PROJECTE DE PROP
KAKURO

SEGONA ENTREGA

Grup 4.2
Alvaro Ariño Cabau, alvaro.arino@estudiantat.upc.edu
Albert Bertran Serrano, albert.bertran.serrano@estudiantat.upc.edu
Victor Manuel Delgado Padilla, victor.manuel.delgado@estudiantat.upc.edu
Victor Latorre López, victor.latorre@estudiantat.upc.edu
Versió de lliurament 1.0

Índex

Capa de presentació **3**
Vista login 3
Vista principal 3
Vista jugar 4
Vista paràmetres 5
Vista perfil 5
Vista estadística 6
Vista gestió del programa 6
Vista selecció del perfil 7
Vista galeria 7
Vista partida 8
Vista ajustaments 8
Vista aventura (extra) 9

Capa de domini **10**
Classe Kakuro 10
Classe Tauler 10
Classe Cella 11
Classe CellaNegra 11
Classe CellaBlanca 12
Classe CjtPartida 12
Classe Partida 12
Classe Usuari 13
Classe Perfil 13
Classe Aventura 13
Classe Ranking 13
Classe Stat 14

Capa de dades **15**
CtrlDades 15
CtrlDataKakuro 15
CtrlDataRanking 15
CtrlDataUsuaris 15
CtrlDataPerfil 15
CtrlDataPartida 15
CtrlDataAventura 16

Millora dels algoritmes respecte la primera entrega **17**
Generador 17
Solucionador 17

Capa de presentació

Aquesta és la nostra idea per la capa de presentació, l'encarregada d'interactuar directament amb l'usuari. Es pot comunicar amb la capa de domini però no amb la de dades. Com a mínim a l'ntrega final entregarem la forma, els colors i altres elements seràn incorporats si dóna temps.
En cada controlador de vista tindrem una funció que inicialitzarà tots els valors i una per posibles comunicacions amb els controladors.
Fem servir la classe javafx per fer aquesta part del treball.

Vista login

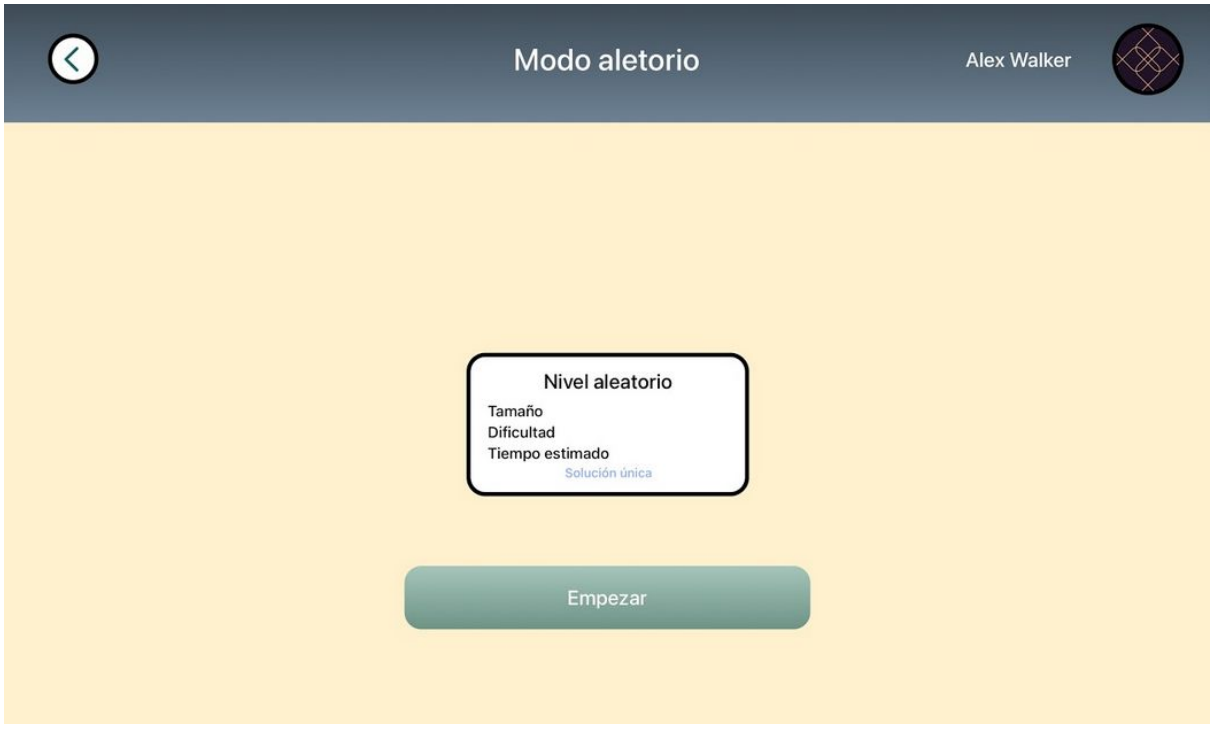
Quan s'executa el joc, veiem la pantalla per fer el login o per registrar-se.

Vista principal

Una vegada ens hem autenticat, podem accedir a la vista principal on podem fer click a jugar, a la galeria de kakuros, les estadístiques, els ajustaments, el perfil o podem sortir.

Vista jugar

Quan fem click a “jugar” podem seleccionar els nivells que volem d’aventura o un kakuro aleatori.

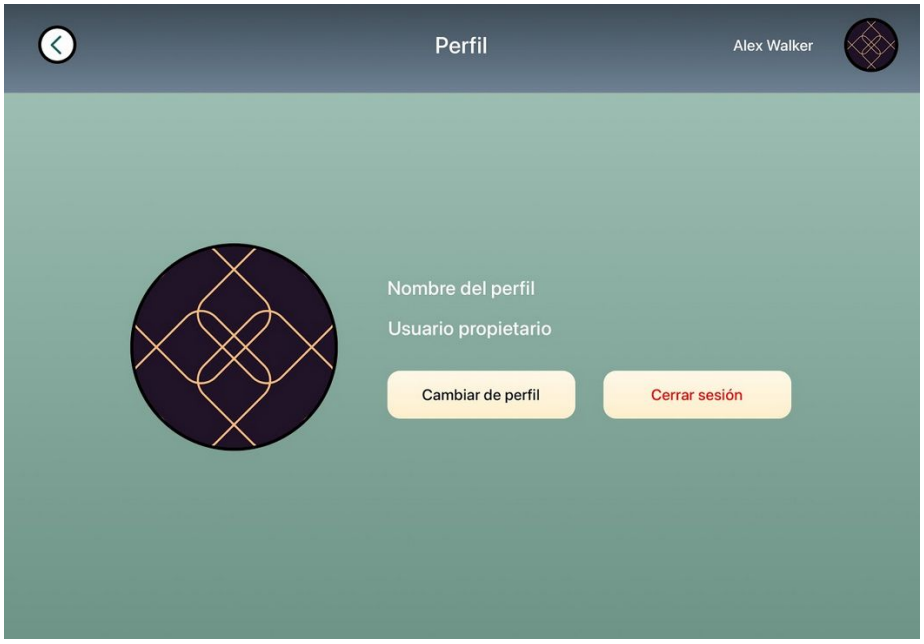


Vista paràmetres

Vista que permet seleccionar els paràmetres del Kakuro per part de l'usuari.

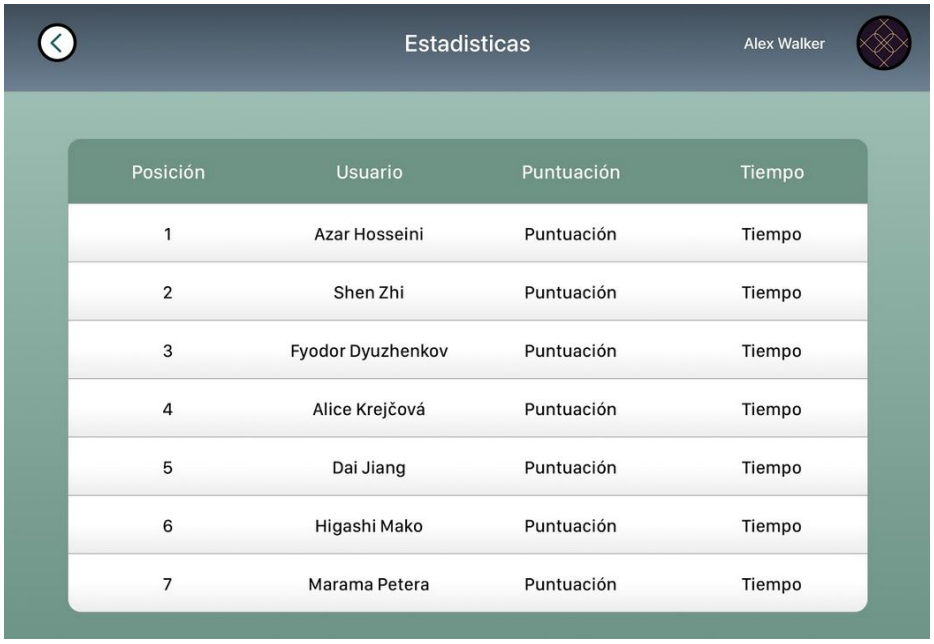
Vista perfil

Aquesta vista mostra el perfil d'un usuari.



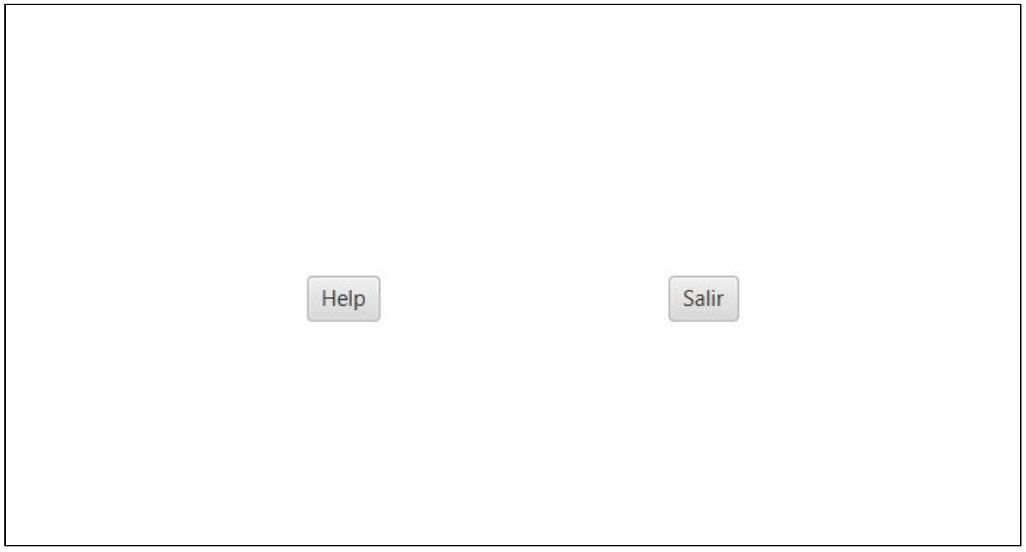
Vista estadística

Aquest apartat del programa mostra les estadístiques dels jugadors de kakuro.



Vista gestió del programa

Aquesta vista és un pop-up que apareixerà quan s'intenta tancar el programa.

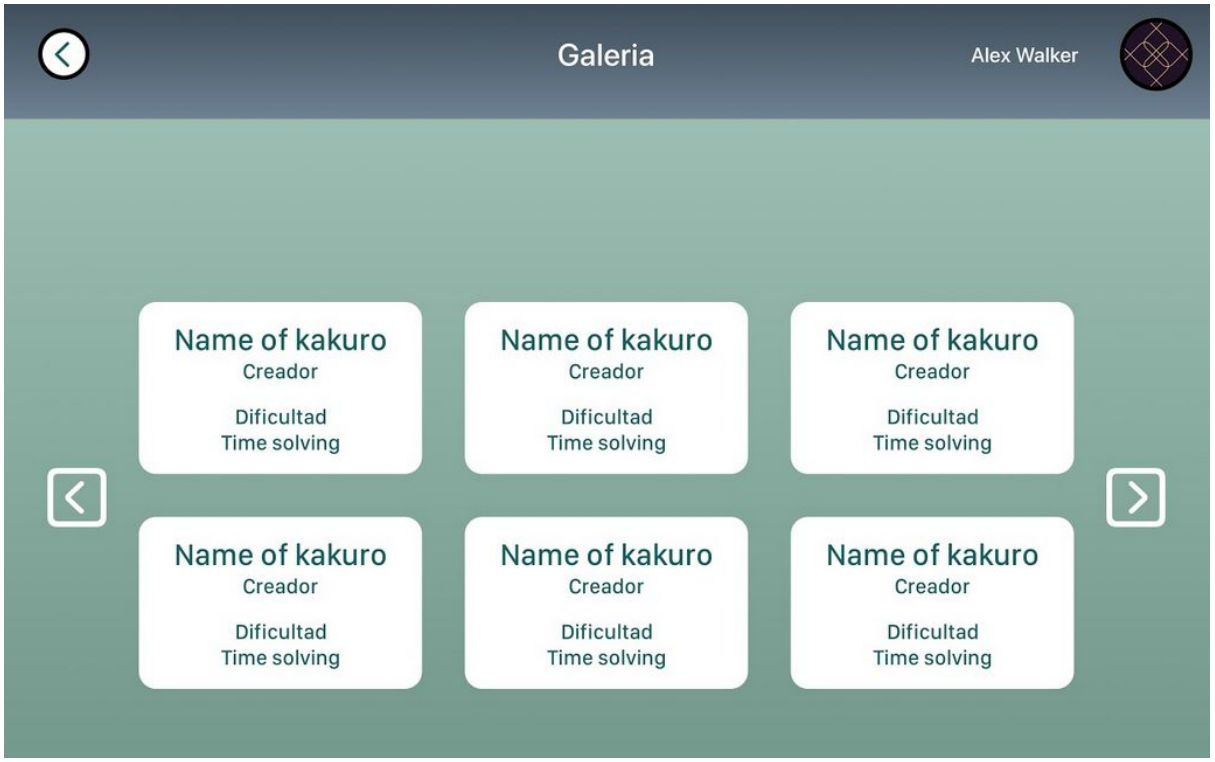


Vista selecció del perfil

Aquí podem veure una selecció entre els perfils que hi ha guardats.

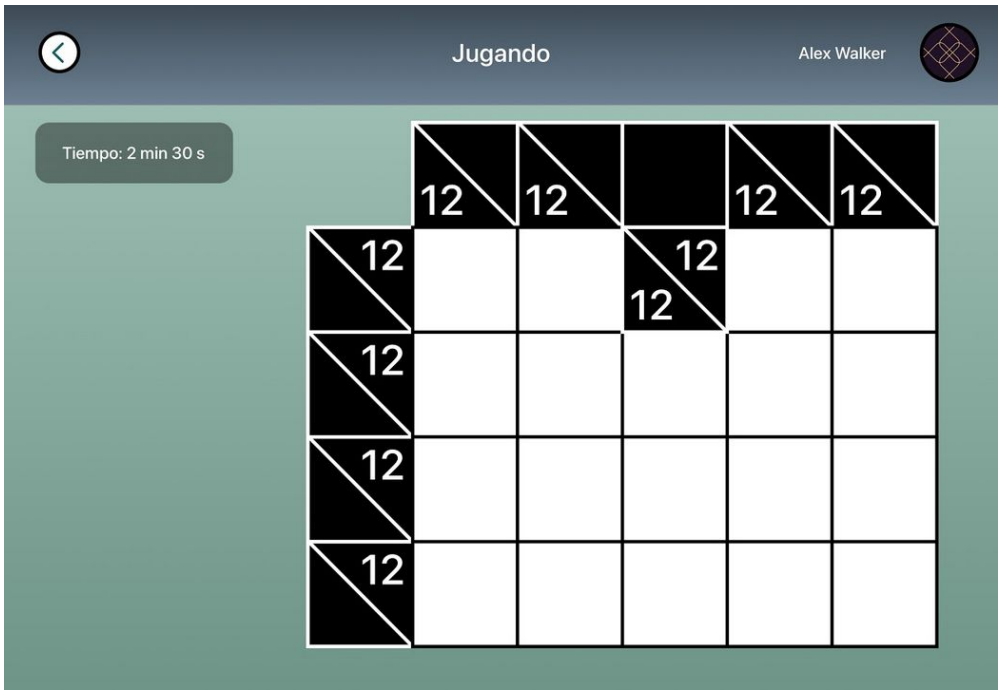
Vista galeria

Galeria dels kakuros guardats en un repositori.



Vista partida

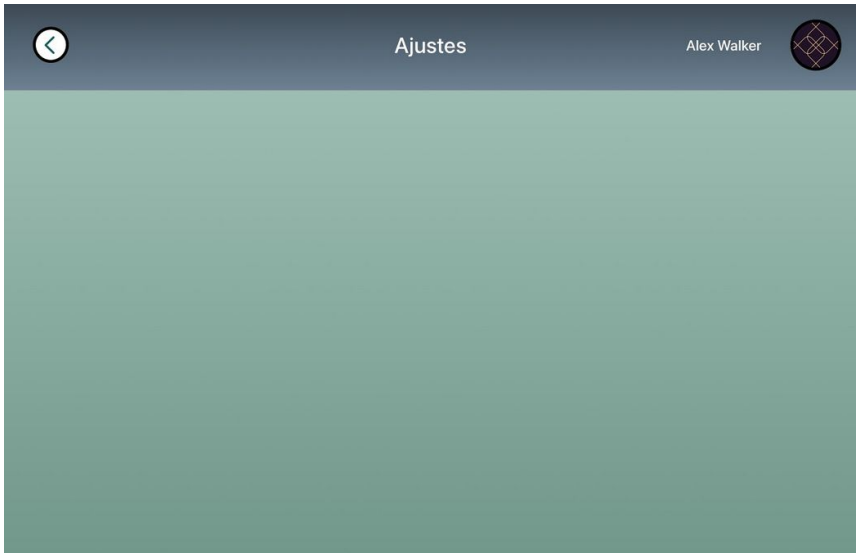
Aquesta és la vista més important del programa. Es tracta de la visió d'un Kakuro. A la pantalla apareixerà un Kakuro, el nom de l'usuari que està jugant, el temps i la puntuació.



Vista ajustaments

Vista dels ajustaments del programa.

En aquest ubicarem posibles canvis de la funcionalitat del programa que puguin ser definits per part de l'usuari.



Vista aventura (extra)

Aquesta vista ha estat plantejada des d'un principi com opcional. Finalment l'hem implementat, i consisteix en un conjunt de kakuros per superar, amb diferents dificultats.

Capa de domini

Descripció de cada classe:

Classe Kakuro

Aquesta classe representa un Kakuro amb el seu Tauler (multiplicitat 1) i un número aleatori aleat. Pot formar part de diverses partides, o de diverses aventures. Les funcions que conté aquesta classe són:

1) **Kakuro(int n,int m):** Constructora amb les dimensions n i m.
2) **Kakuro():** Constructora buida.
3) **generar():** El mètode genera un Kakuro.
4) **generar_usuario(int n, int m, int negras, int blancas):** Es genera un Kakuro amb les especificacions de l'Usuari.
5) **getBoard():** Retorna un Tauler.

Classe Tauler

Aquesta classe representa el tauler d'un Kakuro. Està representat per un conjunt de cel·les dins un Array i una dimensió en n i en m. És per aquesta raó que Tauler i cel·les tenen aquesta multiplicitat, un tauler pot tenir moltes cel·les però una cel·la només pertany a un tauler. A més a més, té relació amb la classe Kakuro perquè un Kakuro té un tauler i un tauler pertany a un Kakuro.

Els mètodes implementats en aquesta classe són els següents:

1) **Tauler(int n, int m):** Constructora de la classe Tauler amb els valors de n i m (amplada i alçada del tauler).
2) **deepCopy(Cella[][] cjt):** Funció per fer una deep copy del conjunt de cel·les.
3) **setTauler(Cella[][] cjt):** Funció per assignar un conjunt de cel·les al tauler actual.
4) **getDimn():** Retorna l'alçada del tauler.
5) **getDimm():** Retorna l'amplada del tauler.
6) **getCella(int i, int j):** Retorna una cel·la en la posició i,j.
7) **possible(int i, int j):** Aquesta funció comproba (retornant true) si una posició és vàlida per assignar-li una cel·la negra.
8) **pintar_celda(int i, int j, int cantidad):** El mètode assigna a una cel·la el tipus cel·la negra.
9) **print_negras(int cantidad):** Assigna la quantitat "cantidad" de negres al tauler.
10) **noPresente(int valor, int i, int j):** retorna true si el valor està present en la mateixa fila i o columna j.
11) **rellenar_celdas_blancas():** Funció que invoca la funció backtracking per omplir les cel·les blanques.
12) **rellenar_blancas_back():** Omple les caselles blanques de números aleatoris entre 1 i 9.
13) **hacer_sumas():** Realitza les sumes de les caselles blanques per tal d'assignar-les a les caselles negres.
14) **borrar_blancas():** Deixa sense valor les caselles blanques.
15) **valorDuplicat(Cella[][] board, int fila, int col, int valor):** comprova si el valor a insertar a la cel·la blanca està duplicat a la fila o a la columna.
16) **valorDuplicatFila(Cella[][] board, int fila, int col, int valor):** comprova si el valor està present a la fila especificada.
17) **valorDuplicatCol(Cella[][] board, int fila, int col, int valor):** comprova si el valor està present a la columna especificada.
18) **solve():** Soluciona el Kakuro implícit.
19) **SolBack2(Cella[][] board, int fila, int col):** Operació recursiva que soluciona el Kakuro.

20) **valorValid(Cella[][] board, int fila, int col, int valor):** Retorna true si ValorValidfila i valorValidCol retornan true.
21) **ValorValidfila(Cella[][] board, int fila, int col, int valor):** Comprova si valor és vàlid a la fila delimitada per fila, col, comprovant els valors introduïts anteriorment.
22) **ValorValidfilaAvall(Cella[][] board, int fila, int col, int valor):** Comprova si valor és vàlid a la fila delimitada per fila, col, comprovant els valors introduïts per sota de la cel·la actual.
23) **valorValidCol(Cella[][] board, int fila, int col, int valor):** Comprova si valor és vàlid a la fila delimitada per col, fila, comprovant els valors introduïts anteriorment.
24) **valorValidColDreta(Cella[][] board, int fila, int col, int valor):** Comprova si valor és vàlid a la fila delimitada per col, fila, comprovant els valors introduïts per sota de la cel·la actual.
25) **print ():** Imprimeix el Kakuro.
26) **printSol():** Imprimeix la solució del kakuro.
27) **validar():** Valida el kakuro actual per verificar si la solució és completa.
28) **comprovarColumna(int i, int j, int valor):** comprova si la suma dels valors de la columna afegint valor dona el valor especificat per la cel·la negra i comprova que no hi hagi cel·les blanques sense valor.
29) **comprovarFila(int i, int j, int valor):** comprova si la suma dels valors de la fila afegint valor dona el valor especificat per la cel·la negra i comprova que no hi hagi cel·les blanques sense valor.

Classe Cella

Classe abstracta representant una casella del tauler. Diverses cel·les conformen un tauler i n'hi ha de dos tipus: blanques i negres. Funcions que conté la classe Cella:

1) **Cella():** Crea una cella buida.
2) **intro_valor_blanca(int z):** Funció abstracta.
3) **getValor_blanca():** Funció abstracta.
4) **setValorFila(int val):** Funció abstracta.
5) **setValorColumna(int val):** Funció abstracta.
6) **acumular_valor_derecha(int s):** Funció abstracta.
7) **acumular_valor_izquierda(int s):** Funció abstracta.
8) **fixCellaBlanca():** Funció abstracta.
9) **resetFixCellaBlanca():**Funció abstracta.
10) **cellaFixed():** Funció abstracta.
11) **color():** Funció abstracta.
12) **getValorDret():** Funció abstracta.
13) **getValorEsquerre():** Funció abstracta.

Classe CellaNegra

Cella del Kakuro que té com atributs valorDret i valorEsquerre.

1) **CellaNegra():** Crea una casella negra amb valor dret i valor esquerra amb valor null.
2) **CellaNegra(int dreta, int esquerra):** inicialita la cel·la negra amb valor dreta i esquerra.
3) **setValorFila(int val):** assigna el valor val a valorDret.
4) **setValorColumna(int val):** assigna el valor val a ValorEsquerre.
5) **acumular_valor_derecha(int s):** Si s'invoca aquesta funció per primer cop (la casella tenia valor null) llavors inicialitzem el valor a 0. Després anem sumant el valor que tenia la nova casella blanca s que entra.

6) **acumular_valor_izquierda(int s):** Si s'invoca aquesta funció per primer cop (la casella tenia valor null) llavors inicialitzem el valor a 0. Després anem sumant el valor que tenia la nova casella blanca s que entra.
7) **getValorDret():** Retorna el valor dret de la casella negra en cas de tenir, sinó retorna null.
8) **getValorEsquerre():** Retorna el valor esquerre de la casella negra en cas de tenir, sinó retorna null.
9) **color():** Retorna blanc(dos valors d'una enumeració).

Classe CellaBlanca

És una cel·la blanca amb un valor que pot ser -1 o entre 1 i 9.

1) **CellaBlanca():** Crea una casella blanca amb valor null.
2) **getValor_blanca():** Retorna el valor entre 1 i 9 de la casella blanca en cas de tenir, sinó retorna null.
3) **intro_valor_blanca(int z):** Assignem el valor a una casella blanca (números entre 1 i 9).
4) **fixCellaBlanca():** deixa fixat un valor de cel·la blanca.
5) **resetFixCellaBlanca():** desfixa el valor d'una cel·la blanca.
6) **cellaFixed():** retorna si una cel·la blanca està fixada.
7) **color():** Retorna blanc(dos valors d'una enumeració).

Classe CjtPartida

Aquesta classe representa un Conjunt de les partides al Kakuro. Les funcions que conté aquesta classe són:

1) **CjtPartida():** Constructora buida.
2) **NuevaPartidaAleatoria():** Crea una partida aleatoria i la afegeix a l'array list de conjunt de partides.
3) **NuevaPartidaDeterminada(int n, int m, int nom):** Crea una partida determinada i la afegeix al array list del conjunt de partides.
4) **SetPartidas(Partida part):** Afegeix una partida ja creada al conjunt de partides.
5) **getPartida(int i):** Retorna la partida en la posició i del Array List de conjunt de partides.
6) **getNumPartides():** Retorna el nombre de partides totals que hi ha al Conjunt.

Classe Partida

Aquesta classe representa una partida al Kakuro, amb el seu temps de joc i estat en el que es troba. Les funcions que conté aquesta classe són:

1. **Partida():** Constructora buida.
2. **Partida(int n, int m, String nom):** Constructora amb les columnes, files y nom definits.
3. **getEstat():** retorna l'estat en el que es troba:
    i. **0:** Jugant, estat per defecte.
    ii. **1:** Pausat, estat en el que no corre el temps i en el que es posa la partida si polsem algun botó per tal de mirar les opcions.
4. **pause():** Canvia l'estat a 1.
5. **restart():** Canvia l'estat a 0.
6. **getTemps():** Retorna el temps de joc.
7. **getKakuro():** Retorna el Kakuro de la partida.
8. **getEstat():** Retorna 1 si l'estat es corrent i 0 si el joc está pausat.
9. **SetKakuroPartida(Kakuro k):** Fa que el kakuro de la partida sigui el kakuro k.
10. **SetTinicial(double i):** Fa que el Tinicial sigui el double i.
11. **SetEstat(int i):** Fa que l'estat siqui el int i.
12. **SetName(String n):** Fa que el nom de la partida sigui el string n.

Classe Usuari

Representa a l'usuari del programa, permet emmagatzemar tots els seu perfils.

L'ID es un UUID que es generar en el moment de la crida del constructor.

1. **Usuari():** Constructora buida
2. **Usuari(String nom, String username):** crea un usuari amb un nom i un username.
3. **getID()**: permet obtenir l'ID
4. **getNom():** permet obtenir el nom de l'usuari.
5. **getUsername():** permet obtenir el username de l'usuari
6. **getNumPerfils()**: retorna el nombre de perfils assignats a l'usuari
7. **setNom():** permet introduir el nom de l'usuari
8. **setPass():** permet afegir una contrasenya a l'usuari
9. **addProfile()**: crea un perfil per l'usuari amb el seu nom

Classe Perfil

Representa un dels perfils de l'usuari, el supòsit d'aquesta classe es poder utilitzar perfils diferents, cadascun amb diferents estadístiques, partides jugades…

L'ID es un UUID que es generar en el moment de la crida del constructor.

1. **Perfil():** Constructora buida
2. **Perfil(String nom):** crea un perfil amb un nom.
3. **setNom():** permet introduir un nom de perfil.
4. **getNom():** permet obtenir el nom del perfil.

Classe Aventura

Aquesta classe representa un conjunt de Kakuros que formen el anomenat "Mode Aventura" on es juguen una determinada quantitat de Kakuros aleatoris. Les funcions que conté aquesta classe són:

1. **Aventura():** Constructora buida amb 8 Kakuros.
2. **getTemps():** Retorna el temps de joc a la Aventura.
3. **getAventura():** Retorna l'ArrayList de l'Aventura.
4. **SetTemps(double ini):** Posa com a TempsIni el tempsini.
5. **SetAventura(ArrayList<Kakuro> av):** seteja av a l'ArrayList d'Aventura.

Classe Ranking

Aquesta classe representa un conjunt amb tots els id dels diferents perfils i el temps total en resoldre tots els kakuros associat a cada perfil. Les funcions que conté són:

1. **getInstance():** Retorna la única instància de la classe ranking.
2. **Ranking():** Constructor per defecte.
3. **inicialitzarValors():** Inicia el ranking.
4. **setRanking(Map<String, Stat> rank):** El ranking de la classe pren per valor el ranking passat a la funció
5. **afegirIndex(String nPerfil, int puntuacio):** Afegeix un nou usuari al ranking, o si ja està present actualitza la seva puntuació.
6. **print():** Pinta el ranking ordenat per puntuació.

Classe Stat

Aquesta classe representa un Struct en C que utilitzem per implementar la classe ranking.

1. **Stat():** Constructor per defecte.
2. **Stat(String nom, int punt):** Constructor amb parametres.
3. **getPuntuació():** Retorna la puntuació del Stat actual.
4. **getPerfil():** Retorna el perfil del Stat actual.
5. **setPuntuació(int p):** Stat actual posa la puntuació actual a p.
6. **setPosicion(int pos):** Assigna la posició dins el ranking a un Stat.

Capa de dades

Aquesta capa és l'encarregada del gestor de disc. És a dir, és l'encarregada de guardar i carregar dades. Pot comunicar-se amb el controlador de domini però no amb les seves classes i amb la capa de presentació directament

CtrlDades

Controlador general de les dades. Inicialitza tots els contraladors de dades. Funcions que conté:

1. **CtrlDades():** creadora.
2. **guardarKakuro(int id, Kakuro k):** guarda el kakuro amb identificador id i kakuro k
3. **guardar_perfil(Perfil p, String nom):** guarda el Perfil.
4. **leer_perfil(String nom):** llegeix el fitxer del perfil nom.
5. **leer_kakuro(String archivo):** llegeix el fitxer d'un kakuro.
6. **leer_ranking():** llegeix el ranking.
7. **actualizar_ranking(Map<String, Stat> rank):** quan es termina el programa, actualitza el ranking.

CtrlDataKakuro

Aquest controlador s'encarrega de guardar els fitxers de kakuro. Utilitzem fitxers .csv per guardar cada kakuro amb el seu tauler en el format corresponent.

1. **getData(String filename):** Obté d'un fitxer filename un Kakuro
2. **guardarKakuro(String id, Kakuro k):** Guarda en un fitxer amb nom id el kakuro k

CtrlDataRanking

S'encarrega de guardar el rànquing dels usuaris. Utilitzem fitxers .csv per guardar el ranking dels usuaris.

1. **obtenirRanking():** Obté el ranking guardat als fitxers de dades.
2. **guardarRanking():** Guarda el ranking al fitxer de dades.
3. **creaFitxerRanking():** Crea el fitxer a on guardar el ranking per primer cop.

CtrlDataUsuaris

Encarregat de guardar els usuaris en fitxers.

1. **getInstance():** constructor del singleton.
2. **CtrlDataUsuaris():** creadora
3. **getData():** extreu l'ArrayList d'Usuaris.

CtrlDataPerfil

Encarregat de guardar les dades del perfil en diferents fitxers.

1. **guardarPerfil(Perfil p, String nom):** Guarda el perfil p en un arxiu amb nom nom.
2. **getPerfil(String filename):** Obté el perfil Perfil d'un arxiu filename.

CtrlDataPartida

Encarregat de guardar les dades de la partida y el conjunt de partides en diferents fitxers.

1. **atoi(String str):** Transforma un string en un int.
2. **atoiTemps(String str):** Transforma un string en un double.
3. **getCjtPartida(String filename):** Obté el CjtPartida d'un fitxer amb nom filename.
4. **guardarCjtPartida(CjtPartida p, String nom):** Guarda el CjtPartida p en un fitxer amb nom nom.
5. **getPartida(String filename):** Obté una partida d'un fitxer amb el nom filename.
6. **guardarPartida(Partida p, String nom):** Guarda la partida p en un fitxer amb nom nom.

CtrlDataAventura

Encarregat de guardar les dades de l'Aventura.

1. **atoi(String str):** Transforma un string en un int.
2. **atoiTemps(String str):** Transforma un string en un double.
3. **cargarAventura(String nom):** Obté una aventura a partir d'un fitxer amb nom nom.
4. **guardarAventura(Aventura av, String nom):** Guarda una aventura av en un fitxer amb nom nom.

Millora dels algoritmes respecte la primera entrega

Respecte a la primera entrega, hem fet canvis en el generador i solucionador. Aquests canvis estan explicats a continuació. A més, hem afegit classes que es poden veure en la capa de domini de l'apartat anterior.

Generador

- En primer lloc generem un kakuro de mida n*m i l'omplim de cel·les negres en funció del valor entrat per l'usuari o assignat aleatòriament per el programa.
- Un cop assignades les cel·les negres assignem valors aleatoris a les cel·les blanques, controlant que no hi hagi valors repetits.
- A continuació fem les respectives sumes dels valors de les files i les columnes i afegim el resultat a les cel·les negres pertinents.
- Per acabar esborrem el valor de les cel·les blanques a excepció del nombre de cel·les blanques que l'usuari vol deixar ja col·locades. En cas que el kakuro es generi aleatòriament pel programa, no deixa cap cel·la blanca.

Solucionador

- Aquesta funció utilitza backtracking per trobar les solucions del kakuro.
- En primer lloc hi ha la funció invocadora, la qual gestiona el nombre de solucions trobades, més d'una, una o zero. A continuació invoca a la funció que solucionarà el kakuro.
- A cada iteració de la funció que implementa el backtracking, es mira en primer lloc si
- hem arribat a una solució final; en segon lloc si hem arribat al final d'una columna, si així és invoquem de nou la funció incrementant la fila; en tercer lloc comprovem si la cel·la és negra, si ho és passem al següent valor i invoquem la funció de nou; per últim comprovem si la cel·la actual està fixada(el valor introduït a la cel·la blanca no es pot modificar), si ho està passem al següent valor i invoquem la funció.
- Un cop passades les comprovacions anem iterant per buscar quin és un valor vàlid per posar a la cel·la especificada per fila,col. Per ser un valor vàlid no pot estar present ni a la fila ni a la columna especificada. Si és vàlid, s'introdueix el nou valor a la cel·la especificada, s'invoca la funció de nou i es treu el valor(per trobar així totes les solucions).
  Estructura de dades utilitzada:
  - Per implementar el kakuro hem utilitzat una array d'arrays de cel·les.