FICHA 03 – INTRODUÇÃO AOS OBJECTOS STRINGS

Objectos

Na **programação orientada a objectos** um **programa** é constituído por um conjunto de entidades chamadas **objectos**. Um **objecto** consiste num conjunto de **dados** (**variáveis**) e **acções** (**métodos**) relacionados.

Qualquer objecto pertence a uma **classe**. Uma **classe** define um conjunto de objectos semelhantes ou um **objecto** do ponto de vista genérico. Desta definição consta a especificação dos seus **atributos** (**variáveis**) e **comportamento** (**métodos**).

A linguagem **Java** possui uma variedade grande de classes de objectos predefinidas. Estas classes estão organizadas em **packages** (pacotes ou grupos de classes relacionadas).

Strings

As **strings** são objectos que servem para armazenamento e manipulação de cadeias de caracteres. Pertencem 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 um **string** por exemplo da seguinte forma:

String nome = "Maria";

Nota: outros tipos de objectos não são criados assim tão facilmente, mas, mais tarde, veremos como.

Um **caracter** de uma **string** é referido pela sua **posição** ou **índice** na **string**. Há que ter em atenção que o índice do **primeiro caracter** é **0**.

De seguida, indicam-se alguns dos **métodos** desta classe:

Método	Funcionalidade
public char charAt (int index)	Devolve o caracter 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 caracter 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.

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

int t = nome.length();

Exercícios

1. Palíndromo

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

2. 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”

3. 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.

Nota: Pode usar os métodos “static String User.readString(String mensagem)” e “static String User.readString()” para receber uma String do utilizador. No primeiro caso, poderá enviar também uma mensagem de *prompt* para o utilzador.

A Classe StringTokenizer

Uma classe importante para a manipulação de cadeias de caracteres é a classe **StringTokenizer** que facilita a divisão de uma **String** em partes ou ‘tokens’. Os separadores por defeito são o espaço, ‘tab’, ‘newline’ e ‘carriage return’. Esta classe pertence à ‘package’ **java.util**, o que implica que deverá colocar a linha “`import java.util.*;`” no início do seu programa quando quiser utilizar esta classe. Para usar esta classe temos que primeiro criar um objecto. Por exemplo, a instrução

StringTokenizer componentes = new **StringTokenizer** (frase);

cria um objecto de nome **componentes** que irá conter as palavras ou ‘tokens’ da *string* **frase**.

Alguns métodos da classe StringTokenizer que permitem aceder aos ‘tokens’, encontram-se na tabela seguinte.

Método	Funcionalidade
public int countTokens ()	Devolve o número de 'tokens' que existem num objectos da classe StringTokenizer.
public boolean hasMoreElements ()	Devolve *true* se ainda existirem 'tokens' num objecto da classe StringTokenizer; *false* no caso contrário.
public String nextToken ()	Devolve o próximo 'token' de um objecto da classe StringTokenizer.

Exercícios

4. 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"

5. ISBN (pode ser avaliado)

A maior parte dos livros publicados actualmente 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 seleccionado de modo a 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 correcto 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 correcto pois 220 é divisível por 11.

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