[Enunciado]

Se pide hacer una aplicación que permita jugar al **SumaTres**. El juego SumaTres es una versión del juego AddThree que se puede encontrar por ejemplo dentro de la aplicación CrazyPlus que está disponible tanto en plataformas Android como iOS:

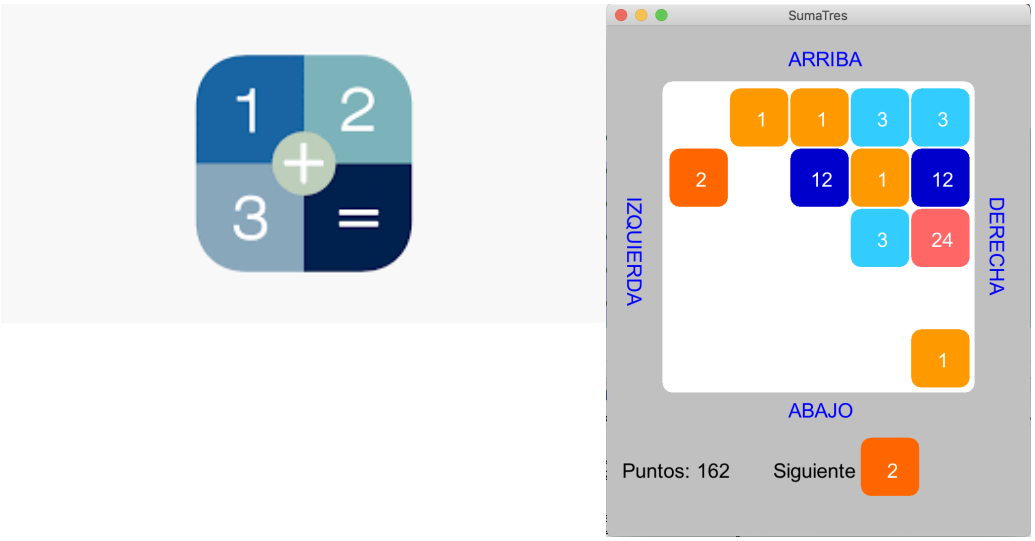

Figura 1: Juego SumaTres

Los objetivos primordiales del Trabajo en Grupo son básicamente dos:

1. Que practiquéis haciendo un programa que puede ser divertido o desafiante.
2. Que desarrolléis, junto con algunos de vuestros compañeros, un programa.

Es decir, el objetivo no es que todos logréis hacer al final que el juego funcione perfectamente (aunque ojalá lo consigáis), sino que **el objetivo es que practiquéis**. Para programar bien hace falta practicar mucho y más ahora que estáis empezando. Aunque al final el juego no funcione, si habéis practicado mucho con él, y habéis aprendido cosas os habrá servido para: 1) mejorar vuestro nivel como programadores, y 2) estar mejor preparados de cara al examen de Enero. Ese el motivo por el que la fecha de entrega coincide justo con el examen de Enero, para que uséis el Trabajo en Grupo como una forma de estudiar la asignatura.

Os recomendamos que el programa lo hagáis de una forma incremental. Empezando por las cosas sencillas poco a poco, probándolas bien hasta que estéis seguros de que funcionan, antes de pasar a las partes más complicadas. Aunque al final no funcione el juego, esas partes que estén bien serán tenidas muy en cuenta en la nota. Hacer muchas partes del programa de golpe y luego tratar de probarlas todas juntas complica mucho la depuración de errores. Es inevitablemente que tengáis errores en las versiones iniciales de cualquier parte del programa ya que muchas no son triviales y es normal cometer algunos errores al programar cada funcionalidad (incluso les ocurre a programadores con experiencia).

[Descripción del juego]

Objetivo del juego: Sumar el mayor número de puntos que se consiguen sumando piezas iguales contiguas (mayores o iguales a 3) o que sumen 3 (2+1). El juego se acaba cuando no se pueden colocar nuevas piezas al no haber ninguna casilla libre y si no se pueden sumar ningún par de piezas contiguas al hacer un movimiento (ver Figura 13).

Inicio del juego: Al principio de la partida el tablero tiene 3 piezas con valores 1, 2 y 3 que están colocadas al azar (ver Figura 12).

Jugadas: Las jugadas constan de los pasos siguientes:

1. **Determinar el valor de la nueva pieza**: Antes de hacer la jugada propiamente dicha, el programa tiene que elegir un valor aleatorio entre 1, 2 ó 3 para la nueva pieza que **se colocará al final de la jugada**.

2. **Seleccionar una dirección de desplazamiento**: El jugador selecciona desplazar las piezas en una de las 4 direcciones: hacia arriba, hacia abajo, hacia la izquierda o hacia la derecha.

3. **Desplazamiento de las piezas**: Una vez elegida una dirección las piezas del tablero **se desplazan** en esa dirección apilándose. Por ejemplo, si el jugador elige el desplazamiento hacia abajo, las piezas caerán hacia abajo por cada columna y solamente quedarán (si quedan) huecos libres en las filas superiores, estando todas las piezas compactadas por columnas en la parte inferior del tablero.

4. **Suma de piezas**: Si al desplazar y apilar las piezas se encuentran contiguas **en esa dirección** dos piezas que sean iguales siendo mayores o iguales que 3, o que sumen 3 (2+1), esas dos piezas se fusionarán sumándose, (con lo que desaparecerá una de las piezas) y el resto de piezas se apilarán en la dirección del desplazamiento elegido. Recordad, **dos unos, o dos doses juntos NO se suman**.

5. **Colocar nueva pieza**: Finalmente se colocará aleatoriamente (en una de las casillas libres) la nueva pieza que se determinó antes de empezar la jugada.

Puntuación: En cada jugada realizada se suman tantos puntos como el valor de las sumas hechas. Por ejemplo, si se fusiona un 1 y un 2 se suman 3 puntos, si se fusionan 3+3, se suman 6 puntos y así sucesivamente.

Fin de la partida: El juego se acaba cuando no se pueden colocar nuevas piezas al no haber ninguna casilla libre Y cuando además no se pueden sumar ningún par de piezas contiguas al hacer un desplazamiento. Es decir, aunque el tablero esté lleno, si al desplazar las piezas en una dirección se pudiesen sumar dos piezas contiguas, el programa debe permitir hacer esa jugada ya que al hacerla se generará al menos una casilla libre para que la nueva pieza se pudiese colocar. **El jugador no está perdido aunque el tablero esté lleno si puede sumar al menos un par de piezas**.

En los siguientes párrafos describimos de forma más precisa alguna de estas partes, en especial las jugadas y cómo determinar el final de la partida, y qué cosas hay que tener en cuenta para implementar dichos elementos en el juego.

Realización de un jugada

0) Dado un tablero cualquiera, con una nueva pieza ya elegida, por ejemplo, el siguiente tablero donde la nueva pieza será un 1 (indicado en la parte inferior de la ventana):

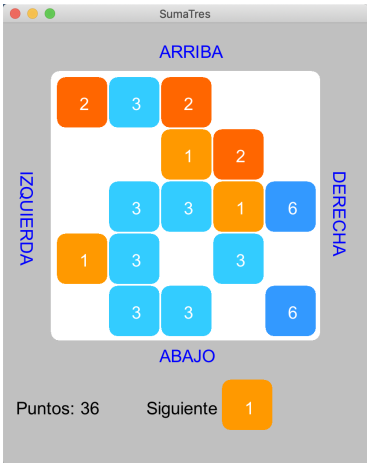

Figura 2: Ejemplo de jugada #1

1) el jugador debe seleccionar en qué dirección desea que se desplacen las piezas, ya sea hacia arriba, hacia abajo, hacia la izquierda o hacia la derecha. Imaginemos que elige desplazar hacia abajo. En ese caso, las piezas se desplazarán hacia abajo por cada columna, apilándose en la parte inferior de cada columna. Dada la posición anterior, el tablero quedaría en el siguiente estado:

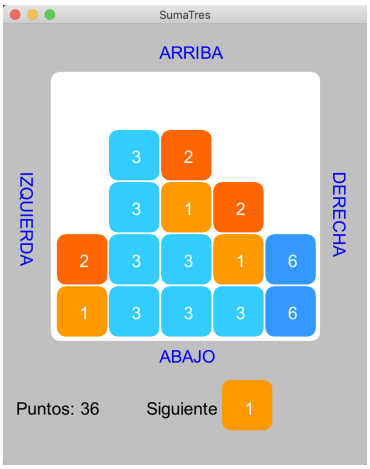

Figura 3: Ejemplo de jugada #1, paso 1, piezas desplazadas

2) Ahora hay que sumar las piezas contiguas que sean iguales o que sumen 3 (1+2), pero, ojo!, y esto es fundamental, **la adyacencia solamente hay que comprobarla en la misma dirección del desplazamiento y empezando en esa dirección**. Es decir, como en el ejemplo las piezas las hemos desplazado hacia abajo, solo miraremos si hay dos piezas iguales (que no sean ni 1 ni 2) o que suman 3 en vertical (1+2), nunca en horizontal, y yendo de abajo a arriba. Por ejemplo, en el tablero anterior, una vez desplazadas las piezas, tenemos los siguientes pares de piezas que deben sumarse.

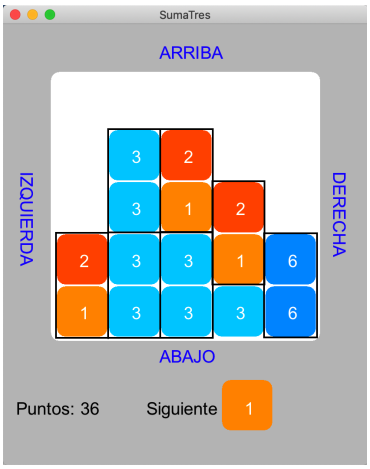

Figura 4: Ejemplo de jugada #1, paso 1, piezas a sumar

3) Las piezas contiguas se suman y de nuevo se compactan en la dirección del desplazamiento (hacia abajo en este caso).

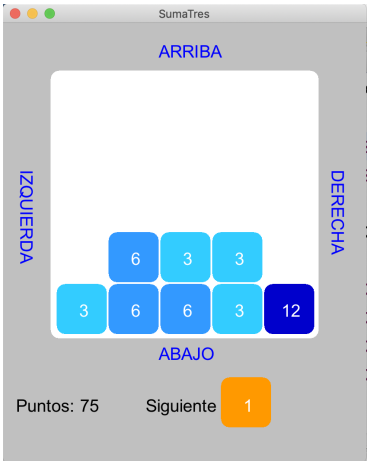

Figura 5: Ejemplo de jugada #1, paso 2, piezas sumadas

Nótese que han quedado piezas juntas que son iguales pero no hay que seguir sumándolas. Se sumarán en la siguiente jugada si el usuario lo desea eligiendo la dirección apropiada.

4) El último paso es colocar la pieza nueva en el tablero. Es importante recalcar que **todo el proceso para generar y colocar nuevas piezas debe ser aleatorio**. En primer lugar, y como se puede observar en las figuras anteriores, el valor de la siguiente pieza se le indica al jugador **antes de hacer la jugada** para que pueda hacerla teniendo en cuenta ese aspecto. En el ejemplo la pieza siguiente será un 1. Luego, una vez que la jugada se ha realizado completamente, el programa elegirá de forma aleatoria una de las casillas vacías que hayan quedado libres tras la jugada para colocar la nueva pieza. Nótese que las casillas vacías no coinciden con las que había antes de hacer la jugada. Es decir, la casilla elegida debe determinarse **DESPUÉS** de hacer la jugada comprobando que no sea una casilla ya ocupada.

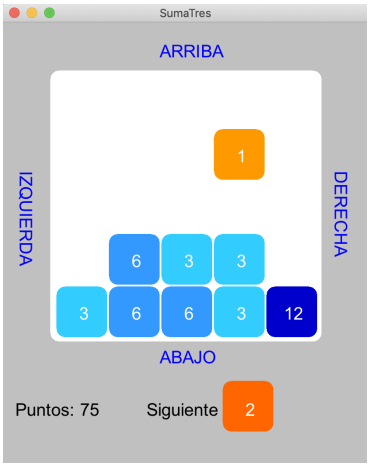

Figura 6: Ejemplo de jugada #1, paso 3, colocar nueva pieza

Veamos otro ejemplo con un movimiento en horizontal.
0) Dado el siguiente tablero

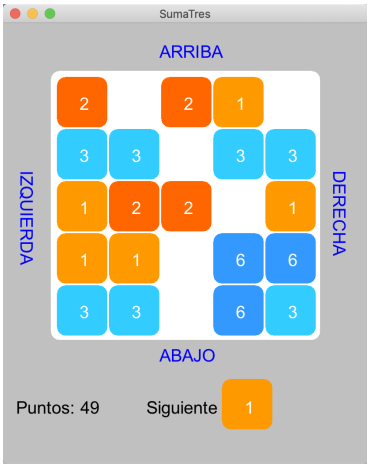

Figura 7: Ejemplo de jugada #2

1) el jugador selecciona desplazar a la izquierda, luego las piezas se compactan desplazándose hacia la izquierda:

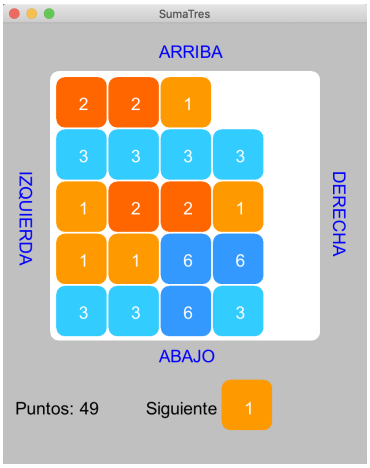

Figura 8: Ejemplo de jugada #2, paso 1, piezas desplazadas

2) Ahora hay que sumar las piezas contiguas que sean iguales (y distintas de 1 y 2) o que sumen 3 (1+2), pero, en este caso hay que comprobar la adyacencia en horizontal y empezando por la izquierda. Las piezas a sumar serían las siguientes:

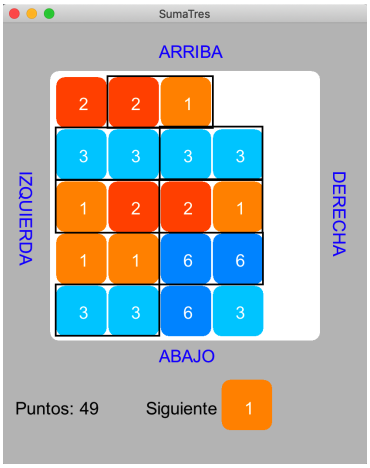

Figura 9: Ejemplo de jugada #2, paso 1, piezas a sumar

Nótese que en la primera y la cuarta fila hay respectivamente dos doses y dos unos juntos pero esos no se suman.

3) Se suman las piezas contiguas y de nuevo se compactan en la dirección del desplazamiento, en este ejemplo a la izquierda.

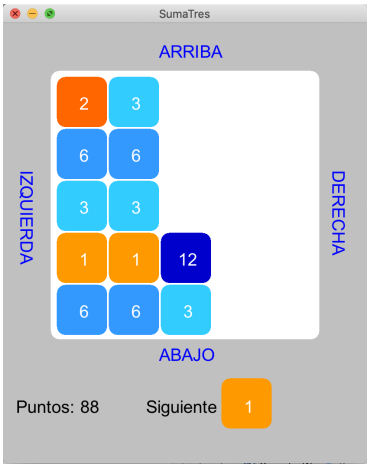

Figura 10: Ejemplo de jugada #2, paso 2, piezas sumadas

4) Por último se coloca la nueva pieza en una casilla vacía y se genera aleatoriamente la nueva pieza para la jugada siguiente que en este caso será un 2.

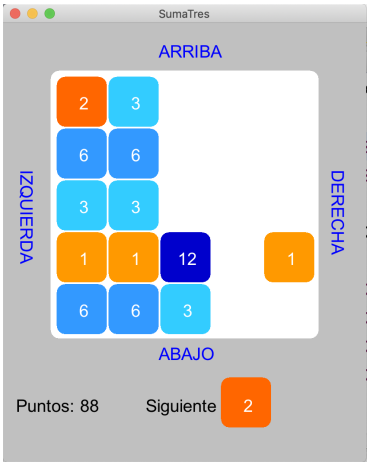

Figura 11: Ejemplo de jugada #2, paso 3, colocar nueva pieza

Inicio de la partida

Como dijimos anteriormente, al principio de la partida el tablero tiene que estar completamente vacío excepto en 3 piezas con valores 1, 2 y 3 colocadas al azar. La siguiente figura muestra 2 ejemplos de tablero inicial.

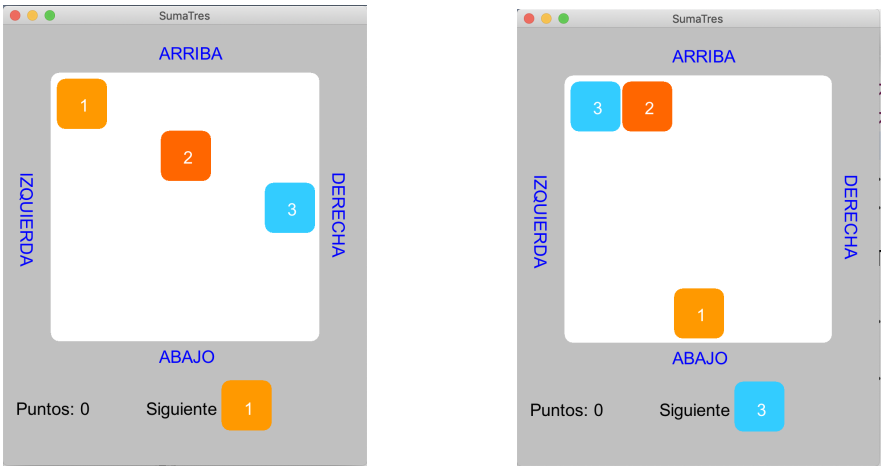

Figura 12: Ejemplo de tableros iniciales

Nótese que además de colocar las primeras 3 piezas ya está generada cuál será la próxima pieza a colocar ya que eso debe hacerse antes de que el jugador realice cada jugada.

Fin de la partida

La regla general es que la partida acaba cuando el tablero esté lleno y no puedan colocarse nuevas piezas. Por ejemplo en el siguiente caso:

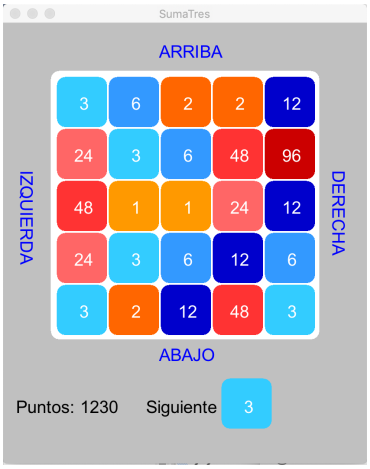

Figura 13: Partida acabada

Como se puede ver, no hay ningún par de piezas que puedan sumarse aunque las piezas se desplazasen en cualquiera de las cuatro direcciones posibles.

En cambio en los dos siguientes tableros, aunque estén llenos, sí que existe algún par de piezas contiguas que pueden sumarse si el jugador elige la dirección apropiada:

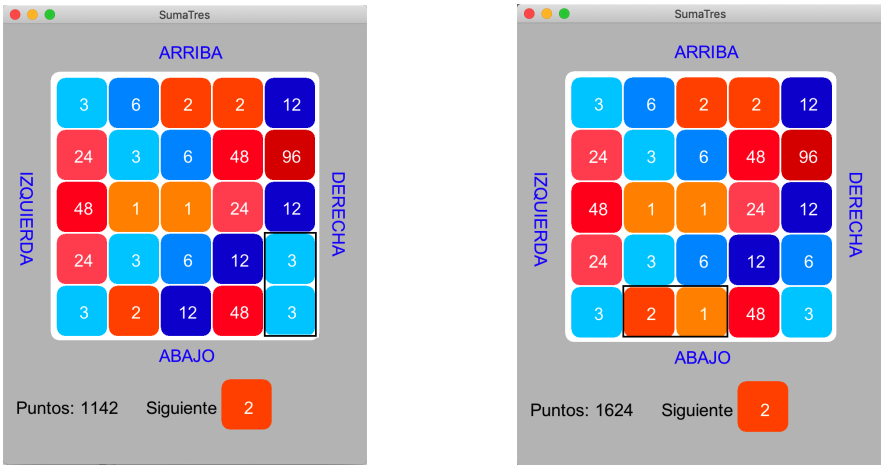

Figura 14: Partidas NO acabadas

En el primer caso habría que elegir el desplazamiento hacia arriba o hacia abajo para sumar los dos treses marcados, y en el segundo caso habría que elegir el desplazamiento a la izquierda o a la derecha para sumar el 1 y el 2. En casos como éstos, la partida obviamente NO debe finalizar ya que el jugador todavía podría poner nuevas piezas.

[Implementación del juego]

Inicio (Constructor). La aplicación debe estar diseñada para permitir jugar con cualquier tamaño de tablero rectangular. Es decir, el constructor de la clase donde se implemente el juego debe tener dos parámetros (número de filas y columnas) para decidir el tamaño del tablero de juego.

Visualización del tablero. Esta parte es la más sencilla ya que consiste básicamente en mostrar en pantalla una matriz.

Generación aleatoria de nuevas piezas y colocación en el tablero. Esta parte también es relativamente sencilla ya que consiste simplemente en generar piezas aleatorias (siguiendo procesos similares a los vistos en algunas prácticas, como Piedra, Papel y Tijera, o Bonoloto) y generar otro número aleatorio para determinar la casilla del tablero donde se colocará la pieza, pero comprobando que dicha casilla esté vacía, ya que si no lo está el proceso deberá repetirse.

Jugadas. La parte más complicada del juego es obviamente hacer las jugadas. Si se siguen los pasos descritos anteriormente, el proceso es un poco más sencillo de lo que parece, pero es algo repetitivo ya que debe hacerse un código similar para las 4 direcciones posibles. Se recomienda hacerlo primero en una dirección y cuando funcione copiarlo y modificarlo apropiadamente para el resto de direcciones. Si no os funciona bien en una dirección es absurdo replicar la misma solución para las demás direcciones. Es mejor centrarse en mejorar la solución para la dirección en la que se está trabajando.

Detectar Fin de Partida. Esta parte también puede ser relativamente complicada. La partida acaba cuando no se pueden colocar las nuevas piezas. Esto equivale a que si hay que colocar una pieza, la partida acabará si el tablero está totalmente lleno, pero si hay 2 piezas contiguas que se puedan sumar entonces la partida debe continuar.

[Opciones para hacer la práctica]

Hay dos opciones para realizar la práctica:

1. Aplicación de consola: la puntuación máxima siguiendo esta opción es de **8 puntos**.
2. Aplicación gráfica: la puntuación máxima en este caso es de **10 puntos**.

[Aplicación de consola]

Se jugará a través de la consola mostrando el estado del tablero mediante una representación basada en caracteres. Las jugadas se leerán de teclado, indicando el jugador la dirección de desplazamiento, por ejemplo mediante un carácter. Se debe mostrar el estado del tablero tras hacer cualquier paso de las jugadas, tanto después de desplazar las piezas, como después de hacer las sumas. Cuando la partida acabe se mostrará por pantalla la puntuación final obtenida.

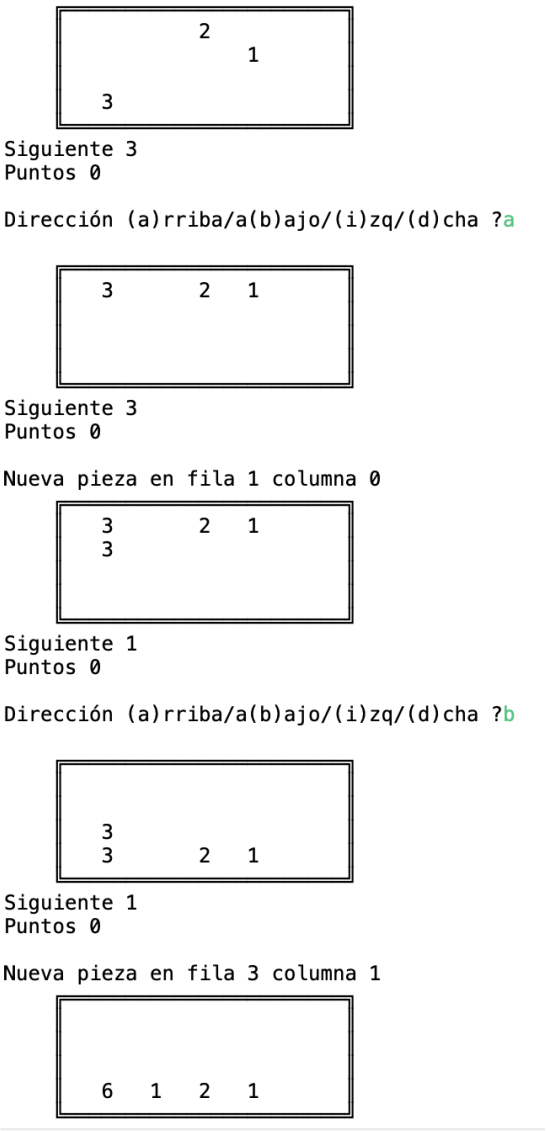

Figura 15: Ejemplo de aplicación de consola

[Aplicación gráfica]

Se jugará a través de una ventana gráfica mostrando el estado del tablero mediante una representación similar a la que aparece en las figuras con ventanas que hay en este documento. Las jugadas deben hacerse mediante clicks de ratón. Se recomienda que para seleccionar la dirección de desplazamiento se compruebe si el usuario ha pinchado arriba, abajo, a la izquierda o la derecha del tablero (véase las etiquetas colocadas alrededor del tablero en las figuras mostradas a lo largo de este enunciado).

Cuando la partida acabe se debe mostrar un mensaje indicando que la partida ha acabado, por ejemplo un mensaje similar al que se ve en la siguiente imagen:

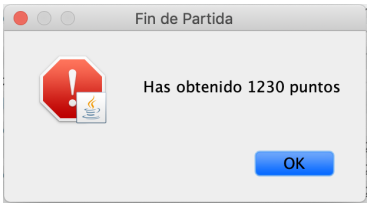

Figura 16: Mensaje final

NOTA IMPORTANTE: Aunque la aplicación sea gráfica, se debe mostrar por la consola el estado del tablero tras cada operación de forma que se pueda comprobar que el juego funciona correctamente. Es una forma que permite además facilitar la depuración del programa (ver Figura 15).

[Cosas que NO hace falta implementar en la versión gráfica]

Hay cosas que **NO SE PIDEN** para facilitar la práctica en su versión gráfica:

- En la aplicación gráfica, no es necesario que se observen las jugadas paso a paso. Basta con que se vean en la traza de la consola. Implementar esto requiere el uso de un Timer. Si algún alumno está interesado en hacerlo puede pedir información a los profesores de cómo se pueden realizar esta característica. En caso de realizarlo finalmente, deberán indicarlo en la memoria de la práctica.

[Normas]

1. Se debe seguir el paradigma de la Programación orientada a objetos, es decir, se debe escribir una clase que implemente el juego y un programa que emplee dicha clase para realizar la aplicación.

2. El programa debe ser original. Si se detecta cualquier tipo de copia, de cualquier fuente o procedencia, los integrantes del grupo o los grupos que se vean implicados **suspenderán la asignatura**. Suspende también los alumnos que pasan la práctica, no solamente los que la copian.

3. Cada grupo, formado por un máximo de 3 alumnos, debe entregar en un documento zip lo siguiente:

   - Una memoria en formato pdf de no más de 4 páginas en el que se expliquen brevemente los aspectos más destacables de la aplicación realizada. En concreto se deben explicar:
     a) La descripción de las clases empleadas.
     b) La forma de representar el tablero y de colocar las fichas.
     c) Los esquemas iterativos empleados para realizar las jugadas.
     d) **(Muy Importante)** Una lista con todos aquellos elementos que no funcionen en la aplicación. Por ejemplo, si la aplicación no puede hacer alguna de las partes del programa descritas anteriormente, se debe indicar en la memoria.
     e) Los aspectos extra que se hayan implementado, si es que existe alguno (aunque no son necesarios).
     f) **NO se deben incluir listados de la aplicación en la memoria**.
   - Otro fichero zip con el proyecto exportado de Eclipse. El nombre debe ser PF-DNI, siendo el DNI el de un miembro cualquiera del grupo.

[Se valorarán los siguientes aspectos]

- Que se empleen las metodologías de programación que se han estudiado durante el curso. Por ejemplo en cuanto al diseño de clases y de esquemas iterativos.
- La eficiencia del programa.
- Los comentarios incluidos en el programa, Javadoc y normales.
- La claridad y sencillez del código.

Hora y fecha límite de entrega: las 14:59 horas del día 14 de Enero de 2021. La entrega se debe hacer a través del Campus Virtual en la Tarea que se habilitará al efecto (basta con que la suba un alumno del grupo).

[Ejemplo de aplicación para capturar los clicks de ratón]

Lo que sigue a continuación es el código fuente de las dos clases que integran el proyecto `Tablero` que se suministra con este enunciado: la clase `Tablero` que puede servir de base para la implementación del juego, y la clase `TestTablero` que constituiría el programa principal. En el fichero `Tablero.java` se encuentra implementado mediante la clase privada `MouseHandler` la manera de detectar los clicks de ratón y las coordenadas donde se producen. La forma de mostrar cuadros de diálogos (`JOptionPane.showMessageDialog`) se estudió en las sesiones prácticas de la asignatura. Para dibujar se puede necesitar emplear: a) para cambiar el color el método `setColor()` y las constantes de color de la clase `Color`, como `Color.blue`, b) para dibujar cuadrados con bordes redondeados se puede usar el método `fillRoundRect()`, con un objeto de tipo `Graphics` y c) para pintar un texto el método `drawString()`.

TestTablero.java

```
1  import javax.swing.JFrame;
2
3  public class TestTablero {
4
5      public static void main(String[] args) {
6          Tablero t = new Tablero();
7
8          JFrame app = new JFrame("Tablero");
9
10         app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11         app.setBounds(0, 0, 440, 470);
12         app.add(t);
13         app.setVisible(true);
14     }
15 }
```

Tablero.java

```
1  import java.awt.Color;
2  import java.awt.Graphics;
3  import java.awt.event.MouseAdapter;
4  import java.awt.event.MouseEvent;
5  import javax.swing.JOptionPane;
6  import javax.swing.JPanel;
7
8  public class Tablero extends JPanel {
9
10     //Aquí irían los atributos necesarios
11
12     //Constructores
```

```
13      Tablero() {
14          //El constructor en realidad debe tener parámetros
15          //para inicializar por ejemplo el tamaño del tablero

17          // Añadimos el 'escuchador' de ratón
18          addMouseListener(new MouseHandler());
19      }

21      //Métodos de la clase que implementan el juego: básicamente poner piezas
22      //nuevas en el tablero, hacer jugadas, saber si la partida acabó o
23      //imprimir el tablero

25      //Método paint
26      public void paintComponent(Graphics g) {
27          super.paintComponent(g);

29          //Aquí iría el código para pintar el estado del tablero

31      }

33      //Clase privada para capturar los eventos del ratón
34      private class MouseHandler extends MouseAdapter {
35          public void mouseClicked (MouseEvent e) {
36              //Mostramos un diálogo con la posición del ratón
37              //para ver un ejemplo de cómo se obtienen las coordenadas
38              //donde se produjo el click
39              JOptionPane.showMessageDialog(null, String.format("Ratón %d, %d \n
                    ",e.getX(),e.getY()));

41              //Aquí irían las instrucciones para comprobar dónde pincho
42              //el usuario y hacer la jugada oportuna

44              //Se pueden llamar a los métodos públicos de la clase Tablero

46              //Seguramente habrá que repintar el tablero si se realizó
47              //una jugada válida
48              repaint();
49          }
50      }
51  }
```