Práctica de Programación : Wordle

Elena del Pilar Fernández Wyzynska

Introducción al Wordle

El Wordle es un juego que consiste en el descubrimiento de una palabra de cinco letras, donde el usuario tiene normalmente 5 intentos para descubrirla, para ganar debe de descubrir la palabra en 5 intentos o menos .

El juego debe procesar cada palabra introducida y debe comprobar que exista, no se acepta si no es el caso. Una vez comprobada su existencia se comparan las dos palabras, la palabra objetivo del día y el intento.
La palabra intento se devuelve con las letras coloreadas en función de la palabra objetivo, si están en la palabra o si se encuentra pero no en la posición correcta

Aspectos adicionales añadidos

Como aspecto adicional he añadido que el usuario pueda modificar a gusto los intentos que quiera tener de manera que si le supone un reto hacerlo en solo 5 pueda tener más oportunidades.

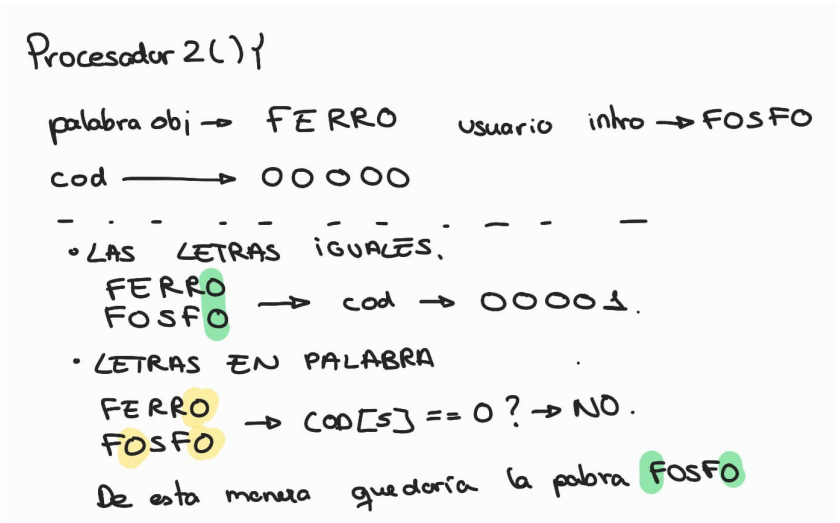
Diseño descendente

- Clase Juego

Primero de todo a nivel general decidí crear una clase Juego en la cual se guardan diferentes aspectos del juego, puesto que tal y como iba haciendo el juego me di cuenta de que era más eficiente tenerlo todo en otra clase, esto podría abrir puertas a una futura implementación del juego, donde se quiera tener un segundo jugador consecutivo y se quisiese comparar ambas partidas.
Aun así por ahora la clase Juego solo es usada para guardar variables como, el nombre del jugador, el número de intentos que ha decidido tener, la hora de inicio y el estado de su partida, es decir si ha ganado o no.

- Clase Palabra

Esta clase ha sido más que necesaria y es la que contiene la lógica principal del juego.
Tenemos subprogramas como SonIguales, que nos compara dos palabras si son iguales o no.
Fue un poco complejo pensar en un método de todos los posibles para implementar el tratado de las palabras introducidas por el usuario, por la característica de que si la palabra introducida por el usuario contiene dos letras que se encuentran solo una vez en la objetivo, el programa debe marcar solo una, ya sea de verde o de amarillo.
Para solucionar este problema recurrí a un método de codificación de la palabra objetivo, es decir, cree un int [ ] de la misma longitud que la palabra y lo rellené de 0, primero hago que vea si alguna está en la posición correcta, si lo está en la posición de esa misma letra en el int[ ] se marca con un 1, de esta manera cuando llegue al segundo proceso para ver si está en la palabra pero no en la posición pues no podrá ser usado.



Para imprimir las letras en color correspondiente es usado un método parecido. Cada vez que encuentra el Procesador 2 una letra que cumpla, se le añade en un int [ ] los numeros 1(verde), 2(amarillo), 3(gris) y estos son mas tarde procesados e imprimidos por el subprograma imprimirletra.

```
// segunda version juego
public void Procesador2(char [] comp, char[] palabraobj){
    int [] aparicion= new int[palabraobj.length]; //para repeticion de letras
    int[] color= new int[comp.length];
    for(int i =0;i<aparicion.length;i++ ){
        aparicion[i]=0;
    }
    // primero miramos las si estan
    for(int i =0;i<palabraobj.length;i++ ){
        if(comp[i]==palabraobj[i]&&aparicion[i]==0){
            color[i]= 1 ;
            aparicion[i]=1;
        }
    }
    //luego las que no estan en la posicion
    for(int i =0;i<palabraobj.length;i++ ){
        if(comp[i]!=palabraobj[i]&&letraenpalabra2(comp[i],palabraobj,aparicion)==true){
            color[i]=2;
            aparicion[i]=1;
        }

    }
    for(int i =0;i<palabraobj.length;i++){
        if(aparicion[i]==0){
            color[i]=3;
        }
    }
    //no se encuentran en el palabra
    for(int i =0; i<comp.length;i++){
    imprimirletra(comp[i],color[i]);
    }
    System.out.println(" ");
}
```

Conclusión

El proceso de resolución ha sido complejo, pasé por varias fases para tenerlo y cada una supuso alguna que otra dificultad. La primera fue con la lectura y escritura de ficheros la cual fue un reto, pero una vez hecho eso, pase a la interfaz general del juego, el menú.

Tenía bastante claro que primero tenía que hacer el juego antes que las estadísticas, y el proceso de programar el código del juego comenzó, tuve varios problemas con la escritura en pantalla de los char arrays, que solucioné pero un problema más complejo se apoderó de mi tiempo.
Desde un principio tenía claro que quería tener todas las palabras del jugador guardadas durante la partida, pensé en hacer atributos en la clase Juego o Palabra pero no tenía mucho sentido así que me decanté por un String Array de palabras, el cual puedo modificar su longitud y podía tener todas las palabras almacenadas.

Aún así el mayor problema fue la lógica de las palabras, he tenido 5 prototipos diferentes antes de llegar a uno que funcionase correctamente tal y como pedía el enunciado, uno de mis errores fue no comprender bien el procesamiento de las letras múltiples, explicado anteriormente.

Hacer pruebas de testeo también ha sido fundamental, pero más que eso comprender los errores que salían, supone a veces un reto comprender qué es lo que está sucediendo en el código. Fue en esos momentos que los ejercicios anteriores cobran sentido, los errores te suenan y ya tienes una idea intuitiva de que es lo que sucede.

Hacer este trabajo en definitiva me ha hecho tener un dominio mejor de las clases y comprender mejor tanto su uso como su función, también ha requerido de un entendimiento de los métodos de recorrido y búsqueda los cuales han sido esenciales para resolver este problema en varias partes, en definitiva ha sido un buen ejercicio para asentar lo enseñado este curso.