# 2ª entrega PROP

Jaume Malgosa, Calin-Constantin Pirau i Pau Charques

Maig 2019

# Índex

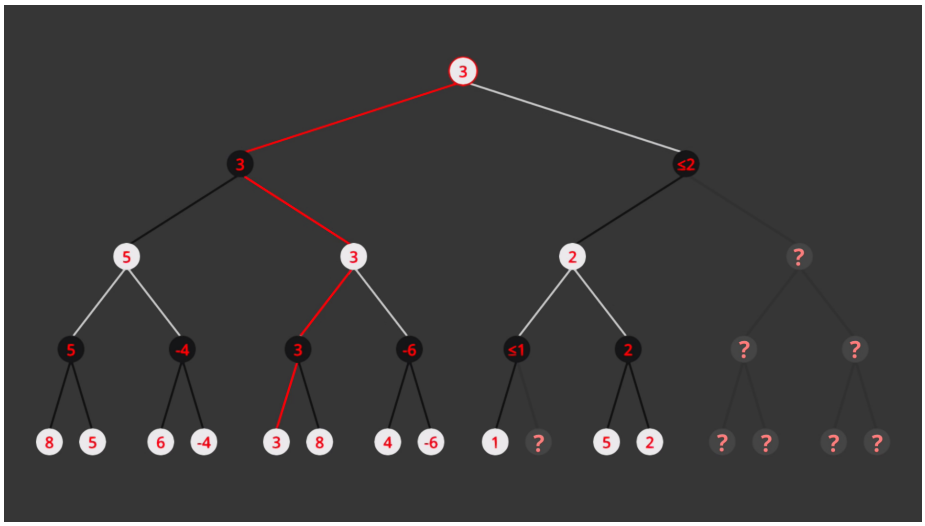
# 1 Introducció

La segona etapa del projecte ha constat en l'afegit el model de disseny en 3 capes, s'ha creat la interfície gràfica i s'ha inclòs una nova màquina, que usa minimax amb Alpha Beta Pruning. Seguidament expliquem amb més detall aquests 3 temes.

# 2 Minimax amb Alpha-Beta Pruning

Per a poder afegir-li dificultat a la màquina s'ha d'incrementar el valor de la seva profunditat, però l'algoritme fet en la primera entrega tardava massa a partir de la profunditat 3, cosa que no seria massa còmoda per l'usuari mitjà. Per tal de fer que no tardés tant, s'ha decidit utilitzar el "Alpha-Beta Pruning", que resumidament serveix per poder tallar el nombre de moviments imaginaris que es miren, cosa que milloraria molt la seva eficiència.

Aquest mètode agafa efecte a partir de la profunditat 2, per poder comparar quins valors té una part dels moviments imaginaris, i veure si és possible que surti algun valor favorable, i si no és el cas, parar la recursivitat d'aquell moviment. Amb la següent imatge es podrà explicar millor.




Totes les arestes que tenen un signe d'interrogació han sigut rebutjades. Es pot imaginar que en un cas real el nombre d'arestes rebutjades és molt major, per tant es guanya bastant en eficiència. En aquest exemple es pot observar que el fill de la dreta de l'arrel pot ser igual o més petit que 2, ja que el primer fill que té és un 2, i sempre agafarà el més petit que hi hagi. Com que l'arrel ja té assegurat un nombre major a 2, no cal seguir amb la cerca de moviments perquè no s'acceptarà cap d'aquella zona. El mateix passa amb el fill de l'aresta que

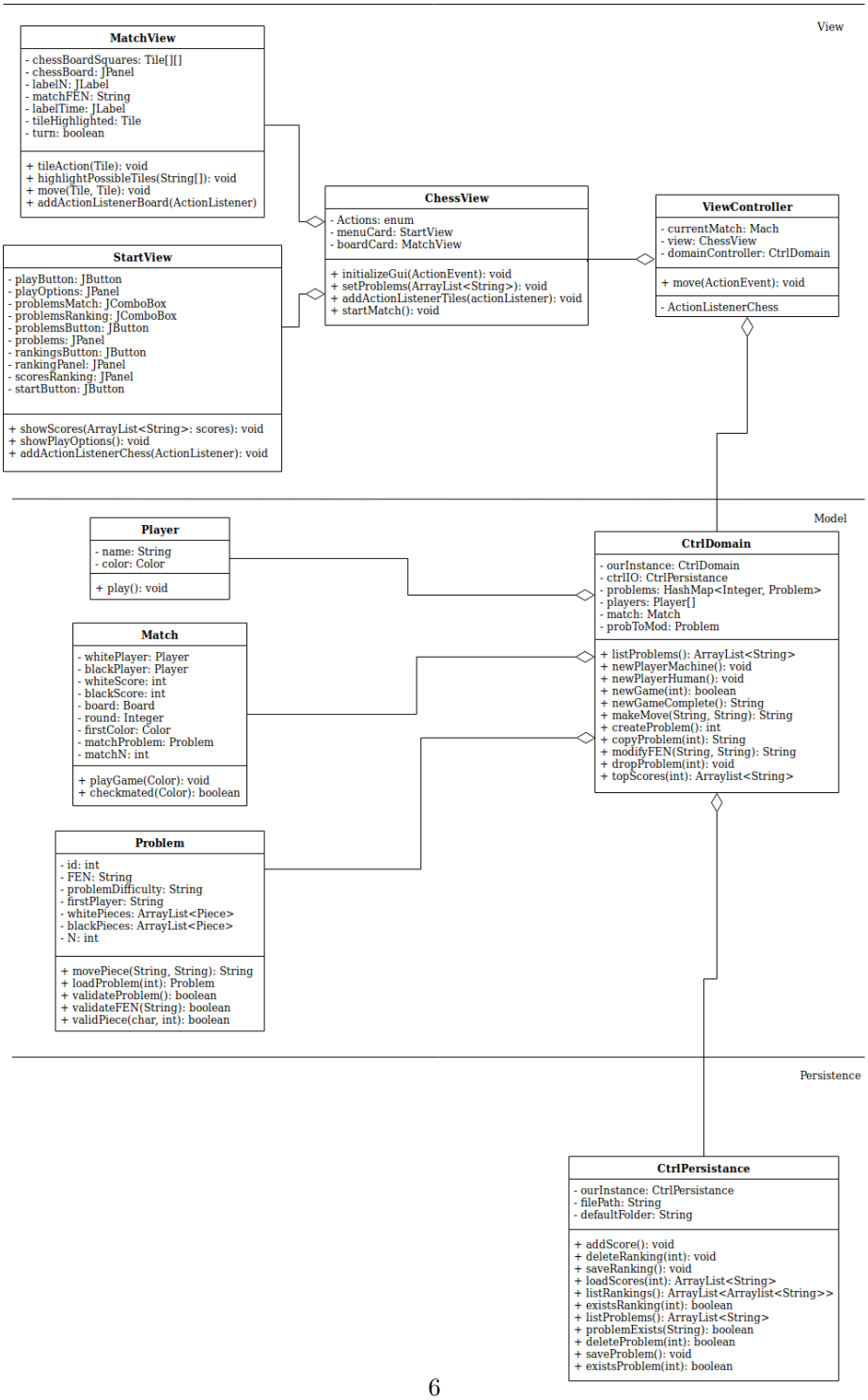
té com a valor menor igual a 1, no cal mirar quin valor té aquella aresta perquè el seu pare agafarà un valor menor o igual a 1, i la casella blanca ja té el valor 2, que és major que 1. En vermell s'ensenya quin és el camí que ha portat el número 3 a l'arrel.

Gràcies a aquest mètode, s'han aconseguit millores molt grans parlant de temps d'execució. Per decidir el mateix moviment en el mateix tauler i amb la mateixa profunditat (4), el minimax sense Alpha-Beta Pruning tarda 848 segons (14 minuts), i el minimax amb Alpha-Beta Pruning tarda 102 segons (1 minut i mig). Cal destacar que ambdós algoritmes han decidit fer el mateix moviment, o un moviment similar amb la mateixa avaluació resultant.

# 3 Model de Diseny de Tres Capes

El disseny en tres capes ha resultat, junt amb la interfície gràfica, el gran repte d'aquesta segona part del projecte, ja que ha estat el primer cop que l'aplicàvem en un projecte. El fet de separar per capes ha facilitat la comprensió del codi, ja que es separa l'accés a fitxers i la interacció amb l'usuari. Això ha permès detectar ràpidament els errors que manteníem de la primera part a més de simplificar la modificació de mètodes existents. Després d'aquesta petita introducció us ensenyem el nostre diagrama de model de 3 capes, seguit de l'explicació de tots els controladors creats a partir del diagrama que hem creat.

# 4 UML Model 3 Capes



# 5 Controladors

El sistema està dividit seguint el disseny en tres capes: Persistència, Domini i Presentació. Cada capa està formada per les classes pròpies a més del controlador pertinent, que és una classe tipus singleton, per tal que no hi hagi més d'una instància i sigui fàcilment accessible des de qualsevol classe i mantenir la coherència en tot el sistema. El disseny en capes permet assolir requisits no funcionals com la canviabilitat o la usabilitat. A continuació s'explica perquè serveix cada capa i el seu controlador i com aquest interactua amb la resta del sistema.

## 5.1 Persistència

La capa de persistència s'encarrega de mantenir la informació estàtica del sistema, així com dades que és preferible mantenir encara que no siguin imprescindibles.

En el sistema, la seva funció principal és la de mantenir els problemes. Els problemes contenen molta informació que necessita mantenir-se entre execució i execució per tal de millorar l'experiència de l'usuari i portar un registre de les puntuacions que s'han obtingut en cada problema.

Aquesta informació consta de la col·locació de les peces codificat en el format FEN a més dels torns necessaris per solucionar el problema i la seva dificultat. Tal com està construït el sistema, cada problema té assignat un ranking per tal de guarda els cinc usuaris amb millor puntuació obtinguda.

Per cada problema es genera, si el problema és solucionable, un fitxer ("P-ID.txt") on ID és un valor enter aleatori per identificar el problema. Aquest fitxer, és el que guarda el FEN, els torns i la dificultat. A més, per cada problema, si ha estat solucionat en algun moment per almenys un jugador de tipus "Humà" existirà també un fitxer ("R-ID.txt") on ID és el mateix que l'identificador del problema al qual pertany. De manera que cal tenir en compte que a l'eliminar un problema de la base de dades, cal eliminar també el seu ranking si existeix.

A l'inici de l'execució del programa, el controlador de persistència s'encarrega enviar tota la informació dels problemes a la capa de domini per tal que siguin dades útils pel joc i pot crear, modificar i eliminar fitxers per satisfer les funcionalitats del sistema a petició del controlador de Domini.

## 5.2 Domini

La capa de Domini s'encarrega de comunicar la capa de persistència, la capa de presentació i les classes que formen el joc. A l'inici de l'execució fa les crides necessàries a persistència per instanciar els problemes existents a la base de

dades i donar d'alta els rankings amb les seves scores. Durant l'execució del joc, crea players, problems i matchs segons indiqui l'usuari i converteix a variables simples objectes complexos del sistema, com ara passar un board a un string o players en enters, per evitar l'acoplament de les capes, ja que trencaria el disseny en tres capes. Aquestes conversions s'usaran per enviar informació a la capa de presentació per poder representar gràficament la partida d'escacs. Aquest controlador és més una connexió entre capes que una classe com a tal.

## 5.3 Presentació

El controlador pare és l'encarregat de crear el controlador de domini i lligar-lo a les vistes.

Aquest controlador té una única vista que conté les diferents finestres del nostre programa, d'aquesta manera amb un únic controlador podem gestionar vàries vistes i interactuar amb elles.

Aquestes vistes s'organitzen amb les classes que s'expliquen a continuació.

### 5.3.1 ChessView

És la vista que conté les altres 3 que mencionarem a continuació. S'encarrega de mostrar la vista que escollim.

### 5.3.2 StartView

Vista inicial on l'usuari escollirà quines accions volen realitzar:

- Play - Visualitzador de partida
- Problem Manager - Podràs crear, modificar, eliminar problemes de la nostra base de dades.
- Ranking - Visualitzador de rankings

### 5.3.3 MatchView

Vista encarregada de mostrar una partida. En aquesta partida podràs jugar una partida contra un altre jugador el qual habràs previament seleccionat en la pantalla StartView. Un cop hagi acabat la partida seràs notificat de si has guanyat o no. Properament seràs retornat a la pantalla inicial.

### 5.3.4 ProblemView

Vista encarregada de mostrar una finestra molt semblant a la de MatchView, però sense les restriccions de moviments, ni la comprovació d'escac i mat... Però sí que conté una matriu per seleccionar les peces que es volen afegir al problema.

- A la dreta de la finestra visualitzareu un array de peces les qual es poden seleccionar per dipostiar-les en el tauler.
- També trobareu una topbar on podreu seleccionar la dificultat del problema, les rondes i el jugador que mourà primer en la partida. Un cop introudïdes totes les dades podeu clicar al botó de Create Problem per a crear-lo, en cas de que el problema introduit sigui invàlid seràs notificat amb un missatge. de la nostra base de dades.s