Como no he encontrado ninguna librería para dibujar grafos que me fuera útil he decidido intentar construir el grafo a partir de rectangle shapes , lines y labels.

ScrollPane tiene eventos por defecto para las teclas izquierda y derecha por lo que hay que añadirle un event filter para que atienda a nuestro evento antes que al que esta por defecto. Aun con esto el evento por defecto no desaparece.

grafo.addEventFilter(KeyEvent.KEY\_RELEASED,

event ->handleKeyAction(event));

nodo raíz situado en el punto medio de la suma de los 17 nodos -1 que es la raíz ya que en el peor caso la raiz tiene número de nodos-1 hijos.

## Clase Nodo V0.1

Contiene once atributos:

**private String símbolo**: es el símbolo que se va a representar en este Nodo.

**private Rectangle rectangle**: es el rectángulo sobre el que va a estar el símbolo y representa al nodo.

**private Nodo parent**: es el nodo padre.

**private LinkedList<Nodo> children**: son todos los hijos del nodo, estos hijos representan los nodos que se van a ejecutar a causa de este nodo.

**private Double posX**: posición en el eje X del nodo.

**private Double posY**: posición en el eje Y del nodo.

**private Line line**: línea que va a juntar el nodo padre con este nodo. (Pendiente de revisión)

**private Label label**: etiqueta que va a mostrar el símbolo del nodo.

**private HashMap<String , Nodo> hermanos**: mapa de nodos que serán los nodos en la pila que suelen aparecer al lado del nodo ejecutado. Están insertados en función de su símbolo.

**private Nodo LeftSibling**: hermano izquierdo ya ejecutado del nodo.

**private Rectangle rectRgla**: rectángulo que engloba a todos los nodos de la regla.

**public Nodo(String simbolo,Nodo parent)**

En el constructor se asignan el símbolo y el padre recogidos y se inicializa los hermanos, los hijos y el rectángulo del nodo con un tamaño 50x50 (Provisional)

## Clase Grafo V0.1

Tiene tres atributos:

**private HashMap<Integer,Nodo> nodos**: es un mapa que guarda los nodos ejecutados en función de paso en el que aparecieron .

**private FicheroXML ejemplo**: guarda los datos obtenidos del fichero XML que tiene los datos del traductor.

**private double posXAnterior**: guarda la posición en el eje X del ultimo nodo ejecutado que se ha insertado. Se utiliza para evitar que los nodos se solapen.

**public Grafo(FicheroXML xml)**

El constructor recoge el fichero XML y lo asigna, además, inicializa el mapa de nodos y la posición anterior como la mitad del numero de nodos -1 por el tamaño de los nodos. (Pendiente de revisión, ya que si el nodo raíz tiene una regla podemos saber cuántos hijos tendrá y calcular mejor su posición).

**public Nodo insertarNodo(Nodo parent,Pane panelPadre,String simbolo,Double posX,Double posY)**

Recoge el nodo que va a ser el padre, el panel donde se va a dibujar, el símbolo correspondiente al nodo y las posiciones en el eje X e Y.

Crea el nodo, asigna el color y la posición del rectángulo y la guarda. Crea la etiqueta con el símbolo, le asigna la posición y guarda la etiqueta. Después guarda el nodo en el mapa de nodos. A continuación, si no es nodo raíz crea la línea que junta el padre con el nodo, la asigna al nodo y pinta en el panel el rectángulo, la etiqueta y la línea. Si el nodo es raíz solo dibuja el rectángulo del nodo y la etiqueta.

Por último, devuelve el nodo creado.

**public HashSet<Nodo> insertarNodoNotExec(Nodo parent,Pane panelPadre,Regla regla,Double posX,Double posY,Nodo hijo)**

Recoge el padre de los nodos a insertar (por revisar), el panel donde se van a dibujar, la regla de donde vienen sus símbolos, las posiciones en el eje X e Y y el nodo ejecutado que va a permanecer como primer hermano.

Inicializa el conjunto de nodos que se va a devolver y crea un array con todos los símbolos de la regla. A continuación, por cada uno de los símbolos si no corresponden al símbolo del primer hermano, se crea un nodo con ese símbolo, se le asigna el color, una opacidad del 50%, una posición en el eje X que será la posición x introducida mas un pequeño espacio de 10 px mas el ancho del nodo y la posición en el eje Y introducida. Después se crea la etiqueta, se le asigna su posición y se guarda en el nodo. Por último, se guarda el nodo en el conjunto que se va a devolver, en el mapa de hermanos y se dibuja en el panel el rectángulo y la etiqueta.

Devuelve el conjunto de nodos no ejecutados insertados.

**public HashSet<Nodo> insertarNodoNotExec(Pane panelPadre,Collection<Nodo> nodosHermanos,Double posX,Double posY,Nodo hijo)**

Recoge el panel sobre el que se pinta, la colección de nodos no ejecutados que se va a repintar, la posición en los ejes X e Y y el primer hermano.

Funciona igual que el anterior salvo porque no se crea el nodo ni la etiqueta, sino que se recolocan.

**public void eliminarNodo(Nodo nodoElim,Pane panelPadre,int number)**

Recoge el nodo a eliminar, el panel sobre el que se va a pintar y el paso en el que está el traductor.

Primero se borran el rectángulo, la línea y la etiqueta del panel. Después, si no es un nodo raíz se elimina el nodo de los hijos del padre, y si el nodo a eliminar es el primer hijo de la raíz se le pone una posición X por defecto (este caso diez, o cero, está por decidir), si no lo es se le la posición X anterior. Por último, se elimina el nodo del mapa de nodos.

**public LinkedList<Nodo> eliminarNodoNotExec(Nodo nodoElim,Pane panelPadre)**

Recoge el nodo ejecutado que corresponde al primer hermano y el panel del que se va a borrar.

Inicializa la lista de nodos que se va a devolver. A continuación, por cado nodo de los hermanos no ejecutados del nodo ejecutado se añade a una lista. Si esta lista no esta vacía por cada uno de los nodos de la lista se borra el rectángulo y la etiqueta del panel.

Devuelve la lista de nodos no ejecutados que se han borrado.

**public void construir(int contador,int pasoSolicitado,Pane panelPadre )**

Recoge el paso desde el que se va a construir, el paso hasta el que se va a construir,no incluido este, y el panel sobre el que se va a pintar.

Por cada paso ,desde el paso inicial introducido(contador) hasta el solicitado, si es el primer paso (correspondiente a la raíz del árbol) se inserta un nodo y se pone la posición anterior a un valor por defecto (0 o 10 , está por decidir), sino, se recupera el padre del nodo que se va a insertar, si el padre no tiene hijos, se inserta el hijo en la posición X anterior y la posición Y será el doble de la altura del rectángulo más la posición Y del padre, se añade el nodo a los hijos del padre y se obtiene la regla correspondiente al nodo, si esta regla nos nula se insertan los nodos no ejecutados correspondientes al lado del nodo ejecutado. Después se inserta el rectángulo correspondiente a la regla con el ancho del rectángulo del nodo más el de todos los nodos no ejecutados hermanos y se actualiza la posición anterior a la del nodo ejecutado insertado.

Si el padre ya tenía hijos, inserta el nodo en la posición X anterior más el ancho del rectángulo más una distancia de separación por defecto y la posición Y será el doble de la altura del rectángulo más la posición Y del padre y se añade el nodo a los hijos del padre. Después se eliminan los nodos no ejecutados del hermano izquierdo y se insertan los de este nodo. A continuación, se obtiene el rectángulo de la regla que tiene el hermano ejecutado izquierdo y si el primer hermano ejecutado no está en la posición por defecto se le añade al ancho del rectángulo la distancia del ultimo nodo no ejecutado del anterior nodo ejecutado más el ancho del nodo ejecutado más el ancho de los nodos hermanos no ejecutados más las distancias de separación menos la posición del primer hermano ejecutado. Si el nodo está en la posición por defecto se le añade el ancho del nodo más el ancho de los hermanos no ejecutados más las distancias de separación. Se guarda el rectángulo en la regla y se actualiza la posición anterior.

**public void eliminar(int contador,int pasoSolicitado,Pane panelPadre )**

Recoge el paso inicial desde el que se va a empezar a borrar, el paso hasta el que se borra y el panel del que se van a borrar los nodos.

Por todo el intervalo de pasos desde contador -1(revisar) hasta el paso final si el nodo que se va a eliminar no es nodo raíz y es el primero de los hijos se eliminan los nodos hermanos no ejecutados, después se busca el nodo no ejecutado del hermano izquierdo que corresponde al símbolo del nodo a eliminar, si existe dicho nodo se vuelve a pintar en el sitio que le corresponde. Por último, se elimina el nodo a eliminar.

**public Nodo NodoNotExec(Nodo nodo)**

Recoge el nodo que se va a buscar.

Si el nodo no es la raíz se busca en los hermanos no ejecutados del hermano izquierdo.si no se encuentra dicho nodo devuelve null.