Pesquisa Classe String

Alunos: Junio César de Oliveira Filho e Létícia Palazzo Borges Severino
Turma: 2º ano informática

O que é String e qual o seu objetivo?

Uma String armazena uma sequência de caracteres. O objeto de String é imutável, o que significa que o texto que ele carrega nunca é alterado. Sempre que um texto precisa ser modificado é utilizado mais espaço em memória para que uma nova String seja criada contendo a nova versão dele.

Como criar uma string?

Para o compilador, qualquer texto entre aspas duplas é uma String. Por esse motivo a criação de um objeto desse tipo não requer a utilização do operador new. Assim, uma String é criada de forma semelhante a um tipo primitivo, utilizando-se a sintaxe [tipo] [nome] = [valor], apesar de se tratar de um tipo por referência - um nome para um objeto em memória.

O **Código 1** apresenta a forma correta de se criar uma instância da classe String.

```
String texto = "Qualquer texto entre aspas é uma String";
```

Código 1. Criação de uma instância da classe String

Também não utilizamos o operador new porque ao fazê-lo forçamos a criação de uma nova String, anulando um recurso de otimização da linguagem que evita que o mesmo texto exista mais de uma vez na memória.

Por exemplo, no **Código 2** criamos uma String utilizando o operador new, o que faz com que mais recursos sejam utilizados do que o necessário.

```
String texto = new String("Qualquer texto entre aspas é uma String");
```

Código 2. Declaração de uma string utilizando o operador new

Toda a vez que o código acima for executado uma nova String será criada contendo o texto "Qualquer texto entre aspas é uma String". Caso o **Código 2** esteja dentro de um loop for repetido por mil vezes, mil objetos serão criados. Contudo, se o **Código 1** estiver dentro de um loop for repetido por mil vezes, apenas um objeto será criado na primeira repetição e reutilizado em todas as demais. Lembre-se, para o Java qualquer texto entre aspas duplas é uma instância da classe String. A prova disso é que podemos escrever um texto entre aspas e invocar a partir dele um método qualquer da classe String, como demonstra o **Código 3**.

```
"Qualquer texto entre aspas é uma String".length();
```

Código 3. Invocando um método qualquer da classe String

Note acima que o texto não foi atribuído a nenhuma variável, mas ainda assim é um objeto plenamente funcional para o compilador, a partir do qual podemos invocar quaisquer métodos da classe String.

Principais métodos da classe String:

charAt

Retorna o caractere em uma localização específica em uma String. Esse método possui um parâmetro do tipo inteiro que é usado como índice, retornando a partir dessa posição inserida nesse parâmetro. É importante lembrar que o índice sempre começa a ser contado do número 0 (zero) em diante. Sendo assim a posição do caractere a em Carlos é 1 e não 2, como se poderia deduzir.

Nesse novo exemplo, no **código a seguir**, a mensagem "O caractere A está na posição 1" será impressa, uma vez que o caractere A está na posição 1 da cadeia de caracteres.

```
String nomeCurso = "JAVA"; if(nomeCurso.charAt(1) == 'A') {
System.out.println("O caractere A está na posição 1"); }
```

Código. Exemplo do método charAt

compareTo

Esse método pode retornar 0 se as strings forem iguais, um número negativo se a string que invoca o compareTo for menor que a string que é passada como um argumento e um número positivo se a string que invoca o compareTo for maior que a string que é passada como argumento.

```
String nome1 = "Carlos"; String nome2 = "Carla";
System.out.println("nome2.compareTo(nome1) =
"+nome2.compareTo(nome1));
System.out.println("nome1.compareTo(nome2) =
"+nome1.compareTo(nome2));
```

Código. Exemplo do método compareTo

Neste caso, compareTo vai nos dar um número negativo no primeiro caso, porque Carla é menor que Carlos, e um número positivo no segundo caso porque Carlos é maior que Carla.

concat

Existem duas formas de unir duas ou mais sequências de caracteres. A mais comum dentre elas é utilizando o operador de adição, como demonstra o **código**.

```
String nomeCompleto = nome + sobrenome;
```

Código 13. Exemplo concatenação de Strings

Uma outra forma de fazer isso é utilizando o método concat. Isso é menos comum, mas ainda é possível. O **código a seguir** ficaria dessa forma utilizando esse método:

```
String nomeCompleto = nome.concat(sobrenome);
```

Código 14. Exemplo de concatenação de Strings

Note que se o valor de nome for Carlos e sobrenome for Henrique, o resultado do código acima será CarlosHenrique, porque não há espaço entre os dois textos.

contains

Pesquisa a sequência de caracteres na string fornecida. Ele retorna verdadeiro se a sequência de valores char forem encontrados nesta String, caso contrário, retorna falso.

```
public boolean contains(CharSequence sequence)
{
   return indexOf(sequence.toString()) > -1;
}
```

Aqui, a conversão de CharSequence em String ocorre e o método indexOf é chamado. O método indexOf retorna O ou um número maior se encontrar a String, caso contrário, -1 é retornado. Portanto, após a execução, o método contains() retorna verdadeiro se a sequência do valor char existe, caso contrário, é falso .

Sintaxe:

```
public boolean contains(CharSequence sequence)
```

```
Parameter : sequence : This is the sequence of

characters to be searched.

Exception :

NullPointerException : If seq is null
```

contentEquals

Método é usado para esta cadeia para o StringBuffer especificado para comparar. Valor de retorno: tais como uma String com o StringBuffer especificado representam a mesma sequência de caracteres, ele retorna true, caso contrário false. Observe o exemplo a seguir:

```
public class Test {

     public static void main(String args[]) {

          String str1 = "String1";

          String str2 = "String2";

          StringBuffer str3 = new StringBuffer( "String1");



          boolean  result = str1.contentEquals( str3 );

          System.out.println(result);

          result = str2.contentEquals( str3 );

          System.out.println(result);

     }
```

```
}
```

startsWith e endsWith

Os métodos startsWith e endsWith aceitam uma string e um número inteiro como argumentos, retornando um valor booleano que indica se a string inicia ou termina, respectivamente, com o texto informado a partir da posição dada.

```
String[] nomes = {"iniciado", "iniciando", "finalizado", "finalizando"};

for (String recebeNomes : nomes) { if (recebeNomes.startsWith("in")) System.out.printf("\"%s\" inicia com \"in\" \n", recebeNomes); }

System.out.println();

for (String recebeNomes : nomes) { if (recebeNomes.startsWith("ici", 2)) System.out.printf("\"%s\" inicia com \"ici\" na posição 2 \n", recebeNomes); }

System.out.println();

for (String recebeNomes : nomes) { if (recebeNomes.endsWith("ado")) System.out.printf("\"%s\" termina com \"ado\" \n", recebeNomes); }
```

Código. Exemplo dos métodos startsWith e endsWith

Length

Retorna o comprimento do texto em uma String. No **código a seguir** é impresso o comprimento do texto.

```
String nomeCurso = "Java";

System.out.printf("\nTamanho da variável nomeCurso: %d", nomeCurso.length());
```

Código. Exemplo do método length

split

O Split é usado para dividir uma string em pequenos pedaços. Para isso, seu método retorna um array de strings, contendo as respectivas partes, definidas de acordo com a string passada como parâmetro, que na verdade funciona como o "agente" divisor da mesma.

De uma forma resumida pense na string "micro-ondas". Supondo que desejamos dividi-la pelo hífen, teríamos que fazer a implementação ilustrado na **Listagem 1**, feita em um Console Application.

Listagem 1.Método Split

```
string texto = "micro-ondas";
string[] retornoSplit = texto.Split('-');
Console.WriteLine(String.Format("Posição 0: {0} \nPosição 1: {1}",
retornoSplit[0], retornoSplit[1])); Console.ReadKey();
```

Note que o método Split espera um char como parâmetro, por isso são usadas as aspas simples. O resultado do código acima é ilustrado pela **figura a seguir.**

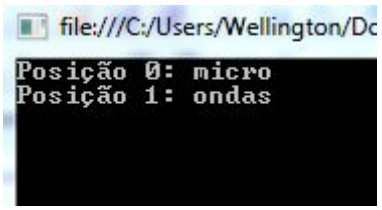

Figura 1. String dividida em dois

Isso é simples, se soubermos em quantos elementos serão divididos nossa string. Mais e se não soubermos? Para isso podemos utilizar um laço for. Suponhamos que nossa string seja "micro-ondas-micro-ondas-micro-ondas". A **Listagem 2** ilustra como podemos fazer para descobrir o número de elementos do array de strings e interar os valores.

Listagem 2. Iterando os valores de um Array de Strings

```
string texto = "micro-ondas-micro-ondas-micro-ondas";

  string[] retornoSplit = texto.Split('-');

  int numeroElementos = retornoSplit.Length;

  for (int i = 0; i < numeroElementos; i++)

{

        Console.WriteLine(String.Format("Posição " + i.ToString() +
": {0}\n", retornoSplit[i]));

}

Console.ReadKey();
```

O uso do Split em seu código, em certas situações, é uma "mão na roda". O resultado do código acima é visto na **Figura 2**.

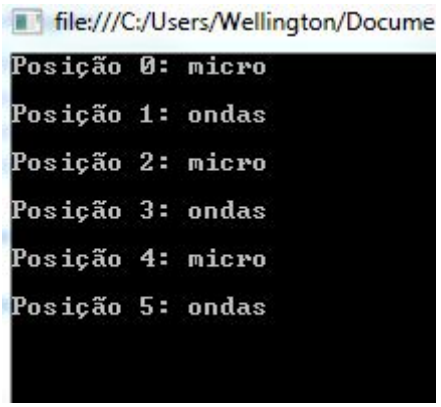

Figura 2. String dividida em seis

toUpperCase

Retorna uma nova string com o conteúdo da original convertido para letras maiúsculas, mantendo a original inalterada.

toLowerCase

De forma semelhante ao anterior, o toLowerCase retorna uma cópia de uma string com todas as letras convertidas para minúsculo, mantendo a original inalterada.

```
String nomeA = "joaquina";
String nomeB = "Paulo";

System.out.println(nomeA.toUpperCase());
System.out.println(nomeB.toLowerCase());
```

Código. Exemplo dos métodos toLowerCase e toUpperCase

Exemplo de Uso do Split:

```
/*
        * Uso do split sem o LIMIT
         * */
        String valor = "DEVMEDIA - Java";
        String[] valorComSplit = valor.split("-");

        for(String s : valorComSplit){
              System.out.println(s);
        }
        Saída:DEVMEDIA
        Java
```

Exemplo de uso do toLowerCase, toUpperCase:

O método toLowerCase converte toda a String para caixa baixa e o toUpperCase faz o inverso, convertendo toda a String para caixa alta. O método trim remove espaços em branco no inicial e no final da String, conforme exemplo da **listagem a seguir.**

Listagem. Uso do toLowerCase, toUpperCase

```
String valor = "DEVMEDIA - Java";

        System.out.println(valor.toLowerCase());
        System.out.println(valor.toUpperCase());
        System.out.println(valor.trim());
```

```
Saída:

devmedia - java

DEVMEDIA - JAVA
```