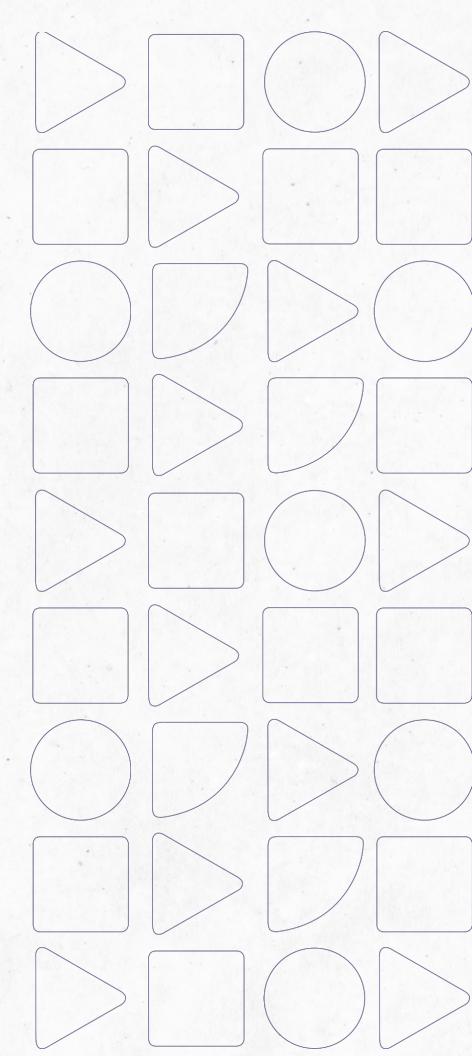


# Introdução à lógica de programação

**Disciplina:** Desenvolvimento de Aplicações



## Conteúdos:

Introdução à lógica de programação.

## Habilidade(s):

# Bloco 1

---

## Duas verdades e uma mentira



- Formem duplas;
- Em cerca de cinco minutos, escrevam duas informações verdadeiras e uma falsa sobre vocês mesmos;
- Um de vocês deve começar lendo as informações que escreveu, enquanto o seu colega de dupla deve adivinhar quais são as verdadeiras e qual é a falsa;
- Depois, invertam os papéis. Agora, o aluno que adivinhou deve ler as informações que escreveu sobre si.

# Variáveis e constantes na programação

Variáveis e constantes são elementos fundamentais na programação, permitindo armazenar e manipular dados. A nomenclatura e as convenções são importantes para a clareza do código. Dominar a declaração, a inicialização e a nomenclatura de variáveis e constantes é crucial para escrever um código claro e legível.



# Variáveis e constantes

Variáveis e constantes são elementos essenciais na programação. Elas nos permitem armazenar e manipular dados de diferentes tipos.

## Variáveis

Espaço na memória para armazenar dados que podem mudar durante a execução do programa.

## Constantes

Valores que permanecem inalterados durante a execução do programa.

## Tente responder

Sabendo o que é uma variável, você consegue fazer alguma alusão para explicar esse conceito?

??



# Variáveis

Tipo	O que aceita?	Exemplo
Int	Números inteiros - positivo ou negativo	1, 2, -4, -12038
Double	Número decimal com duas casas - positivo ou negativo	2.3, 70.6, -405.6
Float	Número decimal com uma casa - positivo ou negativo	23.45, 1.80, -3,14
String	Representa um conjunto de caractere - tipo texto	Arroz doce
Char	Representa um caractere - tipo texto	A, arroz, pera

# Identificadores de variáveis

Identificadores de variáveis são os nomes que damos às nossas variáveis e constantes. Eles devem seguir algumas regras e convenções para garantir a legibilidade do código.

Devem começar com letra ou '\_', não pode ser palavra-chave nem conter caracteres especiais.

Não podem conter espaços e devem ser sensíveis a letras maiúsculas e minúsculas.

Deve-se escolher nomes que descrevam o propósito da variável ou constante.

Devem iniciar com letra minúscula e maiúsculas para cada nova palavra.

Devem seguir padrões de nomenclatura para melhor leitura e entendimento do código.

Exemplos: **idade**, **nomeCompleto**, **\_saldoConta..**

# No código

## Declarando variáveis



Para declarar uma variável em Java, usamos a seguinte sintaxe: **tipo nomeDaVariavel;**

## Inicializando variáveis



Após a declaração, podemos inicializar uma variável atribuindo um valor a ela: **tipo nomeDaVariavel = valor;**

## Declarando e inicializando constantes



As constantes são variáveis cujo valor não pode ser alterado após a inicialização. Em Java, podemos declará-las usando a palavra-chave final: **final tipo NOME\_DA\_CONSTANTE = valor;**

# No código

- **Declaração:** indicar o tipo de dado da variável. Exemplo: "int idade;"
- **Inicialização:** atribuir um valor à variável. Exemplo: "idade = 25;".
- **Declaração e Inicialização em uma linha:** "int idade = 25;".



## No código

- **Declaração de Constantes:** usar a palavra-chave **final** antes da declaração. Exemplo: "final int ANO\_ATUAL = 2023;".



# No código

Exemplo de como calcular a área de um retângulo:

java



Copy code

```
public class CalculoAreaRetangulo {  
    public static void main(String[] args) {  
        int base = 10;  
        int altura = 5;  
        int area = base * altura;  
        System.out.println("A área do retângulo é: " + area);  
    }  
}
```



**Dividam-se em grupos e criem um  
pequeno programa que declare e  
utilize variáveis para armazenar  
informações como nome, idade e nota.**

# Bloco 2

---

## Saída de dados e comentários

A classe *system* oferece métodos como **println()** e **print()** para exibir informações no console. Os comentários podem ser usados para melhorar a compreensão do código. Também é importante saber que dominar a saída de dados e a adição de comentários ajuda a criar programas mais claros e interativos.



# System.out.print e system.out.println

A classe *system* oferece acesso a recursos do sistema, incluindo a saída padrão. Temos dois tipos:  
o **system.out.print** e **system.out.println**.

## System.out.print()

O método **System.out.print()** é usado para imprimir informações no console sem pular para uma nova linha após a impressão.

## System.out.println()

O método **System.out.println()** é usado para imprimir informações no console e pular para uma nova linha após a impressão.

# Comentários

Comentários são textos explicativos nos códigos que não afetam o seu funcionamento. São tipos de comentários:

- **// Comentário de linha**
- **/\* Comentário de bloco \*/**

```
java

// Este é um comentário de linha
System.out.println("Exemplo de comentários");

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



Dividam-se em grupos e criem um pequeno programa que utilize tanto o `System.out.println()` quanto o `System.out.print()` para imprimir diferentes tipos de saídas. Adicionem comentários ao código para explicar o que cada parte faz.

# Bloco 3

---

## Fofoca do bem

Que tal contar para a turma, em forma de fofoca, um pouco sobre as variáveis e como declará-las, além das regras de nomenclatura?

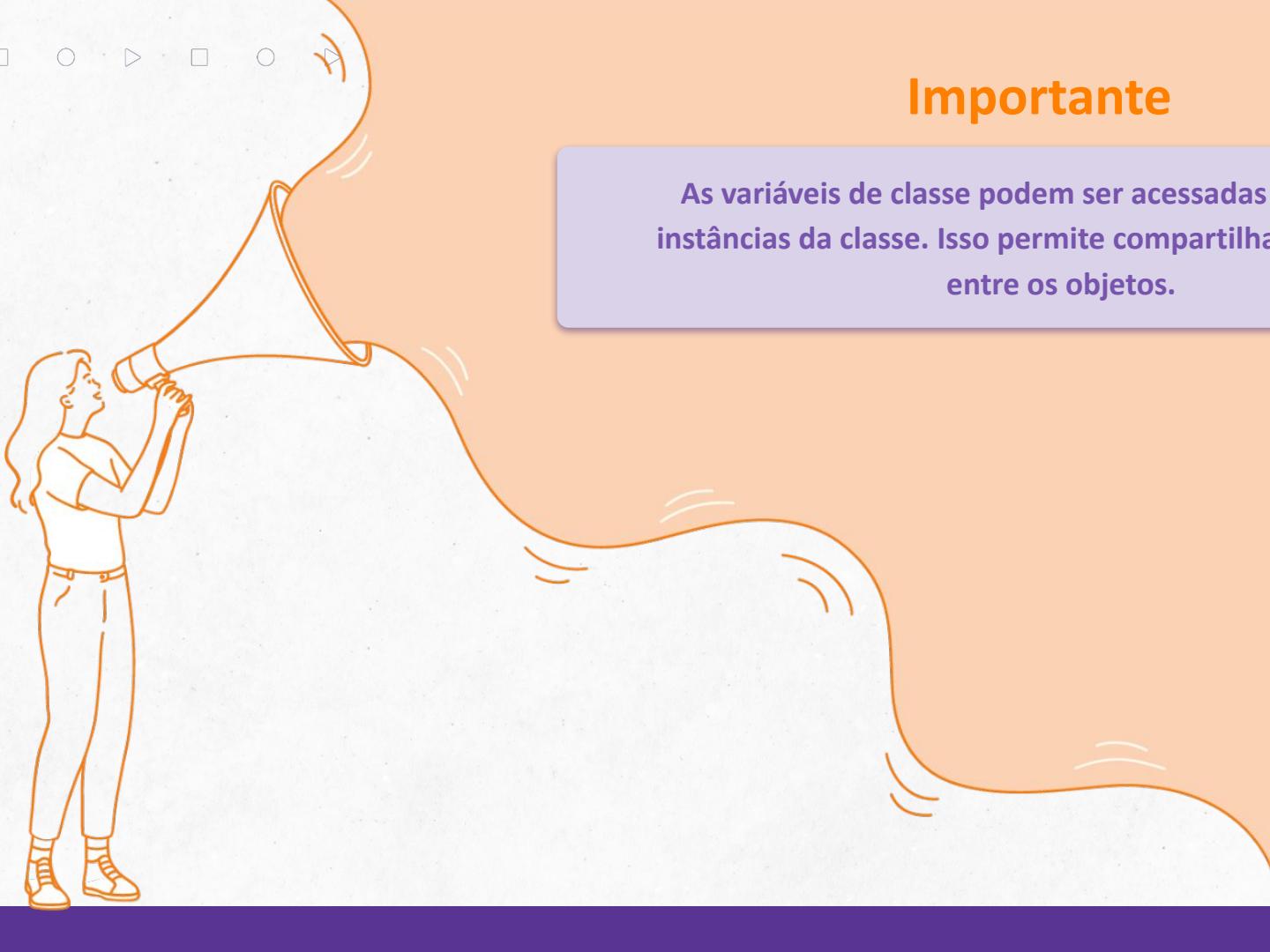


## Variáveis e constantes de classe

Compreender as variáveis de classe e constantes é essencial para criar um código mais organizado e eficiente em Java. As constantes são utilizadas para armazenar as informações comuns a todos os objetos da classe.



## Importante



As variáveis de classe podem ser acessadas por todas as instâncias da classe. Isso permite compartilhar informações entre os objetos.

# Declarando variáveis de classe

- **Declaração:** usar a palavra-chave '**static**' antes da declaração da variável.  
Exemplo: "**static int contador;**";
- **Acesso:** são acessadas usando o nome da classe, não sendo necessário criar uma instância.



## Importante



Temos os modificadores das variáveis antes da declaração, elas definem se o acesso à variável é público, privado ou protegido. Ou seja, a palavra “*private*” é um modificador de acesso, que também pode ser “*public*” e “*protected*”.

# Constantes: conceito e uso

## Constantes



Valores imutáveis que não podem ser alterados após a inicialização.

## Benefícios



Melhora a legibilidade do código e evita alterações acidentais.

## Exemplos



Valores de taxas, valores constantes em cálculos.

# Declarando constantes

- **Declaração de constantes:** usar a palavra-chave '**final**' antes da declaração. Exemplo: "static final double TAXA\_JUROS = 0.05;"
- **Acesso:** são acessadas usando o nome da classe, não sendo necessário criar uma instância.



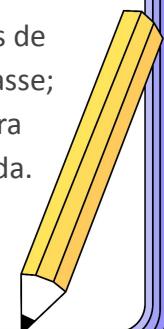
## Discussões e reflexões



# Uso de variáveis de classe e constantes

**Benefícios:** economia de memória ao compartilhar valores entre instâncias.

- **Exemplo 1:** controle de instâncias de uma classe usando variável de classe;
- **Exemplo 2:** uso de constantes para armazenar um valor que não muda.



# Bloco 4

---

► □ ○ ▶ □ ○ ▷ □ ○ ▶ □ ○ ▶ □ ○ ▷ □ ○ ▶ □ ○ ▷ □ ○ ▷ □ ○ ▷

## O que vem à sua mente sobre variáveis e constantes?



## Tente responder

Você acha que é possível realizar cálculos, comparações e avaliações lógicas em um programa? Você consegue dar um exemplo de programa que faz isso?

??



# Operadores aritméticos, relacionais e lógicos

Operadores aritméticos, relacionais e lógicos são fundamentais para realizar cálculos, comparações e avaliações condicionais em Java. Dominar o uso desses operadores permite aos desenvolvedores criar códigos mais eficientes e decisões lógicas precisas.



# Operadores aritméticos

## O que fazem:

os operadores aritméticos, como adição, subtração, multiplicação e divisão, são usados para realizar operações matemáticas com valores numéricos.

## Quais são:

adição (+), subtração (-), multiplicação (\*), divisão (/), módulo (%).

## Exemplo:

calcular o valor total de uma compra com desconto de 10%.

Solução: `double valorCompra = 100.0;`  
`double desconto = 0.1; double valorFinal =`  
`valorCompra - (valorCompra * desconto);`

# Operadores de incremento

## O que fazem:

os operadores de incremento e decremento podem ser utilizados para incrementar ou decrementar, de um em um, uma variável numérica.

## Quais são:

Incremento(++) e decremento (--)

## Exemplo:

Contar de 1 em 1:

Solução:

```
int num = 0;
```

```
num++;
```

# Operadores relacionais

## O que fazem:

os operadores relacionais, como igual, diferente, maior e menor, são usados para comparar valores e produzir resultados booleanos (verdadeiro ou falso).

## Quais são:

Igual (`==`), Diferente (`!=`), Maior (`>`), Menor (`<`), Maior ou Igual (`>=`), Menor ou Igual (`<=`).

## Exemplo:

Verificar se um número é par.

Solução: `int numero = 6; boolean ehPar = numero % 2 == 0;`

# Operadores lógicos

## O que fazem:

os operadores lógicos são usados para combinar expressões booleanas e realizar avaliações condicionais mais complexas.

## Exemplo:

`idade >= 18 && idade <= 30.`

## Quais são:

**AND (&&):** verdadeiro se ambas as expressões forem verdadeiras.

**OR (||):** verdadeiro se pelo menos uma expressão for verdadeira.

**NOT (!):** inverte o valor booleano de uma expressão.

# Expressões booleanas

## O que fazem:

os operadores relacionais são frequentemente usados em expressões booleanas para avaliar condições. Isso é essencial para a execução condicional de trechos de código.

## Exemplo:

Expressões booleanas: avaliam se uma condição é verdadeira ou falsa.

Uso de operadores relacionais: comparação entre valores para tomar decisões condicionais.

Exemplo: **idade > 18 && possuiCNH**



### Caso 1

**Problema:** verificar se uma pessoa tem idade suficiente para votar (idade maior ou igual a 18) e se possui menos de 70 anos.

### Caso 2

**Problema:** calcular a média de três números e verificar se a média é maior que 7.

### Caso 3

**Problema:** calcular o preço final de um produto com base no valor original e aplicar um desconto de 10% se o cliente for um membro *premium*.

# Bloco 5

---

# Quais são os tipos de operadores? O que cada um faz?



# Precedência de operadores

A precedência de operadores determina a ordem em que as operações são avaliadas em uma expressão. Isso é importante para garantir que as expressões sejam avaliadas corretamente e que o resultado seja o esperado. Entender a precedência e o uso de parênteses é essencial para cálculos precisos e evita equívocos em expressões complexas.



# Precedência

## Precedência

Determina a ordem em que os operadores são avaliados em uma expressão.

## Exemplo:

$$2 + 3 * 4$$

Qual é o resultado?

## Como assim?

Sabe a expressão matemática que você aprende na escola? Funciona da mesma forma. Quais são as contas que são realizadas primeiro? Multiplicação e divisão.

# Parênteses

## Precedência

Assim como nas contas matemáticas, nos códigos podemos usar os parênteses () para indicar uma prioridade no cálculo.

## Como assim?

Sabe quando você aprende que primeiro resolve o parêntese, depois a multiplicação e divisão e por fim, adição e subtração?  
Funciona da mesma forma.

## Exemplo:

$$2 + 3 * 4$$

Qual é o resultado?

$$(2 + 3) * 4$$

Qual é o resultado?

# Importância dos parênteses

Em expressões complexas, o uso adequado de parênteses **evita ambiguidades** e garante que as operações sejam realizadas na ordem desejada.

## Exemplo:

um aluno queria pegar os resultados de uma prova, que é dividido em duas notas e multiplicá-lo pelo peso que cada uma valia, para, assim, ter o resultado final. Porém, ele fez da seguinte forma:

**Exemplo:  $5 + 3 * 0,8$ .**

**Solução Correta:  $(5 + 3) * 0,8$**



### Caso 1

**Problema:** o aluno queria saber sua média final e recebeu três notas, mas deve dividir o total apenas por 2. Como ele resolve esse problema?

# Bloco 6

---

## Classe *scanner* e *math*

A classe **scanner** é fundamental para capturar dados da entrada padrão, enquanto a classe **math** oferece funções matemáticas pré-definidas em Java. Essas ferramentas são essenciais para a construção de aplicativos robustos e versáteis.



## Classe *scanner*

A classe **scanner** permite receber entradas de dados do usuário de forma interativa. É uma ferramenta essencial para criar programas interativos e dinâmicos.

### Exemplo de como utilizar:

```
import java.util.Scanner;  
  
Scanner scanner = new Scanner(System.in);  
  
System.out.print("Digite um número: ");  
  
int numero = scanner.nextInt();
```

## Vamos ver mais?

Pegue o papel e observe mais um exemplo de classe *scanner*.



# Uso de métodos da classe *scanner*

**'nextInt()'**

Captura de um número inteiro.

**'nextDouble()'**

Captura de um número decimal.

**'nextLine()'**

Captura de uma linha de texto.

## Classe *math*

A classe *math* fornece diversos métodos para executar operações matemáticas complexas, como cálculos de potência, raiz quadrada, valor absoluto etc.



# Classe Math

**Math.abs(...);**

Retorna o valor do número passado pelo parâmetro em módulo.

**Math.sqrt(...);**

Retorna a raiz quadrada do número passado.

**Math.pow(...);**

Para uma estrutura de potenciação  $a^b$ . Este método retorna o 'a' como base e o 'b' como expoente.

**Math.random(...);**

Um número aleatório que vai de 0 até 1 (0 incluído, 1 nunca será gerado);

# Exemplos

**Math.abs(...);**

```
System.out.println("Metodo abs(-30):  
    " + Math.abs(-30) );
```

**Saída:** Metodo abs(-30): 30

**Math.sqrt(...);**

```
System.out.println("Metodo sqrt(16):  
    " + Math.sqrt(16) );
```

**Saída:** Metodo sqrt(16): 4.0

**Math.pow(...);**

```
System.out.println("Metodo pow(2,3):  
    " + Math.pow(2,3) );
```

**Saída:** Metodo pow(2,3): 8.0

**Math.random(...);**

```
System.out.println("Metodo  
random(): " + Math.random() );
```

**Saída:** Metodo random():  
0.6100207813062897

```
java Copy code

import java.util.Scanner;

public class CalculoMedia {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Digite o primeiro número: ");
        double num1 = scanner.nextDouble();

        System.out.print("Digite o segundo número: ");
        double num2 = scanner.nextDouble();

        System.out.print("Digite o terceiro número: ");
        double num3 = scanner.nextDouble();

        double media = (num1 + num2 + num3) / 3;
        System.out.println("A média dos números é: " + media);

        scanner.close();
    }
}
```

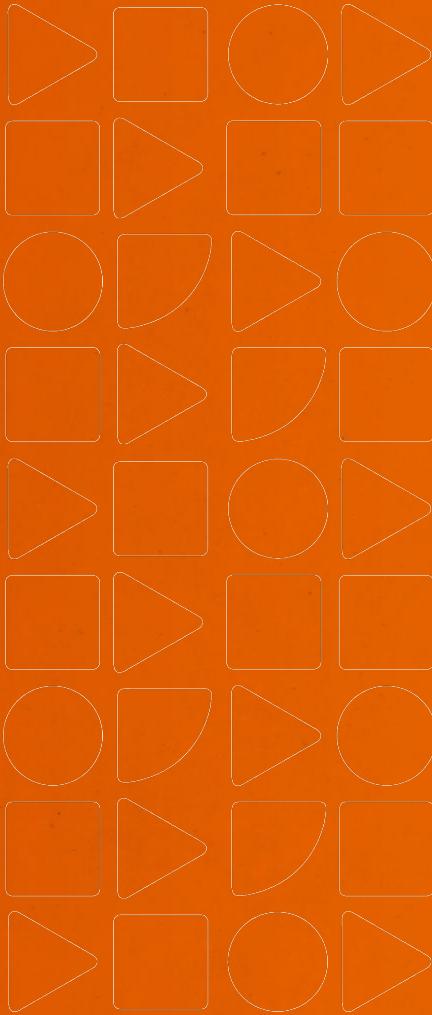


**Dividam-se em grupos e criem um pequeno programa que solicite um número ao usuário, calcule a sua raiz quadrada usando a classe *math* e exiba o resultado.**

## Fechamento

Diga uma palavra sobre o que você achou da aula.





# Referências Bibliográficas

PROZ EDUCAÇÃO. *Apostila de Desenvolvimento de Aplicações*. 2023.