MÉTODOS DA CLASSE STRING

Método length(): retorna um número inteiro com o tamanho da String

```
String nome = JOptionPane.showInputDialog("Digite seu nome:");
int tamanho = nome.length();
JOptionPane.showMessageDialog(null, "Método length retorna quantidade de caracteres incluindo espaços \n " +   "Seu nome possui " + tamanho + " caracteres");
```

Método charAt(): retorna o caracter de uma string de acordo com um índice especificado entre () útil para se verificar a existência de um caracter na string, Ex. suponha que uma determinada string só possa conter números.

```
int x = Integer.parseInt(JOptionPane.showInputDialog("Digite o índice:"));
JOptionPane.showMessageDialog(null, "Método charAr(i): " + nome.charAt(x));
```

Métodos toUpperCase() e toLowerCase() transformam a String em maiúscula ou minúscula. A transformação é apenas para efeito de tela, não muda o conteúdo da variável.

```
JOptionPane.showMessageDialog(null, "toUpperCase: Maiúscula" + nome.toUpperCase());
JOptionPane.showMessageDialog(null, "toLowerCase: Minúscula: " + nome.toLowerCase());
```

Para alterar o conteúdo, a própria variável tem de receber a transformação:

```
JOptionPane.showMessageDialog(null, "" + nome);
nome = nome.toLowerCase();  // transformando em minúscula
JOptionPane.showMessageDialog(null, "" + nome);
nome = nome.toUpperCase();  // transformando em maiúscula
JOptionPane.showMessageDialog(null, "" + nome);
```

Método substring() retorna a cópia de caracteres de uma String indicando o ìndice inicial e o final, iniciando do índice zero, sendo que o caracter a ser exibido no final é sempre um anterior ao o índice indicado.

```
String frase = "Escola Técnica Professor Alfredo de Barros Santos";
inicia no índice 3 (g) e termina no índice 10 (j)
JOptionPane.showMessageDialog(null, "substring: " + frase.substring(3,11));
```

Se indicar índices fora dos limites, aparece o erro: **StringIndexOutOfBoundsException**

Sem o índice final, retorna a partir da posição informada, no caso 10.

```
JOptionPane.showMessageDialog(null, "substring: " + frase.substring(10));
```

Método trim() remove os espaços em branco no início e no final de uma string, mas não remove os espaços entre as palavras, usado para corrigir erros, como por ex. no preechimento de um formulário. Assim como toUpperCase e toLowerCase o efeito é somente de tela, sendo necessário atribuir o resultado a outra variável.

```
String nome2 = "          Retirando os espaços em branco          ";
nome2 = nome2.trim();
JOptionPane.showMessageDialog(null,"Método trim " + nome2);
```

Método replace(): substitui os caracter de uma string, somente na tela. É necessário informar os caracteres a serem substituídos e os que vão substituí-los. Não havendo ocorrência, não há alteração na string. Ex.substituir " " por ""

```
        JOptionPane.showMessageDialog(null, frase);
        JOptionPane.showMessageDialog(null, "Método replace: " + "\n" +
                frase.replace("a", "is"));
```

Método indexOf() localiza caracteres em uma String e retorna a posição(índice). Pode ser utilizado para se buscar uma palavra numa string. Pode-se ou não indicar o índice inicial da busca, caso não encontre nenhuma ocorrência, é retornado o índice -1. Há diferenciação entre maiúsculas e minúsculas.

String.indexOf(<caracter ou substring a ser localizada, <posição inicial>)

```
JOptionPane.showMessageDialog(null,"Método indexOf \n" + frase + "\n" + frase.indexOf("a")      +
" | " + frase.indexOf("a",10)  + " | " +  frase.indexOf("Barros")   + " | " + frase.indexOf("Santos",44) +
" | " + frase.indexOf("Escola"));
```

Método lastIndexOf() localiza a útima ocorrência de um determinado caracter dentro de uma String e retorna a posição(índice). Caso não ocorra a ocorrência, é retornado -1. Há diferenciação entre maiúsculas e minúsculas

String.lastIdexOf(<caracter ou substring a ser localizada)

```
int y =   frase.lastIndexOf("s");
JOptionPane.showMessageDialog(null, "Posição da última letra 's' dentro da
String frase: " + y);
```

Método contains() Retorna verdadeiro (true) se a sequência de caracteres do argumento existe no objeto e falso (false) caso contrário. Ex. procura a ocorrência da palavra 'de' dentro da variável frase

```
boolean contem = frase.contains("de");
```

```
JOptionPane.showMessageDialog(null,"Resultado do método contains: " +
contem);
```

Método isEmpty():retorna verdadeiro (true) se a String estiver vazia e falso (false) se String não estiver vazia.

```
boolean vazio = frase.isEmpty();
JOptionPane.showMessageDialog(null,"Resultado do método isEmpty: " +
vazio);
```

Método startsWith() verifica se uma String inicia-se com o prefixo informado como argumento, caso ela inicie, retorna verdadeiro (true) caso contrário falso (false)

```
boolean prefixo = frase.startsWith("Escola");
JOptionPane.showMessageDialog(null,"Resultado do método startsWith: " +
prefixo);
```

Método endsWith() verifica se o final de uma String é igual ao prefixo informado como argumento, caso ela seja, retorna verdadeiro (true) caso contrário falso (false)

```
boolean fim = frase.endsWith("Santos");
JOptionPane.showMessageDialog(null,"Resultado do método endsWith: " +
fim);
```

Método concat(): utilizado para concatenar 2 strings.

```
String cidade = " Guaratinguetá";
String aux = frase.concat(cidade);
JOptionPane.showMessageDialog(null,"Método concat() \n" + aux);
```

MÉTODOS DA CLASSE MATH

A Classe Math possui diversos métodos especializados em cálculos matemáticos.
Para usar alguns desses métodos, pode ser necessário importar a classe Math, que faz parte do pacote java.lang, disponível com o Java

```
import java.lang.Math;
```

Método ceil( ): Arredonda números tipo double e float para seu próximo inteiro

```
double n1=5.2, n2=5.6, n3= -5.8;
```

```
JOptionPane.showMessageDialog(null, "Arredondamento PARA CIMA com o método ceil: "
+ "\n " + n1 + " = " + Math.ceil(n1) +
   " - " + n2 + " = " + Math.ceil(n2)  +
   " - " + n3 + " = " + Math.ceil(n3));
```

Método floor( ): Arredonda números tipo double e float para seu inteiro anterior

```
JOptionPane.showMessageDialog(null, "Arredondamento PARA BAIXO com o método floor: " +
                                                    "\n" + n1 + " = " + Math.floor(n1) +
                                                    "\n" + n2 + " = " + Math.floor(n2) +
                                                    "\n" + n3 + " = " + Math.floor(n3));
```

Método round( ): arredonda para o inteiro mais próximo, exceto se a parte decimal é .5, nesse caso o arredondamento é para cima.

```
JOptionPane.showMessageDialog(null, "Método Math.round: " +
"\n" + n1 + " = " + Math.round(n1)  +
"\n" + n2 + " = " + Math.round(n2) +
 "\n" + n3 + " = " + Math.round(n3));
```

Método max( ): Verifica o maior número entre dois números.

Método min( ): Verifica o menor número entre dois números.

Pode ser comparado tipos de números diferentes

```
JOptionPane.showMessageDialog(null, "Maior e menor entre DOIS nos. com o método max e min: "
+ "\n" + "Maior: " + Math.max(n1, n2) + "\nMenor: " + Math.min(n1, n2));
```

Utilizando o método para comparar 3 números:

```
JOptionPane.showMessageDialog(null, "Maior de 3 números: " + "\n" + Math.max(n1,Math.max(n2,
n3)));
```

Método sqrt( ): Realiza o cálculo da raiz quadrada, resultando um número tipo double

```
int a = 9;

JOptionPane.showMessageDialog(null, "Raiz quadrada com o método sqrt: " +
                                                    "\n" + Math.sqrt(a));
```

Método pow( ): realiza cálculo de potenciação.

```
JOptionPane.showMessageDialog(null, "Cálculo de potência com o método pow: " +
                                                    "\n" + Math.pow(a,2));
```

Método random( ): gera nos. aleatórios (tipo double) de 0.0 a 1.0 (1 nunca é sorteado).
Colocando int na frente, o número é convertido em inteiro.

O exemplo abaixo gera um número inteiro de 0 a 99 (por isso está multiplicado por 100)
O número 100 nunca será sorteado.

```
JOptionPane.showMessageDialog(null, "Método random " +
                                                "\n" + (int)(Math.random() * 100));
```