Joel Diaz

Proyecto programacion 2048

Índice

1- Descripción de la idea del proyecto

Mi proyecto consiste en recrear el juego 2048. El juego trata de ir moviendo fichas cada uno con un número par. Si al mover la ficha se coloca junto a otra con el mismo número estas se suman. El juego termina cuando todas las celdas se llenan o si se consigue sumar números hasta llegar a 2048.

2- Explicación del desarrollo del programa.

Para abordar este proyecto empecé creando la clase tablero que es una matriz. Su constructor consta de un bucle for que llena la matriz de 0 para representar celdas vacías. La clase tablero solo tiene un método que se encarga de mostrar el tablero en la consola.
Continúe creando la clase control que contiene métodos encargados de hacer comprobaciones dentro del tablero como comprobar si hay celdas vacías en filas o columnas, comprobar si la partida ha terminado con resultado de victoria o derrota. También contiene el método encargado de generar un dos aleatorio en una celda vacía del tablero. Este método se ejecuta dos veces al iniciar y cada vez que se lleva a cabo un movimiento.
La siguiente clase que cree fue movimientos, donde se encuentran todos los métodos que contienen la lógica de hacer los correspondientes cambios en el tablero. Se podrían agrupar en 4 tipos: Arriba, abajo, derecha e izquierda. Cada uno de ellos cuenta con tres métodos, uno encargado de mover todas las celdas de una fila o columna en el sentido correspondiente, un método que suma las celdas en caso de que al moverlas contengan el mismo valor y por último un método que usa los dos anteriores y los aplica en todo el tablero de forma adecuada según el movimiento.

Ejemplo de movimiento:

```java
public  void MoverArribaIndv(int c) { // metodo para mover hacia arriba los valores de una sola columna

    if (Control.ColumnasVacias(c) < tablero.columnas) { // se comprueba si hay fichas en la columna

        for (int i = 0; i < (tablero.columnas - 1); i++) { // se repite el numero de veces como celdas hay en la columna - 1
            for (int j = 0; j < (tablero.filas - 1); j++) {
                if (tablero.tablero[j][c] == 0) { // comprueba que la celda esta vacia y copia la celda de abajo
                    tablero.tablero[j][c] = tablero.tablero[j + 1][c];
                    tablero.tablero[j + 1][c] = 0; // borra la celda de abajo

                }
            }
        }
    }
}

public  void SumarArriba(int c) { // metodo que suma las celdas de las columnas hacia arriba

    if (Control.ColumnasVacias(c) < tablero.filas - 1) {

        for (int i = 0; i < tablero.filas - 1; i++) {

            if (tablero.tablero[i][c] == tablero.tablero[i + 1][c]) {
                tablero.tablero[i][c] *= 2;
                tablero.tablero[i + 1][c] = 0;

            }
        }
    }

}

public  void MoverArribaTotal() { // metodo que usa el metodo MoverArribaIndv y lo usa en todas las columnas

    for (int i = 0; i < this.tablero.columnas; i++) {
        MoverArribaIndv(i);
        SumarArriba(i);
        MoverArribaIndv(i); // repito el metodo porque en la suma de una columna pueden quedar huecos al
                            // sumar 4 celdas
    }
    if (!Control.fin()) {
        Control.Dos();
        } // una vez hecho el movimiento se pone un dos aleatorio
}
```

Para finalizar cree la clase jugar que es la encargada de recoger las entradas del usuario y efectuar el movimiento que este desee. Primero usa el método para crear un dos aleatorios en el tablero dos veces, lo muestra por con consola y luego ejecuta un bucle do while que se efectúa mientras el método fin devuelve false. Por último está el main del programa donde se imprime un mensaje con las instrucciones de juego y se ejecuta el método jugar.

3- Notas importantes sobre el diseño y la implementación.

Algo clave para el desarrollo de este proyecto y que me ha facilitado el desarrollo de los movimientos ha sido la implementación de métodos que analizan el tablero como “ColumnasVacías” y “Filasvacias”.
Cuando me enfrente a desarrollar la lógica de los movimientos empecé con un solo método por movimiento lo cual resultó ser bastante complicado, el hecho de segmentar el movimiento en dos métodos y uno que los conecte me facilitó luego encontrar y resolver errores lógicos dentro de los métodos.

4- Horas estimadas invertidas

El grueso de este trabajo lo he encontrado en el planteamiento de cómo efectuar los movimientos y resolver errores dentro de estos. También me he topado con otro tipo de errores que comentaré más adelante.
Estimo que el trabajo me habrá consumido entre 15 y 20 horas. Considero que se debe a que es la primera vez que me enfrento a la programación orientada a objetos y he cometido errores que no sabía resolver.

5-Problemas o errores encontrados durante el desarrollo

Durante el desarrollo me he encontrado con errores de varios tipos:

Errores lógicos en los movimientos y similares:

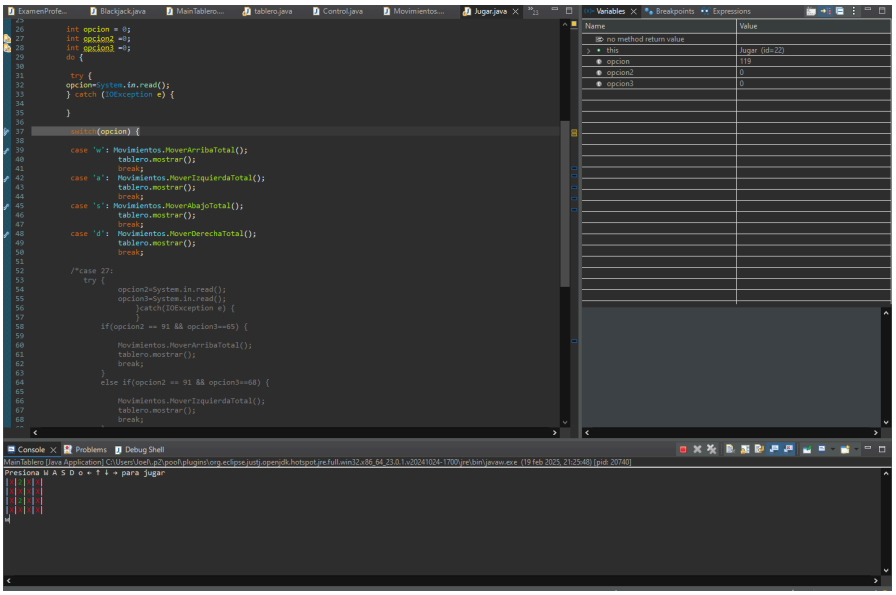
Me he topado con errores menores como poner mal signos o sumar en vez de restar. También me encontré con este error en el método Dos que podía generar números en celdas donde previamente se acaba de hacer una suma.

```
} while (tablero.tablero[f][c] != 0); // anteriormente era while (ColumnasVacias(c) == 0); pero podia generar un
                                      // dos en la celda donde se acabab de sumar un numero

tablero.tablero[f][c] = 2;
```

Errores de prints:

En la fase final del desarrollo me topé con que había ciertos prints que se repetían de forma no deseada y para resolver este error use la herramienta de eclipse de debugeo.



Errores de clases:

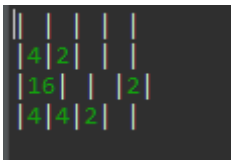
Este ha sido el tipo de error que más tiempo me ha llevado resolver. Al comienzo del desarrollo todo el programa estaba en única clase pero era conveniente organizarlo por clases. Separando los métodos en clases me di cuenta de que además de que no había creado bien los constructores correspondientes de cada clase dándome este error.

```
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "trabajo2048.Movimientos.MoverDerechaTotal()" because "trabajo2048.MainTablero.Movimientos" is null
        at trabajdo2048/trabajo2048.MainTablero.main(MainTablero.java:16)
```

Para resolverlo tuve que crear los constructores de forma correcta y tuve que organizar algún método como por ejemplo tuve que quitar de la clase tablero el método Dos y meterlo en la clase control ya que en un principio llamaba a este método desde el constructor de tablero y esto daba problemas.

Errores visuales.

Esto no es necesariamente un error ya que no impide que el programa funcione correctamente pero si obstaculiza el entendimiento de las salidas por consola y es que cuando se crean fichas de dos, tres o cuatro dígitos la matriz se descuadra tal que así:



Por lo que he investigado esto no tiene solución en java más que crear una interfaz.

6- Detalle relevante sobre el proceso.

En este apartado comentaré aspectos del programa que me gustaría haber implementado y no he podido ya sea por desconocimiento de cómo hacerlo o por falta de tiempo.

Primero intente implementar que el programa fuera capaz de leer las entradas de las flechas del teclado y a su vez “w a s d”.
Esto lo intente con System.in.read() que lee las entradas de bytes con un switch que según el byte en ASCII ejecutaba un case.

```
/*case 27:
   try {
            opcion2=System.in.read();
            opcion3=System.in.read();
                }catch(IOException e) {
                }
        if(opcion2 == 91 && opcion3==65) {

            Movimientos.MoverArribaTotal();
            tablero.mostrar();
            break;
        }
        else if(opcion2 == 91 && opcion3==68) {

            Movimientos.MoverIzquierdaTotal();
            tablero.mostrar();
            break;
        }
        else if(opcion2 == 91 && opcion3==66) {

            Movimientos.MoverAbajoTotal();
            tablero.mostrar();
            break;
        }
        else if(opcion2 == 91 && opcion3==67) {

            Movimientos.MoverDerechaTotal();
            tablero.mostrar();
            break;
        */
```

Las flechas del teclado constan de dos bytes 91 y 65 (en ASCII) por ejemplo para la flecha de arriba por eso están variables opcion 2 y opción 3.

Otro apartado que me gustaría haber implementado es el hecho de borrar la consola con cada entrada del usuario. Esto lo intente de dos formas, la primera fue:

```
1  package trabajo2048;
2
3  public class Consola {
4
5
6
7      public Consola(){
8
9
10
11     }
12
13     public void borrar() {
14
15         System.out.print("\033[H\033[2J");
16         System.out.flush();
17
18     }
19 }
```

Con este print pero no funcionaba posiblemente porque mi terminal no soporta caracteres de entrada ANSI. También lo intente importando la librería JLine manualmente pero no funcionó por la falta de sus dependencias posiblemente. Me hubiera gustado seguir por este camino pero me he quedado sin tiempo, es posible que lo continúe más adelante de forma personal.