MASTERMIND

Descripción Diagrama de Clases Presentación

Projectes de Programació: Quadrimestre Primavera 2022-23

Grup 42.5:
Elsa Boix Solina
Joel Macías Rojas
Ricard Medina Amado
Marcel Sánchez Roca

Descripción del diagrama de clases

1. LoginScreen

Breve descripción de la clase: Se encarga de hacer el LogIn de un usuario, dándole un espacio para introducir su username. Si se intenta iniciar el juego sin introducir un username saltará una excepción.

Descripción de los atributos:

- usernameField: JTextField que permite al usuario ingresar su nombre de usuario.
- loginButton: JButton que permite al usuario iniciar sesión en el juego cuando se clica.
- welcomeIcon: JLabel que muestra el ícono del juego.
- welcomeLabel: JLabel que muestra un mensaje de bienvenida al usuario.

Descripción de los métodos:

- initComponents(): Llama a las funciones encargadas de configurar la ventana, configurar los componentes que se añadirán al panel y configurar el listener para el botón de LogIn.
- configWindow(): Configura la ventana indicando su título y tamaño.
- configComponents: Configura y añade los diferentes componentes al panel. Estos son el botón de LogIn, el icono, el mensaje de bienvenida y el campo para insertar el username.
- initLogInListener: Configura el listener para el botón de LogIn. En caso de no introducir ningún nombre de usuario hace saltar una excepción.

2. Menu

Breve descripción de la clase: Implementa la interfaz gráfica del menú, desde el que tiene la opción de crear una nueva partida, ver la información de cómo jugar, consultar los rankings o cargar una partida guardada.

Descripción de los atributos:

- configButton: Botón que permite al usuario crear una nueva partida.
- rankingButton: Botón que permite al usuario ver los rankings del juego.
- rulesButton: Botón que permite al usuario ver las reglas del juego.
- partidasButton: Botón que permite al usuario cargar una partida guardada previamente.
- panel: El panel principal de la ventana, que contiene todos los componentes.
- menuLabel: Label en el que se imprimirà el texto “Menú”, para indicar que nos encontramos en esta ventana.

Descripción de los métodos:

- initComponents(): Llama a las funciones encargadas de configurar la ventana, crear los componentes que se añadirán al panel y añadir los componentes al panel.
- configWindow(): Configura la ventana indicando su título y tamaño.
- createComponents(): Crea todos los componentes definidos en los atributos.
- addComponents(GridBagConstraints gbc): Añade todos los componentes creados al panel.
- initListeners(): Inicializa el listener para ver los rankings, crear una nueva partida, cargar partida y ver las reglas.

3. PantallaConfiguracion

Breve descripción de la clase: Es la pantalla en la que se configurará una nueva partida. Los campos que se podrán escoger son el nivel de dificultad, el rol y si se quiere jugar con o sin ayuda.

Descripción de los atributos:

- lblRol: Etiqueta que muestra el texto "Rol:".
- panel: El panel principal de la ventana, que contiene todos los componentes.
- cmbRol: Desplegable para seleccionar el rol del jugador (CodeMaker o CodeBreaker).
- btnAceptar: Botón para aceptar la configuración y comenzar el juego.
- lblTitulo: Etiqueta que muestra el título de la ventana.

- exitButton: Botón para salir de la ventana.
- chkAyuda: Etiqueta que muestra el texto "Ayuda".
- cmbAyuda: Desplegable para activar o desactivar la ayuda en el juego.
- lblDificultad: Etiqueta que muestra el texto "Nivel de Dificultad:".
- cmbDificultad: Desplegable para seleccionar el nivel de dificultad del juego (1, 2 o 3).
- panel: Panel que contiene la ventana.

Descripción de los métodos:

- initComponents(): Llama a las funciones encargadas de configurar la ventana e introducir los componentes con los que el usuario podrá interactuar, aparte de los diferentes mensajes usados para identificar cada campo.
- configWindow(): Configura la ventana indicando su título y tamaño.
- configDesign(GridBagConstraints gbc): Configura el diseño de la ventana, modificando los márgenes y centrando los componentes.
- setTitle(GridBagConstraints gbc): Configura el título de esta ventana y lo añade al panel.
- setLevel(GridBagConstraints gbc): Configura y añade al panel un desplegable en el que se puede escoger qué nivel se quiere jugar(1, 2 o 3).
- setHelp(GridBagConstraints gbc): Configura y añade al panel el checkbox para seleccionar si se quiere jugar con ayuda.
- setRol(GridBagConstraints gbc): Configura y añade al panel un desplegable en el que se puede escoger con qué rol se quiere iniciar la partida.
- setAccept(GridBagConstraints gbc): Configura y añade el botón de aceptar en el panel, también configura su listener.
- setExit(GridBagConstraints gbc):Configura y añade al panel el botón de salida. También configura su listener correspondiente.

4. Ranking

Breve descripción de la clase: Implementación de la ventana ranking, que nos da la opción de visitar los tres rankings globales o el ranking personal propio de cada jugador.

Descripción de los atributos:

- easyButton: Botón que carga el ranking global del nivel fácil.
- mediumButton: Botón que carga el ranking global del nivel medio.
- hardButton: Botón que carga el ranking global del nivel difícil.
- personalButton: Botón que carga el ranking personal del usuario.
- exitButton: Botón que cierra la ventana de ranking y muestra la vista del menú principal.
- messageLabel: Etiqueta con el título de la ventana.
- levelLabel: Etiqueta que muestra el texto "Selecciona el nivel".
- labelFont: Fuente utilizada para las etiquetas.
- contentPane: Panel que contiene la ventana.

Descripción de los métodos:

- initComponents(): Llama a las funciones encargadas de configurar la ventana, configurar las fuentes que se usarán, y añadir los componentes al panel.
- configWindow(): Configura la ventana indicando su título y tamaño.
- configFonts(): Configura las distintas fuentes que se usarán
- addComponents(GridBagConstraints gbc): Añade todos los componentes creados al panel.
- initListeners(): Inicializa el listener para ver los distintos niveles de ranking, el ranking personal o salir de esta ventana.

5. RankingFacil

Breve descripción de la clase: Ventana que muestra los jugadores con las mayores puntuaciones obtenidas para el nivel fácil. Se mostrarán los 10 jugadores con mejor puntuación del Ranking.

Descripción de los atributos:

- ranking: TreeMap que guarda el ranking de nivel fácil donde la clave son los jugadores y los valores sus puntuaciones.
- exitButton: Botón que permite salir de la ventana.
- messageLabel: JLabel que muestra un mensaje al usuario indicando que se trata del ranking del nivel fácil.
- labelFont: Font que se utiliza para configurar la fuente utilizada.
- contentPane: Panel que contiene la ventana.

Descripción de los métodos:

- initComponents(TreeMap<String, Integer> gettedRanking): Llama a las funciones encargadas de configurar la ventana, configurar el diseño de la ventana, introducir el mensaje correspondiente al nivel actual y añadir el botón de salida.
- configWindow(): Configura la ventana indicando su título y tamaño.
- configDesign(GridBagConstraints gbc): Configura el diseño de la ventana, modificando los márgenes y centrando los componentes.
- setRanking(GridBagConstraints gbc): Crea un JLabel para cada posición en el ranking y lo añade al control Panel.
- setExitButton(GridBagConstraints gbc): Configura el botón de salida, que te regresa a la ventana de Rankings y añade el botón al control Panel.
- setMessage(GridBagConstraints gbc): Configura el mensaje que indica en que nivel de Ranking te encuentras y lo añade al control Panel.
- initListeners(): Inicializa el listener para salir de esta pantalla y volver a la pantalla de rankings al clicar en el botón de salida.

6. RankingMedio

Breve descripción de la clase: Ventana que muestra los jugadores con las mayores puntuaciones obtenidas para el nivel medio. Se mostrarán los 10 jugadores con mejor puntuación del Ranking.

Descripción de los atributos:

- ranking: TreeMap que guarda el ranking de nivel medio donde la clave son los jugadores y los valores sus puntuaciones.
- exitButton: Botón que permite salir de la ventana.
- messageLabel: JLabel que muestra un mensaje al usuario indicando que se trata del ranking del nivel medio.
- labelFont: Font que se utiliza para configurar la fuente utilizada.
- contentPane: Panel que contiene la ventana.

Descripción de los métodos:

- initComponents(TreeMap<String, Integer> gettedRanking): Llama a las funciones encargadas de configurar la ventana, configurar el diseño de la ventana, introducir el mensaje correspondiente al nivel actual y añadir el botón de salida.
- configWindow(): Configura la ventana indicando su título y tamaño.
- configDesign(GridBagConstraints gbc): Configura el diseño de la ventana, modificando los márgenes y centrando los componentes.
- setRanking(GridBagConstraints gbc): Crea un JLabel para cada posición en el ranking y lo añade al control Panel.
- setExitButton(GridBagConstraints gbc): Configura el botón de salida, que te regresa a la ventana de Rankings y añade el botón al control Panel.
- setMessage(GridBagConstraints gbc): Configura el mensaje que indica en que nivel de Ranking te encuentras y lo añade al control Panel.
- initListeners(): Inicializa el listener para salir de esta pantalla y volver a la pantalla de rankings al clicar en el botón de salida.

7. RankingAlto

Breve descripción de la clase: Ventana que muestra los jugadores con las mayores puntuaciones obtenidas para el nivel alto. Se mostrarán los 10 jugadores con mejor puntuación del Ranking.

Descripción de los atributos:

- ranking: TreeMap que guarda el ranking de nivel alto donde la clave son los jugadores y los valores sus puntuaciones.
- exitButton: Botón que permite salir de la ventana.
- messageLabel: JLabel que muestra un mensaje al usuario indicando que se trata del ranking del nivel alto.
- labelFont: Font que se utiliza para configurar la fuente utilizada.
- contentPane: Panel que contiene la ventana.

Descripción de los métodos:

- initComponents(TreeMap<String, Integer> gettedRanking): Llama a las funciones encargadas de configurar la ventana, configurar el diseño de la ventana, introducir el mensaje correspondiente al nivel actual y añadir el botón de salida.
- configWindow(): Configura la ventana indicando su título y tamaño.
- configDesign(GridBagConstraints gbc): Configura el diseño de la ventana, modificando los márgenes y centrando los componentes.
- setRanking(GridBagConstraints gbc): Crea un JLabel para cada posición en el ranking y lo añade al control Panel.
- setExitButton(GridBagConstraints gbc): Configura el botón de salida, que te regresa a la ventana de Rankings y añade el botón al control Panel.
- setMessage(GridBagConstraints gbc): Configura el mensaje que indica en que nivel de Ranking te encuentras y lo añade al control Panel.
- initListeners(): Inicializa el listener para salir de esta pantalla y volver a la pantalla de rankings al clicar en el botón de salida.

8. RankingPersonal

Breve descripción de la clase: Ventana en la que se muestran las cinco últimas mejores puntuaciones obtenidas por el usuario.

Descripción de los atributos:

- records: Array de enteros que almacena las mejores puntuaciones del usuario.
- exitButton: Botón que permite salir de la ventana.
- messageLabel: JLabel que muestra un mensaje indicando que se están mostrando las mejores puntuaciones del usuario.
- labelFont: Font que define el tipo de fuente a utilizar en los JLabels.
- contentPane: Panel que contiene la ventana.

Descripción de los métodos:

- initComponents(int[] gettedRecords): Llama a las funciones encargadas de configurar la ventana, configurar el diseño de la ventana, introducir el mensaje de que la ventana es de records personales y añadir el botón de salida.
- configWindow(): Configura la ventana indicando su título y tamaño.
- configDesign(GridBagConstraints gbc): Configura el diseño de la ventana, modificando los márgenes y centrando los componentes.
- setRecords(GridBagConstraints gbc): Crea un JLabel para cada una de las cinco mejores puntuaciones del usuario. En caso de no tenerlas son 0.
- setExitButton(GridBagConstraints gbc): Configura el botón de salida, que te regresa a la ventana de Rankings y añade el botón al control Panel.
- setMessage(GridBagConstraints gbc): Configura el mensaje que indica que estamos en la ventana de Records personales y lo añade al control Panel.
- initListeners(): Inicializa el listener para salir de esta pantalla y volver a la pantalla de rankings al clicar en el botón de salida.

9. TurnosMaquina

Breve descripción de la clase: Esta clase se encarga de simular como ha llegado la máquina a la solución hecha por el jugador, introduciendo las combinaciones que ha necesitado en el tablero.

Descripción de los atributos:

- tablero: Es una matriz de objetos Color que representa el tablero del juego. Almacena los colores seleccionados por la máquina en cada posición del tablero.
- NUMERO_FILAS: Es una constante entera que representa el número de filas en el tablero del juego.
- NUMERO_COLUMNAS: Es una variable estática que representa el número de columnas en el tablero del juego y puede ser configurado por el jugador.
- panelTablero: JPanel en el que se configurará el tablero.
- botonContinuar: Botón que mediante un listener permite acceder a la siguiente pantalla.
- message: Label que indica que se está resolviendo la combinación.

Descripción de los métodos:

- initComponents(Color[][] combinaciones): Se encarga de inicializar, llamando al resto de funciones, los atributos e introducirlos en el panel. Por parámetro recibe una matriz que hace referencia a las combinaciones hechas por la máquina para encontrar la combinación de solución.
- configWindow(): Configura la ventana indicando su título y tamaño.
- setTabero(Color[][] combinaciones): Configura el tablero sobre el que se llevará a cabo la simulación de las combinaciones hechas por la máquina.
- setBolas(Color[][] combinaciones): Configura todo lo relativo a la inserción de bolas dentro del tablero para llevar a cabo la simulación de los intentos hechos por la máquina. Utiliza un timer para una mejor simulación, dejando un corto espacio de tiempo para que el usuario vea cómo ha ido evolucionando la solución.

- setComponentes(): Configura un mensaje que informa al usuario de que se está llevando a cabo la simulación. También configura el botón de continuar para acceder a la siguiente ventana con su listener correspondiente.
- addComponents(): Añade al panel el mensaje, el tablero y el botón de continuar.

10. CargarPartida

Breve descripción de la clase: Ventana que permite al usuario seleccionar una partida guardada previamente y cargarla para seguir jugandola.

Descripción de los atributos:

- gameField: Campo de texto que permite al usuario ingresar el número de la partida que desea cargar.
- acceptButton: Botón que activa la función de cargar la partida seleccionada.
- exitButton: Botón que cierra la ventana y devuelve al usuario al menú principal.
- partidas: Lista de las partidas guardadas previamente por el usuario.
- contentPanel: Panel que contiene la ventana.
- listModel: listModel en el que se añadirá en forma de String las partidas guardadas.
- dateList: Lista en la que se mostrarán las partidas y se podrán seleccionar.
- selectedIndex: Índice en el que se guarda que elemento de la dateList está seleccionado.
- scrollPane: Panel desplazable que se usa para mostrar la lista.
- selectedDateLabel: Etiqueta que se usa para mostrar que partida está seleccionada.

Descripción de los métodos:

- initComponents(): Llama a las funciones encargadas de configurar la ventana, configurar el diseño de la ventana, introducir el mensaje de que la

ventana es la encargada de cargar partidas y configurar los botones de selección de que partida quieres cargar.

- configWindow(): Configura la ventana indicando su título y tamaño.
- configDesign(GridBagConstraints gbc): Configura el diseño de la ventana, modificando los márgenes y centrando los componentes. Configura el tamaño del scroll panel, agrega el JLabel con la fecha seleccionada y agrega el JScrollPanel.
- setText(GridBagConstraints gbc): Setea el texto dependiendo de si hay partidas guardadas o no.
- addPartidas(GridBagConstraints gbc): Añade todas las partidas guardadas del usuario a la lista de partidas(listModel).
- configPlayButton(GridBagConstraints gbc): Configura y añade al panel el botón de jugar.
- configExitButton(GridBagConstraints gbc): Configura y añade al panel el botón de salir.
- initListeners(): Inicializa los listeners del botón de salir, del botón de jugar y de la DateList.

11. EndGame

Breve descripción de la clase: Se encarga de informar al usuario de que la partida ya ha finalizado. Se muestra la puntuación que se ha obtenido en la partida y se permite volver al menú principal clicando un botón.

Descripción de los atributos:

- titleLabel: Un objeto JLabel que muestra el título "Fin del juego".
- scoreLabel: Un objeto JLabel que muestra la puntuación obtenida por el jugador.
- backButton: Un objeto JButton que permite al jugador volver al menú principal del juego.
- mainPanel: JPanel principal en el que se añadirán todos los componentes creados.

- centerPanel: JPanel al que se le añade el panel principal para usar FlowLayout y centrar los componentes.
- screenSize: Dimensión que define el tamaño de la ventana

Descripción de los métodos:

- initComponents(Integer score): Llama a las funciones encargadas de configurar la ventana, crear los componentes, configurar el listener para el botón de vuelta al menú principal, crear el panel principal, crear el panel central y centrar la ventana.
- configWindow(): Configura la ventana indicando su título y tamaño.
- createComponents(Integer score): Crea los componentes que conformarán la ventana. Estos son titleLabel, scoreLabel y backButton.
- initListener(): Configura el listener para el botón de vuelta al menú principal.
- configMainPanel(): Crea el JPanel principal, lo configura y añade los componentes.
- centerWindow(): Se encarga de centrar la ventana.

12. CodeMaker

Breve descripción de la clase: Se encarga de la pantalla que corresponde al turno de CodeMaker. Permite al usuario introducir la combinación solución. En caso de ser una combinación incompleta o repetir colores en el nivel 1, saltará una excepción. Permite volver al menú principal y salir de la partida si se desea.

Descripción de los atributos:

- message: JLabel que indica que hay que seleccionar un color para empezar.
- colorSeleccionado: Almacena el color seleccionado por el jugador.
- tablero: Array de Color que representa la combinación introducida en el tablero.
- NUMERO_COLUMNAS: Entero que hace referencia a la cantidad de columnas que hay en el tablero. En nivel 1 y 2 será igual a 4 y en el nivel 3 será igual a 5.
- panelTablero: JPanel que representa el tablero en el que se introducirá la combinación solución.

- panelPrincipal: JPanel principal al que se le añadirán todos los componentes de la ventana.
- panelBolas: JPanel en el que se añaden las seis bolas con las que se podrá interactuar para crear una combinación.
- Bola bola1, bola2, bola3, bola4, bola5, bola6: Bolas con las que se podrá interactuar mediante un clic para seleccionarlas y generar una combinación.
- botonSalir: Botón que permite salir de la partida y volver al menú principal.
- panelSalir: Panel donde se gestiona la salida de la partida y se pide conformación para salir de la partida.
- botonAceptar: Botón para aceptar una combinación solución. En caso de ser una combinación incompleta o repetir colores en el nivel 1 saltará una excepción.

Descripción de los métodos:

- initComponents(Integer numCols): Llama a las funciones encargadas de configurar la ventana, configurar el tablero, configurar el mensaje de la parte superior de la ventana, configurar el panel principal, configurar las bolas, añadir los componentes al panel, configurar el panel de salida y configurar los botones de aceptar y salir.
- configWindow(): Configura la ventana indicando su título y tamaño.
- setBoard(numCols): Configura el panel del tablero y llama a la función encargada de pintar el tablero.
- configMessage(): Configura el mensaje de la parte superior de la ventana.
- configMainPanel(): Configura el panel principal.
- configBalls(): Crea el panel de las bolas y las seis bolas. Añade las bolas al panel y crea un listener para cada una para que el usuario pueda interactuar con ellas.
- addComponents(): Añade el message, el panelTablero y el panelBolas al panel principal.
- configExitButton(): Configura el botón de salida y su listener respectivo. Cuando se clica el botón de salida se genera un panel nuevo con un mensaje y un botón que piden confirmación para abandonar la partida. También se configura el listener de este botón de confirmación,
- configExitPanel(): Configura el panel para cuando se quiere salir de la partida.

- configAcceptButton(): Configura el botón y el listener para introducir una combinación.
- dibujarTablero(JPanel panelTablero): Método privado utilizado internamente para dibujar el tablero de juego en un panel dado. Crea y agrega paneles personalizados (Bola) en el panel del tablero, representando cada espacio del tablero con un color y otras propiedades.
- BolaMouseListener: Clase interna que implementa MouseListener. Maneja los eventos de clic en las bolas de colores del panel de selección y en los espacios del tablero. Actualiza el color seleccionado y asigna colores al tablero según las interacciones del jugador.

13. Bola

Breve descripción de la clase:

La clase "Bola" es una clase que representa una bola gráfica en una interfaz de usuario. Esta clase extiende de JPanel, lo que significa que puede ser agregada y visualizada en un contenedor gráfico.

Descripción de los atributos:

- color: Este atributo de tipo Color almacena el color de la bola. El color se utiliza para dibujar la forma de la bola en la interfaz gráfica.
- colorSeleccionado: Es un atributo de tipo Color que guarda el color seleccionado para la bola. Cuando el usuario hace clic en la bola, se almacena el color actual en este atributo. Esto permite cambiar el color de la bola cuando se selecciona un nuevo color.
- disabled: Es un atributo de tipo boolean que indica si la bola está deshabilitada. Si la bola está deshabilitada, no se pueden realizar acciones sobre ella, como cambiar su color o seleccionarla.
- TAMAÑO_BOLA: Este atributo de tipo int representa el tamaño de la bola. El valor se utiliza para establecer las dimensiones de la bola en la interfaz gráfica. El tamaño de la bola puede ser diferente dependiendo de si se trata de una bola solitaria o una bola del tablero.

- seleccionada: Es un atributo de tipo boolean que indica si la bola está seleccionada. Cuando la bola está seleccionada, se puede arrastrar por la interfaz gráfica moviéndola con el puntero del ratón.
- bolaTablero: Es un atributo de tipo boolean que indica si la bola pertenece al tablero. Si el color de la bola coincide con un color específico, se considera que la bola pertenece al tablero. Esto puede afectar su apariencia o comportamiento en la interfaz gráfica.
- fila: Este atributo de tipo Integer almacena el número de fila de la bola. Se utiliza para identificar la posición de la bola en una estructura de datos o en relación con otras bolas en el tablero.
- columna: Este atributo de tipo Integer guarda el número de columna de la bola. Al igual que el atributo fila, se utiliza para identificar la posición de la bola en una estructura de datos o en relación con otras bolas en el tablero.

Descripción de los métodos:

- Bola(Color colorBola, boolean sol, boolean disabled, Integer numColumna, Integer numFila): Este es el constructor de la clase. Recibe el color de la bola, un valor booleano que indica si la bola corresponde a una bola “solución” (es decir, es la que el usuario selecciona) , un valor booleano que indica si la bola está deshabilitada, y los números de columna y fila de la bola. Al crear una instancia de la clase "Bola", se deben proporcionar estos valores para configurar correctamente el estado inicial de la bola.
- getFila(): Este método devuelve el número de fila de la bola en el tablero. Permite acceder al valor de la fila desde otras partes del programa.
- getColumna(): Este método devuelve el número de columna de la bola en el tablero. Proporciona acceso al valor de la columna desde otras partes del programa.
- changeDisabled(boolean disabled): Este método permite cambiar el estado de habilitación de la bola. Recibe un valor booleano que indica si la bola debe estar habilitada o deshabilitada. Al llamar a este método y proporcionar el valor correspondiente, se puede cambiar la capacidad de interactuar con la bola en la interfaz gráfica.
- getColor(): Este método devuelve el color actual de la bola. Permite obtener el color de la bola desde otras clases del programa para realizar operaciones o comparaciones.

- isDisabled(): Este método devuelve un valor booleano que indica si la bola está deshabilitada. Es útil para verificar el estado de habilitación de la bola en diferentes situaciones y realizar acciones en consecuencia.
- paintComponent(Graphics g): Este método es una implementación sobrescrita del método paintComponent de la clase JPanel. Se encarga de pintar la bola en la interfaz gráfica. Utiliza el objeto Graphics proporcionado como parámetro para dibujar la forma de la bola con el color correspondiente en las coordenadas especificadas.
- esColorTablero(): Este método devuelve un valor booleano que indica si el color de la bola pertenece al tablero. Compara el color de la bola con un color específico para determinar si la bola forma parte del tablero o no. Esto puede ser útil para realizar acciones específicas en las bolas del tablero.
- setColorTablero(): Este método establece el color de la bola como el color del tablero. Cambia el color de la bola al color correspondiente al tablero (marrón) y vuelve a dibujar la bola en la interfaz gráfica para reflejar el cambio.
- setColor(Color colorSeleccionado2): Este método permite establecer el color de la bola utilizando un color seleccionado. Recibe un objeto Color como parámetro y asigna ese color a la bola. La bola se vuelve a dibujar en la interfaz gráfica con el nuevo color.

14. CtrlPresentacion

Breve descripción de la clase: La clase CtrlPresentacion actúa como controlador de la capa de presentación en nuestra aplicación. Es responsable de gestionar la interacción entre las vistas de la interfaz de usuario y la lógica del dominio.

Descripción de los atributos:

- controladorDominio (tipo: CtrlDominio): Un atributo estático que representa una instancia del controlador de dominio CtrlDominio. Permite la comunicación entre la capa de presentación y la capa de dominio.

- numCols (tipo: Integer): Un atributo estático que almacena el número de columnas utilizado en el juego.
- setAyuda (tipo: Boolean): Un atributo estático que indica si la opción de ayuda está activada o desactivada.
- usernameAct (tipo: String): Un atributo estático que almacena el nombre de usuario actualmente en uso.
- nivel (tipo: Integer): Un atributo estático que representa el nivel de dificultad del juego.
- lastRonda (tipo: Integer): Un atributo estático que indica el número de la última ronda jugada.

Descripción de los métodos:

- transformCombination(Color[] colors): Un método privado que toma un array de colores y devuelve una lista de colores del dominio principal (main.domain.Color). Este método se utiliza para convertir los colores seleccionados por el jugador al formato de la enumeration del dominio Color utilizado en el dominio principal.
- carregarvistaLogin(): Carga la vista de inicio de sesión (LoginScreen).
- carregarvistaMastermindGame(): Carga la vista del juego Mastermind (MastermindGame).
- loginUser(String username): Permite al usuario iniciar sesión proporcionando un nombre de usuario. Este método también inicia sesión en el controlador de dominio y restaura el ranking.
- carregarEndGame(): Carga la vista de fin de juego (EndGame) y muestra la puntuación obtenida.
- carregaCodeMaker(): Carga la vista del CodeMaker (CodeMaker), que permite al jugador establecer la combinación a adivinar.
- carregarVistaMenu(): Carga la vista del menú principal (Menu).
- carregarVistaConfiguracion(): Carga la vista de configuración del juego (PantallaConfiguracion).
- carregarVistaRanking(): Carga la vista del ranking general (Ranking).
- carregarVistaRanking1(): Carga la vista del ranking del nivel Fácil (RankingFacil).
- carregarVistaRanking2(): Carga la vista del ranking del nivel Medio (RankingMedio).

- carregarVistaRanking3(): Carga la vista del ranking del nivel Alto (RankingAlto).
- carregarVistaRankingPersonal(): Carga la vista del ranking personal del jugador (RankingPersonal).
- carregarVistaCargarPartida(): Carga la vista para cargar una partida guardada (CargarPartida).
- getPartidasGuardadas(): Obtiene la lista de partidas guardadas, representada como una lista de pares (Pair<String, Date>), que contiene el nombre de la partida y la fecha en la que se guardó.
- getJuegoInfo(): Obtiene la información del juego, devolviendo un String que contiene detalles sobre el juego en curso, como el nivel de dificultad y las reglas específicas del juego.
- getJuegoInfoPuntuacion(): Obtiene la información del juego junto con la puntuación actual, devolviendo un String que incluye detalles sobre el juego y la puntuación obtenida hasta el momento.
- newGame(Integer level, Boolean rol, Boolean ayuda): Inicia un nuevo juego con el nivel de dificultad especificado, el rol del jugador (CodeBreaker o CodeMaker) y la opción de ayuda activada o desactivada. Este método crea una partida en el controlador de dominio y configura los atributos correspondientes para el juego.
- submit(Color[] colors): Envía la combinación de colores seleccionada por el jugador al controlador de dominio para evaluarla. Devuelve un array de colores que representa el feedback dado por el juego en respuesta a la combinación enviada. Los colores en el array de feedback indican si los colores de la combinación son correctos y están en la posición correcta (color negro), si son correctos pero están en la posición incorrecta (color blanco) o si no son correctos (color gris).
- reiniciarPartida(): Reinicia la partida actual, restableciendo todos los elementos y configuraciones relacionados con el juego. Crea una nueva instancia de la vista del juego MastermindGame y la muestra al jugador.
- setAyudaPartida(): Activa la opción de ayuda para la partida actual, solicitando ayuda al controlador de dominio.
- setSolution(Color[] colors): Establece la combinación solución para la partida actual. Toma una combinación de colores seleccionada por el jugador y la

envía al controlador de dominio para establecerla como la combinación solución. Devuelve un Integer que representa el número de rondas jugadas hasta ese momento.

- finRonda(String turnoAct): Realiza las acciones correspondientes al finalizar una ronda del juego. Recibe como parámetro el turno actual y, según las condiciones, muestra la vista de fin de juego si se han completado suficientes rondas, o bien muestra la vista del CodeMaker o del MastermindGame para continuar jugando.
- cargaCombMaquina(): Carga la vista de los turnos de la máquina (TurnosMaquina), mostrando las combinaciones realizadas por la máquina en los turnos anteriores.
- guardarPartida(): Llama a la función encargada de guardar partida en el Dominio.
- cargarPartida(string Date): Llama a la función encargada de cargar partida en el Dominio para que cargue la partida con la fecha indicada por parámetro.

15. MasterMindGame

Breve descripción de la clase: Se encarga de manejar la lógica y la interfaz gráfica del juego. En esta clase es donde se lleva a cabo el turno de CodeBreaker, pudiendo insertar combinaciones y recibiendo el feedback para estas. Aparte permite activar la ayuda, salir de la partida guardandola y reiniciar el turno.

Descripción de los atributos:

- colorSeleccionado: Es un objeto de la clase Color que representa el color seleccionado por el jugador.
- serialVersionUID: Es un atributo estático final de tipo long que se utiliza para controlar la versión de serialización de la clase.
- NUMERO_FILAS: Es una constante entera que representa el número de filas en el tablero del juego.
- NUMERO_COLUMNAS: Es una variable estática que representa el número de columnas en el tablero del juego y puede ser configurado por el jugador.
- tablero: Es una matriz de objetos Color que representa el tablero del juego. Almacena los colores seleccionados por el jugador en cada posición.

- solucion: Matriz de objetos Bola que representa la solución del juego. Almacena las bolas de colores que representan la combinación correcta.
- botonWelcome: JLabel que muestra un mensaje de “seleccionar color para empezar”.
- filasRest: Es un atributo de tipo Integer que representa el número de filas restantes en el tablero.
- setAyuda: Es un atributo de tipo Boolean que indica si la opción de ayuda está activada o no en el juego.
- panelPausa: JPanel que se utilizará cuando la partida esté en modo pausa.
- panelTablero: JPanel utilizado para representar el tablero.
- panelBolas : JPanel utilizado para mostrar las seis bolas:
- bola1, bola2, bola3, bola4, bola5, bola6: Bolas con las que el usuario podrá interactuar para crear sus combinaciones.
- botonSalir: Botón para salir de la partida.
- panelSalir: JPanel usado después de que el usuario clique al botón de salir.
- botonAyuda: Botón para activar la ayuda durante la partida.
- submitB:Botón para aceptar una combinación hecha por el usuario.
- botonPausar: Botón para pausar la partida.
- panelBotones: JPanel en el que estarán todos los botones.
- botonSize: Dimension utilizada para los botones.
- botonReiniciar : Botón usado para reiniciar la partida.
- botonReanudarPausa : Botón que será visible después de pausar la partida y permitirá reanudarla.
- botonSalirPausa : Botón que será visible después de pausar la partida y permitirá salir de la partida.
- panelSur: JPanel en el que se ubicarán el panel de Bolas, Botones y Salir.

Descripción de los métodos:

- initComponents(Integer numCols, Boolean ayuda): Setea el número de columnas y la ayuda. Es la función encargada de llamar al resto de funciones que configurarán las ventanas, sus componentes y los listeners.

- dibujarTablero(JPanel panelTablero): Este método se utiliza para dibujar el tablero del juego en la interfaz gráfica. Recibe como parámetro un objeto de la clase JPanel que representa el panel donde se mostrará el tablero. El método utiliza bucles for para iterar sobre las filas y columnas del tablero y agrega los componentes necesarios para representar cada posición del tablero.
- configWindow(): Configura la ventana indicando su título y tamaño.
- paudePanel(): Configura el panel de pausa, seteando el color de fondo y tamaño.
- boardPanel(): Configura el panel en el que se dibujará el tablero y llama a la función encargada de dibujarlo(dibujarTablero).
- ballsPanel(): Configura el panel de las Bolas y las añade todas al panel.
- ballsListeners(); Crea un MouseListener para cada una de las Bolas.
- configExit(): Configura el botón de salida y su listener correspondiente. Aparte configura las opciones de “Guardar” o “No Guardar Partida” que aparecen después de la interacción con el botón de salir. Configura los listeners para los botones referentes a guardar o no la partida.
- configHelp(): Configura el botón de ayuda y su listener correspondiente. También configura los botones que aparecen después de clicar en el botón de ayuda pidiendo confirmación para realizar la acción. Configura los listeners para los botones de confirmación.
- configSubmit(): Configura el botón de aceptar una combinación introducida y su listener correspondiente. Se encarga de mostrar el feedback correspondiente a la combinación introducida. Si se ha acabado la partida o el turno, se muestra un mensaje y configura un botón con su listener para acceder a la siguiente pantalla.
- configPause(): Configura el botón de pausar y su listener correspondiente. En el listener se modifican los paneles para ponerlos en “modo pausa” y se desactiva la interacción con los botones que no correspondan a este modo.
- configRestart(): Configura el botón de reiniciar y su listener correspondiente para poder reiniciar el turno.
- pauseGestion(): Configura los botones del modo pausa que permiten reanudar o salir de la partida y sus listeners correspondientes.

- configSouthPanel: Configura el panel de la parte inferior en el que se añaden el panel de Bolas, Botones y Salir.
- configMessage: Configura el JLabel que se muestra en la parte superior de la ventana.