Strings

Page

- Uma String é um conjunto de caracteres
- Atenção que um caracter não é uma String
- Para diferenciar entre strings e caracteres usam-se as `" "` para as strings e `' '` para caracteres

```java
"Isto é uma string" // exemplo de string
'I' // exemplo de caracter
"A" // isto é uma string com um único caracter
'A' // isto é um caracter
```

- Em Java existe o tipo `String` para representar as strings
  - As strings são objetos
  - As strings não são arrays de caracteres

```java
String string = "Isto é uma string"; // exemplo de string
String nome = "Felizberto Desgraçado"; // outra string
char vogais[] = {'a', 'e', 'i', 'o', 'u' }; // array de
caracteres
```

- Como as strings são objetos não obedecem às mesmas regras dos tipos de variáveis estudados até agora
  - Obedecem às regras dos objetos

- Como quase todos os programas usam strings é necessário saber como manipulá-las
  - Toda a manipulação envolve o uso de métodos

Manipulações de strings

- Algumas manipulações de strings
    - Saber o tamanho de uma string
    - Concatenação de strings
    - Comparações de strings
    - Manipulação de caracteres dentro de uma string
    - Pesquisas em strings
    - Separação de strings

Tamanho de uma string

- Usar o método `length()`
    - Retorna o nº de caracteres presentes na String
    - É sempre um valor inteiro

```java
String nome = "Felizberto Desgraçado";
System.out.println( "O nome " + nome + " tem " + nome.length()
+ " caracteres");  // O nome Felisberto Desgraçado tem 21
caracteres
```

Concatenação de strings

- A concatenação de strings é a possibilidade de agrupar (somar) 2 ou mais strings
    - É a única manipulação que é suportada em Java sem o uso de métodos

```java
String primeiroNome = "Felizberto";
String apelido = "Desgraçado";

String nomeCompleto = primeiroNome + " " + apelido;
System.out.println( "O nome completo é" + nomeCompleto)  // O
nome completo é Felizberto Desgraçado
```

    - Reparar que até agora temos usado extensivamente a concatenação de strings

- O Java transforma automaticamente todos os tipos de dados predefinidos para String

```java
int num = 10;
float f = 2.0f;

System.out.println( "num = " + num + " f = " + f );// + num - Transformação de inteiro para String e consequente concatenação
                                                    // + f - Transformação de float para String e consequente concatenação
```

Comparações de strings

⚠️ - Para verificar se 2 strings são iguais NÃO se deve usar o `==`
- Quando aplicado a objetos o `==` indica se se trata do mesmo objeto e não se os objetos têm o mesmo valor
- Se aplicado a pessoas, por exemplo, o `==` verifica se a pessoa é a mesma e não se tem o mesmo nome, Bilhete de identidade, etc

```java
String s1 = "Isto não funciona bem";
                                                    // Os objetos são diferentes apesar de terem o mesmo conteúdo. Neste caso
                                                    // não são a mesma string, apesar do texto ser o mesmo
String s2 = "Isto não funciona bem";

if( s1 == s2 ) {
    System.out.println( "São iguais" );
} else {
    System.out.println( "Não são iguais" );      // Não são iguais
}
```

- Para verificar se 2 strings são iguais deve-se usar:
  - O método `equals`
  - O método `equalsIgnoreCase`

Método Equals:

- O `equals` liga ao conteúdo

```java
String s1 = "Isto funciona bem";
String s2 = "Isto funciona bem";

if( s1.equals( s2 ) )
    System.out.println( "São iguais" );
else
    System.out.println( "Não são iguais" );

// OUTPUT - São iguais
```

Método equalsIgnoreCase:

- Enquanto que o método anterior liga à diferença entre minúsculas e maiúsculas, o método `equalsIgnoreCase` não liga para essa diferença, ignora completamente se as strings apresentam caracteres maiúsculos ou minúsculos.

```java
String s1 = "Isto NÃO FUNCIONA";
String s2 = "Isto não funciona";

if( s1.equals( s2 ) )
    System.out.println( "São iguais" );
else
    System.out.println( "Não são iguais" );

// OUTPUT - Não são iguais
```

```java
String s1 = "Isto FUNCIONA BEM";
String s2 = "Isto funciona bem";

if( s1.equalsIgnoreCase( s2 ) )
    System.out.println( "São iguais" );
else
    System.out.println( "Não são iguais" );
```

```
// OUTPUT - São iguais
```

- Para comparar 2 strings pode-se usar o método `compareTo`
  - A sintaxe é `s1.compareTo(s2)` e retorna
    - Um valor < 0 se s1 < s2
    - Um valor > 0 se s1 > s2
    - 0 se s1 for igual a s2
  - A comparação é feita alfabeticamente

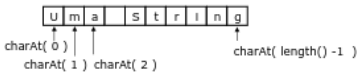
```javascript
String amalia = "Amália";
String eusebio = "Eusébio";

System.out.print("Os nomes ordenados alfabeticamentes são:");
if( amalia.compareTo( eusebio ) < 0 )
    System.out.println( amalia + ", " + eusebio );
else
    System.out.println( eusebio + ", " + amalia );

// OUTPUT - Os nomes ordenados alfabeticamente são: Amália, Eusébio
```

- Também se pode usar o método `compareToIgnoreCase`
  - A sintaxe e resultado é igual ao `compareTo`
  - A comparação é feita alfabeticamente, mas sem ligar às minúsculas e maiúsculas

```javascript
String zebra = "Zebra";
String antes = "antes";

if( zebra.compareTo( antes ) < 0 )
    System.out.println( zebra + ", " + antes );
else
    System.out.println( antes + ", " + zebra );

// OUTPUT - Zebra, antes
```

```JavaScript
String zebra = "Zebra";
String antes = "antes";

if( zebra.compareToIgnoreCase( antes ) < 0 )
    System.out.println( zebra + ", " + antes );
else
    System.out.println( antes + ", " + zebra );

// OUTPUT - antes, Zebra
```

- Uma string pode ser vista como um conjunto de caracteres
    - Logo deve-se poder manipular cada um deles em particular
        - Tal como num array os caracteres individuais são indexados e cemeçam no índice zero
    - Os caracteres individuais são acedidos usando o método `charAt(idx)`



- Uma string pode ser vista como um conjunto de caracteres
    - Exemplos de `charAt`

```JavaScript
String nome = "Felizberto Desgraçado";

char inicial = nome.charAt( 0 );
char ultima = nome.charAt( nome.lenght()- 1 );

System.out.println( "Inicial = " + inicial );
System.out.println( "Última = " + ultima );

// OUTPUT:
    // Inicial - F
    // Última - o
```

```JavaScript
String nome = "Felizberto Desgraçado";

for( int i = 0; i < nome.lenght(); i++ )
```

```
    System.out.println( nome.charAt( i ) );

// OUTPUT:
    // Felizberto Desgraçado
```

- Pode-se obter uma substring de uma dada string usando 2 métodos `substring()`
  - `substring(ini, fim)`
    - retorna a string entre os índices ini (inclusive) e fim (exclusive)
  - `substring(ini)`
    - retorna a substring que começa no índice ini (inclusive) até ao final da string

```
JavaScript
String nome = "Felizberto Desgraçado";

String proprio = nome.substring( 0, 10 );
String apelido = nome.substring( 11 );

System.out.println( "Nome próprio = " + proprio );
System.out.println( "Apelido = " + apelido );

// OUTPUT:
    // Nome próprio = Felizberto
    // Apelido = Desgraçado
```

- Para fazer pesquisas numa substring podem-se usar os métodos seguintes:
  - `indexOf(ch)`
    - indica em que índice aparece o caracter ch pela primeira vez
  - `indexOf(ch, idx)`
    - indica em que índice, superior a idx, aparece o caracter ch pela primeira vez
  - `indexOf(str)`
    - indica em que índice aparece a string str pela primeira vez
  - `indexOf(str, idx)`
    - indica em que índice, superior a idx, aparece a string str pela primeira vez

⚠️ Se não encontrarem o especificado, o índice retornado é -1

```JavaScript
String nome = "Felizberto Desgraçado";

int primeiroZ = nome.indexOf( 'z' );
int segundoE = nome.indexOf( 'e', 3 );
int berto = nome.indexOf( "berto" );

System.out.println( "Primeiro z encontrado em " + primeiroZ );
System.out.println( "Encontrado um e em " + segundoE );
System.out.println( "berto começa em " + berto );

// OUTPUT:
    // Primeiro z encontrado em 4
    // encontrado um e em 6
    // berto começa em 5
```

- Uma tarefa útil é partir a string em pedaços usado para isso um dado critério
    - Usa-se o método `split`
        - `split (critério)`
            - O critério é representado por uma string
            - Os critérios podem ser combinações de caracteres
            - Ou apenas um caracter
        - Retorna uma array de strings

```JavaScript
String nome = "Felizberto João Alegre Desgraçado";

String nomes[] = nome.split( “ " ); // usar o espaço como separador

for( int i = 0; i < nomes.lenght; i++ )
System.out.println( nomes[ i ] )

// OUTPUT:
    // Felizberto
    // João
```

```
    // Alegre
    // Desgraçado
```

Métodos