Ciências ULisboa | Informática

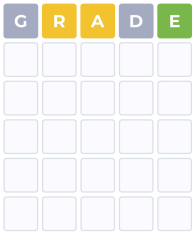
Introdução à Programação

Licenciatura em Engenharia Informática

Trabalho: 1ª parte

2023/2024

Ipurdle



O trabalho de programação que vos é proposto em IP é sobre uma variante do jogo do *Wordle* batizada de *Ipurdle*. Cada partida deste jogo é jogada numa quadrícula como mostrada na figura ao lado e com um dicionário de palavras definido pela equipa docente de IP (no exemplo, palavras de 5 letras). O objetivo do jogador é, a partir da pista recebida para cada palavra jogada, adivinhar a palavra. A pista para cada jogada consiste em sinalizar quais as letras da palavra jogada que se encontram na posição certa (a verde no exemplo), as letras que pertencem à palavra, mas estão numa posição errada (a amarelo) e as letras que não pertencem à palavra (a cinzento). O jogo termina quando a palavra é revelada ou após ter sido atingido o número máximo de tentativas (6, no exemplo).

No *Ipurdle*, e ao contrário do que acontece no *Wordle*, a palavra a adivinhar não está definida à partida. À medida que o jogo progride vão sendo escolhidas as pistas a dar a cada jogada de forma a tornar a descoberta da palavra mais difícil.

Em que consiste o trabalho, afinal?

A vossa tarefa é desenvolver código Java que permita jogar o *Ipurdle*, com uma interface textual básica. A vossa implementação está sujeita a um conjunto de restrições descritas de seguida, que tem como objetivo vos guiar no desenvolvimento da solução e ao mesmo tempo garantir que exercitam os conceitos já lecionados.

A implementação pretendida:

- Usa *Strings* constituídas por letras maiúsculas para representar as palavras.
- Usa números inteiros com os algarismos 3, 2 e 1 para representar as pistas: o algarismo 3 corresponde à letra na posição certa, o 2 à letra na posição errada e 1 à letra que não pertence à palavra. Por exemplo, o número 12213 representa a pista da primeira jogada da figura acima.

  Esta representação e a relação de < entre inteiros induz uma ordem total entre todas as pistas. No caso de palavras de tamanho 5, a pista menor de todas é 11111 e a maior é 33333 e temos 11111 < 11112 < 11113 < 11121 < 11122 < ... < 33331 < 33332 < 33333.
- Usa duas classes fornecidas: 1) `DictionaryIP`, cujos objetos mutáveis representam dicionários de palavras válidas de um determinado tamanho (dado na construção); 2) `StringColouring`, com várias funções que permitem obter *Strings* coloridas. Deve consultar a documentação destas classes para ver que funcionalidades oferecem.
- Só usa o que está disponível na versão 8 do Java e que tenha sido lecionado até à aula teórica 12 (consultar os sumários e os docentes em caso de dúvida).

Mais concretamente, devem implementar uma classe **`Ipurdle`** que inclua:

- A definição de uma função chamada `validClue` que, dados dois números inteiros `clue` e `size`, e assumindo que `size` é um número maior que zero, verifica se `clue` representa uma pista para uma

palavra com o tamanho `size`. Note que isto corresponde a verificar as seguintes condições:

`clue` é positivo, `clue` tem `size` dígitos, `clue` é composto apenas pelos dígitos `1`, `2` e `3`

e a retornar `true` se todas as condições se verificarem e `false` caso contrário.

- A definição de uma função chamada `minClue` que, dado um número inteiro `size`, e assumindo que `size` é um número maior que zero, retorna a menor pista para palavras desse tamanho.
- A definição de uma função chamada `isMaxClue` que, dados dois números inteiros `clue` e `size`, e assumindo que `size` é um número maior que zero e `clue` representa uma pista para palavras de tamanho `size`, verifica se `clue` é a maior pista para palavras desse tamanho.
- A definição de uma função chamada `nextClue` que, dados dois números inteiros `clue` e `size`, e assumindo que

  `size` é um número maior que zero, `clue` representa uma pista para palavras de tamanho `size` e `clue` não é a maior pista para palavras de tamanho `size`

  calcula o número que representa a pista imediatamente a seguir, ou seja, o menor número inteiro maior que `clue` que representa uma pista para palavras de tamanho `size`.
- A definição de um procedimento chamado `printClue` que, dada uma *String* `guess` que se assume não ser *null* e um número inteiro `clue` que se assume representar uma pista para `guess`, imprime `guess` com as suas letras coloridas de acordo com a `clue`. Devem ser coloridas a verde as letras que na pista têm 3, a amarelo as letras que na pista têm 2 e a preto as letras que na pista têm 1.
- A definição de uma função chamada `clueForGuessAndWord` que, dadas duas *Strings* `guess` e `word`, que se assumem ter o mesmo tamanho, retorna o inteiro que representa a pista a dar à jogada `guess` se a palavra a adivinhar for `word`.

  No caso de uma letra de `word`, que só ocorre uma vez nesta palavra, estar em várias posições erradas de `guess`, apenas a letra na posição mais à esquerda é identificada como letra certa na posição errada.
- A definição de uma função chamada `howManyWordsWithClue` que, dado um objeto do tipo `DictionaryIP` que se assume não ser *null,* um número inteiro `clue` que se assume representar uma pista para palavras desse dicionário e uma *String* `guess` que se assume ter o tamanho certo, retorna o número de palavras válidas do dicionário que se fossem a palavra a adivinhar, dariam origem à pista `clue` para `guess`.
- A definição de uma função chamada `betterClueForGuess` que, dado um objeto do tipo `DictionaryIP`, que se assume não ser *null* e uma *String* `guess` que se assume ter o tamanho certo, retorna o inteiro que representa a pista para `guess` que serve para mais palavras do dicionário dado. No caso de haver pistas empatadas, dentre estas, é escolhida a menor.

  Note que as pistas podem ser percorridas recorrendo às funções `minClue` e a `nextClue` descritas acima. A pista calculada diz-se que é a melhor porque é a que torna mais difícil o jogador acertar na palavra.
- A definição de um procedimento chamado `playGuess` que, dado um objeto do tipo `Dictionary` que se assume não ser *null*, e uma *String* `guess` que se assume ter o tamanho certo:
  1. calcula a melhor pista para `guess` face ao dicionário dado, recorrendo à função anterior, `betterClueForGuess`
  2. remove do dicionário todas as palavras que não resultariam nessa pista
  3. retorna a pista
- Um método `main` que deverá usar os dois procedimentos anteriores e
  1. Declarar e inicializar a variável `maxAttempts` a 6.
  2. Declarar duas variáveis `gameWordsDictionary` e `puzzlesDictionary` do tipo `Dictionary` e inicializá-las com dois novos dicionários de palavras de tamanho 5. O primeiro dicionário tem as palavras válidas, i.e., as que podem ser jogadas. O segundo dicionário tem as palavras que ainda podem ser a palavra a descobrir, e que vai mudando à medida que vão sendo feitas jogadas e sendo escolhidas as pistas a dar.

3. Obter uma palavra do utilizador (*guess*) que, de acordo com `gameWordsDictionary`, seja válida, jogar essa palavra sobre o dicionário `puzzlesDictionary` e imprimir uma versão colorida dessa palavra de acordo com a pista obtida; repetir isto enquanto o jogo não terminar.

Podem ainda incluir na vossa classe outros procedimentos ou funções que considerem úteis. É fornecida a classe **`IpurdleTest.java`** que serve para testarem individualmente os diferentes elementos que constituem a solução e que eventualmente ajuda a clarificar qual comportamento pretendido de cada um destes elementos. Devem colocar esta classe, e os *.class* fornecidos, na mesma diretoria que o ficheiro I*purdle.java*.

Exemplo ilustrativo de uma execução do programa

```
$java Ipurdle
```

```
Bem vindo ao jogo Ipurdle!
Neste jogo as palavras têm tamanho 5. O dicionário tem apenas palavras em inglês relacionadas com IP.
Tens 6 tentativas para adivinhar a palavra. Boa sorte!
Palavra a jogar? WHILE
Palavra com a pista >  WHILE
Palavra a jogar? BLANE
Palavra invalida, nao existe no dicionario.
Palavra a jogar? BLADER
Palavra invalida, tamanho errado.
Palavra a jogar? FIELD
Palavra com a pista >  FIELD
Palavra a jogar? ABOVE
Palavra com a pista >  ABOVE
Parabens, encontraste a palavra secreta!
```

Tarefa Adicional (Opcional)

A tarefa adicional, que é opcional e que permite aceder a um bónus na nota, consiste em ter o método `main` preparado para ler o tamanho das palavras e o número máximo de tentativas a partir de argumentos da linha de comandos e usar o valor 5 e 6 apenas quando estes valores não são fornecidos.

O que entrego?

O ficheiro **`Ipurdle.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 código que siga as normas de codificação em Java, bem alinhado e com um número de colunas adequado.

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 19 de Novembro às 23h.

Quanto vale o trabalho?

Esta primeira parte do trabalho é cotada para 10 valores e irá somar à nota da segunda parte. O bónus na nota correspondente à tarefa adicional é de meio valor.