Universidade de Aveiro, DETI

Padrões e Desenho de Software

Guião das aulas práticas

LEI – Licenciatura em Engenharia Informática

Ano: 2018/2019

Lab I.

Objetivos

Os objetivos deste trabalho são:

- Rever e aplicar conceitos de programação adquiridos anteriormente: arrays bidimensionais, genéricos, ciclos for-each, tipos enumerados.
- Rever e praticar técnicas de desenvolvimento de software: implementar uma especificação de classe, programa com múltiplos componentes, e ficheiros JAR

1.01 Word Search Solver

O objetivo deste trabalho é escrever um programa em JAVA para resolver *Sopas de Letras*. A entrada do programa é um único ficheiro de texto contendo o puzzle e as palavras a encontrar. Exemplo (poderá pesquisar outros online):

```
QLXRXBXCLH
YHQTOPJQUJ
VFMHSNZZZL
DWPPEUEUQK
MKCATSILTC
LFTRTRVLRX
RNHGKMEQUM
VODQBYCEKQ
CWVFKCTKVU
WUXIMOZESS
list, set,
graph, stack, queue, tree
```

A saída é a lista de palavras, bem como a posição em que se encontram no puzzle.

(a) Requisitos de Entrada

O programa deve verificar se:

- O puzzle é sempre quadrado, com o tamanho máximo de 60x60.
- As letras do puzzle estão em maiúscula.
- Na lista, as palavras podem estar só em minúsculas, ou misturadas.
- As palavras são compostas por caracteres alfabéticos.
- As palavras têm de ter pelo menos 3 caracteres.
- A lista de palavras pode conter linhas em branco.
- Cada linha pode ter mais do que uma palavra, separadas por vírgula, espaço ou ponto e vírgula.
- Todas as palavras da lista têm de estar no puzzle e apenas uma vez.
- A lista de palavras não pode conter palavras duplicadas ou frases redundantes (por exemplo, não pode conter BAG e RUTABAGA ao mesmo tempo).

(b) Requisitos de Saída

A lista de palavras do puzzle retornadas pelo WSSolver tem de estar na mesma ordem das palavras passadas ao construtor. As palavras têm de estar em maiúsculas.

(c) Exemplo de Execução

Exemplo de execução com os dados anteriores:

```
$java WSSolver sdl_01.txt
LIST            4       5,8     left
SET             3       3,5     down
GRAPH           5       7,4     up
STACK           5       5,6     left
QUEUE           5       4,9     left
TREE            4       5,5     downright

$java WSSolver –timing sdl_01.txt
Elapsed time (secs): 0.047
LIST            4       5,8     left
SET             3       3,5     down
GRAPH           5       7,4     up
STACK           5       5,6     left
QUEUE           5       4,9     left
TREE            4       5,5     downright
```

O tempo é medido entre a leitura do ficheiro e a listagem de resultados.

1.02 Word Search Generator

Escreva o programa WSGenerator, que cria automaticamente qualquer *Sopa de Letras* de acordo com o formato apresentado no exercício anterior, e assumindo os mesmos requisitos de entrada.

(a) Exemplo de Execução

Assumindo que o ficheiro "*wordlist_1.txt"* contém a lista de palavras (uma por linha, ou uma lista por linha)

```
$java WSGenerator wordlist_1.txt
QLXRXBXCLH
YHQTOPJQUJ
VFMHSNZZZL
DWPPEUEUQK
MKCATSILTC
LFTRTRVLRX
RNHGKMEQUM
VODQBYCEKQ
CWVFKCIKVU
WUXIMOZESS
list, set
graph, stack, queue, tree

>java WSGenerator wordlist_1.txt sdl_01.txt
```

O resultado é o mesmo do anterior, mas guardado no ficheiro "*sdl_01.txt*".

Nota importante: para cada aula prática, deverá ser usada no *git* uma nomenclatura uniforme (*aula1*, *aula2*, *aula3*, …) para permitir uma identificação mais fácil dos projetos.

Bom trabalho!