Manual Técnico

Versión 0.1.1

Mayo 2017

Introducción

Este manual tiene el fin de presentar a detalle la composición de la aplicación “Sakura Kanji Collection”, tanto de la interfaz gráfica como del sistema interno, para mejorar su entendimiento y poder optimizar tiempos al realizar el mantenimiento de la aplicación.

0. Antes de empezar

Sakura Kanji Collection es un proyecto elaborado únicamente con Java. Se utilizó la librería Swing de java (`javax.swing`) para la interfaz gráfica.

0.1. Conceptos del área:

El público al que está destinada la aplicación es muy específico: Estudiantes del idioma japonés; por esta razón, en la composición del programa se encuentran conceptos especializados, los cuales se detallan a continuación:

- **Kanji**: Es uno de los sistemas de escritura japonesa. Lo componen logogramas, los cuales pueden tener asociadas distintas lecturas dentro del idioma, dependiendo de su contexto y uso. Fueron ingresados a Japón a través de China.
- **Logograma**: Unidad de un sistema de escritura, que por sí sola representa una palabra o significado, compuestos de elementos gráficos ordenados para transmitirlos. Se diferencian de los sistemas de escritura *fonéticos*, en los que cada símbolo representa un sonido (como lo es el alfabeto latino utilizado por el idioma español).
- ***Hiragana***: Segundo sistema de escritura japonés. Derivado del sistema de kanji, éste es simplemente un *silabario*, es decir, un sistema de escritura fonético en el que cada símbolo representa (o se aproxima a) una sílaba. Se compone de 46 caracteres. Es utilizado para representar las palabras nativas japonesas, sufijos, prefijos, y *partículas*.
- ***Katakana***: Tercer sistema de escritura japonés (vaya lío). De composición idéntica al Hiragana, excepto que el símbolo que representa cada sílaba posee una escritura distinta. Es utilizado para adaptar y escribir palabras extranjeras, onomatopeyas, y en ocasiones para resaltar palabras que comúnmente se escribirían en kanji o en Hiragana.
- **JLPT**: Es el Examen de Aptitud del Idioma Japonés (en inglés: Japanese Languaje Proficiency Test). Categoriza la habilidad del manejo del idioma en 5 niveles, descritos comúnmente con una “N” al inicio: N1, N2, N3, N4, y N5, donde N1 es el más alto y N5 el más bajo. De esta forma, el manejo de los kanji también se evalúa en esta prueba, y se categoriza en los mismos niveles.
- **Número de Trazos**: Cantidad de trazos con la que se escribe (de manera correcta) un kanji.
- **Frecuencia**: Número que determina qué tan frecuente se observa un kanji en determinados medios (como los periódicos japoneses). El número es único en cada medio que es evaluado, sin embargo, puede variar dependiendo el medio en el que se basó el estudio. Éste, y otros datos como el número de trazos se pueden consultar en diccionarios de japonés en línea.

- **Lectura Onyomi**: “Onyomi” literalmente significa “Lectura del sonido”. Es la lectura que se dio a cada kanji a partir de adaptar la pronunciación nativa china, aunque para algunos kanji ha evolucionado en más de una. Suele utilizarse cuando una palabra se compone de dos o más kanji. En los diccionarios, esta lectura suele incluirse usando el sistema *Katakana**.***
- **Lectura Kunyomi**: Contrario a la anterior, Kunyomi son las lecturas que representan el significado, la forma nativa en la que se dice la palabra, o lo que se aproximó a ella cuando fueron introducidos. De la misma forma, pueden existir varías de estas lecturas para un solo kanji, aunque en este caso también puede diferir su significado o su carácter léxico (por ejemplo, si representa un sustantivo, un adjetivo o un verbo). En los diccionarios, estas lecturas suelen aparecer escritas usando el sistema *Hiragana*.

0.2. Detalles de elaboración:

Para la elaboración del programa se ocupan algunos recursos que comúnmente no se encuentran predeterminados en computadoras occidentales, tales como caracteres especiales. A la hora de trabajar se debe considerar lo siguiente:

- **Codificación UTF-8**: El programa maneja siempre caracteres especiales, los cuales se presentan en el formato de codificación de caracteres UTF-8. Al momento de compilar el programa, es necesario especificar al compilador que se trabaja con dicha codificación agregando **`-encoding utf-8`** a las instrucciones.
  También es recomendable trabajar en un entorno o editor de textos que permita esta codificación.
  No es necesario configurar la consola de Windows.
  Así mismo, las fuentes que utiliza el programa tienen soporte para caracteres especiales. Si el objetivo es cambiar alguna fuente, primero se debe comprobar que tiene soporte para estos caracteres.
- **Elaboración sin IDE**: El programa está desarrollado sin hacer uso de ningún entorno de desarrollo integrado (IDE), por lo que la GUI puede presentar problemas en algunos dispositivos, (la mayoría solo visuales, no afectan al funcionamiento de la aplicación), debido a no tener la optimización que suelen integrar los IDE automáticamente.
- **Idioma del código**: Aunque no es una regla dar nombres a identificadores en concreto, para mantener la legibilidad al código y mejorar la comprensión de cada elemento, los identificadores son nombrados con palabras en inglés, o en su defecto, con la *romanización* de las palabras en japonés, es decir, escribir el sonido de la palabra con letras del alfabeto latino (como lo es escribir “Kanji”). Los comentarios y los textos en la interfaz se muestran en español.

0.3. Motivación:

Para un hispanohablante, el japonés es un idioma difícil de dominar, pues contrasta en muchos sentidos con el idioma español. Desde tener otra estructura hasta manejar tres sistemas de escritura distintos al mismo tiempo.

Uno de los campos más complicados de llegar a dominar es la lectura/escritura. Aunque se puede aprender a manejar el idioma solo hablado, ello no garantiza poder leer un texto con la misma habilidad. La escritura japonesa maneja tres sistemas de escritura, dos fonéticos, y uno logográfico: los kanji. Los kanji son caracteres que no representan sonidos, sino conceptos, significados, y que por lo tanto, al inicio son complejos de comprender.

Para un estudiante de japonés, los kanji representan la parte más complicada de estudiar, pues debe acostumbrarse no solo a forma de escritura y pronunciación. Los kanji se conforman por una forma de escritura que supera incluso los 20 trazos, y pueden combinar más de 5 lecturas diferentes. Sin embargo, en muchas ocasiones es posible entender una palabra escrita en kanji sin necesidad de saber cómo se pronuncia, sino basándose en los significados de cada kanji que la conforma. “Sakura Kanji Collection” es una aplicación pensada para apoyar al estudiante de japonés en el estudio de los kanji, basado en la creación de Tarjetas de Aprendizaje, las cuales son generadas por el propio usuario, ingresando la información que considere necesaria, con el fin de que pueda personalizar su estudio como más le convenga.

0.5. Manejo de Versiones:

Las versiones de este programa se componen de tres números que indican lo siguiente en este orden:

1. Versión de lanzamiento: Indica con un 0 sí la aplicación sigue en estado de desarrollo y no ha sido lanzada, o 1 si la aplicación ya está disponible.
2. Versión del Kanji: Indica en qué versión se encuentra el kanji para indicar que se presentan cambios en la clase. Incrementa cuando se hacen cambios significativos a la clase **Kanji**.
3. Versión del programa: Indica qué versión del programa es con respecto de la versión del kanji. Se incrementa cada vez que se terminaron de implementar todas las nuevas funciones planificadas para esa versión y se obtiene una versión estable, y se reinicia a 0 cada vez que el kanji incrementa de versión.

1. Análisis del Problema

1.1. Problema:

Se desea crear una aplicación de creación de Tarjetas de Aprendizaje, la cual pueda guardar la información esencial de un kanji, así como información adicional proporcionada por el usuario, y que además las almacene ordenadamente, con el fin de optimizar la búsqueda de un kanji en específico, o de un grupo de kanji.

La aplicación debe ofrecer las funciones básicas de “Agregar” y “Buscar”, y la búsqueda debe ser a través de una clave o de un conjunto de características del kanji; la primera buscará un kanji en específico, la segunda buscará un conjunto de kanji que cumplan con dicha característica.

Al momento de la búsqueda, deberá poderse seleccionar un kanji, lo que desplegará la información que ha introducido el usuario (mostrar la Tarjeta del kanji).

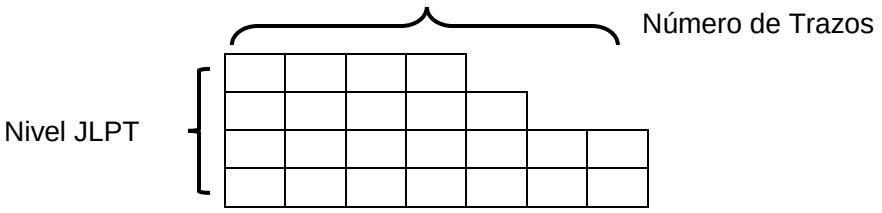
1.2. Modelado de la solución:

El diagrama *SystemClassDiagram*[1.1] ilustra las clases que se modelaron para dar solución al problema de forma interna (no incluye la construcción de la interfaz gráfica).

Primero se modeló la clase **Kanji**, la cual almacena toda la información relacionada con el kanji (nivel, trazos, frecuencia, lecturas y significados, por cada objeto intanciado). Además, almacena de forma estática y como constante l*a versión del modelo de la clase*. La versión de la clase Kanji se indica para dar compatibilidad a las listas de kanji generadas con versiones anteriores de “Sakura Kanji Collection”, de tal forma que permita abrir listas (archivos) de kanji hechas con versiones anteriores, y puedan ser actualizadas al volver a guardar con la nueva versión.

La clase **Kanji** compone a la clase **Node**, que es un nodo dentro de la clase **AVLTree** (Árbol AVL). La clase **Node** es privada a **AVLTree**, por lo que no se puede instanciar objetos **Node** fuera de ella, su uso es exclusivo para el árbol.

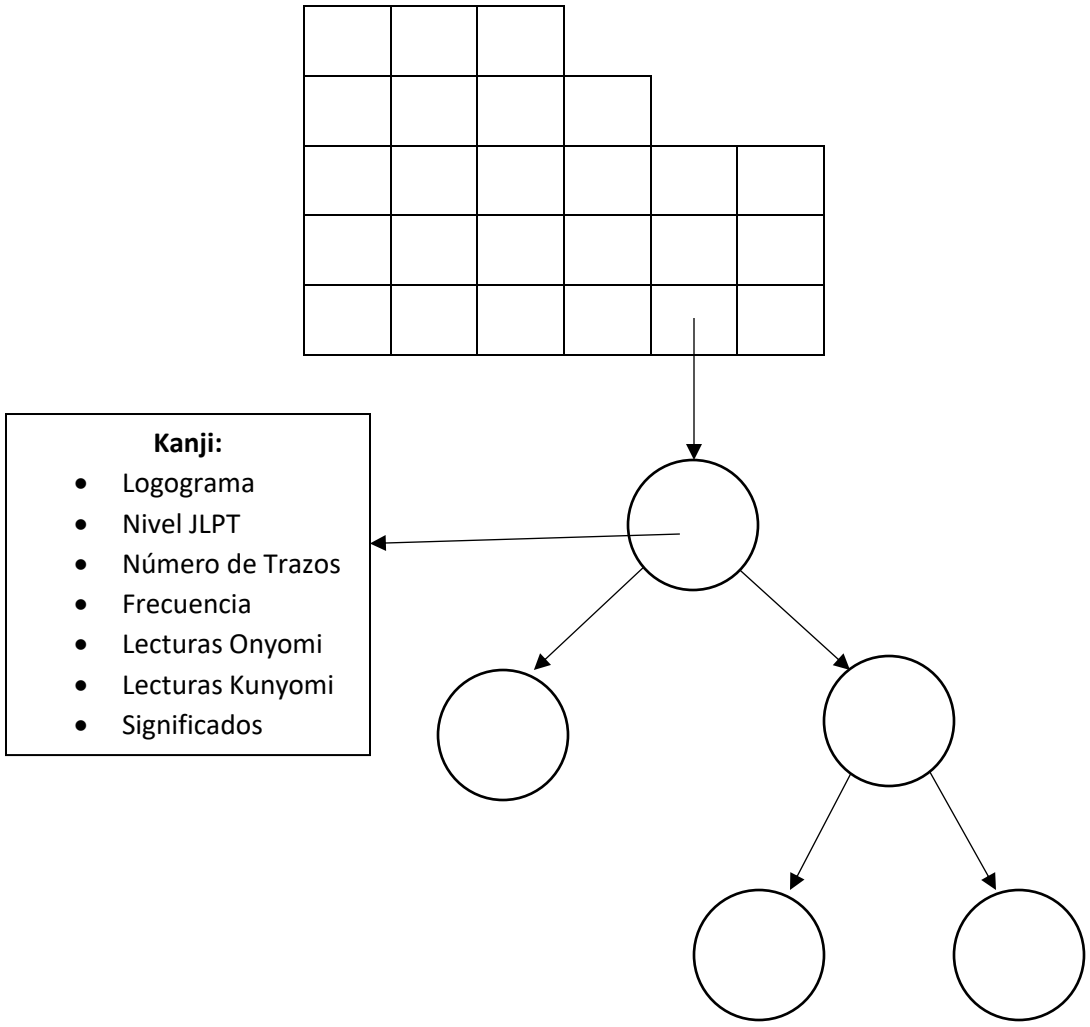
Para almacenar los kanji de una forma ordenada, se utiliza una *tabla hash* de dos dimensiones, construida de la siguiente forma (ilustrativa, no es la tabla final):



1.1 *SystemClassDiagram.png* se encuentra ubicado en el directorio *doc/diagrams* de la documentación en forma digital, o impreso al final de este manual.

La clase **KanjiTable** es la encargada de adminstrar la tabla, la cual es un arreglo bidimensional en el que cada fila no tiene el mismo número de columnas. Las filas representan el Nivel JLPT del kanji, y la segunda el Número de Trazos que componen al kanji. Cada fila posee un tamaño de columnas distinto dado que cada Nivel JLPT es distinto en número de trazos. El más grande en el nivel N5 alcanza los 14 trazos, mientras que el más grande en N1 alcanza los 24. Sin embargo, existen columnas añadidas para dar flexibilidad a este aspecto.

Cada espacio de la tabla es un objeto de clase **AVLTree**, el cual es un árbol binario equilibrado, que orderna los kanji por su Frecuencia. El uso del arbol AVL maneja las colisiones una vez que la función hash ha hecho el trabajo de dar la dirección de memoria donde se encuentran los kanji con un Nivel JLPT y un Número de trazos determinado, y además optimiza la búsqueda ya que la Frecuencia es un número único y ordinal.



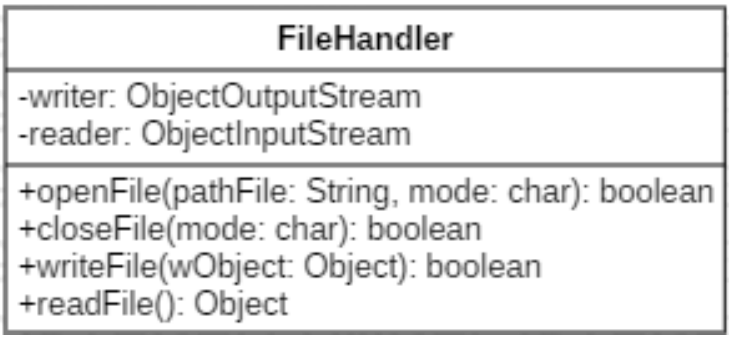
Para hacer uso de la tabla hash se genera una *clave* para cada kanji, la cual será única, y se genera utilizando el Nivel JLPT, el Número de Trazos y la Frecuencia:

Nivel JLPT # de Trazos Frecuencia

Los niveles permitidos son de 1 a 5, y el número de trazos para cada nivel está especificado en los arreglos *minStrokes* y *maxStrokes* de la clase **KanjiTable** (donde la posición 0 es el nivel N1). Las funciones en el programa que solicitan el nivel JLPT o el número de trazos de un kanji, establecen como inicio el 1, y dentro éste se convierte a índice empezando desde 0. Esta característica está indicada en un comentario al inicio de cada función.

Las excepciones **InvalidLevelException** y **InvalidStrokeCountException** se crearon para manejar por separado cuándo es introducido un nivel inválido y cuándo es introducido un número de trazos inválido, en lugar de utilizar la excepción de Java **ArrayIndexOutOfBoundsException**.

Por último, para permitir el guardado de archivos se creó la clase **FileHandler**, la cual posee un stream de entrada y un stream de salida, ambos estáticos, así como también lo son sus métodos:



Sólo se permite abrir un archivo a la vez, por lo que antes de abrir otro archivo se debe cerrar el anterior que estaba abierto. Todas las clases que se encuentran en el diagrama *SystemClassDiagram* implementan la interfaz **Serializable** de Java para poder ser guardadas en archivos. La extensión que se determinó para guardar los archivos que sean generados con el programa es *.skc.*

2. Desarrollo del Sistema

El programa permite las siguientes acciones:

- **Agregar Kanji**: Despliega un área donde el usuario ingresará la información de un kanji al sistema. La información que resulta obligatoria son los datos para generar la clave, y que no se podrán cambiar después de haberlo ingresado.
- **Búsqueda**: Despliega un área donde el usuario puede ingresar una clave para obtener un kanji en específico, o puede ingresar por separado el nivel o el número de trazos para desplegar grupos de kanji que cumplan con esos criterios. La búsqueda por criterios utiliza ambos criterios al mismo tiempo, por lo que la búsqueda es reducida si se especifican ambos.
- **Visualización de Detalles**: Permite ver los detalles de un kanji que fueron introducidos por el usuario. Los datos deben aparecer organizados para garantizar su rápida comprensión y poder ser utlizada como herramienta de estudio.
- **Modificación de los Datos**: Permite modificar las lecturas y los significados de un kanji, cuantas veces se desee, y la nueva información será guarda sobre el mismo objeto (instancia de **Kanji**). No se permite la modificación de los datos que generan la clave.
- **Guardar la Lista en un Archivo**: La lista puede ser guardada en un archivo en el dispositivo. El archivo de la lista posee un indicador de versión, con el que se garantizará la compatibilidad de archivos de versión más antigua a la versión más nueva, pero no al revés (no se puede viajar en el tiempo, obviamente).
- **Abrir Lista**: La lista se puede recuperar y seguir trabajando sobre ella.

2.1. Proceso de Agregar Kanji:

El usuario rellena cada campo por separado, y se comprueba que sean datos válidos. Una vez que lo sean, se genera un kanji con los datos proporcionados, y se obtiene su clave única. La clave única se busca en la tabla hash, y si existe, entonces el kanji no se agrega a la tabla, y se le marca al usuario una advertencia. Si la clave no existía, entonces se agrega a la tabla. La tabla registra el nuevo ingreso en la variable *sum*, que marca la cantidad de kanjis que posee la tabla.

Las lecturas y significados que ingrese el usuario están separadas por comas “,” o por el punto japonés “・”. Cuando el kanji ingresa al sistema, cada lectura y significado es separado de los demás en donde el usuario haya colocado dichos signos, por lo que lecturas y significados se almacenan en arreglos para mantenerlos separados.

Además, los anteriores pueden no especificarse y dejarse en blanco. Si esto ocurre, el sistema coloca por defecto al arreglo del campo vacío la leyenda “(No especificado)”, la cual no será guardada al archivo

2.2. Proceso de Búsqueda:

Por clave: El usuario ingresa una clave compuesta por una *N* al inicio, seguida de 7 dígitos. La clave primero es validada del nivel y del número de trazos; si no lo es, se le muestra una advertencia al usuario. Si es válida, se obtiene la posición de la tabla hash, y se procede a buscar la frecuencia dentro del árbol en esa posición. Si existe, se muestra ese kanji, si no, se muestra una advertencia.

Por características: El usuario especifica un nivel JLPT y/o un número de trazos, y se buscan todos los kanji que cumplan con esos criterios en la tabla hash. La tabla hash posee funciones que retornan ya sea: todos los kanji de todos los árboles en una fila determinada; todos los kanji de los árboles en las columnas para una número de trazos determinado (columnas que no están en la misma posición para cada fila), o todos los kanjis en un árbol de una fila y columna determinada.

Una vez obtenida la lista, se le muestra al usuario para que seleccione el kanji del que desee mostrar su información.

2.3. Visualización de Detalles:

Dentro de la sección de búsqueda, el usuario puede seleccionar un kanji para ver sus detalles. Al hacerlo, se muestran los detalles divididos en cuatro secciones: logograma (el kanji), detalles (nivel, número de trazos y frecuencia), lecturas (onyomi y kunyomi), y significados.

2.4. Modificación de los Datos:

En la visualización de datos se ofrece la opción de edición de datos, la cual solo permite modificar las lecturas y los significados del kanji. La operación reutiliza las operaciones el proceso de agregar kanji, excepto que esta vez no solicita los datos que conforman la clave. El resto del proceso es igual, excepto que en lugar de ser agregado a la tabla hash, simplemente se recupera cuál era el kanji original y se modifican los datos con los del nuevo kanji generado (kanji de clave idéntica).

2.5. Guardar la Lista en un Archivo:

Para guardar la lista de kanji en un archivo primero se guarda la versión de kanji que posee la clase **Kanji**. Luego, se guarda la lista de kanji transformando cada kanji a una cadena de texto, denominada *cadena de formato*, que especifica la información del kanji dividiendo cada campo y cada lectura y significado, La clase **Kanji** posee un constructor que permite crear un kanji indicando solo su cadena de formato.

Las cadenas de formato son guardadas en un **ArrayList**, y luego este objeto se escribe al archivo de forma binaria utilizando la clase **FileHandler**. El archivo es guardado con la extensión del programa en el directorio del sistema que el usuario especificó.

2.6. Abrir Lista:

Se obtiene la ruta del archivo que el usuario especificó, y primero se valida que el archivo es efectívamente un archivo creado con el programa. Si no lo es, entonces muestra una advertencia. Si el archivo es válido, entonces primero toma la versión del kanji para decidir como trabajarla (a la versión 0.1.1, aún no existe manejo de las versiones puesto que se trata de la primera versión del kanji).

Para la Versión 1 del Kanji, toma cada cadena de formato guardada y construye un objeto kanji con ella, el cual luego es agregado a la tabla hash.

2.7. Cadena de Formato

La cadena de formato de un kanji es la forma de representar con texto cada campo de un objeto de la clase **Kanji**. La versión actual a este manual (versión de kanji 1) se compone de la siguiente manera:

*logograma/nivel jlpt/número de trazos/frecuencia/lectura onyomi 1*lectura onyomi 2*...*lectura onyomi n/lectura kunyomi 1*lectura kunyomi 2*...*lectura kunyomi m/significado 1*significado 2*...*significado k*;*

Se utilizan los siguientes separadores:

- ***/*** : Separador de campos. Indica dónde termina cada uno de los campos de la clase kanji, para que la cadena pueda ser separada.
- ***** : Separador de lecturas y significados. Indica dónde termina cada una de las lecturas o significados del kanji. También indica si dicho campo está vacío, si este separador se encuentra sólo entre dos separadores de campos.
- ; : Separador de kanji. Indica dónde finaliza toda la información de un kanji.

El siguiente ejemplo muestra una constitución específica de una cadena de formato para un kanji:

保

Nivel: N1
Número de trazos: 9
Frecuencia: 146
Onyomi: ほ・ほう
Kunyomi: たもつ
Significados: Proteger, Conservar, Garantía

保/1/9/146/ほ*ほう/たもつ/Proteger*Conservar*Garantía;

La cadena de formato es una herramienta pensada para dar compatibilidad a futuras versiones del programa con listas de kanji hechas con versiones anteriores, en donde los cambios del programa también modificaron la clase **Kanji** en sus campos.

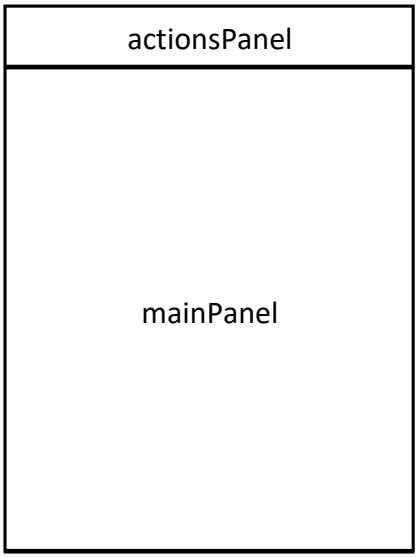
El objetivo es que se lean las cadenas de formato, y dependiendo de la versión, se maneje de una determinada forma para construir el kanji. Si hacen falta datos, puede construirse el kanji sin ellos, rellenando solo los campos que se tienen, y luego solicitar al usuario los datos que hacen falta.

De esta forma también es posible controlar a qué versiones de listas se les seguirán dando soporte (pues dar soporte a una versión acumula líneas de código). Si el usuario intentara abrir una lista de una versión muy antigua, puede suceder que ya no se abra. Es por esta razón que cada vez que se abra una lista de una versión antigua, el archivo se reescribirá con la versión actual que maneje el sistema, y se maximiza la compatibilidad a lo largo del tiempo.

3. Desarrollo de la Interfaz Gráfica

3.1. Distribución del entorno:

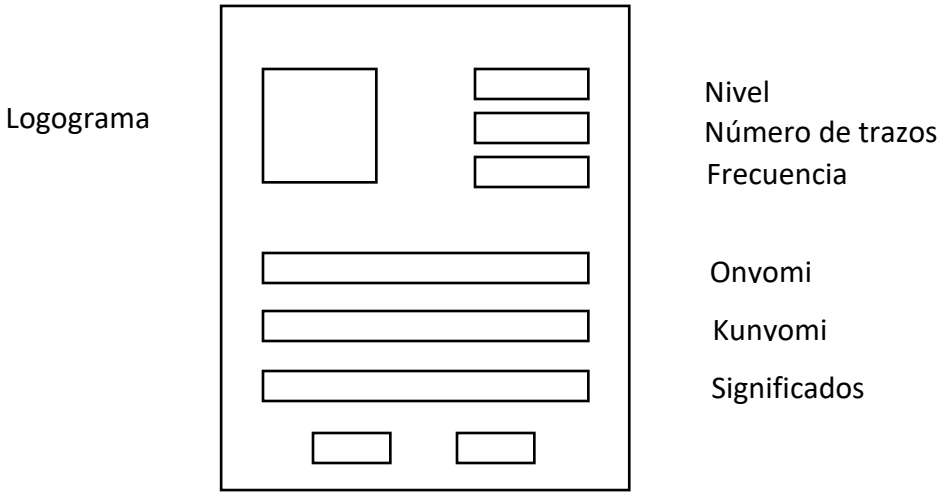
La aplicación se pensó como una aplicación pequeña, ligera y fácil de usar, y que además no invada tanto espacio en la pantalla del dispositivo donde se utilice. Por esa razón se eligió crear una ventana vertical, con dentro solo dos paneles: *Panel de Acciones* y *Panel Principal (de Visualización)*, que se colocaron como se muestra a continuación:



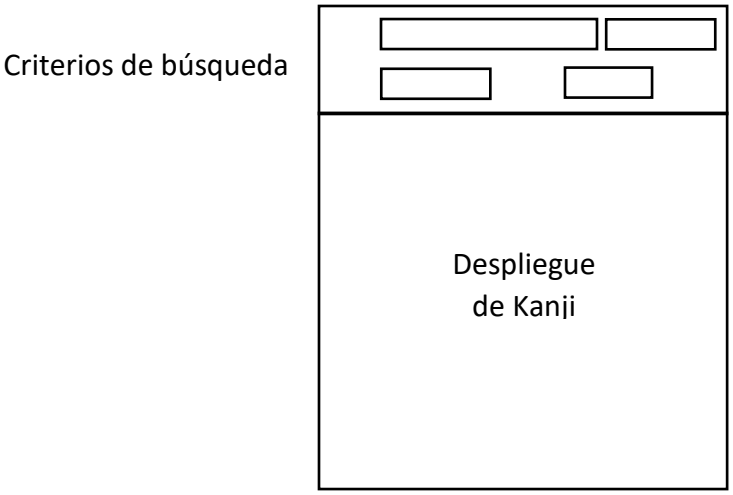
Dentro del panel de acciones se colocarán los botones que requiera tener una sección. Se procura que sean pocos botones. Dentro del panel principal se colocarán los distintos paneles de visualización (panel para agregar un kanji, para buscar, etc.). Los paneles están dentro de la ventana principal (JFrame) que tiene una dimensión fija de *540 x 720*.

Se crearon 3 clases para manejar las primeras 4 acciones expuestas en al inicio de la **sección 2**, cada una con una estructura diseñada para cada función. Estas clases son:

- **AddKanjiPanel**: Panel con areas de texto donde el usuario ingresa los datos de un kanji. Posee dos botones, para aceptar los datos ingresados o para cancelarlos. Adicionalmente, se utiliza en la modificación del kanji, donde en este caso específico los campos rellenan antes con la información del kanji, y los campos de la información que compone la clave se bloquean para evitar que sean modificados.
  El panel se compone de la siguiente forma:



- **SearchKanjiPanel**: Panel conformado por dos páneles más: uno que contiene áreas de texto y un selector para permitir la búsqueda por clave o por características; el otro panel desplegará una cuadrícula de kanji (ver clase **KanjiGridPanel**)
  El panel se compone de la siguiente forma:



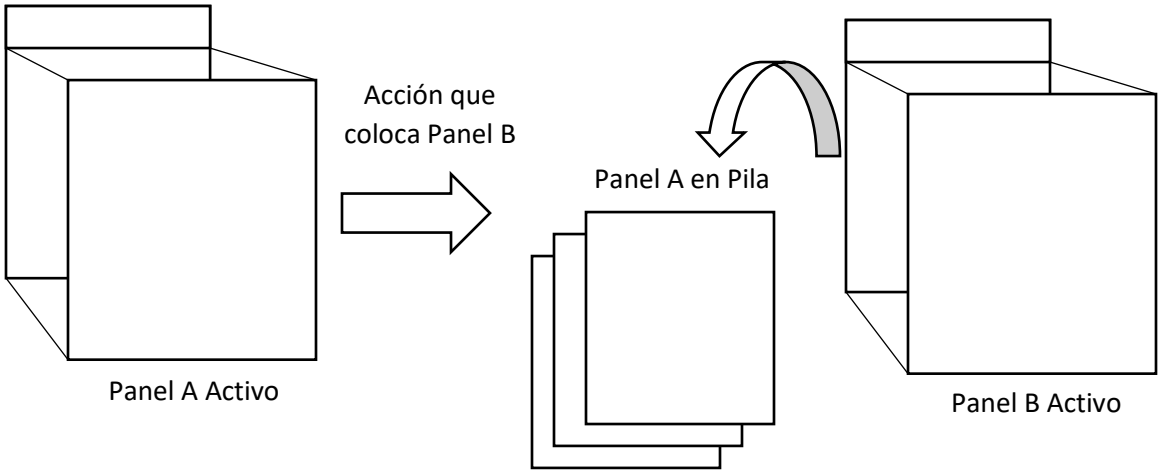
  Adicionalmente se tiene la clase **KanjiGridPanel**, que es introducida en la sección de “Despliegue de kanji”. Consta de una cuadrícula de paneles, donde se muestran todos los kanji que cumplan con los criterios de búsqueda. Si esta lista supera los 25 elementos, la cuadrícula se extenderá hacia abajo y aparecerá una barra de desplazamiento.
- **KanjiPanel**: Este panel muestra la información de un kanji cuando es seleccionado en la cuadrícula. Cambia los botones del panel de acciones para dar solo las opciones de regresar a la búsqueda o editar el kanji que se está mostrando. El botón de editar vuelve a desplegar un panel tipo **AddKanjiPanel**.

3.2. Manejo de paneles en secuencia:

La mayoría de las acciones que requieren intercambiar paneles en el panel de visualización lo hacen creando nuevos objetos de la respectiva clase del panel que se ha de colocar; esto produce que los datos de los anteriores paneles también se pierda (se pierde la referencia).

Si es necesario guardar un panel para después poder recuperarlo con la última información que poseía, existe la varíable *lastPanels*, que es instancia de una pila hecha especialmente para guardar paneles (ver clase **PanelStack**). De esta forma se pueden guardar cuantos paneles sean necesarios.

Esta operación se utiliza para conservar la información del panel de búsqueda cuando un kanji es seleccionado para desplegar sus detalles; el usuario puede regresar a la pantalla de búsqueda donde se encontraba en lugar de volver a ingresar los criterios de búsqueda. De la misma forma, cuando se decide editar un kanji, el panel de detalles que estaba activo se almacena, para en caso de que el usuario decida cancelar la operación, se pueda recuperar el panel ya construído.



3.3. Fuentes:

El programa utiliza determinadas fuentes en cada componente gráfica que maneje texto, tanto por estética, como para manejar los caracteres especiales. Las fuentes se hayan en una carpeta junto con el programa (directorio *fonts*) para que el usuario no tenga la necesidad de instalar estas fuentes en su sistema. Se manejan 3 fuentes (todas de uso libre):

- Meiryo font
- Yu Gothic font
- DFKaiSho-SB font (fuente de Microsoft para el idioma chino)

Las fuentes se cargan al programa abriendo el archivo con la clase **File** de Java, y posteriormente se crea una fuente con la clase **Font**, también de Java, para así mantenerla guardada en el sistema mientras el programa está en ejecución.

Las fuentes son objetos con acceso privado de la clase **MainWindow** (no del objeto), por lo que se debe acceder a ellas a través de sus respectivas funciones *get*.

No se utilizan solo tres objetos **Font**, si no que cada objeto de la clase **Font** en el programa tiene una fuente y un formato específico, para cumplir una tarea que se requiera en el programa. Cada vez que se cree una componente con texto, se debe cambiar su fuente a una de las establecidas para el programa (o en su defecto, si es necesario, definir una nueva).

Si los archivos para las fuentes no estuvieran en la carpeta *fonts*, el programa lanza una advertencia de ello e inmediatamente se cierra, pues se generarían problemas de visualización. Si esto ocurre, el manual de usuario da solución al problema, dando acceso nuevamente a las funciones para volver a ser agregadas.

3.4. Manejo de Eventos:

El manejo de eventos en este programa se realiza por medio de clases anónimas y se centraliza en el objeto de la clase **MainWindow**, es decir, que cada acción a realizar cuando ocurre un evento es manejada en un método propio del objeto *mainWindow*.

3.4.1. Asignación de Escuchadores de Eventos:

Cuando una componente requiera de tener un escuchador de eventos (independientemente del tipo) éste debe ser asignado dentro de los métodos de *mainWindow* (salvo algunas exepciones), utilizando la definición por clase anónima; por ejemplo:

```
addButton.addMouseListener(new MouseAdapter() {
  public void mouseClicked(MouseEvent evt) {
    addButtonMouseClicked();
  }
});
```

Se crea una clase anónima de algún *Adapter*, y se redefinen los métodos que sean necesarios. Cada método redefinido deberá llamar a una función que se deberá crear exclusivamente para manejar las acciones cuando ocurra dicho evento, y el identificador de este método se nombra colocando primero el nombre del objeto al que se le añade el escuchador, seguido del nombre del evento (método que se redefine). Este nuevo método no es *static*, por lo que pertenece al objeto instanciado, y su acceso es privado.

Este manejo de eventos organiza el programa de tal forma que solo *mainWindow* es capaz de realizar cambios sobre el programa, haciendo que cuando otro objeto reciba un evento, simplemente “avise” a *mainWindow*. También hace que los eventos se concentren en un solo lugar, y sea más fácil administrarlos. No

es necesario que todo la acción esté dentro del método para el evento; éste puede llamar a métodos de otras clases para concretar la acción, pero el control del programa siempre regresará a *mainWindow*.

Además, esto evita que el método creado tenga que recibir parámetros, pues solo módifica los atributos de *mainWindow*, o llama a los métodos de otros objetos. El único parámetro que puede ser necesario es el evento generado por Java, si es que el método es ocupado por más de una componente (por ejemplo, los botones en la cuadrícula de Kanji).

3.4.2. Excepciones de la asignación:

Hay algunas excepciones para la asignación de escuchadores de eventos: si el escuchador solo trabajará bajo los campos de la clase donde se define, entonces no es necesario asignar el escuchador en **MainWindow**, se asigna dentro de la clase que lo contiene, y el método que se crea también es parte de dicha clase; si el escuchador debe ser asignado fuera de **MainWindow**, pero trabajará sobre los atributos de ésta, entonces se deberán crear dos métodos: uno *static* en **MainWindow**, que en su cuerpo solo contiene la llamada al segundo método, el del objeto *mainWindow*, y en la asignación del escuchador, el objeto llamará al metodo *static* (véase la clase **KanjiGridPanel** y la asignación de escuchadores de eventos del mouse para cada botón de la cuadrícula).

```
private void initKanjiGrid(ArrayList<Kanji> kanjiGroup) {
  int i = 0;
  for (Kanji kanji : kanjiGroup) {
    KanjiButton newButton = new KanjiButton(kanji);
    newButton.addMouseListener(new MouseAdapter(){
      public void mouseEntered(MouseEvent evt) {
        newButton.setBackground(new Color(252, 199, 207));
      }

      public void mouseExited(MouseEvent evt) {
        newButton.setBackground(new Color(255, 241, 243));
      }

      public void mouseClicked(MouseEvent evt) {
        MainWindow.kanjiButtonMouseClicked(evt);
        newButton.setBackground(new Color(255, 241, 243));
      }
    });

    add(newButton);
    i++;
  }
```

(en amarillo: los eventos no alteran a mainWindow*, en rojo: llamada al método* static*)*

```
public static void kanjiButtonMouseClicked(MouseEvent evt) {
  mainWindow.kanjiGridMouseClicked(evt);
}

public void kanjiGridMouseClicked(MouseEvent evt) {
```

(declaración del método static *y la llamada al método del objeto)*

3.5. Imágenes del programa:

Se utilizan imágenes en el programa para modificar el ícono de la aplicación y colocar el logo en el panel principal. Éstas imágenes se encuentran en la carpeta *lib*, y son cargadas al programa utilizando las clases **Image** y **ImageIcon** de Java. No se utilizan imágenes en botones o en alguna otra parte del programa.

4. Limitaciones

4.1. De Almacenamiento:

El programa está pensado para apoyar a un estudiante de japonés que está aún por debajo de cualquiera de los niveles que marca el Examen de Aptitud del Idioma Japonés (JLPT); sin embargo, dicho examen solo abarca el manejo de los 2,136 kanji que se conocen como “Kanji de uso regular”, aunque no son todos los que se utilizan en el idioma. Cualquiera que esté fuera de esta lista no será posible almacenarlo, pues no cumple con ninguna característica que solicita el programa.

No obstante, esta lista se considera suficiente para leer un periódico japonés, una novela, o cualquier otro texto de la vida cotidiana. Si se llega a utilizar un kanji que no esté dentro de esta lista, existe la regla de acompañarlo con su lectura (una ayuda de lectura denominada *furigana*), por lo que el estudio de esos caracteres se limita a grados superiores, como posgrados universitarios.

A su vez, en el mantenimiento de esta aplicación se debe estar al pendiente de los posibles cambios que puedan surgir en las listas de kanji evaluadas en cada nivel del JLPT, puesto que pueden cambiar de orden y alterar los límites de los arreglos delimitados por el número de trazos (las columnas), para poder cambiar la estrutura de la tabla hash como sea adecuado.

4.2. De Funcionalidad:

De momento, la aplicación no posee una función para eliminar un kanji de la lista, por lo que una vez agregado y guardado el archivo, el kanji no puede ser removido, solo editado. Esto debido a que se utlizan árboles AVL para último paso del almacenamiento del kanji, y la eliminación de éstos árboles resulta muy compleja. Se trabaja ya en ideas para sustituír de otra forma a la eliminación, permitiendo que un kanji pueda ya no estar almacenado en la lista.

La solución de este problema también lleva a la solución del problema de la edición de los campos que determinan la clave del kanji; no es posible editar los datos que conforman la clave, dado que se debería eliminar el kanji del árbol en el que se encuentra, y agregarlo a otro en la tabla.

La única vía posible (mas no la deseada) para llevar a cabo estos procesos, es editar la lista sin guardarla, luego comprobar que los datos son correctos. Si no lo son, se cierra la lista sin guardar, lo que hace que el nuevo elemento nunca se escriba al archivo.

4.3. De Compatibilidad:

La aplicación está desarrollada sin hacer uso de IDE, en un dispositivo con Windows 10, por lo que no está optimizada para trabajar gráficamente en otros sistemas operativos. Puede que la visualización de los elementos de la ventana sea distinto si se trabaja en otro sistema operativo (el funcionamiento interno no se ve afectado).

4.4. De Portabilidad:

La aplicación solo está disponible como aplicación de escritorio, y se requiere que el sistema operativo tenga instalada la Máquina Virtual de Java para poder funcionar.

4.5. De Conocimiento del Área (para desarrolladores):

Esta aplicación está dirigida a un público muy específico, por lo que se utilizan en todo momento conceptos especiales que requieren un poco de estudio para comprenderlos.

Es recomendable también conocer un poco de la *curva de aprendizaje* de este idioma, y del proceso que utilizan tanto escuelas como personas autodidactas al aprenderlo, para así poder comprender mejor el funcionamiento y objetivo de la aplicación, y poder idear nuevas funcionalidades e implementarlas en futuras versiones.

5. Trabajo a Futuro:

5.1. Prueba y Correción de Errores:

La aplicación se ha probado hasta ahora con 4 estudiantes de japonés utilizando la aplicación por una semana. No se han presentado errores, tanto en GUI como en el funcionamiento del programa. Sin embargo, se considera seguir haciendo pruebas, con más estudiantes y durante más tiempo, que utilicen distintos dispositivos, y que trabajen con distintas técticas de estudio, para así detectar errores, incompatibilidades, y necesidades que esta herramienta aún no satisfaga dentro de sus objetivos.

5.2. Nuevas Funcionalidades:

5.2.1. Borrar Kanji de la Lista:

Se trabaja en una forma eficiente y sencilla de lograr la eliminación de un kanji de la lista.

Una solución posible es implementar los algoritmos de eliminación para árboles AVL, los cuales retiran el nodo del árbol, y en seguida balancean el árbol de ser necesario. Estos algoritmos no son tan sencillos de manejar, además de que trabajan con muchos casos posibles, por lo que su implementación puede llevar más tiempo del deseado.

Otra solución que se considera es trabajar con un sistema de marcado, en el que el nodo que se desea eliminar en realidad no es retirado del árbol, sino que solo es marcado como “eliminado”. El nodo permanece en memoria, pero es ignorado por toda función del programa (incluído el guardado en archivo), y será reescrito cuando la inserción de un nuevo kanji pueda ocupar su lugar (coincidiendo en nivel, número de trazos, y teniendo una frecuencia que pueda reemplazar la de dicho nodo). Si ninguna nueva inserción llega a reemplazar al nodo, éste simplemente desaparecerá cuando el archivo vuelva a ser guardado en el archivo, por lo que cuando se vuelva a abrir, ya no se encontrará ni en el archivo, ni en memoria.

Se deja puerta abierta a otras alternativas.

5.2.2. Guardar Kanji en otra lista:

Los kanji pueden ser categorizados de muchas maneras, por lo que un estudiante de japonés puede desear crear listas que lleguen a compartir kanji. Se considera dar la opción al usuario de que, dado un kanji en una lista abierta, pueda incluirlo en otro archivo, sin necesidad de abrir el archivo de esa otra lista y volver a introducir los datos del kanji para agregarlo.

Esta opción debe dar la facilidad de agregar los kanjis a cuantas otras listas se desee, pero debe siempre analizar si el kanji ya existe dentro de ese archivo al que se quiere agregar.

5.2.3. Sistema de Estudio de las Tarjetas:

Esta es una de las funcionalidades principales que se desea agregar al sistema antes de concretarse como una herramienta al estudio. Hasta el momento las funciones que ofrece la aplicación son más que nada de consulta, que ya dan al usuario facilidad para organizarse; pero aún no se tiene un medio por el cual sea capaz de generar aprendizaje.

Las Tarjetas de Aprendizaje (también conocidas como *flashcards*) son una herramienta ampliamente utilizada en distintos ámbitos, uno de ellos, el aprendizaje de un idioma. Este método se basa en generar tarjetas, donde por un lado se tiene una pregunta, y por el otro su respuesta. En este caso en concreto, en lugar de una pregunta, se tiene por un lado el logograma que se desea aprender, y por el otro el significado que representa (o uno de ellos).

Se planea introducir al programa una actividad que maneje las Tarjetas de Aprendizaje para que el usuario practique con ellas de manera más interactiva. El programa tomará los kanji que tenga la lista, y de manera aleatoria los mostrará en pantalla; el usuario deberá responder cuál es el significado de dicho kanji que se mostró, y se registra si su respuesta fue correcta o errónea.

La forma de ingresar la respuesta aún está en proceso de decisión; puede ser introducida de manera manual, haciendo que la cadena de texto coincida exactamente con la del significado; o también puede optarse por mostrar múltiples opciones, entre ellas la correcta (también podría optarse por introducir ambos).

Así mismo, podría realizarse esta función de forma inversa: se muestra un significado, y el usuario selecciona el kanji adecuado.

Esta función sigue en proceso de estudio para encontrar la forma más adecuada de que el usuario pueda estudiar sus listas de forma interativa, sencilla, rápida y además se le ofrezca algún tipo de retroalimentación.

5.3. Lanzamiento de la Aplicación:

Se planea lanzar esta aplicación al público una vez se considere que representa una herramienta de utilidad para el estudio de los kanji, lo que conlleva implementar las tres funciones descritas anteriormente, además de realizar las pruebas correspondientes para evitar errores graves.

Ápendice: Diagrama de Clases