



High performance. Delivered.

Application Delivery Fundamentals: Java

Module 5-6: String

[illegible]

“CLASSE STRING”

- O modo mais direto de criar uma string é:

Variáveis do tipo String guardam referências a objetos, e não um valor, como acontece com os tipos primitivos;

- A classe String possui 15 métodos construtores, que nos permitem fornecer o valor inicial da string usando diferentes fontes, tais como arrays de caracteres: .

```
String str = new String("ACCENTURE");
```

```
String str = "ACCENTURE";
```

- ```
String str = "Técnicas de Programação II";

int tamanho = str.length();

System.out.println("Tamanho: " + tamanho);
```

```
String str = "Técnicas de Programação II";

System.out.println("Primeira Letra: " + str.charAt(0));

System.out.println("Ultima Letra: " + str.charAt(str.length()-1));
```

```
Primeira Letra: T
Ultima Letra: I
```

```
String name;
name = "Wally";
name = "Pat";
```

Diagram illustrating a hash table using separate chaining:

- Slot 1 (Key: 132456) points to a linked list containing: W, a, l, l, y.
- Slot 2 (Key: 146234) points to a linked list containing: P, a, t.







[illegible]

- Lista completa:  
<https://docs.oracle.com/javase/7/docs/api/java/lang/String.html>

## public class `String`

---

|                                                 |                                                                                               |
|-------------------------------------------------|-----------------------------------------------------------------------------------------------|
| <code>String(String s)</code>                   | <i>create a string with the same value as <code>s</code></i>                                  |
| <code>String(char[] a)</code>                   | <i>create a string that represents the same sequence of characters as in <code>a[]</code></i> |
| <code>int length()</code>                       | <i>number of characters</i>                                                                   |
| <code>char charAt(int i)</code>                 | <i>the character at index <code>i</code></i>                                                  |
| <code>String substring(int i, int j)</code>     | <i>characters at indices <code>i</code> through <code>(j-1)</code></i>                        |
| <code>boolean contains(String substring)</code> | <i>does this string contain <code>substring</code>?</i>                                       |
| <code>boolean startsWith(String prefix)</code>  | <i>does this string start with <code>prefix</code>?</i>                                       |
| <code>boolean endsWith(String postfix)</code>   | <i>does this string end with <code>postfix</code>?</i>                                        |
| <code>int indexOf(String pattern)</code>        | <i>index of first occurrence of <code>pattern</code></i>                                      |
| <code>int indexOf(String pattern, int i)</code> | <i>index of first occurrence of <code>pattern</code> after <code>i</code></i>                 |
| <code>String concat(String t)</code>            | <i>this string, with <code>t</code> appended</i>                                              |
| <code>int compareTo(String t)</code>            | <i>string comparison</i>                                                                      |
| <code>String toLowerCase()</code>               | <i>this string, with lowercase letters</i>                                                    |
| <code>String toUpperCase()</code>               | <i>this string, with uppercase letters</i>                                                    |
| <code>String replace(String a, String b)</code> | <i>this string, with <code>as</code> replaced by <code>bs</code></i>                          |
| <code>String trim()</code>                      | <i>this string, with leading and trailing whitespace removed</i>                              |
| <code>boolean matches(String regexp)</code>     | <i>is this string matched by the regular expression?</i>                                      |
| <code>String[] split(String delimiter)</code>   | <i>strings between occurrences of <code>delimiter</code></i>                                  |
| <code>boolean equals(Object t)</code>           | <i>is this string's value the same as <code>t</code>'s?</i>                                   |
| <code>int hashCode()</code>                     | <i>an integer hash code</i>                                                                   |

2f48696768207065726

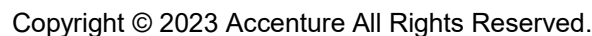
65642c2f48696768207

[illegible]

# Comparação

- Operador == compara tipos primitivos
- O método equals compara objetos

```
String s = new String("abc");
String s2 = new String("abc");
```





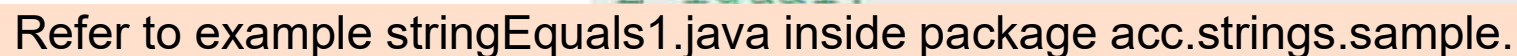
Não é igual!

Retorna **true** se, e somente se, a string representa a mesma sequência de caracteres que o objeto passado como argumento.

```
String s = new String("abc");
String s2 = new String("abc");

if (s1.equals(s2))
 System.out.println("É igual!");
else
 System.out.println("Não é igual!");
```

É igual!



- **int compareTo(String s)**– compara duas strings lexicograficamente. Retorna um inteiro que indica se a string maior (retorno  $> 0$ ), igual (retorno  $= 0$ ) ou menor (retorno  $< 0$ ) que a string passada como argumento.



Refer to example `StringCompareTo1.java`, `StringEndsWith` inside package `acc.strings.sample`.

Uma vez criado um objeto da classe String, ele não pode ser modificado.

- ```
String s = "abc";  
String s2 = s;  
s = s.concat("def");  
  
System.out.println(s);  
System.out.println(s2);
```

```
abcdef
abc
```

Classe String - Imutabilidade

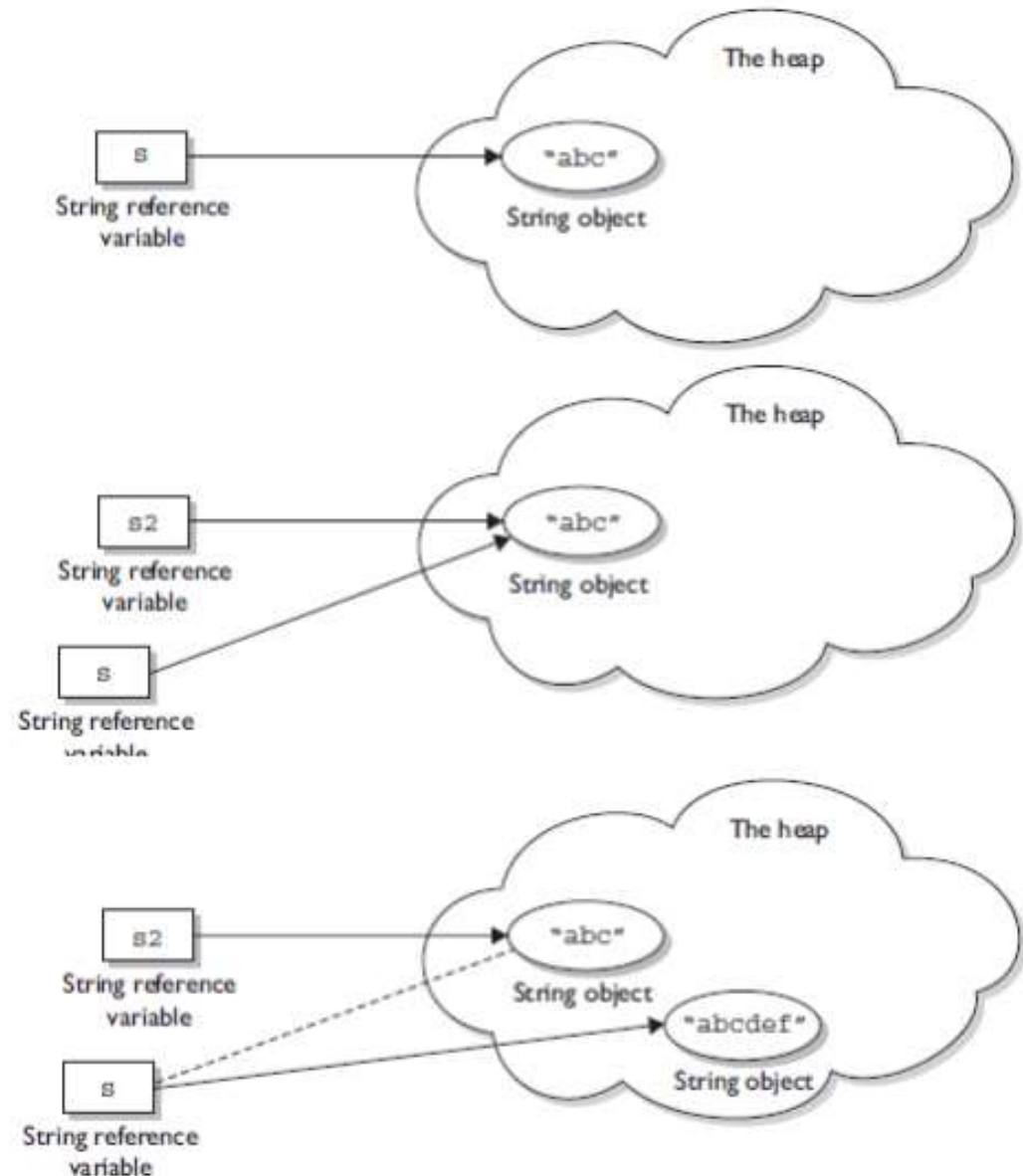
2f48696768207065726

65642e2f48696768207

```
String s = "abc";
```

```
String s2 = s;
```

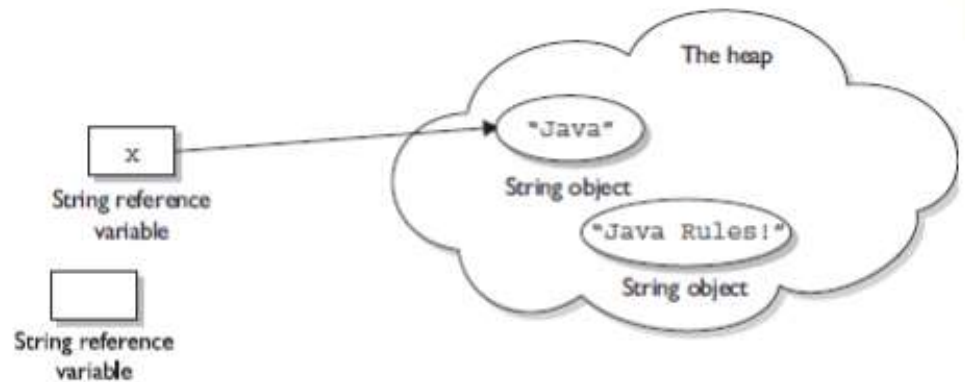
```
s = s.concat("def");
```



Refer to example `stringConcat1.java` inside package `acc.strings.sample`.

[illegible]

Java




Copyright © 2023 Accenture All Rights Reserved.

Classe String – StringBuffer e StringBuilder

- A classe **String** é **imutável**. Sendo assim, quando algo é concatenado a uma String, esta não é alterada. O que ocorre é a criação de uma nova String.
- Por isso trabalhar com uma mesma String diversas vezes pode ter um efeito colateral: gerar inúmeras Strings temporárias.
- Isto prejudica a performance da aplicação consideravelmente.

- Copyright © 2023 Accenture All Rights Reserved.

A classe **StringBuffer** é *sincronizada* para acesso concorrente, enquanto que a **StringBuilder** não tem *sincronização*.

-  Refer to example StringBuffer1.java StringBuilder.java inside package acc.strings.sample.


Classe String – StringBuffer e StringBuilder

Use **String** para manipular com valores constantes:

- Valores literais;
- Textos carregados de fora da aplicação;
- Textos em geral que não serão modificados intensivamente.

- Use **StringBuffer** para alterar textos: – Acrescentar, concatenar, inserir, etc.
- Prefira usar **StringBuffer** para construir Strings
 - Concatenação de strings usando "+" é extremamente cara.

- **String nextLine()**– Retorna a ultima linha de texto digitada no console;

 Refer to example `StringScanner.java` inside package `acc.strings.sample`.

É possível analisar os caracteres de uma Strings usando o método **charAt**. Exemplo:

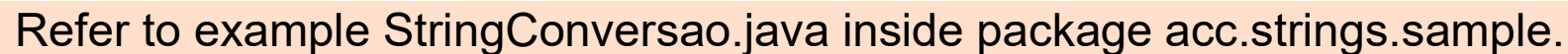
```
public static int conta_letra(String str, char letra)
{
    int i, total = 0;
    for (i = 0; i < str.length(); i++)
    {
        if (str.charAt(i) == letra)
            total++;
    }
    return total;
}
```

```
String str1 = "Tecnicas de Programacao II";  
System.out.println("Total: " + contaLetra(str1, 'a'));
```



Refer to example `StringContaLetras.java` inside package `acc.strings.sample`.

- ```
String str1 = "85.1";
double num1 = Double.parseDouble(str1);
```



Copyright © 2023 Accenture All Rights Reserved.

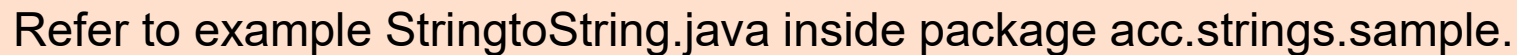
1. **Introduction**



# Classe String

## toString

- O método `toString()` faz parte de todos os objetos em java por ser um método da classe `Object`
- Sobrescrevendo o comportamento deste método podemos definir uma representação de um objeto como uma `String`





[illegible]

- 

