Índice

1. Descripción del diagrama de clases

1.1. IPartida y FactoriaPartida

- Nombre de la classe:
  - Partida y FactoriaPartida
- Breve descripció de la classe:
  - Clases responsables de desacoplar las variantes de partida del resto del sistema mediante el patrón Simple Factory. IPartida es una interfaz implementada por PartidaCodemaker y PartidaCodebreaker, en cambio, FactoriaPartida es un singleton.
- Cardinalidad:
  - IPartida: Uno para cad partida.
  - FactoriaPartida: Singleton, uno único.
- Descripción de los atributos:
  - *IPartida sin atributos*
  - *FactoriaPartida (instancia Singleton e instancia IPartida)*
- Descripción de las relaciones:
  - Relación de implementacion con las variantes de Partida "PartidaCodemaker/PartidaCodebreaker": Indica que las variantes de "Partida" implementan esta interficie.
- Descripción de los métodos:
  - (FactoriaPartida) creaPartida(int, List<Object>, Perfil):: IPartida: Método creador de partidas que dependiendo de la configuración establecido en el segundo parámetro, se crea una PartidaCodemaker o Codebreaker con el identificador específico del primer parámetro y se asocia al perfil indica en en el tercer parámetro.
  - (IPartida) ejecutarTurno(List<Integer>):: Integer: Método abstracto implementado por las variantes de Partida encargado de ejecutar el turno.
  - (IPartida) calculaPuntuacion():: Integer: Método abstracto implementado por las variantes de Partida responsable de calcular y devolver la puntuación de la partida en cuestión.

1.2. Partida

- Nombre de la classe:
  - Partida
- Breve descripció de la classe:
  - Clase central del diagrama, creada y configurada por el usuario, y contenedora de su tablero del juego.
- Cardinalidad:
  - Una para cada partida creada por el usuario.
- Descripción de los atributos:
  - id: Identificador de la partida.
  - acabada: Booleano indicador del estado de la partida (finalizada : en curso).
  - codigoSecreto: Lista de colores correspondientes al codigo secreto de la partida.
  - puntuacion: Variable almacenadora de la puntuación de la partida.
  - configuracion: Objeto de tipo “Configuracion” correspondiente a la configuración de la partida.
  - tablero: Objeto de tipo “Tablero” correspondiente al tablero de la partida.
  - perfil: Objeto de tipo “Perfil” correspondiente al perfil de la partida.
  - pistasDisponibles: Entero que contiene el numero de pistas disponibles:
  - ganada: Booleano que contiene el estado de victoria de la partida.
- Descripción de las relaciones:
  - Relación de asociación con la clase “Configuración”: Indica la configuración aplicada a la partida.
  - Relación de asociación con la clase “Tablero”: Indica qué tablero contiene la información de la partida, correspondiente al tablero físico.
  - Relacion de asociación con la clase “Perfil”: Indica el perfil asociado a la partida.
- Descripción de los métodos:
  - Partida(int,List<Object, Perfil):: Partida: Constructor de partida que precisa de tres parámetros: (Identificador de partida (“id”) : Lista contenedora de los atributos de configuración de la partida y el perfil asociado a la partida)
  - generaResultado(List<Integer>):: List<Integer>: Método para obtener el resultado correspondiente a la adivinanza (parámetro List<Integer>) comparada con el código secreto de la partida.

    - compruebaValidezFila(List<Integer>):: void: Método que comprueba la validez de la longitud de la fila introducida respecto a la configuración de la partida.
    - avanzarTurno():: void: Método que incrementa en uno el turno actual (variable de “Tablero”), mediante su función incrementado (función de “Tablero”).
    - calculaPuntuacion():: double: Método abstracto encargado de calcular la puntuacion de partida dependiendo de la variante de esta.
    - ejecutarTurno(List<Integer>):: int: Método abstracto encargado de ejecutar un turno y devolver el estado de la partida como int {0: en curso, -1, perdida, 1 ganada), dependiendo de la variante de partida.
    - obtenPista():: List<Integer>: Método abstracto encargado de obtener la pista correspondiente a la variante de partida.

1.3. PartidaCodemaker

- Nombre de la classe:
    - PartidaCodemaker
- Breve descripció de la classe:
    - Clase que hereda de la clase “Partida”, y que representa la variación de partida correspondiente al rol de Codemaker.
- Cardinalidad:
    - Una para cada partida creada por el usuario en que este juegue como Codemaker.
- Descripción de los atributos:
    - maquina (Maquina): Objeto de tipo “Maquina” con el que se ejecutarán los algorismos implementados (FiveGuess : Genetics).
- Descripción de las relaciones:
    - Relación de generalización con la clase “Partida”: Indica que es una variante de partida.
    - Relación de asociación con la clase “Maquina”: Indica que por cada PartidaCodemaker, se le atribuye una Maquina para escoger el algoritmo y configuración de este a ejecutar.
    - Relación de implementación con la clase “IPartida”: Indica que implementa la interfaz IPartida y por tanto sus métodos definidos.
- Descripción de los métodos:
    - PartidaCodemaker(int,List<Object, Perfil):: PartidaCodemaker: Constructor de partida que precisa de tres parámetros: (Identificador de partida (“id”) : Lista contenedora de los

atributos de configuración de la partida), con el añadido de configurar (configuraMaquina() la “Maquina” perteneciente a esta partidaCodemaker y por último el perfil asociado a la partida.
    - configuraMaquina():: void: Método que configura la máquina de acuerdo a la configuración de la partida.
    - calculaPuntuacion():: Integer: Método que calcula y devuelve la puntuación de la partida correspondiente al usuario como Codemaker.
    - ejecutarTurno(List<Integer>):: Integer: Método que realiza la ejecución del turno del tipo partidaCodemaker, requiriendo en caso de justo haber iniciado la partida, el código secreto como parámetro, de lo contrario, se lanza una excepción. Seguidamente, obtiene la adivinanza realizada por la máquina, al igual que su resultado (obtenResultado()), ahora bien, este resultado debe ser informado por el usuario; informa ambos al tablero, avanza el turno (avanzarTurno()) y comprueba si la partida debe acabar, desembocando en victoria, derrota o continuación.
    - sincronizarConfiguracionGenetic():: void: Método encargado de sincronizar la configuración de la partida con la de la maquina.
    - obtenPista():: List<Integer>: Método encargado de obtener la pista correspondiente al rol codemaker.

1.4. PartidaCodebreaker

- Nombre de la classe:
    - PartidaCodebreaker
- Breve descripció de la classe:
    - Clase que hereda de la clase “Partida”, y que representa la variación de partida correspondiente al rol de Codebreaker.
- Cardinalidad:
    - Una para cada partida creada por el usuario en que este juegue como Codebreaker.
- Descripción de los atributos:
    - codigoRevelador (Integer[]) : Array de integers que contiene el codigo revelado hasta el momento.
    - posicionesReveladas (Integer[]): Array de integers que contienen que posiciones se han revelado.
    - numeroPosicionesReveladas (Integer): Entero que contiene el numero de posiciones reveladas hasta el momento.
- Descripción de las relaciones:

    - Relación de generalización con la clase “Partida”: Indica que es una variante de partida.
    - Relación de implementación con la clase “IPartida”: Indica que implementa la interfaz IPartida y por tanto sus métodos definidos.
- Descripción de los métodos:
    - PartidaCodebreaker(int,List<Object, Perfil):: PartidaCodebreaker: Constructor de partida que precisa de tres parámetros: (Identificador de partida (“id”) : Lista contenedora de los atributos de configuración de la partida y el perfil asociado a la partida), con el añadido de generar el código secreto (generarCodigoSecreto()) e informando a la “Partida”.
    - calculaPuntuacion():: Integer: Método que calcula y devuelve la puntuación de la partida correspondiente al usuario como Codebreaker.
    - ejecutarTurno(List<Integer>):: Integer: Método que realiza la ejecución del turno del tipo partidaCodebreaker, requiriendo en cada turno, la adivinanza como parámetro, de lo contrario, se lanza una excepción. Seguidamente, se obtiene el resultado (obtenResultado()), informa ambos al tablero, avanza el turno (avanzarTurno()) y comprueba si la partida debe acabar, desembocando en victoria, derrota o continuación.
    - generarCodigoSecreto(int, int):: List<Integer>: Método que genera y devuelve un código al azar correspondiente a la configuración de la partida necesaria (int: longitud del código, int: número de colores diferentes).
    - obtenPista():: List<Integer>: Método encargado de obtener la pista correspondiente al rol de codebreaker.

1.5. Tablero

- Nombre de la classe:
    - Tablero
- Breve descripció de la classe:
    - Clase contenedora del tablero de adivinanzas y resultados, que representa el tablero físico.
- Cardinalidad:
    - Uno para cada partida creada por el usuario.

- Descripción de los atributos:
    - numeroTurnoActual: Número de turno actual de la partida jugada en ese tablero.
    - tablero: Matriz correspondiente al tablero de adivinanzas.
    - tableroResultados: Matriz correspondiente al tablero donde se informan los resultados relativos a las adivinanzas.
- Descripción de las relaciones:
    - Relación de asociación con la clase “Partida”: Permite que cada “Partida” posea su tablero.
- Descripción de los métodos:
    - Tablero(int, int): Constructor de tablero e inicializador, tanto del tablero de adivinanzas como el de resultados.
    - informaAdivinanza(List<Integer>): Informa en “tablero”, en la posición del turno actual, la adivinanza pasada como parámetro.
    - informaResultado(List<Integer>): Informa en “tableroResultados”, en la posición del turno actual, el resultado pasado como parámetro.
    - avanzarTurno(): Incrementa en uno el atributo numeroTurnoActual.
    - obtenAdivinanza(List<Integer>): Obtiene la adivinanza de “tablero”, en la posición del turno actual.
    - obtenResultado(List<Integer>): Obtiene el resultado de “tableroResultados”, en la posición del turno actual.

1.6. Configuración

- Nombre de la classe:
    - Configuracion
- Breve descripció de la classe:
    - Clase contenedora de la configuración de las partidas.
- Cardinalidad:
    - Una para cada partida creada por el usuario.
- Descripción de los atributos:

    - numeroTurnos (Integer): Número de turnos máximos de la partida.
    - dificultad (Integer): Nivel de dificultad de la partida.
    - nivelAyuda (Integer): Nivel de ayuda al usuario en la partida.
    - numeroColores (Integer): Número de colores diferentes posibles en la partida.
    - longitudCodigo (Integer): Longitud de los codigos de la partida.
    - esCodemaker (Boolean): Indica si el jugador es Codemaker o de lo contrario, Codebreaker.
- Descripción de las relaciones:
    - Relación de asociación con la clase “Partida”: Permite que cada “Partida” posea su configuración.
- Descripción de los métodos:
    - Configuracion(List<Object>):: Configuración: Constructor mediante un parámetro lista que contiene los valores de la configuración.

1.7. Máquina y derivadas

- Nombre de las clases:
    - Máquina, MáquinaGenetic
- Breve descripció de la classe:
    - Clases contenedoras del algoritmo Genetic junto con la interfaz de este “Máquina”.
- Cardinalidad:
    - Uno por cada “PartidaCodemaker”.
- Descripción de los atributos:

  – *sin atributos* –
- Descripción de las relaciones:
    - Relación de Máquina de asociación con la clase “PartidaCodemaker”: Indica que cada partida en la cual el jugador interactúe como Codemaker “PartidaCodemaker” tiene su máquina de algoritmos asignada.

- Descripción de los métodos:
    - solve(List<Integer>):: List<List<Integer>>: Método definido por la interfaz e implementado por la máquina genetic. Tomando el código secreto como parámetro en forma de lista de enteros, se encarga de la ejecución del algoritmo escogido, y devuelve una lista de códigos correspondientes a la lista de mejores códigos,

uno por generación, conteniendo un máximo de maxSteps códigos.
    - MáquinaGenetic(int, int, int):: MáquinaGenetic: Constructor de MáquinaGenetic en cuyos parámetros corresponden al número de colores diferentes posibles, a los maxSteps o número de generaciones máximas y a la longitud de los códigos, correspondiente a la partida donde se ejecuta el algoritmo.

1.8. Color

- Nombre de la classe:
    - Color
- Breve descripció de la classe:
    - Clase representativa de los colores que contienen los códigos (usada en el entorno del algoritmo Genetic).
- Cardinalidad:
    - Tantos como la longitud de los códigos tratados en el algoritmo, para cada código.
- Descripción de los atributos:
    - valor (Object): Atributo en forma Object, que contiene el número/valor del color en cuestión.
    - valorMaximoColor(int): Atributo correspondiente al número máximo de colores diferentes que puede contener un código.
- Descripción de las relaciones:
    - Relación de asociación con la clase “Código”: Permite dar forma a la relación que hace de un color parte de un código compuesto por múltiples colores.
- Descripción de los métodos:
    - colorRandom():: Integer: Método que devuelve un entero al azar, generado entre [0,maxColorValue).

1.9. Código

- Nombre de la classe:
    - Código
- Breve descripció de la classe:
    - Clase representativa de los códigos que contienen las poblaciones de códigos (usada en el entorno del algoritmo Genetic).
- Cardinalidad:

    - Tantos como la longitud de las poblaciones de códigos, más auxiliares usados.
- Descripción de los atributos:
    - longitudCodigo (Integer): Atributo en forma Integer, que contiene la longitud que debe tener el código.
    - fitness (Double): Atributo que favorece el rendimiento del algoritmo, y contenedor del fitness del código (parámetro del algoritmo Genetic).
    - colores (Vector<Color>): Vector de colores que contiene los colores que forman el código.
- Descripción de las relaciones:
    - Relación de asociación con la clase “PoblaciónCodigos”: Permite dar forma a la relación que hace de un código parte de una población de códigos.
    - Relación de asociación con la clase “CalculoFitness”: Indica que la clase CalculoFitness contiene un código perteneciente, en este caso, al código secreto.
- Descripción de los métodos:
    - Código():: Código: Constructor de Código que además inicializa el vector de colores.
    - setColorRandom(int):: void: Método encargado de insertar un color generado al azar en la posición, indicada por el parámetro, en el vector de colores.
    - inicializaCódigo():: void: Método responsable de inicializar el código con valores al azar.
    - fitnessCódigo():: double: Método responsable de calcular en caso que este valga 0, el atributo fitness del código.

1.10. PoblaciónCodigos

- Nombre de la classe:
    - PoblaciónCodigos
- Breve descripció de la classe:
    - Clase representativa de las poblaciones de códigos (usada en el entorno del algoritmo Genetic).
- Cardinalidad:
    - Tantos como generaciones y/o steps evolucione una población, además de auxiliares.

- Descripción de los atributos:
    - códigos (Código[]): Array de objetos de tipo Código, que contiene los códigos de la población.
- Descripción de las relaciones:
    - Relación de asociación con la clase “Código”: Permite dar forma a la relación que hace de una población de códigos el conjunto de códigos.
- Descripción de los métodos:
    - PoblaciónCodigos(int, boolean):: PoblaciónCodigos: Constructor de PoblaciónCodigos que atribuye el tamaño de la población al primer parámetro y la decisión de inicializarla como segundo.
    - mejorCodigo ():: Código: Método encargado de devolver el código más fit/apto de la población en cuestión.

1.11. CalculoFitness

- Nombre de la classe:
    - CalculoFitness
- Breve descripció de la classe:
    - Clase responsable de llevar a cabo el calculo del fitness/aptitud de los códigos respecto el código secreto (usada en el entorno del algoritmo Genetic).
- Cardinalidad:
    - Una única instància, clase estática.
- Descripción de los atributos:
    - codigoSecretoGenetic (Código): Atributo de tipo Código que contiene el código secreto y/o solución con la que comparar los códigos recibidos.
- Descripción de las relaciones:
    - Relación de asociación con la clase “Código”: Indica que la clase CalculoFitness contiene un código perteneciente, en este caso, al código secreto.
- Descripción de los métodos:
    - maxFitness():: double: Método que devuelve el valor máximo posible de fitness obtenible por los códigos.
    - fitnessCodigo(Código):: double: Método encargado de calcular y devolver el fitness/aptitud del código.
    - obtenResultado(Código):: List<Integer>: Método responsable de obtener el resultado correspondiente al código recibido como parámetro comparado al código secreto (atributo de CalculoFitness).

1.12. Evolución

- Nombre de la classe:
  - Evolución
- Breve descripció de la classe:
  - Clase responsable de llevar a cabo la evolución de los códigos (usada en el entorno del algoritmo Genetic).
- Cardinalidad:
  - Uno por cada MáquinaGenetic.
- Descripción de los atributos:
  - numTorneos (Integer): Atributo que contiene el parámetro a tener en cuenta en el proceso de selección.
  - umbralRecombinación (double): Atributo correspondiente al umbral usado en el proceso de recombinación.
  - tasaMutación (double): Atributo correspondiente a la tasa de mutación usada en el proceso de mutación.
  - umbralPermutación (double): Atributo correspondiente al umbral usado en el proceso de permutación.
  - umbralInversión (double): Atributo correspondiente al umbral usado en el proceso de inversión.
- Descripción de las relaciones:

  – *sin relaciones* –
- Descripción de los métodos:
  - evolucionaPoblacionCodigos(PoblaciónCódigos):: PoblaciónCodigos: Método encargado de causar la evolución de la población de códigos parámetro, siguiendo la ruta Selección, Recombinación, Mutación, Permutación y Inversión.
  - seleccionaCódigo(PoblaciónCódigos, int):: Código: Método encargado del proceso de selección de la población indicada como parámetro y con el parámetro del proceso numTorneos como segundo parámetro del método.
  - recombinaCódigos(Código,Código,double):: Código: Método encargado del proceso recombinación de los dos códigos parametrizados influenciada por el umbral de recombinación correspondiente al tercer parámetro.
  - mutaCódigo(Código, double):: Código: Método encargado del proceso de mutación del código parámetro influenciada por la tasa de mutación correspondiente al segundo parámetro.
  - permutaCódigo(Código, double):: Código: Método encargado del proceso de permutación del código parámetro influenciada por el umbral de permutación correspondiente al segundo parámetro.

- invierteCódigo(Código, double):: Código: Método encargado del proceso de inversión del código parámetro influenciada por el umbral de inversión correspondiente al segundo parámetro.

1.13. ConfiguracionGenetic

- Nombre de la classe:
  - ConfiguracionGenetic
- Breve descripció de la classe:
  - Clase responsable de almacenar la configuración usada en el algoritmo Genetic (usada en el entorno del algoritmo Genetic).
- Cardinalidad:
  - Una única instància, clase estática.
- Descripción de los atributos:
  - númeroColores (Integer): Atributo correspondiente al número de colores diferentes de la configuración de la partida.
  - longitudCódigo (Integer): Atributo correspondiente a la longitud de los códigos de la partida.
  - maxGeneraciones (Integer): Atributo almacenador del número máximo de generaciones a generar en la evolución y/o llamados “maxSteps”.
  - númeroCódigosPoblación (Integer): Atributo que corresponde al número de códigos que deberán almacenar las poblaciones de códigos.
  - dificultad (Integer): Atributo que contiene la dificultad de la partida que influencia a las maquinas.
- Descripción de las relaciones:

  – sin relaciones –
- Descripción de los métodos:

  – sólo contiene “getters” y “setters” –

1.14. ConfiguracionPerfil

- Nombre de la classe:
  - ConfiguracionPerfil
- Breve descripció de la classe:
  - Clase responsable de almacenar la configuración usada en la clase Perfil.
- Cardinalidad:
  - Una instancia diferente para cada perfil
- Descripción de los atributos:
  - tema (Integer): Atributo correspondiente al tema seleccionado por el usuario.
  - volumen (Integer): Atributo correspondiente al volumen configurado por el usuario.
- Descripción de las relaciones:
  - Relación de asociación con la clase “Perfil”: Permite que cada “Perfil” posea su configuración.
- Descripción de los métodos:

  – *sólo contiene “getters” y “setters”* –

1.15. Estadisticas

- Nombre de la classe:
  - Estadisticas
- Breve descripció de la classe:
  - Clase responsable de almacenar las estadísticas de las partidas jugadas por un usuario.
- Cardinalidad:
  - Una instancia diferente para cada perfil
- Descripción de los atributos:
  - partidasGanadas (Integer): Atributo correspondiente al número de partidasGanadas registradas por un usuario (Perfil).
  - partidasPerdidas (Integer): Atributo correspondiente al número de partidasPerdidas registradas por un usuario (Perfil).
  - mejorPuntuacion (double): Atributo que almacena la mayor puntuación conseguida en una partida por un jugador (Perfil).
  - idMejorPartida (Integer): Atributo que corresponde al id de la mejor partida jugada por un jugador (Perfil).
- Descripción de las relaciones:
  - Relación de asociación con la clase “Perfil”: Permite que cada “Perfil” posea sus Estadisticas.
- Descripción de los métodos:
  - anadirPartidaPerdida(): Método encargado de añadir 1 al contador de partidas perdidas por el “Perfil”

- anadirPartidaGanada(): Método encargado de añadir 1 al contador de partidas ganadas por el “Perfil”
- informarPuntuacion(double,int): Método encargado de recibir la nueva posible puntuacion con su id de partida correspondiente y comparar la puntuacion actual con la nueva. En caso de que sea mejor la nueva puntuación se actualiza la puntuacion i se actualiza el id de la partida
- obtenPorcentajeVictoria(): Método encargado de devolver el % de victoria

1.16. Logro

- Nombre de la classe:
  - Logro
- Breve descripció de la classe:
  - Clase responsable de almacenar un logro con su id, descripción y puntuación.
- Cardinalidad:
  - Una instancia para cada logro de perfil diferente
- Descripción de los atributos:
  - id (Integer): Atributo correspondiente al identificador del logro.
  - descripcion (String): Atributo correspondiente a la descripción de un logro.
  - puntuacion (Integer): Atributo que indica la puntuación que obtenemos al completar un logro.
- Descripción de las relaciones:
  - Relación de asociación con la clase “LogroPerfil”: Permite que cada “LogroPerfil” acceda a la información de 1 Logro.
- Descripción de los métodos:

  – sólo contiene “getters” y “setters” –

1.17. LogroPerfil

- Nombre de la classe:
  - LogroPerfil
- Breve descripció de la classe:
  - Clase responsable de almacenar la información de los logros conseguidos por un “Perfil”.
- Cardinalidad:
  - n instancias por cada perfil
- Descripción de los atributos:
  - logrado (Boolean): Atributo que indica si un logro ha sido conseguido.
  - fechaLogrado (LocalDate): Atributo correspondiente a la fecha en la que se ha conseguido un logro.

    - logro (Logro): Atributo almacenador del logro conseguido, es decir, un puntero a estos.
- Descripción de las relaciones:
    - Relación de asociación con la clase “Perfil”: Permite que cada “Perfil” acceda a la información de * Logro. Por tanto, por cada logro obtenido por “Perfil”, tenemos una instancia de “LogroPerfil”.
- Descripción de los métodos:

  – sólo contiene “getters” y “setters” –

1.18. Perfil

- Nombre de la classe:
    - Perfil
- Breve descripció de la classe:
    - Otra de las clases “centrales” del diagrama. Se ocupa de guardar todo lo relacionado con el usuario.
- Cardinalidad:
    - Una única instància, clase estática.
- Descripción de los atributos:
    - nombreUsuario (String): Atributo correspondiente al nombre de usuario. También es identificador de “Perfil”.
    - contrasena (String): Atributo correspondiente a la contraseña del usuario. Va asociada al nombre del usuario y se requiere de esta para iniciar sesión.
    - estadisticas (Estadisticas): Atributo almacenador de las estadisticas de juego de un usuario, es un puntero a estas.
    - partidas (ArrayList<Partida>): Atributo que corresponde a las partidas de un usuario, o sea, un vector de punteros a sus partidas.
    - fechaRegistro (LocalDate): Atributo que almacena la fecha de registro del usuario.
    - ultimaConexion (LocalDateTime): Atributo que almacena la fecha y hora de la ultima conexion del usuario
    - logroPerfil (LogroPerfil[]): Atributo que almacena los logros del usuario, es decir, un puntero a los logros de este.
    - configPerfil (ConfiguracionPerfil): Atributo que almacena un puntero a la configuración de un usuario.
- Descripción de las relaciones:
    - Relación de asociación con la clase “Partida”: Permite que cada “Partida” sea solamente jugada por un único “Perfil”.
- Descripción de los métodos:

- contrasenaCorrecta(String): Método que comprueba si la contraseña introducida como parámetro es la contraseña del usuario que intenta acceder.
- numeroLogroSuperado(): Método que devuelve el numero de logros superados por un usuario.
- obtenIdsPartidasSinAcabar(): Método que devuelve una lista de los ids de las partidas sin finalizar del perfil
- obtenIdsPartidasAcabadas(): Método que devuelve una lista de los ids de las partidas finalizadas del perfil

1.19. RankingGlobal

- Nombre de la classe:
  - RankingGlobal
- Breve descripció de la classe:
  - Clase responsable de almacenar el ranking global, es decir, las partidas con mayor puntuación entre todas.
- Cardinalidad:
  - Singleton
- Descripción de los atributos:
  - ranking(RankingGlobal): Instancia del ranking.
  - partidas (Partida[]): Atributo correspondiente a las partidas almacenadas en el ranking, en nuestro caso máximo 10 partidas.
- Descripción de las relaciones:

  – *sin relaciones* –
- Descripción de los métodos:
  - anadirPartida(Partida): Método que añade una partida al RankingGlobal y actualiza el ranking de partidas.
  - reset(): Método que “resetea” el ranking de partidas.

1.20. RecordsMovimientos

- Nombre de la classe:
  - RecordsMovimientos
- Breve descripció de la classe:
  - Clase responsable de almacenar las partidas que se han acabado usando menos turnos.
- Cardinalidad:
  - Singleton
- Descripción de los atributos:
  - recordsMovimients (RecordsMovimientos): Instancia de recordMoviments

    - records (Pair<Integer,String>[][]): Atributo que guarda una matriz que almacena en cada (fila,columna), correspondientes a longitudCodigo y numeroColores, un jugador y el número de turnos que ha necesitado para solucionar esa configuración.
- Descripción de las relaciones:

  – *sin relaciones* –
- Descripción de los métodos:
    - anadirPartida(String,int,int,int): Método que añade un usuario a RecordsMovimientos teniendo en cuenta la longitud del codigo a romper, el numero de colores del juego y el numero de turnos que ha necesitado para ganar la partida.
    - obtenRecord(int,int): Método que devuelve un pair (turnos, Usuario) con el record de la configuracion pasada por parametro.
    - reset(): Método que “resetea” el records