Curso de Java

Métodos

- Declaração
- Ativação
- Variáveis locais
- Valor de retorno
- Passagem de parâmetros
- Parâmetros de saída
- Número variável de parâmetros
- Variáveis globais

- *Métodos* em Java são análogos a *funções* e *procedimentos* em outras linguagens como VB e Pascal.
- O uso do nome *método* ao invés de *função* ou *procedimento* está relacionado aos conceitos de orientação a objetos, já comentados.

Um exemplo


```
static int mdc(int a, int b){
   while(a != b)
         if(a > b) a -=b; else b -=a;
   return a;
}
```

- Neste exemplo
  - **int mdc** define o nome (mdc) e o tipo do *valor retornado* (**int**) pelo método **mdc**.
  - **(int a, int b)** define os *parâmetros* (a e b, inteiros) usados pelo método.
  - o modificador **static** indica que o método não está vinculado a nenhum objeto (objetos serão tratados mais adiante).
  - os comandos entre **{** e **}** correspondem ao 'corpo' do método e serão executados cada vez que o mesmo for ativado.
  - o comando **return** encerra a execução do método e define o valor devolvido pelo mesmo.

Métodos: Exemplo de uso


```
public class Exemplo {
        static int mdc(int a, int b){
                while(a != b) if(a > b) a -=b; else b -=a;
                return a;
        }

        public static void Main(string[] args){
                int x = 33; int y = 605;
                int m = mdc(x,y);
                System.out.println(
                                        "x:"+x+" y:"+y,
                                        "mdc:"+m
                                    );
        }
}
```

- Neste exemplo
  - o comando **int m = mdc(x,y);** atribui como valor inicial da variável **m** o valor retornado pela ativação do método feita em **mdc(x,y)**.
  - Os parâmetros usados nessa ativação são os valores de **x** e **y**: durante execução do método, o parâmetro **a** é substituído pelo ***valor*** de **x** e o parâmetro **b** é substituído pelo ***valor*** de **y**.

- Um método pode ter *variáveis locais*, que são variáveis declaradas 'dentro' do método.
- Variáveis locais podem ser utilizadas apenas *dentro* do método onde foram declaradas.
- Os *parâmetros* do método também são locais a ele.


```
static float media(float[] v){
        float s = 0;
        for(int i = 0; i < v.length; i++) s += v[i];
        return s/v.length;
}
```

- Neste exemplo
  - **v**, **s** e **i** são variáveis locais ao método **media**.
  - **v** é parâmetro do método, que é usado como uma variável local.

Outro exemplo


```
public class Exemplo{
  static float media(float[] v){
    float s = 0.0F;
    for(int i = 0; i < v.length; i++) s += v[i];
    return s/v.length;
  }
  public static void main(String[] args){
    float[] dados = { 5.0F, 3.5F, 4.2F, 0.1F,
                      9.2F, 10.8F, 12.0F
                    };
    float m = media(dados);
    System.out.println("media:"+m);
  }
}
```

- Métodos diferentes podem ter parâmetros ou variáveis locais com nomes comuns.
- Nesse caso, cada método tem os seus próprios parâmetros e variáveis locais, independentemente de os nomes serem os mesmos.

- Todos exemplos de métodos apresentados até aqui retornam um valor ao serem chamados.
- Esse valor pode ser usado para atribuir um valor a uma variável ou mesmo numa expressão, como por exemplo:

```
m = mdc(x,y)*10;
```

- Em alguns casos, não se deseja que o método retorne um valor. Nesse caso, o tipo retornado pelo método é definido como **void**.
- Exemplo:

```
static void writeMax(float[] v){
    float m = v[0];
    for(int i = 1; i < v.length; i++)
        if(v[i] > m) m = v[i];
    System.out.println("maximo:"+m);
}
```

Um exemplo


```
static void writeMax(float[] v){
  float m = v[0];
  for(int i = 1; i < v.length; i++) if(v[i] > m) m = v[i];
  System.out.println(" maximo:"+m);
}

static void writeMin(float[] v){
  float m = v[0];
  for(int i = 1; i < v.length; i++) if(v[i] < m) m = v[i];
     System.out.println(" minimo:"+m);
}

public static void main(String[] args){
  float[] dados = { 5.0F, 3.5F, 4.2F, 0.1F,
                    9.2F, 10.8F, 12.0F };
   writeMin(v);
   writeMax(v);
}
```

- Nos métodos apresentados até agora, os parâmetros são *passados por valor*.
- Exemplo: numa chamada de método como

```
m = mdc(x,y);
```

os valores das variáveis x e y são atribuídos aos parâmetros a e b do método mdc.

- Numa chamada a um método com parâmetros passados *por valor*, o valor passado como parâmetro pode ser uma constante ou mesmo o resultado de uma expressão, como por exemplo:

```
m = mdc(115,x+y*8);
```

- Durante a execução do método mdc, o valor dos parâmetros é alterado. As alterações são feitas nas variáveis locais do método, correspondentes aos parâmetros e não afetam as variáveis usadas na chamada ao método.

Em Java, parâmetros de tipos primitivos sempre são passados por valor

- Ao se passar um vetor como parâmetro *por valor*, os seus elementos se comportam como se fossem passados por referência.
- Isso significa que se alterarmos o valor de um elemento do vetor dentro do método, essa alteração vai ocorrer também no elemento do vetor passado como parâmetro.


```
public static void inverte(int[] v){
        int k = v.length-1;
        for(int i = 0; i < v.length/2; i++){
                int t = v[i];
                v[i] = v[k-i];
                v[k-i] = t;
        }
}
```

- Após uma chamada a este método, como por exemplo em `"inverte(w);"`, os elementos do vetor **w** serão invertidos.

- Os exemplos apresentados até aqui usam variáveis locais aos métodos.
- Em Java é possível definir variáveis ‘fora’ dos métodos, ou atributos das classes.
- Os atributos podem ser usadas pelos métodos da classe.