Descripción de las clases

# 

# Descripción de las ventanas de la capa de presentación

(Nota: Las imágenes de la carpeta ‘Ventanas capa de diseño’ y las clases de este documento pueden verse sujetas a cambios de cara a la entrega final.)

**Nombre de la ventana:** *PantallaLogIn*

**Descripción de los componentes:**

* **Usuario (Campo de texto):** Nombre de usuario.
* **Contraseña (Campo de texto):** Contraseña asociada al usuario.
* **Iniciar sesión (Botón):** Intenta iniciar sesión con los campos anteriores.
* Si consigue iniciar sesión -> Navega a *PantallaInicio*.
* Si falla -> Mensaje de error y se borra el campo Contraseña , no navega a ninguna pantalla.
* **Crear usuario (Botón):** Intenta crear un usuario con los campos anteriores.
* Si consigue crear el usuario -> Muestra un mensaje de correcta creación, no navega a ninguna pantalla.
* Si falla -> Mensaje de error y se borra el campo Contraseña , no navega a ninguna pantalla.

**Nombre de la ventana:** *PantallaInicio*

**Descripción de los componentes:**

* **Jugar (Botón):** Navega a la *PantallaGestion*.
* **Historial (Botón):** Navega a la *PantallaConsulta*.
* **Clasificaciones (Botón):** Navega a la *PantallaKakuros*.
* **Salir (Botón):** Cierra la aplicación.
* **Cerrar sesión (Botón):** Navega a *PantallaLogIn*.

**Nombre de la ventana:** *PantallaConsulta*

**Descripción de los componentes:**

* **Lista de Partidas Anteriores (Lista):** Lista de las partidas anteriores del usuario.
* **Volver atrás(Botón):** Navega a *PantallaInicio.*

**Nombre de la ventana:** *PantallaKakuro*

**Descripción de los componentes:**

* **Lista de Partidas Anteriores (Lista):** Lista de los kakuros disponibles, al seleccionar uno se navega a *PantallaRankingORecord* conservando la Id del kakuro seleccionado para futuras pantallas.
* **Volver atrás(Botón):** Navega a *PantallaInicio.*

**Nombre de la ventana:** *PantallaRankingORecord*

**Descripción de los componentes:**

* **Ranking (Botón):** Navega a PantallaRanking.
* **Record Personal (Botón):** Navega a PantallaRecord.
* **Volver atrás(Botón):** Navega a *PantallaKakuro.*

**Nombre de la ventana:** *PantallaRanking*

**Descripción de los componentes:**

* **Lista de Partidas Anteriores (Lista):** Lista del ranking del kakurospreviamente seleccionado
* **Volver atrás(Botón):** Navega a *PantallaRankingORecord.*

**Nombre de la ventana:** *PantallaRecord*

**Descripción de los componentes:**

* **Lista de Partidas Anteriores (Lista):** Record del kakuro previamente seleccionado.
* **Volver atrás(Botón):** Navega a *PantallaRankingORecord.*

**Nombre de la ventana:** *PantallaGestion*

**Descripción de los componentes:**

* **Crear Kakuro (Botón):** Navega a la *PantallaCreacionKakuros*.
* **Jugar Partida (Botón):** Navega a la *PantallaSelecciónPartida*.
* **Volver atrás (Botón):** Navega a *PantallaInicio*.

**Nombre de la ventana:** *PantallaSelecciónPartida*

**Descripción de los componentes:**

* **Nueva Partida (Botón):** Navega a *PantallaNuevaPartida*.
* **Cargar Partida (Botón):** Navega a *PantallaCargarPartida*.
* **Volver atrás(Botón):** Navega a *PantallaGestion*.

**Nombre de la ventana:** *PantallaCreacionKakuros*

**Descripción de los componentes:**

* **Importar kakuro desde un fichero (Botón):** Navega a *PantallaSeleccionArchivo*.
* **Generar Kakuro (Botón):** Navega a *PantallaCreacionKakuro*.
* **Volver atrás(Botón):** Navega a *PantallaGestion*.

**Nombre de la ventana:** *PantallaSeleccionArchivo*

**Descripción de los componentes:**

Es un frame File Chooser de swing windows predefinido. En líneas generales es un navegador de archivos clásico que permite seleccionar un archivo y clicar open para seleccionarlo, o cancel para salir de la pantalla.

**Nombre de la ventana:** *PantallaNuevaPartida*

**Descripción de los componentes:**

* **Kakuros disponibles (Lista):** Lista con los kakuros disponibles para jugar.
* **Nivel de ayuda (Grupo de botones):** Selecciona el nivel de ayuda asociado a la partida.
* **Modo de juego (Grupo de botones):** Selecciona el modo de juego asociado a la partida.
* **Empezar partida (Botón):** Navega a *PantallaJugarPartida*.
* **Volver atrás (Botón):** Navega a *PantallaSelecciónPartida*.

**Nombre de la ventana:** *PantallaCargarPartida*

**Descripción de los componentes:**

* **Partidas empezadas (Lista):** Lista seleccionable de partidas empezadas por el usuario.
* **Continuar partida (Botón):** Navega a *PantallaJugarPartida*.
* **Volver atrás (Botón):** Navega a *PantallaSelecciónPartida*.

**Nombre de la ventana:** *PantallaJugarPartida*

**Descripción de los componentes:**

* **Tabla del kakuro(Grid):** Tablero de campos de texto, algunos editables algunos no que representan el kakuro jugado.
* **Rendirte(Botón):** Finaliza la partida sin puntuación, navega a *PantallaFinalización*.
* **Guardar partida (Botón):** Guarda el estado de la partida.
* **Comprobar solución (Botón):** Manda comprobar que la partida es correcta
* Si la solución es correcta -> Navega a *PantallaFinalización*.
* Si no lo es -> Mensaje de error.
* **Volver atrás (Botón):** Navega a *PantallaSelecciónPartida*.

**Nombre de la ventana:** *PantallaPartidaAcabada*

**Descripción de los componentes:**

* **Puntuacion (Label):** Muestra la puntuacion obtenida en la partida acabada.
* **Volver atrás (Botón):** Navega a *PantallaGestion*.

# Descripción de las clases de la capa de dominio

## Clases

**Nombre de la clase:** *Usuario*

**Breve descripción de las clases:** Persona que puede jugar a kakuros.

**Cardinalidad:** Número de usuarios registrados.

**Descripción de los atributos:**

**Nombre:** Nombre único del usuario (no static).

**Contraseña:** Contraseña del usuario (no static).

**Descripción de los métodos:**

**void comparePassword(string pwd):** compara el password introducido con el del usuario.

**Descripción de las relaciones:**

* Relación de asociación con la clase “Partida”: indica las partidas empezadas por un usuario.
* Relación de dependencia con el controlador de usuarios.

**Nombre de la clase:** *Kakuro*

**Breve descripción de las clases:** Puzzle de suma en cruz que pueden jugar los usuarios.

**Cardinalidad:** Número de kakuros almacenados.

**Descripción de los atributos:**

**ID:** Id única del kakuro.

**Casillas Negras:** Lista de casillas negras (con suma) del kakuro.

**Descripción de los métodos:**

**boolean esNegra(int i, int j):** Retorna true si la casilla en la posición (i,j) del tablero es una Casilla negra, false en caso contrario.

**Matrix<String> toArrayList():** Convierte un miembro de la clase kakuro (con la solución) en el formato de Matriz de String para pasarlo entre capas y almacenarlo.

**Matrix<String> toArrayListPartida():** Convierte un miembro de la clase kakuro (con los números introducidos durante una partida) en el formato de Matriz de String para pasarlo entre capas y almacenarlo.

**Descripción de las relaciones:**

* Relación de asociación con la clase “Partida”: indica qué partidas se están jugando con ese kakuro.
* Relación de composición con la clase “Casilla”: indica que casillas contiene un kakuro.
* Relación de Uso con las clases “Algoritmo Creación” y “Algoritmo de Resolución”: Ambas clases Utilizan la clase Kakuro para Resolver y Generar Kakuros.
* Relación de dependencia con el controlador de kakuros.**Nombre de la clase:** *Casilla*

**Nombre de la clase:** *Casilla*

**Breve descripción de las clases:** Clase abstracta. Cada una de las celdas de un kakuro, ya sean blancas o negras.

**Cardinalidad:** Número de casillas de todos los kakuros.

**Descripción de los atributos:**

**Pos:** posición de la casilla en el kakuro

**Descripción de las relaciones:**

* Relación de composición con la clase “Kakuro”: indica por qué casillas está compuesto este kakuro.
* Relaciones de herencia con “Casilla Negra” o “Casilla Blanca”: indica para una casilla dada indica si esta es blanca o negra.

**Nombre de la clase:** *Casilla Blanca*

**Breve descripción de las clases:** Celdas blancas de un kakuro, aquellas que pueden ser editadas con números del 1 al 9.

**Cardinalidad:** Número de casillas blancas de todos los kakuros.

**Descripción de los atributos:**

**NúmeroElegido:** Candidato (número entre 1 y 9) colocado en la casilla blanca o predeterminado (-1) si no hay número colocado.

**NúmeroSolución:** Candidato (número entre 1 y 9) determinado por el algoritmo de resolución.

**Modificable:** Booleano que indica si una casilla blanca es modificable, i.e. es una casilla dada por la ayuda, o no.

**casillaNegraVerticalCoord:** posición en el kakuro de la casilla negra que encabeza su tramo vertical.

**casillaNegraHorizontalCoord:** posición en el kakuro de la casilla negra que encabeza su tramo horizontal.

en el kakuro.

**Posibles:** Candidatos (números entre 1 y 9) posibles para esa casilla blanca (utilizado por el algoritmo de resolución).

**Descripción de las relaciones:**

* Relación de herencia con “Casilla”.

**Nombre de la clase:** *Casilla Negra*

**Breve descripción de las clases:** Celdas no editables de un kakuro, que pueden contener información sobre la sumas de los tramos del puzle.

**Cardinalidad:** Número de casillas negras de todos los kakuros.

**Descripción de los atributos:**

**SumaTotalVertical:** En lo que debe resultar la suma de las casillas blancas en la columna de la casilla negra (empezando por la casilla inferior a la casilla negra) hasta encontrar otra casilla negra o el “final” del kakuro.

**SumaTotalHorizontal:** En lo que debe resultar la suma de casillas blancas en la fila de la casilla negra (empezando por la casilla a la derecha de la casilla negra) hasta encontrar otra casilla negra o el “final” del kakuro.

**NumCellsVertical:** Número de casillas blancas verticales que forman el tramo indicado por la casilla negra.

**NumCellsHorizontal:** Número de casillas blancas horizontales que forman el tramo indicado por la casilla negra.

**secuenciasCandidatasVertical:** Lista de las secuencias posibles del tramo vertical que define esa casilla negra. Depende del tamaño del tramo y de la suma.

**secuenciasCandidatasHorizontal:** Lista de las secuencias posibles del tramo horizontal que define esa casilla negra. Depende del tamaño del tramo y de la suma.

**candidatosVertical:** Lista de los posibles candidatos (intersección) de las casillas blancas del tramo vertical.

**candidatosHorizontal:** Lista de los posibles candidatos (intersección) de las casillas blancas del tramo horizontal.

**convergenVertical:** Lista de números ya hallados en la resolución del tramo vertical.

**convergenHorizontal:** Lista de números ya hallados en la resolución del tramo horizontal.

**Descripción de las relaciones:**

* Relación de herencia con “Casilla”.

**Nombre de la clase:** *Modo*

**Breve descripción de las clases:** Clase abstracta. Modalidades de resolución de kakuros, con sus respectivas penalizaciones.

**Cardinalidad:** 2.

**Descripción de los atributos:**

**Modo:** indica el modo de juego (hasta el momento normal o ranked).

**Penalización:** indica la penalización de ese modo de juego.

**Descripción de las relaciones:**

* Relación de asociación con “Partida”: indica el modo de juego que va a tener esa partida.
* Relación de herencia con las subclases (instancias) “ModoRanking”, “ModoNormal”.

**Nombre de la clase:** *Partida*

**Breve descripción de las clases:** La elección de un Usuario sobre un Kakuro y modo y en una hora de inicio.

**Cardinalidad:** Tantas como Partidas se hayan empezado.

**Descripción de los atributos:**

**Id:** identificador de la partida

**Tiempo inicio sesión:** Instante en el que se reanuda/inicia esa partida.

**Tiempo finalización:** Instante en el que se finalizó la partida, si ha terminado.

**Tiempo transcurrido:** Tiempo total que ha estado esa partida siendo jugada.

**Puntuación:** Puntuación calculada en función del tiempo transcurrido y el nivel de ayuda de una partida.

**Hora inicio sesión:** hora en la que se reanuda la resolución de la partida.

**Hora fin sesión:** hora en la que el usuario finaliza una sesión de resolución de la partida (sin finalizar el kakuro).

**Descripción de los métodos:**

**void acabarSesion():** guarda el estado de la partida actual.

**void IniciarSesion():** carga la partida y actualiza los atributos correspondientes.

**void acabarPartida(boolean rendido):** finaliza la partida, en caso de ser una rendición se tiene en consideración.

**Matrix<String> toArrayList():** Convierte un miembro de la clase partida en el formato de Matriz de String para pasarlo entre capas y almacenarlo.

**void establecerPuntuacion** y **double formulaPuntuacion(modo m, nivelAyuda nV):** al terminar una partida, le asigna según corresponda la puntuación siguiendo la fórmula indicada.

**Descripción de las relaciones:**

* Relación de asociación con “Usuario”: indica el usuario propietario de esa partida.
* Relación de asociación con “Modo”: indica el tipo de juego y las normas que se jugarán en la partida.
* Relación de dependencia con el controlador de partidas.

**Nombre de la clase:** NivelAyuda

**Breve descripción de las clases:** Nivel de ayuda asociado a la partida con su respectiva penalización.

**Cardinalidad:** Cuatro.

**Descripción de los atributos:**

**TipoAyuda:** string con el nombre del tipo de ayuda que se asociará a la partida

**Penalización:** indica la penalización asociada al tipo de ayuda.

**Descripción de las relaciones:**

* Relación de asociación con la clase “partida”: indica el nivel de ayuda asociado a esa partida.
* Relación de herencia con las subclases (instancias) NivelAyudaNada, NivelAyudaPoco, NivelAyudaMedio, NivelAyudaMucho.

## Controlador de dominio y controladores de clase

Estas clases están relacionadas con la gestión de la clase de dominio, y su comunicación con el resto de capas.

### Controlador de dominio

Es el encargado de llamar a los controladores individuales de cada una de las clases principales de la capa de dominio, además de intercambiar la información con la capa de persistencia y la capa presentación.

**Atributos**

* cuentaActual: cuenta del usuario que tiene la sesión iniciada. Su id es la que se usa para crear, cargar y guardar las partidas.

**Cardinalidad:** 1

**Métodos**

Funcionalidades que el controlador de la capa de dominio delega a los controladores individuales de clase, para que se comuniquen con la capa de persistencia, si es preciso.

Métodos de usuario

* **bool** comprobarUsuario(**String** user, **String** password): comprueba que las credenciales introducidas por un usuario se encuentren almacenadas en el mapa de usuarios o en la capa de persistencia. De ser así, retorna verdadero; falso en el caso contrario.
* **bool** crearUsuario(**String** user, **String** password): crea un nuevo usuario con nombre *user* y contraseña *password* en el mapa del gestor de usuarios y en la capa de persistencia. Retorna verdadero si se ha dado de alta correctamente.

Métodos de kakuro

* **int** guardarKakuro(**Kakuro** k): guarda la información del kakuro k en el mapa del controlador de kakuros y en la capa de persistencia. Retorna la id del nuevo kakuro.
* **Matrix<String>** leerKakuro(**int** id): busca el kakuro con el id dado en el mapa de kakuros del controlador y lo retorna en formato de matriz de String. Si no lo encuentra lo busca en la capa de persistencia. Si tampoco se halla allí, escribe por consola que el kakuro con el id dado no existe, y devuelve una matriz de string vacía.
* **Matrix<String>** leerPreviewKakuros(): lee de la capa de persistencia el fichero con el preview de los kakuros y lo retorna en formato matriz de string.
* **Matrix<String>** crearKakuro(**Matrix<String>** kakuro): crea un nuevo kakuro con la información pasada por parámetro. Lo guarda en el mapa del gestor de kakuros y en la capa de persistencia. Retorna la matriz con la información creada.

Métodos de partida

* **int** guardarPartida(**Partida** p): guarda la partida p en el mapa del controlador de partidas y la envía a la capa de persistencia.
* **Matrix<String>** leerPartida(**int** id): busca la partida con el id dado en el mapa de partidas del controlador y lo retorna en formato de matriz de string. Si no lo encuentra busca en la capa de persistencia. Si tampoco se halla allí, escribe por consola no existe una partida con ese id dado, y devuelve una matriz de string vacía.
* **Matrix<String>** leerPreviewPartida(): lee de la capa de persistencia el fichero con el preview de las partidas del usuario actual y lo retorna en formato matriz de string.
* **Matrix<String>** leerHistorialPartidas(): lee de la capa de persistencia el fichero con el historial de las partidas del usuario actual y lo retorna en formato matriz de string.
* **Matrix<String>** crearPartida(**Matrix<String>** partida): crea una nueva partida enviando la información recibida al controlador de persistencia y además lo guarda en el atributo map. Retorna la matriz con la información creada.

Métodos de Ranking

* **bool** guardarRanking(**Ranking** r): guarda la información del ranking r en el mapa del gestor de rankings y en la capa de persistencia. Retorna la verdadero si se ha guardado correctamente.
* **Matrix<String>** leerRanking(**int** id): busca el ranking del kakuro con el id dado en el mapa de rankings del controlador y lo retorna en formato de matriz de string. Si no lo encuentra lo busca en la capa de persistencia. Si tampoco se halla allí, escribe por consola que el kakuro con el id dado no tiene partidas, y devuelve una matriz de string vacía.

Métodos de Record

* **bool** guardarRecord(**partida** p): guarda la partida como record si esta tiene mejor puntuación que la actualmente guardada para ese usuario y kakuro. Retorna la verdadero si se ha guardado correctamente.
* **Matrix<String>** leerRecord(**int** id): busca el record del kakuro con id dado para el usuario actual en el mapa de records del controlador y lo retorna en formato de matriz de string. Si no lo encuentra lo busca en la capa de persistencia. Si tampoco se halla allí, escribe por consola que el kakuro con el id dado no tiene partidas, y devuelve una matriz de string vacía.

# 

### Controladores de clase

(Nota: en este apartado se explicitan los atributos y clases de manera genérica, puesto que los cinco controladores son muy similares y, por lo tanto, resulta más cómodo y breve resumir la información conjuntamente de manera genérica.)

Son los responsables en los que el controlador de dominio delega la interacción con las clases de manera individual.

**Atributos**

repoControlador: mapa que actúa de caché para la clase del controlador. Se inicializa al crear la instancia del controlador.

**Métodos**

En este caso, **clase** puede referirse tanto a **Kakuro** (k), **Usuario** (u), **Partida** (p), **Record** (re), o **Ranking** (ra). No todas las clases tienen porque tener todos los métodos implementados. Aún así,

* **int/bool** guardarClase(**clase** c): envía a la capa de persistencia los datos de la clase c. En algunos casos retorna la id de la clase si se ha realizado correctamente, en otros retorna verdadero. [k,p]
* **bool** actualizarClase(**clase** c, **partida** p): actualiza la instancia de la clase c en la capa de persistencia. Si se trata de un ranking, puede añadirse la partida p en su posición el ranking. Si se trata de un récord, comprueba si la partida p tiene mejor puntuación que el récord. [ra,re]
* **Matrix<String>** leerClase(**int** id): lee de la capa de persistencia los datos de una instancia concreta de la clase c con la clave externa id. En el caso de kakuro, también se permite leer de ficheros de texto. [k,p,ra,re]
* **Matrix<String>** leerPreviewClase([**string** u]): lee de la capa de persistencia los datos de todas las instancias deseadas de la clase c, con tal de obtener un catálogo que mostrar en la capa de presentación. En el caso de partidas, requiere del nombre del usuario para presentar sólo sus partidas. [k,p]
* **Matrix<String>** leerHistorial([**string** u]): solo disponible para la clase partida. Lee de la capa de persistencia los datos de todas las instancias partidas de un usuario u, tanto acabadas como actuales, con tal de obtener un catálogo que mostrar en la capa de presentación.
* **int/bool** crearClase(): crea una nueva instancia a la capa de persistencia los datos de la clase c. En el caso de kakuro retorna su id, en el de usuario, verdadero si se ha realizado correctamente. [k,u]
* **clase** fromArrayList(**Matrix<String>** c): convierte la matriz de string a una instancia de la clase. [k,p,ra,re]

En algunas clases, hay algunas funciones adicionales para facilitar y hacer más legible la traducción de la matriz de string a la clase kakuro.

# Capa de persistencia

## Controlador de persistencia

Es el encargado de recibir peticiones de la capa de dominio, tanto de lectura como de escritura y delegarlas a los gestores de datos, también crea el directorio datos donde se almacenarán todos las instancias.

**Atributos**

* Path: Indica la ruta donde se crearán los directorios para almacenar los datos
* nombreCarpeta: nombre del directorio donde se almacenan los datos

**Cardinalidad: 1**

**Métodos**

Funcionalidades que implementa el controlador de la capa de persistencia llamando a los gestores de datos.

Métodos de Lectura de Clases:

* **Matrix<String>** leerInfoKakuro(**String** Id): devuelve el kakuro con el Id dado.
* **Matrix<String>** leerInfoPartida(**String** Id, **String** usuario): devuelve la información de la partida con el Id dado que tiene como jugador al usuario dado.
* **Matrix<String>** leerInfoRanking(**String** Id): devuelve las partidas con mayor puntuación asociadas al kakuro con el Id dado.
* **Matrix<String>** leerInfoRecord(**String** Id, **String** usuario): devuelve las partidas con mayor puntuación asociadas al kakuro con el Id dado y con el usuario dado.
* **Matrix<String>** leerInfoUsuario(): devuelve todos los usuarios existentes.

Métodos de Lectura de Previews:

* **Matrix**<String> leerPreviewKakuros(): devuelve los ids y la información relevante de todos los kakuros.
* **Matrix**<String> leerPreviewPartidas(**String** usuario): devuelve los ids y la información relevante de todas las partidas actuales asociadas al usuario dado.
* **Matrix**<String> leerHistorialPartidas(**String** usuario): devuelve los ids y la información relevante de todas las partidas asociadas al usuario dado, tanto las actuales como las ya finalizadas.

Métodos de Escritura:

* **int** guardarInfoKakuro(**Matrix<String>** kakuro,**Matrix<String>** info): guarda el kakuro en un fichero y además guarda la info en el archivo de preview. En el caso de haberse guardado retorna la Id que tendrá ese kakuro, en caso de no existir retorna -1.
* **int** guardarInfoPartida (**String** usuario, **Matrix<String>** info): guarda la partida asociada al usuario dado y además guarda la info en el archivo preview. En el caso de haberse guardado retorna la Id que tendrá esa partida, en caso de no existir retorna -1.
* **void** actualizarInfoRanking(**String** Id, **Matrix<String>** info): actualiza el conjunto de partidas ordenadas asociadas al kakuro con el Id dado, añadiendo la nueva partida pasada por parámetro si es preciso.
* **void** actualizarInfoRecord(**String** Id, **String** usuario, **Matrix<String>** info): actualiza la mejor partida asociada al kakuro con el Id y usuario dados con una partida pasada por parámetro, si es preciso.
* **boolean** guardarInfoUsuario(**Vector<String>** info): guarda a los usuarios con sus respectivas contraseñas encriptadas encontradas en info.

## 

## Gestores de datos

(Nota: similarmente que con los controladores de clase de la capa de dominio, en este apartado se explicitan los atributos y clases de los gestores de manera genérica.)

**Atributos**

Atributos de los distintos gestores. De cara a la última entrega, es posible que algunos de estos pasen a estar “hard-coded” en lugar de almacenados en un fichero.

**String** path: path del gestor en el sistema.

**String** nombreFichero y nombreCarpeta: nombre de los ficheros y carpetas. Concatenadas en el path para modificar o acceder ficheros.

**String** ficheroInfo: nombre del fichero que contiene las previews de los repositorios de kakuro y partida.

**String** ficheroEstado: nombre del fichero que contiene la id que se usará para el siguiente kakuro o partida.

**Integer** ID: id del kakuro o partida.

**Métodos**

**void** crearCarpeta() y **void** crearFichero(): crea la carpeta o fichero con el path dado si estos no existen.

**int** siguienteIDKakuro/Partida(): devuelve la id del siguiente elemento de la clase kakuro o partida.

**void** guardarEstadoGestor(): guarda el estado del gestor en un fichero para reinicializarlo en su próxima llamada.

**int** guardarClase(**Matriz<String>** c): guarda toda la información de un elemento de la clase.

**void** guardarPreviewClase(**Matriz<String>** c): guarda los datos de c en el fichero preview del gestor.

**Matriz<String>** leerClase(**String** id): retorna el fichero id relacionado con la clase. En el caso del usuario se retorna todos los usuarios del sistema.

**Matriz<String>** leerPreviewClase(**String** u): lee del fichero preview los datos de todas las instancias deseadas de la clases kakuro o partida.