Aula Prática 11

Resumo:
- Dicionários;
- Listas de pares chave-valor.

Exercício 11.1

Considere o problema de autenticar utilizadores que pretendam aceder a um determinado programa ou sistema informático. A autenticação baseia-se no nome e senha do utilizador. Usando a classe `KeyValueList` do pacote `p2utils`, desenvolva um programa que leia os nomes e senhas contidos num ficheiro dado na linha de comando e em seguida simule o processo de autenticação de utilizadores. O programa deve terminar quando for detectado "fim de ficheiro"(`EOF`), o que numa consola Unix se faz introduzindo Ctrl+D no início de uma linha.

```
Login:      carlos
Password:   minhasenha
Authentication successful

Login:      ines.m
Password:   ines$m
Authentication failed

Login:      jose.antunes
Password:   senutna
Authentication successful

Login:      <Ctrl-d>
```

Exercício 11.2

Construa um programa que determine e apresente a tabela de frequências das palavras existentes em vários ficheiros de texto cujos nomes são dados pela linha de comando. A tabela deve mostrar o número de ocorrências de cada palavra (frequência absoluta) bem como a respetiva percentagem de ocorrência (frequência relativa). O processo deve ser

independente de as palavras estarem em maiúsculas ou minúsculas [1]. Caracteres não alfanuméricos devem ser ignorados [2]. Utilize a classe `KeyValueList` na resolução deste problema. Neste caso, cada palavra será uma chave de acesso e o elemento associado será simplesmente o contador de ocorrências dessa palavra.

Exercício 11.3

Introduza as seguintes melhorias à classe `KeyValueList`:

- O acesso a cada valor ou elemento na lista de pares chave-valor tem complexidade linear no número de elementos. Assim, quanto maior o número de elementos, mais crítico se torna optimizar o acesso. Uma optimização possível consiste em manter a lista ordenada por chave. Assim, modifique o método `set` de forma a garantir que a lista está sempre ordenada por chave.

- Modifique também o método `contains` de forma a tirar partido do facto de a lista estar sempre ordenada por chave. Só precisa de percorrer a lista até encontrar a chave procurada (caso em que retorna `true`) ou uma chave maior (caso em que retorna `false`).

- Implemente também um método `toString` que devolva uma representação da lista em cadeia de caracteres de acordo com o seguinte formato: $\{(k_1, e_1), ..., (k_n, e_n)\}$.

[1] Pode usar a função `toLowerCase()` da classe `String` para converter as palavras para minúsculas antes de as introduzir na lista.

[2] Pode usar a função `split()` da classe `String` para separar cada linha de texto em palavras, tratando quaisquer caracteres não alfanuméricos como separadores: split ("[^\\p{Alnum}]+").