



CIÊNCIAS DA COMPUTAÇÃO

FACULDADE DE CIÊNCIAS | UNIVERSIDADE AGOSTINHO NETO



FUNDAMENTOS DE PROGRAMAÇÃO

Docente:

✓ Lufialuiso Sampaio Velho, MSc.

Monitor

✓ João Pedro

Conteúdo



Cap. VII

Strings

CAP. VII - String

- ❑ Em Java existe o recurso do tipo **String** para representar o conjunto de caracteres. Não como um tipo de dado primitivo mas como uma referência (objecto).
- ❑ **String** é um tipo de dado que permite armazenar um conjunto de caracteres.
- ❑ **Características**
 - Imutáveis: não se podem modificar
 - São referências
 - Contêm operações

CAP. VII. String - Declaração

❑ Por ser uma classe, para sua utilização é necessária a criação de um objecto do tipo **String**.

❑ Declaração 1: `String nomeDaString;`

❑ Declaração 2: `String nomeDaString = new String();`

❑ **Inicialização:**

Método construtor

```
String nomeDaString = new String(); //Inicializa com espaço em branco ("")
String nomeDaString = null; //Inicializa com o valor default para os objectos
String nomeDaString = new String("Ola"); //Inicializa com a String (Olá)
String nomeDaString = "Ola mundo"; //Inicializa com a String (Olá mundo)
```

CAP. VII - String

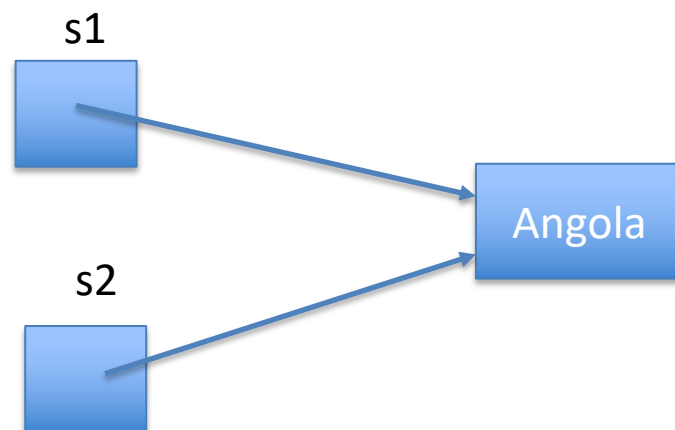
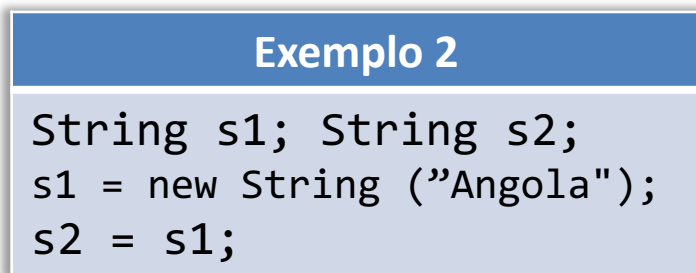
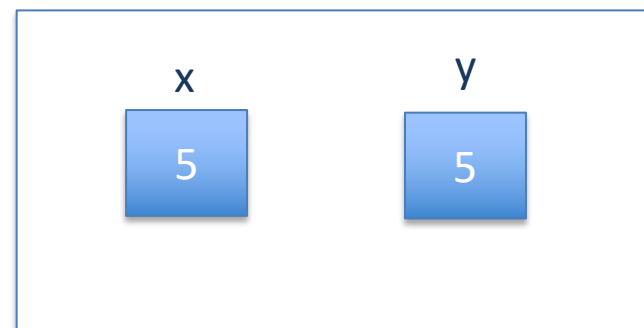
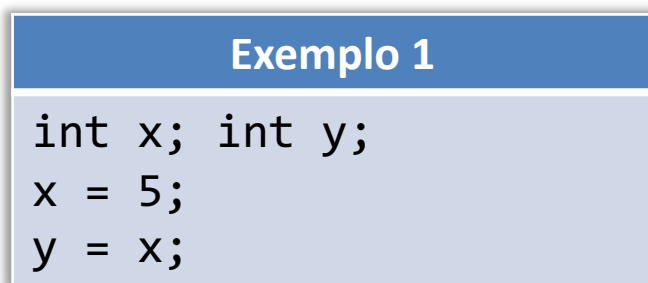
- ❑ A representação de uma String obedece a regra dos vectores de char. Ou seja cada caractere está localizado em determinada posição contando de 0 à n-1.
- ❑ EX: String s = "JAVA";

S =

J	A	V	A
0	1	2	3

CAP. VII - String

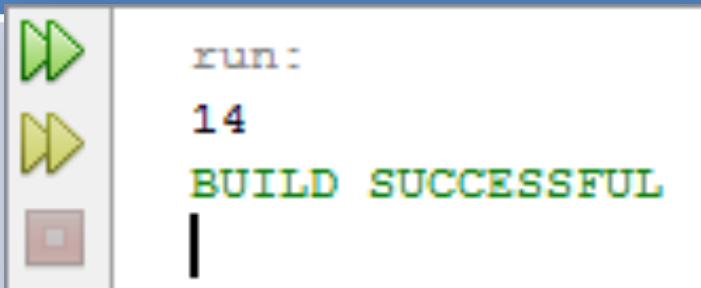
- ❑ Uma das principais características que a **String** possui é a imutabilidade. Isto é, o conteúdo de uma **String** não altera por qualquer que seja a operação em que esteja envolvida.



- **s1** e **s2** possuem a mesma referência para o objecto.

CAP. VII. String - Manipulação

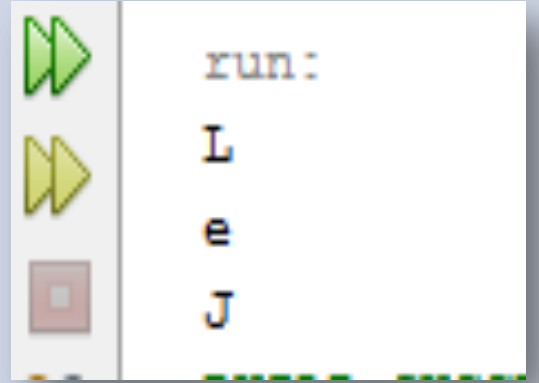
- ❑ Como toda a classe, a String possui um conjunto de métodos que facilitam a sua manipulação:
- ❑ **length()** – método que retorna o comprimento em número de caracteres de uma String.

Exemplo	Resultado
<pre>String s = "Linguagem Java"; System.out.println(s.length());</pre>	 <pre>run: 14 BUILD SUCCESSFUL </pre>

- **Nota:** `length()` \neq `length` (propriedade associada à determinação da dimensão de uma vector)

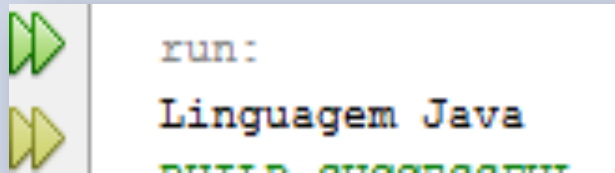
CAP. VII. String - Obter caracteres individuais

- ❑ `charAt()` - método que retorna um caractere em uma determinada posição.

Exemplo	Resultado
<pre>String s = "Linguagem Java"; System.out.println(s.charAt(0)); System.out.println(s.charAt(7)); System.out.println(s.charAt(10));</pre>	

CAP. VII. String - Concatenação

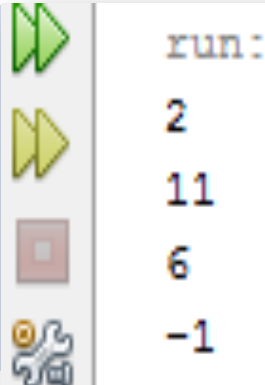
❑ **concat()** - método que retorna a concatenação (união) de duas Strings.

Exemplo	Resultado
<pre>String s1 = "Linguagem "; String s2 = "Java"; String s3 = s1.concat(s2); System.out.println(s3);</pre>	 <pre>run: Linguagem Java</pre>

Nota: Pode-se também concatenar usando o operador (+): **String s3 = s1 + s2;**

CAP. VII. String – Localização de Ocorrência

- ❑ **indexOf()** – método que retorna a posição da primeira ocorrência de um caracter ou uma sequencia de caracteres dentro de uma String. Retorna -1 se não localizar a ocorrência.

Exemplo: Inicio -Fim	Resultado
<pre>String s = "Linguagem Java"; System.out.println(s.indexOf('n')); System.out.println(s.indexOf('a', 7)); System.out.println(s.indexOf("gem")); System.out.println(s.indexOf("JAVA"));</pre>	

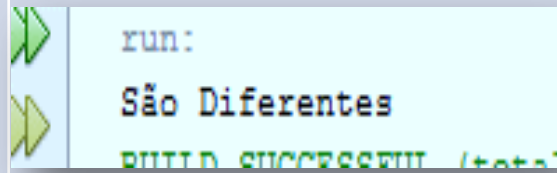
- **s.indexOf('n')** – retorna **2** (localização do caracter **n** na string **s**).
- **s.indexOf('a', 7)** – retorna **11** (localização do caracter **a** na string **s** começando pela posição 7).
- **s.indexOf("gem")** – retorna **6** (localização inicial da sequência **gem** na string **s**).
- **s.indexOf("JAVA")** – retorna -1 (sequência **JAVA** não localizada na string **s**).

- ❑ **O método lastIndexOf()** – semelhante ao anterior mas este obtém a posição da ultima ocorrência, fazendo a pesquisa do fim ao começo da string.

CAP. VII. String - Comparação

❑ **equals()** e **equalsIgnoreCase()** – compara duas Strings e retorna **true** se as duas Strings forem iguais ou **false** caso as Strings sejam diferentes.

- **equals()** - considera maiúsculas e minúsculas na comparação.
- **equalsIgnoreCase()** - ignora o facto de maiúsculas ou minúsculas.

Exemplo	Resultado
<pre>String s1 = "Linguagem Java"; String s2 = "Linguagem JAVA"; if (s1.equals(s2)) System.out.println("São Iguais"); else System.out.println("São Diferentes");</pre>	 <p>run: São Diferentes BUILD SUCCESSFUL (total</p>

- Nota: **s1 == s2** – faz comparação entre referências, Não deve ser usado para saber se duas string são iguais.

CAP. VII. String - Comparação

❑ **compareTo()** e **compareToIgnoreCase()** – são dois métodos que permitem comparar duas Strings.

- **compareTo** - considera maiúsculas e minúsculas na comparação.
- **compareToIgnoreCase** - ignora o facto de maiúsculas ou minúsculas.
- **s1.compareTo(s2)** {
 - s1 < s2 - retorna um valor negativo
 - s1 = s2 - retorna zero
 - s1 > s2 – retorna um valor positivo

Exemplo	Resultado
<pre>String s1 = "Java"; String s2 = "Fundamentos"; System.out.println("s1.compareTo(s2): " + s1.compareTo(s2)); System.out.println("s2.compareTo(s1): " + s2.compareTo(s1));</pre>	<pre>s1.compareTo(s2): 4 s2.compareTo(s1): -4</pre>

CAP. VII. String - Conversão

- ❑ **toUpperCase()**- retorna uma nova String com caracteres em maiúsculo.
- ❑ **toLowerCase()**- retorna uma nova String com caracteres em minúsculo.

Exemplo	Resultado
<pre>String s1 = "agora"; String s2 = "JAVA"; System.out.println(s1 + " - Covertiu-se em: " + s1.toUpperCase()); System.out.println(s2 + " - Covertiu-se em: " + s2.toLowerCase());</pre>	<pre>agora - Covertiu-se em: AGORA JAVA - Covertiu-se em: java</pre>

CAP. VII. String - Sub-cadeias

❑ **substring()** - retorna parte de uma String dependendo do intervalo.

Exemplo	Resultado
<pre>String s = "Linguagem Java"; System.out.println("1: "+ s.substring(0,8)); System.out.println("2: "+ s.substring(5));</pre>	<pre>1: Language 2: agem Java</pre>

- **s.substring(0,8)** – obtém do caracter 0 até ao caracter 8 não inclusive.
- **s.substring(5)** – obtém do caracter 5 até ao fim.

CAP. VII. String - Exercícios

1. Implemente um programa em Java para ler uma String e diga se a mesma é capicua. Uma String é capicua se a String original for igual a sua inversa. Ex: ana – inverso: ana; ovo – inverso ovo.
2. Implemente um programa que lê duas Strings e imprime a quantidade de vezes que cada caractere da primeira ocorre na segunda.
3. Implemente um programa que lê uma String, e imprime a String resultante das seguintes operações e a quantidade de substituições caso ocorra:
 - 'a' substitui por 't'
 - 'e' substitui por 'h'
 - 'i' substitui por 'a'

Próxima Aula – Classes

