Tarea UD 3. Clases y objetos. POO. Herencia. Arrays (III).

INTRODUCCIÓN

Wordle y ***AppGWordle***

En esta parte de la tarea necesitaremos las clases creadas en tareas anteriores: ***WordleColor****,* ***WordleLetter****.* ***WordleWord****,* ***WorleWordSizeException*** *(que serán proporcionada una vez finalice el período de entrega de la tarea anterior), pues crearemos dos clases llamadas* ***Wordle*** *(el juego) y la aplicación que hace uso del juego,* ***AppGWordle****.*

A diferencia de las otras tareas, en esta se trata de ***crear las dos clases ajustándose a la documentación del API*** *que se proporciona. La documentación del API describe algoritmos cuando se considera necesario, además de toda la interface de las clases. La documentación del API tiene dos clases más, pero no se usan o se puede prescindir de ellas para esta tarea.*

A diferencia de las otras tareas, y dada la proximidad del examen, no es obligatoria, por lo que si se tienen perfectas las anteriores se tendrá asegurado un 1 en la nota. Aun así, recomiendo realizarla, pues ayudará para el estudio del examen. Además, aquellos que la superen y hayan aprobado el examen presencial tendrán un 0,3 más en la nota de la evaluación. Se podrá hacer una videoconferencia para comprobar la autoría si se tienen dudas sobre ello.

Wordle

public final class ***Wordle***

La clase ***Wordle*** representa al juego. En ella se declara un *array* de ***WordleWord***, *wordleWords*, del tamaño del número de intentos del juego (aparecen tantas palabras como intentos) cuyos elementos al principio son todos vacíos, pues no se han escrito ningún intento.

Por ejemplo, si en el juego tenemos 6 intentos, wordleWords será un array de 6 objetos de tipo ***WordleWord****. Además de ese array, tiene una cadena como la palabra a adivinar, palabra, y los intentos (podría prescindirse de esta propiedad).*

A mayores, tiene un **atributo final**, ***dictionary***, con la lista de palabras del diccionario. Podría haber sido un atributo estático para facilitar el trabajo, pero cada juego tiene su diccionario (podríamos tener un Wordle de cine, de otros idiomas, etc.). También dispone del nombre del archivo en dónde están almacenadas las palabras del juego: ***wordsFile***

ATRIBUTOS

- ***intentos***: de tipo entero, representan los intentos del juego, que coinciden con el tamaño del array de *WordleWord*.
- ***palabra***: de tipo cadena de caracteres, la palabra a adivinar.
- ***wordleWords***: array de tipo *WordleWord* que guarda cada uno de los intentos. Uno por fila.
- ***dictionary***: atributo final (cambian sus valores, no el objeto, por eso se declara final) de tipo *ArrayList<String>*, con la lista de palabras válidas del juego. Si no se sabe o se desea, puede hacerse con un array u otra lista, escribiendo las palabras a mano en el código para poder avanzar y realizar pruebas del programa. No se penaliza en la evaluación.
- ***wordsFile***, nombre del archivo del diccionario. Los diccionarios se guardan en archivos de texto dentro del directorio DICTIONARY_DIRECTORY, que es "/diccionarios/" y se cargan en el *dictionary* la primera vez que se invoca al método *getRandomPalabra()*.

CONSTANTES (STATIC FINAL)

- ***RANDOM_GENERATOR***, de tipo ***java.util.Random***, es un generador de números aleatorios. Se emplea para obtener un índice aleatorio entre la lista de palabras válidas dentro del método *getRandomPalabra()*.

- ***DICTIONARY_DIRECTORY***, cadena con el nombre del directorio en el que están los archivos de texto con los diccionarios del juego. Su valor es "/diccionarios/".

AppGWordle

public class ***AppGWordle*** *extends Object*

Aplicación del juego de ***Wordle***. Inicialmente muestra el panel del juego y va pidiendo intentos.

En el método main o entrada al programa:

1. Establece si se desea realizar con colores o no (***opcional***), viendo el número de argumentos pasados al programa (en la versión proporcionada en la segunda tarea, se incluye un atributo estático *WordleLetter.withConsoleColors* de tipo boolean, que permite indicar si lo queremos mostrar con colores o no).
2. Crea un juego de *Wordle* con el constructor por defecto.
3. Imprime el juego de Wordle.
4. Va pidiendo palabras y mostrando el Wordle mientras no supere el número de intentos o no adivine la palabra oculta.
5. La palabra debe estar en el diccionario. Si no está no cuenta el intento.

ENTREGA

Genérese un único documento **PDF con el código del programa y las capturas de compilación y ejecución.** Además, debe proporcionar el **código fuente, *Wordle.java*, *AppGWordle.java* y** el archivo ***jwordle.jar*** para poder ser ejecutado.

Entréguese un archivo comprimido de extensión ZIP o 7Z con:

- PDF con formato *Apellido1Apellido2NombreTarea0303PROG.pdf*
- Código fuente: ***Wordle.java*, *AppGWordle.java***
- Archivo comprimido para ejecutar: ***jwordle.jar***.

El formato de documento debe ser:

*Apellido1Apellido2NombreTarea0303PROG.zip (o *.7z)*

Por ejemplo:

WittgensteinLudwigJosefTarea0303PROG.zip (sólo tiene un apellido)