# Curso de Java

## aula 05

Prof. Anderson Henrique

# Strings : Manipulação de Texto

- Construção
- Localização
- Comparação
- Extração
- Modificação

# Construção

- Podemos criar Strings literais

Ex.: String s1 = “Write Once”;

- Podemos criar Strings concatenando com outra String

Ex.: String s2 = s1 + “Run AnyWhere”;

- Podemos utilizar o método construtor da String

Ex.: String s3 = new String(“Java Virtual Machine”);

- Podemos construir String utilizando um array de caracteres

Ex.: char[ ] array = {‘J’,’a’,’v’,’a’};

String s4 = new String(array);

# Operações Básicas

- O método length( ) retorna a quantidade de caracteres, o comprimento de uma String

Ex.: int tamanho = s1.length( );

- Localizar um determinado caractere dentro da String, informando o índice

Ex.: char letra = s1.charAt(2)

- O método toString( ) retorna a própria String

String texto = S1.toString( );

# Localização

- Encontrar um determinado caractere dentro de uma String , retorna a posição onde esse caractere ou palavra se encontra

Ex.: int posicao = s3.indexOf(“J”)

- Encontrar a última ocorrência dentro de uma String, retorna a última posição onde esse caractere ou palavra se encontra

Ex.: int ultimaposicao = lastIndexOf(‘a’);

- Informa se essa String está ou não vazia, retorna valor booleano

Ex.: boolean vazia = s3.isEmpty( );

# Comparação

- Método que compara os valores entre duas Strings

Ex.: String texto = “TXT”;

boolean x = texto.equals(“txt”);

- Método que compara os valores entre duas Strings, e ignora letras maiúsculas ou minúsculas

Ex.: String nome = “Maria”;

boolean y = nome.equalsIgnoreCase(“maria”);

- Compara se o texto começa com um determinado texto ou caractere (prefixo)

Ex.: String curso = “JSE”;

boolean z = curso.startsWith(“JS”);

- Compara se o texto termina com um determinado texto ou caractere (sufixo)

Ex.: String professor = “Anderson”;

boolean w = professor.endsWith(“son”);

- Compara se a String é maior ou menor que outra String, vem antes ou depois, retorna int

Ex.: int c = “amor”.compareTo(“bola”); // -1

- Compara se uma String está dentro de outra String em uma determinada posição

Ex.: String so = “Olhe, olhe!”;

Boolean b = so.regionMatches(6, “Olhe”, 0, 4);

O parâmetro true, ignora letras maiúsculas e minúsculas no método regionMatches(true)

# Extração

- Quero encontrar uma substring, ou seja, um pedaço de texto dentro de um texto maior

Ex.: String s = “O Brasil é lindo”;

String sub = s.substring(11);

- Quero encontrar uma substring, utilizando o método sobrecarregado substring, passando a posição inicial e a final

String subsub = s.substring(2, 8);

# Modificação

- Concatenar, ou seja, unir ou juntar duas ou mais Strings

Ex.: String pais = “O Brasil “;

String juntar = pais.concat(“ é lindo”);

- Substituir caracteres de uma String, replace(antigo, novo)

Ex.: String s6 = “Aline Sousa”;

s6.replace(‘s’, ‘z’);

- Substituir apenas a primeira ocorrência de um caractere

s6.replaceFirst(“ ”,”_”);

- Substituir todas as ocorrências de um caractere

s6.replaceAll(“ ”,”_”);

- Modifica toda a String em letras maiúsculas, toUpperCase( )

Ex.: String frase = “que país é esse?”;

frase.toUpperCase( );

- Modifica toda a String em letras minúsculas, toLowerCase( )

frase.toLowerCase( );

- Elimina todos os espaçamentos de caracteres desnecessários, antes e depois

Ex.: String spc = “ espaços ”;

spc.trim( );

# Tokenização

- É o processo de pegar grandes quantidades de dados e dividi-los em pedaços pequenos.
- Esses dados são compostos por duas coisas: tokens e delimitadores.
- Tokens sãos os pedaços propriamente ditos dos dados, e os delimitadores são as expressões que separam os tokens uns dos outros.

Ex.: String s = “XHTML; CSS; JavaScript; jQuery; Java”;

String[ ] tokens = s.split(“;”);

Ex.: String frase = “Venha estudar a linguagem Java”;

String[ ] palavras = frase.split(“ ”);

# StringBuffer e StringBuilder

- Um texto de um objeto String nunca pode ser modificado, para isso existe a classe StringBuffer que permite a modificação do seu conteúdo.
- As classes tem o mesmo conjunto de métodos, a única diferença é que na classe StringBuffer os métodos são sincronizados, ou seja, são muito uteis quando você está em um ambiente multiprocessado.

Ex.: StringBuffer s0 = new StringBuffer( );

Ex.: StringBuilder s1 = new StringBuilder(“Java”);

s1.capacity;

- O método capacity informa a capacidade de armazenamento de novos caracteres dentro desses objetos sem a necessidade de alocar mais memória.

s1.reverse( );

- O método reverse reverte a ordem dos caracteres de uma String

s1.append(“ Trabalhando ”);

char[ ] c = {‘c’, ’o’, ’m’};

s1.append(c).append(“ Textos. ”);

- O método append permite adicionar novos textos em uma String

Ex.: StringBuilder url = new StringBuilder(“www.java.com”);

url.delete(0, 4).toString( );

- O método delete permite remover textos em uma String

Prosseguiremos no próximo slide...

Professor: Anderson Henrique

Programador nas Linguagens Java e PHP