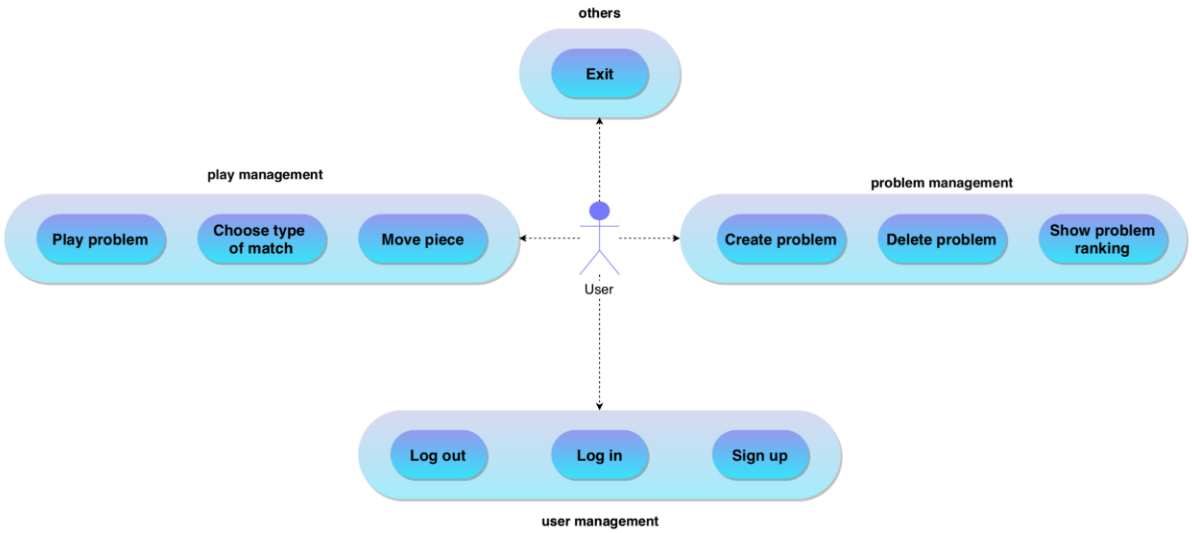
Índex documentació

(segons tutor va explicar a classe)

1. Casos d'ús
    1.1. Diagrama
    1.2. Descripció detallada
2. Model conceptual de dades
    2.1. UML
    2.2. Especificació detallada
3. Breu descripció de les EDs i algorismes utilitzats per a implementar les funcionalitats pricipals
4. Relació de les classes implementades per cada membre del grup
5. Relació de les llibreries externes utilitzades (breu descripció) ← parlat amb el tutor

1. Casos d'ús

1.1. Diagrama



1.2. Especificació detallada

Bloc: others

Exit

1. **Nom:** Exit
2. **Actors que intervenen:** Usuari
3. **Descripció:** L'usuari tanca el programa.
4. **Diàleg típic:**
    a. L'usuari decideix escollir l'opció de "Tancar".
    b. El sistema tanca el programa.
5. **Errors i vies alternatives: --**

Bloc: User management

Log out

1. **Nom:** Log out
2. **Actors que intervenen:** Usuari
3. **Descripció:** L'usuari decideix tancar la seva sessió.
4. **Diàleg típic:**
    a. L'usuari escull l'opció de "Log out".
    b. El sistema tanca la sessió i torna a la pantalla d'inici.
5. **Errors i vies alternatives: --**

Sign in

1. **Nom:** Sign up

2. **Actors que intervenen:** Usuari
3. **Descripció:** L'usuari es registra en el sistema
4. **Diàleg típic:**
    a. L'usuari escull l'opció de "Sign up".
    b. El sistema dema a l'usuari la informació necessària per completar el registre.
    c. L'usuari introdueix la informació.
    d. El sistema mostra un missatge de que el registre s'ha completat amb èxit.
5. **Errors i vies alternatives:** Si l'usuari introdueix alguna dada incorrecta, el sistema mostrarà un error.

Log in

1. **Nom:** Log in
2. **Actors que intervenen:** Usuari
3. **Descripció:** L'usuari decideix iniciar sessió amb un usuari ja existent.
4. **Diàleg típic:**
    a. L'usuari decideix escollir l'opció de "Log in".
    b. El sistema demana a l'usuari les dades necessàries per poder iniciar sessió.
    c. L'usuari introdueix les seves dades d'inici de sessió.
    d. El sistema inicia la sessió.
5. **Errors i vies alternatives:** Si l'usuari introdueix les seves dades malament el sistema mostra un missatge d'error dient quina dada es errònia i el perquè, i es torna al punt 3.

Bloc: Problem management

Create problem

1. **Nom:**Create problem
2. **Actors que intervenen:**Usuari
3. **Descripció:**L'usuari introdueix un problema a la BD.
4. **Diàleg típic:**
    a. L'usuari introdueix un problema i l'envia al sistema.
    b. El sistema comprova que el problema és vàlid (fen vàlid i solució en n jugades)
    c. El problema es carrega a la BD.
5. **Errors i vies alternatives:**Si el problema no és vàlid, es mostra un missatge d'error i es torna a la pantalla anterior.

Delete problem

1. **Nom:**Delete problem
2. **Actors que intervenen:**Usuari
3. **Descripció:**L'usuari elimina un problema creat per ell mateix de la DB.
4. **Diàleg típic:**
    a. L'usuari selecciona el problema que vol eliminar i ho envia al sistema.
    b. El sistema esborra el problema de la BD.

    c. El sistema informa a l'usuari que el problema ha estat esborrat correctament.
5. **Errors i vies alternatives: --**

Show problem ranking

1. **Nom:**Show problem ranking
2. **Actors que intervenen:**Usuari
3. **Descripció:**L'usuari consulta el ranking d'un problema
4. **Diàleg típic:**
    a. L'usuari escull la opció consultar ranking dins d'un problema en el menú propi del problema (mirar esquema de iniProgram)
    b. El sistema mostra el ranking exisent d'aquell problema
5. **Errors i vies alternatives:**Si el problema encara no té ranking, el sistema mostra un missatge d'error.

Bloc: Play management

Play problem

1. **Nom:**Play problem
2. **Actors que intervenen:**Usuari
3. **Descripció:**L'usuari comença una partida (match)
4. **Diàleg típic:**
    a. L'usuari ja té seleccionat el problema (mirar esquema de iniProgram)
    b. L'usuari continua en "choose type of match" i juga
5. **Errors i vies alternatives:**__

Choose type of match

1. **Nom:**Choose type of match
2. **Actors que intervenen:**Usuari
3. **Descripció:**L'usuari tria el tipus de la partida (match) que ha iniciat a "play problem".
4. **Diàleg típic:**
    a. Es mostren les opcions de jugada
        i. H1 vs H2
        ii. H1 vs M1
        iii. H1 vs M2
    b. L'usuari tria una opció
    c. L'usuari juga
5. **Errors i vies alternatives:**__

Move piece

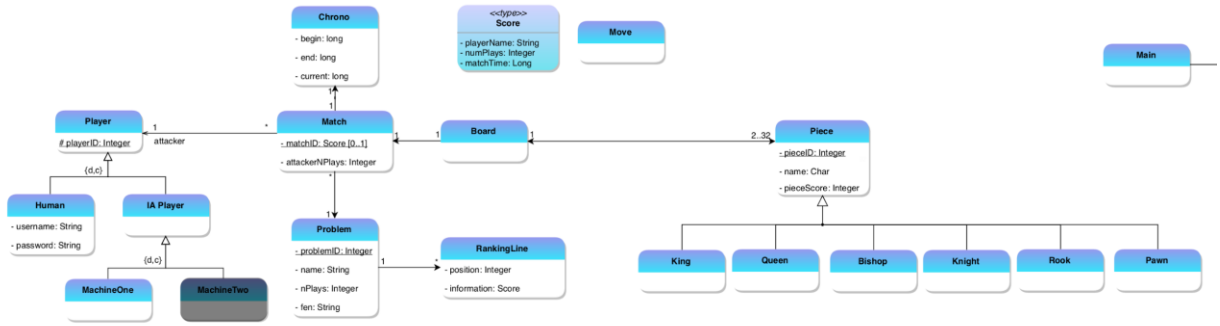
1. **Nom:**Move piece
2. **Actors que intervenen:**Usuari o Màquina
3. **Descripció:**En un problema, un usuari o una màquina mou una peça.
4. **Diàleg típic:**
    a. Usuari:
        i. Usuari (que ja està jugant partida), escull peça i escriu on la vol moure

        ii. Si es pot moure, el sistema la mou. Si no, el sistema mostra error i torna al primer punt.
    b. Màquina:
        i. Màquina escull peça i li passa al sistema el moviment.
        ii. El sistema la mou.
5. **Errors i vies alternatives:**Si l'usuari no posa una posició correcta, el sistema mostra error i torna al primer punt.

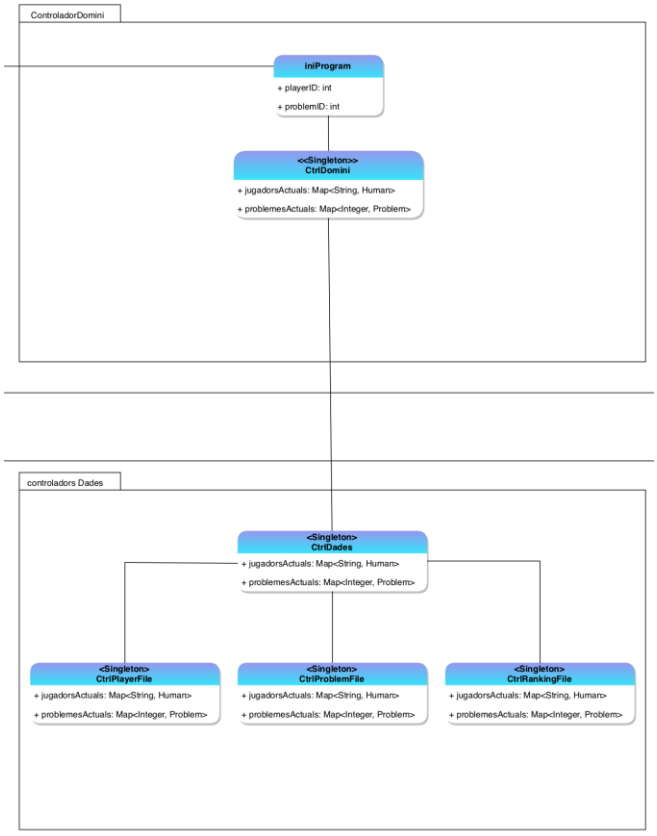
2. Model conceptual de dades

2.1. UML

- UML model (el main es connecta amb iniProgram, que fa de Capa de Presentació auxiliar).



- UML controladors



2.2. Especificació detallada

Model

PLAYER

Estructura bàsica d'un Jugador (player). Identificat per un **playerID**, que heredaran els seus fills. Funcionalitats bàsiques.

- Human: Estructura bàsica d'un Jugador Humà (human player). Identificat per un **playerID**, amb un **usuari** i una **contrasenya** per poder fer *sign up*i després *log in.*Funcionalitats bàsiques.
- IA Player: Estructura bàsica d'un Jugador amb una *IA*(machine player). Identificat per un **playerID**, que heredaran els seus fills. Funcionalitats bàsiques.
    - Machine One
      Estructura bàsica d'un Jugador amb una IA (machine player), en concret amb l'algorisme minmax. Identificat per un **playerID**. La seva funcionalitat principal és executar l'algoritme per poder guanyar una partida (Match).
    - Machine Two
      [No l'utilitzem en aquest entregable.]

MATCH

Estructura bàsica d'una Partida (Match). Identificat per un **matchID**, amb un **timer** per saber el temps que ha trigat el jugador atacant, el **nombre de jugades** que ha fet aquest, a quin **problema**pertany, i l'estat actual de la partida (**actualBoard**). Les seves funcionalitats públiques són:

> Jugar: H1 vs H2 || H1 vs M1 || M1 vs M1
> Moure peça (que s'utilitza a la funció anterior, tant un Hx com una Mx)
> Imprimir taulell **actualBoard**.

CHRONO

Estructura bàsica d'un cronòmetre. Té un temps d'inici (**begin**), temps de fi (**end**), i el càlcul del temps total que porta un usuari jugat en una partida (**current**). Això serveix per cronometrar a un usuari en un match. Si aquest guanya, s'introduirà al ranking les dades d'aquell match i tots els RankingLines s'ordenaran per aquest temps (**Match::timer**).

PROBLEM

Estructura bàsica d'un Problema (Problem). Identificat per un **problemID**, amb un **nom** per saber el tipus de problema (e.g. MateAlReiEn2) [topic], el nombre de jugades que permeten guanyar un problema (**nplays**) [difficulty], la notació **FEN** del problema, i un vector amb tot el seu **ranking**. Les seves funcionalitats públiques són:

> Validar Problema: es valida que el FEN sigui vàlid i que el problema tingui solució en nplays.
> Obtenir i afegir Ranking Lines: aquesta funcionalitat és per tal de poder interactuar

amb la BD (fitxers de text), per poder recuperar el ranking d'un problema si es tanca el programa. Bàsicament obrté o inserta RankingLines del problema.
Obtenir matriu: es transforma la notació FEN en una Board per tal de poder tractar el problema de manera més fàcil.
Qui comença: aquesta funció retorna, a partir de la notació FEN, qui começa la partida (true = blanques).

BOARD

Estructura bàsica d'un Taulell (Board). Té una **matriu** de peces (Piece) inicialitzada a null, que després s'omple amb la notació FEN del seu problema. Les seves funcionalitats públiques són:

Posar peça: posar una peça en una posició de la matriu.
Moure peça: esborrar la peça d'una posició i posar-la a una altra.
Saber si hi ha mate: a partir de saber qui comença, es mira si aquest té mate en aquella board.

RANKING LINE

Estructura bàsica d'una línia de ranking (RankingLine). Amb el **problema** al que pertany, la **posició** que ocupa en aquest, i la seva informació bàsica (**Score**). Funcionalitats bàsiques.

PIECE

Estructura bàsica d'una peça (Piece). Classe abstracta. Identificat per un **pieceID**, amb un **nom**per saber el tipus de peça (e.g. 'r' equival a black rook, 'B' equival a white bishop), el taulell en el que està posicionada (**actualBoard**) i el seu valor (**pieceScore**) per poder executar el minMax (cada tipus de peça en té un diferent). Les seves funcionalitats públiques són:

Saber color: retorna true si la peça és blanca
Comprovar moviment: abstacta - implementada per cada tipus de peça. Es comprova que el tipus de peça específic es pot moure des d'unes coordenades d'inici cap a unes cordenades destí.

- King: Té implementada la funció de comprovar moviment segons el moviment que pot fer.
- Queen: Té implementada la funció de comprovar moviment segons el moviment que pot fer.
- Bishop: Té implementada la funció de comprovar moviment segons el moviment que pot fer.
- Knight: Té implementada la funció de comprovar moviment segons el moviment que pot fer.
- Rook: Té implementada la funció de comprovar moviment segons el moviment que pot fer.
- Pawn: Té implementada la funció de comprovar moviment segons el moviment que pot fer.

SCORE

Estructura bàsica de la puntuació (Score) donada a un usuari si guanya un partit. Consisteix en el nom del jugador humà (**playerName**), el nombre de jugades (**nplays**) que ha trigat en guanyar, i el temps que ha trigat en jugar-lo (**matchTime**) en mil·lisegons. Funcionalitats bàsiques.

- Move: Estructura bàsica d'un moviment (Move). Té unes coordenades d'inici (**oldX**, **oldY**), unes coordenades destí (**newX**, **newY**), i la **peça** que es vol moure. Funcionalitats bàsiques.
- Main: Classe esàtica que s'executa i que crida a iniProgram.

Controladors

INI PROGRAM

Classe que simula la capa de presentació a nivell de terminal. Un cop és creada per el main, s'arriba als següents "menús". Té un **Humà** i un **Problema** actuals per tal de poder gestionar els estats d'aquests menús.

Initial menu: (human no definit, problem no definit)

- Sign up: es crea una fila d'usuari-contrasenya la BD.
- Log in: es crea una instància d'humà i es va al següent menú.
- Testing: permet testejar les classes.
- Exit: surt del programa.

Main menu: (human definit, problem no definit)

- Log out: permet tornar al menú anterior.
- Select problem: ensenya els problemes actuals de la BD i permet triar-ne un, es crea llavors una instància de problema i es va al següent menú.
- Create problem: es crea una fila de problema a la BD.

Problem menu: (human definit, problem definit)

- Log out: permet tornar al primer menú.
- Go back: permet tornar al menú anterior.
- Play problem: es crea una instància de Match i es va al següent menú.
- Delete problem: esborra el problema actual de la BD, i les RankingLines associades.
- Show problem ranking: ensenya les RankingLines del problema definit que hi ha a la BD.

Play menu: (human definit, problem definit)

- X vs Y: Permet tiar quina opció de joc es vol fer i es posa a jugar.
- Go back: permet tornar al menú anterior.

CTRLDOMINI
Controlador que, a partir de iniProgram i Problem, permet interactuar amb la Capa de Dades.

CTRLDADES
Controlador que, a partir de CtrlDomini, permet interactuar amb els controladors dels fitxers de la BD (següents).

- CtrlProblemFile: Controlador que llegeix i esciu en el fitxer de problems.txt. Com que la "capa de presentació" ja s'encarrega de validacions, només respon a funcions com, per exemple, showProblems(), que et mostra tots els problemes de la BD.
- CtrlPlayerFile: Controlador que llegeix i esciu en el fitxer de players.txt. Com que la "capa de presentació" ja s'encarrega de validacions, només respon a funcions com, per exemple, signUp(username, password), que insereix un usuari a la BD.
- CtrlRankingFile: Controlador que llegeix i esciu en els fitxer de rankingsX.txt {x = problemID}. Existeix un fitxer per a cada problema. Com que la "capa de presentació" ja s'encarrega de validacions, només respon a funcions com, per exemple, deleteRankingLinesOfAProblem(problemID), que esborra el fitxer corresponent.

Drivers

Per fer els drivers, hem sefuit l'estructura: Bottom-up. Segons ja s'ha parlat amb el tutor, per culpa de les navegabilitats dobles, hem hagut de suposar que algunes estaven testejades sense fer-ne stubs, com Board ↔ Piece. Tampoc hem fet driver de Piece, ja que és una classe abstracta.
El driver implementat amb JUnit és el de la classe Match.
Cadascuna de les línies separa un nivell, començant pel l'inferior.

3. Breu descripció de les EDs i algorismes utilitzats per a implementar les funcionalitats pricipals

Per a la primera entrega n'hi han dues funcionalitats principals:

1. **Hi ha dos tipus de jugadors, humans (H1 i H2) i màquines (per a aquesta entrega només M1). S'ha de poder jugar una partida en els següents escenaris:**
    - H1 vs H2
    - H1 vs M1
    - M1 vs M1

    I per tant s'ha de tenir implementat el jugador màquina.

2. **La comprovació de que un problema introduït té solució**

    Per a implementar les anteriors funcionalitats hem optat per fer servir *IntelliJ*. Per a implementar la primera funcionalitat hem fer servir l'algorisme de min-max, i per a la segona funcionalitat hem fet servir un dfs.

    Com funcionen els algorismes?

    - Min-max:

    L'algorisme agafa com a paràmetres la taula amb l'estat actual del problema, un atribut per saber de quin color és el torn i la profunditat a la qual volem arribar. Aquest acabarà retornant un moviment comprès per dues coordenades (la de la peça a moure, i la del destí), i una instància de peça (la que es mourà).

    L'algorisme el que fa és escolli, d'entre tots el possibles escenaris en n jugades vista (profunditat), quin serà el millor moviment que pot fer, suposant que el seu contrincant faci el pitjor moviment per a ell.

    Per a poder fer aquest anàlisi hem hagut de donar valors a cada peça del taulell, tenint en compte la importància i el color (signe positiu per a les blanques i negatiu per a les negres):

    Peó = 10 | Cavall = 30 | Alfil = 30 | Torre = 50 | Reina = 90 | Rei = 900

    Una vegada escollit el millor moviment possible, el programa cridarà a una funció per a comprovar si la partida es troba en posició d'escac i mat, i si es així, proclamar el guanyador de la partida.

    - Dfs:

    L'algorisme agafa com a paràmetres el número de jugades en les que s'ha de poder arribar a la situació d'escac i mat, un atribut per saber quin color comença, i un taulell amb a posició inicial del problema a validar.

L’algorisme el que fa és analitzar tots el escenaris possibles fins a torbar-ne un que acaba en escac i mat en el número de jugades indicat, o menys, independentment de que n’hi hagi més possibles escenaris.

La forma de trobar aquest escenaris és la de anar, per a cada branca de l’arbre de jugades, fins a la profunditat indicada (número de jugades en les que s’ha de poder arribar a la situació d’escac i mat).

4. Relació de les classes implementades per cada membre del grup

Ordenat per carpetes:

- Main:
  - Dades:
    - CtrlDades -> Aarón
    - CtrlPlayerFile -> Aarón
    - CtrlProblemFile -> Aarón
    - CtrlRankingFile -> Aarón
  - Domini:
    - Controladors de domini:
      - CtrlDomain -> Aarón
    - Model:
      - Bishop -> Júlia
      - Board -> Sergi
      - Chrono -> Sergi
      - Human -> Júlia
      - IAPlayer -> Júlia
      - IniProgram -> Júlia
      - King -> Júlia
      - Knight -> Júlia
      - MachineOne -> Sergi
      - Main -> Júlia
      - Match -> Sergi
      - Move -> Júlia
      - Pawn -> Júlia
      - Piece -> Sergi
      - Player -> Júlia
      - Problem -> Sergi
      - Queen -> Júlia
      - RankingLine -> Aarón
      - Rook -> Júlia
      - Score -> Aarón
- Testing:
  - Drivers:
    - JUnit:
      - MatchTest -> Sergi
    - DriverBishop -> Sergi
    - DriverBoard -> Aarón
    - DriverChrono -> Aarón
    - DriverCtrlDomain -> Aarón
    - DriverHuman -> Aarón

    - DriverIAPlayer -> Aarón
    - DriverKing -> Sergi
    - DriverKnight -> Sergi
    - DriverMachineOne -> Aarón
    - DriverMove -> Júlia
    - DriverPawn -> Sergi
    - DriverPiece -> Sergi
    - DriverPlayer -> Júlia
    - DriverProblem -> Júlia
    - DriverQueen -> Sergi
    - DriverRankingLine -> Aarón
    - DriverRook -> Sergi
    - DriverScore -> Aarón
- Stubs:
    - BoardStub -> Sergi
    - CtrlDomainStub -> Júlia
    - MatchStub -> Sergi
    - PieceStub -> Sergi
    - ProblemStub -> Júlia

5. Relació de les llibreries externes utilitzades

Llibreries per fer el driver de junit (**Match**):

- junit-4.12 (https://github.com/junit-team/junit4/blob/master/doc/ReleaseNotes4.12.md)
    - hamcrest-core-1.3 (http://hamcrest.org/JavaHamcrest/)