LCP - Aula 04

Prof. Lucas Guerreiro

Aula anterior

- Arrays
- Coleções
- Exemplos de uso de vetores em geral

Aula 04

Objetivo: Strings. Manipulação de Strings. Expressões Regulares.

Caracteres

- Char: tipo primitivo para armazenamento de um único caracter.
- Exemplo:

```
char c = 'c';

char[] lcp = {'l', 'c', 'p'};
```

Manipulação de Strings

- String em Java: um objeto que representa uma cadeia de caracteres.
- Duas formas de declaração:

```
String s1 = "lcp";

String s2 = new String("lcp");
```

- Strings podem ser alocadas no pool de constantes (`s1`) ou na heap da JVM (`s2`).
- Strings são imutáveis.

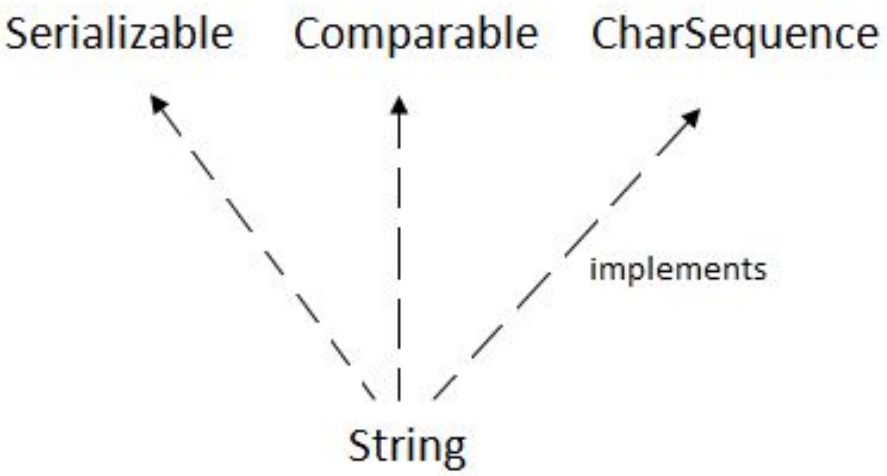
Strings

```
String s1 = "lcp";

String s2 = "lcp";
```

- Ao alocar `s2`, a JVM identifica que já existe o valor `"lcp"` no pool de constantes e aponta para este valor.
- Ao alterar um valor (`s1 = "LCP"`), uma nova String é criada no pool e este valor será referenciado pela variável.
- Podemos ainda criar Strings a partir de cadeias de caracteres:

```
char[] c = {'l', 'c', 'p'};

String s = new String(c);
```

Manipulação de Strings



Manipulação de Strings

```
String s1 = "olá a todos";
char[] charArray = new char[5];
System.out.printf("Tamanho de %s sé %d", s1, s1.length());
System.out.printf("%n%s invertida = ", s1);
for (int i = s1.length() - 1; i >= 0; i--)
   System.out.print("%c", s1.charAt(i));
s1.getChars(0, 5, charArray, 0);
System.out.printf("Letras selecionadas: ");
for (char character : charArray)
   System.out.print(character);
```

Strings01.java

Comparação de Strings

```
String s1 = "lcp";

s1 == "lcp" // true ou false?

String s2 = new String("lcp");

s2 == "lcp"; // true ou false?

s2.equals(s1);

s2.equals("lcp");
```

Strings02.java

Manipulação de Strings

```
String s = "ABABABABAB";

s = s.replace("B", "C");

System.out.println(s);

String scores = "azul, amarelo, vermelho";

String[] cores = scores.split(",");

for (String c : cores)

     System.out.println(c);

for (String c : cores)

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

int posAmarelo = scores.indexOf("amarelo");

System.out.println("amarelo começa em: " + posAmarelo);

String amarelo = scores.substring(posAmarelo, posAmarelo + "amarelo".length());

System.out.println(amarelo);
```

Strings03.java

Exercício

Receber uma frase do usuário (console) e exibir quantas palavras existem nesta frase e qual a palavra mais longa.

Exercício

```
static void maior1(String frase){
    String[] tokens = frase.split(" ");
    int maior = 0;
    String palavraMaior = "";
    for (String t : tokens){
        if (t.length() > maior){
            maior = t.length();
            palavraMaior = t;
        }
    }
    System.out.printf("temos %d palavras, sendo que a maior é %s",
tokens.length, palavraMaior);
}
```

Strings04.java

Exercício 1 - modo 2

```
static void maior2(String frase){

        String[] tokens = frase.split(" ");
        String palavraMaior =
Arrays.stream(tokens).max(Comparator.comparingInt(String::le
ngth)).get();
        System.out.printf("temos %d palavras, sendo que a
maior é %s", tokens.length, palavraMaior);
    }
```

Strings04.java

Exercício

Método: `toLowerCase()` -> converte caracteres para minúsculo

`toUpperCase()` -> converte caracteres para maiúsculo

Criar 1 método que recebe uma string qualquer e retorna: NeStEeStIlO.

Exercício

```
static String mudaletra(String entrada){
    String final1 = entrada.toLowerCase();
    char[] chars = final1.toCharArray();
    String nova = "";
    for (int i = 0; i < chars.length; i++){
        if (i % 2 == 0)
            nova += Character.toUpperCase(chars[i]);
        else
            nova += chars[i];
    }
    return nova;
}
```

Strings04.java

StringBuilder

- Classe que permite a mutabilidade de Strings.
- Quando a capacidade (`capacity()`) determinada internamente é excedida, ela é expandida dinamicamente.

```
StringBuilder cores = new StringBuilder();

cores.append("azul");

cores.append("amarelo");

cores.append("vermelho");

System.out.println(cores.toString());

cores.insert(4, "BRANCO");

System.out.println(cores.toString());
```

Métodos StringBuilder

- `length`
- `capacity`
- `setLength`
- `ensureCapacity`
- `charAt`
- `setCharAt`
- `getChars`
- `reverse`
- `append`
- `deleteChartAt`
- `delete`

Expressões regulares

- Sequências de caracteres/símbolos que definem strings.
- Utilizadas para validar se strings correspondem à uma expectativa de cadeia de caracteres.

Caractere	Correspondências	Caractere	Correspondências
\d	qualquer dígito	\D	qualquer caractere que não seja um dígito
\w	qualquer caractere de palavra	\W	qualquer caractere que não seja uma palavra
\s	qualquer caractere de espaço em branco	\S	qualquer caractere que não seja um espaço em branco

Quantificador	Correspondências
*	Localiza a zero ou mais ocorrências do padrão.
+	Localiza a uma ou mais ocorrências do padrão.
?	Localiza a zero ou uma ocorrência do padrão.
{*n*}	Localiza exatamente *n* ocorrências.
{*n*,}	Localiza pelo menos *n* ocorrências.
{*n*, *m*}	Localiza entre *n* e *m* (inclusive) ocorrências.

Expressões regulares

```
static void regex(){

        System.out.println("123".matches("\\d\\d\\d"));
        System.out.println("123".matches("\\d{3}"));
        System.out.println("13".matches("\\d\\d\\d"));
        System.out.println("13d".matches("\\d\\dd"));
        System.out.println("oi".matches("\\w\\w"));
        System.out.println("aa".matches("\\w\\w"));
        System.out.println("X".matches("[A-z]"));
        System.out.println("X1".matches("[A-z][0-9]"));
        System.out.println("1".matches("[A-z][0-9]"));
        System.out.println("X12".matches("[A-z][0-9]"));
        System.out.println("X12".matches("[A-z][0-9]*"));
        System.out.println("X".matches("[A-z][0-9]*"));
        System.out.println("X".matches("[A-z][0-9]+"));
        System.out.println("X".matches("[A-z][0-9]?"));
        System.out.println("X1".matches("[A-z][0-9]?"));
        System.out.println("X12".matches("[A-z][0-9]?"));
        System.out.println("X12".matches("[A-z][0-9]{2,4}"));
        System.out.println("X12345".matches("[A-z][0-9]{2,4}"));
        System.out.println("X 1".matches("[A-z]\\s[0-9]?"));
        System.out.println("X1".matches("[A-z]\\s[0-9]?"));

 }
```

Strings05.java

Exercício

Criar dois validadores de placas de carro:

- Placas antigas: ABC-1234
- Placas novas: ABC1D23

Exercício

```
public static void placas(){
        System.out.println("ABC-1234".matches("[A-Z]{3}-[0-9]{4}"));
        System.out.println("ABC-12345".matches("[A-Z]{3}-[0-9]{4}"));
        System.out.println("ABC1234".matches("[A-Z]{3}-[0-9]{4}"));


        System.out.println("ABC1D23".matches("[A-Z]{3}[0-9]{1}[A-Z]{1}[0-
        9]{2}"));
        System.out.println("ABC1234".matches("[A-Z]{3}[0-9]{1}[A-Z]{1}[0-
        9]{2}"));
        System.out.println("ABC12X4".matches("[A-Z]{3}[0-9]{1}[A-Z]{1}[0-
        9]{2}"));
}
```

Strings05.java

Expressões Regulares

- Além do método matches dos objetos String, temos as classes **Pattern** e **Matcher** (java.util.regex).

- **Pattern** e **Matcher** permitem o uso de expressões regulares pré-compiladas, aumentando a performance das verificações.

Expressões Regulares

```
public static void placas2(){

    Pattern pat = Pattern.compile("[A-Z]{3}-[0-9]{4}");
    String placa = "ABC-1234 " + "ABC-12345 " + "DCB-3598 " +
"ABC1234 " + "ABC1D23";

    Matcher mat = pat.matcher(placa);
    while (mat.find())
        System.out.println(mat.group());
}
```

Strings05.java

Exercícios - 1

Escreva um programa que receba um nome completo em qualquer tipo de capitalização e devolva o nome completo com apenas a primeira letra de cada nome maiúscula e as demais minúsculas. Obs.: tratar para que as preposições "da", "de" e "do" fiquem minúsculas.

Exemplo: JOÃO CARLOS DA sILVA => **João Carlos da Silva**

Exercícios - 2

- Receber duas Strings string1 e string2.
- Gerar uma nova String alternando entre os caracteres de string1 e string2, o restante da string maior entrará no final da nova String gerada.

Exemplo:

"string1" e "STRING2" => **sStTrRiInNgG12**

Exercícios - 3

- Receber uma string e utilizando manipulação de caracteres verificar se uma String é um palíndromo.

Exemplo:

"SUBINOONIBUS" = **VERDADEIRO**