

Linguagem de Programação

STRINGS

Prof. Me. Humberto Zanetti

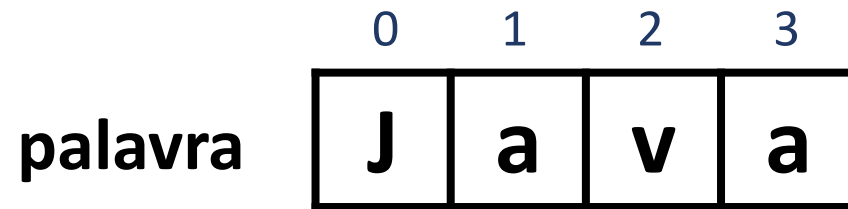


Strings

- O tipo String é um vetor especial, que possui alguns métodos de manipulação
 - String na verdade é uma classe dentro do Java
 - Não se declara como um vetor
- Como tipo é muito usado para entrada de dados, podemos trabalhar como sendo um tipo (*variável do tipo String*) ou manipulando no formato vetor.

Strings

```
String palavra;  
palavra = "Java";
```



- Note que a String possui o formato de vetor, mas na declaração não está explícito isso.
 - Não definimos o tamanho (quantidade de elemento caracteres) que a String terá, sendo alocado dinamicamente conforme o uso.

String

- Então podemos usar como se fosse um vetor?

```
String palavra;  
palavra = "Java";  
System.out.println(palavra[0]); → Erro!
```

- Mesmo tendo a formação de um vetor, uma String não pode ser manipulada como em vetor no Java.
 - Há maneiras particulares no Java para Strings.

String X vetor de caracteres

Podemos criar uma String a partir de um vetor de caracteres

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

Mas não podemos fazer uma atribuição direta...

```
letras = caracteres; → Erro!
```

...temos que fazer uma conversão!

```
letras = String.valueOf(caracteres);
```

Leitura e impressão

Utilizamos o `nextLine()` do `Scanner` para leitura e o `System.out.println()` para impressão:

```
Scanner entrada = new Scanner(System.in) ;  
String palavra;  
palavra = entrada.nextLine() ;  
System.out.println("Palavra:" + palavra) ;
```

Dica: para nossa língua portuguesa com caracteres acentuados, podemos passar o *charset* “`latin1`” ou “`ISO-8859-1`” para o `Scanner`:

```
Scanner entrada = new Scanner(System.in, "latin1") ;
```

Métodos – manipulação de String

- **length()**

- *string.length()* – retorna um valor inteiro
- determina o número de caracteres de uma String.
- Ex: tamanho = string.length();
- Notem que aqui é uma função, precisa dos **parênteses!**

```
String palavra = "Java";  
int tam;  
tam = palavra.length();  
System.out.println("Tamanho: "+ tam);
```

Saída: Tamanho: 4

Métodos – manipulação de String

- **charAt()**

- *string.charAt(int posição)* – retorna um caracter
- captura um caracter de uma String em uma posição específica.
- Ex: `character = string.charAt(2);`

```
String palavra = "Java";  
System.out.println("1.a letra:" + palavra.charAt(0));
```

Podemos usar para imprimir os caracteres separadamente:

```
for(int i = 0; i < palavra.length(); i++) {  
    System.out.println(palavra.charAt(i));  
}
```


Métodos – manipulação de String

- **equals()**

- *string1.equals(String string2)* – retorna **true** ou **false**
- verifica o conteúdo de duas Strings quanto à igualdade de conteúdo. O conteúdo de duas Strings não pode ser verificado com a utilização do operador `==`, pois o mesmo irá comparar as referências dos objetos e não seu conteúdo.
- Ex: `saolguais = string1.equals(string2);`

- **equalsIgnoreCase()**

- *string1.equalsIgnoreCase(String string2)* – retorna **true** ou **false**
- verifica o conteúdo de duas Strings quanto à igualdade de conteúdo, ignorando variações entre maiúsculas e minúsculas.
- Ex: `saolguais = string1.equalsIgnoreCase(string2);`

Métodos – manipulação de String

```
String string1 = "Java";  
String string2 = "java";
```

A expressão é **falsa**, pois "Java" e "java", são diferentes, devido a letra J

```
if (string1.equals(string2)) {  
    System.out.println("São iguais!");  
}
```

Com o equalsIgnoreCase, a diferença do J maiúsculo e minúsculo é ignorada, então a expressão é **verdadeira**.

```
if (string1.equalsIgnoreCase(string2)) {  
    System.out.println("São iguais!");  
}
```

Métodos – manipulação de String

- **compareTo()**

- *string1.compareTo(String string2)* – retorna um valor inteiro
- compara duas Strings e devolve 0 (zero) se ambas forem iguais; um número negativo se o String que chama o método for menor que o String passado como parâmetro ou um número positivo se o String que chama o método for maior que o String que for passado como parâmetro. O significado do maior e menor referência da **tabela ASCII**.
- Ex: valor = string1.compareTo(string);

- **compareToIgnoreCase()**

- *string1.compareToIgnoreCase(String string2)* – retorna um valor inteiro
- compara duas Strings ignorando variações entre maiúsculas e minúsculas e devolve 0 (zero) se ambas forem iguais; um número negativo se o String que chama o método for menor que o String passado como parâmetro ou um número positivo se o String que chama o método for maior que o String que for passado como parâmetro.
- Ex: valor = string1.compareToIgnoreCase(string2);

Métodos – manipulação de String

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(72	48	110	H	104	68	150	h
9	9	11		41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Métodos – manipulação de String

```
String string1 = "Java";  
String string2 = "java";  
String string3 = "java";  
String string4 = "javc";  
String string5 = "javac";
```

`string1.compareTo(string2) -> retorna -32`

Diferença de 32 valores na tabela ASCII

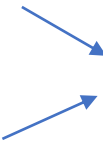


`string2.compareTo(string3) -> retorna 0`

`string2.compareTo(string5) -> retorna -1`

`string5.compareTo(string2) -> retorna 1`

Quando são tamanhos diferentes, é retornado se há mais ou menos caracteres



`string2.compareTo(string4) -> retorna -2`

`string1.compareToIgnoreCase(string2) -> retorna 0`

Ignora a diferença entre maiúscula e minúscula



Métodos – manipulação de String

- **substring()**
- *string.substring(int inicioIndex)* – retorna uma string
 - retorna uma novo String, copiando um trecho indicado de um String específico. Neste caso, da posição (inicioIndex) até o fim da String.
 - Ex: novaString = string.substring(2);
- *string.substring(int inicioIndex, int fimIndex)* – retorna string
 - retorna um novo String, copiando um trecho indicado de um String específico.
 - Neste caso, da posição (inicioIndex) até a posição (fimIndex – 1) ou seja, inicioIndex inclusive e fimIndex exclusive.
 - Ex: novaString = string.substring(2, 9);

Métodos – manipulação de String

```
String frase = "Java é muito fácil!";  
String recorte = frase.substring(4);  
System.out.println(recorte);
```

Saída: é muito fácil!

```
String frase = "Java é muito fácil!";  
String recorte = frase.substring(4,12);  
System.out.println(recorte);
```

Saída: é muito

Métodos – manipulação de String

- **replace()**

- *String = string.replace(char velhoChar, char novoChar);*
- substitui cada ocorrência de um caracter em um String por outro. Substitui todas as ocorrências de velhoChar por novoChar.
- Ex: novaString = string.replace('o', 'O');

- **toUpperCase()**

- *String = string.toUpperCase();*
- gera um novo String com todas as letras em maiúsculas.
- Ex: novaString = string.toUpperCase();

- **toLowerCase()**

- *String = string.toLowerCase();*
- gera um novo String com todas as letras em minúsculas.
- Ex: novaString = string.toLowerCase();

Métodos – manipulação de String

```
String palavra = "Java";  
System.out.println(palavra.replace('a', 'A'));
```

Saída: JAvA

```
System.out.println(palavra.toUpperCase());
```

Saída: JAVA

```
System.out.println(palavra.toLowerCase());
```

Saída: java

Métodos – manipulação de String

- **indexOf()**

- *istring.indexOf(char character)* – retorna um inteiro
- retorna a posição da primeira ocorrência de um character num String; Retorna um valor inteiro com a posição do character ou -1 caso o character não exista no String.
- Ex: `posicao = string.indexOf('a')`

- *istring.indexOf(char character, int deOnde)* – retorna um inteiro

- retorna a posição da primeira ocorrência de um character num String a partir da posição especificada no segundo argumento (deOnde); Retorna um valor inteiro com a posição do character ou -1 caso o character não exista no String.
- Ex: `posicao = string.indexOf('a', 5);`

- *string1.indexOf(String string2)* – retorna um inteiro

- retorna a posição do primeiro character da primeira ocorrência de um String numa outra String; Também pode se passar um segundo argumento de onde iniciar a pesquisa.
- Ex: `posicao = string.indexOf("bc");`

Métodos – manipulação de String

```
String frase = "Programando em Java";  
char letra = 'a';  
int posicao;  
posicao = frase.indexOf('a');  
System.out.println("Posição da letra: "+posicao);
```

Saída: Posição da letra: 5

```
posicao = frase.indexOf('a', 10);  
System.out.println("Posição da letra: "+posicao);
```

Saída: Posição da letra: 16

```
posicao = frase.indexOf("Java");  
System.out.println("Posição da letra: "+posicao);
```

Saída: Posição da letra: 15

Métodos – manipulação de String

- **startsWith()**

- *string.startsWith(String prefixo)* – retorna um true ou false
- verifica se uma String é iniciada com uma sequência determinada de caracteres, retornando verdadeiro em caso afirmativo.
- Ex: `booleano = string.startsWith("bc");`

- **endsWith()**

- *string.endsWith(String sufixo)* – retorna um true ou false
- verifica se uma String é encerrada com uma sequência determinada de caracteres, retornando verdadeiro em caso afirmativo.
- Ex: `booleano = string.endsWith("bc");`

```
String frase = "Programando em Java";  
System.out.println(frase.startsWith("Programando")) ;  
System.out.println(frase.endsWith("Java")) ;  
System.out.println(frase.startsWith("em", 12)) ;
```

Podemos começar por qualquer índice

Saída: true
true
true

Outros métodos

- Há ainda muitos outros métodos de manipulação:
 - trim()
 - format()
 - toCharArray()
 - getBytes()
 - isEmpty()
 - split ()
 - etc...
- Ver em:

<https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>

Dúvidas!?

humberto.zanetti@fatec.sp.gov.br