Ciências ULisboa | Informática

Introdução à Programação

Licenciatura em Engenharia Informática

Trabalho: 3ª parte

2020/2021

Sopa-de-Letras

O trabalho de programação que vos é proposto em IP é em torno do jogo *Sopa-de-Letras* (em inglês, *Word Search*). 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 várias palavras. O jogo pode ser jogado com tempo limitado ou sem tempo, caso em que apenas termina quando todas as palavras escondidas foram encontradas.

Na 2ª fase, implementaram as funcionalidades que permitiam jogar uma partida de *Sopa-de-Letras* com palavras escondidas apenas em linhas e colunas. Nesta 3ª fase as palavras também podem estar escondidas nas diagonais (que podem tanto ir para a direita, como no exemplo da figura, como para a esquerda). A cossa tarefa é desenvolver classes para uma aplicação que permite um jogador jogar partidas de *Sopa-de-Letras* com tempo limitado e com pontuação.

Em que consiste o trabalho, afinal?

A vossa tarefa é programar um tipo enumerado `Direction` cujos valores representem os quatro tipos de direções das jogadas do *WordSearch* —`HORIZONTAL`,`VERTICAL`,`DIAGONAL_RIGHT`,`DIAGONAL_LEFT`— e três classes Java:

- `Move`, uma classe cujos objetos representam jogadas de *WordSearch* para partidas com quadrículas de determinadas dimensões
- `Puzzle`, uma classe cujos objetos representam *puzzles* de partidas de *WordSearch*
- `WordSearch`, uma classe cujos objetos representam partidas de *WordSearch*

A API oferecida por cada uma destas classes é detalhada de seguida.

`Move`. Os objetos deste tipo representam jogadas válidas para quadrículas de uma determinada dimensão. Uma jogada define um conjunto de posições contíguas da quadrícula que vão desde a posição inicial *pos1* até à posição final *pos2*. Uma posição *pos* na quadrícula é definida por um par *(row,col)* com o número da linha e o número da coluna. Os números das linhas e colunas começam em 1. A classe deve incluir:

- `public static boolean definesMove(int row1, int col1, int row2, int col2, int rows, int columns)` que verifica se `row1,col1,row2,col2` são efetivamente linhas e colunas da quadrícula de dimensão `rows` x `columns` e definem posições contíguas de uma linha, coluna ou diagonal. No caso de posições contíguas de uma linha, a posição inicial está à esquerda da final; no caso de posições contíguas de uma coluna ou diagonal, a posição inicial está acima da posição final.

- `public Move(int row1, int col1, int row2, int col2, int rows, int columns)` que, assumindo que `definesMove(row1, col1, row2, col2, rows, columns)` constrói a jogada com os dados fornecidos.
- `public int startRow()/startColumn()` que devolve a linha/coluna da posição inicial da jogada.
- `public int endRow()/endColumn()` que devolve a linha/coluna da posição final da jogada.
- `public Direction direction()` que devolve a direção da jogada.
- `public int rows()/columns()` que devolve o número de linhas/colunas.

`Puzzle`. Um objeto deste tipo representa um *puzzle* de uma partida de *WordSearch* e é constituído por uma quadrícula, um conjunto de palavras escondidas nas linhas, colunas e diagonais da quadrícula. A classe deve incluir:

- `private static boolean isHidden(char[][] board, String word)` que, assumindo que `board` é uma matriz e `word!=null`, verifica se `word` está escondida em alguma linha, coluna, diagonal direita ou diagonal esquerda de `board`, na ordem normal ou invertida.
- `public static boolean definesPuzzle(char[][] board, String[] hiddenWords)` que, assumindo que `board` é uma matriz e `hiddenWords!=null`, verifica se `hiddenWords.length>0`, `hiddenWords` não tem palavras repetidas, e, para todo `0≤i<hiddenWords.length`, a palavra `hiddenWords[i]` tem pelo menos um caracter e está escondida numa linha, coluna ou diagonal de `board`.
- `public Puzzle(char[][] board, String[] hiddenWords)` que, assumindo que `definesPuzzle(board, hiddenWords)`, constrói um *puzzle* com dados fornecidos.
- `public int rows()/columns()` que devolve o número de linhas/colunas da quadrícula.
- `public int numberHiddenWords()` que indica o número de palavras escondidas.
- `public char[][] board()` que devolve uma cópia independente da quadrícula.
- `public String getWord(Move move)` que, se retorna a palavra escondida nas posições definidas por `move` se for esse o caso, senão retorna `null`. Este método assume que `move!=null`, `move.rows()==rows()` e `move.columns()==columns()`.

`WordSearch`. Um objeto deste tipo representa uma partida do jogo com o *puzzle* e a duração máxima, dados em tempo de construção. Estes objetos registam as palavras já encontradas e a pontuação corrente, cujo cálculo é feito da seguinte forma: uma palavra que é encontrada após terem decorrido `t` segundos — desde a última palavra encontrada ou desde o início, se for a primeira— vale o seguinte número de pontos:

- `wordPoints`, se `t≥meanTime`
- `(1+meanTime-t)*wordPoints`, se `t<meanTime`

onde `meanTime` é `duration()/puzzle().numberHiddenWords()` e `wordPoints` é `puzzle.rows()* puzzle().columns()/10`. Sugestão: podem usar o método `static long currentTimeMillis()` da classe `java.lang.System` para obter o tempo corrente em milissegundos[1]. Se registarem este tempo de cada vez que for encontrada uma palavra, vão conseguir calcular os segundos que decorreram desde a última palavra encontrada.

A classe deve incluir:

- um construtor `public WordSearch(Puzzle puzzle, int durationInSeconds)` que, assumindo que `puzzle!=null`, `durationInSeconds>0` e `durationInSeconds/puzzle.numberHiddenWords()>5`, constrói uma partida com os dados fornecidos no estado inicial.
- `public Puzzle puzzle()` que devolve o puzzle da partida.
- `public int duration()` que devolve a duração máxima da partida em segundos.
- `public int howManyFoundWords()` que devolve o número de palavras já encontradas.
- `public String[] foundWords()` que devolve um vetor com as palavras já encontradas de tamanho `howManyFoundWords()`.

[1] https://docs.oracle.com/javase/8/docs/api/java/lang/System.html

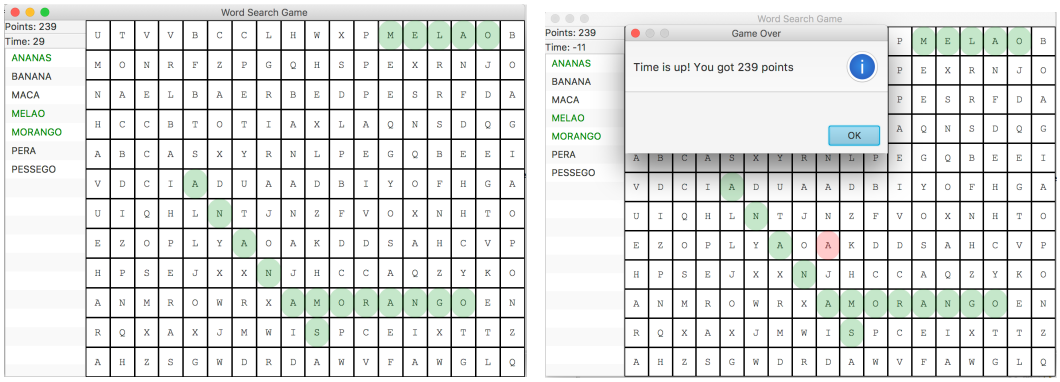
- `public int score()` que devolve a pontuação corrente.
- `public boolean isFinished()` que indica se a partida está terminada.
- `public boolean play(Move move)` que se ainda não foi ultrapassado o tempo limite, regista a jogada no estado da partida e indica se havia uma palavra escondida nas posições indicadas, caso contrário termina a partida e retorna falso. A partida é adicionalmente terminada se já não há palavras por encontrar. Este método assume que `move!=null`, `move.rows()==puzzle().rows()`, `move.columns() == puzzle().columns()` e `!isFinished()`.
- `public String toString()` que dá uma representação textual do estado da partida.

Como posso testar o meu código?

Pode exercitar o seu código através da execução de um programa com uma interface gráfica que é fornecida. Para tal, precisa apenas de colocar as suas classes juntamente com as que são fornecidas e mandar executar a classe `WordSearchGame`[2] com

```
$ java WordSearchGame <pathToTextFileWithPuzzle>
```

onde `<pathToTextFileWithPuzzle>` representa um caminho para um ficheiro de texto com o puzzle num formato apropriado para ser lido recorrendo à classe `PuzzleReader` também fornecida. A interface gráfica, como ilustrado abaixo, permite ao jogador jogar uma partida de 2 minutos com o puzzle fornecido. A escolha das jogadas faz-se selecionando a posição inicial da grelha e arrastando o cursor até à posição final. A passagem do tempo mostrada na interface é apenas indicativa: o fim do jogo ocorre apenas quando todas as palavras escondidas foram encontradas ou quando é feita uma jogada já para lá do tempo limite.



Tarefas Adicionais (Opcionais)

As tarefas adicionais, opcionais e que permitem aceder a valores de bónus, são:

- Uma classe `PuzzleGenerator` cujos objetos são geradores aleatórios de objetos `Puzzle`. A classe deve incluir:
  - `public PuzzleGenerator()` que cria um gerador de puzzles.
  - `public Puzzle nextPuzzle()` que devolve um novo puzzle gerado aleatoriamente.

  Notem que podem, por exemplo, gerar *puzzles* sempre com as mesmas palavras, fazendo apenas variar a grelha, nomeadamente os sítios em que as palavras estão escondidas.
- Uma classe `WordSearchGameTxt` com uma versão textual da interface do jogo que permite jogar várias partidas e é executada, como a interface gráfica, com o caminho para o ficheiro com o puzzle como argumento. No caso de ter feito o `PuzzleGenerator` então deve ser possível executar o programa `WordSearchGameTxt` sem nenhum argumento, sendo criadas partidas com *puzzles* gerados aleatoriamente, até o utilizador indicar que não quer jogar mais.

O que entrego?

- Os ficheiros `Direction.java`, `Move.java`, `Puzzle.java` e `WordSearch.java` e, no caso de ter escolhido fazer a tarefa adicional, `WordSearchGameTxt.java` e/ou `PuzzleGenerator.java`. Não há relatório a entregar porque o vosso software é a vossa documentação. Assim, não se esqueçam

[2] Se a sua versão do Java 8 for OpenJDK precisa de instalar adicionalmente o OpenJFX.

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.

- Um ficheiro *Declaracao.pdf* com uma declaração a dizer que o conteúdo do que estão a entregar é exclusivamente da vossa autoria. Esta declaração deve ser assinada pelos dois membros do grupo. Pode ser assinado com uma assinatura manual digitalizada ou assinado digitalmente com Cartão de Cidadão ou Chave Móvel Digital.

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 12 de janeiro de 2021 às 20h.

Quanto vale o trabalho?

Esta 3ª parte do trabalho é cotada para 7.5 valores. A tarefa adicional é cotada para 2.5 valores. A nota final do trabalho será dada por `min(notaFase1 + notaFase2 + notaFase3 + notaTarefaAdc, 20)`.