Introdução à Programação

Um enfoque orientado a construção de modelos em programas baseados em objetos

Gustavo Motta
Departamento de Informática - UFPB

8. Manipulação de Strings

- A classe `String`
  - Usada para representação e manipulação seqüências de caracteres
    - Inclui métodos úteis para úteis para processamento de textos
      - Tamanho do string, i. e., a quantidade de caracteres armazenados
      - Caractere existente numa dada posição do string
        - `"Java".lenght()` ➔ `4`
        - `"Java".charAt(0)` ➔ `'J'`
        - `"Java".charAt(2)` ➔ `'v'`
        - `"Java".charAt(4)` ➔ `java.lang.StringIndexOutOfBoundException`
        - `"Java".toCharArray()` ➔ `{'J','a','v','a'}`
    - Construção de instâncias da classe `String`
      - `String curso = "Ciência da Computação";`
      - `String curso = new String("Ciência da Computação");`
      - `char[] caracteres = {'C','i','ê','n','c','i','a',' ', 'd','a',' ','C','o','m','p','u','t','a','ç','ã','o'};`
      - `String curso = new String(caracteres);`

8. Manipulação de Strings

- A classe `String`
  - Exemplo – Jogo da Forca
    - 1. Inicializa-se um string com palavra a ser adivinhada e cria-se um array de booleanos, cada posição correspondendo a um caractere da palavra, inicializada com `false`, para indicar que nenhum caractere foi adivinhado. Inicializa-se um array de 26 posições booleanas para indicar as letras que já foram utilizadas
    - 2. Mostra-se para o usuário quais letras já foram adivinhadas e quais letras já foram utilizadas
    - 3. Pede-se para o usuário entrar com uma letra, que é incluída no rol das letras usadas e marca-se a letra nas posições correspondentes da palavra a ser adivinhada, se pertinente
    - Caso todas as letras da palavra tenha sido adivinhada, então o

8. Manipulação de Strings

- A classe `String`
  - Métodos para comparação de string
    - Strings não devem ser comparados com == – por que?
      - `String curso = "Computação";`
      - `"Computação".equals(curso)` → **true**
      - `"computação".equals(curso)` → **false**
      - `curso.equals("Computação")` → **true**
      - `curso.equals("Comuptação")` → **false**
      - `curso.equals(curso)` → **true**
      - `"Computação".equalsIgnoreCase(curso)` → **true**
      - `"cOmPutaÇãO".equalsIgnoreCase(curso)` → **true**
      - `curso.equalsIgnoreCase("cOmPutaÇãO")` → **true**
      - `curso.equalsIgnoreCase("cOmPutaÇã")` → **false**
      - `curso.equalsIgnoreCase(curso)` → **true**

8. Manipulação de Strings

- A classe `String`
  - Métodos para comparação de strings
    - Considerando apenas o início ou o fim do string
      - `String curso = "Computação";`
      - `curso.startsWith("Comp")` → **`true`**
      - `curso.startsWith("comp")` → **`false`**
      - `curso.startsWith("Computação")` → **`true`**
      - `curso.startsWith("Computaçãoo")` → **`false`**
      - `curso.startsWith("")` → **`true`**
      - `curso.endsWith("ação")` → **`true`**
      - `curso.endsWith("Ação")` → **`false`**
      - `curso.endsWith("Computação")` → **`true`**
      - `curso.endsWith("Computaçãoo")` → **`false`**
      - `curso.endsWith("")` → **`true`**

8. Manipulação de Strings

- A classe `String`
  - Métodos para procura de substrings
    - Verificar se um string contém outro string
      - `String curso = "Computação";`
      - `curso.indexOf("ação")` → **6**
      - `curso.indexOf("o")` → **1**
      - `curso.indexOf("uta")` → **4**
      - `curso.indexOf("cação")` → **-1**
      - `curso.indexOf("")` → **0**
      - `curso.indexOf("Comp")` → **0**

8. Manipulação de Strings

- A classe `String`
  - Métodos para transformação de strings
    - String em Java são imutáveis

Os métodos de de processamento de string podem ser combinados

`curso.toUpperCase().trim().substring(12)` → **"OMPUTAÇÃO"**

- `curso.substring(12)` → **"Computação "**
- `curso.substring(12, 16)` → **"Comp"**
- `curso.substring(16, 12)` → **java.lang.StringIndexOutOfBoundsException**
- `curso.substring(16, curso.length())` → java.lang.StringIndexOutOfBoundsException

8. Manipulação de Strings

- A classe `String`
  - Métodos para conversão de tipos com strings
    - Converte valores de tipos nativos para strings e vice-versa
      - Tipos nativos para String
        - `String.valueOf(10)` ➔ "10"
        - `String.valueOf(15.4)` ➔ "15.4"
        - `String.valueOf('v')` ➔ "v"
    - `char[] carac = {'J', 'a', 'v', 'a'};`
      `String.valueOf(carac)` ➔ "Java";
      - String para tipos Nativos
        - `Interger.parseInt("10")` ➔ 10
        - `Float.parseFloat("3.2383")` ➔ 3.2383
        - `Integer.parseInt("10.33")` ➔ `java.lang.NumberFormatException`
        - `Float.parseFloat("3.2a383")` ➔ `java.lang.NumberFormatException`

8. Manipulação de Strings

- A classe `StringBuffer`
  - Permite criação e manipulação de strings modificáveis
    - Lembre que instância da classe `String` são imutáveis
  - A criação de instâncias de `StringBuffer` é mais custosa que a criação de instâncias da classe `String`, porém a manipulação é bem mais eficiente
  - Instâncias da classe `StringBuffer` têm duas variáveis de instância do tipo inteiro associadas
    - Uma para indicar o *comprimento* do string, isto é, a quantidade de caracteres armazenados num dados momento
    - Outra para indicar a capacidade, que corresponde ao número máximo de caracteres que podem ser armazenados pela instância num dado momento
      - A capacidade pode ser expandida automática ou manualmente

8. Manipulação de Strings

- A classe `StringBuffer`
  - `StringBuffer str = new StringBuffer(50);`
  - `StringBuffer str = new StringBuffer("Java");`
  - `str.append(' '); str.append("Language ");`
  - `str.append(10);`
  - `System.out.println(str);` → Java Language 10
  - `str.insert(14, "é ");`
  - `System.out.println(str);` → Java Language é 10
  - `str.delete(5, 13);`
  - `System.out.println(str);` → Java é 10
  - `str.reverse();`
  - `System.out.println(str);` → 01 é avaJ