



Strings

Richarlyson A. D'Emery

site: <https://sites.google.com/site/profricodemery/mpoo>






grupo: http://groups.google.com/group/mpoo_uast

email grupo: mpoo_uast@googlegroups.com

contato: rico_demery@yahoo.com.br

Sumário



-  Criação
-  Comparação
-  Concatenação
-  Métodos da classe String
-  StringBuffer

Criação de Strings

- Atribuir um literal a uma variável
 - `String cor = "verde";`
- Usando o construtor e o operador new
 - `String cargo = new String ("Gerente Geral");`
- Strings são concatenadas com o operador +
 - `System.out.println("Região" + "Metropolitana");`
 - `System.out.println("Cor: " + cor);`
 - `System.out.println("Valor " + valorA);`

Comparação de Strings

- Não use o operador `==` para comparar Strings.
 - Não se deve usar com comparações de referências: pode levar a erro de lógica
 - Compara as referências para determinar se elas se referem ao mesmo objeto, mas não se os objetos têm o mesmo conteúdo
 - Ex.:

```
String s1 = "Sexta";  
String s2 = "sexta";  
String s3 = new String ("Sexta");
```

```
System.out.println(s1=="Sexta");//mesmo conteúdo  
System.out.println(s1==s3);
```

Comparação de Strings

■ Métodos

- `equals()`
- `equalsIgnoreCase()`
 - se não for importante a caixa alta (maiúsculas ou minúsculas):

- Ex.:

```
String nome = "Lisa";  
if (!nome.equals("lisa"))  
    System.out.println("Não é Lisa");  
if (nome.equalsIgnoreCase("lisa"))  
    System.out.println("É Lisa");
```

- `compareTo()`, retorna:
 - **0** se as Strings forem iguais
 - **Negativo** se a String que invoca `compareTo()` for menor que o parâmetro
 - **Positivo** se a String que invoca `compareTo()` for maior que o parâmetro

Concatenação de Strings

- Use o operador `+` para concatenar Strings:
 - `String nome = "Lisa" + "Simpson";`
 - `String str2 += str1;`
- Sempre que pelo menos um dos operandos do operador `+` for uma string a operação será uma concatenação de Strings:
 - `String tipo = "Fusca" + 1600;`
- Use o método `toString` para obter uma String a partir de qualquer objeto Java

Principais métodos de String

- `charAt(index)`
 - retorna o caracter na posição index
- `trim()`
 - retira espaços em branco nas extremidades de uma String
- `subStr(inicio,fim)`
 - retorna uma sub-string começando na posição inicio e terminando na posição anterior a fim
- `concat(string)`
 - retorna uma string concatenada
- `length()`
 - retorna a quantidade de caracteres da String

Principais métodos de String

- `replace("v", "V")`
 - Substitui v por V
- `toUpperCase()`
 - Converte letras minúsculas em maiúsculas
- `toLowerCase ()`
 - Converte letras maiúsculas em minúsculas
- `toString ()`
 - Converte um objeto em String (informação sobre a classe do objeto)
 - É usado para garantir que o retorno de uma informação será String

Principais métodos de String

■ hashCode()

- faz uso da tabela *hash*
 - Informa qual o código (*hash*) utilizado pela String
 - Não confundir com a classe HashTable
- Exemplo

```
public class UsandoHash {  
    public static void main(String[] args) {  
        String s1="conteúdo";  
        String s2="Conteúdo";  
        String saída="";  
  
        saída += "código hash de s1: " + s1.hashCode() + "\n"+  
                "código hash de s2: " + s2.hashCode() + "\n";  
  
        System.out.println(saída);  
    }  
}
```

A classe StringBuffer

- Representa uma cadeia de caracteres que pode ser alterada
 - É uma *string* dinamicamente modificável e redimensionável.
- Não é possível atribuir um objeto do tipo **StringBuffer** a uma variável do tipo String
- Concatenação de strings são efetuadas através da classe **StringBuffer**

A classe StringBuffer

■ Exemplo:

```
public class UsandoStringBuffer {  
    public static void main(String[] args) {  
        StringBuffer buffer;  
        String s;  
  
        buffer = new StringBuffer("Exemplo de StringBuffer");  
        //Erro: s=buffer;  
        s = new String(buffer);  
  
        System.out.println(buffer);  
        System.out.println(s);  
    }  
}
```

Exercício 1



- Escreva um programa que execute o seguinte trecho de código:

```
String str1 = "Sexta";  
String str2 = "sexta";  
String str3 = new String("Sexta");  
System.out.println(str1.equals(str2));  
System.out.println(str1.equals(str3));
```

Exercício 2

- Escreva um método para criar um comando SQL

- Parâmetros:

nome das colunas selecionadas ou * para todas

nome da tabela

condições da cláusula WHERE

Ex.:

```
criaSelect("*", "ALUNOS", "alunos", "Robson");
```

- Mostrar a string resultante

```
select * from ALUNOS where alunos="Robson"
```

Exercício 3

- Escreva um método que insira uma string em outra, indicando a posição.

- Ex.:

```
String str1 = "Teste";  
String str2 = "Uva";  
insere(str1, str2, 2);
```

Resultaria na string "TeUvaste"



FIM

Prof. Richarlyson D'Emery

site: <https://sites.google.com/site/profricodemery/mpoo>

grupo: http://groups.google.com/group/mpoo_uast

email grupo: mpoo_uast@googlegroups.com

contato: rico_demery@yahoo.com.br