Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática

Fundamentos da Programação

Profa. Luana Müller

Manipulação de *Strings* com Java

De acordo com a Documentação oficial (ORACLE), *"Strings, which are widely used in Java programming, are a sequence of characters. In the Java programming language, strings are objects. The Java platform provides the String class to create and manipulate strings."*

Documentação disponível em: https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

Logo, Strings são objetos provenientes de uma classe que cria e manipula cadeias de caracteres (chars), e nos permite assim, armazenar palavras e textos. Diferentemente de outros tipos de dados primitivos (int, char, double, boolean, etc) a String possui comandos que nos permite solicitar coisas específicas da palavra/texto com as quais estamos trabalhando.

Se quisermos, por exemplo, remover espaços em branco e substituirmos por um underline, teremos um comando específico para isso. Ou se quisermos "pegar" somente uma determinada letra de uma String, teremos outro comando para nos ajudar nesta tarefa. Logo, manipular *Strings* envolve conhecer e compreender como utilizar esses comandos.

No Java, esses comandos são conhecidos como métodos. Métodos são compostos por 3 partes:

a) Um tipo de retorno: o retorno define o resultado do comando que estamos usando, ou seja, o que iremos obter após a sua execução. Por exemplo:
   - O método .length() nos retorna um número inteiro que corresponde ao tamanho da palavra (número de caracteres).
   - O método .isEmpty() retorna um booleano que será falso sempre que houver 0 caracteres na palavra.

b) Um nome: o nome do método é sempre referente a ação que o mesmo executa.

c) Uma lista de parâmetros: os parâmetros, são informações que o método exige que informemos para conseguirmos realizar a ação. Por exemplo:
   - O método .replace(char oldChar, char newChar) substitui todas as ocorrências de uma letra na palavra por outra. No entanto, ele exige que informemos a ele qual a letra queremos substituir (oldChar) e a por qual letra iremos substituí-la (newChar).

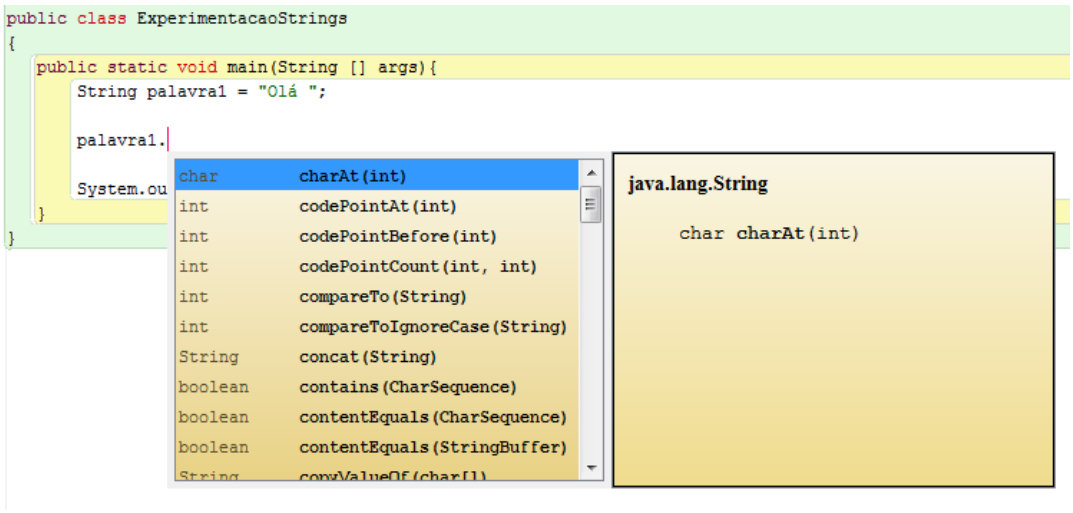
Experimentação com Strings

Criar String é um procedimento simples. Basta declarar uma variável com o tipo String, e adicionar um conteúdo a mesma. Se estivermos usando a classe Scanner para ler as informações digitadas pelo usuário, usamos os métodos .next() ou .nextLine() para fazer a leitura de Strings.

```
public class ExperimentacaoStrings
{
    public static void main(String [] args){
        String palavra1 = "Olá ";

        System.out.println(palavra1);
    }
}
```

Manipular Strings envolve chamar os métodos adequados e trabalhar com seus resultados (retornos). Para chamar um método de uma String basta escrever o nome da String seguido de um ponto (.) e após, pressionar Ctrl + Space. Você observará uma janela semelhante à da imagem a seguir:



A primeira informação é referente ao tipo de dado que o método retorna. A informação entre parênteses após o nome do método, é referente aos parâmetros necessários. Por exemplo, o método chartAt(int) irá retornar um char, e exige que passemos um int para utilizarmos o mesmo.

Caso o tipo de retorno seja void, significa que o método não irá retornar informações.

(Informações sobre o que cada método faz podem ser encontradas na documentação oficial da ORACLE).

Concatenação

Concatenação nada mais é do que juntar Strings em uma só. Isto pode ser feito de duas formas:

a) usando o sinal de adição (+) como operador de concatenação.

```
public class ExperimentacaoStrings
{
    public static void main(String [] args){
        String palavra1 = "Olá ";
        String palavra2 = "mundo!";

        String tudoJunto = palavra1 + palavra2;
        System.out.println(tudoJunto);
    }
}
```

b) Usando o método .concat que retorna uma nova String formada da junção da String principal com a String indicada como parâmetro.

```
public class ExperimentacaoStrings
{
    public static void main(String [] args){
        String palavra1 = "Olá ";
        String palavra2 = "mundo!";

        String tudoJunto = palavra1.concat(palavra2);
        System.out.println(tudoJunto);
    }
}
```

Tamanho da String

Podemos descobrir o número de caracteres que compõe a String usando o método .length().

```
public class ExperimentacaoStrings
{
    public static void main(String [] args){
        String palavra1 = "Olá ";
        String palavra2 = "mundo!";

        String tudoJunto = palavra1.concat(palavra2);
        int tamanhoDaString = tudoJunto.length();

        System.out.println("Tamanho da String: " + tamanhoDaString);
    }
}
```

Converter todas as letras da String para maiúsculo ou minúsculo

Para colocar todas as letras em maiúsculo ou minúsculo, podemos usar os métodos .toUpperCase() e .toLowerCase(), como na imagem a seguir.

```
public class ExperimentacaoStrings
{
    public static void main(String [] args){
        String palavra1 = "Olá ";
        String palavra2 = "mundo!";

        String tudoJunto = palavra1.concat(palavra2);
        String tudoMaiusculo = tudoJunto.toUpperCase();
        String tudoMinusculo = tudoJunto.toLowerCase();

        System.out.println("Original: " + tudoJunto);
        System.out.println("Maiusculo: " + tudoMaiusculo);
        System.out.println("Minusculo: " + tudoMinusculo);
    }
}
```

Observe que a String original não é modificada.

Comparação

Comparar uma String não é uma tarefa tão simples como comparar dois números, por isso não podemos usar somente operadores relacionais para tal tarefa. No entanto, a classe String nos oferece diversos métodos, possibilitando assim fazer diversas comparações sobre o conteúdo de uma String.

O método .equals verifica se duas Strings são iguais, e retorna true ou false como resultado.

```
public class ExperimentacaoStrings
{
    public static void main(String [] args){
        String palavra1 = "Olá ";
        String palavra2 = "mundo!";

        String tudoJunto = palavra1.concat(palavra2);
        String outraPalavra = "olá mundo!";

        boolean ehIgual = tudoJunto.equals(outraPalavra);

        System.out.println("São iguais? " + ehIgual);
    }
}
```

Observe o resultado deste método. As duas Strings possuem o mesmo texto, porém o resultado deste método foi false, indicando que as Strings são diferente.

Por se tratar de uma linguagem Case Sensitive, o Java considera letra maiúsculas e minúsculas diferentes. No caso do exemplo, podemos observar que há uma diferença na primeira letra das Strings. Altere o método .equals por .equalsIgnoreCase e observe a mudança do resultado.

No entanto se precisarmos comparar lexograficamente (comparação alfabética) duas Strings, temos outros métodos para tal tarefa. O método compareTo e compareToIgnoreCase possuem esta finalidade. Ambos métodos retornam um número de acordo com a comparação feita entre a String que chama o método, e aquela que está sendo passada por parâmetro.

```
public class ExperimentacaoStrings
{
    public static void main(String [] args){
        String palavra1 = "Ana";
        String palavra2 = "João";

        int retorno = palavra1.compareToIgnoreCase(palavra2);
        System.out.println("Retornou " + retorno);
        if(retorno == 0){
            System.out.println("São iguais");
        } else if(retorno < 0){
            System.out.println(palavra1 + " vem antes de " + palavra2);
        } else {
            System.out.println(palavra2 + " vem antes de " + palavra1);
        }

    }
}
```

Altere a variável palavra1 e escreva um nome que venha após "João". Observe o novo resultado.

Caractere na posição

Podemos também obter um caractere que se encontra em alguma posição dentro da string(). Para isso, usaremos o método charAt. O método charAt exige que passemos por parâmetro um número, que corresponde a posição da String da qual queremos o caracter. A primeira posição de um String é sempre a 0 (zero).

```
public class ExperimentacaoStrings
{
    public static void main(String [] args){
        String palavra = "Fundamentos";

        char letra1 = palavra.charAt(0);

        System.out.println("A primeira letra é " + letra1);
    }
}
```

Altere seu programa, e informe o número 20 como parâmetro do método. Porque ocorreu este erro? Como podemos evitar que ele aconteça?

CaractereS na posição

Substring é uma porção ou parte da String principal da qual pode formar outra String, ou seja, uma Substring é uma String formada a partir de uma String principal.

O primeiro parâmetro, obrigatório, é a posição que a substring deve iniciar (lembrando-se que Strings sempre começam da posição 0). O segundo parâmetro, opcional, é a posição final da substring, caso este parâmetro não seja indicado, a substring sempre irá até o último caractere encontrado.

```java
public class ExperimentacaoStrings
{
    public static void main(String [] args){
        String site = "http://pucrs.br";

        String substring1 = site.substring(7);
        String substring2 = site.substring(7,12);

        System.out.println("Sub 1: " + substring1);
        System.out.println("Sub 2: " + substring2);
    }
}
```

Substituir caracteres (ou sequencias de caracteres)

Podemos substituir letra por outras, ou até mesmo sequencias de caracteres por outras usando o método replace. Passamos ao método, por parâmetro, a String que deve ser substituída, e a String que irá substituí-la, conforme imagem.

```java
public class ExperimentacaoStrings
{
    public static void main(String [] args){
        String site = "http://pucrs.br";

        String siteComWWW = site.replace("http://", "www.");

        System.out.println("Site antes: " + site);
        System.out.println("Site depois: " + siteComWWW);
    }
}
```

Verificar conteúdo da string

Um método muito útil para verificar o conteúdo de uma String é o **contains**(), que retorna true se houver a sequência de caracteres especificada no parâmetro.

```java
public class ExperimentacaoStrings
{
    public static void main(String [] args){
        String site = "http://pucrs.br";

        boolean temTrecho = site.contains("pucrs");

        System.out.println("Contém? " + temTrecho);
    }
}
```