Classe String

Instituto Federal do Espirito Santo – Ifes
Campus Colatina – Programação Orientada a Objetos I
Professor: Giovany Frossard Teixeira

Classe String

- A classe String é responsável pela manipulação de conjuntos de caracteres. A sintaxe para manipulação de strings é a seguinte:

< Nome da string>.<nome-do-método>(<argumentos>)

- A classe String não precisa ser instanciada utilizando o comando "new". Para instanciar uma variável do tipo String basta iniciá-la com um valor.

Método length

- Retorna o tamanho (tipo int) de uma string. Sintaxe:

<string>.length()

Ex:

```
package string;
public class Length {
    public static void main(String[] args) {
        String frase = "Vou estudar esse modulo";
        int tamanho = frase.length();
        System.out.println("O tamanho da string: \"" + frase + "\" eh " + tamanho);
    }
}
```

Obs. Notem o caracter de escape \ para que possamos colocar o " na mensagem a ser impressa.

Método charAt

- Retorna o caracter (char) da string que se localiza no índice passado como parâmetro. Sintaxe:

< string >.charAt(<indice>)

Ex:

```
package string;

public class CharAt {

    public static void main(String[] args) {

        int num = 2;

        char n = "AACDD".charAt(num);

        System.out.println("O caracter na posicao " +
      num + " eh " + n);

    }

}
```

- Obs. As strings em Java, assim como em C, começam na posição 0, logo o caracter que será impresso é C.

Métodos toUpperCase e toLowerCase

- toUppperCase: Retorna uma String com todas as letras maiúsculas a partir da String que chamou o método. Sintaxe:

  < string >.toUpperCase()

- toLowerCase: Retorna uma String com todas as letras minúsculas a partir da String que chamou o método. Sintaxe:

  < string >.toLowerCase()

Ex:

```
package string;

public class UpperLowerCase {

    public static void main(String[] args) {

        String nome = "Giovany";

        System.out.println(nome);

        System.out.println(nome.toUpperCase());

        System.out.println(nome);

        System.out.println(nome.toLowerCase());

        System.out.println(nome);

    }

}
```

Método substring

- Retorna uma substring, a partir da String que chamou o método, definida pelos índices informados como parâmetros. Sintaxe:

<string>.substring(<indice_incial>, [<indice_final>])

Obs. O índice final está entre [] por ser opcional.

Ex:

```
package string;

public class SubString {

    public static void main(String[] args) {

        String frase = "O rato roeu";

        System.out.println(frase.substring(2, 6));

        System.out.println(frase.substring(6));


    }

}
```

Método trim

- Retorna uma String sem espaços em branco no início e no final dela, a partir da String que chamou o método.
  Sintaxe:

  < string >.trim()

Obs. Os espaços em branco entre palavras se mantem.

Ex:

```
package string;

public class Trim {

    public static void main(String[] args) {

        String nome = "  Estados  Unidos  ";

        System.out.println(nome);

        System.out.println(nome.trim());

        nome = nome.trim();

        System.out.println(nome);

        System.out.println(nome.trim());

    }

}
```

Método replace

- Retorna uma String com substrings trocadas, a partir da String que chamou o método. As trocas são feitas de acordo com os parâmetros do método, onde aparecer a substring1 estará a substring2. Sintaxte:

<string>.replace(<substring1>,
<substring2>)

Ex:

```
package string;

public class Replace {

    public static void main(String[] args) {

        // retirando espaços em branco
        dentro de dentro da string

        String frase1 = "Mariana gosta de
         nana banana".replace(" ", "");

        frase1 = frase1.replace("na", "NA");

        System.out.println(frase1);

    }

}
```

Método valueOf

- Retorna uma String a partir de um número ou de uma cadeira de caracteres. Sintaxe:

String.valueOf(<valor a ser convertido>)

Ex:

```
package string;

public class ValueOf {

    public static void main(String[] args) {

        String x = "";

        x = x + String.valueOf(23) +
        String.valueOf(true) + " - ";

        x = x + String.valueOf(Math.PI);

        System.out.println(x);

    }

}
```

- Obs. Notar que o operador + promove a concatenação de strings, comportamento semelhante ao encontrado no Visualg com o tipo caracter.

Método indexOf

- Retorna o índice de uma substring ou caracter a ser buscado, se não encontrar retorna -1. Sintaxe:

  <string>.indexOf(< caracter ou substring >, [posição inicial da busca])

Ex:

```
package string;

public class IndexOf {

    public static void main(String[] args) {

        String texto = "Linguagem de
       Programacao";

        char caracter = 'a';

        int indice = texto.indexOf(caracter);

        System.out.println(indice);

        indice++;

        indice = texto.indexOf(caracter, indice);

        System.out.println(indice);

        indice = texto.indexOf("acao", indice);

        System.out.println(indice);

    }
}
```

Dúvidas?