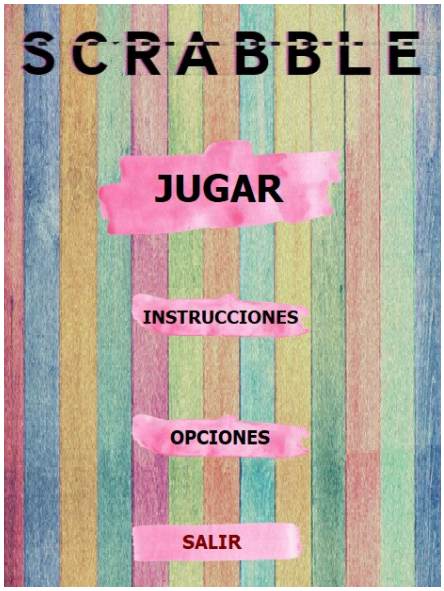
SCRABBLE

Trabajo práctico dual presentado para la cátedra de Lenguaje de programación 2.
Lenguaje de programación utilizado: C#

Integrantes: -Elias Bogado
-Evelyn Gimenez

La clase inicio y su form.

```
public partial class Inicio : Form
```



Como en todo programa, Separable cuenta con un menú de inicio, donde en primera instancia el usuario ya se ve habilitado a interactuar con el juego.

La primera opción visible y la mayor de todas es la opción de jugar. Este botón pide al usuario un nombre con el cual registrarse en el tablero para poder jugar y comenzar con su función principal.

En segundo lugar está el botón de instrucciones, donde se muestra al usuario un .gif, que se encarga de explicar todas las funciones dentro de la interfaz del tablero.

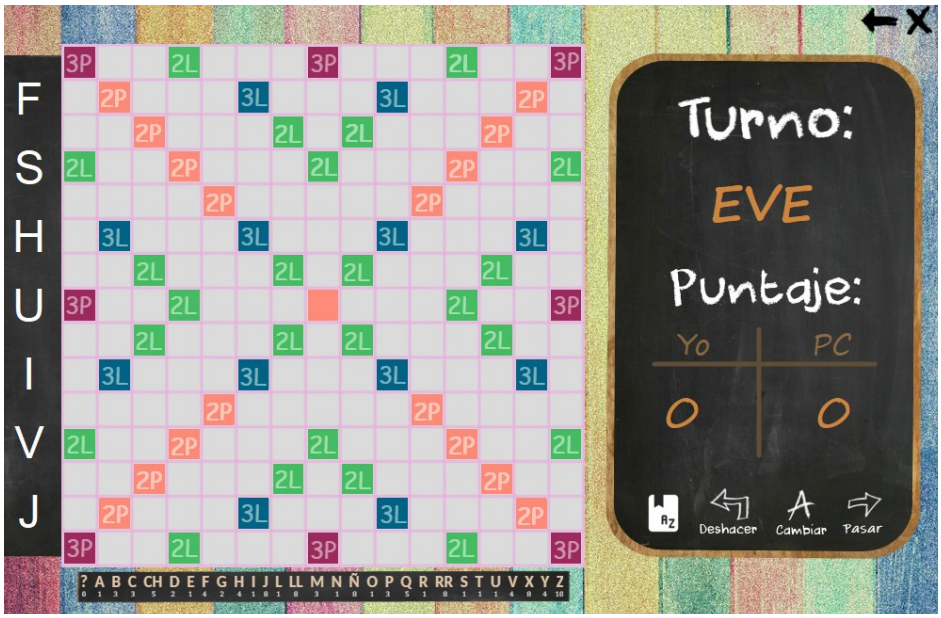
Antes del botón salir tenemos las opciones del juego, este nos permite elegir antes de jugar a quien le va pertenecer el primer turno, así como la dificultad del juego.

Por ultimo el infaltable botón salir, que nos permite terminar con la ejecución del programa.

Profundicemos el botón opciones, este botón permite al usuario elegir quien comenzará el juego y la dificultad que este tendrá.
La primera de estas opciones se realiza cambiando valores contenidos en variables de turno, mientras la segunda representa la cantidad de letras que la IA puede utilizar para permutar y realizar sus jugadas. Funciones que serán explicadas más adelante.
Cabe destacar que la rapidez del juego se ve aumentada cuando esta dificultad baja, mientras que el tiempo de respuesta aumenta cuando este lo hace.

La clase tablero y sus propiedades

public partial class Tablero : Form



El tablero principal del juego, en ella podemos ver todo lo necesario para realizar jugadas según las reglas del juego.

Es posible ver las letras, sus valores correspondientes, a la izquierda un atril de letras pertenecientes al usuario, botones para pasar el turno, cambiarlo, deshacer una jugada puesta en el tablero y un diccionario para consultar palabras.

También podemos observar el turno actual y la tabla de puntajes.

La introducción de jugadas se realiza mediante el mouse, seleccionando una de las letras, guardándolas en la mano para introducirlas en una posición del tablero, en ese momento se hará visible el botón jugar, para que la palabra introducida y la forma sean verificadas, todo ello en base a las reglas del Scrabble en español.

-Las funciones principales dentro de Tablero

private void Tablero_Load(object sender, EventArgs e)

Al iniciar el tablero se carga automaticamente el atril del usuario, asi este se vuelve visible y utilizable.

Del mismo modo se analizan las opciones elegidas por el usuario en el menu Inicio, ya que de elegir que la IA realice la primera jugada, desde esta función se realiza su llamado a jugar.

private void Buscar(object sender, EventArgs e)

Esta funcion se encarga de mostrar mensajes en pantalla cuando el usuario está realizando una mala jugada o es necesario acomodar las fichas una bez realizada la jugada.

Asi tenemos dentro de tablero una gran cantidad de funciones que ayudan al usuario en la experiencia dentro del juego.

La clase LetrasPalabras

class LetrasPalabras

Dentro de esta clase es donde se definen todas las variables y funciones necesarias para comparar palabras, letras, ver el puntaje de la jugada realizada y todo lo que implique letras y numeros dentro del programa.

Aqui podemos encotrar las siguientes variables y funciones:

```
Dictionary<int, string> dictionary = new Dictionary<int,string>
{
        {0,"?"},{1,"A"},{2,"B"},{3,"C"},{4,"CH"},{5,"D"},{6,"E"},{7,"F"},{8,"G"},{9,"H"},
{10,"I"},{11,"J"},{12,"L"},{13,"LL"},{14,"M"},{15,"N"},{16,"Ñ"},{17,"O"},{18,"P"},{19,"Q"},
{20,"R"},{21,"RR"},{22,"S"},{23,"T"},{24,"U" },{25,"V"},{26,"X"},{27,"Y"},{28,"Z"}
 };
```

Un diccionario que posee una letra y su numero dentro del juego, el 1 representa una A, el 28 una Z, y así sucesivamente.

public void Cargar_atril(string devolver)

Cargar atril se encarga de llenar los espacios vacios del atril del usuario, cambiar las letras que no quiera y mantener el atril listo para jugar.

public void Crear_atril(ref Label atril)

Crear atril carga objetos por referencia de nuestro label a una matriz dentro de LetrasPalabras, para que pueda ser manipulado facilmente dentro de las instancias de la clase en el juego.

public int Consultar_palabra(string palabra)

Consultar palabra es la funcion que da utilidad al boton diccionario dentro del tablero, según la opalabra elegida esta se encarga de buscarla y retornar un resultado positivo o negativo a la busqueda.

La clase casilla

class Casilla

Casilla es la variable utilizada para crear una matriz con toda la información correspondiente a cada label dentro del tablero, cada una de estas posiciones contiene un valor, una letra, un multiplicador, etc. Es decir, todo lo que el tablero posee pero de forma más liviana para trabajar con ella dentro de las funciones siguientes.

La clase estado Tablero

class Estado Tablero

Esta clase puede considerarse el main dentro de todo el programa, la clase encargada de conectar, dar y recibir, la dueña de todas las funciones que hacen del programa lo que es.

Entre sus funciones podemos encontrar:

public void Cargar_tablero(ref Label labelR)

Cargar tablero es el encargado de crear la matriz de casilla recibido por referencia desde el tablero original, asigna a cada espacio de la matriz sus valores correspondientes.

public int Cargar_jugada(string letra_marcada, string nombreLabel)

Cargar jugada ingresa en la casilla matriz el valor de la letra jugada por el usuario y dueña de las condiciones y reglas de comodin, letras especiales (CH,LL,RR), de añadirlas o eliminarlas según el usuario lo desee.

public string Verificar_palabra(ref int jugada, ref int puntaje)

Una de las funciones principales del programa, sin esta ninguna jugada sería valida, tendría puntos o pudiera ser verificada.

Esta función abarca el 75% del estado Tablero.

Se compone de 3 partes.

Lectura y eliminación, verificación y puntaje

La lectura se separa en 3 procesos importantes, el primero se encarga de asegurarse que las letras ingresadas se encuentran en la misma linea, con ello la determinación del sentido de la palabra, la segunda de que no haya espacios en blanco, ya que no es valida la inversionista de los palabras en un mismo turno, como tercer paso tenemos la verificación de esta palabra mediante las funciones vertical y horizontal, con los cuales obtenemos la validación completa de la jugada y el puntaje obtenido.

En estos casos ya se haya contemplado la jugada de una sola letra y las jugadas colaterales que pudiesen ser realizadas por el usuario.

Segunda parte

Class IA

public void Encuentra_palabras_Horizontales(Informacion[,] info)

- La función se encarga de encontrar todas las palabras horizontales en la información del nodo actual.
- Encuentra las palabras recorriendo la matriz (info) que contiene todas las letras del tablero, de modo que, al preguntar si la posición actual de la matriz contiene una letra, entonces en esa fila existe una o mas palabras. Las palabras son almacenadas en en un vector de string sin lengh, debido que no sabemos cuantas palabras podemos formar en el trascurso del juego

public void Encuentra_palabrasVerticales(Informacion[,] info)

- La función se encarga de encontrar todas las palabras verticales en la información del nodo actual.
- Encuentra las palabras recorriendo la matriz (info) que contiene todas las letras del tablero, de modo que, al preguntar si la posición actual de la matriz contiene una letra, entonces en esa columna existe una o mas palabras. Las palabras son almacenadas en en un vector de string sin lengh, debido que no sabemos cuantas palabras podemos formar en el trascurso del juego
-

public void Cargar_atril(int seed)

- Cargar atril, carga las letras en el atril de la IA, recibiendo un “seed” que se suma al evviorment.TickCount para aumentar las probabilidades de que la IA tenga palabras distintas a las que ya tenía.
-

public void BucadorCoincidencias(string palabra)

- Es una función corta pero una de las más útiles en el código, lo que hace es, determinar si entre todas las permutaciones posibles del atril y palabraPadre, existen coincidencias de nuestro diccionario, sabiendo así, cual de las combinaciones posibles puede jugar la IA.
-

public void Permutaciones(string[] vec, int k, int n)

- Function que determina todas las posibles combinaciones(permutaciones) de las letras que posee el atril de la IA y las palabras existentes del tablero
- Estas permutaciones se almacenan todas sumadas en un string de modo que en el esten contenidas las combinaciones y acelere el proceso de lectura, puesto que es mas rápido leer una tipo de dato de un bite, que un txt de 40320 lineas
-

public string RecorteAlphaBeta(List<string> mejoresJugadas, Nodo nodo)

- Lo que hace es esta función es básicamente determinar el llamado de cada hijo en otra función, técnicamente colocaPalabra es la que realiza el recorte, pero recorteAlphaBeta es la que se queda con la lista y el mejor info para el nodo actual que es llamado
-

public int ColocaPalabraIA(Nodo nodo, string palabra)

- Como dice su nombre, el método es quizás el mas importante de IA, porque es que intenta o “ ve ” la forma en la que cada sub hijo de cada palabra padre será colocada en el tablero.

Viendo la infinidad de casos que existen de colocar las palabras en el tablero y que sean verificadas. Decidimos separarlos en dos cosos grandes, que a su vez se subdividen en otros casos. Adelante explicaremos la forma en la que estas palabras son procesadas por la IA

- **La palabra a jugar esta contenida en la palabra del nodo padre :**
  Esto quiere decir que la palabra a jugar se econtrara de algún modo, contenida en la palabra padre (palabra que fue usada para crear la nueva palabra). Dicho esta, podemos afirmar que la palabra será en base a una palabra que ya esta en el tablero, y la IA tendrá que escribir la nueva palabra en dirección de la que ya esta en el tablero

- **La palabra hijo es complemento del padre:** se da cuando la palabra a jugar se encuentra después de la palabra padre, para saber donde se encuentra o se escribe las posiciones de esta nueva palabra, en base al inicio de la palabra padre mas su lengh

- **La palabra padre es complemento del hijo:** se da cuando la palabra a jugar se encuentra antes de la palabra padre, para saber dónde se encuentra o se escribe las posiciones de esta nueva palabra, en base al inicio de la palabra padre menos el lengh de la palabra hijo

- **la palabra padre se encuentra en el medio:** Para este caso en particular se desprende al padre del hijo, ya que no es el, el que forma la palabra principal. Para este caso se suma y se resta el lengh del resto, ya que existe una única manera de escribir la palabra, y es usando el lengh de los hijos.

- **Caso general:** es cuando el padre esta contenido en el hijo, pero solo en una coincidencia, para este caso, se busca la coincidencia en la que la palabra padre contiene al hijo, y se escribe sacando el lengh de la palabra hijo tomando como referencia la posición de la palabra padre.

  Para este caso se utiliza SinPadreHorizontal() y SinPadreVertical()
  Ambos haces verificación de lo explicado arriba, pero considerando la dirección de la palabra padre, ya que para este caso, si el padre es horizontal, se escribe verticalmente, y al revés para el caso opuesto

public string VerificarJugadaIA(Nodo nodo)

- Secuencialmente hablando, este seria el paso final para que un hijo, pueda considerarse candidato para su nodo padre, ya que esta función, verifica si lo que intento jugar la IA en el nodo actual es “ correcto ” según dictan las reglas del Scrabble.
  Si el info actual devuelve “Success” eso quiere decir que la jugada se pudo realizar
  Si devuelve “LOSS” es porque en algún lugar, la escritura en información no cumple con los estándares de las reglas(si existen letras adyacentes, tienen que formar palabras también)
  Donde , verificaVertical y verificaHorizontal son complementos de la esta, separándola, en dos casos grandes de verificación para la IA

Class Nodo

La clase nodo, es la que hace “Link” con sus otros hermanos u hijos del mismo tipo
Esta clase no contiene métodos, pero si contiene los tipo de datos que serán ente nodo y nodo, además de eso, tienen un tipo de dato definido por nosotros llamado información, que lo que guarda es el estado actual del tablero, que será heredado de nodo a nodo.

Class Información

Esta clase, es en realidad tratada como un tipo de dato creado por nosotros, la razón por la que es clase y no un tipo de dato definido por nosotros mismos, es por el alcance y la versatilidad que tendrá este al poder ser construido en cualquier clase a la que lo llamemos.
Info, guarda todos los valores actuales del tablero y es único por cada turno.

Class Arbol

La clase árbol, es que tiene propia potestad, es el quien crea los hijos, los padres y la raíz.
También se encarga de leer el árbol para encontrar un < predicador >