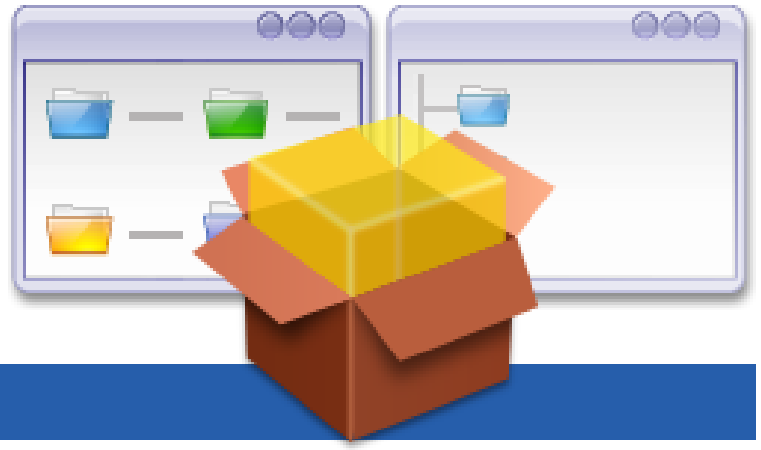
Programação Orientada a Objetos

String

Prof. Tito Kenzo



String

- É uma seqüência de caracteres Unicode.
- Seus literais são representados em aspas duplas.
- Literais são convertidas automaticamente em instâncias pelo compilador.

- Exemplo:

```
String cor = " verde";
```

ou

```
String cor = new String ("cor")
```

- Está definida no pacote java.lang.String

Construção

- A Classe String fornece construtores para criação de Strings.
  - Cria uma String vazia com o valor “”.

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

  - Cria uma cópia do objeto referenciado por str.

```
String str2 = new String (str);
```

  - Cria uma String a partir de um array de caracteres.

```
char[] arr;
...
String str3 = new String (arr);
```

Concatenação

- Utiliza-se o operador sobrecarregado + para concatenar Strings.

```
String nome = "Joao";
String sobrenome = "Silva";
String nomeCompleto = nome + sobrenome;
```

- Quando um dos operando do operador + é uma String, o outro operando é automaticamente convertido para string.

```
String valorStr = "Valor Atual = ";
double valor = 34.45;
System.out.println(valorStr + valor);
```

Alteração

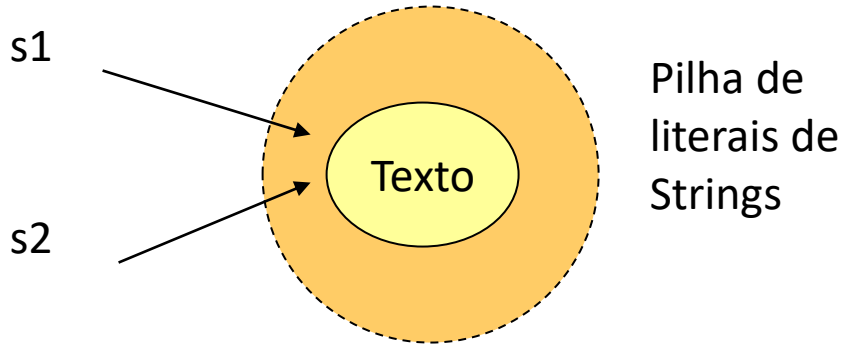
- Uma instância da classe String é “imutável”. Ou seja, não pode ser alterada depois de criada.

  - Cada literal internamente é uma instância de String.
  - Classes em Java podem possuir uma pilha para estas Strings.
  - Quando uma literal é compilada, o compilador adiciona uma String apropriada à pilha.
  - No entanto, se a mesma literal aparecer na classe, ela já está sendo representada.
  - O compilador não cria uma nova cópia, ao invés tenta utilizar uma existente.

Alteração

- Exemplo:

```
String s1 =“Texto”;
String s2 =“Texto”;
```



s2 = “Texto Completo”; O que vai acontecer?

Comparação

- Não se utiliza o operador == para comparar Strings.
  - Teste referências quando utilizado com Objetos.

- Utiliza-se o método equals() da classe String.

- Utiliza-se o método equalsIgnoreCase() da classe String se não for importante o caso (maiúscula ou minúscula);

```
String str1 = new String (“Java”);
String str2 = new String (“Java”);

str1 == str2 // retorna falso
str1.equals(str2) retorna verdadeiro
```

Outros Métodos

- **int length()**

  Retorna o número de caracteres da String corrente.

- **String substring(int startIndex)**

  Retorna a substring, começando do startIndex até o final da String corrente.

- **String compareTo(String otherString)**

  Realiza uma comparação léxica; retorna um int menor que zero se a string corrente é menor que otherString, igual a 0 se as String forem idênticas e maior que zero quando a String corrente for maior que otherString.

Outros Métodos

- Descrever o que os métodos abaixo fazem:
  - charAt
  - endsWith
  - replace
  - toLowerCase
  - toString
  - trim

Classe StringBuffer

- Uma instância em Java da classe StringBuffer representa uma String que pode ser modificada dinamicamente.

- Possui construtores similares aos da classe String.

- Possui métodos similares aos da classe String.

Tarefinha

- 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));
  - System.out.println(str1==str2);
  - System.out.println(str1==str3);

- Interprete as saídas

Tarefinha

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

  Exemplo:

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

- Resultaria na String “TeUvaste”