**Documentación de Snake**

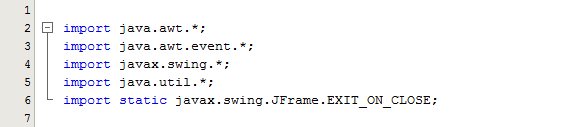
Importando Herramientas de UI

AWT: Abstract Window Toolkit, o algo así como Juego de Herramientas Abstractas para Ventanas. Son clases que permiten generar entornos con componentes gráficos comunes a todas las plataformas, y gestionar eventos de teclado y ratón, entre otros. El aspecto visual sí es dependiente de la plataforma.

Swing: Proporciona componentes de presentación visual independiente a la plataforma en la que se ejecuta. Swing extiende AWT y añade nuevas características, mejoras y componentes para interactuar con el usuario, tales como árboles, pestañas, tablas, etc. Estas clases fueron construidas junto con Netscape.

El paquete java.awt proporciona una serie de clases e interfaces para realizar programas que se ejecuten en un entorno gráfico.

Por ende antes de iniciar el desarrollo de nuestro juego es importante importar las herramientas necesarias para gestionar la parte gráfica. El uso del \* en java.awt.\* significa que estamos importando todos los elementos de esa herramienta de la misma forma con java.awt.event.\*, etc.



La clase Snake deriva de JFrame de manera en la que sea una JFrame pero con las características que deseemos, ya que si creamos nuestro juego utilizando una JFrame de swing no nos permite modificar sus propiedades porque java los declara como instancia privada.

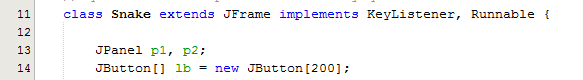
Seguidamente utilizamos implements para aplicar interfaces de java en la clase. Las interfaces de nuestro interés son KeyListener que es la interfaz encargada de hacer que nuestra clase sea sensible a eventos del teclado, de esa forma utilizamos inputs del teclado y Runnable.   
La interfaz Runnable debe ser implementada por cualquier clase cuyas instancias están destinadas a ser ejecutadas por un hilo o Thread.

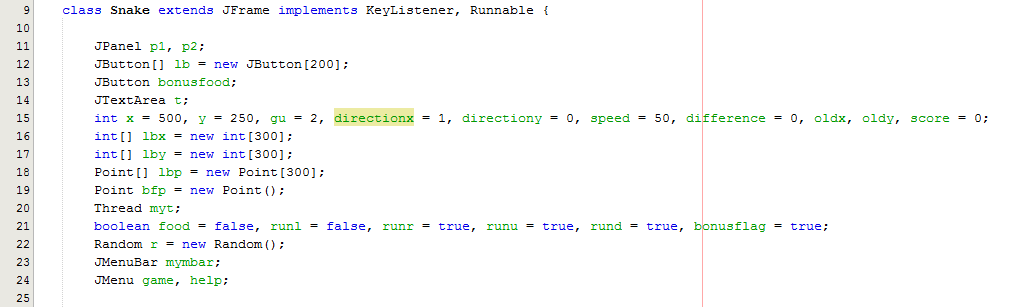
Para indagar mas sobre la interfaz KeyListener véase:

<https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyListener.html>

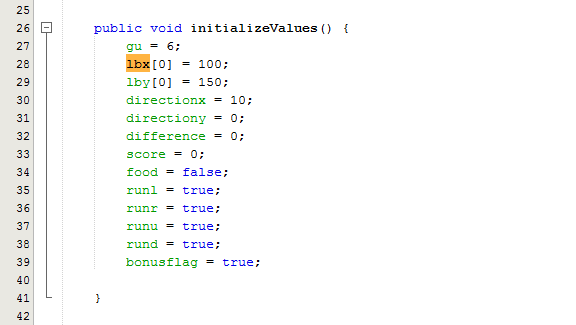
Para indagar más sobre la interfaz Runnable véase:

<https://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html>



Inicializamos las variables que necesitaremos como atributos de la clase Snake:

Declaramos p1 y p1 de tipo JPanel refiriéndose a Panel 1 y a Panel 2 que ocuparan un espacio en la ventana. Tambien se necesita un arreglo de tipo Jbutton que será el arreglo de botones que conforma a nuestro snake. Bonusfood es el botón que se instanciará para la comida del snake. La variable de tipo JTextArea es la que utilizamos para incluir texto en cualquier Panel. Las variables de tipo int serán necesarias para definir valores cuánticos en la clase snake, tales como gu refiriendose al tamaño del snake, speed refiriendose a la velocidad con la que descanza el thread, score refiriendose al puntaje y la dirección del snake ya sea de izquiera a derecha en x o arriba hacia abajo en y. Los arreglos lbx y lby de tipo int que representan las coordenadas x y de nuestra ventana. Tambien necesitamos declarer un Thread o hilo, necesario para el ciclo del juego. Las variables booleanas son para validar la dirección en la que el usuario desea dirigir al snake. Finalmente el JmenuBar y sus elementos.



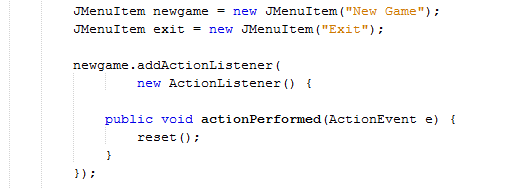
La clase Snake necesita una función pública que solo se encarga de iniciar los valores de cada variable declarada, también es útil para cuando se requiera reiniciar el juego.

Barra de Menu

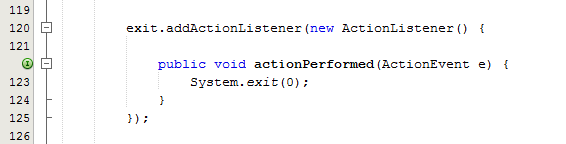
La función createbar() es la encargada de crear la barra de menú y sus componentes

Primerio se instancia un JmenurBar y se procede a instanciar sus elementos o pestañas de la barra de menú.

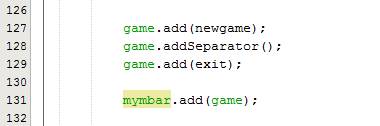
La instancia game es instanciada utilizando el constructor de la clase JMenu que recibe como parámetro un String que define el Label de la pestaña. También se debe instanciar los ítems dentro de la pestaña, newgame será el ítem encargado de reiniciar el juego.



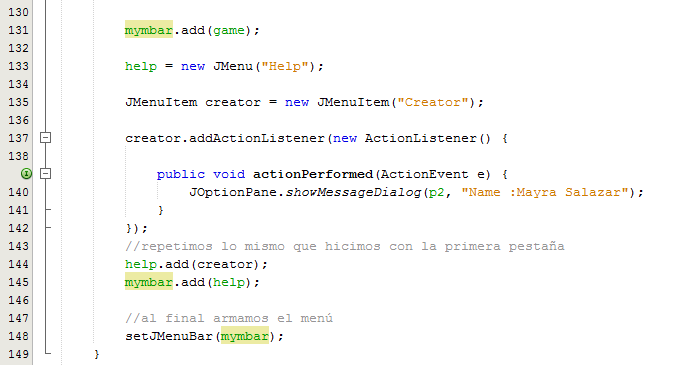
Cada ítem tiene una propiedad llamada actionListener refiriendose a la acción que debe realizar una vez que activemos el evento clic de cuando lo seleccionamos. La acción que realizara newgame será la de llamar el evento reset() el cual reinicia el juego.



El ítem exit simplemente se encarga del llamar el evento éxito exit de la ventana.

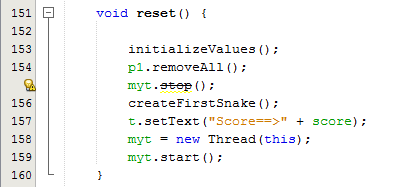


Luego de estructurar las partes de esta pestaña “Game” es necesario agregar los ítems a la tab game y por cada ítem agregar un separador. Y al completarlo se añade la tab a la barra de menú.



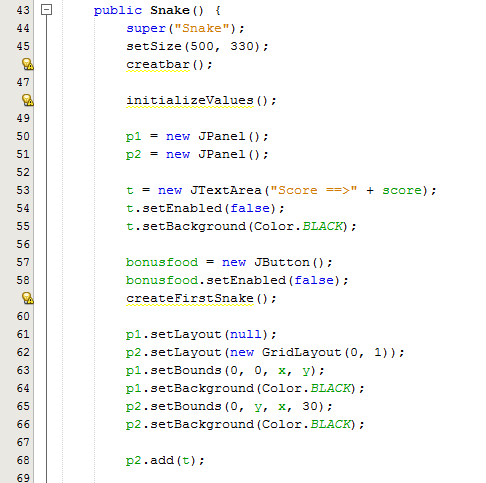
Para crear la siguiente pestaña es realmente el mismo procedimiento, la diferencia son los labels y el evento accionado, en este caso mostrar un JOptionPane con un Dialogo, para esto se utiliza la función showMessageDialog() que recibe como parámetro un panel y un String que representa el mensaje que se mostrará.

EL metodo reset()



EL método reset utilizado para el evento de New Game simplemente llama el método de initializeValues() para restablecer los valores de las variables, detener el hilo, instanciar una nueva snake, setear las puntuación del Texto y crear un nuevo hilo que cumplirá su función como nuevo ciclo del juego.

Instanciando la clase Snake



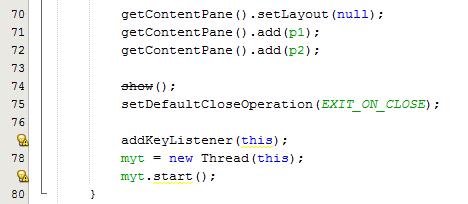
La clase Snake define el tamaño de la ventana, ya que la clase Snake hereda de JFrame esta misma contiene la función setSize que nos permite definir las dimensiones de la ventana.

Ya que existe nuestra función para crear la barra de menú simplemente se llama ese método e inicializa las variables.

Se setea el valor del texto con el score inicializado en este caso 0, se instancia la comida del snake y se llama el método que instancia el snake.

Una buena práctica de programación al trabajar con UI es vaciar el panel en el que añadiremos instancias para evitar duplicaciones o errores.

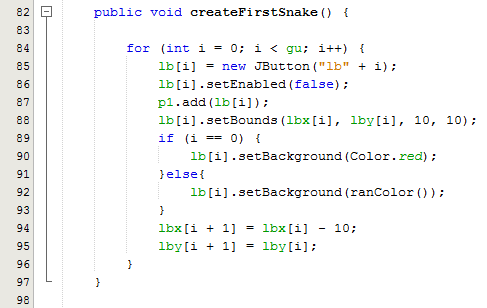
Luego se añaden los elementos correspondientes a cada panel, en p1 se definen las dimensiones del panel con las variables definidas de X y Y como parámetro y se le setéa su color de fondo, mismo procedimiento para p2 pero se le añade la variable de texto.



Finalmente misma practica vaciando el contenido de la ventana y añadiendo el panel 1 y el panel 2 a la ventana, es importante llamar la función setDefaultCloseOperation() recibiendo de parámetro “EXIT\_ON\_CLOSE” para que la ventana pueda cerrar al dar clic en la x.

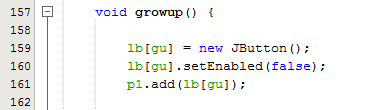
La clase Snake también se encarga de iniciar el KeyListener para que empiece a esperar inputs del teclado y finalmente instancia e inicia el Thread encargado del ciclo del juego.

Intanciando una snake

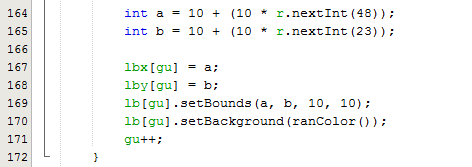


Nuestra funcion de crear una snake es un for donde la condicion del for debe ser igual a la variable gu la cual se refiere al tamaño inicial que tendrá el gusanito. En el for se van instanciando botones que constituyen cada vertebra de la snake, donde gracias a las variables lbx y lby se le dice en que coordenadas de la ventana se colocaran estos botones y se define su color. Por ultimo de define que por default el snake avanzará en dirección a la derecha al escribir lbx[i + 1] = lbx[i] – 10 y que mantendrá su posición en y igual.

Metodo para hacer crecer el snake

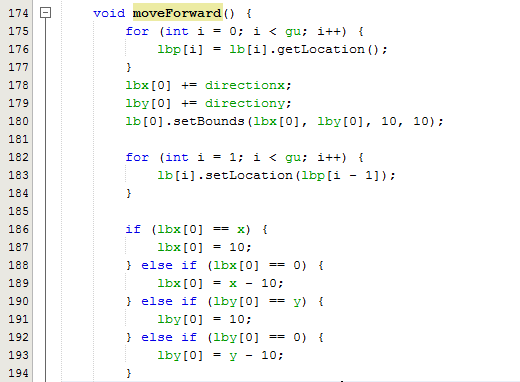


Como ya se sabe lb es el arreglo de botones que conforman el cuerpo del snake y para hacerlo crecer le ponemos gu que es el tamaño actual del gusanito como posición del arreglo y le creamos un nuevo botón que luego se deshabilita para evitar que se le dé clic. Por ultimo se añade al panel 1.

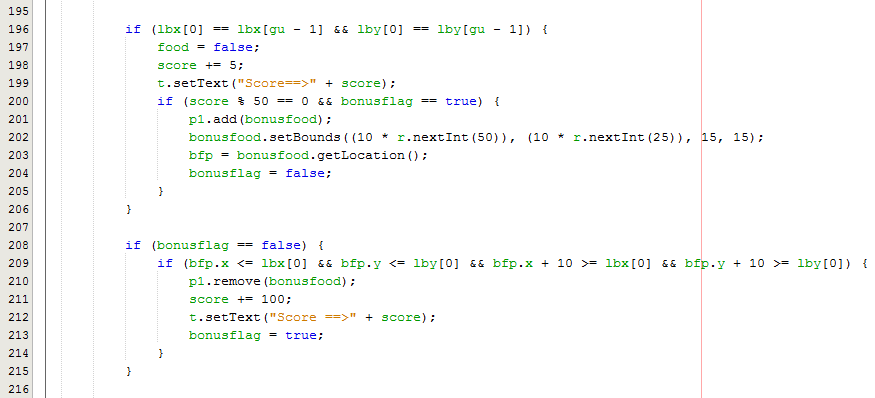


La variable r es nuestra variable random, a y b serán las coordenadas donde aparecerá la comida nueva después de ser comida porque growup implica que el snake ha consumido un botón de comida. Al añadirlo en el array del lbx y lby ya tenemos ese valor numérico para esa coordenada. Por último se le suma 1 a la variable que determina la longitud del snake.

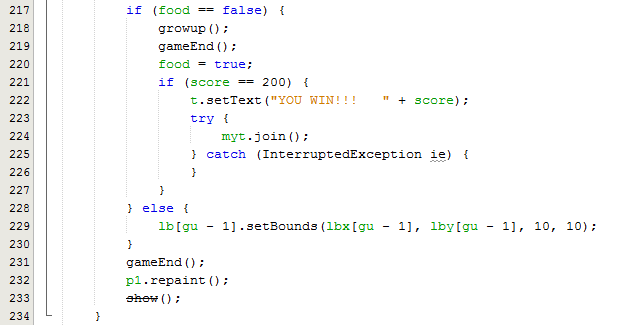
Lógica de movimiento



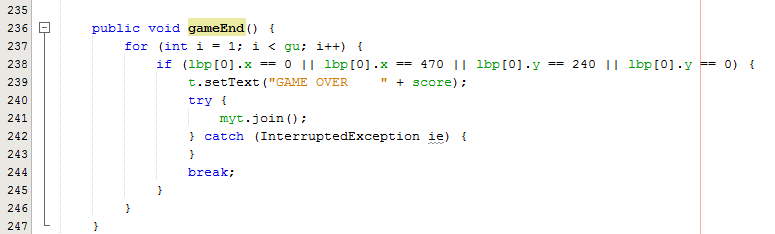
El for del inicio indica las coordenadas en las que se encuentra el snake. La función principal de este método es valodar la secuencia de movimiento del snake que seria de 10 pixeles según la dirección en la cual sea llamado ya sea moviéndose de izquierda a derecha en x o de arriba hacia abajo en y.



Obviamente en la función de movimiento verifica si el gusanito osea gu ha hecho contacto con comida que esté en la coordenada lbx y lby. De ser así le suma 5 puntos al score. Para que crezca food debe ser falso y mas adelante se llama la funcion growup.

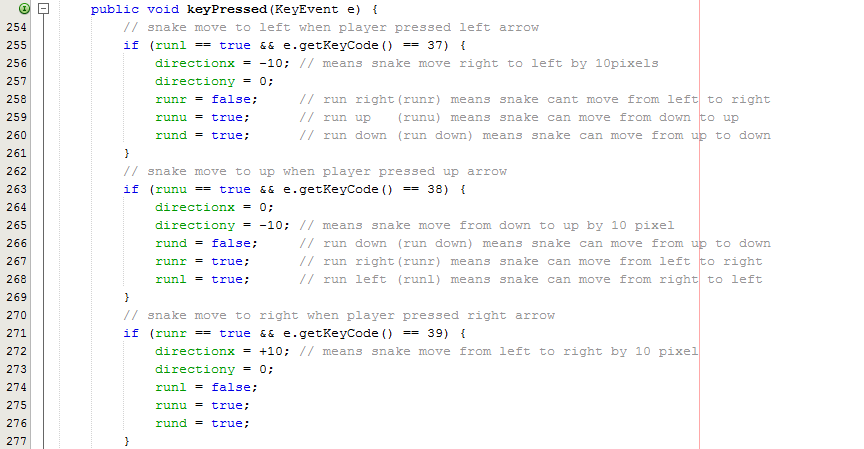


Esta parte del método valida si el jugador ha perdido llamando la funcion gameEnd()



gameEnd solamente verifica que la cabeza del snake no toque ciertas coordenadas de la ventana, si es asi se cambia el texto a Game Over y se rompe el ciclo con un Break.

Logica de dirección



Consiste en reconocer los inputs del teclado encargados de la dirección, las validaciones sirven para determinar en qué direcciones puede cambiar dependiendo de la dirección en la que avanza el snake.

Bibliografia

Programming Snake in Java!

<https://www.youtube.com/watch?v=S_n3lryyGZM>

Making snake in java  
<https://www.youtube.com/watch?v=FABTl1Q1byw> (Part 1)

<https://www.youtube.com/watch?v=kTSYU3z3vT8> (Part 2)

<https://www.youtube.com/watch?v=JA3tR3A6Tuo> (Part 3)

<https://www.youtube.com/watch?v=07UdiMlvDcI> (Part 4)

<https://www.youtube.com/watch?v=8NwKW3VEKKY> (Part 5)

<https://www.youtube.com/watch?v=D2bJ60CKS3c> (Part 6)

Programming Simple Snake Game in Java Eclipse

<https://www.youtube.com/watch?v=9ob_j8rim2g>

Create Game snake with java Netbeans

<http://gadgetz4all.blogspot.com/2010/11/membuat-game-snake-dengan-netbeans.html>