

Linguagem de Programação Orientada a Objetos I

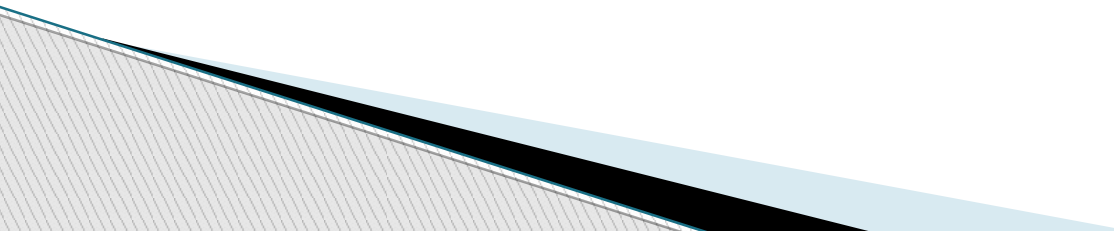
Classe String e Wrapper Classes

Prof. Tales Bitelo Viegas

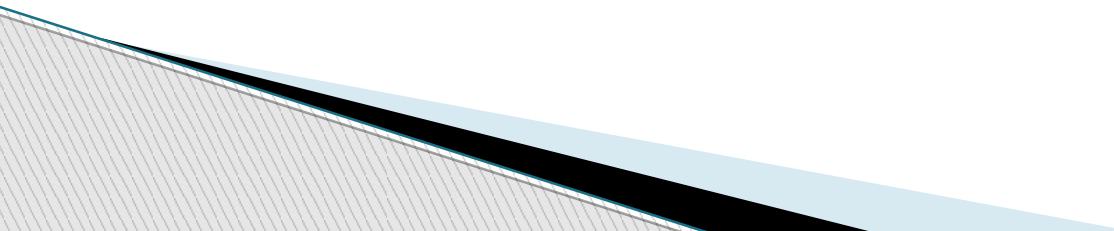
<https://fb.com/ProfessorTalesViegas>

The bottom of the slide features a decorative graphic consisting of several overlapping, wavy, horizontal bands. The colors transition from a light blue at the top, through a dark grey, to a light grey at the bottom. The waves create a sense of movement and depth.

Introdução

- ▶ Um objeto da classe String representa uma string de caracteres
 - ▶ Como as outras classes, String possui construtores e métodos
 - ▶ Diferente da maioria das outras classes, String possui 2 operadores, + e += (usados para concatenação)
- 

String Literais

- ▶ São objetos anônimos da classe String
 - ▶ São definidos por texto envolto em aspas duplas: "Isto é uma String literal"
 - ▶ Não necessitam ser instanciadas (new)
 - ▶ Podem ser atribuídas para variáveis String
 - ▶ Podem ser passadas como parâmetros
 - ▶ Possuem métodos que podem ser chamados
- 

String Literais - Exemplos

```
// Define um literal para uma variável String  
String nome = "Tales";
```

```
// Executando um método  
char primeiro = "Tales".charAt(0);
```

```
// Executando um método na variável  
char primeiro = nome.charAt(0);
```



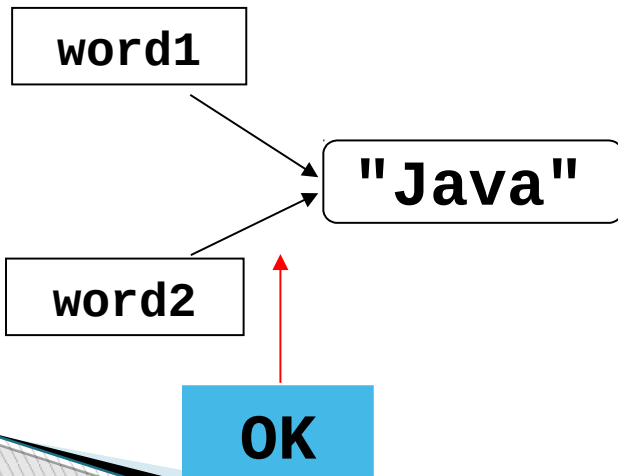
Imutabilidade

- ▶ Uma vez criada, uma String não pode ter seu valor alterado
- ▶ Nenhum dos métodos altera o valor

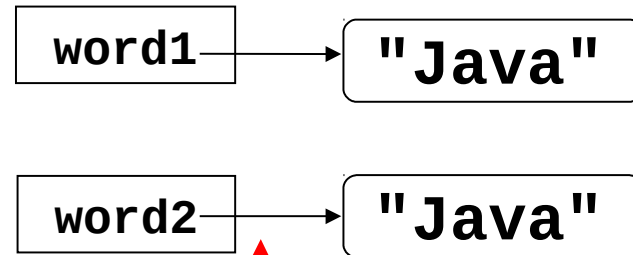
Vantagem da Imutabilidade

- ▶ Usa menos memória

```
String word1 = "Java";  
String word2 = word1;
```



```
String word1 = "Java";  
String word2 = new String(word1);
```

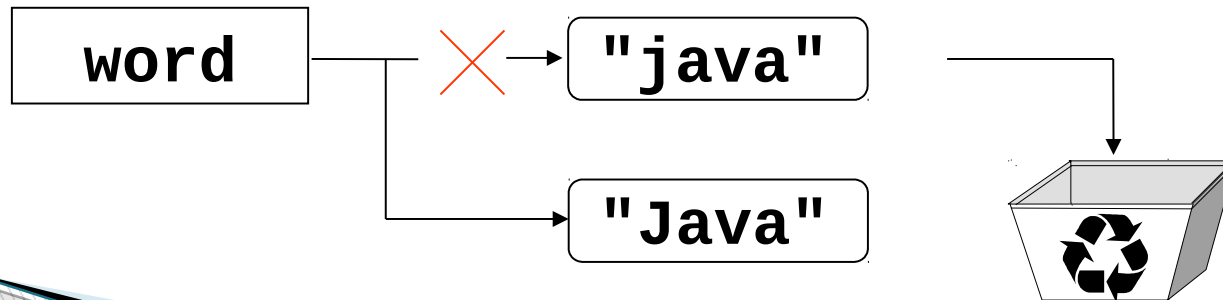


**Menos eficiente:
desperdício de memória**

Desvantagem da Imutabilidade

- ▶ Menos eficiente – Necessário criar outro objeto para pequenas mudanças na String

```
String word = "java";  
char ch = Character.toUpperCase(word.charAt (0));  
word = ch + word.substring (1);
```



String vazia

- ▶ Uma String vazia não possui caracteres. Seu tamanho é zero

```
String word1 = "";  
String word2 = new String();
```

**Strings
vazias**



- ▶ Não é o mesmo que uma String não-inicializada

```
private String errorMsg;
```

**errorMsg é
null**



Construtor sem argumentos

- ▶ Construtor sem argumentos cria uma String vazia

```
String empty = new String();
```

- ▶ Geralmente utilizamos através de literal

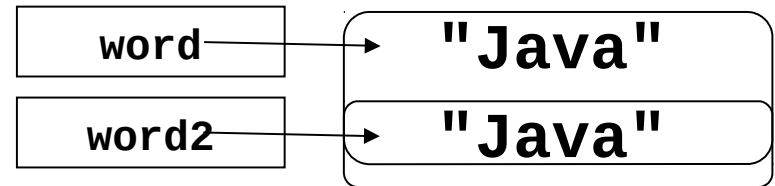
```
String empty = "";           //nada entre as aspas
```

Contrutores para cópia

- ▶ Cria uma cópia da String existente.
- ▶ Não é o mesmo que uma atribuição

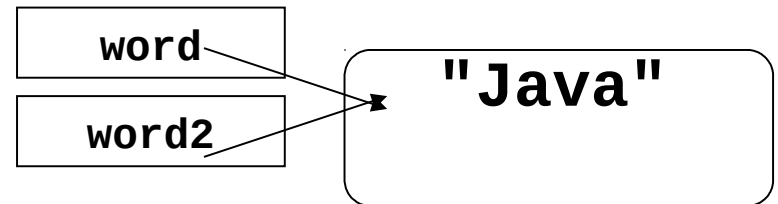
Construtor: Cada variável aponta para um objeto diferente.

```
String word = new String("Java");  
String word2 = new String(word);
```



Atribuição: As duas variáveis apontam para o mesmo objeto

```
String word = "Java";  
String word2 = word;
```



Outro Construtor

- ▶ Possibilidade de transformar um array de char em uma String

```
char[] palavra = {'J', 'a', 'v', 'a'};  
String word = new String(palavra); //"Java"
```

- ▶ O inverso pode ser obtido pelo método toCharArray()

```
String word = "Java";  
char[] palavra = word.toCharArray();  
    // {'J', 'a', 'v', 'a'};
```

Métodos - length, charAt

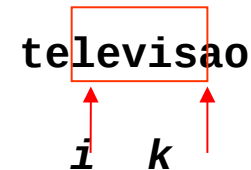
- ▶ `int length()` - Retorna o tamanho da String
- ▶ `char charAt(i)` - Retorna o caracter da posição `i` na String (iniciam em 0)

Retorna:

| | | |
|-----------------------------------|---|------------------|
| <code>"Trabalho".length();</code> | → | <code>8</code> |
| <code>"Ulbra".charAt(2);</code> | → | <code>'b'</code> |

Métodos - substring

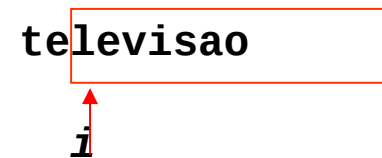
- ▶ Retorna uma nova String copiando caracteres de uma String existente
- `String subs = word.substring (i, k);`
 - Retorna uma substring iniciando no caracter **i** e terminando no caracter **k**
- `String subs = word.substring (i);`
 - Retorna uma substring a partir do caracter **i** até o final da string



televisao

i *k*

The diagram shows the string "televisao" with a red box highlighting the substring "levisao". Two red arrows point to the start and end of the box, labeled with the variables *i* and *k* respectively.



televisao

i

The diagram shows the string "televisao" with a red box highlighting the entire string. A red arrow points to the start of the box, labeled with the variable *i*.

```
"televisao".substring (2,7);  
"imutavel".substring (2);  
"bob".substring (9);
```

Returns:

→ "levis"

→ "utavel"

→ Erro!

Métodos - concatenação

```
String word1 = "re", word2 = "think"; word3 = "ing";  
int num = 2;
```

- `String result = word1 + word2;`
 //concatena word1 e word2 "rethink"
- `String result = word1.concat(word2);`
 // o mesmo que word1 + word2 "rethink"
- `result += word3;`
 // concatena word3 à result "rethinking"
- `result += num;`
 //converte num para String e concatena à result
 "rethinking2"

Métodos - pesquisa

0 2 6 10 15



String name = "President George Washington";

Retorna:

| | |
|--------------------------|----|
| name.indexOf ('P'); | 0 |
| name.indexOf ('e'); | 2 |
| name.indexOf ("George"); | 10 |
| name.indexOf ('e', 3); | 6 |
| name.indexOf ("Bob"); | -1 |
| name.lastIndexOf ('e'); | 15 |

**(começa a partir
da posição 3)**

(não contém)

Métodos - Igualdade

`boolean b = word1.equals(word2);`
retorna **true** se a string **word1** é igual a **word2**

`boolean b = word1.equalsIgnoreCase(word2);`
retorna **true** se a string **word1** é igual a **word2**, ignorando maiúsculas ou minúsculas

```
b = "Gremio".equals("Gremio");//true  
b = "Gremio".equals("gremio");//false  
b = "Gremio".equalsIgnoreCase("gremio");//true
```

```
if(team.equalsIgnoreCase("gremio"))  
    System.out.println("Vai " + team);
```


Métodos - Trim

String word2 = **word1.trim** ();

retorna uma nova String formada por
word1 removendo espaços em branco no
início ou no final da String

Não remove espaços do meio

```
String word1 = " Oi Bob ";  
String word2 = word1.trim();  
//word2 é "Oi Bob" - sem espaços no início ou  
// no final  
//word1 ainda é " Oi Bob " - com espaços
```

Métodos - Replace

String word2 = word1.**replace**(oldCh, newCh);
retorna uma nova string formada de
word1 substituindo todas as ocorrências
de **oldCh** por **newCh**

```
String word1 = "raro";  
String word2 = "raro".replace('r',  
'd');  
//word2 é "dado", mas word1 ainda é  
"raro"
```

Métodos - Mudando "case"

```
String word2 = word1.toUpperCase();  
String word3 = word1.toLowerCase();
```

retorna uma nova string a partir de **word1** convertendo todos os caracteres para maiúsculas (Upper) ou minúsculas

```
String word1 = "HeLlO";  
String word2 = word1.toUpperCase();//"HELLO"  
String word3 = word1.toLowerCase();//"hello"  
//word1 ainda é "HeLlO"
```

Números em Strings

► 3 maneiras para converter:

- `String s = "" + num`

```
s = "" + 123; //"123"
```

- `String s = Integer.toString (i)`

- `String s = Double.toString (d)`

```
s = Integer.toString(123); //"123"
```

```
s = Double.toString(3.14); //"3.14"
```

- `String s = String.valueOf(num)`

```
s = String.valueOf(123); //"123"
```

Wrappers Classes

► Wrappers

- Objetos que tem relação direta com dados primitivos
- Objetos permitem a passagem por referência
- Conjunto de métodos para manipulação
- Para cada dado primitivo existe um empacotador correspondente
- Consultar documentação da API (pertencem ao pacote `java.lang`)

Wrappers Classes

- ▶ Correspondência:
 - dados primitivos x wrappers

| Dado Primitivo | Wrapper Class |
|----------------|------------------|
| boolean | Boolean |
| char | Character |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |

Wrappers Classes - Exemplo

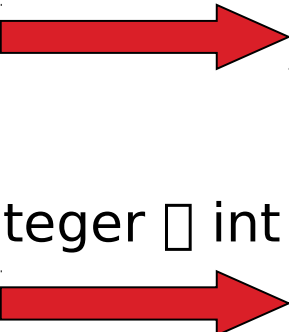
▶ Exemplo: Classe Integer (parte 1)

```
public class WrapperTest {
```

```
    public static void main(String[] args) {
```

```
int i1 = 1; // criando um objeto Integer a partir de um int
Integer objInt1 = new Integer(i1);
// ou simplesmente:
Integer objInt2 = 1;

Integer i2 = objInt1.intValue();
// ou simplesmente:
Integer i3 = objInt2;
```



Wrappers Classes - Exemplo

▶ Exemplo: Classe Integer (parte 2)

```
Integer num1 = new Integer(10);
Integer num2 = new Integer(10);
// testando valores
if (num1.equals(num2))
    System.out.println("Valores sao iguais !");
else
    System.out.println("Valores sao diferentes !");
// testando referencias
if(num1==num2)
    System.out.println("As referencias sao iguais !");
else
    System.out.println("As referencias sao diferentes !");
}
```


Funções Úteis

- ▶ Integer.parseInt(String)
 - ▶ Integer.toOctalString(int)
 - ▶ Float.parseFloat(String)
 - ▶ Float.toHexString(Float)
 - ▶ Boolean.getBoolean(String)
 - ▶ Double.parseDouble(String)
- 