Universidad Estatal a Distancia.

Escuela de ciencias exactas y naturales

PROGRAMACION INTERMEDIA

PROYECTO2

Olman Rojas Espinoza

Carnet: 0110900754

Grupo #2

AGOSTO 2014

INTRODUCCION

En el siguiente Proyecto se va a realizar un programa de juego tipo ahorcado, este programa va a consistir de dos módulos.

El primer módulo es donde se va a escoger un archivo para introducir las palabras con las que se quieren jugar.

El segundo módulo consiste en el juego en sí.

El sistema se va a desarrollar en lenguaje JAVA y el patrón de diseño es MVC, donde primeramente vamos a definir las vistas y luego la lógica en si para entregar los datos.

Dicho esto, la primera interfaz gráfica va albergar 3 botones, los cuales serán:

La Configuración.

El Juego en Si.

Salir del Programa.

En algunos casos se utilizará algunas librerías propiamente del API de JAVA, como son las librerías de Random() y las de FileHandler(), se construirán algunas clases, para hacer los llamados desde los botones y evitar confundir al lector del código de fuente, desviarse en la lectura dentro de la interfaz gráfica.

Esta fórmula también va a permitir que se puedan reutilizar las clases, tal es el caso entre los dos primeros botones el de Configuración y el del Juego en sí, van a utilizar la misma clase de FileHandler() para manejar el archivo de configuración.

INTRODUCCION.

A Continuación se definen las clases a utilizar:

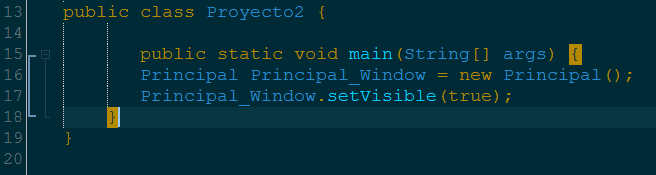


Como se puede apreciar en la imagen las clases a utilizar son las siguientes:

* Ahorcado.java
* Configuracion\_Frm.java
* FileHandler.java
* Principal.java
* Proyecto2.java

# Proyecto2.java

Esta clase contiene la clase main, que es la principal aquí es donde todo el programa comienza, para empezar el programa en sí, se hace referencia a una instancia de Principal.java donde principal contiene el formulario primario que contiene los 3 botones.



# Principal.java

Esta clase contiene un formulario para trabajar con el programa, este formulario consiste de 3 botones.

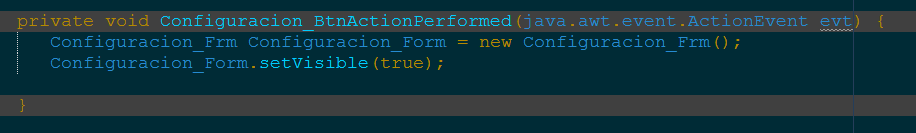
Los cuales son:

* Configuración
* Juego
* Salir

Cada botón maneja un evento:

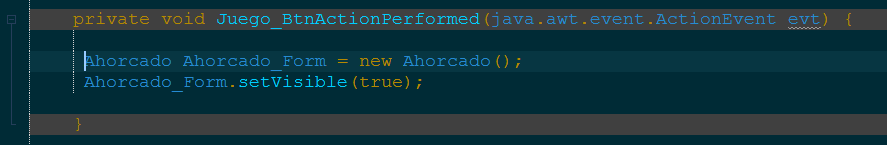
# Botón de Configuración:

Este botón, hace una instancia del Formulario Configuracion\_Frm, en donde veremos cómo se configura un archivo, a su vez lo hace visible.

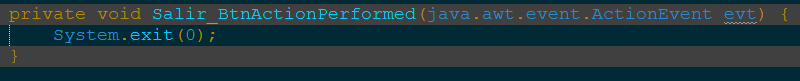


# Botón de Juego:

Este botón instancia un formulario de la clase Ahorcado y lo hace visible

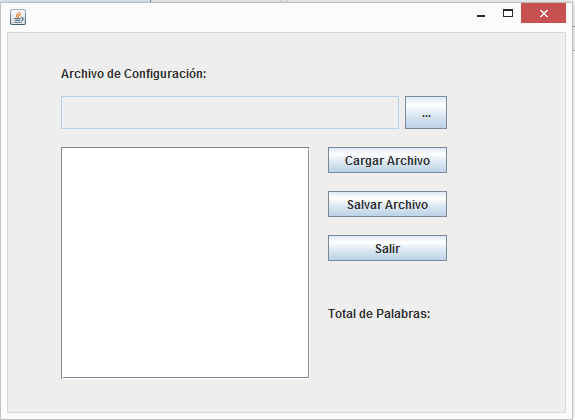


# Botón de Salir.

Como su nombre lo indica es para terminar el programa.

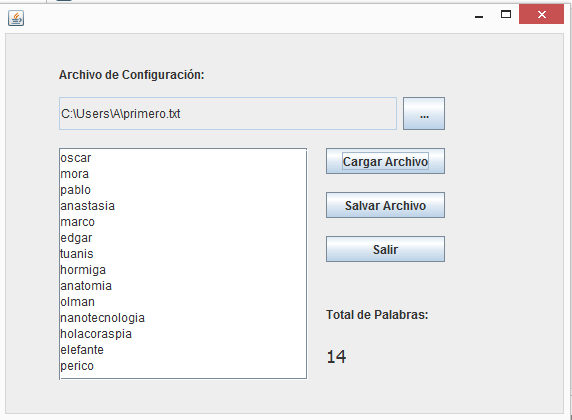
# Clase Configuracion\_Frm

Esta clase contiene el formulario para manipular el archivo de palabras en una forma mas amigable con el usuario:

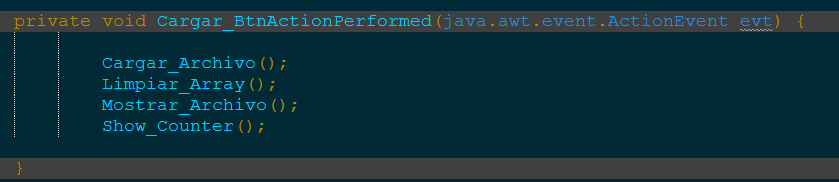


Si se pulsa el botón que contiene los 3 puntos (…), este hace referencia a un método de la clase FileHandler, que procede a hacer un llamado al método JFileChooser el cual permite al usuario escoger un archivo de interés.

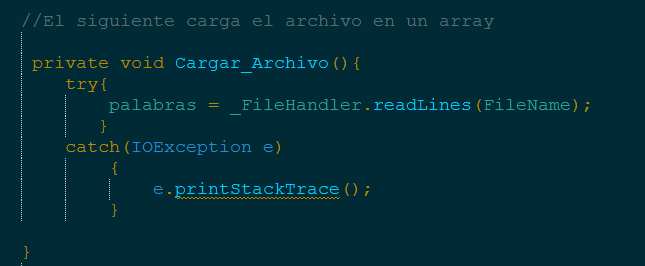
Cuando se escoje el archivo y se pulsa el botón de “Cargar Archivo”, este hace una lectura del archivo y lo despliega en el cuadro en blanco o TextArea:



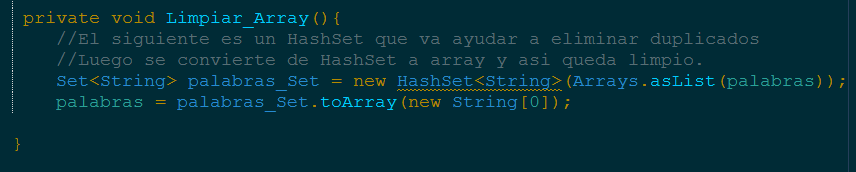
El botón de “Cargar Archivo” hace la referencia a los siguientes métodos:



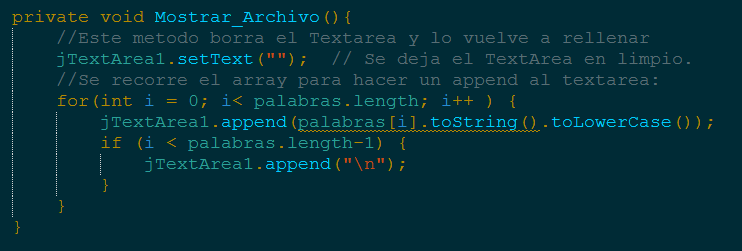
Donde “Cargar\_Archivo” hace referencia a una instancia de la clase FileHandler que se llama readlines(), el método readlines, lee el archivo en forma secuencial y devuelve un array, este array se conserva en un array tipo string llamado palabras:



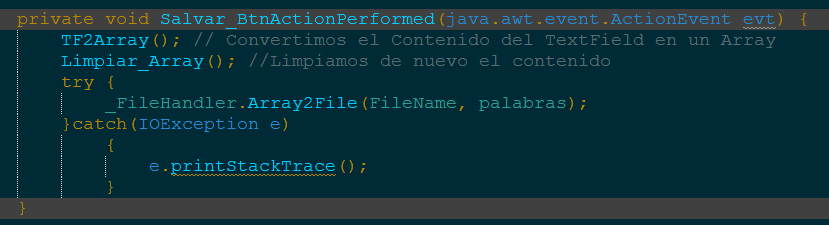
Para evitar la repetición de palabras, se hace un llamado al siguiente método: Limpiar\_Array(), este método lo que hace es convertir el Array “palabras” en un HashSet, y luego lo vuelve a convertir en un Array, de esta manera evitamos la repetición de cualquier palabra.



El siguiente método es mostrar Archivo, en donde se va a recorrer el array, se evitan las letras en mayúscula y se despliega en la caja de texto TextArea:

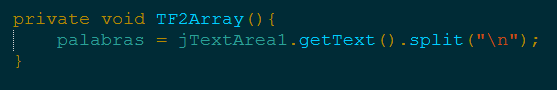


Si se desea dejar el archivo en blanco, o agregar nuevas palabras existe la opción de utilizar “Salvar Archivo”:

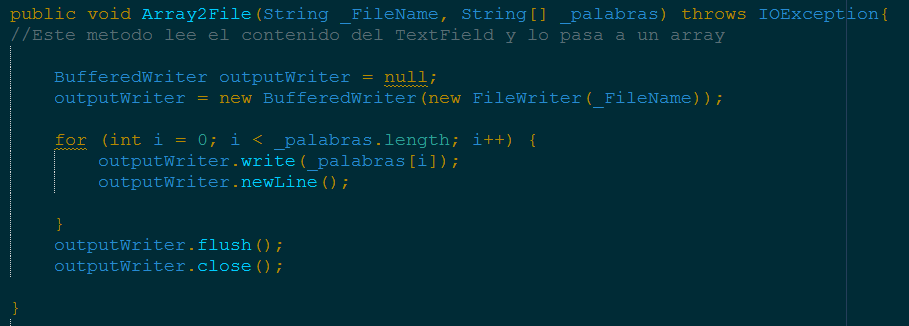


Este botón, contiene 3 métodos:

TF2Array() donde tomamos el contenido del TextArea y lo convertimos en un array, la formula para aplicar este método consiste en utilizar el método Split y el regex esta en el carácter “\n”:



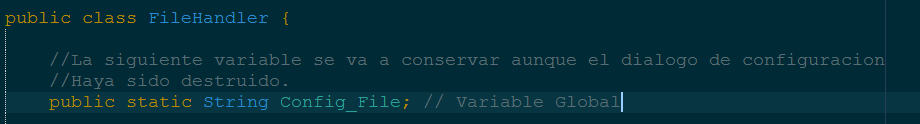
Una vez que el TextArea ha sido cargado en un array, se procede a reutilizar el método limpiar array, para evitar el duplicado de palabras, y luego se hace un llamado a la instancia de FileHandler método Array2File() para que imprima las palabras en un archivo.



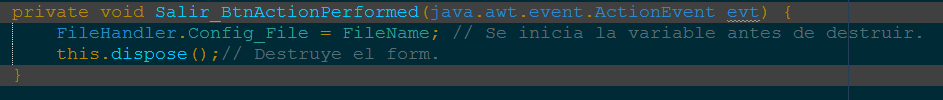
Por ultimo tenemos el botón de salir, el cual hace referencia al método dispose(); lo que hace es destruir el formulario y desligar sus cargas en memoria.

Ahora se preguntará como hace el siguiente botón “Juego” para hacer referencia al archivo que se configuró.

Para ello se declaró en la clase FileHandler una variable global que se conserva.



Y la forma de decirle al formulario de Configuración que inicialice esa variable es en el botón de salir:

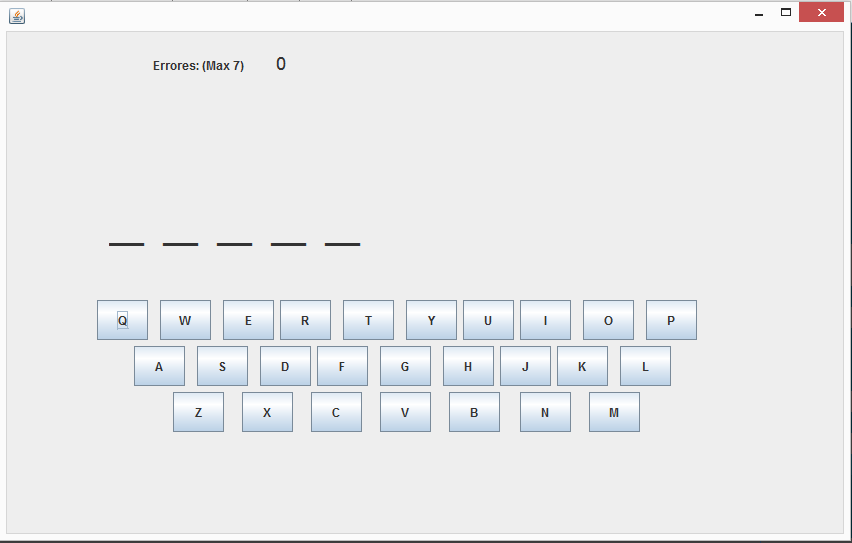


# Botón de Juego

Este botón hace referencia a la Clase Ahorcado, la clase Ahorcado contiene tanto el juego visual como su lógica:

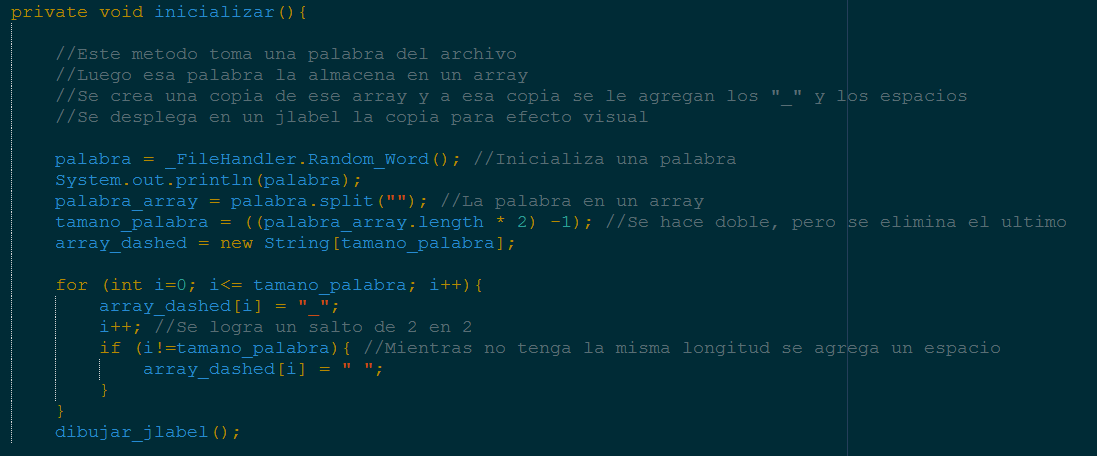
# Clase Ahorcado

Esta clase contiene el juego en sí, cuando se hace llamada esta es su visualización:

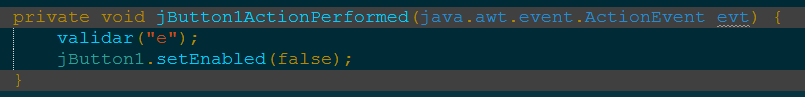


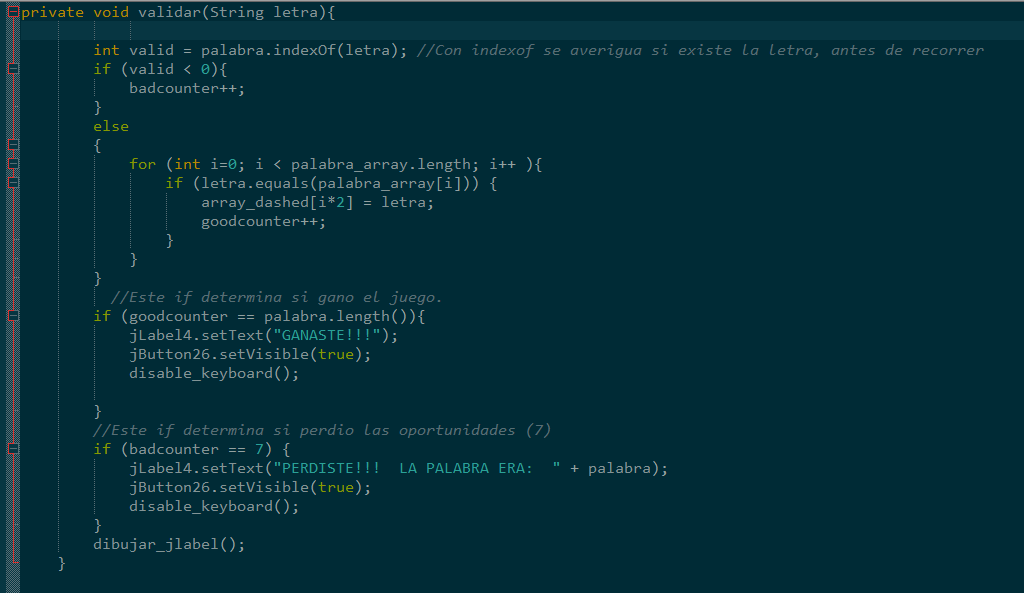
Se puede observar un contador de Errores, asi como la palabra enmascarada por “underscores” y una especie de teclado virtual, donde verdaderamente son 26 botones cada uno con una identificación y una llamada a un método llamado validar().

Cuando esta clase se inicializa dentro de su constructor se hace llamado a un método inicializar(), el cual contiene la lógica para traer una palabra aleatoria del archivo de configuración y a su vez crea 2 arrays, 1 array que contiene la palabra en si, y otro array donde remplaza las letras por los underscores “\_” y espacios en blanco:



La siguiente lógica consiste en pulsar cualquier letra para adivinar la palabra, cuando se pulsa cualquier letra esta hace un llamado al método validar() y a su vez deshabilita la letra para intuir al usuario de que ya utilizó la misma:





En donde la logica de validar() reside en preguntar primero con IndexOf() si la letra escogida existe, si esta existe entonces se mete de lleno en las validaciones de lo contrario aumenta un contador de malas escogencias.

A su vez toma el segundo array y remplaza los “underscores” “\_” por las letras adivinadas y un método llamado dibujar\_jlabel() lo que promueve es repintar la palabra tomando como referencia el array.

Si los contadores se cumplen, se notifica si gano o perdió, y para terminar el juego se utiliza la lógica de deshabilitar el teclado “disable\_keyboard()” lo que hace es invisibilizar los botones, y pone en visible el botón de salir. Si se desea jugar de nuevo se procede a pulsar el botón de “Juego” con lo cual se instancia de nuevo toda la lógica.

CONCLUSION

Como se pudo observar a lo largo de este proyecto2, Java permite muy eficazmente la reutilización de métodos y clases, con lo cual reduce la reinvención del código.

El programa de Ahorcado descrito en este documento se puede apreciar como la configuración de un archivo y su formateo de contenido puede limpiarse utilizando la transformación de un tipo primitivo a otro tipo, evitando duplicados y haciendo ordenamiento del mismo.

Se pudo observar como el uso de una variable global para el apoyo de información entre clases, y cuando se hace la destrucción de un objeto como desaparece de la memoria pero se conserva la información de la variable global.

Algo muy característico de este tipo de aplicaciones, es que existe lógica que se puede obviar utilizando trucos visuales, y es el ejemplo en el caso de este juego cuando se cumplen si ha ganado o perdido, eliminando los botones del teclado. Aunque la lógica sigue en pie y el contador puede continuar, al no haber elementos para continuar el usuario intuye en que la aplicación ha terminado, y no existe otro remedio que cerrar la ventana y empezar un nuevo juego.

BIBLIOGRAFIA

Gootooru, N. (2014, 7 27). *Program: How to iterate through HashSet?* Retrieved from javatonovice.com Java Example Programs: http://www.java2novice.com/java-collections-and-util/hashset/iterate/

Oracle Corporation. (2014, 07 27). *The Set Interface.* Retrieved from The Java Tutorials: http://docs.oracle.com/javase/tutorial/collections/interfaces/set.html

Patel, V. (2014, 7 27). *Convert Arrays to Set in Java.* Retrieved from viralpatel.net: http://viralpatel.net/blogs/convert-array-to-set-java-arraylist/

*Show simple open file dialog using JFileChooser.* (2014, 7 27). Retrieved from Code Java: http://www.codejava.net/java-se/swing/show-simple-open-file-dialog-using-jfilechooser