MANIPULAÇÃO DE STRINGS

Todos outros valores que utilizamos em Java, com excepção dos tipos ditos primitivos, são objectos. Um dos objectos mais utilizados é a String (com S maiúsculo). Strings podem ser declaradas de duas formas: como variáveis normais (mas lembrando que Strings são objetos) e com construtores (um construtor especifica como um objecto deve ser inicializado).

• String str = "ABC"; *// como variável*

• str = new String(); *// construtor padrão. Declara e instancia uma String vazia*

• str = new String("olá!"); */* construtor sobrecarregado. Declara e instancia uma String com o conteúdo "olá!" */*

Métodos mais comuns para manipulação de Strings

• **length()**
int v = s2.length();
determina o número de caracteres de uma String.
Ex: int tamanho = s2.lenght();

• **charAt()**
char c = s2.charAt(int pos);
captura um caracter de uma String em uma posição específica.
Ex: char caracter = s2.charAt(2);

• **equals()**
boolean b = s2.equals(String s1);
verifica o conteúdo de duas Strings quanto à igualdade de conteúdo. O conteúdo de duas Strings não pode ser verificado com a utilização do operador ==, pois o mesmo irá comparar as referências dos objetos e não seu conteúdo.
Ex: boolean saoIguais = s2.equals(s1);

• **equalsIgnoreCase()**
boolean b = s2.equalsIgnoreCase(String s1);
verifica o conteúdo de duas Strings quanto à igualdade de conteúdo, ignorando variações entre maiúsculas e minúsculas.
Ex: boolean saoIguais = s2.equalsIgnoreCase(s1);

• **compareTo()**
int v = s2.compareTo(String s1);
compara duas Strings e devolve 0 (zero) se ambas forem iguais; um número negativo se a String que chama o método for menor que a String passado como parâmetro ou um número positivo se a String que chama o método for maior que a String que for passado como parâmetro. O significado do maior e menor é alfabeticamente.
Ex: int valor = s2.compareTo(s1);

• **compareToIgnoreCase()**
int v = s2.compareToIgnoreCase(String s1);
compara duas Strings ignorando variações entre maiúsculas e minúsculas e devolve 0 (zero) se ambas forem iguais; um número negativo se a String que chama o método for menor que a String passado como parâmetro ou um número positivo se a String que chama o método for maior que a String que for passado como parâmetro. O significado do maior e menor é alfabeticamente.
Ex: int valor = s2.compareToIgnoreCase(s1);

• **substring()**
String str = s2.substring(int beginIndex);
retorna uma nova String, copiando um trecho indicado de uma String específico. Neste caso, da posição (*beginIndex*) até o fim da String.
Ex: String novaString = s2.substring(2);

String str = s2.substring(int beginIndex, int endIndex);
retorna uma nova String, copiando um trecho indicado de uma String específico. Neste caso, da posição (*beginIndex*) até a posição (*endIndex – 1*) ou seja, *beginIndex* inclusive e *endIndex* exclusive.
Ex: String novaString = s2.substring(2, 9);

• **replace()**
String str = s2.replace(char oldChar, char newChar);
substitui cada ocorrência de um caracter em uma String por outro. Substitui todas as ocorrências de *oldChar* por *newChar*.
Ex: String novaString = s2.replace('o', 'O');

• **indexOf()**
int v = s2.indexOf(char caracter);
retorna a posição da primeira ocorrência de um caracter numa String; Retorna um valor inteiro com a posição do caracter ou -1 caso o caracter não exista na String.
Ex: int posicao = s2.indexOf('a');

int v = s2.indexOf(char caracter, int deOnde);
retorna a posição da primeira ocorrência de um caracter numa String a partir da posição especificada no segundo argumento (deOnde); Retorna um valor inteiro com a posição do caracter ou -1 caso o caracter não exista na String.
Ex: int posicao = s2.indexOf('a', 5);

int v = s2.indexOf(String s1);
retorna a posição do primeiro caracter da primeira ocorrência de uma String numa outra String; Também pode se passar um segundo argumento de onde iniciar a pesquisa.
Ex: int posicao = s2.indexOf("bc");

• **lastIndexOf()**
int v = s2.lastIndexOf(. . .);
funciona semelhante ao método indexOf, com todas as sobrecargas, mas localiza a última ocorrência de em uma String, inicia a pesquisa a partir do final da String; Retorna um valor inteiro com a posição do caracter ou -1 caso o caracter não exista na String.
Ex: int posicao = s2.indexOf('a');

• **toUpperCase()**
String str = s2.toUpperCase();
gera uma nova String com todas as letras em maiúsculas.
Ex: String novaString = s2.toUpperCase();

• **toLowerCase()**
String str = s2.toLowerCase();
gera uma nova String com todas as letras em minúsculas.
Ex: String novaString = s2.toLowerCase();

• **trim()**
String str = s2.trim();
gera uma nova String removendo os caracteres em branco no início e fim da String original.
Ex: String novaString = s2.trim();

• **startsWith()**
boolean b = s2.startsWith(String prefixo);
verifica se uma String é iniciada com uma sequência determinada de caracteres, retornando verdadeiro em caso afirmativo.
Ex: boolean verd = s2.startsWith("bc");

• **endsWith()**
boolean b = s2.endsWith(String sufixo);
verifica se uma String é encerrada com uma sequência determinada de caracteres, retornando verdadeiro em caso afirmativo.
Ex: boolean verd = s2.endsWith("bc");