Manual Técnico

CLASE MAIN

public static void main(String[] args)
Es el main; nunca se cierra mientras el programa esté corriendo. Este inicia el programa.

CLASE MENU

public Menu()
Crea un menú principal.

private void syeMouseClicked(java.awt.event.MouseEvent evt)
Abre la ventana de serpientes y escaleras.

private void thMouseClicked(java.awt.event.MouseEvent evt)
Abre la ventana de torres de Hanoi.

private void LacercaDeMouseClicked(java.awt.event.MouseEvent evt)
Despliega un mensaje con los datos del creador.

private void puMouseClicked(java.awt.event.MouseEvent evt)
Abre la ventana de puzzle.

private void salirMouseClicked(java.awt.event.MouseEvent evt)
Cierra el programa.

CLASE PUZZLE

public Puzzle()
Crea un nuevo Puzzle y lo inicializa.

private void mAbrirActionPerformed(java.awt.event.ActionEvent evt)
Abre un juego guardado.

private void mGuardarActionPerformed(java.awt.event.ActionEvent evt)
Guarda un juego.

private void mPuntuacionesActionPerformed(java.awt.event.ActionEvent evt)
Abre la ventana de puntuaciones.

private void mSalirActionPerformed(java.awt.event.ActionEvent evt)
Sale al menu principal.

private void BdesordenarActionPerformed(java.awt.event.ActionEvent evt)
Desordena las imágenes para empezar un nuevo juego (si el jugador lo aprueba).

private void BpausaActionPerformed(java.awt.event.ActionEvent evt)
Pausa el juego hasta que el jugador mueva las imágenes.

private void CuadroMouseClicked(java.awt.event.MouseEvent evt)
Metodo que se ejecuta al hacer clic a una imagen. Enciende el cronómetro.

private void mover(cuadro cuad,byte posActual)
Método que valida cuál es el cuadro que se tiene que mover. Invoca al método Proceso para mover el cuadro.

private void ini()
Inicializa los objetos icono y Cuadro, y asigna el icono a cada Cuadro. También asigna vacio=8 porque esa posición tiene el Cuadro que está vacío al inicializarse.

private void Proceso(cuadro Temp, cuadro Cuad, byte PosActual)
Mueve el cuadro en donde se originó el evento y cambia el valor a vacio. Si el cronómetro está detenido, lo inicia.

public void correrTiempo()
Es el cuerpo del cronómetro. Lleva el control del tiempo.

private void TerminarJuego()
Método que termina el juego porque el jugador ganó. Detiene el cronómetro, bloquea el tablero y pide el nombre al jugador para ponerlos en los puntajes altos si es que lo merece.

private void nuevoJuego()
Pone el cronometro y los contadores en 0 y asigna false a la variable Jterminado.

private void cambiarImagen()
Asigna una nueva imagen a cada icono y despues le asigna a cada Cuadro un icono, en la posición que corresponde cada icono. Detiene el cronómetro y asigna true a Jterminado para que al llamar al método que desordena las imágenes no se pida la confirmación de un nuevo juego; después se ejecuta nuevoJuego y asigna false a Jterminado.

private boolean preguntar(String mensaje, String titulo)
Solo es un JOptionPane que despliega una pregunta y toma el valor que el jugador escoge.

CLASE CUADRO

public cuadro(byte iconoInicial,short x,short y,ImageIcon i)
Crea un nuevo cuadro.
Parametros:
iconoInicial: índice del icono con el que inicia.
x: posición en x.

y: posición en y.
i: el icono con el que empieza.

public cuadro()
Crea un cuadro vacio.

CLASE PTOOLS

public Ptools()
Crea una ventana nueva para las 10 puntuaciones más altas.

private void BaceptarActionPerformed(java.awt.event.ActionEvent evt)
Oculta la ventana de las puntuaciones.

void guardar(cuadro[] Cuadro, byte vacio, boolean Jterminado, String carpetaImagen, byte segs, byte mins, byte hrs, int movs)
Guarda un juego.

String abrir(JTextArea Texto)
Abre un juego guardado.

void ponerEnPuntajes(String nombre, String tiempo, String carpetaImagen, int movs)
Si el jugador tiene un tiempo entre los mejores 10, lo agrega a los punteos altos (llama al método updateTabla).

private void updateTabla(String file, JTable tabla, String nombre, String tiempo, int movs)
Actualiza y ordena la tabla y el archivo de punteos que corresponde.

void cargarPuntuaciones()
Carga las puntuaciones en las tablas (si existe algun punteo). Este método lo invoca la clase Puzzle al entrar a Puntuaciones.

private void limpiarTabla(JTable tabla)
llena la tabla de " " para evitar el NullPointerException.

void reportar(int movs, String tiempo, String carpetaImagen, String nombre)
Genera los reportes de todos los juegos que se han ganado.

CLASE PPILA

public Ppila( )
Crea una nueva Ppila con el fondo o final como Null.

public boolean esVacia( )
Dice si la Ppila está vacia o no.

public void vaciar( )
Vacía la Ppila.

public void push(byte numero)
Ingresa un nuevo nodo a la Ppila con el valor que se da como parámetro.

public byte getNumero( ) throws Exception
Devuelve el valor del atributo 'Numero' que se ingresó de último.

public void pop( ) throws Exception
Elimina el último nodo ingresado.

public boolean esRepetido(byte numero)
Dice si el parametro es repetido o no.

CLASE PNODO

public PNodo(byte num, PNodo sgte)
Crea un PNodo.

public PNodo(byte num)
Crea un PNodo con Psiguiente=null.

CLASE ESCALERAS

public Escaleras()
Crea el juego Escaleras. Llama al método cargarIconos().

private void mPosicionesActionPerformed(java.awt.event.ActionEvent evt)
Abre la ventana de las posiciones de cada casilla en el archivo que está en uso.

private void mJugarActionPerformed(java.awt.event.ActionEvent evt)
Si ya se escogió un tablero, comienza un juego nuevo y abre el panel para agregar los jugadores. Invoca Limpiar().

private void BterminarActionPerformed(java.awt.event.ActionEvent evt)
Hace visibles a los jugadores agregados, los agrega al form, los posiciona, invoca ActualizarJuego() y esconde el panel para agregar jugadores.

private void BborrarTodosActionPerformed(java.awt.event.ActionEvent evt)
Si el usuario confirma, borra todos los jugadores que se han agregado a esta partida.

private void BagregarJugadorActionPerformed(java.awt.event.ActionEvent evt)
Si el tablero tiene casilla de inicio, agrega un nuevo jugador a la cola con sus atributos, pero no lo hace visible, ni lo agrega al form ni lo posiciona. (ver BterminarActionPerformed).

private void mCargarActionPerformed(java.awt.event.ActionEvent evt)
Abre una ventana para escoger el archivo que tiene los datos del tablero que se va a usar. Después llama al método crearTablero().

private void dadoMouseClicked(java.awt.event.MouseEvent evt)
Genera un número aleatorio entre 0 y 4 para el dado. Después bloquea el dado para que no tiren hasta que sea habilitado otra vez (se habilita cuando ya le toque al otro jugador). También invoca MoverJugador(int), ActualizarJuego() y actualiza datos de la cola y del Jugador actual.

private void crearTablero()
Crea cada casilla del tablero leyendo de un archivo. Le asigna a cada una su penalización, ubicación y sus propiedades.

private void cargarIconos()
Carga las imágenes para los íconos de cada casilla, pero no los asigna, solo los carga.
Indice 0 = inicio;
Indice 1 = fin;
Indice 2 = normal;
Indice 3 = serpiente;
Indice 4 = escalera;
Indice 5 = tirar;

private void ActualizarJuego()
Actualiza los labels con los datos del jugador actual y activa el dado para que el jugador actual pueda tirar si es que el juego no ha terminado.

private void Limpiar()
Limpia los labels que llevan datos de los jugadores y reinicia las variables necesarias para que el juego funcione bien (turnosExtra, juegoTerminado y permitirPenalizaciones).

private void MoverJugador(int cantidad)
Hace que el Jugador se mueva hacia la casilla que indica el parámetro cantidad, invocando Caminar(int,int). Después invoca Penalización(), cuando el Jugador llego a su Casilla destino.

private Casilla Caminar(int nombreCasilla,int espera)
Mueve al Jugador a la Casilla que tenga el nombre del parámetro nombreCasilla. También retorna la casilla a la que se movió el Jugador.

private boolean existeCasilla(int i)
Retorna true solo si existe una Casilla con el nombre del parámetro i.

private void Penalizacion()
Hace la penalización de la Casilla actual al Jugador que esté en ella solo si no hay penalizaciones activas.

private void reiniciarJuego()
Borra todos los Jugadores del juego y de la cola. Después invoca Limpiar().

private void terminarJuego()
Termina el juego.

CLASE ETOOLS

private void BregresarActionPerformed(java.awt.event.ActionEvent evt)
Oculta la ventana.

void HacerReportes(String nombre,int lanzamientos,String fecha,String hora,String competidores)
Hace los reportes en html.

void cargarPosiciones(JTextArea tmpCargar)
Carga las posiciones de las casillas en el arhivo en uso.

CLASE JUGADOR

public Jugador(String nombre,byte numero_de_jugador,Casilla casillaInicial,Jugador siguiente,ImageIcon icon)
Crea un nuevo Jugador.
Parametros:
nombre: el nombre del jugador.
numero_de_jugador: el número de jugador en el orden en que se van ingresando.
casillaInicial: la casilla con que se va a inicializar el jugador (la casilla (tipo Casilla) de inicio del archivo de entrada).
siguiente: el siguiente jugador (de tipo Jugador).
icon: el ícono del jugador.

public Jugador(String nombre,byte numero_de_jugador,Casilla casillaInicial,ImageIcon icon)
Crea un nuevo Jugador con los mismos parámetros que el otro constructor pero asigna el jugador siguiente como null.

CLASE CONTROLJUGADORES

public boolean existenJugadores()
Retorna true si existe por lo menos un jugador ya ingresado a la cola, false de lo contrario.

public void borrarTodos()
Borra todos los jugadores de la cola y asigna null al espejo.

public boolean nuevoJugador(String nombre,Casilla casillaInicial,ImageIcon icon)
Ingresa un nuevo jugador a la cola. Retorna true si lo ingresó; si el jugador ya existía y no lo ingresó retorna false. Actualiza el espejo.
Parametros:
nombre: el nombre del jugador.
casillaInicial: la casilla con que se va a inicializar el jugador (la casilla (tipo Casilla) de inicio del archivo de entrada).
icon: el ícono del jugador.

public void jugadorSiguiente()
Mueve el espejo al siguiente Jugador de la cola; si el espejo está en el último Jugador, el espejo toma el valor entero de la cola para poder moverse al primer Jugador de la cola. Se usa para cambiar el turno al Jugador siguiente. Si la cola está vacía, no hace nada.

public Jugador jugadorActual()
Retorna el jugador actual. (retorna el que está apuntando el espejo).

public Jugador primerJugador()
Al invocar, se apunta a toda la cola de Jugadores. Retorna el primer Jugador.

protected boolean esRepetido(String nombre)
Retorna true si encuentra un jugador con el mismo nombre que el parámetro, si no, retorna false. (solo uso interno).

public byte cantidadJugadores()
Retorna la cantidad total de jugadores en la cola.

CLASE CASILLA

public Casilla(int col,int f,String etiq,String penal,int cant,ImageIcon icono,int nom)
Crea una nueva Casilla.
Parametros:
fila=f;
columna=col;
nombre o número de la casilla=nom;
cantidad de pasos/tiros por la penalización=cant;
penalización (dice si avanza,tira,retrocede,etc)=penal;
etiqueta de la penalización=etiq;
icono para la casilla=icono;

public Casilla(int col,int f,String etiq,String penal,int cant,Icon icono,int nom,Casilla siguiente)
Hace lo mismo que el otro método pero el parámetro icono es tipo ‘Icon’ y asigna la casilla siguiente con el parámetro ‘siguiente’.

CLASE CONTROLCASILLA

public controlCasilla()
Es el constructor; solo asigna null a los atributos.

public Casilla primera()
Retorna la primer casilla.

public boolean estaVacia()
Dice si la lista está vacía o no.

public void addCasilla(Casilla cas,ImageIcon i)
Agrega una nueva casilla a la lista.

public void borrarTodas()
Borra todas las casillas en la lista.

void crearPila(int total)
En caso que el tablero esté al revés, crea una pila para darle vuelta a las casillas.