

## Material de Apoio Aula 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:

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

#### Alguns métodos:

<b>char charAt(int index)</b>	Retorna o caractere que esta na posição indicada
<b>int compareTo(String anotherString)</b>	Compara lexicograficamente duas strings.
<b>int compareToIgnoreCase(String str)</b>	Compara lexicograficamente duas strings ignorando caixa alta ou baixa.
<b>String concat(String str)</b>	Concatena no final da string o parâmetro recebido.
<b>int indexOf(int ch)</b>	Retorna o índice da primeira ocorrência do caractere ch.
<b>int indexOf(int ch, int fromInd)</b>	Retorna o índice da primeira ocorrência do caractere ch a partir da posição fromInd.
<b>int indexOf(String str)</b>	Retorna o índice da primeira ocorrência da sub-string str dentro da string corrente.
<b>int indexOf(String str, int fromInd)</b>	Retorna o índice da primeira ocorrência da sub-string str dentro da string corrente a partir da posição fromInd.
<b>int lastIndexOf(int ch)</b>	Retorna o índice da última ocorrência do caractere ch.
<b>int length()</b>	Retorna o tamanho, em caracteres, da string.
<b>String replace(char oldChar, char newChar)</b>	Retorna uma nova string, a qual é uma cópia da string original, sendo que todas as ocorrências de oldChar foram trocadas por newChar.
<b>String substring(int beginInd)</b>	Retorna uma nova string que é a sub-string da string corrente iniciando em beginInd.
<b>String substring(int beginIndex, int endIndex)</b>	Retorna uma nova string que é uma sub-string da string corrente, iniciando em beginIndex e terminando em endIndex.
<b>String toLowerCase()</b>	Converte a string para caixa baixa.
<b>String toUpperCase()</b>	Converte a string para caixa alta.
<b>static String valueOf(boolean b)</b>	Retorna uma string representando um valor booleano.
<b>static String valueOf(char c)</b>	Retorna uma string representando um caractere.
<b>static String valueOf(double d)</b>	Returns Retorna uma string representando um valor ponto flutuante duplo.
<b>static String valueOf(float f)</b>	Retorna uma string representando um valor em ponto flutuante simples.
<b>static String valueOf(int i)</b>	Retorna uma string representando um valor inteiro.

## Exercícios

1. Crie uma classe Nome. Esta classe possui como estado interno um objeto String inicializado na construção.
  - a. 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.
  - b. Crie um método tudoMinusculo, que coloque todos os caracteres da string em minúsculo (caixa baixa).
  - c. Crie um método que retorne o tamanho, em número de caracteres, da string armazenada.
  - d. Crie um método que conte o número de ocorrências de um determinado caractere recebido como parâmetro.
  - e. 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 a mesma string.
2. Partindo da classe criada no Exercício 1, considere que a cadeia de caracteres armazenada representa o nome de uma pessoa.
  - a. Crie um método que coloque em caixa alta o primeiro caractere de cada componente do nome (Ex: gerson geraldo homrich cavaleiro retorna Gerson Geraldo Homrich Cavaleiro).
  - b. Crie um método que retorne o primeiro prenome e o último sobrenome do nome (Ex: Gerson Geraldo Homrich Cavaleiro retorna Gerson Cavaleiro).
  - c. Crie um método que retorne as iniciais de um nome (Ex: Gerson Geraldo Homrich Cavaleiro retorna GGHC).
3. 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("Cavaleiro")` vai retornar true para um objeto armazenando Gerson Geraldo Homrich Cavaleiro).
4. 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.
5. Reimplemente o Exercício 6 do material de Apoio 6 (pode ser usado qualquer dos comandos de controle de iteração: while, do...while ou for). No entanto, não imprima diretamente os caracteres na tela, gere uma “imagem” da figura em uma String e somente após ter construído a imagem, apresente-a com uma única chamada de saída na tela.
6. **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.
  - a. Variante 1: o Jogador 2 não tem limite de erros e o programa termina somente quando a palavra for descoberta.
  - b. 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: gerencie dois strings, um com a palavra a ser descoberta e outro, com igual tamanho, inicializado com “\_” (caractere sublinha) representando o que já foi descoberto. A cada letra descoberta, mostrar na tela a palavra em formação.
7. **Cálculo de séries – Exercício extra**
  - a. Certifique-se que o Item b do Exercício 2 da lista de Apoio 6 foi implementado (cálculo de Fatorial);
  - b. Implemente o cálculo do valor de Fibonacci de um número n. Sabe-se que  $Fibonacci(1) = Fibonacci(0) = 1$  e que  $Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2)$  para qualquer  $n > 1$ .
  - c. Identifique o que objetos destas classes têm, ou podem ter, em comum.
8. **Cálculo de área – Exercício extra**
  - a. Implemente uma classe Quadrado. Esta classe deve definir como estado interno dois atributos (double) que sejam correspondentes ao lado do quadrado e a sua área. Implemente para esta classe o método `calculaArea` que retorna a área do quadrado e atualiza seu estado interno.
  - b. Faça o equivalente ao item a deste exercício para círculo, triângulo, etc.
  - c. Identifique o que objetos destas classes têm, ou podem ter, em comum.