PROP

Classes compartides

Dades del projecte

Clúster	7
Versió de lliurament	2.1
Data d'entrega	19 de novembre de 2012

Membres del clúster

Othello

Moré Guardiola, Àlex	alex.more
Pla Alonso, Alex	alex.pla
Salvany Peyri, Horaci	horaci.salvany

Gomoku

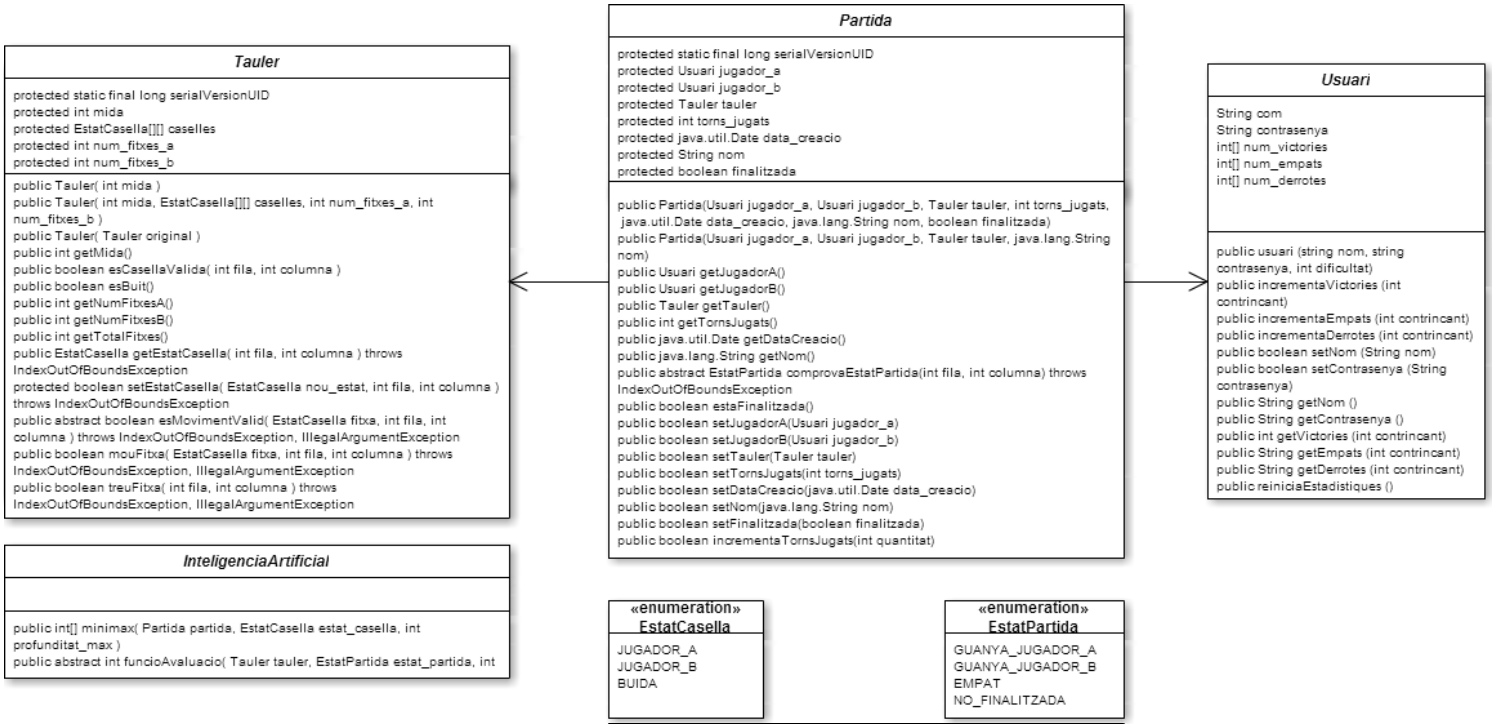
Contreras Pinilla, Mauricio Ignacio	mauricio.ignaci.contreras
Gimenez Ortega, Alexander	alexander.gimenez
Riera Perez, Genis	genis.riera.perez

Hex

Ferrer González, Javier	javier.ferrer.gonzalez
Girona San Miguel, Guillermo	guillermo.girona
Junyent Martín, Marc	marc.junyent
Sánchez Barrera, Isaac	isaac.sanchez.barrera

Índex

Diagrama estàtic de les classes compartides



Para visualizarlo en grande se puede acceder a la URL:
[Diagrama Cluster]

Especificació de les classes compartides

Enum EstatCasella

- JUGADOR_A
- JUGADOR_B
- BUIDA

Classe Tauler

Representa un tauler d'un joc. Les caselles vàlides són les que tenen la fila i la columna en l'interval 0..mida-1.
Té un control de les fitxes que té cada jugador al tauler i revisa que les fitxes s'afegeixen d'acord amb la normativa del joc mitjançant un mètode redefinible.

Atributs

- protected static final long serialVersionUID
  ID de serialització
- protected int mida
  La mida del tauler.
- protected EstatCasella[][] caselles
  Array bidimensional mida × mida amb l'estat actual de les caselles.
- protected int num_fitxes_a
  Quantitat de fitxes que té el jugador A
- protected int num_fitxes_b
  Quantitat de fitxes que té el jugador B.

Mètodes públics i protegits

- public **Tauler**(int mida)
  Constructor del tauler. Crea un tauler de la mida desitjada amb totes les caselles buides (EstatCasella.BUIDA).
  Paràmetres:
  - mida
    Les dimensions del tauler.

- public **Tauler**(int mida, EstatCasella[][] caselles, int num_fitxes_a, int num_fitxes_b)
  Constructor que inicialitza el tauler a un estat diferent del per defecte. No comprova que els paràmetres siguin correctes.
  Paràmetres:
  - mida
    Les dimensions del tauler
  - caselles
    Un array bidimensional mida × mida amb l'estat inicial
  - num_fitxes_a
    La quantitat de fitxes que té el jugador A al tauler
  - num_fitxes_b
    La quantitat de fitxes que té el jugador B al tauler

- public **Tauler**(Tauler original)
  Constructor per còpia. Crea un nou tauler idèntic a original.
  Paràmetres:
  - original
    Tauler que es vol copiar

- public int **getMida**()
  Consulta la mida del tauler.
  Retorna:
  La mida del tauler.

- public boolean **esCasellaValida**(int fila, int columna)
  Comprova si una casella és vàlida dins el tauler.
  Paràmetres:
  - fila
    Fila de la casella dins el tauler.
  - columna
    Columna de la casella dins el tauler.

  Retorna: Cert si la posició (fila, columna) és una casella vàlida. Fals altrament.

- public boolean **esBuit**()
  Consulta si el tauler és buit
  Retorna:
  Cert si el tauler no té cap fitxa. Fals altrament.

- public int **getNumFitxesA**()
  Consulta les fitxes del jugador A.
  Retorna:
  La quantitat de fitxes del jugador A.

- public int **getNumFitxesB**()
  Consulta les fitxes del jugador B.
  Retorna:
  La quantitat de fitxes del jugador B.

- public int **getTotalFitxes**()
  Consulta la quantitat de fitxes que hi ha al tauler.
  Retorna:
  La quantitat total de fitxes que tenen els dos jugadors al tauler.

- public EstatCasella **getEstatCasella**(int fila, int columna) throws *IndexOutOfBoundsException*
  Consulta l'estat d'una casella del tauler.

Paràmetres:

- fila
  Fila de la casella del tauler que es vol consultar.
- columna
  Columna de la casella del tauler que es vol consultar.

Retorna:
L'estat actual de la casella.
Excepcions:

- *IndexOutOfBoundsException* si (fila, columna) no és una casella vàlida.

- protected boolean **setEstatCasella**(EstatCasella nou_estat, int fila, int columna) throws *IndexOutOfBoundsException*
  Canvia l'estat d'una casella i actualitza els comptadors. No realitza comprovacions de normativa.
  Paràmetres:
  - nou_estat
    Estat nou de la casella
  - fila
    Fila de la casella del tauler que canvia d'estat
  - columna
    Columna de la casella del tauler que canvia d'estat

  Retorna:
  Cert si el canvi ha estat realitzat amb èxit. Fals altrament.
  Excepcions:
  - *IndexOutOfBoundsException* si (fila, columna) no és una casella vàlida.

- public abstract boolean **esMovimentValid**(EstatCasella fitxa, int fila, int columna) throws *IndexOutOfBoundsException*, *IllegalArgumentException*
  Comprova si un moviment és vàlid.
  Paràmetres:
  - fitxa
    Fitxa que es vol comprovar
  - fila
    Fila de la casella dins el tauler.
  - columna
    Columna de la casella dins el tauler.

  Retorna:
  Cert si el moviment és vàlid. Fals altrament.
  Excepcions:
  - *IndexOutOfBoundsException* si (fila, columna) no és una casella vàlida.
  - *IllegalArgumentException* si fitxa no és de cap jugador (és EstatCasella.BUIDA).

- public boolean **mouFitxa**(EstatCasella fitxa, int fila, int columna) throws *IndexOutOfBoundsException*, *IllegalArgumentException*
  Mou la fitxa a la casella indicada i actualitza els comptadors.
  Paràmetres:
  - fitxa
    Fitxa que es vol col·locar.
  - fila
    Fila de la casella dins el tauler.
  - columna
    Columna de la casella dins el tauler.

  Retorna:
  Cert si s'ha realitzat el moviment. Fals altrament.
  Excepcions:
  - *IndexOutOfBoundsException* si (fila, columna) no és una casella vàlida.
  - *IllegalArgumentException* si fitxa no és de cap jugador (és EstatCasella.BUIDA) o el moviment no és vàlid.

- public boolean **treuFitxa**(int fila, int columna) throws *IndexOutOfBoundsException*, *IllegalArgumentException*
  Treu la fitxa de la casella indicada i actualitza els comptadors.
  Paràmetres:
  - fila
    Fila de la casella dins el tauler.
  - columna
    Columna de la casella dins el tauler.

  Retorna:
  Cert si s'ha realitzat el moviment. Fals altrament.
  Excepcions:
  - *IndexOutOfBoundsException* si (fila, columna) no és una casella vàlida.
  - *IllegalArgumentException* si la casella és buida (és EstatCasella.BUIDA).

- public boolean **intercanviaFitxa**(int fila, int columna) throws *IndexOutOfBoundsException*, *IllegalArgumentException*
  Intercanvia la fitxa d'una casella amb la de l'altre jugador i actualitza els comptadors. Llança excepció IllegalArgumentException si (fila, columna) no és una casella vàlida. o si no conté cap fitxa.
  Paràmetres:
  - fila
    Fila de la casella dins el tauler.
  - columna
    Columna de la casella dins el tauler.

  Retorna:
  Cert si s'ha intercanviat la fitxa. Fals altrament.
  Excepcions:
  - *IndexOutOfBoundsException* si (fila, columna) no és una casella vàlida.
  - *IllegalArgumentException* si la casella és buida (és EstatCasella.BUIDA).
- public String **toString**()
  Crea un String amb tota la informació del tauler.
  Retorna: El String amb la informació completa del tauler.

Enum EstatPartida

Representa l'estat d'una partida

- GUANYA_JUGADOR_A
- GUANYA_JUGADOR_B
- EMPAT
- NO_FINALITZADA

Classe Partida

Representa una partida on juguen dos usuaris i que es desenvolupa a un tauler. S'identifica per la seva data i hora de creació, però també té un nom assignat per facilitar la seva identificació de cara als usuaris. Conté informació relativa al nombre de torns jugats i a l'estat de finalització de la partida.

Atributs (protected)

- protected static final long serialVersionUID
  ID de serialització
- protected Usuari jugador_a
  Usuari que farà de jugador A
- protected Usuari jugador_b
  Usuari que farà de jugador B
- protected Tauler tauler
  Tauler on es desenvolupa la partida
- protected int torns_totals
  Nombre de torns completats *(para obtener a qué jugador le toca, cada uno implementaría el mod 2 y a tirar)*
- protected String nom
  Cadena de text que serveix per anomenar la partida
- protected boolean partida_finalitzada
  Indica si la partida ha estat finalitzada o no

Mètodes

- public **Partida**(Usuari jugador_a, Usuari jugador_b, Tauler tauler, int torns_jugats, java.util.Date data_creacio, java.lang.String nom, boolean finalitzada)
  Constructora amb tots el paràmetres
  Paràmetres:
  - jugador_a
    Usuari que fa de jugador A
  - jugador_b
    Usuari que fa de jugador B
  - tauler
    Tauler on es desenvolupa la partida
  - torns_jugats
    Torns completats a la partida
  - data_creacio
    Data i hora de creació de la partida
  - nom
    Nom de la partida

- finalitzada
  Indica si ha estat finalitzada o no

- public **Partida**(Usuari jugador_a, Usuari jugador_b, Tauler tauler, java.lang.String nom)
  Constructora alternativa per partides que no han estat jugades
  Paràmetres:
  - jugador_a
    Usuari que farà de jugador A
  - jugador_b
    Usuari que farà de jugador B
  - tauler
    Tauler on es desenvoluparà la partida
  - nom
    Nom de la partida

- public Usuari **getJugadorA**()
  Mètode consultor del jugador A
  Retorna:
  Usuari que fa de jugador A

- public Usuari **getJugadorB**()
  Mètode consultor del jugador B
  Retorna:
  Usuari que fa de jugador B

- public Tauler **getTauler**()
  Mètode consultor del tauler
  Retorna:
  Tauler on es desenvolupa la partida

- public int **getTornsJugats**()
  Mètode consultor del nombre de torns jugats
  Retorna:
  Nombre de torns jugats

- public java.util.Date **getDataCreacio**()
  Mètode consultor de la data i hora de creació de la partida
  Retorna:
  Data i hora de creació de la partida

- public java.lang.String **getNom**()
  Mètode consultor del nom de la partida
  Retorna:

Nom de la partida

- public boolean **estaFinalitzada**()
  Mètode consultor de si una partida ha estat finalitzada o no
  Retorna:
  true si la partida ha estat finalitzada; *false* en cas contrari

- public abstract EstatPartida **comprovaEstatPartida**(int fila, int columna) throws *IndexOutOfBoundsException*
  Mètode consultor de l'estat de la partida. Els paràmetres permeten aportar informació a sobre de l'últim moviment d'interès realitzat (normalment l'últim realitzat correctament).
  Paràmetres:
  - fila
    Fila del moviment d'interès
  - columna
    Col·lumna del moviment d'interès

  Retorna:
  L'estat de la partida
  Excepcions:
  - *IndexOutOfBoundsException* si (fila, columna) no és una coordenada dins dels límits del tauler on es desenvolupa la partida

- public boolean **setJugadorA**(Usuari jugador_a)
  Mètode modificador del jugador A
  Paràmetres:
  - jugador_a
    Usuari que farà de jugador A

  Retorna:
  true si el canvi s'ha realitzat, false si no s'ha realitzat ja que el valor proveït no és vàlid

- public boolean **setJugadorB**(Usuari jugador_b)
  Mètode modificador del jugador B
  Paràmetres:
  - jugador_b
    Usuari que farà de jugador B

  Retorna:
  true si el canvi s'ha realitzat, false si no s'ha realitzat ja que el valor proveït no és vàlid

- public boolean **setTauler**(Tauler tauler)
  Mètode modificador del tauler
  Paràmetres:
  - tauler
    Tauler on es desenvoluparà la partida

  Retorna:

true si el canvi s'ha realitzat, false si no s'ha realitzat ja que el valor proveït no és vàlid

- public boolean **setTornsJugats**(int torns_jugats)
  Mètode modificador del nombre de torns jugats
  Paràmetres:
  - torns_jugats
    Nombre de torns jugats

  Retorna:
  true si el canvi s'ha realitzat, false si no s'ha realitzat ja que el valor proveït no és vàlid

- public boolean **setDataCreacio**(java.util.Date data_creacio)
  Mètode modificador de la data i hora de creació
  Paràmetres:
  - data_creacio
    Data i hora de creació

  Retorna:
  true si el canvi s'ha realitzat, false si no s'ha realitzat ja que el valor proveït no és vàlid

- public boolean **setNom**(java.lang.String nom)
  Mètode modificador del nom de la partida
  Paràmetres:
  - nom
    Nom de la partida

  Retorna:
  true si el canvi s'ha realitzat, false si no s'ha realitzat ja que el valor proveït no és vàlid

- public boolean **setFinalitzada**(boolean finalitzada)
  Mètode modificador de l'estat de finalització de la partida
  Paràmetres:
  - finalitzada
    Indica si la partida ha estat finalitzada o no

  Retorna:
  true si el canvi s'ha realitzat, false si no s'ha realitzat ja que el valor proveït no és vàlid

- public boolean **incrementaTornsJugats**(int quantitat)
  Incrementa el nombre de torns jugats en la quantitat indicada
  Paràmetres:
  - quantitat
    Quantitat en la que incrementar el nombre actual de torns jugats

  Retorna:
  true si el canvi s'ha realitzat, false si no s'ha realitzat ja que el valor proveït no és vàlid

- public String **toString**()
  Crea un String amb informació de la partida.
  Retorna: Text amb informació de la partida

Classe InteligenciaArtificial

La classe InteligenciaArtificial proporciona una implementació estàndard de l'algorisme MiniMax amb l'optimització de poda alfa-beta. Per a més informació sobre el funcionament de l'algorisme poden consultar aquest enllaç: http://www.lsi.upc.edu/~bejar/heuristica/docmin.html

Atributs

(no en té cap)

Mètodes

- **public int[] minimax**(Partida partida, EstatCasella estat_casella, int profunditat_max);
  Donada una partida en una certa situació i la fitxa del jugador que ha de moure durant el torn actual, calcula quina és la millor posició del tauler on realitzar el següent moviment, seguint l'algorisme MiniMax. Com que per aconseguir aquest càlcul és necessari generar una estructura arbòria on cada nivell representa el pròxim torn i, en un mateix nivell, es generen tots els possibles moviments a realtzar, també cal tenir un límit que trunqui la cerca, per evitar que el cost temporal del MiniMax augmenti exponencialment.
  Paràmetres:
  - partida
    Objecte de la classe Partida que representa la partida actual en joc.
  - estat_casella
    Representa la fitxa del jugador que ha de disputar el torn actual de la partida *partida*.
  - profunditat_max
    Representa el nivell límit en la cerca arbòria del moviment òptim.

  Retorna:
  Retorna la posició del tauler òptima on el jugador controlat per aquesta intel·ligència artificial ha de fer el seu moviment. La posició ve representada per les seves dues coordenades dins del tauler (número de fila i número de col·lumna).

- **public abstract int funcioAvaluacio**(Tauler tauler, EstatPartida estat_partida, int profunditat, EstatCasella fitxa_jugador);
  Avalua la disposició d'un objecte de la classe Tauler seguint l'heurísitca que s'implementi.
  Paràmetres:
  - tauler
    Objecte de la classe Tauler sobre el qual es disputa una partida.
  - estat_partida
    Descriu en quin estat ha quedat *tauler* en funció de l'últim moviment efectuat sobre aquest.
  - profunditat
    És la profunditat a la que s'ha arribat durant l'exploració de les diferents possibilitats de moviment. Cada unitat de profunditat representa un torn jugat de la partida.

  Retorna:
  Retorna un enter indicant l'avaulació de *tauler.*

Classe Usuari

Representa un usuari del joc. S'identifica pel seu nom, el cual té assignada una contrasenya. També conté informació sobre les estadístiques del usuari en les partides jugades en diferents nivells de dificultat (número de victories, número de empats i número de derrotes).

Atributs

- String nom
  El nom del usuari
- String contrasenya
  La contrasenya del usuari en el sistema.
- int[] num_victories
  Array on cada posició hi ha el número de victories del usuari contra diferents tipus de contrincants.
- int[] num_empats
  Array on cada posició hi ha el número de empats del usuari contra diferents tipus de contrincants.
- int[] num_derrotes.
  Array on cada posició hi ha el número de derrotes del usuari contra diferents tipus de contrincants.

Mètodes públics

- public **Usuari** (string nom, string contrasenya, int dificultat)
  Constructor d'usuari. Crea un usuari amb el nom i contrasenya desitjats. Els altres atributs s'inicialitzen a 0.
  Paràmetres:
  - nom
    Nom de l'usuari.
  - contrasenya
    Contrasenya asignada al usuari.
  - dificultat
    Número total de dificultats en el joc (1 contra altres usuaris + N contra màquina).

- public **incrementaEmpats** (int contrincant)
  Mètode per incrementar les victòres d'un usuari contra un contrincant determinat.
  Paràmetres:
  - contrincant
    Contrincant contra qui ha guanyat l'usuari.

- public **incrementaEmpats** (int contrincant)
  Mètode per incrementar els empats d'un usuari contra un contrincant .
  Paràmetres:
  - dificultat
    Contrincant contra qui ha empatat l'usuari.

- public **incrementaDerrotes** (int contrincant)
  Mètode per incrementar les derrotes d'un usuari contra un contrincant
  Paràmetres:
  - dificultat
    Contrincant contra qui ha perdut l'usuari.

- public boolean **setNom** (String nom)
  Mètode per definir el nom d'usuari.
  Paràmetres:
  - nom
    Nom de l'usuari que se li vol assignar.

  Retorna:
  true si el canvi s'ha realitzat, false si no s'ha realitzat ja que el valor proveït no és vàlid.

- public boolean **setContrasenya** (String contrasenya)
  Mètode der definir una contrasenya.
  Paràmetres:
  - contrasenya
    Contrasenya de l'usuari que se li vol assignar.

  Retorna:
  true si el canvi s'ha realitzat, false si no s'ha realitzat ja que el valor proveït no és vàlid.

- public String **getNom** ()
  Mètode consultor del nom d'usuari.
  Retorna:
  un String amb el nom de l'usuari.

- public String **getContrasenya** ()
  Mètode consultor de la contrasenya de l'usuari.
  Retorna:
  un String amb la contrasenya de l'usuari.

- public int **getVictories** (int contrincant)

Mètode consultor de les victòres contra un contrincant concret.
Paràmetres:
- contrincant
  Contrincant contra qui ha guanyat l'usuari.

Retorna:
El número de victòries contra un contrincant concret.

- public int **getEmpats** (int contrincant)
  Mètode consultor dels empats contra un contrincant concret.
  Paràmetres:
  - contrincant
    Contrincant contra qui ha empatat l'usuari.

  Retorna:
  El número de empats contra un contrincant concret.

- public int **getDerrotes** (int contrincant)
  Mètode consultor de les derrotes contra un contrincant concret.
  Paràmetres:
  - contrincant
    Contrincant contra qui ha perdut l'usuari.

  Retorna:
  El número de derrotes contra un contrincant concret.

- public **reiniciaEstadistiques** ()
  Mètode que reinicia les estadístiques de victòries, empats i derrotes d'un usuari

Estàndards de codificació i estil

- Característiques generals dels fitxers
    - Límit de 120 caracters per línia.
    - Finals de línia tipus *Unix* (*LF*)
    - La indentació es realitzarà amb tabuladors de 4 espais.
    - Tots els fitxers han d'acabar amb una línia en blanc.
- Idioma:
    - Variables, mètodes i comentaris/documentació serà en català (excepte per les paraules *get* i *set* als mètodes de consulta i modificació)
    - Ela geminada es simplificaran per ela simple, *l* (col·loca fitxa => colocaFitxa()).
- Noms:
    - Els noms de classe estaran en *UpperCamelCase*.
    - Els noms de mètodes estaran en *lowerCamelCase*.
    - Els noms de variables estaran en *snake_case*.
    - Els noms dels mètodes seran en imperatiu (`intercanviaFitxa()`, `incrementaTorn()` …).
- Comentaris: definim 3 tipus
    - Comentaris "de línia": simplement // Breu justificació necessària
    - Comentarios de bloc: /* Explicació llarga de més d'una línia, amb cada línia intermitja començant per *. El bloc acaba amb */
    - Comentarios de capçalera: /** Descripció del mètode i els seus paràmetres i altres propietats segons la sintaxi de Javadoc */
- Altres consideracions d'estil:
    - Els blocs de codi, encara que el seu cos sigui d'una única línia, s'obriran i tancaran amb les claus corresponents.
    - Les claus d'un bloc de codi han d'estar a una línia exclusiva per elles (no estarà permès per tant: `if ( true ) { return true }`)
    - Als blocs *if*, hi han espais abans i després del parèntesi inicial
    - En les crides i definicions dels mètodes, al llistat de paràmetres, hi ha un parèntesi després del parèntesi d'obertura (sense espais abans d'aquest)
- Consideracions tèniques:
    - Sempre que sigui possible s'utilitzaran tipus primitius de dades (*e.g. int* abans que *Integer*).
    - Els *setters* han de retornar booleans encara que pugin llençar excepcions

- Exemple de codi que segueix l'estàndar:

```
/**
 * Consulta el número del jugador que hi ha a la casella (fila, columna)
 *
 * @param fila    Fila on està la casella que es vol consultar
 * @param columna Col·lumna on està la casella que es vol consultar
 * @return El número del jugador que hi ha a la casella, null si no hi ha.
 */
public Integer getNumJugadorCasella( Integer fila, Integer columna )
{
        // bloc if d'exemple, no té sentit, només per a demostrar espais
         if ( true )
         {
              System.out.println( usuari.getNom() );
         }

        return tauler[fila][columna];
}
```

Grups responsables de cada classe

- Classe Tauler i EstatCasella — Grup Hex
- Classe Usuari — Grup Othello
- Classe InteligenciaArtificial, Partida i EstatPartida — Grup Gomoku