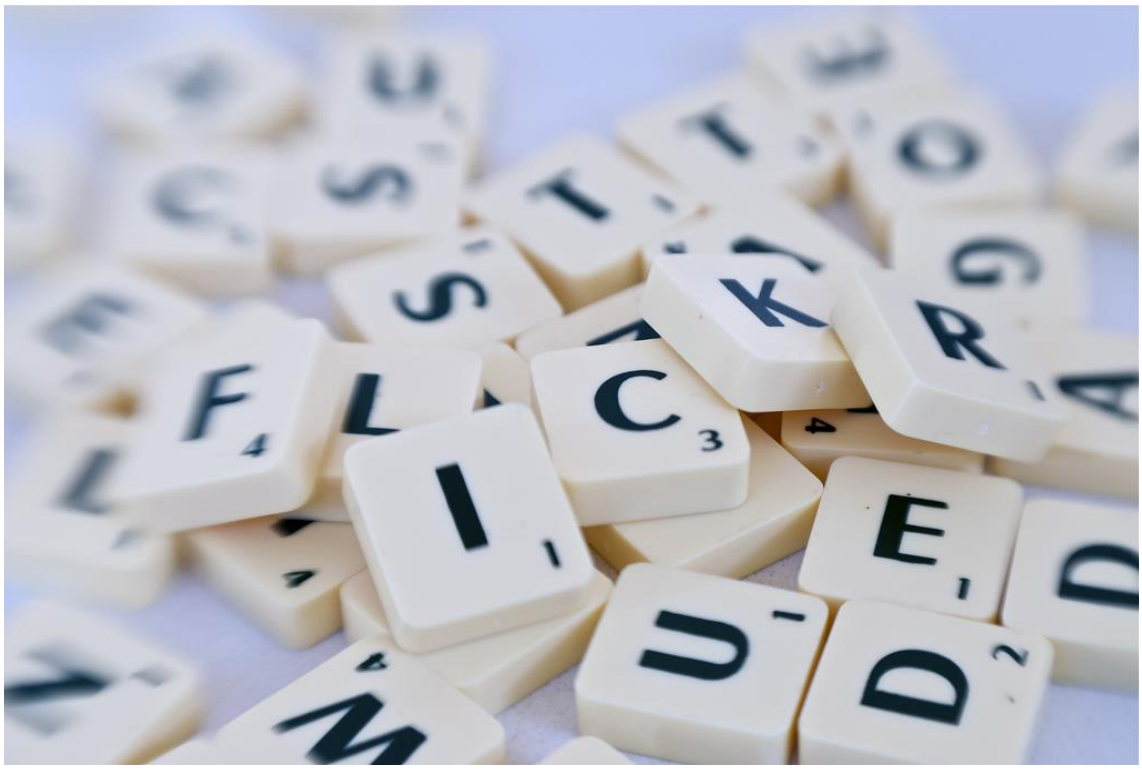
Scrabble



Nadal Llabrés Belmar

41616427-C

Índice

1. Descripción de la práctica.

La práctica consiste en crear una simplificación del juego conocido como “scrabble”. El juego no dispondrá de interfaz gráfica ya que no hemos estudiado esta parte aún.

El juego estará pensado para que un jugador juegue contra el ordenador, en cada turno se sorteará una cantidad de letras que podrá venir definida en el juego, ser aleatoria o que el propio jugador elija dicha cantidad antes de iniciar la partida. En el turno, el jugador deberá crear una palabra válida con las letras que le hayan aparecido con el objetivo de alcanzar la máxima puntuación posible. Una vez haya introducido la palabra, se le sumará la puntuación correspondiente y el turno pasará al ordenador. El juego transcurrirá así hasta que se llegue a la puntuación objetivo establecida antes de iniciar la partida. Finalmente se mostrará el ganador y finalizará la partida.

La práctica se tiene que programar con lenguaje Java basándose en la filosofía de la programación orientada a objeto. El tipo de dato String sólo se usará para entrada y salida de datos.

En la configuración inicial de la partida se podrá elegir con que idioma (Castellano, Catalán o Ingles) se querrá jugar la partida.

2. Estructuras de datos utilizadas.

Para llevar a cabo la práctica se hará uso de 4 archivos adicionales:

- lletres.txt
- dic.es
- dic.cat
- dic.en

Estos archivos han sido proporcionados por el profesorado. A continuación, se hará una breve explicación de cada uno de ellos.

2.1. lletres.txt

El primer archivo llamado “lletres.txt” es el que tendrá todas las letras del alfabeto con su puntuación. Su estructura es la siguiente:

```
1  :a:1
2  :a:1
3  :a:1
4  :a:1
5  :b:3
6  :c:3
7  :d:2
```

Así hasta llegar a la letra “z”. A la derecha de cada letra está su puntuación correspondiente. Como se puede ver en la imagen las vocales están repetidas para que haya más opciones de poder crear palabras.

2.2. dic.es, dic.ca y dic.en

```
10566  awee
10567  aweigh
10568  aweing
10569  aweless
10570  awes
10571  awesome
10572  awesomely
10573  awesomeness
10574  awesomenesses
```

Cada archivo contiene todas las palabras del diccionario de cada lengua, hay una palabra por línea. En algunos archivos están ordenadas alfabéticamente y en otros están por número de caracteres.

Para cada consulta se debe acceder a dicho archivo, no se pueden cargar en su totalidad dentro de la memoria.

3. Clases y métodos.

El proyecto ha sido distribuido en un total de 8 clases (incluida la del *main*) y 55 métodos.

Clases:

- Scrabble
- FicherosPalabrasIn
- FicherosPalabrasOut
- LT
- Letra
- Palabra
- Palabras
- Partida

A continuación, se verá la estructura y la división de las diferentes partes.

3.1. Clase principal

Nombre	Número de métodos	Número de líneas
Scrabble	5	250

3.1.1. Método *main*

En esta clase se encuentra el método *main* que está distribuida en dos grandes partes, el turno del jugador y el turno del ordenador.

La primera parte de esta clase forma parte del menú inicial de configuración donde se hace la llamada a los métodos correspondientes para ir mostrando en pantalla las diferentes opciones disponibles. Una vez que la configuración esta hecha llega la parte de los turnos.

Las dos partes de que forman los turnos (jugador y ordenador) son muy similares. Se hace la llamada a los métodos que muestran el panel de letras aleatorio, se recoge la palabra que introduce el usuario y se llama a los métodos validadores pasándoles por parámetro la palabra que ha sido introducida, se muestra la puntuación obtenida y antes de pasar al siguiente turno se visualiza un pequeño resumen de las puntuaciones de los jugadores al momento.

3.1.2. Método *cabeceraMenuInicial*

Es el encargado de imprimir por pantalla la cabecera o presentación del juego. Tiene una forma similar a la siguiente:

```
**************************************
JUEGO SCRABBLE
**************************************
```

3.1.3. Método *panelLetras*

Realiza la visualización de las letras sorteadas a través de la llamada a otros métodos y su puntuación. La salida es la siguiente:

a	b	c	d	e	f	g	h	i	j
1	1	1	2	1	4	3	7	1	6

3.1.4. Método *resumenPuntuacion*

Antes de dar por finalizado cada turno se hará una llamada a este método para ver el estado actual de las puntuaciones.

Puntuación Jugador: 7

Puntuación Ordenador 13

3.1.5. Método *mensajeDespedida*

Cuando la partida finaliza se llama a este método para visualizar un mensaje de despedida al jugador.

¡Partida finalizada! ¡Hasta pronto!

3.2. Clase Partida

Nombre	Número de métodos	Número de líneas
Partida	9	115

Recoge toda la configuración de la partida que introducirá el usuario antes de empezar y también dispone de un método encargado de realizar el sorteo del turno inicial.

Atributos:

- String nombreJugador
- int puntuacionMaxima
- int numeroDiccionario
- int turnoInicial
- String [] IDIOMA

3.2.1. Método constructor

Su trabajo es inicializar los atributos con unos valores iniciales. En este caso los valores no serán pasados por parámetro, sino que ya vienen definidos dentro de la clase.

3.2.2. Método *configuracionInicial*

Encargado de preguntar el nombre al jugador, la puntuación máxima que se jugará la partida, y el idioma. Toda esa información se almacena en los atributos de clase correspondientes.

3.2.3. Método *resumenConfiguracionInicial*

Muestra un breve resumen de todas las configuraciones que se han escogido en el método anterior.

----------Resumen----------

El nombre del jugador es: Nadal

La partida se jugará a 32 puntos.

El idioma del diccionario seleccionado es: Castellano.

3.2.4. Método *getIdioma*

Nos devuelve el String correspondiente al idioma seleccionado. Se ha usado en el método anterior para que en vez de aparecer "El idioma seleccionado es: 1." aparezca "El idioma seleccionado es: Castellano."

3.2.5. Método *sorteoTurnoInicial*

Mediante la función Math.random() se elige un número aleatorio entre 0 y 1. Si el número aleatorio es 0 se mostrará "Empezará jugando el jugador", sino "Empezará jugando el ordenador". Finalmente se devuelve dicho valor.

3.2.6. Método *getPuntuacionMaxima*

Devuelve el valor del atributo puntuacionMaxima.

3.2.7. Método *getTurnoInicial*

Devuelve el número que se ha sorteado para el turno inicial.

3.2.8. Método *getNombreJugador*

Devuelve el nombre del jugador que se ha introducido en la configuración inicial.

3.2.9. Método *getDiccionario*

Devuelve el número de diccionario seleccionado para jugar.

3.3. Clase Letra

Nombre	Número de métodos	Número de líneas
Letra	8	168

Contiene los métodos relacionados con el fichero “lletres.txt” y los que generan el panel de letras aleatorio.

Para ello se ha hecho uso de los siguientes atributos:

- int letra
- int puntuacionLetra
- int FINAL_FICHERO = -1;
- int SEPARADOR_LETRAS = 58;
- int CR = 13;
- NUMERO_LETRAS_PANEL = 14;
- Letra [] letrasFinales = new Letra[NUMERO_LETRAS_PANEL];

**: El separador de letras se corresponde al carácter “:” y el CR al Card Return (13);

3.3.1. Método constructor

Establece un valor inicial a la letra (-1) para identificarla como que aún no ha sido leído ningún valor. En la puntuación se pone como valor inicial el 0. Ninguno de esos valores se pasan por parámetro.

3.3.2. Método *contarLineasFichero*

Devuelve el número de líneas que tiene el fichero de letras para así poder establecer un tamaño al array donde cargaremos todas las letras con su puntuación. Esto nos sirve porque el fichero de letras contiene una letra por línea. El método es “private” porque sólo se usa dentro de esa misma clase.

3.3.3. Método *recogerLetrasFichero*

Encargado de leer las letras del fichero “letres.txt” y su correspondiente puntuación y almacenarlo en un array de objetos Letra. Finalmente, se devuelve ese array con todos los objetos letra con su letra y su puntuación.

3.3.4. Método *getLetrasRandom*

Escoge aleatoriamente valores de entre 0 y 26 y ese valor será la posición donde va a recoger la letra del array de letras generado por el método anterior. Cada una de ellas será almacenada dentro de otro array de objetos letra con la letra y la puntuación que le corresponde. Finalmente, todas las letras sorteadas se almacenarán dentro de otra estructura de datos llamada "letrasFinales" que es un atributo de clase. De este modo conservaremos las letras que se han sorteado en el turno para poder validar la palabra introducida.

3.3.5. Método *getLetra*

Devuelve el valor del atributo letra.

3.3.6. Método *getValor*

Devuelve el valor del atributo puntuacionLetra.

3.3.7. Método *getNumeroLetrasPanel*

Devuelve el valor del atributo NUMERO_LETRAS_PANEL.

3.3.8. Método *getLetrasFinales*

Devuelve el array de objetos letra llamado "letrasFinales", en él están ubicadas todas las letras sorteadas con su puntuación.

3.4. Clase Palabras

Nombre	Número de métodos	Número de líneas
Palabras	5	332

En esta clase están reunidos todos los métodos relacionados con la búsqueda de palabras en los diccionarios y el cálculo de la puntuación. No dispone de atributos.

3.4.1. Método *buscarEnDiccionario*

Este método recibe por parámetro un objeto Palabra que buscará en el diccionario correspondiente al que se haya elegido en la configuración inicial. Si la palabra ha sido hallada en el diccionario, se devolverá una booleana como "true" si no, como "false".

3.4.2. Método validarConLetras

Este método también recibe por parámetro un objeto Palabra que introducirá dentro de un array de tipo *char* para comprobar que cada carácter existe en las letras sorteadas.

La estrategia usada para validar la palabra es la siguiente:

1- Se carga la palabra dentro de un array de tipo *char*.

2- Se crea un array con las letras qua han sido sorteadas, otro con el alfabeto completo y por último uno dónde se contabilizarán las veces que ha aparecido cada letra en el sorteo.

3- Si la palabra introducida es mayor al número de letras sorteadas se devuelve "false" directamente.

4- Se coge la primera letra de la palabra y se busca la posición dónde corresponde la misma en el alfabeto.

5- Una vez se tiene la posición se comprueba el array donde se han contabilizado que en la misma posición haya un valor igual o mayor a 1.

6- De ser así, se sigue con la siguiente letra hasta el final. Si alguna falla, se detiene el proceso y se devuelve el valor "false" a través de una variable booleana.

7- Si el proceso continua correctamente se devolverá el valor "true".

3.4.3. Método *puntuacionPalabra*

A través de un objeto Palabra pasado por parámetro busca cada letra de dicha palabra en el array de letras del fichero y cuando la encuentra recoge su puntuación. Esta puntuación se va acumulando dentro de un acumulador hasta que se llega al final de la palabra. Se devuelve el valor del acumulador.

3.4.4. Método *puntuacionPalabraCPU*

Este método funciona igual que el anterior a excepción de que el parámetro que le llega es un array de tipo *Char* porque el método que piensa la palabra del ordenador lo devuelve con ese tipo de dato.

3.4.5. Método *palabraCPU*

Nos devolverá la palabra pensada por el ordenador. Su funcionalidad es escoger una palabra del diccionario adecuado que valide con las letras sorteadas y que tenga una longitud mínima.

La estrategia usada es la siguiente:

1- Se cargan en un array las letras que han aparecido en el sorteo.

2- Se selecciona el diccionario según el elegido por el usuario.

3- Mientras haya palabras, recorre el diccionario hasta que encuentra una palabra que cumpla los requisitos de longitud establecidos.

4- Una vez la tiene, comprueba que se pueda hacer con las letras sorteadas.

5- Si se puede hacer la muestra en pantalla.

6- En caso de no ser posible se desecha la palabra y se sigue recorriendo el diccionario hasta que se encuentre una válida.

7- Si no se encuentra ninguna el ordenador dirá “No se me ocurre ninguna…”.

Devuelve un array de tipo *Char* con la palabra.

3.5. Clase *Palabra*

Esta clase ha sido proporcionada por el profesor de la asignatura y tiene diferentes métodos que han sido de gran utilidad.

Por ejemplo, el método de *numeroDeCaracteres, lectura, hayPalabras*.

3.6. Clase *FicherosPalabrasIn*

También proporcionada por el profesor de la asignatura, contiene las funcionalidades para llevar a cabo la gestión de fichero a nivel de palabras en modalidad de entrada.

3.7. Clase *FicherosPalabrasOut*

Proporcionada por el profesor de la asignatura, contiene las funcionalidades para llevar a cabo la gestión del fichero en modalidad de salida.

3.8. Clase *LT*

Contiene métodos para la introducción de datos a través del teclado. Hay un método para cada tipo de dato a introducir.

4. Pruebas realizadas para verificar el correcto funcionamiento.

Verificar un correcto funcionamiento y saber que pruebas realizar para garantizarlo es imprescindible para poder dar finalizada la práctica con éxito. Aparte de comprobar visualmente que el juego se ejecuta bien, se han realizado las siguientes pruebas más específicas:

Turno inicial

El turno inicial debe sortearse aleatoriamente, para ello ha sido necesario comprobar que realmente es así. La comprobación se ha hecho en otro proyecto de NetBeans dónde se ejecutaba la sentencia correspondiente al turno aleatorio y se podía visualizar en la salida que realmente se hacía de forma aleatoria.

Letras sorteadas y puntuación

Ha sido necesario fijarse bien en el panel de letras y comprobar que el rango de números aleatorios que escogen las letras de forma aleatoria lo hagan desde el principio hasta el final del array dónde están todas las letras, es decir, que se de la posibilidad de que salgan letras "A" y letras "Z".

Aparte de lo anterior, se ha comprobado que la puntuación que aparece debajo de cada letra se corresponda con la correcta de dicha letra.

Validación de las palabras

En el proceso de validación de la palabra con las letras sorteadas hay que tener en cuenta que no es suficiente que se verifique que cada letra de la palabra forme parte del panel sorteado, si no que también hay que contar cuantas veces aparece cada letra en el panel y que sólo se usen el número de veces que han aparecido.

Ejemplo:

A	T	P	E	S	K	B	N
1	1	2	1	3	7	2	1

Si no se contasen correctamente el número de veces que aparece cada letra, en este caso se nos daría por válida la palabra "pepe". Para ello se han hecho diversas pruebas de ese tipo para verificar que funciona bien.

Diccionario correcto

Habiendo seleccionado el diccionario “castellano”, se han introducido palabras en catalán y se ha comprobado que nos dice que la palabra es incorrecta, de esa manera verificamos que se accede al diccionario correspondiente con el que ha elegido el usuario.

También, cuando la palabra introducida es correcta, se ha hecho una búsqueda manual dentro del fichero para comprobar que realmente existe.

Puntuación de la palabra

Cuando el juego nos devuelve la puntuación obtenida en la palabra ha sido necesario ver que es la correcta.

Puntuación máxima

Al llegar a la puntuación máxima se ha comprobado que se muestre el jugador que tenga el valor más alto y que en caso de haber un empate se muestre por pantalla que no hay un claro ganador.

Palabras con la letra “Ñ”

Todas las palabras que tenían la letra “Ñ” han sido sustituidas por la letra “N” ya que sino daba errores de ejecución.

Partidas reales

Y para terminar de verificar que todo funciona sin problemas se han disputado unas cuantas partidas y se ha ido viendo que todo concurre con normalidad.

5. Manual del usuario.

¡Bienvenido al juego del *Scrabble*!

A continuación, se dará una breve explicación de cómo jugar de forma correcta al juego.

1. Reglas del juego
2. Objetivos
3. Funcionamiento
4. Información

1. Reglas del juego

El juego *Scrabble* consiste en repartir una cantidad de letras a cada usuario y a partir de esas letras se tienen que crear palabras.

Al iniciar el juego será necesario realizar una pequeña configuración previa donde habrá que introducir el nombre del jugador, la puntuación objetivo y seleccionar el idioma del diccionario. Una vez completado este paso, estaremos listos para empezar a jugar.

En esta versión la modalidad de juego es contra la máquina. Quién llegue antes a la puntuación objetivo será el ganador.

2. Objetivos

El objetivo del juego es crear la palabra de máxima puntuación posible en cada turno. Las letras poco comunes tienen una puntuación mucho más alta que las letras más comunes, por ejemplo, las vocales.

3. Funcionamiento

Al iniciar el juego deberemos introducir las configuraciones iniciales. Un ejemplo puede ser el siguiente:

```
**************************************************
                JUEGO SCRABBLE
**************************************************

            Configuraciones iniciales

Introduce el nombre del jugador: nadal

Introduce la puntuación máxima de la partida (Mínimo 1): 60

¿Qué diccionario de palabras quieres usar? (1-3).
            [1] Castellano.
            [2] Catalán.
            [3] Inglés.
Respuesta: 1
```

Una vez introducidas las configuraciones se nos mostrará un resumen de lo introducido y quién empezará jugando.

```
--------------------Resumen--------------------

El nombre del jugador es: nadal
La partida se jugará a 60 puntos.
El idioma del diccionario seleccionado es: Castellano.
Empezará jugando el ordenador.
-----------------------------------------------
```

Se nos pedirá que pulsemos *enter* para dar comienzo a la partida.

El ordenador pensará una palabra para jugar. Una vez pensada se mostrará la palabra y se puntuará. Después del turno se enseñará como están los marcadores.

```
Turno del ordenador
s   p   e   e   s   i   z   p   o   a   i   o   z   h
1   3   1   1   1   1   9   3   1   1   1   1   9   4
Palabra: apeeis
¡Palabra correcta!
¡8 puntos!
------------------------------------------
Puntuación de nadal: 0

Puntuación del ordenador: 8
------------------------------------------
```

Al siguiente turno será para el jugador.

```
Turno de nadal
f   c   e   g   o   u   x   i   q   i   u   j   n   y
4   3   1   2   1   1   8   1   5   1   1   8   1   4
Palabra: ciego
¡Palabra correcta!

¡8 puntos!
------------------------------------------
Puntuación de nadal: 8

Puntuación del ordenador: 8
------------------------------------------
```

Irá transcurriendo la partida hasta que se llegue a la puntuación final. Una vez se llegue se mostrará el ganador.

```
------------------------------------------
Puntuación de nadal: 8

Puntuación del ordenador: 66
------------------------------------------

¡HA GANADO EL ORDENADOR!
Juego finalizado. ¡Hasta pronto!
```

6. Conclusiones.

La práctica ha sido de gran utilidad para asentar los conceptos vistos en clase, sobretodo el temario relacionado con la programación orientada a objetos (POO) y la lectura de ficheros.

La parte bastante interesante ha sido el hecho de enfrentarse a un "gran problema" y saber subdividirlo en problemas más pequeños y más fáciles de solucionar, para así ir haciendo por partes toda la práctica.

Algunas de las dificultades que se han tenido durante el desarrollo de la práctica han sido crear una buena estructura de clases y usar un número adecuado de ellas, así como los atributos de cada una. También la parte que ha llevado más trabajo ha sido la de validar las palabras introducidas por el usuario o por el ordenador con las letras que habían sido sorteadas.

Personalmente me ha servido para estudiar para el examen y para aprender a "buscarme la vida" con cada problema que iba surgiendo a medida que iba avanzando.