Projectes de Programació

Entorn de resolució de problemes d'escacs

Q2 2018/2019

Versió del lliurament 2.0

mohamed mohamed mahmoud elkassar, omar

puigdomenech roca, jan

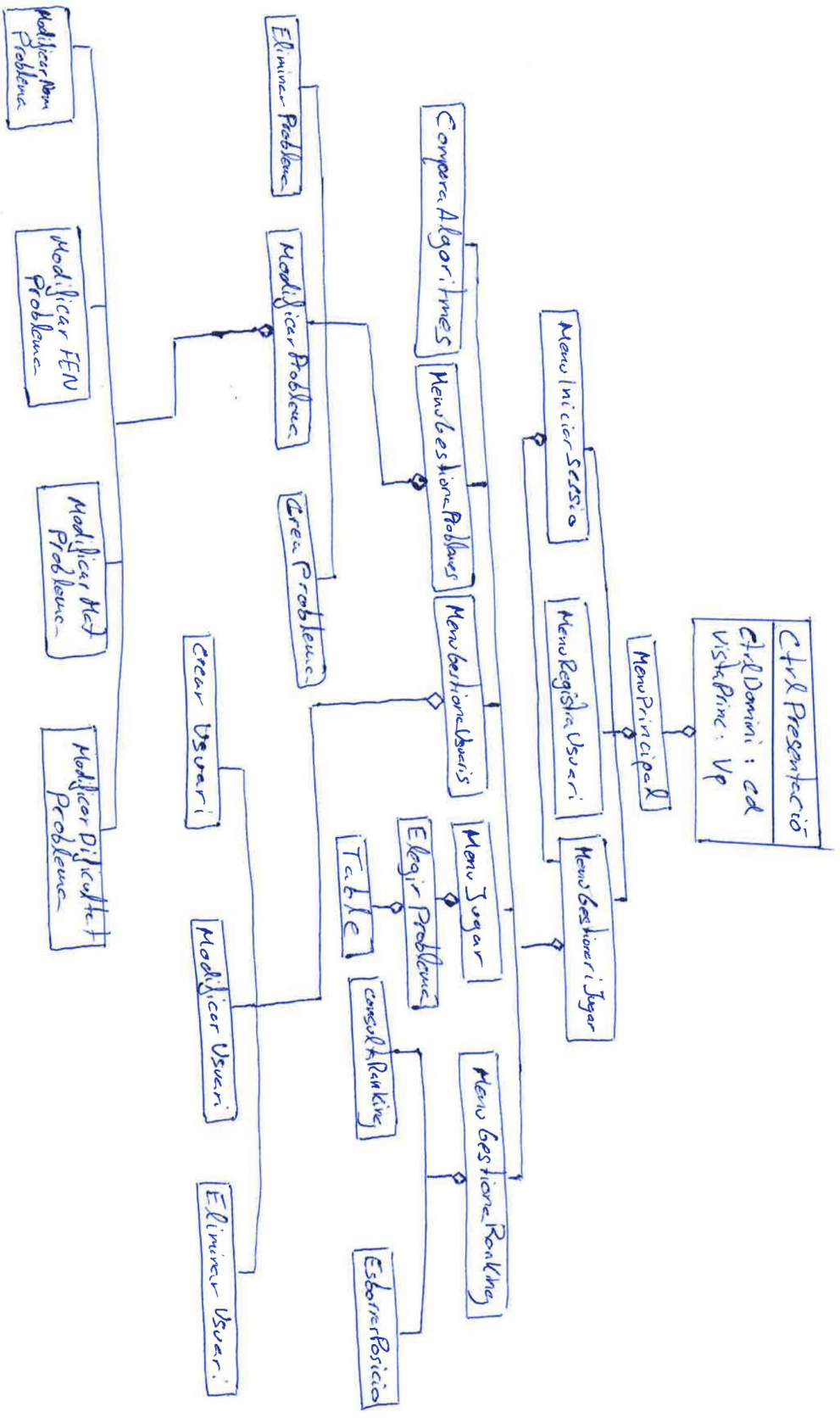
omar.mohamed.mohamed

jan.puigdomenech

Índex

1. Disseny

1.1 Disseny Capa Presentació



2. 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	Omar
Rellotge	Albert
Usuari	Jan
Màquina	Jan

Classe	Membre
ConsultaRanking	Jan
CrearProblema	Jan
CrearUsuari	Jan
EliminarProblema	Jan
EliminaUsuari	Jan
EsborraPosRanking	Jan
MenuGestionaProblemes	Jan
MenuGestionaRanking	Jan
MenuGestionariJugar	Jan
MenuGestionaUsuaris	Jan
MenuIniciarSessio	Jan
MenuPrincipal	Jan
MenuRegistraUsuari	Jan
ModificarDifProblema	Jan
ModificarFENProblema	Jan
ModificarMatProblema	Jan
ModificarNomProblema	Jan
ModificarProblema	Jan
ModificarUsuari	Jan
Table	Omar
ComparaAlgoritmes	Jan
ElegirProblema	Jan
MenuJugar	Jan

3. Millores en Estructures de Dades i Algoritmes

1)

```
public static boolean ComprovarFen (String fen);
```

Abans l'algoritme mirava que el format estigues correcte tan per les caselles que tenia cada fila com per les peçes que hi havien en el tauler. Mirava que no hi haguessin més files ni tampoc menys de 8 i que cada fila tingues 8 slots. Comprobava que hi haguessin el nombre de peçes de cada tipus i color dins del rang vàlid però no mirava si els chars entrats que representen les peces eren els correctes. Ja no admet caràcters que no formen part del format FEN.

2)

```
public static boolean calcular_algorithme (int mate , int n ,final Tauler tauler , boolean atacador ,final Color defens ,List<Moviment> solucionG, int [] stastatistiques)
```

L'algoritme calcula tots els moviments valids per la peça en el tauler. Per cada posició vàlida calcula totes les posicions vàlides i d'aquesta manera va iterant fins que arriba al nombre de jugades N per fer el mat. Un cop hi arriba mira si ha trobat una solució vàlida d'entre totes les possibilitats. Aquest algoritme ha estat millorat amb el que el segueix.

```
public static boolean calcular_algorithme_avancat (int mate , int n ,final Tauler tauler , boolean atacador ,final Color defens ,List<Moviment> solucionG, int [] stastatistiques)
```

L'algoritme avançat calcula per la posició on es troba la peça totes les possibles posicions que poden generar "Jaque Mate" i per cada moviment mira els N moviments que es poden generar a partir del defensador i despres torna a calcular els moviments que presenten "Jaque mate" i aixi rescursivament fins que trobi un moviment que per cada moviment del defensador l'algoritme pugui fer un moviment o mes en que arribi al mat.

Si la crida a alguna de les dues funciona per calcular l'algoritme tarda més d'un determinat temps aleshores s'interromp la crida.

3)

```
public static List<Moviment> Get_Move_Atacs (Tauler tauler , Color defens)
```

Aquesta funcio per un determinat tauler i per un determinat Color busca els moviments que podrien generar "Jaque Mate" en una sola jugada. Es complementa amb el calcular_algorithme_avancat.

4)

```
    public Rellotge() {
        sumaTemps1 = 0;
        sumaTemps2 = 0;
    }

public void SumR1 (long T1) {
     sumaTemps1 += T1 ;
    }
    public void SumR2 (long T2) {
     sumaTemps2 += T2 ;
    }
```

Hem afegit la funcio SumR1 i SumR2 a Rellotge que permet modificar el temps i permet afegir i substreure unitats segons els càlculs de temps que es creguin pertinents.