Aula Prática 11

Resumo:
- Aplicações de dicionários;
- Dicionários implementados como 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

O programa `P112` determina e apresenta a tabela de frequências de ocorrência das palavras contidas num ou mais ficheiros de texto cujos nomes são dados na linha de comando. O programa deve usar uma `KeyValueList` para resolver o problema. Cada palavra será uma chave de acesso e o valor associado será simplesmente o contador de ocorrências dessa palavra.

a. Complete a função principal para atualizar o contador correspondente a cada palavra processada. Para tornar o processo insensível a diferenças entre maiúsculas e minúsculas, pode usar a função `toLowerCase()` da classe `String` para converter as palavras para minúsculas antes de as introduzir na lista.

b. Complete o método `toString(start, mid, end)` da classe `KeyValueList` que deve devolver uma representação da lista como uma sequência de pares *(chave, valor)* separados entre si pela string `mid` e delimitada pelas strings `start` e `end`. Verifique o resultado das invocações no programa.

c. Complete a função `mostFrequent` para descobrir a palavra mais frequente e indicar a sua frequência relativa.

Exercício 11.3

O acesso a cada elemento numa `KeyValueList` 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. Crie uma classe `ImprovedKeyValueList`, com as seguintes melhorias em relação à classe `KeyValueList`:

- 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 da ordenação dos elementos. Assim, só precisa de percorrer a lista até encontrar a chave procurada (caso em que retorna `true`) ou até encontrar uma chave maior (caso em que retorna `false`).

Faça uma cópia do programa `P112.java` e modifique-o de forma a usar e testar a nova classe.