

Material de Apoio 8

Cadeias de Caracteres – a classe String

Qualquer cadeia de caracteres delimitada por "" (aspas) representa uma string. Por exemplo: "ab+cd3". Uma cadeia de caracteres representa uma constante, ou seja, sua representação em memória não pode ser alterada. A classe String possibilita a criação de objetos para manipular strings. Estes sim, permitem alterar o conteúdo da cadeia, uma vez que a área de memória responsável pelo armazenamento dos caracteres é mantida em tempo de execução. A classe String oferece métodos que permitem:

- Examinar individualmente os caracteres da seqüência;
- Comparar duas strings;
- Fazer buscas na string;
- Extrair sub-strings;
- Criar cópias da string com todos caracteres em maiúsculo (ou minúsculo).

Outra funcionalidade adicionada a objetos string é o operador de concatenação, realizado através do + (operador +).

Maiores informações em: <http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html>

Alguns construtores:

String()	Cria um novo objeto, cuja seqüência inicial de caracteres encontra-se vazia. <code>s = new String();</code>
String(char[] values)	Cria um novo objeto, cuja seqüência inicial é um array de caracteres. <code>char data[] = {'a', 'b', 'c'};</code> <code>String str = new String(data);</code>
String(String original)	Cria um novo objeto, contendo uma cópia do conteúdo do objeto passado como parâmetro. <code>s1 = new String(s);</code>

Alguns métodos:

char charAt(int index)	Retorna o caracter que esta na posição indicada
int compareTo(String anotherString)	Compara lexicograficamente duas strings.
int compareToIgnoreCase(String str)	Compara lexicograficamente duas strings ignorando caixa alta ou baixa.
String concat(String str)	Retorna uma nova string, contendo a concatenação da string corrente com o parâmetro recebido.
int indexOf(int ch)	Retorna o índice da primeira ocorrência do caracter ch.
int indexOf(int ch, int fromInd)	Retorna o índice da primeira ocorrência do caracter ch a partir da posição fromInd.
int indexOf(String str)	Retorna o índice da primeira ocorrência da sub-string str dentro da string corrente.
int indexOf(String str, int fromInd)	Retorna o índice da primeira ocorrência da sub-string str dentro da string corrente a partir da posição fromInd.
int lastIndexOf(int ch)	Retorna o índice da última ocorrência do caracter ch.
int length()	Retorna o tamanho, em caracteres, da string.
String replace(char oldChar, char newChar)	Retorna uma nova string, a qual é uma cópia da string original, sendo que todas as ocorrências de oldChar foram trocadas por newChar.
String substring(int beginInd)	Retorna uma nova string que é a sub-string da string corrente iniciando em beginInd.
String substring(int beginIndex, int endIndex)	Retorna uma nova string que é uma sub-string da string corrente, iniciando em beginIndex e terminando em endIndex.
String toLowerCase()	Retorna uma nova string, onde a string original encontra-se convertida para caixa baixa.
String toUpperCase()	Retorna uma nova string, onde a string original encontra-se convertida para caixa baixa.
static String valueOf(boolean b)	Retorna uma string representando um valor booleano.
static String valueOf(char c)	Retorna uma string representando um caracter.
static String valueOf(double d)	Retorna uma string representando um valor ponto flutuante duplo.
static String valueOf(float f)	Retorna uma string representando um valor em ponto flutuante simples.
static String valueOf(int i)	Retorna uma string representando um valor inteiro.

Exercícios

- Crie uma classe Nome. Esta classe possui como estado interno um objeto String inicializado na construção.
 - Crie dois métodos construtores. Um que não recebe parâmetros e lê uma cadeia de caracteres do teclado e outro que recebe como parâmetro a cadeia inicial.
 - Crie um método `tudoMinusculo`, que coloque todos os caracteres da string em minúsculo (caixa baixa).
 - Crie um método que retorne o tamanho, em número de caracteres, da string armazenada.
 - Crie um método que conte o número de ocorrências de um determinado caractere recebido como parâmetro.
 - Construa um classe Teste, implementando o método `public static void main(String a[])` que crie dois objetos da classe nome e verifique se os objetos possuem ou não o mesmo string.
- Partindo da classe criada no Exercício 1, considere que a cadeia de caracteres armazenada representa o nome de uma pessoa.
 - Crie um método que coloque em caixa alta o primeiro caractere de cada componente do nome (Ex: `gerson geraldo homrich cavalheiro` retorna `Gerson Geraldo Homrich Cavalheiro`).
 - Crie um método que retorne o primeiro prenome e o último sobrenome do nome (Ex: `Gerson Geraldo Homrich Cavalheiro` retorna `Gerson Cavalheiro`).
 - Crie um método que retorna as iniciais de um nome (Ex: `Gerson Geraldo Homrich Cavalheiro` retorna `GGHC`).
- Partindo da classe criada no Exercício 1, implemente um método que permite verificar se a pessoa cujo nome está armazenado pertence a uma determinada família (Ex. `pertenceFamilia("Cavalheiro")` vai retornar `true` para um objeto armazenando `Gerson Geraldo Homrich Cavalheiro`).
- Crie uma classe `NomeFilho`. Esta classe deve receber, no construtor, dois objetos da classe Nome (Exercício 3) e um terceiro parâmetro representando o prenome de uma pessoa. O atributo mantido nesta classe representa o nome do filho cujos pais possuem os nomes recebidos como parâmetro no construtor. O método construtor deve gerar o nome do filho, considerando que:
 - O prenome deve ser aquele informado como terceiro parâmetro.
 - O primeiro sobrenome é o sobrenome da mãe. Note que, para o nome a mãe pode ocorrer um de dois casos. Primeiro caso: o último sobrenome da mãe é igual ao último sobrenome do pai – nesta situação o sobrenome de família da mãe é seu penúltimo sobrenome. Segundo caso: o último sobrenome da mãe não é igual ao último sobrenome do pai (a mãe não adotou o sobrenome do marido) – nesta situação o último sobrenome da mãe é o sobrenome da família da mãe.
 - Defina um segundo construtor que considere que o nome do pai não é informado.
- Implemente uma classe `Desenha`, que possua quatro. Três destes métodos devem receber um valor inteiro como parâmetro e deve ser capaz de montar sejam capazes de montar em uma string, as seguintes forma geométricas usando o caractere `*`:

<code>void quadrado(int n)</code>	<code>void triangulo(int n)</code>	<code>void diagonal(int n)</code>
n corresponde ao lado	n corresponde a base e a altura	n corresponde ao lado
<pre>***** ***** ***** ***** ***** ***** ***** *****</pre>	<pre> * ** *** **** ***** ***** *****</pre>	<pre> ***** * ***** ** ***** *** ***** **** ***** ***** * ***** \</pre>

O quarto método deve ser capaz de “imprimir” a referida string na tela.

- Jogo da Forca.** Implemente um jogo da forca, onde o Jogador 1 entra com uma palavra e o Jogador 2 tenta adivinhar a palavra “escondida”. Para cada letra “descoberta” a palavra deve tomar forma na tela.
 - Variante 1: o Jogador 2 não tem limite de erros e o programa termina somente quando a palavra for descoberta.
 - Variante 2: o Jogador 2 tem um número limitado de erros possíveis: assumo um valor igual ao número de partes que um corpo humano pode ser dividido :-). O programa termina quando a palavra for descoberta ou quando o Jogador 2 já tiver sido “enforcado”.
 - Dica: gere dois strings, um com a palavra a ser descoberta e outro, com igual tamanho, inicializado com “_” (caractere sublinhado) representando o que já foi descoberto. A cada letra descoberta, mostrar na tela a palavra em formação.