Ciências ULisboa | Informática

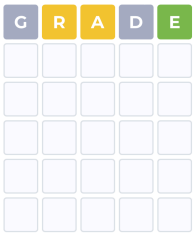
Introdução à Programação

Licenciatura em Engenharia Informática

Trabalho: 2ª 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. Concretamente, isto é conseguido escolhendo sempre a pista que deixa um conjunto maior de palavras possíveis.

Em que consiste o trabalho, afinal?

Nesta segunda fase, a vossa tarefa é desenvolver (de novo) código Java que permita jogar o *Ipurdle*. A solução pretendida, em termos de funcionamento, é em tudo igual à da primeira fase (detalhes sobre o cálculo das pistas não são, portanto, repetidos aqui). No entanto, a solução pretendida nesta fase está sujeita a um conjunto de restrições diferentes. Estas restrições têm como objetivo garantir que praticam o desenvolvimento de soluções que envolvem programar classes que definem tipos, definição e utilização de enumerados e utilização de vetores. A vossa solução vai permitir jogar o *Ipurdle* tanto com uma interface gráfica (fornecida) como com uma interface textual.

Concretamente, a vossa tarefa é programar:

- um tipo enumerado `LetterStatus` com os valores `INEXISTENT`, `WRONG_POS` e `CORRECT_POS` representando os três possíveis elementos de uma pista
- uma classe `Clue` cujos objetos representam pistas para palavras de um determinado tamanho equipadas com uma relação de ordem total (todas as pistas têm, portanto, um número de ordem; este número de ordem é o mesmo que o considerado na primeira fase do trabalho)
- uma classe `Board` cujos objetos representam o estado do tabuleiro de um jogo de *Ipurdle*
- uma classe `IpurdleGame` cujos objetos representam jogos de *Ipurdle*
- uma classe `IpurdleTxt` que implementa uma interface textual do jogo.

A API oferecida por cada uma destas classes é detalhada de seguida. Devem usar apenas o que está disponível na versão 8 do Java e que tenha sido lecionado na disciplina (consultar os sumários e os docentes em caso de dúvida).

`Clue`. Os objetos deste tipo representam pistas para palavras de um determinado tamanho. A classe deve incluir os seguintes construtores e métodos públicos:

- `public Clue(LetterStatus[] elements)`, que assumindo que `elements!=null`, constrói uma pista com os elementos dados
- `public Clue(int orderNumber, int wordSize)`, que assumindo que `wordSize>0` e $1 \leqslant$ `orderNumber`$\leq 3^{\texttt{wordSize}}$, constrói uma pista para uma palavra de tamanho `wordSize` e com o número de ordem `orderNumber`
- `public int length()`, que dá o tamanho da pista
- `public int orderNumber()`, que dá o número de ordem de uma pista
- `public LetterStatus[] letterStatus()`, que dá um vetor com os elementos da pista
- `public boolean isMax()` que indica se a pista é a maior entre todas as pistas do seu tamanho
- `public String toString()` que dá uma representação textual de uma lista onde são usados os símbolos *, o e _ para representar letra na posição correta, letra na posição errada e letra inexistente.

`Board`. Os objetos deste tipo representam o estado do tabuleiro de um jogo de *Ipurdle*. A classe deve incluir:

- `public Board(int wordSize, int maxGuesses)` que, assumindo que `wordSize≥1` e `maxGuesses≥1`, cria um tabuleiro para o jogo de *Ipurdle* com os dados fornecidos no estado inicial (vazio)
- `public int wordLength(), public int maxGuesses()` e `public int guesses()` que permitem saber o tamanho das palavras que podem ser guardadas no tabuleiro, o número máximo de tentativas e quantas já foram realizadas (que variam entre 0 e `maxGuesses()`), respetivamente
- `public void insertGuessAndClue(String guess, Clue clue)` que regista palavra e pista fornecidas, assumindo que `guess.length()==clue.length()==wordLength()` e `guesses()< maxGuesses()`
- `public String toString()` que dá uma representação textual do estado do tabuleiro como se ilustra abaixo:

```
+---------------+
| WHILE | ____* |
+---------------+
| FIELD | __o__ |
+---------------+
| ABOVE | ***** |
+---------------+
```

`IpurdleGame`. Os objetos deste tipo representam uma partida do jogo de *Ipurdle* com um dicionário de palavras fixado pela equipa docente de IP e enumeradas abaixo.

```
"JAVA", "LOOP", "EXIT", "TRUE","LONG", "THIS",
"BREAK","WHILE","GRADE","PUPIL", "FIELD", "BASIC", "ABORT",
"ABOVE","FALSE","FLOAT","SHORT","CLASS","FINAL",
"STATIC","METHOD","STRING","RETURN","RANDOM","EQUALS","OBJECT","FUNCTION",
"VARIABLE","INTEGER","SCANNER"
```

A classe deve usar:

- um vetor para representar as palavras válidas de uma partida, ou seja, as que podem ser jogadas (que nunca mudam)
- um vetor de Booleanos para representar as palavras que ainda podem ser a palavra a descobrir (que vão mudando à medida que vão sendo feitas jogadas).

A classe deve incluir:

- `public IpurdleGame(int wordSize, int maxGuesses)` que, assumindo `wordSize≥1` e `maxGuesses≥1`, cria uma partida do jogo *Ipurdle* com os dados fornecidos no estado inicial
- `public int wordLength(), public int maxGuesses()` e `public int guesses()` que permitem saber o tamanho das palavras que podem ser jogadas, o número máximo de tentativas e quantas já foram realizadas, respetivamente
- `public boolean isValid(String guess)` que, assumindo que `guess!=null`, indica se a palavra é válida, ou seja, tem o tamanho certo e pertence ao dicionário
- `public boolean isOver()` que indica se a partida já terminou, ou seja, a palavra foi descoberta ou foram esgotadas as tentativas
- `private Clue clueForGuessAndWord(String guess, String word)` que, assumindo `guess.length()==word.length()`, retorna a pista a dar a `guess` se a palavra a adivinhar for `word`
- `public Clue playGuess(String guess)` que, assumindo que `isValid(guess)` e `!isOver()`, faz a jogada (com tudo o que isso implica) devolvendo a pista para `guess` que serve para mais palavras.

Importante: este método deve percorrer o vetor das palavras válidas apenas duas vezes!

- `public String toString()` que dá uma representação textual do estado da partida como ilustrado

```
Ipurdle with words of 5 letters.
Remaining guesses: 4
       +---------------+
       | WHILE | ____* |
       +---------------+
       | FIELD | __o__ |
       +---------------+
```

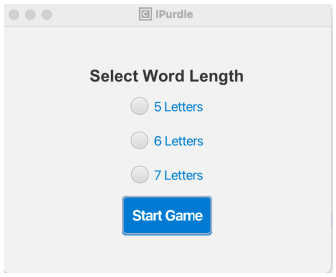
`IpurdleTxt`. Classe com um método `main` que deverá criar uma partida de *Ipurdle* com palavras de 5 letras e 6 tentativas. O método deve suportar a interação com o jogador: obter uma palavra do utilizador que seja válida, jogar essa palavra e imprimir o estado do jogo; repetir isto enquanto o jogo não terminar.

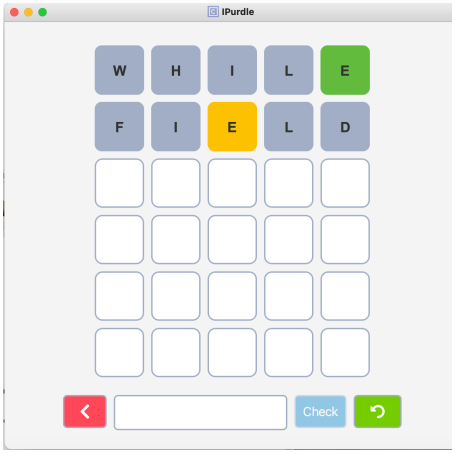
Tarefa Adicional (Opcional)

A tarefa adicional, que é opcional e que permite aceder a um bónus na nota, consiste em programar a classe `IpurdleGame` de forma a ler as palavras do dicionário de um ficheiro `Dicionario.txt` que se assume existir na mesma diretoria do código e que se assume ter uma palavra por linha.

Como posso testar o meu código?

Uma alternativa é usar a classe `IpurdleTest` fornecida, a qual exercita os vários métodos das classes a programar em alguns cenários. Pode ainda exercitar o seu código através da execução da classe `IpurdleTxt` e da classe `IpurdleGUI` fornecida. Esta última classe, juntamente com o resto do material fornecido, permite jogar o *Ipurdle* com uma interface gráfica (GUI), como ilustrado abaixo. Para tal, precisa apenas de colocar os ficheiros com o código desenvolvido juntamente com o que é fornecido (ou seja, na mesma diretoria) e nessa diretoria dar o comando `java IpurdleGUI`.





O que entrego?

Os ficheiros `LetterStatus.java`, `Clue.java`, `Board.java`, `IpurdleGame.java` e `IpurdleTxt.java`. 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 classe e método público 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 17 de Dezembro às 23h.

Quanto vale o trabalho?

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