String

Strings são utilizadas frequentemente em várias linguagens de programação, não apenas Java. Embora Strings sejam uma sequência de caracteres em Java, e não um array de caracteres.

Em Java, o importante é entender que string não é um tipo de dado, mas sim uma classe. E suas variáveis são, na verdade, objetos dessa classe.

Em síntese, tanto um array de caracteres como uma string se apresentam da mesma forma.

Então, qual a vantagem de se usar uma classe e não um array de tipo primitivo?

A resposta se baseia exatamente no uso da orientação a objetos e no fato de que existem muitos métodos que podemos utilizar em um objeto instanciado da classe String, mesmo sendo os objetos desta classe imutáveis, ou seja, uma vez instanciados não podemos mudar o que está guardado dentro do objeto String.

Dentre as possibilidades oferecidas pela classe String podemos destacar algumas como:

- Concatenação;
- Tamanho da String;
- Converter para maiúsuculas ou minúsculas;
- Fazer comparações que incluem verificações no início e no fim;
- Extrair um caracter específico da string;
- Achar um índice da string, inclusive recursivamente (de trás para frente);
- Substituir uma parte da string ou ela completamente;
- Verificar se uma string contém outra;
- Dividir uma string em array ou vetor;
- Obter uma porção da string inteira, o que chamamos de substring.

Objetos Strings podem ser instanciadas usando um array de caracteres ou bytes com o operador new. Ou por meio de uma string literal. String literal é qualquer sequência de caracteres que estiver entre aspas (").

```
public class ExemploCriarString {
    public static void main(String[] args) {
        char[] charArray = { 'T', 'I', ' ', 'E', 'x', 'p', 'e', 'r', 't'};
        /* construindo uma string a partir de um array de char */
        String string1 = new String(charArray);
        /* construindo uma string a partir de uma string literal */
        String string2 = "www.tiexpert.net";

        System.out.println(string1);
        System.out.println(string2);
    }
}
```

Concatenação

Concatenação nada mais é do que juntar strings numa só. Isto pode ser feito de duas formas: uma usando o método concat() da classe String ou usando o sinal de adição (+) como operador de concatenação.

O método concat() retorna uma nova string formada da junção da string principal com a string indicada como parâmetro.

De uma forma mais simples, podemos usar o + para juntar várias strings ao mesmo tempo. Para mais detalhes, veja o artigo principal sobre concatenação.

Tamanho da string - length()

A classe String possui um método acessor que retorna o tamanho da String. Esse método é length().

```
public class TamanhoString {
    public static void main(String[] args) {
        String str = "TI Expert";
        System.out.println(str + " possui " + str.length() + " caracteres.");
    }
}
```

Letras Maiúsculas e Minúsculas

Podemos facilmente deixar todas as letras de uma sequência de caracteres maiúscula ou minúscula usando os métodos toUpperCase() e toLowerCase() respectivamente.

```
public class StringMaiusculoMinusculo {
    public static void main(String[] args) {
        String email = "MeuNoMe@MEUDominiO.coM.bR";
        String hexadecimal = "#aa33ff";
        System.out.println(email.toLowerCase());
        System.out.println(hexadecimal.toUpperCase());
    }
}
```

Comparação

Há várias formas de se fazer comparações com uma string, mas vale sempre lembrar que Java é case sensitive - diferencia letras maiúscula de minúsculas e vice-versa. Embora haja métodos próprios para comparações em que não há importância em diferenciar letras maiúsculas e minúsculas.

A comparação mais simples é usando o próprio operador de igualdade (==), mas por se tratar de um objeto é preferível que se use o método específico chamado equals().

O método equals() compara o objeto string com outro objeto, se ambos possuírem conteúdos iguais, então, é retornado verdadeiro (true), caso contrário falso (false). Isso quer dizer que se, por exemplo, tiver um objeto da classe Integer que vale 15 e uma string contendo os caracteres 1 e 5, a comparação de ambos resultará em verdadeiro.

A função equals() é case sensitive, para fazer uma comparação ignorando esta característica basta usar o método equalsIgnoreCase().

```
public class ComparacaoIgualdadeString {
    public static void main(String[] args) {
        String string1 = "TI Expert";
        String string2 = "ti expert";
        System.out.println("São iguais? (case sensitive)");
        System.out.println(string1.equals(string2) ? "sim" : "não");
        System.out.println("São iguais? (sem case sensitive)");
        System.out.println(string1.equalsIgnoreCase(string2) ? "sim" : "não");
    }
}
```

Há também uma forma de comparar strings lexicograficamente. Dessa forma, podemos verificar se uma string é identica (quando retorna-se 0), ou se ela tem um valor menor (quando retorna-se um número abaixo de 0) ou se ela tem um valor maior (quando retorna-se um número acima de 0).

O método para fazer tais comparações é o compareTo() (case sensitive) ou compareToIgnoreCase() (sem case sensitive).

```
public class ComparacaoString {
    public static void main(String[] args) {
        String string1 = "TI Expert";
        String string2 = "ti expert";
        int comparacao = string1.compareTo(string2);
        System.out.println("Comparação entre string1 e string2 (sensitive
case)");
        if (comparacao > 0) {
            System.out.println("string1 é lexicograficamente maior que
string2");
        } else if (comparacao < 0) {
            System.out.println("string1 é lexicograficamente menor que
string2");
        } else {
            System.out.println("string1 é lexicograficamente igual a string2");
        }
    }
}
```

Início e fim

Outra forma de fazer comparações é fazer testes no início e no fim de uma string.

Podemos fazer tais comparações usando dois métodos: startsWith() e endsWith().

StartsWith() verifica se há uma string no começo de outra string. StartsWith() também possui um segundo parâmetro opcional que determina a compensação inicial, ou seja, caso necessite verificar a string não da posição 0, mas de uma posição mais adiante.

EndsWith() verifica se há uma string no final de outra string. Diferentemente de startsWith(), o método endsWith() não possui compensação.

```
public class InicioFim {
    public static void main(String[] args) {
        String string1 = "http://www.tiexpert.net";
        System.out.println("A string " + string1 + " é:");
        // verifica se há 'http:' no inicio da string
        if (string1.startsWith("http:")) {
            System.out.println("uma URL");
        }
        /*
         * verifica se há 'www' no início da string, mas apenas a partir do
         * 8o. caracter, ou seja, após o prefixo 'http://', portanto deverá
         * ser compensado 7 caracteres
         */
        if (string1.startsWith("www", 7)) {
            System.out.println("uma página da web");
        }
        if (string1.endsWith(".br")) {
            System.out.println("um site registrado no Brasil");
        } else {
            System.out.println("não é um site registrado no Brasil");
        }
    }
}
```

Caracter na posição

Podemos também obter um caracter que se encontra em alguma posição dentro da string. Para isso, usaremos o método charAt().

CharAt() recebe um inteiro como argumento que indica a posição que queremos da string.

Importante: Strings seguem o mesmo conceito de vetores e arrays, portanto, seu primeiro caracter não está na posição 1, mas na posição 0.

```
public class ExtrairCaracter {
    public static void main(String[] args) {
        String string1 = "TI Expert";
        char caracter = string1.charAt(3);
        System.out.println("O 4o. caracter desta string é " + caracter);
    }
}
```

Índice da String

A classe string possui uma forma de encontrar o índice da primeira ocorrência de uma string dentro de outra string.

O método indexOf() retorna um número inteiro que indica exatamente em que posição ocorre uma string de busca, ou retorna um valor menor que 0 caso não encontre o valor requisitado. Assim como startsWith(), indexOf() também possui um segundo argumento que determina a compensação a ser feita caso a busca não possa ser feita desde a posição 0, mas de uma posição posterior.

Além do método indexOf(), há também o método lastIndexOf() que faz a mesma coisa que indexOf(), porém de forma recursiva (de trás para frente ou do fim da string para o começo)

```
public class IndiceString {
  public static void main(String[] args) {
    String string1 = "www.tiexpert.net";
    int posicao;
    posicao = string1.indexOf("tiexpert");
    if (posicao >= 0) {
      System.out.println("A string tiexpert começa na posição " + posicao);
    } else {
      System.out.println("Não há tiexpert na string");
    }
    posicao = string1.lastIndexOf(".com");
    if (posicao >= 0) {
      System.out.println("A string .com começa na posição " + posicao);
    } else {
      System.out.println("Não há .com na string");
    }
  }
}
```

Substituir string ou parte dela

Podemos substituir todas as ocorrências de uma string por uma nova string resultando em uma nova string de retorno.

Para executarmos esta operação usamos o método replace() que tem dois parâmetros: o primeiro será o que desejamos procurar dentro da string e o segundo a string que colocaremos no lugar da antiga.

Nota: replace também funciona com o tipo primitivo char.

```
public class SubstituirString {
    public static void main(String[] args) {
        String string1 = "http://tiexpert.net";
        System.out.println(string1.replace("http://", "www."));
    }
}
```

Verificar conteúdo da string

Um método muito útil para verificar o conteúdo de uma string é o contains().

Contains() retorna verdadeiro (true) se houver a sequência de caracteres especificada no parâmetro.

```
public class VerificarConteudo {
    public static void main(String[] args) {
        String string1 = "http://www.tiexpert.net";
        System.out.println("É uma página da web?");
        System.out.println(string1.contains("www.") ? "sim" : "não");
    }
}
```

Dividir em vários vetores ou arrays

Também é possível dividir uma string em um vetor ou array de strings, o que possibilita trabalhar com pedaços menores e mais lógicos da string.

Para dividir a string podemos usar o método split(). O método split usa uma expressão regular para fazer a divisão. Para simplificar o entendimento, podemos usar como parâmetro uma outra string como base. Mas vale atentar no fato de que a expressão utilizada para ser feita a quebra desaparece no fim do processo. Ex.: Se minha string for 15:30:00, se utilizarmos os dois pontos (:) para a separação conseguiremos um array com 3 posições que seriam: [0]->15, [1]->30, [2]->00.

```
public class Exemplo {
    public static void main(String[] args) {
        String string1 = "15:30:00";
        String[] stringDividida = string1.split(":");
        for (int i = 0; i < stringDividida.length; i++) {
            System.out.println("stringDividida[" + i + "]=" +
stringDividida[i]);
        }
    }
}
```

Substring

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.

Para obtermos uma substring usamos um método homônimo (de mesmo nome) com um ou dois parâmetros.

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 caracter encontrado.

Nota: O caracter na posição inicial fará parte da substring, porém o caracter na posição final não faz parte da substring. Vejamos o exemplo: Se tivermos uma string de 9 caracteres, portanto, de 0 a 8 posições, se fizermos a substring da seguinte forma: stringOriginal.substring(3,8);

0	1	2	3	4	5	6	7	8
T	I		E	X	P	E	R	T

O resultado será Exper. Então, se quisermos que a 8ª posição entre, devemos usar a posição seguinte 9, assim: stringOriginal.substring(3,9);

```
public class ExemploSubstring {
    public static void main(String[] args) {
        String url = "www.tiexpert.net";
        String dominio = url.substring(4);
        String site = url.substring(url.indexOf('.') + 1,
url.lastIndexOf('.'));
        System.out.println("Análise da string:");
        System.out.println("Domínio: " + dominio);
        System.out.println("Site: " + site);
    }
}
```