

[Enunciado]

Se pide hacer una aplicación que permita jugar a una versión del Color Tiles. Una versión más compleja de este juego se puede encontrar en Google Play (buscar Color Tiles Addictive Puzzle) o también online en:

http://en.gamesaien.com/game/color_tiles/

Los objetivos primordiales del Trabajo en Grupo son básicamente dos:

1. Que practiquéis haciendo un programa que os puede resultar interesante, divertido o desafiante. Al fin y al cabo es una aplicación *real*.
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:

- mejorar vuestro nivel como programadores, y
- estar mejor preparados de cara al examen de Enero.

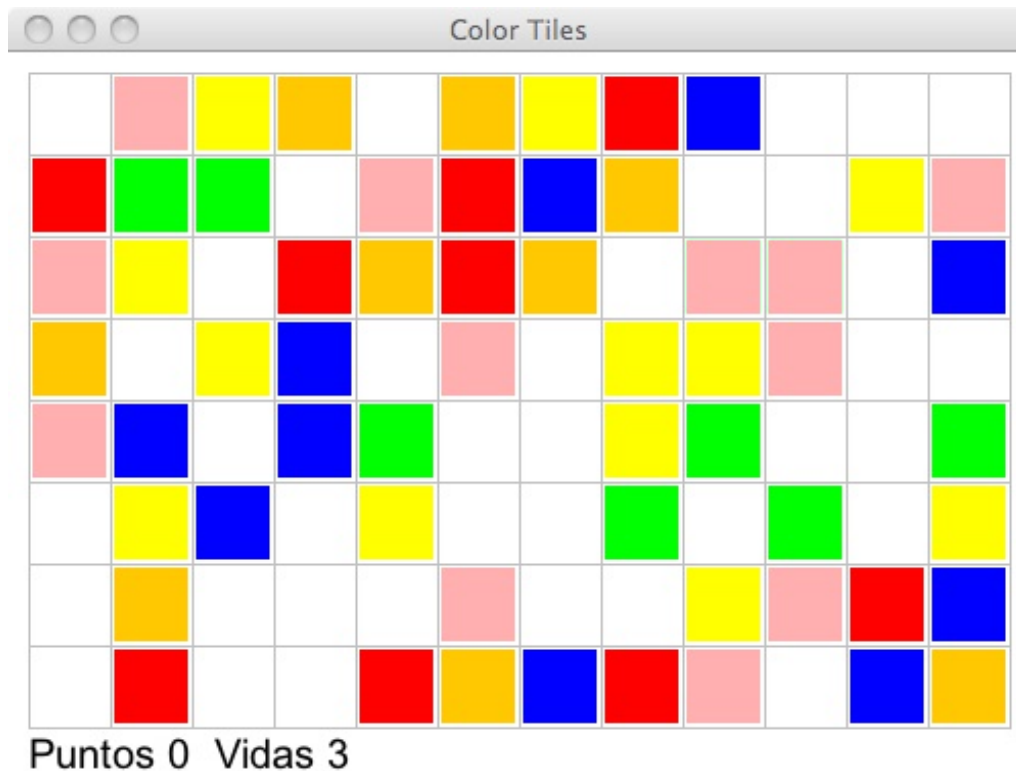
Ese el motivo por el que la fecha de entrega coincide justo con el examen de Enero, para que uséis el Trabajo en Grupo como una forma de estudiar la asignatura.

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).

Se recomienda por tanto ir haciendo las diferentes partes indicadas en la sección **Implementación del Juego** en la Página 7. Lo primero que debe hacerse y probarse es la inicialización del tablero y su visualización. Una vez que eso funcione se puede pasar a las siguientes partes del juego. Probad cada parte antes de pasar a la siguiente.

[Descripción del juego]

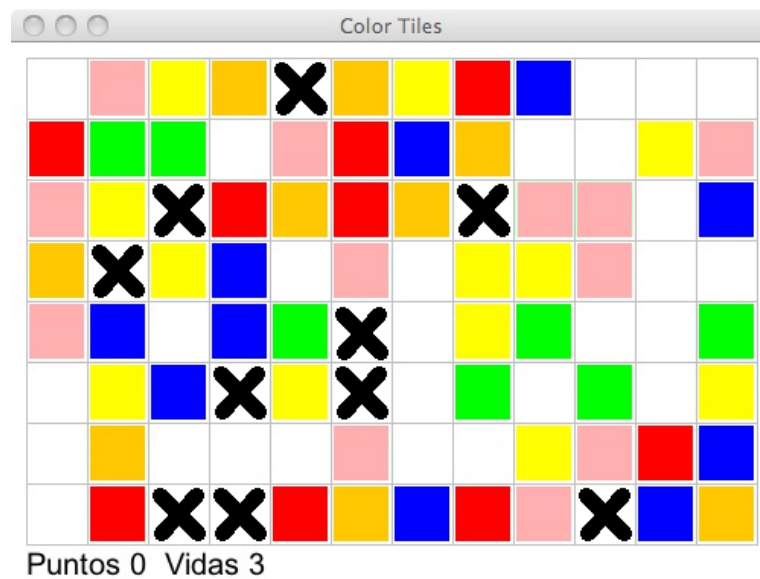
El juego es basa en un tablero rectangular de casillas, que pueden tener un cuadrado de un color o estar vacías. Inicialmente el tablero se rellena aleatoriamente con una serie de cuadrados de colores, tal como se muestra en la siguiente figura:



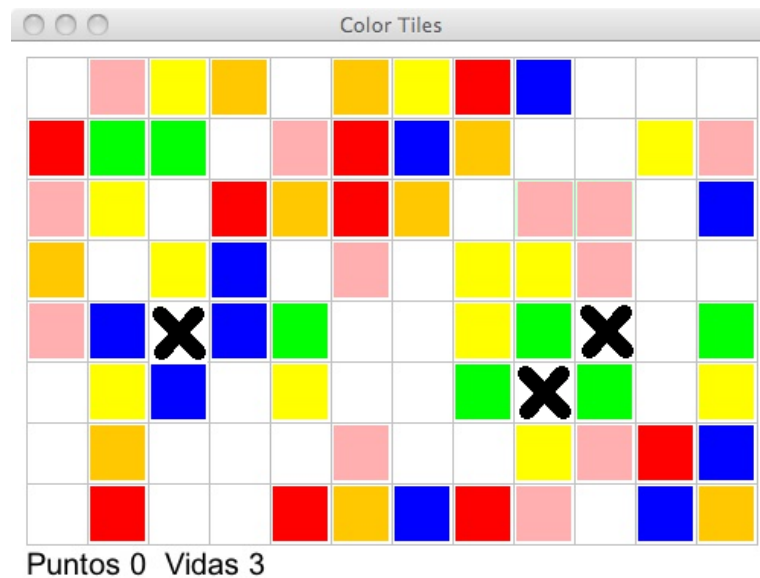
El jugador debe tratar de eliminar todos los cuadrados posibles. Para ello debe ir pulsando sucesivamente en las casillas en blanco, de forma que se eliminan los primeros cuadrados de colores en las cuatro direcciones (arriba, abajo, izquierda y derecha) y siempre y cuando entre dichos cuadrados haya más de uno del mismo color, eliminándose solamente los cuadrados que compartan color. Es decir, si alguno de esos cuadrados no tiene el mismo color que otro, se mantendrá en el tablero. Nótese que al elegir un casilla en blanco es posible que no tenga 4 cuadrados de colores en su entorno, puede tener 4, 3, 2, 1 o ninguno.

Existen un total de cuatro posibilidades de eliminación:

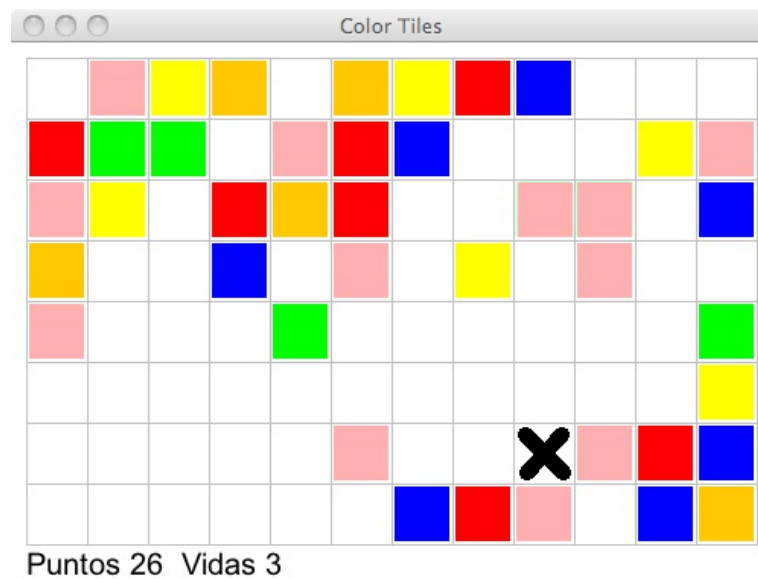
1. Que solamente haya dos cuadrados de un mismo color.



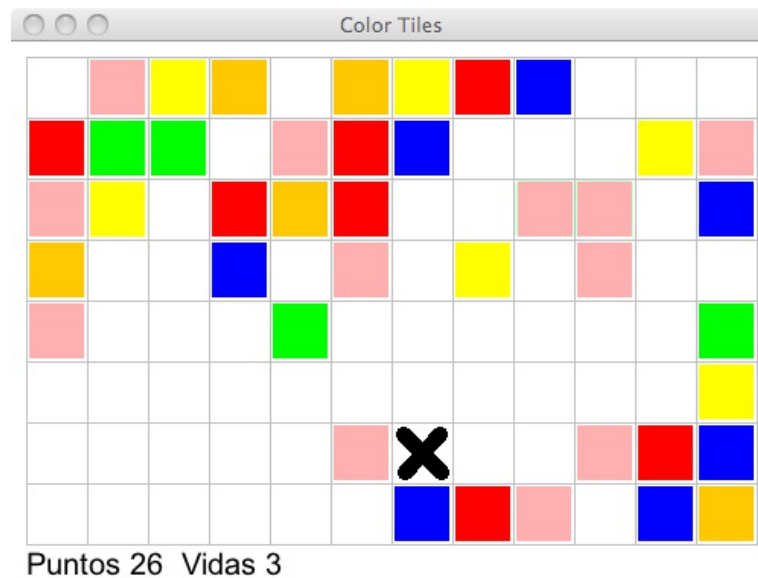
2. Que haya tres cuadrados de un mismo color.



3. Que haya cuatro cuadrados de un mismo color.



4. Que haya dos parejas de cuadrados de dos colores distintos.



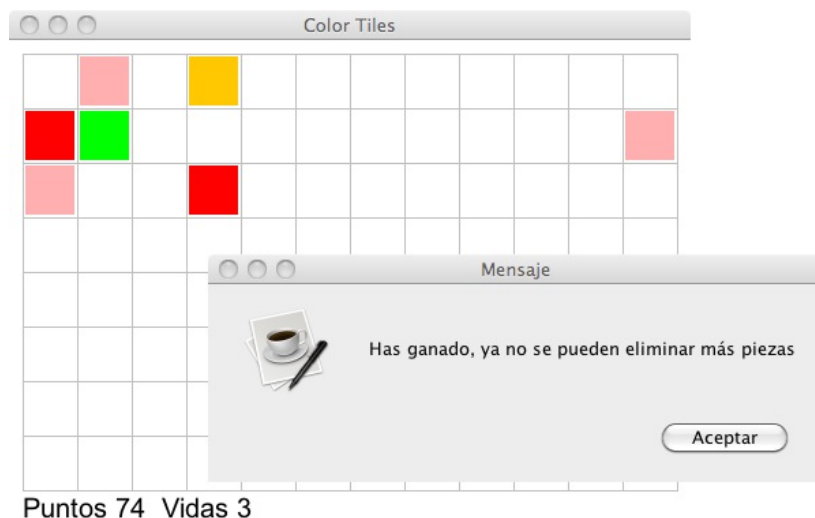
El juego debe funcionar del siguiente modo:

- **Colores.** El tablero solamente podrá tener piezas de seis colores: rojo, verde, azul, amarillo, rosa y naranja, tal como muestran los ejemplos anteriores.
- **Tablero.** El tablero tendrá unas dimensiones (nº de filas y columnas) y un número aproximado de cuadrados de colores que deben poder ser cambiados de forma sencilla. Para ello se pide que el constructor de la clase que implemente el juego (y que debe ser el método encargado de crear e inicializar el tablero) tenga dos parámetros para indicarle las dimensiones deseadas y otro parámetro con el porcentaje de casillas que contengan un cuadrado de colores (por ejemplo, un valor de 0.6 representaría que aproximadamente el 60 % de las casillas estarían ocupadas por un cuadrado de colores) .
- **Inicialización del tablero.** Un elemento que puede resultar novedoso es inicializar el tablero conteniendo un cierto número de cuadrados de colores. Esto se puede realizar fácilmente mediante números aleatorios. Para ello se recomienda el siguiente funcionamiento usando el método `Math.random()` que genera un número aleatorio entre 0 y 1. Para decidir que si una casilla va a estar en blanco u ocupada, se genera un número aleatorio y si éste es menor que el porcentaje de casillas ocupadas, entonces la casilla tendrá un cuadrado de algún color. Esto garantiza que el porcentaje de casillas ocupadas sea aproximadamente el solicitado. Para decidir el color del cuadrado, de nuevo se puede sacar un número aleatorio que decida entre los seis colores disponibles. Se podría usar el método `nextInt()` de la clase `Random`. Se realizaron ejemplos del uso de números aleatorios en las prácticas (juego Piedra, papel y tijera o clase Primitiva).
- **Realización de jugadas.** El jugador irá realizando jugadas, seleccionando la casilla elegida de alguna forma. El juego debe comprobar si la casilla elegida es válida, en cuyo caso realizará la jugada eliminando las piezas oportunas si fuera el caso, o inválida, por ejemplo cuando se escoja un casilla que no está libre, o una casilla fuera del tablero.
- **Puntuación.** El jugador irá sumando puntos con cada jugada que dependerá de las piezas que se vayan eliminado. Por cada jugada en la que elimine dos piezas sumará 2 puntos, cuando elimine tres piezas 5 puntos, y si elimina cuatro piezas en cualquiera de las dos variantes posibles sumará un total de 10 puntos.
- **Vidas.** El jugador tendrá asignadas inicialmente un cierto número de vidas. El número de vidas puede ser incluido como uno más de los parámetros de configuración al iniciar el juego. Cada vez que el usuario elija una casilla en blanco y no elimine ningún cuadrado, se le restará una vida. No se le deben restar vidas cuando elija una casilla inválida, esto es, ocupada por un cuadrado de colores o fuera del tablero. En esos casos simplemente se le permitirá elegir una nueva casilla.

- **Visualización de la partida.** El juego mostrará tras cada jugada el estado del tablero, así como la puntuación del jugador y el número de vidas con las que todavía cuenta.
- **Fin del juego.** El juego finalizará por dos motivos:
 1. **Cuando el jugador se quede sin vidas.**



2. **Cuando no se puedan eliminar más cuadrados de colores.** Para comprobar este supuesto la clase debe contar con un método que compruebe si existe alguna casilla libre que si fuera elegida produciría la eliminación de algún cuadrado de colores. Si ese método no descubre ninguna casilla en blanco que cumpla eso significará que el jugador ya no puede eliminar más cuadrados de colores y por tanto el juego finaliza.



[Implementación del juego]

Inicialización del tablero. En el constructor de la clase se debe crear el tablero (de acuerdo a los parámetros con el número de filas y columnas) y rellenarlo con las piezas iniciales (de acuerdo al porcentaje del casillas ocupadas que es el tercer parámetro del constructor). Para esto se recomienda seguir el algoritmo descrito antes cuando se explicó esta parte del juego.

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

Jugadas. La parte más complicada del juego es obviamente hacer las jugadas. Se basa en usar búsqueda en las 4 direcciones posibles y detectar las cuatro posibles casillas implicadas en la jugada.

Detectar Fin de Partida. Esta parte también es complicada. La partida acaba cuando el jugador no tiene vidas (esto es fácil de detectar) o cuando no se pueden eliminar nuevas piezas (esta parte es difícil). Este último caso también se hace con búsqueda. Se puede basar en usar un método similar al de hacer las jugadas ya que serviría para comprobar si hay al menos una casilla que produciría una jugada válida que eliminase piezas. Para probar esto, en lugar de jugar partidas completas, se recomienda inicializar directamente el tablero en una situación en la que no se pueden hacer jugadas y comprobar que el método las detecta bien.

[Opciones para realizar la práctica]

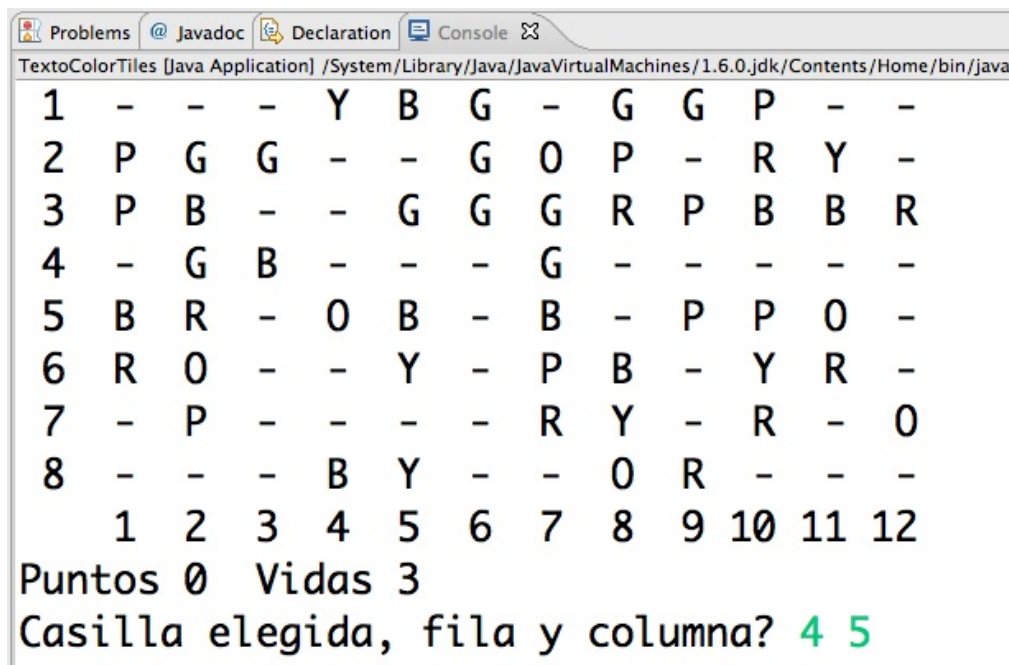
Hay dos opciones para realizar la práctica:

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. A cada color se le puede asignar una letra, por ejemplo, siguiendo sus respectivas iniciales en inglés (para evitar duplicados), tendríamos rojo ('R'), verde ('G'), azul ('B'), amarillo ('Y'), rosa ('P') y naranja ('O'). Las casillas en blanco se puede representar mediante el carácter '-'. Las jugadas se leerán de teclado, indicando el jugador la posición de la casilla elegida en cada jugada.

En las imágenes siguientes aparecen ejemplos del funcionamiento de la aplicación de consola:



```

TextoColorTiles [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java
1  -  -  -  Y  B  G  -  G  G  P  -  -
2  P  G  G  -  -  G  O  P  -  R  Y  -
3  P  B  -  -  G  G  G  R  P  B  B  R
4  -  G  B  -  -  -  G  -  -  -  -  -
5  B  R  -  O  B  -  B  -  P  P  O  -
6  R  O  -  -  Y  -  P  B  -  Y  R  -
7  -  P  -  -  -  -  R  Y  -  R  -  O
8  -  -  -  B  Y  -  -  O  R  -  -  -
    1  2  3  4  5  6  7  8  9 10 11 12
Puntos 0  Vidas 3
Casilla elegida, fila y columna? 4 5

```

Figura 1: Tablero inicial y ejemplo de eliminación de 4 piezas

```
Problems @ Javadoc Declaration Console
TextoColorTiles [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java
1  -  -  -  Y  B  G  -  G  G  P  -  -
2  P  G  G  -  -  G  O  P  -  R  Y  -
3  P  B  -  -  -  G  G  R  P  B  B  R
4  -  G  -  -  -  -  -  -  -  -  -  -
5  B  R  -  O  -  -  B  -  P  P  O  -
6  R  O  -  -  Y  -  P  B  -  Y  R  -
7  -  P  -  -  -  -  R  Y  -  R  -  O
8  -  -  -  B  Y  -  -  O  R  -  -  -
    1  2  3  4  5  6  7  8  9 10 11 12
Puntos 10  Vidas 3
Casilla elegida, fila y columna? 3 5
```

Figura 2: Eliminación de 2 piezas

```
Problems @ Javadoc Declaration Console
TextoColorTiles [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java
1  -  -  -  Y  -  G  -  G  G  P  -  -
2  P  G  G  -  -  G  O  P  -  R  Y  -
3  P  -  -  -  -  G  G  R  P  B  B  R
4  -  G  -  -  -  -  -  -  -  -  -  -
5  B  R  -  O  -  -  B  -  P  P  O  -
6  R  O  -  -  Y  -  P  B  -  Y  R  -
7  -  P  -  -  -  -  R  Y  -  R  -  O
8  -  -  -  B  Y  -  -  O  R  -  -  -
    1  2  3  4  5  6  7  8  9 10 11 12
Puntos 12  Vidas 3
Casilla elegida, fila y columna? 3 2
```

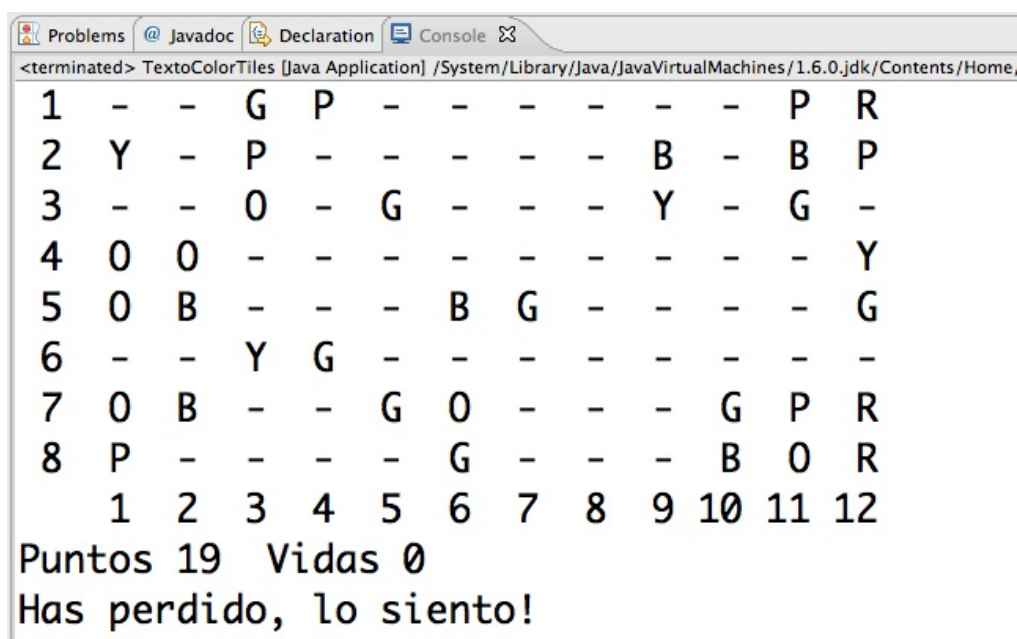
Figura 3: Eliminación de 3 piezas

```
Problems @ Javadoc Declaration Console
TextoColorTiles [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java
1 - - - Y - - - G G P - -
2 P - - - - - - 0 P - R Y -
3 P - - - - - - - R P B B -
4 - - - - - - - - - - - -
5 B R - - - - - - - P - - -
6 - - - - - - - - - - - -
7 - P - - - - - - - - - -
8 - - - B - - - - 0 - - - -
  1 2 3 4 5 6 7 8 9 10 11 12
Puntos 45  Vidas 3
Casilla elegida, fila y columna? 5 1
Pieza elegida incorrecta
```

Figura 4: Casilla no válida

```
Problems @ Javadoc Declaration Console
<terminated> TextColorTiles [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java
1 - - - Y - - - G G P - -
2 P - - - - - - - - R Y -
3 P - - - - - - - - B B -
4 - - - - - - - - - - -
5 - - - - - - - - - - -
6 - - - - - - - - - - -
7 - - - - - - - - - - -
8 - - - - - - - - - - -
  1 2 3 4 5 6 7 8 9 10 11 12
Puntos 55  Vidas 2
Has ganado, ya no se pueden eliminar más piezas
```

Figura 5: Fin del juego por haber eliminado todos los cuadrados de colores posibles



```
<terminated> TextoColorTiles [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home,  
1 - - G P - - - - - P R  
2 Y - P - - - - B - B P  
3 - - O - G - - - Y - G -  
4 O O - - - - - - - Y  
5 O B - - - B G - - - G  
6 - - Y G - - - - - -  
7 O B - - G O - - - G P R  
8 P - - - - G - - - B O R  
1 2 3 4 5 6 7 8 9 10 11 12  
Puntos 19 Vidas 0  
Has perdido, lo siento!
```

Figura 6: Fin del juego por no tener más vidas

[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 que describen el juego al inicio de este documento.

Las jugadas se leerán mediante un click de ratón, pinchando el jugador en la casilla deseada. Si la casilla elegida está en blanco se procederá a realizar la jugada, eliminando los cuadrados de colores oportunos y sumando los puntos que correspondan, o sustrayendo una vida si la casilla elegida no supone la eliminación de ningún cuadrado de colores. Si el usuario elige una casilla no válida, simplemente se ignorará el click, o se mostrará un mensaje avisando al jugador que debe pulsar sobre casillas libres.

Cuando la partida acabe se mostrará por pantalla una ventana indicando si el jugador ha ganado (por eliminar todos los cuadrados de colores posibles) o ha perdido (por quedarse sin vidas). Ver ejemplos en las páginas que describen el juego y los requisitos.

[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 8:59 horas del día 25 de Enero de 2022. 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 dibujar figuras (método `fillRect()`) y de mostrar cuadros de diálogos (`JOptionPane.showMessageDialog()`) se estudió en las sesiones prácticas de la asignatura. Para dibujar líneas y para escribir textos pueden emplearse respectivamente los métodos `drawLine()` y `drawString()`. Ambos métodos con un objeto de tipo `Graphics`.

TestTablero.java

```
1 import javax.swing.JFrame;

3 public class TestTablero {

5     public static void main(String[] args) {
6         Tablero t = new Tablero();

8         JFrame app = new JFrame("Tablero");

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;

8 public class Tablero extends JPanel {

10    //Aquí irían los atributos necesarios
```

```
12 //Constructores
13 Tablero() {
14     //El constructor debe tener los parámetros oportunos
15     //para inicializar el tablero y el juego

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

21 //Métodos de la clase que implementan el juego: básicamente hacer una
22 //jugada, dibujar el estado del tablero y comprobar si la partida se acabó

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

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

30 }

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

40         //Aquí irían las instrucciones para comprobar si el
41         //click del ratón se produjo en una posición correcta
42         //y hacer la jugada correspondiente

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

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