


Java básico

Fundamentos III






Manipulação de texto com a classe String



String é uma sequência ou cadeia de caracteres. Java tem uma classe embutida no pacote `java.lang` a qual encapsula a estrutura de dados de uma string. Este pacote não precisa ser importado, já que isso é realizado automaticamente.



Como as strings são objetos, existem métodos para manipulá-las que permitem, entre outras coisas, comparar o conteúdo de duas strings, pesquisar caracteres específicos de uma string, obter seu tamanho, etc.



Criando uma string vazia:

String s = new String ();

Criando uma string a partir de um array de caracteres:

Java permite construir uma string a partir de um array de caracteres: char

```
char caracteres[] = {'a','b','c'};  
String texto = new String(caracteres);  
System.out.println(texto);
```

Criando uma String a partir de um array de bytes:

Podemos criar uma string a partir de um array de bytes através do construtor String (byte [] bytes).

```
byte bytes [ ] = {65, 66, 67};  
String s = new String (bytes);
```

O compilador Java permite instanciar Strings simplesmente definindo um valor literal a ela.

```
String str = “abc”;
```


Substituição de Caracteres

Para substituir caracteres de uma string, usa-se o método replace da classe String. Esse método substitui todas as ocorrências de um determinado caractere por outro.

Por exemplo:

```
String str = "Hello";  
String resultado = str.replace ("l", "w");  
System.out.println(resultado)
```

A variável resultado receberá a string "Hewwo".

Concatenação de Strings

Em Java o sinal + (mais) age como operador de concatenação dos objetos String.

Por exemplo:

String str = “Oi”; String texto = str + “Mundo”;

Quando um dos operandos não for uma string, Java automaticamente realiza a conversão do tipo para uma representação de String. Por exemplo:

int tres = 3;

String resultadoFinal = 3 + “ palavra “ + tres;

Substring

Para extrair um texto de uma String, pode ser usado o método substring, que retorna uma nova string com o trecho selecionado nas posições indicadas. Em Java a primeira posição de uma string é zero (0). Por exemplo:

String str = “Hello World”;

**String resultado = str.substring (6);
“World”**

Substring

String resultado = str.substring (3, 8) “lo Wo”

Nesse último exemplo, a variável resultado receberá uma nova string que contém os caracteres que estão da posição 3 até a posição 8 exclusive (sem o caractere da posição 8).

Alterando os caracteres de uma String para maiúsculo

Para converter todos os caracteres de uma string para maiúsculas usa-se o método `toUpperCase ()`.

Por exemplo:

`String str = "Hello";`

`String resultado = str.toUpperCase (); "HELLO"`

Após a chamada do método, a variável resultado receberá a string "HELLO".

Alterando os caracteres de uma String para minúsculo

Para converter todos os caracteres de uma string para minúsculo usa-se o método **toLowerCase ()**.

Por exemplo:

```
String str = "Hello";
```

```
String resultado = str.toLowerCase ( ); "hello"
```

Após a chamada do método, a variável resultado receberá a string "hello".

Retirando espaços em branco

Para retirar espaços em branco ao final e início de uma string, usa-se o método **trim().**

String str = “ Hello ”;

String resultado = str.trim (); “Hello”

Após essa operação, a variável resultado receberá a string “Hello”.

Extração de Caractere

Para extrair um caractere de uma string, usa-se o método **charAt ().**

Este método retorna o caractere na posição da string especificada.


Exemplo:

String str = “Hello”;

char c = str.charAt (1); “e”

Comparação de Strings

Para comparar se duas strings são iguais, podemos usar o método `equals` da classe `String`. Existe também o método `equalsIgnoreCase` que compara duas strings ignorando maiúsculas e minúsculas. Esses dois métodos retornam como resultado um valor boolean. Veja os exemplos:



```
String s1 = “Hello”;  
String s2 = “HELLO”;  
boolean b1 = s1.equals (“Hello”);  
boolean b2 = s1.equals (s2);  
boolean b3 = s1.equalsIgnoreCase (s2);  
boolean b4=s1.equalsIgnoreCase(“azul”);
```

Tamanho de uma string

O tamanho de uma string, ou seja, o número de caracteres pode ser obtido pelo método length. Este método retorna um número inteiro.

```
String s = “abc”;
```

```
int tam = s.length( ); 3
```

O código acima retorna 3, já que a string s possui 3 caracteres.


Identificando a posição de caracteres ou substrings em uma String

Podemos buscar a posição de caracteres ou substrings em uma String através dos métodos **indexOf** e **lastIndexOf**. Estes métodos retornam o índice do caractere que está sendo procurado ou índice do início da substring buscada.



```
String str = “Hello World World2”;  
int pos = str.indexOf (“l”); 2
```

**retorna o índice da primeira
ocorrência de “l”.**



int pos = str.lastIndexOf (“|”); 15
retorna o índice da última
ocorrência de “|”.

Ordem
O método String compareTo (String)
pode ser usado para determinar a
ordem de strings: maior que, igual ou
menor que. Este método retorna um
número inteiro: <0 (negativo): se a
string é menor do que parâmetro; >0
(positivo): se a string é maior que o
parâmetro; =0 (zero): se as strings são
iguais.

compareTo e compareToIgnoreCase

Ambos fazem comparação de duas Strings, sendo que o primeiro (compareTo) considera letras maiúsculas e minúsculas na comparação e o segundo (compareToIgnoreCase) ignora qualquer diferença de minúsculas ou maiúsculas.

String valor = "CDD4.0 - Java";

System.out.println(valor.compareTo("CDD4.0 - Java") == 0 ? true : false);

System.out.println(valor.compareTo("CDD4.0 - JAVA") == 0 ? true : false);

System.out.println(valor.compareToIgnoreCase("CDD4.0 - JAVA") == 0 ? true : false);

endsWith e startsWith

O método `endsWith` verifica se a `String` termina com o valor especificado, por outro lado o `startsWith` verifica se a `String` começa com o valor especificado. Sendo que o método `startsWith` tem duas variações: uma com o parâmetro “`int toffset`” e outra sem, onde o método que contém o parâmetro “`int toffset`” serve para dizer de onde deve começar a verificação do início da `String`

endsWith e startsWith

```
String valor = "CDD - Java";
```

```
System.out.println(valor.endsWith("Java"));  
System.out.println(valor.startsWith("C"));  
System.out.println(valor.startsWith("DD", 1));
```

Contando palavras de um texto usando StringTokenizer

```
import java.util.StringTokenizer;
```

```
StringTokenizer      exemplo      =      new  
StringTokenizer("O mundo não é mais o mesmo,  
mas não iremos desistir nunca");  
    System.out.println(exemplo.countTokens());
```

StringTokenizer

A classe StringTokenizer divide uma String em partes, chamadas Tokens, que são separadas por espaços em branco, vírgula, - , : .

EXERCÍCIOS

- 1. Faça um programa que, a partir do texto abaixo, “ texto para retirar espaços no início e fim ” e imprima o texto removendo os espaços no início e fim do texto.**

EXERCÍCIOS

3. Faça um programa que, a partir de um texto digitado pelo usuário, conte o número de palavras (palavra é definida por qualquer sequência de caracteres delimitada por espaços em branco) e exiba o resultado.

EXERCÍCIOS

4. compare os 2 textos abaixo e diga se são iguais.

String texto01 = "Será que são iguais?";

String texto02 = "Será que são iguais?";

EXERCÍCIOS

5. Faça um programa que receba de um usuário, um texto e exiba esse texto em letras maiúsculas.

EXERCÍCIOS

**6. dado o Array a seguir, {"a", "vida", "é", "bela"}
faça um programa que crie um string com o
texto retirado do Array e imprima. no seguinte
formato.**

A VIDA É BELA

EXERCÍCIOS

7. Refaça o exercício anterior, usando a maneira como o Mestre yoda falaria..

BELA É VIDA A

