Projectes de Programació

Entorn de resolució de problemes d'escacs

Q2 2018/2019

Versió del lliurament 1.0

mohamed mohamed mahmoud elkassar, omar
puigdomenech roca, jan
ribera lasa, albert

omar.mohamed.mohamed
jan.puigdomenech
albert.ribera.lasa

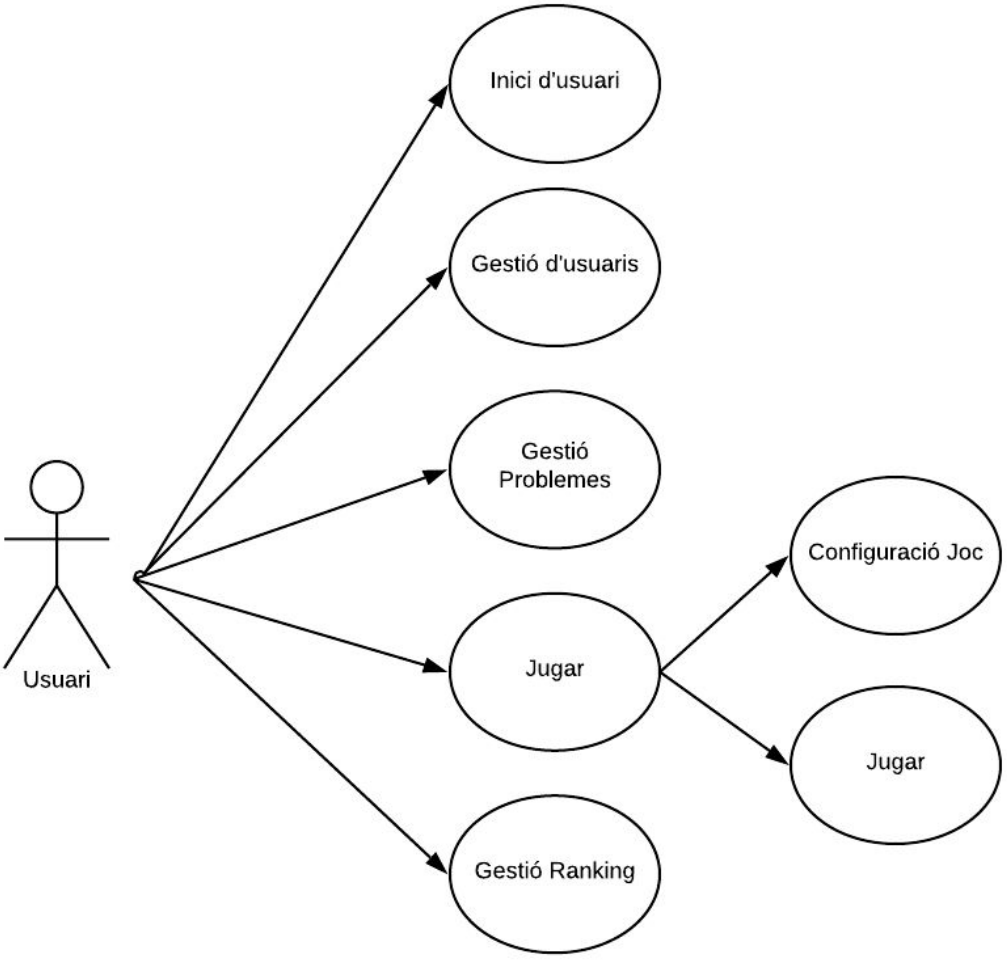
Índex

Casos d'ús 2
Diagrama 2
Descripció 3
Model conceptual 4
Diagrama 4
Descripció 5
Relació classes per membre del grup 8
Documentació de les funcionalitats principals 9

1. Casos d'ús

Diagrama de casos d'ús i breu descripció de cada cas d'ús.

1.1. Diagrama



1.2. Descripció

Inici d'usuari

En iniciar el programa l'usuari decidirà si vol registrar-se amb un nou usuari, iniciar sessió amb un usuari creat anteriorment o continuar sense registrar-se, i per tant, utilitzar l'usuari convidat.

Gestió Usuaris

L'usuari podrà crear, modificar, eliminar o consultar els usuaris existents als fitxers de dades.

Cadascun d'aquests usuaris constarà d'un nom d'usuari.

Gestió Problemes

L'usuari podrà crear, modificar, eliminar o consultar els problemes existents.

Cada problema consta de: el nom del problema, el FEN, els moviments per arribar al mat, la dificultat i el color del jugador que ataca.

Jugar

Configuració joc

Prèviament a iniciar una partida serà necessari que l'usuari introdueixi el nom del problema a resoldre i els jugadors implicats, ja siguin usuaris o la màquina.

Jugar

Una vegada configurada la partida els usuaris podran mirar de resoldre el problema indicant en cada torn quin és el moviment a realitzar. Una vegada arribats al número de torns en que s'havia de fer el mat finalitza el joc indicant s'hi s'ha resolt o no el problema, en cas afirmatiu el temps de resolució s'afegeix al rànquing.

Gestió Ranking

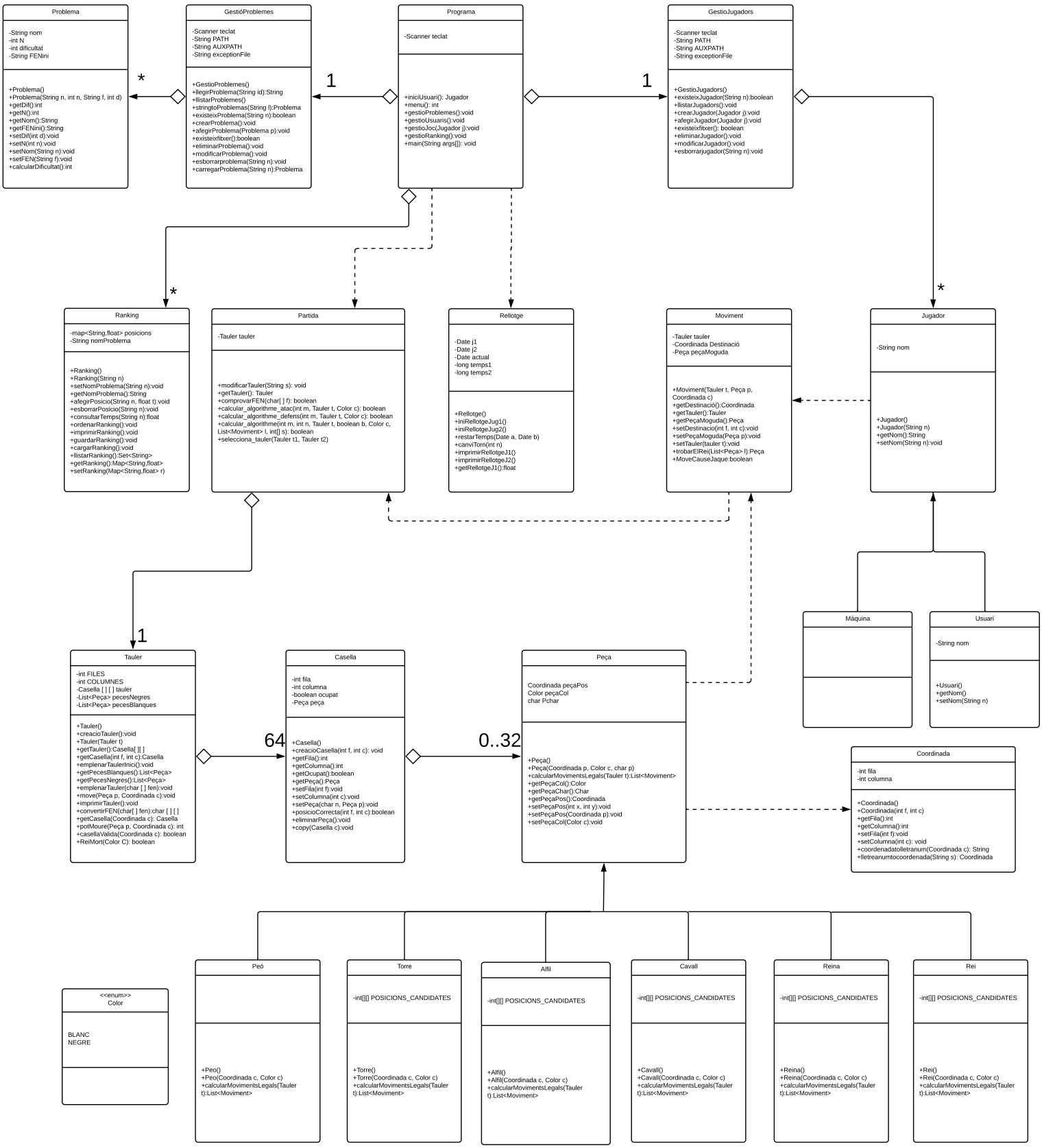
L'usuari podrà consultar el rànquing de temps per a cada problema i també eliminar alguna de les posicions si així ho desitja.

2. Model conceptual

Diagrama estàtic complet del model conceptual de dades en versió disseny i breu descripció de cada classe.

2.1. Diagrama

Representat a la següent pàgina


Problema
-String nom
-int N
-int dificultat
-String FENini
+Problema()
+Problema(String n, int n, String f, int d)
+getDif():int
+getN():int
+getNom():String
+getFENini():String
+setDif(int d):void
+setN(int n):void
+setNom(String n):void
+setFEN(String f):void
+calcularDificultat():int
Gestió Problemes
-Scanner teclat
-String PATH
-String AUXPATH
-String exceptionFile
+GestioProblemes()
+llegirProblema(String id):String
+llistarProblemes()
+stringtoProblemas(String l):Problema
+existeixProblema(String n):boolean
+crearProblema():void
+afegirProblema(Problema p):void
+existeixfitxer():boolean
+eliminarProblema():void
+modificarProblema():void
+esborrarproblema(String n):void
+carregarProblema(String n):Problema
Programa
-Scanner teclat
+iniciUsuari(): Jugador
+menu(): int
+gestioProblemes():void
+gestioUsuaris():void
+gestioJoc(Jugador j):void
+gestioRanking():void
+main(String args[]): void
GestioJugadors
-Scanner teclat
-String PATH
-String AUXPATH
-String exceptionFile
+GestioJugadors()
+existeixJugador(String n):boolean
+llistarJugadors():void
+crearJugador(Jugador j):void
+afegirJugador(Jugador j):void
+existeixfitxer(): boolean
+eliminarJugador():void
+modificarJugador():void
+esborrarjugador(String n):void
Ranking
-map<String,float> posicions
-String nomProblema
+Ranking()
+Ranking(String n)
+setNomProblema(String n):void
+getNomProblema():String
+afegirPosicio(String n, float t):void
+esborrarPosicio(String n):void
+consultarTemps(String n):float
+ordenarRanking():void
+imprimirRanking():void
+guardarRanking():void
+cargarRanking():void
+llistarRanking():Set<String>
+getRanking():Map<String,float>
+setRanking(Map<String,float> r)
Partida
-Tauler tauler
+modificarTauler(String s): void
+getTauler(): Tauler
+comprovarFEN(char[ ] f): boolean
+calcular_algorithme_atac(int m, Tauler t, Color c): boolean
+calcular_algorithme_defens(int m, Tauler t, Color c): boolean
+calcular_algorithme(int m, int n, Tauler t, boolean b, Color c, List<Moviment> l, int[] s): boolean
+selecciona_tauler(Tauler t1, Tauler t2)
Rellotge
-Date j1
-Date j2
-Date actual
-long temps1
-long temps2
+Rellotge()
+iniRellotgeJug1()
+iniRellotgeJug2()
+restarTemps(Date a, Date b)
+canviTorn(int n)
+imprimirRellotgeJ1()
+imprimirRellotgeJ2()
+getRellotgeJ1():float
Moviment
-Tauler tauler
-Coordinada Destinació
-Peça peçaMoguda
+Moviment(Tauler t, Peça p, Coordinada c)
+getDestinació():Coordinada
+getTauler():Tauler
+getPeçaMoguda():Peça
+setDestinacio(int f, int c):void
+setPeçaMoguda(Peça p):void
+setTauler(tauler t):void
+trobarElRei(List<Peça> l):Peça
+MoveCauseJaque:boolean
Jugador
-String nom
+Jugador()
+Jugador(String n)
+getNom():String
+setNom(String n):void
Màquina
Usuari
-String nom
+Usuari()
+getNom()
+setNom(String n)
Tauler
-int FILES
-int COLUMNES
-Casella [ ] [ ] tauler
-List<Peça> pecesNegres
-List<Peça> pecesBlanques
+Tauler()
+creacioTauler():void
+Tauler(Tauler t)
+getTauler():Casella[ ][ ]
+getCasella(int f, int c):Casella
+emplenarTaulerInici():void
+getPecesBlanques():List<Peça>
+getPecesNegres():List<Peça>
+emplenarTauler(char [ ] fen):void
+move(Peça p, Coordinada c):void
+imprimirTauler():void
+convertirFEN(char[ ] fen):char [ ] [ ]
+getCasella(Coordinada c): Casella
+potMoure(Peça p, Coordinada c): int
+casellaValida(Coordinada c): boolean
+ReiMort(Color C): boolean
Casella
-int fila
-int columna
-boolean ocupat
-Peça peça
+Casella()
+creacioCasella(int f, int c): void
+getFila():int
+getColumna():int
+getOcupat():boolean
+getPeça():Peça
+setFila(int f):void
+setColumna(int c):void
+setPeça(char n, Peça p):void
+posicioCorrecta(int f, int c):boolean
+eliminarPeça():void
+copy(Casella c):void
Peça
Coordinada peçaPos
Color peçaCol
char Pchar
+Peça()
+Peça(Coordinada p, Color c, char p)
+calcularMovimentsLegals(Tauler t):List<Moviment>
+getPeçaCol():Color
+getPeçaChar():Char
+getPeçaPos():Coordinada
+setPeçaPos(int x, int y):void
+setPeçaPos(Coordinada p):void
+setPeçaCol(Color c):void
Coordinada
-int fila
-int columna
+Coordinada()
+Coordinada(int f, int c)
+getFila():int
+getColumna():int
+setFila(int f):void
+setColumna(int c): void
+coordenadatolletranum(Coordinada c): String
+lletreanumtocoordenada(String s): Coordinada
<<enum>>
Color
BLANC
NEGRE
Peó
+Peo()
+Peo(Coordinada c, Color c)
+calcularMovimentsLegals(Tauler t):List<Moviment>
Torre
-int[][] POSICIONS_CANDIDATES
+Torre()
+Torre(Coordinada c, Color c)
+calcularMovimentsLegals(Tauler t):List<Moviment>
Alfil
-int[][] POSICIONS_CANDIDATES
+Alfil()
+Alfil(Coordinada c, Color c)
+calcularMovimentsLegals(Tauler t):List<Moviment>
Cavall
-int[][] POSICIONS_CANDIDATES
+Cavall()
+Cavall(Coordinada c, Color c)
+calcularMovimentsLegals(Tauler t):List<Moviment>
Reina
-int[][] POSICIONS_CANDIDATES
+Reina()
+Reina(Coordinada c, Color c)
+calcularMovimentsLegals(Tauler t):List<Moviment>
Rei
-int[][] POSICIONS_CANDIDATES
+Rei()
+Rei(Coordinada c, Color c)
+calcularMovimentsLegals(Tauler t):List<Moviment>
*
1
1
*
*
1
64
0..32

2.2. Descripció

Casella

Representa una casella concreta del tauler, amb la seva posició i s'hi indica també si esta ocupada per alguna peça i en cas afirmatiu per quina d'elles.
La podem utilitzar per comprovar si una casella del tauler és correcta.

Color

Color és un enum, que és una "classe" especial que representa un grup de constants (variables immutables, com a variables finals) en el nostre cas els colors de les peces BLANC i NEGRE. La podem utilitzar per saber de quin color és una peça o conèixer la direcció dels moviments de les peces en un tauler.

Coordinada

Es tracta simplement d'un número de fila i un numero de columna per a poder ser utilitzat en altres classes, també l'utilitzem per traduir un moviment en format FEN a numeros i a l'inrevés.

GestioJugadors

Classe per a la creació, modificació, consulta i supressió de jugadors de la nostra base de dades.

GestioProblemes

Classe per a la creació, modificació, consulta i supressió de problemes de la nostra base de dades.

Moviment

S'utilitza per canviar de posició una peça, en un tauler concret i amb una destinació. També l'utilitzem per saber si un canvi de posició causa un mat.

Partida

En aquesta classe tenim el càlcul de l'algoritme per a que la maquina pugui resoldre un problema, i la comprovació de la correctesa del FEN.

Peça

Representació de les entitats que podem moure per tal d'intentar resoldre una partida.

Subpeces

Els diferents tipus de peces (peó, torre, cavall, rei, reina, alfil) amb els seus possibles moviments.

Problema

Representació de la informació necessària per a crear una nova partida conté el nom del problema, el fen, el nombre de torns per arribar al mat i la dificultat.

Programa

Aquesta classe és l'encarregada de la interfície del programa i mostrarà a l'usuari els passos a seguir per tal de poder jugar una partida i seguir la seva resolució.

Ranking

Classe per a la creació, modificació, consulta i supressió del ranking i les diferents posicions dels jugadors per a cada problema.

Rellotge

Aquesta classe ens permetrà tenir un seguiment del temps emprat per cada jugador en les diferents jugades d'una partida.

Tauler

Representació del conjunt de caselles que formen l'espai de joc, aquesta classe la utilitzem per convertir la notació fen a un tauler o també per evaluar si el tauler esta en situació de mat.

Usuari

Classe que conté la informació d'un usuari, és a dir, el nom.

Jugador

Classe que conté la informació d'un jugador, és a dir, el nom.

Màquina

Classe que conté la informació de la maquina.

3. Relació classes per membre del grup

Relació de les classes implementades per cada membre del grup.

Classe	Membre
GestióJugadors	Jan
GestióProblemes	Jan
Ranking	Albert
Peça	Omar
Alfil	Omar
Cavall	Omar
Peó	Omar
Rei	Omar
Reina	Omar
Torre	Omar
Casella	Albert
Tauler	Omar
Color	Albert
Coordinada	Albert
Jugador	Jan
Moviment	Omar
Partida	Omar
Problema	Jan
Programa	Albert
Rellotge	Albert
Usuari	Jan
Màquina	Jan

4. Documentació de les funcionalitats principals

Breu descripció de les estructures de dades i algorismes utilitzats per a implementar les funcionalitats principals.

Paràmetres

A continuació llistarem el paràmetres utilitzats en la funcionalitat principal juntament amb una breu descripció de cadascun.

- Mate: es un enter que conté en quants torns s'ha d'aconseguir el mat per aquell problema.
- n: es un enter que utilitzarem com a contador i així poder saber quants torns ens queden per arribar al mat.
- Tauler: conte tota la informació del tauler sobre que es fa la jugada.
- Atacador: es un boolean que ens indica si el jugador M1, la màquina, està atacant o defensant.
- Defens: ens indica el color de les peces del jugador que defensa.
- SolucionG: es un vector de moviments que un cop executat l'algoritme retorna la informació (el tauler , la peça que es mou i la destinació de la peça) de cada moviment que fa la maquina.
- Stastatistiques : es un array de 2 posicions que per cada moviment retorna quants casos el resultat acaba en mat i quants no.

Algoritme

El nostre algoritme es basa en un backtracking que consta de tres parts , el cas base que és quan s’arriba al torn en que s’havia de fer el mat i es comprova si es escac i mat.

La segona part tracta de buscar com atacar, en aquest cas s’han de calcular els moviments necessaris per a que es mori el rei de l’altre color, ho farem provant tots els moviments.
I la tercera part es el cas en que la maquina estarà defensant, per tant, hem de trobar un moviment que eviti que el rei mori i sinó tornar el moviment amb menys casos en que el rei mori.

Cada crida te un temps màxim (30 segons) en cas que passi aquest temps , si estem atacant retorna l'últim estat (és a dir si s'ha trobat una solució o no) i no segueix calculant les estadístiques , en cas del defensador retorna que no hi ha solució per protegir el rei ja que no pot provar tots els moviments.