Ciências ULisboa | Informática

Introdução à Programação

Licenciatura em Engenharia Informática

Trabalho: 2ª parte

2020/2021

Sopa-de-Letras

O trabalho de programação que vos é proposto em IP é em torno do jogo *Sopa-de-Letras*. Cada partida de *Sopa-de-Letras* é jogada numa quadrícula como a da figura, onde estão "escondidas" várias palavras. Em cada jogada, o jogador seleciona um conjunto de posições contíguas da quadrícula. Se as letras nessas posições formam uma das palavras escondidas, essa palavra é considerada encontrada. No exemplo ao lado, podemos ver que já foram encontradas três palavras. O jogo pode ser jogado com tempo limitado ou sem tempo, caso em que apenas termina quando todas as palavras escondidas foram encontradas.

A 1ªfase do trabalho serviu para se familiarizarem com o jogo. Na 2ª fase, o objetivo é implementar as principais funcionalidades do jogo, como se descreve abaixo.

Em que consiste o trabalho, afinal?

Nesta fase vamos considerar que a quadrícula está preenchida com letras e que temos de procurar certas palavras que se encontram escondidas nas linhas e nas colunas da quadrícula. A quadrícula é representada internamente no programa por uma matriz de caracteres enquanto que as palavras escondidas são representadas por *Strings*. Uma jogada, que consiste num conjunto de posições contíguas da quadrícula todas na mesma linha ou coluna, é representado por um vetor com quatro inteiros: linha e coluna da posição inicial e linha e coluna da posição final. Por exemplo, a representação de uma das jogadas feitas no jogo que é mostrado na figura é `[14,6,14,14]` e as letras nestas posições (na ordem inversa) formam uma das palavras escondidas —JULGADORA— pelo que essa palavra está assinalada como "encontrada".

A vossa tarefa é desenvolver código Java que permita a um jogador jogar uma partida de *Sopa-de-Letras*. O programa deve ser capaz de: obter um puzzle, constituído por uma quadrícula e a lista de palavras que estão escondidas, e validar que este é válido; ler, validar e registar as jogadas.

Mais concretamente, devem implementar uma classe `WordSearch` que inclua:

- Uma função `boolean isHidden(char[][] board, String word)` que, assumindo que `board` é uma matriz e `word!=null`, verifica se `word` está escondida em alguma linha ou coluna de `board`, na ordem normal ou invertida.
- Uma função `boolean isValidGame(char[][] board, String[] hiddenWords)` que, assumindo que `board` é uma matriz e `hiddenWords!=null`, verifica se para todo `0≤i<hiddenWords.length`, a palavra `hiddenWords[i]` está escondida em `board`, isto é, `isHidden(board, hiddenWords[i])`.
- Uma função `boolean isValidMove(int[] move, int rows, int columns)` que assumindo que `move!=null, rows>0` e `columns>0`, verifica se `move` tem quatro elementos e são efetivamente

linhas e colunas da quadrícula de dimensão `rows` x `columns` e definem posições contíguas todas na mesma linha ou na mesma coluna, representadas da esquerda para a direita se estiverem na mesma linha e de cima para baixo se estiverem na mesma coluna.

- Um procedimento `int[] readMove(Scanner sc, int rows, int columns)`, que lê uma jogada através do canal dado. Se o que é lido for uma jogada válida para uma quadrícula de dimensão `rows` x `columns`, então o procedimento retorna um vetor com a jogada; caso contrário, deve ser enviada uma mensagem de erro para o ecrã e lida de novo a jogada.
  Assumindo que `sc!=null, rows > 0` e `columns > 0`, o procedimento assegura que o vetor retornado é uma jogada válida, i.e., `isValidMove(\result, rows, columns)`.
- Uma função `String findWord(char[][] board, int[] move, String[] hiddenWords)` que verifica se alguma das palavras escondidas foi encontrada na jogada `move`. A função retorna essa palavra ou `null` se não tiver sido encontrada nenhuma palavra. A função deve assumir que `board` é uma matriz, `isValidGame(char[][] board, String[] hiddenWords)`, e `isValidMove(move, board.length, board[0].length)`.
- Um procedimento `void printPuzzle(char[][] board, String[] hiddenWords)` que deve imprimir a quadrícula e o número de palavras que ele tem escondidas. A função deve assumir que `board` é uma matriz e `hiddenWords!=null`.

Adicionalmente, a classe `WordSearch` deve ser programada de forma a ser possível executar o programa da seguinte forma

```
$ java WordSearch <pathToTextFileWithPuzzle>
```

onde `<pathToTextFileWithPuzzle>` é o caminho para um ficheiro de texto com o puzzle num formato apropriado para ser lido recorrendo à classe `PuzzleReader` fornecida. Recorrendo ao construtor e métodos desta classe, às funções e procedimentos listados anteriormente e eventualmente a outros que considere apropriados definir, a implementação do método `main` da classe `WordSearch` deve:

1. obter o puzzle do ficheiro dado
2. verificar que o puzzle lido é válido
3. se for inválido o programa deve terminar com uma mensagem apropriada
4. caso contrário, deve imprimir a quadrícula e o número de palavras que tem escondidas
5. até todas as palavras escondidas no puzzle terem sido adivinhadas,
   - deve ler, validar e registar cada jogada;
   - deve imprimir as palavras encontradas e o número de palavras ainda escondidas;
6. quando o jogo terminar, deve imprimir uma mensagem apropriada.

Utilizando o ficheiro `Puzzle.txt` fornecido, a execução do programa deve ser semelhante ao exemplificado abaixo, onde o texto introduzido pelo utilizador está representado a castanho e negrito.

```
$ java WordSearch Puzzle.txt
    1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
 1  V  J  F  P  I  Q  H  O  Z  P  Y  E  X  X  X
 2  Y  S  J  R  E  N  I  D  O  S  V  V  E  W  N
 3  O  X  F  O  B  I  F  J  K  W  F  M  K  H  V
 4  V  X  P  G  V  M  C  O  N  X  F  O  O  M  J
 5  N  M  P  R  T  G  R  X  R  F  V  I  B  R  A
 6  K  F  K  A  O  B  Y  O  M  M  H  F  E  B  X
 7  V  Y  L  M  M  Z  A  D  Z  Y  A  X  N  F  J
 8  Q  M  U  A  K  K  Q  H  M  X  F  T  G  Y  W
 9  R  N  G  C  A  V  A  J  R  F  X  G  E  Q  T
10  K  U  A  A  Y  F  N  Q  W  L  A  Q  N  C  G
11  A  P  D  O  U  D  N  T  Q  E  M  T  H  O  A
12  B  I  A  H  S  I  J  P  N  A  X  I  A  Z  Y
13  V  Q  A  U  L  N  O  G  G  J  F  H  R  P  C
14  A  C  I  T  A  M  R  O  F  N  I  G  I  X  F
15  S  T  Y  X  D  N  C  O  Q  O  H  Q  A  I  U
16  D  G  D  Q  H  X  L  R  G  R  J  D  I  P  S
17  G  W  N  A  L  O  R  D  O  S  B  U  R  I  L
18  Z  E  R  W  W  V  L  V  D  B  H  F  L  U  V
19  G  I  O  H  M  H  N  C  E  E  K  E  W  K  U
20  A  W  L  D  S  F  N  X  R  O  V  A  H  J  W
```

```
Hidden words: 4

Give your move: 9 5 9 8
Found words: JAVA
Hidden words: 3

Give your move: 10 5 10 18
Your move is invalid.

Give your move: 6 13 15 13
Found words: JAVA ENGENHARIA
Hidden words: 2

Give your move: 11 4 1 4
Your move is invalid.

Give your move: 1 4 11 4
Found words: JAVA ENGENHARIA PROGRAMACAO
Hidden words: 1

Give your move: 14 1 14 11
Found words: JAVA ENGENHARIA PROGRAMACAO INFORMATICA
Hidden words: 0

Good work. All hidden words were found.
```

O que entrego?

O ficheiro **`WordSearch.java`** com a solução. Não há relatório a entregar porque o vosso software é a vossa documentação. Assim, não se esqueçam de comentar condignamente a vossa classe. Devem incluir no início da classe um cabeçalho *javadoc* com `@author` (nome e número dos alunos que compõem o grupo). Para cada procedimento/função definidos há que preparar um cabeçalho incluindo a sua descrição, e, se for caso disso, `@param`, `@requires`, `@ensures` e `@return`. Apresentem um texto "limpinho", que siga as normas de codificação em Java, bem alinhado e com um número de colunas adequado. Consultem, na página da disciplina, o *Guia de Estilo Java para IP*.

Como entrego o trabalho?

Um dos alunos do grupo entrega o trabalho através da ligação que, para o efeito, existe na página da disciplina no *moodle*. O prazo de entrega é dia 13 de Dezembro às 23h55.

Quanto vale o trabalho?

Esta 2ª parte do trabalho é cotada para 7.5 valores e irá somar às notas das outras duas partes.