Especificació domini

Classe controlador_domini:

Es la classe encarregada de gestionar totes les classes de domini. També es la classe encarregada de comunicar-se amb la classe Persistence per tal d'escriure o llegir tot el que sigui necessari dels fitxers de dades.

Atributs:

- controlador_domini instance -> Instancia del controlador de domini, implementa el patró singleton.

- jugador J -> Instància de la classe jugador que fa referència al jugador que ha fet el login.

- partida P -> Instància de la classe partida que fa referència a la partida que es juga.

- Board B -> Instància de la classe Board que es fa servir per crear taulers personalitzats.

Mètodes:

- controlador_domini() -> Es la constructora de la classe, es privada ja que implementa el patró singleton

+ static controlador_domini getInstance() -> Mètode public que retorna la única instància de la classe en cas que ja existeixi, en cas que no existeixi primer la crea i després la retorna.

+ boolean crear_usuari(String name, String password) -> Mètode públic que realitza una crida al mètode createUser del controlador de persistencia. En cas que ja existeixi un usuari amb el nom passat com a paràmetre es captura l'excepció que saltara des del controlador de persistencia i retornarà false, i en cas que tot vagi bé retornarà true.

+ boolean comprovar_credencials (String nom, String password) -> Mètode públic que realitza una crida al mètode getPassword del controlador de persistencia i comprova si la contrasenya passada com a paràmetre és igual que la que retorna el mètode. En cas que siguin iguals retorna true i en cas que siguin diferents retorna false. Si l'usuari no existeix saltará la excepció IOException que es capturara i retornarà false.

+ Vector<Vector<String>> consultar_ranking(boolean Diagonal, boolean Horitzontal, boolean Vertical) -> Mètode públic que realitza una crida al mètode readRanking del controlador de persistencia i retorna un vector de vectors de Strings amb el contingut del ranking del mode de joc passat com a paràmetre fent servir els tres booleans.

+ void act_ranking () -> Mètode públic que realitza una crida al mètode actualitzar_ranking de la classe ranking amb els paràmetres que s'extreuen de la partida un cop aquesta ha acabat.

+ String[] consultar_partides_guardades() -> Mètode públic que realitza una crida al mètode getGames del controlador de persistencia i retorna un conjunt de Strings amb el nom de les partides guardades pel jugador atribut de la classe.

+ char [] carregar_partida(String nom_partida) -> Mètode públic que realitza una crida al mètode readGame amb el nom de la partida que es vol carregar del controlador de persistencia i genera una instància de partida amb els paràmetres obtinguts de la crida al mètode readGame. Retorna un conjunt de caràcters que representen el tauler de la partida.

+ char [] crear_partida (int num_maq, String j1, String j2, String tauler, boolean d, boolean h, boolean b, boolean or) -> Mètode públic que genera una instància de partida amb els atributs passats com a paràmetre i retorna un conjunt de caràcters que representen el tauler de la partida.

+ char [] colocar_fitxa_wc (int x, int y, char color) -> Mètode que permet colocar un fitxa del color "color" al tauler sese capturar. Serveix per poder crear taulers personalitzats. Retorna l'estat del tauler després de posar la fitxa a la posició passada com a paràmetre.

+ char [] eliminar_fitxa (int x, int y) -> Mètode que permet eliminar una fitxa del tauler. Serveix per poder eliminar una fitxa mentre es crea un tauler personalitzat.

+ char [] colocar_fitxa(Point p, char color) -> Mètode públic que realitza una crida al mètode jugar de la classe partida i retorna un conjunt de caràcters que representen el tauler de la partida després de realitzar la tirada.

+ void guardar_partida(String nom_partida, boolean overwrite) -> Mètode públic que realitza una crida al metode toString de la classe partida i fa servir el mètode seveGame del controlador de persistencia per escriure el String generat. En cas de que ja existeix una partida amb el nom passat com a paràmetre es captura l'excepció ConfirmationException. En aquest cas es retorna un false. En cas que l'usuari vulgui sobreescriure la partida es torna a realitzar la mateixa crida però aquest cop amb el paràmetre overwrite a true.

+ void guardar_tauler(String nom) -> Mètode encarregat de guardar el tauler creat per l'usuari amb el nom passat com a paràmetre.

+ void rendirse(int jugador) -> En cas que un dels dos jugadors de la partida decideixi rendir-se la partida s'acaba i es compte com a puntuació les fitxes que tenen cadascun dels jugadors en el moment de la rendició. Sempre es compta com a derrota per al jugador que s'ha rendit.

+ void esborrar_tauler(String nom) -> Mètode que s'encarrega d'esborrar el tauler guardat amb nom "nom" creat pel jugador de la instància de controlador_domini.

+ void esborrar_partida(String nom) -> Mètode encarregat d'esborrar la partida guardada amb nom "nom" guardada pel jugador de la instància de controlador_domini.

+ void esborrar_usuari() -> Mètode que s'encarrega d'esborrar el jugador de la instància de controlador_domini.

Classe jugador:

Classe encarregada de representar un jugador.

Atributs:

- username -> Nom que rep el jugador, és el seu identificador, no pot haver-hi dos jugadors amb el mateix username.

Mètodes:

+ jugador (String nom) -> Constructora de la classe. El paràmetre nom es convertirà en el username del jugador.

+ jugador() -> Constructora de la classe sense paràmetres. Es crea un jugador sense username.

+ String getUsername() -> Mètode que ens permet obtenir el username del jugador.

+ void setUsername(String red) -> Metode que ens permet afegir o canviar el username a un jugador, en cas que aquest no sigui una maquina.

Classe maquina:

Subclasse de jugador que permet representar un jugador maquina, es a dir un jugador que no està controlat per un humà.

Atributs:

- int profunditat -> És la profunditat que farà servir l'algorisme a l'hora de calcular les possibles jugades.

- String algorisme -> És el nom de l'algorisme que farà servir la màquina. Els possibles noms son Minimax i MinimaxPodes.

Mètodes:

+ maquina (int profunditat, String algorisme) -> Constructora de la classe. Al jugador maquina se li assignaran la profunditat i l'algorisme passats com a paràmetre. El username que rebrà la maquina sera el nom de l'algorisme + la profunditat (Minimax7).

+ maquina (String nom) -> Constructora de la màquina a partir d'un string. En cas que l'string passat sigui valid (nom algorisme + profunditat) es creara una instància de la classe amb username =algorisme + profunditat i profunditat i algorisme els indicats en l'string nom.

+ int getProfunditat() -> Mètode que ens permet obtenir l'atribut profunditat de la maquina.

+ String getAlgorisme() -> Mètode que retorna el nom de l'algorisme que està fent servir aquesta maquina.

+ Point pensa_jugada(char color, Board t, mode_joc gm) -> Mètode que realitza una crida a la funció pensa de la classe algorisme passant els paràmetres pertinents. Es retorna el punt calculat per l'algorisme.

Classe ranking

Classe que ens permet representar el ranking d'un mode de joc.

Atributs:

- SortedSet <entrada_ranking> set -> Set que ens permet mantenir ordenat el ranking mitjançant la interficie comparable de la classe entrada_ranking

Mètodes:

+ void StringToSet(String s) -> Mètode que ens permet omplir un SortedSet a partir d'un String del ranking. Aquest String serà del tipus -> nomjugador:victories:derrotes:puntuacio:nomjugador:victories:derrotes:puntuacio….

+ void actualitzar_ranking(String nom, int puntuacio, boolean victoria) -> Mètode que afegeix una entrada al ranking en cas que sigui la primera entrada del jugador en el ranking, en cas que el jugador ja tingués una entrada previa s'actualitzara aquesta. Si la puntuació passada com a paràmetre es superior a la puntuació de la seva entrada del ranking se substituirà aquesta per la nova puntacio. En cas de que el boolea victoria sigui true se sumara una victoria en el còmput de victòries del jugador i en cas de ser false se sumarà una derrota.

+ String toString() -> Mètode que ens permet obtenir un string amb el contingut del ranking.

Classe entrada_ranking:

Classe que ens permet contenir les entrades del ranking.

Atributs:

+ String nom -> Nom del jugador

+ int victories -> Victòries aconseguides en el mode de joc del ranking

+ int derrotes -> Derrotes aconseguides en el mode de joc del ranking

+ int puntuacio -> Puntuació més alta obtinguda en una partida del mode de joc del ranking

Mètodes:

+ int compareTo(entrada_ranking o) -> Mètode que compara l'entrada_ranking passada com a paràmetre amb la instància de la classe. Retorna -1 en cas de que la instància de la classe sigui més gran, 1 en l'altre cas. Per comparar les dues instàncies el primer que es mira quina és la puntuació més alta, en cas d'empat es mira el nombre de victòries - el nombre de derrotes de cada instància i per últim en cas d'empat es mira el nom de les dues entrades.

Classe Board:

Classe encarregada de representar un tauler i implementar les funcions necessàries per tal de modificar-lo.

Atributs:

- char [][] tiles -> Matriu de caràcters que representa les caselles del tauler. Una "E" representa una casella buida, una "W" representa una casella ocupada per una peça blanca i una "B" representa una casella ocupada per una peça negra.

- HashSet<Point> eadj -> HashSet que ens permet mantenir guardades les caselles adjuntes a les caselles que contenen alguna peça però en les que no es pot posar cap fitxa perquè no es podrà capturar.

- HashSet<Point> badj -> HashSet que ens permet mantenir les caselles adjuntes a les caselles que contenen alguna peça en les quals es podrà posar una fitxa negra.

- HashSet<Point> wadj -> HashSet que ens permet mantenir les caselles adjuntes a les caselles que contenen alguna peça en les quals es podrà posar una fitxa blanca.

- int white -> Enter que conté la quantitat de peces blanques en el tauler

- int black -> Enter que conté la quantitat de peces negres en el tauler

Mètodes:

+ Board() -> Constructora de la classe sense paràmetres. Omple tot l'atribut tiles amb "E", és a dir construeix un tauler buit.

+ Board(mode_joc gm) -> Constructora de la classe amb el paràmetre mode_joc. Construeix un tauler amb les caselles centrals ocupades i segons el mode de joc passat com a paràmetre afegeix les caselles pertinents als atributs badj, wadj i eadj.

+ Board(String s, mode_joc gm) -> Constructora de la classe a partir d'un string. Permet generar un tauler a partir d'un string que conté el valor de les caselles. Comprova que el tauler sigui vàlid per al mode de joc passat com a paràmetre i afegeix les caselles pertinents als atributs eadj, wadj i badj.

+ Board(Board that) -> Mètode que ens permet copiar un tauler a partir del tauler passat com a paràmetre

+ int pieces(char player)-> Mètode que retorna un enter amb el nombre de peces del color passat com a paràmetre. El color pot ser "W" o "B".

- boolean is_valid(int x,int y, mode_joc gm) -> Mètode que retorna true en cas de que la posició passada com a paràmetre sigui una casella buida i tingui com a mínim una casella adjacent ocupada tenint en compte el mode de joc. Altrament retorna false.

- boolean can_capture(int x, int y, char color, int i, int j) -> Mètode que retorna true en cas que posant una peça del color "color" a la posició "x" i "y" es pugui capturar com a mínim una peça en la direcció passada fent servir els enters "i" i "j". Aquests enters seran 1, 0 o -1 i permeten representar una direcció, per exemple la direcció dreta estaria representada per x = 1 i y = 0.

- boolean can_capture_gen(int x, int y, char color, mode_joc gm)-> Mètode encarregat de realitzar les crides necessàries al mètode can_capture per la posició “x” i “y” i el color “color” segons el mode_joc passat com a paràmetre.

+ int set(int x, int y, char color, mode_joc gm) -> Mètode que permet posar una fitxa en la posició “x” i “y” del color “color” fent servir el mètode capture i retorna el nombre de fitxes capturades aplicant el mode de joc passat com a paràmetre. En cas que no es pugui capturar cap fitxa el valor retornat sera 0 i no es posarà la nova peça al tauler.

+ void setwc(int x, int y, char color) -> Mètode que s’encarrega d’afegir la peça en el tauler en la posició “x” i “y”, és a dir que afegeix una “W” o una “B” en l’atribut tiles segons el color passat com a paràmetre.

+ void delete(int x, int y) -> Mètode que s’encarrega d’eliminar una peça del tauler, és a dir posa una “E” en l’atribut tiles en la posició “x” i “y”.

- int capture(int x, int y, char color, mode_joc gm) -> Mètode que s’encarrega de realitzar els canvis pertinents en el tauler en cas de que es pugui capturar una o més fitxes col·locant una fitxa del color “color” en la posició “x” i “y”. Retorna el nombre de peces capturades en cas de poder-ne capturar i en cas que no, retorna un 0 i no col·loca la fitxa.

- modify_adj(int x, int y, mode_joc gm) -> Mètode que manté els atributs aedj, badj i wadj actualitzats després de cada modificació del tauler.

- char contains(int i,int j) -> Mètode que retorna el contigut d’una casella de l’atribut tiles. Retorna “E” en cas de que la casella estigui buida, “B” si està ocupada per una peça negra i “W” si està ocupada per una peça blanca.

+ HashSet<Point> available_positions(char color) -> Mètode que retorna un HashSet que conté totes les posicions en les que es pot posar una fitxa del color “color”. És a dir retorna els atributs badj o wadj segons el color passat com a paràmetre.

+ boolean can_play(char color) -> Mètode que retorna true en cas de que hi hagi com a mínim una casella on es pugui posar una peça del color “color”. Per poder posar una peça cal que com a mínim es pugui capturar una peça del contrincant.

+ String toString() -> Mètode que retorna un string amb el contingut del tauler.

- boolean isCorrect(mode_joc gm) -> Mètode retorna true en cas que el tauler permeti jugar utilitzant el mode de joc passat com a paràmetre, en altre cas retornarà false.

Classe mode_joc:

Classe que permet representar el mode de captura. Les captures es poden realitzar en diagonal, vertical i horitzontal o escollint una parametrització d'aquests paràmetres.

Atributs:

- boolean diagonal -> Booleà que indica si les captures es poden realitzar en diagonal o no.

- boolean horitzontal-> Booleà que indica si les captures es poden realitzar en horitzontal o no.

- boolean vertical -> Booleà que indica si les captures es poden realitzar en vertical o no.

Mètodes:

+ void setHoritzontal(boolean horitzontal) -> Mètode que ens permet modificar el paràmetre horitzontal. En cas que es vulgui posar aquest paràmetre en false i els paràmetres vertical i diagonal estiguin ja en fase no ens ho deixarà fer ja que si no obtindríem un mode de joc en el que no es podria capturar de cap manera.

+ void setVertical(boolean vertical) -> Mètode que ens permet modificar el paràmetre vertical . En cas que es vulgui posar aquest paràmetre en false i els paràmetres horitzontal i diagonal estiguin ja en false no ens ho deixarà fer ja que si no obtindríem un mode de joc en el que no es podria capturar de cap manera.

+ void setDiagonal(boolean diagonal) -> Mètode que ens permet modificar el paràmetre diagonal. En cas que es vulgui posar aquest paràmetre en false i els paràmetres horitzontal i vertical estiguin ja en false no ens ho deixarà fer ja que si no obtindríem un mode de joc en el que no es podria capturar de cap manera.

+ boolean isHoritzontal() -> Mètode que retorna l'atribut horitzontal.

+ boolean isDiagonal() -> Mètode que retorna l'atribut diagonal.

+ boolean isVertical() -> Mètode que retorna l'atribut vertical.

+ public mode_joc() -> Constructora de la classe sense paràmetres. Posa tots els atributs en true, és a dir, habilita tots els modes de captura.

+ mode_joc(boolean diagonal, boolean horitzontal, boolean vertical) -> Constructora de la classe amb paràmetres. Posa el valor indicat pels paràmetres als atributs. En cas de que tots els paramtres siguin false es fa servir la constructora per defecte i per tant es posen tots els atributs en true ja que si no obtindríem un mode de joc on cap captura estaria permesa.

+ public mode_joc (String joc) -> Constructora de la classe a partir d'un string. Crea un mode de joc a partir d'un string format per tres caràcters que poden ser 1 o 0 i que representen el valor que tenen els atributs diagonal, vertical i horitzontal.

+ public String toString() -> Mètode que transforma el mode de joc en un string per tal de poder-lo emmagatzemar en un fitxer. L'string resultant es un string de 3 caràcters on cada caràcter pot ser un 0 o un 1 i indica el valor d'un dels atributs.

Classe partida:

Classe que ens permet emmagatzemar tot els paràmetres referents a una partida. També conte tots els mètodes necessaris per poder jugar.

Atributs:

- int num_maquines -> Enter que ens diu quantes màquines juguen la partida. Aquest nombre pot ser 0 en cas de jugar humà contra humà, 1 en cas de jugar humà contra màquina o 2 en cas de jugar màquina contra màquina.

- jugador j1 -> Instància de la classe jugador que pot representar un humà o una màquina.

- jugador j2 -> Instància de la classe jugador que pot representar un humà o una màquina.

- Bord t1 -> Instància de la classe Board que permet contenir el tauler de la partida.

- mode_joc mj -> Instància de la classe mode_joc que permet contenir el mode de joc de la partida.

- boolean ordre -> Booleà que indica quin jugador juga amb les peces blanques i quin amb les peces negres. Si ordre es true vol dir que el jugador j1 juga amb les peces blanques i el jugador j2 amb les negres. En cas de que ordre sigui false el jugador j1 jugarà amb les negres i el jugador j2 amb les blanques.

Mètodes:

+ partida (int num_maquines, jugador J1, jugador J2, Board T, mode_joc m, boolean ordre) -> Constructora de la classe amb paràmetres. Omple els atributs de la instància de la classe amb els paràmetres passats.

+ partida () -> Constructora de la classe sense paràmetres. Omple els atributs de la classe amb instàncies buides de les classes dels atributs.

+ void jugar(Point a,jugador j, char color) -> Mètode que permet realitzar una tirada. En cas que el jugador passat com a paràmetre sigui un humà es realitza una crida al mètode set de la classe Board. En cas que el jugador sigui una màquina es crida al mètode pensa_jugada de la classe màquina i just després es crida al mètode set amb el punt retornat per la funció pensa jugada.

+ mode_joc getMj() -> Mètode que retorna el mode de joc de la partida

+ boolean getOrdre() -> Mètode que retorna l'atribut ordre de la partida

+ Board getBoard() -> Mètode que retorna el tauler de la partida.

+ jugador getJ1 -> Mètode que retorna el jugador j1 de la partida

+ jugador getJ2 -> Mètode que retorna el jugador j2 de la partida

+ void ini_partida () -> Mètode que s'encarrega de realitzar la primera tirada de la partida en cas de jugar humà contra màquina si la màquina juga amb les peces blanques.

+ String toString() -> Mètode que transforma la instància de partida en un string per tal de poder-lo emmagatzemar en un fitxer.

Classe algorisme:

Mètodes:

+ abstract Point pensa(int depth, char color, Board t, mode_joc gm) -> Mètode que donat uns paràmetres retorna el millor moviment calculat fent servir les funcions de Minimax o Minimax amb podes.

double heuristic(Board b, mode_joc gm) -> Mètode que retorna l'avaluació del tauler

- double stability(Board b, int x, int y, mode_joc gm) -> Mètode que retorna un double que indica l'estabilitat d'una peça en una casella.

Classe Minimax:

Classe que implementa l'algorisme minimax.

Mètodes:

+ Point pensa(int depth, char color, Board t, mode_joc gm) -> Mètode que retorna la posició del tauler calculada per minimax per col·locar la fitxa. Per calcul·lar aquesta posició es realitza una crida al mètode recursiu minimax_rec. Aquesta posició depèn del tauler, el color de la fitxa a col·locar, la profunditat que fa servir l'algorisme a l'hora de calcluar i el mode de joc que s'està fent servir en la partida.

- double minimax_rec(int depth, char color, int h, Board t, mode_joc gm) -> Mètode privat recursiu que calcula cada tauler possible fins a la profunditat màxima. En arribar al cas base retorna el valor de la heurística fins a h==0, que seleccionarà el moviment amb valor més alt si Max és true i el valor més baix si Max és false.

Classe MinimaxPodas:

Classe que implementa l'algorisme Minimax amb podes.

Mètodes:

+ Point pensa(int depth, char color, Board t, mode_joc gm) -> Mètode que retorna la posició del tauler calculada per minimax amb podes per col·locar la fitxa. Per calcul·lar aquesta posició es realitza una crida al mètode recursiu minimaxAB. Aquesta posició depèn del tauler, el color de la fitxa a col·locar, la profunditat que fa servir l'algorisme a l'hora de calcluar i el mode de joc que s'està fent servir en la partida. El funcionament de l'algorisme i els seus paràmetres s'expliquen amb més profunditat al document d'estructures de dades i algorismes.

- double minimaxAB(int depth, char color, Board b, mode_joc gm, double alpha, double beta) -> Mètode recursiu que retorna el valor òptim calculat per l'heurística donat un moviment inicial, una profunditat, un mode de joc, un tauler i un color.