02

Criando e Manipulando Strings

Existem duas maneiras tradicionais de criar uma `String` :

```
String nomeDireto = "Java";
String nomeIndireto = new String("Java");
```

COPIAR CÓDIGO

A comparação entre esses dois tipos de criação de `Strings` é feita na seção ::Test equality between strings and other objects using == and equals():: testequality

Existem outras maneiras não tão comuns:

```
char[] nome = new char[]{'J', 'a', 'v', 'a'};
String nomeComArray = new String(nome);

StringBuilder sb1 = new StringBuilder("Java");
String nome1 = new String(sb1);

StringBuffer sb2 = new StringBuffer("Java");
String nome2 = new String(sb2);
```

COPIAR CÓDIGO

Como uma `String` não é um tipo primitivo, ela pode ter valor `null` , lembre-se disso:

```
String nome = null; // null explicito
```

COPIAR CÓDIGO

Podemos concatenar `Strings` com o `+`:

```
String nome = "Certificação" + " " + "Java";
```

COPIAR CÓDIGO

Caso tente concatenar `null` com uma `String`, temos a conversão de `null` para `String`:

```
String nula = null;
System.out.println("nula: " + nula); // imprime nula: null
```

COPIAR CÓDIGO

O Java faz a conversão de tipos primitivos para `Strings` automaticamente, mas lembre-se da precedência de operadores:

```
String nome = "Certificação" + ' ' + "Java" + ' ' + 1500;
System.out.println(nome);

String nome2 = "Certificação";
nome2 += ' ' + "Java" + ' ' + 1500;
System.out.println(nome2);

String valor = 15 + 00 + " certificação";
System.out.println(valor); // imprime "15 certificação",
// primeiro efetuando uma soma
```

COPIAR CÓDIGO

Strings são imutáveis

O principal ponto sobre Strings é que elas são imutáveis:

```
String s = "caelum";
s.toUpperCase();
System.out.println(s);
```

COPIAR CÓDIGO

Esse código imprime `caelum` em minúscula. Isso porque o método `toUpperCase` não altera a `String` original. Na verdade, se olharmos o javadoc da classe `String` vamos perceber que **todos os métodos que parecem modificar uma String na verdade devolvem uma nova**.

```
String s = "caelum";
String s2 = s.toUpperCase();
System.out.println(s2);
```

COPIAR CÓDIGO

Agora sim imprimirá `CAELUM` , uma nova String. Ou, usando a mesma ::referência:::

```
String s = "caelum";
s = s.toUpperCase();
System.out.println(s);
```

COPIAR CÓDIGO

Para tratarmos de "strings mutáveis", usamos as classes `StringBuffer` e `StringBuilder`.

Lembre-se que a `String` possui um array por trás e, seguindo o padrão do Java, suas posições começam em 0:

```
// 0=g, devolve 'g'
char caracter0 = "guilherme".charAt(0);

// 0=g 1=u, devolve 'u'
char caracter1 = "guilherme".charAt(1);

// 0=g 1=u 2=i, devolve 'i'
char caracter2 = "guilherme".charAt(2);
```

COPIAR CÓDIGO

Cuidado ao acessar uma posição indevida, você pode levar um `StringIndexOutOfBoundsException` (atenção ao nome da `Exception`, não é `ArrayIndexOutofBoundsException`):

```
char caracter20 = "guilherme".charAt(20); // exception
char caracterMenosUm = "guilherme".charAt(-1); // exception
```

COPIAR CÓDIGO

Principais métodos de String

O método `length` imprime o tamanho da `String`:

```
String s = "Java";
System.out.println(s.length()); // 4
System.out.println(s.length); // não compila: não é atributo
System.out.println(s.size()); // não compila: não existe size
                              // em String Java
```

COPIAR CÓDIGO

Já o método `isEmpty` diz se a `String` tem tamanho zero:

```
System.out.println("".isEmpty()); // true
System.out.println("java".isEmpty()); // false
System.out.println(" ".isEmpty()); // false
```

COPIAR CÓDIGO

Devolvem uma nova String:

- `String toUpperCase()` - tudo em maiúscula;
- `String toLowerCase()` - tudo em minúsculo;
- `String trim()` - retira espaços em branco no começo e no fim;
- `String substring(int beginIndex, int endIndex)` - devolve a substring a partir dos índices de começo e fim;
- `String substring(int beginIndex)` - semelhante ao anterior, mas toma a substring a partir do índice passado até o final da String;
- `String concat(String)` - concatena o parâmetro ao fim da String atual e devolve o resultado;
- `String replace(char oldChar, char newChar)` - substitui todas as ocorrências de determinado `char` por outro;

- `String replace(CharSequence target, CharSequence replacement)` - substitui todas as ocorrências de determinada `CharSequence` (como String) por outra.

O método `trim` limpa caracteres em branco nas duas pontas da `String`:

```
System.out.println("     ".trim()); // imprime só a quebra de lir
System.out.println(" ".trim().isEmpty()); // true
System.out.println(" guilherme "); // imprime guilherme
System.out.println(". ."); // imprime '. .'
```

COPIAR CÓDIGO

O método `replace` substituirá todas as ocorrências de um texto por outro:

```
System.out.println("java".replace("j", "J")); // Java
System.out.println("guilherme".replace("e", "i")); // guilhirmi
```

COPIAR CÓDIGO

Podemos sempre fazer o ::chaining:: e criar uma sequência de "transformações" que retornam uma nova `String`:

```
String parseado = "  Quero tirar um certificado oficial de  Java

// imprime: "QUERO TIRAR UM CERTIFICADO OFICIAL DE JAVA!"
System.out.println(parseado);
```

COPIAR CÓDIGO

Para extrair pedaços de uma `String`, usamos o método `substring`. Cuidado ao usar o método `substring` com valores inválidos, pois eles jogam uma `Exception`. O segredo do método `susbtring` é que ele não inclui o caractere da posição final, mas inclui o caractere da posição inicial:

```
String texto = "Java";

// ava
System.out.println(texto.substring(1));

// StringIndexOutOfBoundsException
System.out.println(texto.substring(-1));

// StringIndexOutOfBoundsException
System.out.println(texto.substring(5));

// Java
System.out.println(texto.substring(0, 4));

// ava
System.out.println(texto.substring(1, 4));

// Jav
System.out.println(texto.substring(0, 3));

// StringIndexOutOfBoundsException
System.out.println(texto.substring(0, 5));

// StringIndexOutOfBoundsException
System.out.println(texto.substring(-1, 4));
```

COPIAR CÓDIGO

Comparação:

- `boolean equals(Object)` - compara igualdade caractere a caractere (herdado de `Object`);
- `boolean equalsIgnoreCase(String)` - compara caractere a caractere ignorando maiúsculas/minúsculas;
- `int compareTo(String)` - compara as 2 Strings por ordem lexicográfica (vem de `Comparable`);
- `int compareToIgnoreCase(String)` - compara as 2 Strings por ordem lexicográfica ignorando maiúsculas/minúsculas.

E aqui, todas as variações desses métodos. Não precisa saber o número exato que o `compareTo` retorna, basta saber que será negativo caso a `String` na qual o método for invocado vier antes, zero se for igual, positivo se vier depois do parâmetro passado:

```
String texto = "Certificado";
System.out.println(texto.equals("Certificado")); // true
System.out.println(texto.equals("certificado")); // false
System.out.println(texto.equalsIgnoreCase("certificado"));//true

System.out.println(texto.compareTo("Arnaldo")); // 2
System.out.println(texto.compareTo("Certificado")); // 0
System.out.println(texto.compareTo("Grécia")); // -4

System.out.println(texto.compareTo("certificado")); // -32

System.out.println(texto.compareToIgnoreCase("certificado"));//0
```

COPIAR CÓDIGO

Buscas simples:

- `boolean contains(CharSequence)` - devolve `true` se a String contém a sequência de `chars` ;
- `boolean startsWith(String)` - devolve `true` se começa com a String do parâmetro;
- `boolean endsWith(String)` - devolve `true` se termina com a String do parâmetro;
- `int indexOf(char)` e `int indexOf(String)` - devolve o índice da primeira ocorrência do parâmetro;
- `int lastIndexOf(char)` e `int lastIndexOf(String)` - devolve o índice da última ocorrência do parâmetro.

O código a seguir exemplifica todos os casos desses métodos:

```
String texto = "Pretendo fazer a prova de certificação de Java";

System.out.println(texto.indexOf("Pretendo")); // imprime 0
System.out.println(texto.indexOf("Pretendia")); // imprime -1
System.out.println(texto.indexOf("tendo")); // imprime 3

System.out.println(texto.indexOf("a")); // imprime 10
System.out.println(texto.lastIndexOf("a")); // imprime 45
System.out.println(texto.lastIndexOf("Pretendia")); //imprime -1

System.out.println(texto.startsWith("Pretendo")); // true
System.out.println(texto.startsWith("Pretendia")); // false

System.out.println(texto.endsWith("Java")); // true
System.out.println(texto.endsWith("Oracle")); // false
```

COPIAR CÓDIGO