Entornos de Usuario
Esther Durá Martínez

Manual del Usuario

MASTERMIND

Cristal Campos Abad y Raúl Abella Bioque

Contenido

VNIVERSITAT DE VALÈNCIA

Cristal Campos Abad
Raúl Abella Bioque

Introducción

El objetivo de este proyecto es desarrollar un juego fácil y sencillo a través de la interfaz de Java. El juego trata del clásico Mistermind, "Maestro de los colores". El juego consiste en:

- Un Jugador 1 elige una combinación de colores.
- Un segundo Jugador intenta adivinar dicha combinación.
- Se disponen de 5 intentos. No adivinarlo a tiempo significa perder.
- Al finalizar cada intento se proporcionarán unas pistas de colores:
- Negro (color y posición correctos), Rojo (color correcto) o Blanco (incorrecto).

Aplicación

Para nuestro proyecto hemos utilizado la arquitectura Modelo-Vista-Controlador (M-V-C), que como sabemos, se trata de una arquitectura de software que nos permite separar los datos de la interfaz y la lógica de control en tres componentes distintos:

- Modelo: información con la cual opera el sistema en funciones definidas y declaraciones de datos.
- Vista: representación de la información, normalmente será la interfaz del usuario con la que interactuará y permitirá el desarrollo del juego.
- Controlador: encargado de responder a los eventos del usuario y llamar al modelo.

Modelo

En la clase modelo hemos definido las funciones que se van a realizar durante el juego, además de guardar en todo momento los distintos datos que recogemos, como, por ejemplo, los datos del jugador tanto nombre como combinaciones, etc.

Primero hemos declarado los parámetros que vamos a usar:

```java
/**
 * @brief Modelo de la aplicación
 * @author Raúl Abella Bioque
 * @author Cristal Campos Abad
 */
public class MastermindModelo
{
    // Número de rondas a jugar
    private int rondas = 5;
    // Ronda actual
    private int rondaActual = 0;

    // Colores elegidos por el J1
    private Color[] solucion = {Color.GRAY, Color.GRAY, Color.GRAY, Color.GRAY};
    // Colores elegidos por el J2
    private Color[] intento = {Color.GRAY, Color.GRAY, Color.GRAY, Color.GRAY};
    // Códigos de las pistas
    private int[] pistas = new int[4];

    // Todos los colores y sus nombres
    private String[] nombresColores = {"Negro", "Naranja", "Rosa", "Amarillo", "Blanco", "Rojo", "Azul", "Verde",};
    private Color[] colores = {Color.BLACK, Color.ORANGE, Color.PINK, Color.YELLOW, Color.WHITE, Color.RED, Color.BLUE, Color.GREEN};

    // Quién juega? 1 = J1, 2 = J2
    private int jActivo;
    // Usuario
    private String usuario;
    // Vector del ranking
    private String[] ranking;
    // Vector de puntos
    private int[] puntosRank = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

    // Paso
    private int paso;

    // Puntos
    int puntos = 0;

    // Imagen
    private BufferedImage imagen;
    private String imagenFileName = "";
```

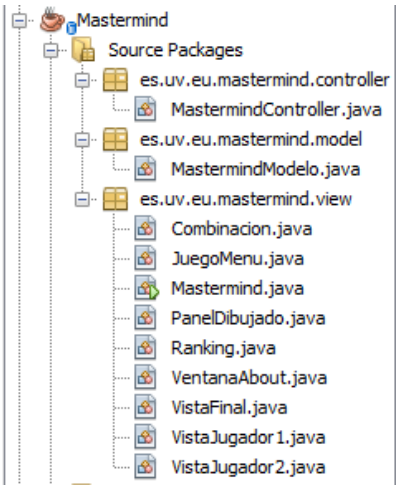
A continuación, incluimos las cabeceras de las funciones que hemos desarrollado en el modelo:

- `public void setRondas (int n)`
- `public int getRondas ()`
- `public Boolean pasarRonda (Boolean pistas)`
- `public int getRondaActual ()`
- `public void setSolucion (String s)`
- `public Color getSolucion (int i)`
- `public void setIntento (String s)`
- `public Color getIntento (int i)`
- `public void setJugador (int i)`
- `public int getJugador ()`
- `public void resetPaso ()`
- `public int getPaso ()`
- `public int [] getPistas ()`
- `public void setUsuario (String usuario)`
- `public String getUsuario ()`
- `public BufferedImage getImagen ()`
- `public String getImagenFileName ()`
- `public void setImagen (String img)`
- `public String [] getRanking ()`
- `public void setRanking ()`
- `public int [] getRankingPuntos ()`

- `public void resetIntento ()`
- `public void resetSolucion ()`
- `public void resetRondas ()`
- `public void resetPistas ()`
- `public Color convertirColor (String command)`
- `public void comprobarIntento ()`
- `public int calculaPuntos ()`
- `public int getPuntos()`
- `public void resetJuego ()`
- `public void quickSort(String[] A, int[] B, int izq, int der)`

Vista

Hemos definidos las siguientes clases para poder ir creando a lo largo del juego distintas ventanas con la que el jugador podrá visualizar e interactuar.



Creamos también el Main, que es la clase que comunica todas las clases creadas dentro de la arquitectura M-V-C entre sí.

```
/**
 * @param args the command line arguments
 */
public static void main(String[] args)
{
    MastermindModelo model = new MastermindModelo();
    VistaJugador1 view = new VistaJugador1(model);
    VistaJugador2 view2 = new VistaJugador2(model);
    VistaFinal fin = new VistaFinal(model);
    Ranking rank = new Ranking(model);
    VentanaAbout about = new VentanaAbout();
    MastermindController controlador = new MastermindController(view, view2, fin, rank, about, model);

    view.setSize(800, 550);
    view.setVisible(true);
}
```

Controlador

En esta clase se definen la mayoría de las funciones de la aplicación y qué va a pasar cuando se detecte un evento. Primero lo que hemos hecho ha sido asignar a cada una de las vistas los diferentes Listeners necesarios. Las clases (listeners) implementadas dentro de esta son:

- MastermindControllerWindowListener: Cierra la aplicación completa.
- MastermindControllerActionListener: Actúa en consecuencia del comando recibido.

```
@Override
public void actionPerformed(ActionEvent ae)
{
    String command = ae.getActionCommand();

    switch(command)
    {
        case "VolverEmpezar":
            System.out.println("Volver a Empezar juego");

            jugador1.setSize(800, 550);
            jugador1.setVisible(true);

            jugador2.setVisible(false);

            fin.setVisible(false);

            model.setJugador(1);
            model.resetJuego();

            jugador1.resetSolucion();
            jugador2.resetIntentos();
            break;
```

Cristal Campos Abad
Raúl Abella Bioque

Manual del Jugador

Como hemos visto, el juego trata de que el Jugador 2 debe adivinar la combinación que el Jugador 1 previamente ha establecido, con el objetivo de ganar puntos. En nuestro caso un jugador habrá ganado cuando en una ronda haya completado la combinación, y se llevará 100 puntos.

Ventana del Jugador 1

Es la primera ventana que se abre. Para ello hemos codificado en el controlador que todas las demás de momento no serán visibles. En la ventana 1 tenemos los siguientes elementos:

Barra de Menú.

En ella hay tres padres: Menú, ¿Qué es? y Ranking, cada cabecera está configurada con su listener para que despliegue unos ítems.

- **Menú**: en él encontramos dos ítems, 'Volver a empezar' y 'Salir'. 'Volver a empezar' lo hemos configurado para que nos lleve en cualquier momento del juego a la Ventana del Jugador 1, y además, a través de funciones del controlador, haga un reset de todos los vectores y parámetros que se han guardado durante la partida. 'Salir' simplemente cierra la aplicación.
- **¿Qué es?**: en él encontramos un ítem de 'Ayuda', donde si el jugador pulsa, a través del comando, mostramos la ventana About. Así, el jugador puede ver una breve explicación del juego y sus reglas.
- **Ranking**: en él encontramos un ítem de 'Puntos', donde si el jugador pulsa, a través del comando, mostramos la ventana Ranking. De esta forma, el jugador puede ver una clasificación de puntos de otros jugadores (en la misma sesión).

Botones

Para implementar los botones hemos hecho una clase llamada Combinación, que usaremos en ambas ventanas, tanto la del Jugador 1 como del Jugador 2.

Como vemos, en ella hemos declarado un vector de 8 botones (8 colores distintos), organizados con un GridLayout, a dos por fila. A cada botón le hemos asignado un Listener, y además, cuando se pulse uno, dependiendo de si es Jugador 1 o 2, el color se guardará en un vector u otro (*solución* para J1 e *intento* para J2).

Para diferenciar qué jugador es el que provoca el evento, hemos pasado un parámetro en función de la ventana abierta. En estos momentos nos encontramos en la Ventana del Jugador 1, por tanto cada botón que se pulse se mandará al modelo, añadiendo los colores en un vector hasta llegar a 4. Por más que se pulse ya no se guardarán más colores en el vector, ya que está lleno. Además, si vemos que el color se repite, este no se guarda duplicado en el vector.

Para saber qué color se ha pulsado hemos añadido a cada botón un comando correspondiente a su color (Ej: para el negro, el comando "Negro"). Además tenemos otro vector con todos los colores (Ej: Color.BLACK). Estos dos vectores coinciden en todas sus posiciones. Es decir, en la primera posición del primer vector encontramos "Negro", y en el segundo vector "Color.BLACK".

Una vez el controlador registra ese evento, y siempre que no hayamos elegido ya 4 colores, llama al modelo mediante la función *setSolucion*. Esta función es la que asigna el color elegido al vector correspondiente (solución, en este caso). Dentro de la misma se hace una llamada a la función *convertirColor*, que es la encargada de hacer ese cambio String a Color. Además le diremos a la vista que actualice su panel *Combinación*, para que muestre correctamente el código elegido.

Por último, en el Sur de la ventana, hemos añadido un botón, el cual al pulsarlo cierra la Ventana del Jugador 1 y abre la del Jugador 2. Además le indica al modelo que ahora va a ser el Jugador 2 el que juegue.

Ventana del Jugador 2.

Como hemos dicho antes, una vez el Jugador 1 le ha dado al botón de Empezar, la aplicación sabe que ahora será el turno del Jugador 2, y todos sus datos se guardarán en sus vectores correspondientes. Observamos que la Ventana del Jugador 2 tiene el mismo menú y los mismos botones que el Jugador 1.

Añadimos un elemento nuevo a la derecha, donde se podrán ver todos los intentos anteriores.

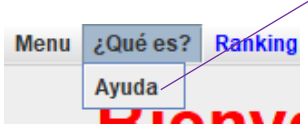
Primero, el Jugador 2 escribirá su nombre para que quede constancia de él y de sus puntos, en caso de que tenga los suficientes para estar en el ranking.

A continuación, el Jugador 2 elige una combinación y le da a Siguiente. El sistema, mediante funciones del modelo, compara con la combinación del Jugador 1 y muestra una puntación acorde.

Finalmente, le da otra vez al botón de siguiente, y si no ha acertado todas, pasará a la ronda siguiente, donde se recargará la Ventana del Jugador 2 con la combinación de la ronda anterior guardada. Además, si acertó algún color en su posición, lo mantendrá y no se podrá cambiar.

Ventana Ayuda.

Es una Ventana que le dará al jugador la oportunidad de conocer de qué trata el juego y las reglas a seguir. Recordamos que cualquier jugador puede acceder a esta desde el menú.



Ventana del Ranking.

Es una ventana que le dará al jugador la oportunidad de conocer el ranking de los jugadores, es decir, el nombre y cuántos puntos han obtenido, así como su posición. Recordamos que cualquier jugador puede acceder desde el menú.