

A decorative wavy line in yellow and white on the left side of the cover.

MÉTODOS

MICHELLE NERY NASCIMENTO
PUC MINAS
ALGORITMOS E TÉCNICAS DE PROGRAMAÇÃO

DEFINIÇÃO

- **Método** é um conjunto de comandos, com uma função **bem definida** e o **mais independente possível** em relação ao resto do programa.

MOTIVAÇÃO

- Desenvolvimento de programas por fases ou refinamentos sucessivos (“dividir para conquistar”).
- Reaproveitamento de código.

BENEFÍCIOS

- A independência do módulo permite uma manutenção mais simples e evita efeitos colaterais em outros pontos do programa.
- Testes e correções podem ser feitos em separado.
- Um módulo independente pode ser utilizado em outros programas que requeiram o mesmo processamento.

SINTAXE

[**tipo de retorno**] [**nome**] (**[lista de
parâmetros]**)

{

lista de comandos;

}

SINTAXE

- **[tipo de retorno]**: tipo de dado retornado pelo método. Ex.: `int`, `char`, `double`, `float` ...
Se o método não retornar um valor, seu tipo deve ser **`void`**.
- **[nome]**: O nome do método segue as mesmas regras do nome das variáveis. A primeira letra deve ser minúscula.
- **[lista de parâmetros]**: pode ter zero ou mais parâmetros, sendo que cada um é composto por seu tipo e uma variável. Devem ser separados por vírgula.
 - Ex.: **`double`** metodoDoidao (**`int`** x, **`int`** y, **`double`** z, **`char`** m)
 - Cuidado: frequentemente, os alunos erram e colocam (`int` x, y) **<=ERRO!**

OBSERVAÇÕES

- A definição do método especifica os nomes e tipos de quaisquer parâmetros obrigatórios.
- Os parâmetros funcionam como **variáveis locais** dos métodos.
- Os métodos são chamados passando a quantidade (e tipos) de parâmetros necessários.
- Quando a chamada de método é concluída, o método retorna um resultado, ou simplesmente o controle, ao seu chamador.

OBSERVAÇÕES

- Quando o código de chamada chama o método, ele fornece valores concretos, chamados **argumentos**, para cada parâmetro.
 - Se o método não tiver parâmetros, a lista de argumentos, estará vazia.
 - Os argumentos devem ser compatíveis com o tipo de parâmetro.

OBSERVAÇÕES

- Quando um método é chamado, o programa faz uma cópia dos valores de argumento do método e os atribui aos parâmetros correspondentes do método.
- Quando o controle do programa retorna ao ponto em que método foi chamado, os parâmetros do método são removidos da memória.
- Um método pode retornar no máximo um valor.

OBSERVAÇÕES

- Se o método não retornar um resultado (tipo **void**), o controle retornará quando o fluxo do programa alcançar a chave direita de fechamento do método ou quando a instrução

`return;`

for executada.

- Se o método retornar um resultado (ou seja, diferente de **void**), a instrução

`return expressão;`

avalia a *expressão* e então imediatamente retorna o valor resultante ao chamador.

EXEMPLO

```
void myMeth() {  
    for(int i=0; i < 10; i++) {  
        if(i == 5) return;    // para em 5  
        System.out.println(i);  
    }  
}
```

- Aqui, o laço for só será executado de 0 a 5, porque quando i for igual a 5, o método retornará.

EXEMPLO

- Podemos ter várias instruções return em um método, principalmente se houver duas ou mais saídas dele.:

```
void myMeth() {  
    // ...  
    if(done) return;  
    // ...  
    if(error) return;  
    // ...  
}
```

- Nesse caso, o método retorna ao terminar ou se um erro ocorrer. No entanto, tome cuidado, porque a existência de muitos pontos de saída em um método pode desestruturar o código; por isso evite usá-los casualmente. Um método bem projetado tem pontos de saída bem definidos.

ESCOPO DE VARIÁVEIS

- **Variáveis globais:** valem a partir do ponto em que foram declaradas.
- **Variáveis locais:** valem dentro do { e } em que foram declaradas.
 - Ex.: O escopo de uma declaração de variável local que aparece na seção de inicialização do cabeçalho de uma instrução `for` é o corpo da instrução `for` e as outras expressões no cabeçalho.

ALGUNS MÉTODOS VISTOS

- **double pow(double base, double expoente)**
- **double sqrt(double numero)**

EXEMPLO 1

```
public static void soma(double n1, double n2)
{
    double resultado = n1 + n2;
    System.out.println("O resultado é: " + resultado);
}

public static void main(String[] args) {
    Scanner leia = new Scanner(System.in);
    System.out.println("Digite o primeiro número: ");
    double n1 = leia.nextDouble();
    System.out.println("Digite o segundo número: ");
    double n2 = leia.nextDouble();
    soma(n1, n2);
}
```

EXEMPLO 2

```
import java.util.Scanner;
public class ExemploMetodo {
    static void Main(string[] args) {
        Scanner sc = new Scanner (system.in);

        int num1, num2, maior;
        System.out.println("Digite um número inteiro:");
        num1 = sc.nextInt();

        System.out.println("Digite outro número inteiro:");
        num2 = sc.nextInt();

        maior = maximo(num1, num2);

        System.out.println("Maior: " + maior);
    }

    public static int maximo(int a, int b)
    {
        int resposta;

        if (a > b)
            resposta = a;
        else
            resposta = b;

        return resposta;
    }
}
```


EXERCÍCIO 1

- Faça um método que mostre na tela os n primeiros números inteiros, positivos e ímpares em ordem **crescente**.
 - Este método deve conter/chamar um outro método chamado “Ímpar” que calcula os ímpares.

EXERCÍCIO 1

```
public static int impar(int n)
{
    int resp = 2 * n + 1;
    return resp;
}
```

```
public static void imparesCrescente (int n)
{
    for(int i = 0; i < n; i++)
    {
        System.out.println (impar(i));
    }
}
```

EXERCÍCIO 2

- Faça um método que mostre na tela os n primeiros números inteiros, positivos e ímpares em ordem **decrescente**.
 - Este método deve conter/chamar um outro método chamado “impar” que calcula os ímpares.

EXERCÍCIO 2

```
public static int impar(int n)
{
    int resp = 2 * n + 1;
    return resp;
}
```

```
public static void imparesDecrescente (int n)
{
    for(int i = n-1; i >= 0; i--)
    {
        System.out.println (impar(i));
    }
}
```

EXERCÍCIO 3

- Faça um método que retorne o fatorial de um número natural n .

EXERCÍCIO 3

```
public static double fatorial(int n)
{
    double fat = 1;
    for(int i = n; i >= 1; i--)
    {
        fat *= i;
    }
    return fat;
}
```

EXERCÍCIO 4

- Faça um método “**Termo**” que retorne o **i-ésimo** termo da sequência abaixo:

$$\frac{1}{3*5}, \frac{2}{3*5^3}, \frac{4}{3*5^9}, \frac{8}{3*5^{27}}, \dots$$

- Considere a contagem dos termos: 0, 1, 2, 3, ...

EXERCÍCIO 4

```
public static double termo(int i)
{
    double resp = Math.pow(2,i);
    resp /= (3 * Math.pow(5,(Math.pow(3,i))));
    return resp;
}
```


EXERCÍCIO 5

- Faça um método “**MostrarSequência**” que mostre os n primeiros termos da sequência abaixo. Aproveite o método “**Termo**” anterior.

$$\frac{1}{3*5}, \frac{2}{3*5^3}, \frac{4}{3*5^9}, \frac{8}{3*5^{27}}, \dots$$

EXERCÍCIO 5

```
public static double termo(int i)
```

```
{
```

```
    double resp = Math.pow(2,i);
```

```
    resp /= (3 * Math.pow(5,(Math.pow(3,i))));
```

```
    return resp;
```

```
}
```

```
public static void mostrarSequencia(int n)
```

```
{
```

```
    for(int i = 0; i <= n; i++)
```

```
    {
```

```
        System.out.println(termo(i));
```

```
    }
```

```
}
```

EXERCÍCIO 6

- Faça um método “somaTermo” que efetue o **somatório** dos n primeiros termos da sequência abaixo:

$$\frac{1}{3*5}, \frac{2}{3*5^3}, \frac{4}{3*5^9}, \frac{8}{3*5^{27}}, \dots$$

- Aproveite novamente o método “**Termo**” anterior.

EXERCÍCIO 6

```
double termo(int i)
```

```
{
```

```
    double resp = Math.pow(2,i);
```

```
    resp /= (3 * Math.pow(5,(Math.pow(3,i))));
```

```
    return resp;
```

```
}
```

```
double somaTermo(int n)
```

```
{
```

```
    double soma = 0;
```

```
    for(int i = 0; i <= n; i++)
```

```
    {
```

```
        soma += termo(i);
```

```
    }
```

```
    return soma;
```

```
}
```

EXERCÍCIO 7

- Faça um método “produto” que efetue o **produtório** dos n primeiros termos da sequência abaixo:

$$\frac{1}{3*5}, \frac{2}{3*5^3}, \frac{4}{3*5^9}, \frac{8}{3*5^{27}}, \dots$$

- Aproveite novamente o método “**Termo**” anterior.

EXERCÍCIO 7

```
public static double termo(int i)
```

```
{
```

```
    double resp = Math.pow(2,i);
```

```
    resp /= (3 * Math.pow(5,(Math.pow(3,i))));
```

```
    return resp;
```

```
}
```

```
public static double produto(int n)
```

```
{
```

```
    double produto = 1;
```

```
    for(int i = 0; i <= n; i++)
```

```
    {
```

```
        produto *= termo(i);
```

```
    }
```

```
    return produto;
```

```
}
```

EXERCÍCIO 8

- Faça a chamada de todos os métodos anteriores pelo método principal do programa `Main()`.

EXERCÍCIO 8

```
static void main(string[] args){  
    int n;  
    System.out.println ("Entre com o valor de n inteiro:");  
    n = sc.nextInt();  
    imparesCrescente (n);  
    imparesDecrescente (n);  
    System.out.println ("Fatorial de n:" + fatorial(n));  
    System.out.println ("termo:" + termo(n));  
    mostrarSequencia(n);  
    System.out.println ("somatorio:" + somaTermo(n));  
    System.out.println ( "produto:" + produto(n));  
}
```


EXERCÍCIOS

- Ler x Receber e Mostrar x Retornar
- Pode implicar em:
 - Método sem retorno e sem parâmetro;
 - Método com retorno e sem parâmetro;
 - Método sem retorno e com parâmetro;
 - Método com retorno e com parâmetro.

EXERCÍCIO 1

- Faça um método que **leia** 2 números e **mostre** a soma deles

EXERCÍCIO 1

Leia/Mostre (Método sem retorno e sem parâmetro)

```
void metodo1()
{
    int n1, n2;
    ler: n1, n2;
    escrever: n1+n2;
}

void main()
{
    metodo1();
}
```

EXERCÍCIO 2

- Faça um método que **receba** 2 números e **mostre** a soma deles.

EXERCÍCIO 2

Receba/Mostre (Método sem retorno e com parâmetro)

```
public static void metodo2(int num1, int num2)
{
    imprima (num1+num2);
}

Void main()
{
    int a,b;
    metodo2(a, b);

}
```

EXERCÍCIO 3

- Faça um método que **receba** 2 números e **retorne** a soma deles.

EXERCÍCIO 3

Receba/Retorne (Método com retorno e com parâmetro)

```
int metodo3(int n1, int n2)
```

```
{
```

```
    return (n1+n2);
```

```
}
```

```
void main()
```

```
{
```

```
    int a, b, somaFinal;
```

```
    somaFinal = metodo3(a, b);
```

```
    imprima ("soma=" + somaFinal);
```

```
}
```

EXERCÍCIO 4

- Faça um método que **receba** 2 números e **retorne** a soma deles.

EXERCÍCIO 4

Receba/Retorne (Método com retorno e com parâmetro)

```
int metodo4(int n1, int n2)
{
    return (n1+n2);
}
```

```
Void main()
{
    int a, b, soma;
    ler a, b;
    soma = metodo4(a, b);
}
```

EXERCÍCIO 5

- Faça um método *int multiploCinco(int n)* que **recebe** um número inteiro *n* e **retorna** o *n*-ésimo múltiplo de cinco.

EXERCÍCIO 5

```
int multiploCinco(int n)
{
    return n * 5;
}
```

EXERCÍCIO 6

- Faça um método *void exemplo00()* para ler um número inteiro n e mostrar o n-ésimo múltiplo de cinco que será calculado usando o método anterior

EXERCÍCIO 6

```
void exemplo00()  
{  
    int n, multiplo;  
    ler: n;  
    multiplo = multiploCinco(n);  
    escrever: multiplo;  
}
```

EXERCÍCIO 7

- Faça um método **void** *mostrarMultiploCinco(int n)* que **recebe** um número inteiro **n** e **mostra** na tela os **n** primeiros múltiplos de cinco.

EXERCÍCIO 7

```
void mostrarMultiploCinco(int n)
{
    for(int i = 0; i < n; i++)
    {
        escrever: multiploCinco(i);
    }
}
```

EXERCÍCIO 8

- Faça um método *int* ***multiploTresMaisUm(int n)*** que **recebe** um número inteiro *n* e **retorna** o *n*-ésimo múltiplo de três mais um.

EXERCÍCIO 8

```
int multiploTresMaisUm(int n)
{
    return (n * 3) + 1;
}
```

EXERCÍCIO 9

- Faça um método **void** *exemplo02()* para **ler** um número inteiro *n* e **mostrar** o *n*-ésimo múltiplo de três mais um que será calculado usando o método anterior

EXERCÍCIO 9

```
void exemplo02(){  
    int n, multiplo;  
    ler: n;  
    multiplo = multiploTresMaisUm(n);  
    escrever: multiplo;  
}
```

EXERCÍCIOS

- 1) Faça um método que calcule e imprima a multiplicação de 2 valores inteiros recebidos.
- 2) Faça um método para realizar a multiplicação e a divisão de 2 números inteiros. Retorne e imprima os dois resultados.
- 3) Faça um programa contendo uma sub-rotina que retorne 1 se o número digitado for positivo ou 0 se for negativo.