FICHA 3 - STRINGS

As **strings** são objetos da classe **`String`** que servem para armazenamento e manipulação de cadeias de caracteres. Esta classe pertence ao 'package' **`java.lang.`** Este 'package' tem a particularidade de ser automaticamente importado para todos os programas em **Java** – não necessita, portanto, de ser feita explicitamente a sua importação como acontece com outros 'packages'.

Uma vez que as **strings** são muito usadas, podemos criar uma **string** de forma simplificada (ao contrário de outros objetos). Por exemplo:

```
String nome = "Maria";
```

Um **carácter** de uma **string** é referido pela sua **posição** ou **índice** na **string**. Há que ter em atenção que o índice do **primeiro carácter** é **0**, tal como se se tratasse de uma tabela de caracteres.

De seguida, indicam-se alguns dos **métodos** desta classe disponibilizados às suas instâncias (objetos):

Método	Funcionalidade
`public char charAt (int index)`	Devolve o carácter armazenado na posição `index`.
`public int compareTo (String anotherString)`	Compara uma string com anotherString. Devolve 0 se forem iguais, um valor negativo se `string < anotherString`, um valor positivo no caso contrário.
`public String concat (String str)`	Concatena uma string com `str`.
`public int indexOf (int ch)`	Devolve a posição numa string do carácter `ch`.
`public int length ()`	Devolve o número de caracteres de uma string.
`public String substring (int beginIndex)`	Devolve uma substring de outra string - a partir da posição `beginIndex`.
`public String toUpperCase ()`	Converte uma string para maiúsculas.
`public String[] split (String regex)`	Converte uma string em substrings delimitadas pela String "delimitadora" passada como parâmetro (ou pelo início/fim da String original). Devolve uma tabela de Strings.

Por exemplo, para determinar o tamanho de **nome** bastava a instrução:

```
int t = nome.length();
```

O método **`length()`** do objeto **nome** devolve o tamanho da sua **string**.

Exercícios

1. Caracteres de uma frase

Escreva um programa que dada uma frase imprima todos os caracteres dessa frase, exceto os espaços, um carácter por linha.

2. Número de vogais

Escreva um programa que dada uma *string* apresente no ecrã o número de vogais que existem nessa *string*.

3. Palíndromo

Escreva um programa que dada uma *string* determine se é um palíndromo (i.e., se se lê da mesma forma do princípio para o fim e do fim para o princípio).

4. Junta "p"

Escreva um programa que, dada uma *string* **s** inserida pelo utilizador, lhe acrescente um "p" entre cada duas vogais consecutivas encontradas.
Exemplo:

s = "aeiou" → **resultado** = "apepipopu"
s = "cadeira" → **resultado** = "cadepira"

5. Conta palavras

Escreva um programa que, dadas duas *strings*, a primeira contendo uma frase e a segunda uma palavra, determine quantas vezes a palavra aparece na frase (admita que as palavras estão separadas exclusivamente por espaços em branco).
Exemplo: "gato" aparece 1 vez na frase "o gato comeu o biscoito"

6. Elimina palavras

Escreva um programa que leia uma *string* **s** dada pelo utilizador, e que a converta numa *string* que contenha apenas as palavras onde a letra 'a' apareça mais do que uma vez.

Exemplo:
s = "A Maria compra uma camisa amarela " → "Maria camisa amarela"

7. Data

Escreva um programa que leia uma data (data) no formato dd/mm/aaaa e a converta no formato dd de mmmm de aaaa. Exemplo de output para data =

"12/12/2022":

Exemplo: 12 de dezembro de 2022

8. ISBN

A maior parte dos livros publicados atualmente são identificados pelo ISBN (*International Standard Book Number*). Este número é composto por uma sequência de 10 dígitos decimais. Os primeiros 9 dígitos são utilizados para identificar o livro, e o décimo dígito é utilizado para verificar se os 9 dígitos precedentes estão convenientemente formados. O valor deste dígito de confirmação é selecionado de modo que o valor calculado pelo algoritmo descrito abaixo seja divisível por 11.

O algoritmo de verificação do ISBN é o seguinte: são calculadas duas somas, *s1* e *s*2, tendo por base os dígitos do ISBN (os dígitos devem ser separados por um espaço). A soma *s1* é obtida pela soma parcial dos dígitos do ISBN. A soma *s2* é a soma das somas parciais existentes em *s*1. O ISBN é considerado correto se o valor final de *s2* for divisível por 11.

Escreva um programa em Java que permita visualizar a aplicação do algoritmo, incluindo as somas parciais e totais, de acordo com o exemplo seguinte. O seu programa deverá também indicar se o ISBN é válido.

Exemplo:

```
ISBN original 0 8 9 2 3 7 0 1 0 6
Somas parciais (s1) 0 8 17 19 22 29 29 30 30 36
Somas totais (s2) 0 8 25 44 66 95 124 154 184 220
```

O ISBN dado é correto pois 220 é divisível por 11.

Nota: Para converter uma string num número inteiro pode utilizar o seguinte método da classe *Integer*: **`int parseInt(String)`**.

9. Palavras Compostas

Encontre as palavras compostas de duas palavras num dicionário. Uma palavra composta de duas palavras é uma palavra no dicionário que é a concatenação de exatamente duas palavras no dicionário. A entrada do problema consiste num número de palavras em letra minúscula, uma por linha. Estas palavras constituem o nosso dicionário. A saída consiste de todas as palavras compostas existentes no dicionário, uma por linha, por ordem alfabética.

Exemplo:

Input:

```
15
a
passeio
teu
fura
cão
ateu
furacão
dado
sol
mar
fim
marfim
soldado
ajuda
programar
```

Output:

```
ateu
furacão
marfim
soldado
```

10. Corretor ortográfico

Escreva um programa em Java que comece por ler um conjunto de 10 palavras diferentes. Seguidamente, o programa deve ler uma frase e tentar fazer a sua correção ortográfica, com base nas palavras obtidas inicialmente. Para cada palavra da frase, diferente das palavras da lista inicial, o programa deve sugerir as palavras dessa lista que obedeçam aos seguintes critérios:

1. As duas primeiras letras são iguais;
2. Das restantes letras, apenas duas podem ser diferentes;
3. Cumulativamente, pode ter mais uma ou duas letras no final.

Para cada sugestão, o programa deve perguntar ao utilizador se aceita ou não a troca de palavras sugerida. O utilizador pode responder com 's' ou 'n', consoante a sua decisão seja de aceitar ou não, respetivamente.

Finalmente, o programa deve escrever a frase resultante da correção.

Exemplo:

```
interessa
interessar
interessante
discipulo
disciplinar
disciplina
estamos
estar
este
esta
esda disciplima é muito intesessan
esda -> estar? n
esda - > este? n
esda -> esta? s
disciplima -> disciplinar? n
disciplima -> disciplina? s
intesessan -> interessar? n
intesessan -> interessante? s
esta disciplina é muito interessante
```