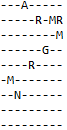
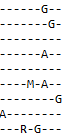
**5 BOLAS EN LÍNEA**

En esta práctica se presenta el juego de 5 bolas en línea al jugador, a través de un programa en Scala. Dicho programa se compone de 22 funciones diferentes que llevan a cabo las diferentes acciones necesarias para la ejecución del juego.

Antes de empezar a hablar de las funciones, mencionar el importe de la anotación **@tailrec**, que marca al compilador que la siguiente instrucción es recursiva de cola, a fin de optimizar el recorrido sobre las diferentes listas que utilizaremos. También se han importado la librería Random, para tener valores aleatorios, y ReadInt(), para recibir números por teclado.

La primera función de la que hablar es por supuesto el main(), en la cual crea una lista, llenándola de valores ‘-’, que representan espacios vacíos del tablero, y después se llama a la función crearTablero(list , int), que toma como parámetros la lista anterior y el número de bolas a introducir y, a través del objeto Random elige aleatoriamente una posición y un color y lo inserta en la lista que le hemos pasado, repitiendo esto el número de veces que se le haya pasado por tablero. Una vez creado el tablero, se llama a la principal función del juego, bucleJuego().

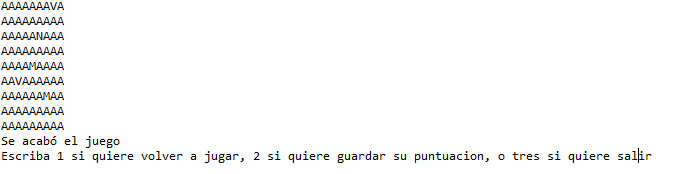


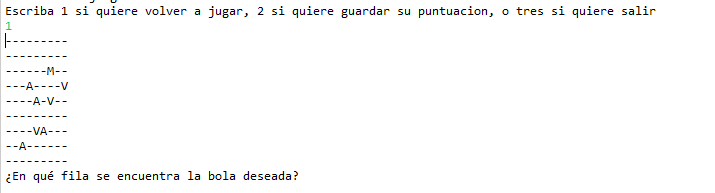
En las imágenes anteriores se pueden ver dos ejemplos de la creación del tablero y la colocación aleatoria de las bolas.

El método bucleJuego() recibe como parámetros una lista de caracteres, que actúa como el tablero y un entero que representa la puntuación del usuario. Lo primero que hace el método es llamar a la función mostrarTablero(List[Char]) que muestra la lista que recibe por parámetro, haciendo un salto de línea cada 9 posiciones. Para esto existe otra función interna mostrarDentro(List[Char],int) que conserva el índice a través del parámetro int, con el cual se comprueba si toca hacer un salto de línea o no, e imprime el siguiente elemento de la lista.

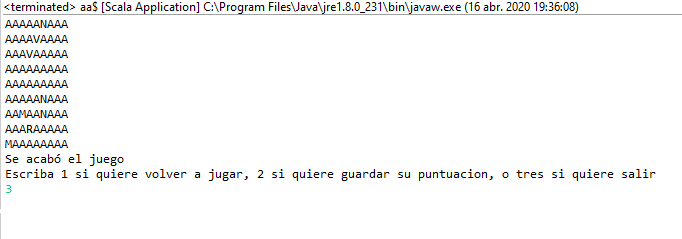
Lo siguiente es un condicional sobre comprobarFin(List(Char)), una función que recibe el tablero como parámetro y comprueba si existe algún valor ‘-’. En caso de que lo haya, quedan casillas libres y la partida no ha terminado. Esta función utiliza una función interior comprobarDentro (List(Char),Int), que es recursiva, pasando el índice actual como parámetro hasta llegar al final de la lista o el primer ‘-’.

En caso de que se haya acabado la partida, se informa al jugador y se le da la opción de jugar otra vez, guardar su puntuación o salir. Esta función comprueba también si el juego ha terminado debido a que el usuario haya vaciado el tablero, lo que supone una partida ganada en este juego.



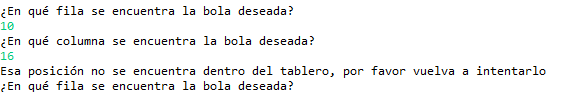


Al elegir uno la partida vuelve a empezar



Al elegir 3 el proceso acaba.

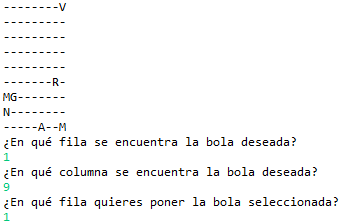
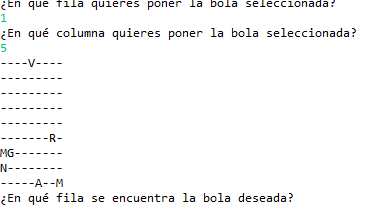
Suponiendo que la partida no ha acabado, se pide al jugador la bola que quiere mover a través de seleccionarBola(List(Char)), que pide fila y columna por teclado y la transforma a un índice a través de la función conseguirIndice(Int,Int), que multiplica la fila por el lado del tablero y le suma la columna para tener el índice. Con el índice obtenido se comprueba si son datos válidos, esto es, si están dentro de las dimensiones del tablero y si existe una bola en esa posición. Si no lo son, se vuelve a llamar a la función, pidiendo nuevos datos. En caso de que se elija una posición existente pero vacía, se avisa también al usuario de que no es válida la posición.



En caso de que la posición se valida, se ejecuta pedirMovimiento(List(Char),Int) que pide por teclado la nueva posición objetivo y transforma la información en un índice con conseguirIndice(), como en el paso anterior. Con el indice conseguido, se llama a comprobarMovimiento(List(Char), Int, Int) que recibe como parámetro el tablero, la posición inicial y la final, y devuelve un valor booleano dependiendo si el movimiento seleccionado se puede realizar o no. Para esto llama a calcularMovimiento(List(Char), Int, Int) que recibe los mismo parámetros y comprueba los movimientos válidos con los parámetros recibidos, devolviendo -1 si no hay movimientos válidos disponibles, mientras que nos devolverá el índice de la siguiente posible posición en caso de haberla. En caso de que el movimiento elegido no sea válido, se dará la opción al usuario de elegir un nuevo movimiento.

Si el movimiento era válido, se llama a realizarMovimiento(Char, Int, List[Char]), que inserta la bola, marcada por el parámetro de tipo carácter, en la posición, marcada por el parámetro entero, en la lista de caracteres del parámetro. Para esto se ayuda de la función recorrerTablero(List[Char], Int) que devuelve el valor de la bola que se encuentre en la posición del parámetro en la lista recibida, utilizando la función auxiliar recorrerDentro(List[Char], Int, Int) que utiliza uno de los enteros para iterar la lista hasta encontrar el otro entero, que actúa como índice, guardando la posición actual en el parámetro.

El método realizarMovimiento se llama dos veces, primero para poner la bola en su nueva posición, y luego otra vez para colocar un espacio vacío en la antigua posición de la bola.

Aquí vemos un ejemplo del movimiento de una bola verde (V en el tablero) en la fila 1. En este caso, la generación de nuevas bolas está desactivada para la simplificación del ejemplo. Una vez realizado el movimiento, se pide al jugador un nuevo movimiento para seguir avanzando en el juego.