

# TI-220 Java Orientado a Objetos

---

ANTONIO CARVALHO - TREINAMENTOS

A solid blue horizontal bar spanning the width of the slide, located at the bottom.

# Métodos

---

# Métodos

---

- Os métodos são rotinas ou funções que possuem acesso a instância do objeto, podendo alterar suas características e invocar outros comportamentos. Através do uso de métodos é possível dividir e organizar o programa em rotinas, facilitando a compreensão do código.
- Os métodos são identificados por um nome e podem receber (*input*) e retornar informações (*output*).
- No Java os métodos:
  - Podem ou não retornar objetos
  - Possuem *namespace* próprio portanto as variáveis criadas dentro do método fazem parte de um escopo local

# Métodos

---

Os métodos no Java declaram o tipo de informação a ser retornada assim como em outras linguagens fortemente tipadas.

O conjunto de informações correspondente ao **tipo da informação retornada**, assim como o **nome do método** e seus **tipos de parâmetros de entrada** são conhecidos como assinatura.

**int**   **fazAlgo(** **boolean parametro1,**   **String parametro2)**

Retorno   Nome   Tipos dos parâmetros

(*output*)   (*input*)

# Métodos

---

## Sintaxe:

```
[modificadores] <tipo de informação retornada> <nome do método> ([<Tipo  
paramentro1> <Nome Paramentro1>,...[<Tipo paramentroN> <Nome paramentroN>])  
{  
    return [<valor>];  
}
```

## Exemplo :

```
boolean matricularDisciplina( String disciplina ) {  
    System.out.println("Matricula efetivada em " + disciplina);  
    return (true);  
}
```

# Métodos

---

O método pode ser invocado pelo nome e um par de parenteses contendo ou não parâmetros, obedecendo a forma como foi declarado.

```
Aluno a = new Aluno();  
boolean b = a.matricularDisciplina( "Matematica" );  
System.out.println( b );  
System.out.println( a.matricularDisciplina("POO") );
```

# Métodos

- Os parâmetros passados para a função são atribuídos aos identificadores através da ordem em quem foram passados.

- Exemplo:

```
void fazAlgo( int numero1, int numero2, float numero3 ) {  
    System.out.println( "Este é o 1o parametro informado :" + numero1);  
    System.out.println( "Este é o 2o parametro informado :" + numero2);  
    System.out.println( "Este é o 3o parametro informado :" + numero3);  
}  
  
ClasseExemplo c = new ClasseExemplo();  
c.fazAlgo(10, 10, 4.5);
```

- Nesta caso o número float foi para a 3ª variável e os números inteiros para a 1ª e 2ª variável.

# Métodos

Os parâmetros de tipos primitivos são passados por valor (*passed by value*) é feita uma cópia do valor para a variável do parâmetro. As modificações não afetam a variável original.

```
void executa() {  
    int x = 10;  
    int y = 20;  
    int z = 30;  
    fazAlgo( x, y, z );  
}  
void fazAlgo( int a, int b, int c ) {  
    a = a + 5;  
    System.out.println( "Valor A: " + a );  
    System.out.println( "Valor B:" + b );  
    System.out.println( "Valor C:" + c );  
}
```



# Métodos

---

Já os parâmetros de tipos complexos são passados por referência, de maneira que se as propriedades internas do objeto forem alteradas isto afeta o objeto original (pois é o mesmo).

```
void executa() {  
    StringBuffer s = new StringBuffer( "Linha1 \n" );  
    acrescentaLinha( s );  
    System.out.println( "Texto: " + s.toString() );  
}  
void acrescentaLinha( StringBuffer texto ) {  
    texto.append( "Linha 2 \n" );  
}
```

# Métodos que usam Lista Variável de Argumentos (var-args)

---

Através do uso do var-args é possível passar um número variável de argumentos no momento de chamada de um método.

**parâmetro** : informação solicitada pelo método no momento em que é declarado, compõem sua assinatura

EX:

```
class Funcionario {  
    float salario;  
    public void adicionarBeneficios( float a, float b ) {  
        salario += a;  
        salario += b;  
    }  
}
```

**argumento** : informação passada ao método no momento em que ele é chamado.

EX:

```
Funcionario f = new Funcionario();  
f.adicionarBeneficios( 100.0, 200.0 );
```

# Métodos que usam Lista Variável de Argumentos (var-args)

---

Sintaxe :

```
[modificador de acesso] [outros modificadores] <tipo> <nome> ([ [<tipo1> <parametro1>],  
    [<tipo2> <parametro2>],  
    [<tipo N-1> <parametroN-1>],  
  
    [<tipo var-args>... <parametro var-args>]  
]);
```

Tipo var-args pode ser um tipo primitivo ou uma objeto

**Regras:**

1. Pode haver outros parâmetros em um método que espera um var-args, porém o parâmetro var-args precisa ser o último parâmetro esperado.
2. So pode haver um parâmetro var-args por método

# Métodos que usam Lista Variável de Argumentos (var-args)

---

Exemplo de métodos que utilizam var-args

```
public int somar (int... valores) { }  
public int somareElevar(int exp, int... valores){ }  
public String concatenar(String... textos){ }
```

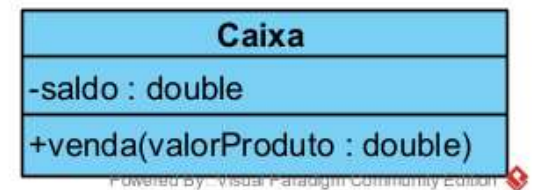
O parâmetro do tipo var-args é recebido pelo método como sendo um vetor

```
class Funcionario {  
    float salario;  
    public void adicionarBeneficios( float ... beneficios ) {  
        for (float bene : beneficios) {  
            salario += bene;  
        }  
    }  
}
```

# Métodos - Exercício

---

Crie uma classe chamada Caixa que faça venda de produtos, conforme o layout ao lado. A função da venda deverá acrescentar o valor do produto no saldo, e deverá retornar o valor do novo saldo

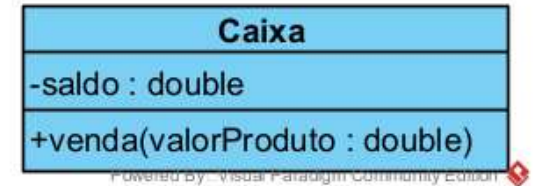


# Métodos - Exercício

---

Com base no exercício anterior o seu cliente pediu para que as vendas possam receber diversos produtos simultaneamente.

Faça a modificação no método venda de maneira que possa somar os valores de diversos produtos de uma única vez.



# Métodos - Exercício

---

Nosso calendário atual é Gregoriano, mas vamos trabalhar com o calendário Juliano.

Faça uma classe chamada `ConversorData` contendo 3 atributos: ano, mês e dia, todos do tipo **int**.

Crie um construtor para esta classe de maneira a receber estas 3 informações e colocá-las nos atributos criados anteriormente

Crie um método chamado `dataJuliana` que utilize os atributos e retorne o número do dia correspondente no calendário Juliano

Fórmula :

$$\begin{aligned} \text{diaJuliano} = & ( 1461 * ( \text{ano} + 4800 + ( \text{mes} - 14 ) / 12 ) ) / 4 + \\ & ( 367 * ( \text{mes} - 2 - 12 * ( ( \text{mes} - 14 ) / 12 ) ) ) / 12 - \\ & ( 3 * ( ( \text{ano} + 4900 + ( \text{mes} - 14 ) / 12 ) / 100 ) ) / 4 + \text{dia} - 32075 \end{aligned}$$

Para testar crie um objeto do tipo `ConversorData` com os valores para a data de início das olimpíadas do Brasil 05 de Agosto de 2016, e ao invocar o método `dataJuliana()` o resultado deverá ser : 2457606

# Dúvidas

---

