Master Mind

2a Entrega PROP

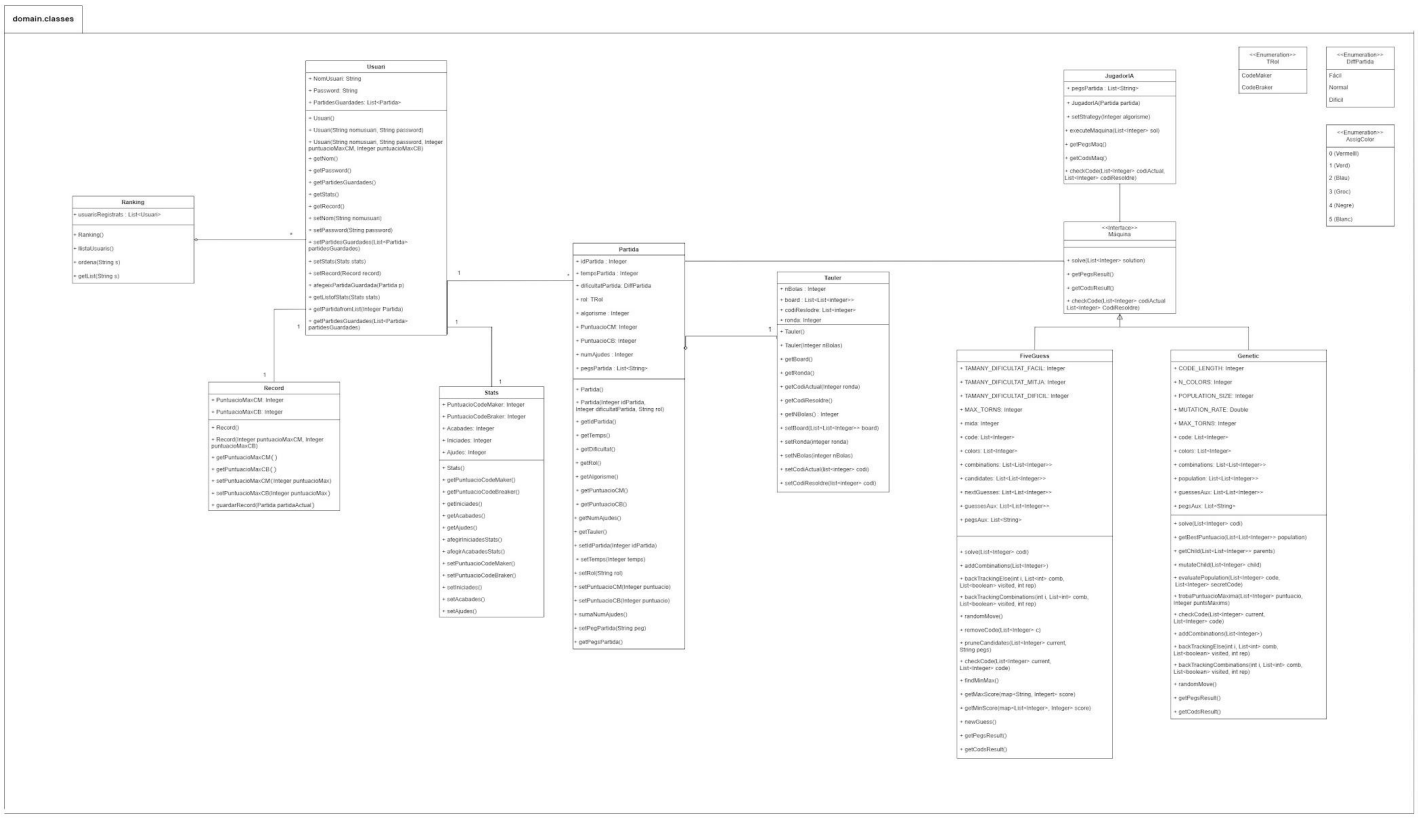
Identificador de l'equip: 12.5

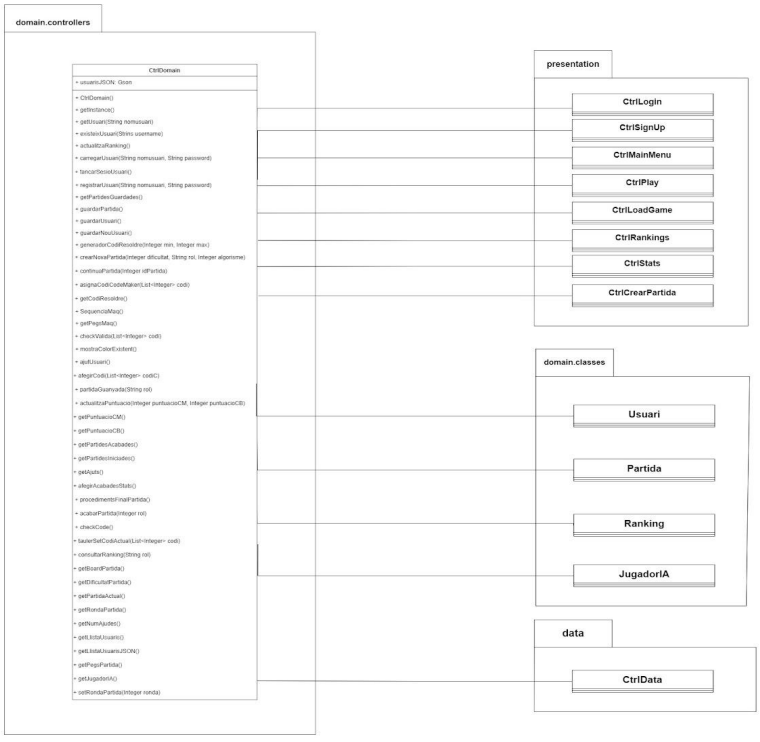
Duran López, Marc (marc.duran.lopez@estudiantat.upc.edu)
Gibert García, Miquel (miquel.gibert.garcia@estudiantat.upc.edu)
Jaume Morera, Sergi (sergi.jaume@estudiantat.upc.edu)
Puigdemont Monllor, Arnau (arnau.puigdemont@estudiantat.upc.edu)

Versió del lliurament: 1.0

ÍNDEX

1. Diagrama de la capa de domini





NOTA: Tots els diagrames es mostren al directori DOCS en format pdf.

2. Descripció de les classes del domini

Classe FiveGuess

Aquesta classe és una subclasse de la interfície Màquina i mitjançant diferents càlculs adivina el codi proposat pel jugador. Disposa de 5 Integers mida, TAMANY's utilitzat per la longitud dels codis (depèn de la dificultat de la partida) i un MAX_TORNS que defineix els torns màxims. També conté un code List<Integer> amb el que es compara cada nou intent, un colors List<Integer> utilitzat per generar totes les combinacions. Conté també tres List<List<Integer>> combinations, candidates i nextGuesses, que contenen les possibles combinacions i les combinacions possiblement correctes. Té un List<List<Integer>> guessesAux on s'emmagatzemen els codis utilitzat per adivinar el codi correcte, un List<String> pegsAux on s'emmagatzemen els feedback resultants dels codis utilitzats a guessesAux.

La conté dos getters senzilles: getCodsResult() i getPegsResult().

Altres mètodes complexos són:

- **List<List<Integer>> solve(List<Integer> codi)**: Funció cridada per la creadora a partir de la interfície per resoldre la partida en cas de ser codebreaker. Necesita un valor entre 0 i 1 per executar el codi amb colors repetits o sense.

- **List<Integer> addCombiantion(Integer[] combination):** Aquesta funció també ajuda a backtrackingCombinations i és la part en que s'afegeix el codi generat en la iteració i a un array de combinacions.

- **Void backtrackingElse(int i, Integer[] combination, boolean[] visited, int repetits):** Aquesta funció és la part de l'else de la funció backtrackingCombinations, per tant, ajuda a generar totes les combinacions possibles.

- **Void backtrackingCombinations(int i, Integer[] comb, boolean[] visited, int rep)**: Genera totes les possibles combinacions de colors sabent si hi hauran repetides i dificultat.
- **List<Integer> randomMove()**: Retorna una combinació random.

- **Void removeCode(Integer[] c)**: Elimina un codi dels arrays combinations i candidates.

- **Void pruneCandidates(Integer[] current, String pegs)**: Elimina tots els candidats amb feedback diferents a l'últim intent.

- **String checkCode(Integer[] current, Integer[] code)**: Retorna els feedback resultants de comparar l'últim intent amb el codi correcte.

- **List<List<Integer>> findMinMax()**: Retorna un array de codis amb alta possibilitat de ser el codi final.

- **int getMaxScore(Map<String, Integer> scoreCount)**: Retorna el màxim valor trobat al Map.

- **int getMinScore(Map<Integer[], Integer> score)**: Retorna el mínim valor trobat al Map.

- **List<Integer> newGuess()**: Retorna el primer valor que existeix al vector generat per la funció minmax().

Classe Genetic

Aquesta classe és la segona subclasse de la interfície Màquina i mitjançant altres càlculs també adivina el codi proposat pel jugador. Disposa de 7 Integers mida, TAMANY's utilitzat per la longitud dels codis (depèn de la dificultat de la partida), un Integer per definir la quantitat de colors, un Integer per determinar la mida de la població i un MAX_TORNS que defineix els torns màxims. També conté un double que representa el ratio de mutació, un code List<Integer> amb el que es compara

cada nou intent, un colors List<Integer> utilitzat per generar totes les combinacions. Conté també dos List<List<Integer>> combinations i population que contenen les possibles combinacions i la població actual. Té un List<List<Integer>> guessesResult on s'emmagatzemen els codis utilitzat per adivinar el codi correcte, un List<String> pegsResult on s'emmagatzemen els feedback resultants dels codis utilitzats a guessesResult.

La conté dos getters senzilles: getCodsResult() i getPegsResult().

Altres mètodes complexos són:

- **List<List<Integer>> solve(List<Integer> codi):** Funció cridada per la creadora a partir de la interfície per resoldre la partida en cas de ser codebreaker. Necesita un valor entre 0 i 1 per executar el codi amb colors repetits o sense.

- **List<Integer> getBestPuntuacio(List<List<Integer>> population):** Aquesta funció busca individu amb més puntuació i el retorna.

- **List<Integer> getChild(List<List<Integer>> parents):** Obté un fill a partir de dos codis pares que es mezclen a partir d'un valor de mezcla.

- **List<Integer> mutateChild(List<Integer> child):** A partir del fill generat a getChild, es varia lleument el fill segons un altre valor de mutació.

- **Integer evaluatePopulation(List<Integer> code, List<Integer> secretCode):** Funció que obté la puntuació d'un individu segons el codi secret a adivinar.

- **boolean trobaPuntuacioMaxima(List<Integer> puntuacio, int puntsMaxims):** Funció que busca un individu a la població amb màxima puntuació, és a dir, busca si s'ha generat el codi a adivinar.

- **String checkCode(Integer[] current, Integer[] code):** Retorna els feedback resultants de comparar l'últim intent amb el codi correcte.

- **List<integer> addCombinations(Integer[] combination):** Aquesta funció també ajuda a backtrackingCombinations i és la part en que s'afegeix el codi generat en la iteració i a un array de combinacions.

- **Void backtrackingElse(int i, Integer[] combination, boolean[] visited, int repetits):** Aquesta funció és la part de l'else de la funció backtrackingCombinations, per tant, ajuda a generar totes les combinacions possibles.

- **Void backtrackingCombinations(int i, Integer[] comb, boolean[] visited, int rep):** Genera totes les possibles combinacions de colors sabent si hi hauran repetides i dificultat.

- **List<Integer> randomMove():** Obté un codi random de les possibles combinacions

Interfície Maquina

La classe Maquina és una interfície trucada per la classe JugadorIA per definir la estrategia amb la que es resol el codi del Mastermind. Els dos algorismes a triar són FiveGuess i Genetic.

Classe Partida

Aquesta classe guarda la informació de les partides que s'estan jugant i les partides guardades. Cada partida conté un idPartida que permet identificar-les. També conté els valors de temps, dificultat, rol i ajuts, aquests són utilitzats per calcular la puntuacioCM i puntuacioCB un cop la partida ha finalitzat.
La classe crea l'objecte Tauler, aquest defineix el tauler de la partida.

Els mètodes de la classe són setters i getters.

Classe JugadorIA

Aquesta classe permet un control sobre les regles de la partida. Mira que es compleixin quan el jugador juga com a CodeMaker i permet triar l'algoritme que utilitzarà la interfície màquina per a resoldre la seqüència creada per el jugador.

Els mètodes de la classe són (obviant setters i getters):

- **Void setStrategy(Integer algorisme)**: Crea un objecte del algorisme que es vol utilitzar en la interfície Màquina segons el paràmetre enter que es passa. En cas de passar un 1 es crea el FiveGuess i es crea el Genètic en els altres casos.

- **List<List<Integer>> executeMaquina(List<Integer> sol)**: Crida a la funció solve de la interfície Maquina del algorisme triat. Retorna una llista de llistes de integers que representa el board amb les seqüències donades fins arribar a la solució passada per paràmetre.

- **List<String> getPegsMaq()**: Aquesta funció retorna una llista de string la informació del procés de adivinar la secuencia per part del algorisme.
  -B: El color es troba en la secuencia a resoldre en la posició correcte.
  -W: El color es troba en la secuencia.
  - : El color no es troba en la secuencia a resoldre.

- **List<List<Integer>> getCodsMaq()**: Aquesta funció retorna la llista de codis emmagatzemats a la classe FiveGuess o Genètic per utilitzar-los de nou al carregar partida.

- **String checkCode(List<Integer> codiActua, List<Integer> codiResoldre)**: Aquesta funció retorna una string amb la informació de la situació actual de la teva seqüència. De la mateixa manera que abans (B,W,-).

Classe Ranking

Aquesta classe s'encarrega de mostrar el record (puntuació màxima) dels usuaris registrats ordenats de millor a pitjor. Per a cada rol, es mostra el seu propi ranking i per fer-ho, la classe conté una List<Usuari> usuarisRegistrats amb tots els usuaris, que tenen un Objecte Record on tenen guardada la puntuació màxima segons el rol. El ranking es mostra retornant una matriu[n][3], on n son el número d'usuaris registrats. Per a cada fila de la matriu, es mostra el número del usuari al ranking, el seu nom i el seu record. Quan es mostra el ranking, els usuaris ja estan ordenats per puntuació.

Els mètodes de la classe són (obviant setters i getters):

- **void ordena(String s)**: Aquesta funció ordena els usuaris amb rol "s" segons la seva màxima puntuació.
- **String[][] getList(String s)**: Retorna una matriu amb la posició al ranking, el nom dels usuaris ja ordenats i amb la seva puntuació màxima de rol "s".

Classe Record

Aquesta classe s'encarrega d'emmagatzemar les puntuacions màximes de cada usuari i les guarda en dues variables Integer puntuacióMaxCM i puntuacioMaxCB, segons el rol en el que juga.

Els mètodes de la classe són (obviant setters i getters):

- **void guardarRecord(Partida partidaActual)**: La funció rep la partida per accedir a la puntuació i actualitzar-la al record, si s'escau. Es té en compte el rol de la partida a l'hora d'actualitzar el record.

Classe Stats

Aquesta classe guarda les estadístiques totals de cada usuari. Les estadístiques que s'emmagatzemen són: la puntuació total com a codemaker, la puntuació total com a codebreaker, la quantitat de partides començades i acabades i la quantitat d'ajuts utilitzats.

Els mètodes de la classe són setters i getters.

Classe Tauler

Aquesta classe s'encarrega de crear el tauler de la partida segons el valor nBoles que se li passi a la creadora (nBoles s'entén com els diferents espais que hi ha en una fila del tauler). Aquest conté informació sobre l'estat actual del joc, com és el número de rondes que s'han jugat, la combinació secreta que ha de ser adivinada i els intents de combinació del jugador per a adivinar la secreta.

Els mètodes de la classe són setters i getters.

Classe Usuari

Aquesta classe s'encarrega de crear un usuari que s'identifica per un nom i té una contrasenya. L'usuari conté una llista amb les partides que aquest ha guardat, les seves estadístiques i el seu record.

Els mètodes de la classe són (obviant setters i getters):

- **void afegeixPartidaGuardada(Partida p)**: Afegeix la partida "p" a la llista de partides guardades de l'usuari.

- **List<Integer> getListofStats(Stats stats)**: Retorna una llista d'enters amb les estadístiques "stats" de l'usuari

- **Partida getPartidafromList(int partida)**: Busca la partida amb identificador "partida" a la llista de partides guardades. Si la troba la retorna i la elimina de la llista, si no, no retorna res.

- **List<Integer>getPartidesGuardades(List<Partida>partidesGuardades)**: Retorna la llista de partides guardades que té l'usuari emmagatzemades.

Classe CtrlDomain

Aquesta classe s'encarrega de la comunicació amb les classes Usuari, Partida, Ranking i JugadorIA. Crea un JSON llistaUsuaris que conté els usuaris registrats al sistema.

Els mètodes de la classe són (obviant setters i getters):

- **JsonReader getUsuari(String nomusuari)**: Busca a la llista d'usuaris del sistema l'usuari corresponent al nom d'usuari passat per paràmetre i el retorna en format JSON.

- **Boolean existeixUsuari(String nomusuari)**: Funció que crida al controlador de persistència i aquest retorna true si el nom nomusuari es troba en la llista de jugadors registrats. Retorna false en cas contrari.

- **Void ActualitzaRanking()**: Funció que actualitza el objecte ranking ja creat.

- **Void carregarUsuari(String nomusuari, String password)**: Funció que permet carregar l'usuari (inicia sessió).

- **Void tancarSessioUsuari()**: Funció que permet a l'usuari tancar sessió.

- **Void registrarUsuari(String nomusuari, String password)**: Funció que permet a l'usuari registrarse.

- **Void guardarPartida()**:Guarda la partida.

- **Void guardarUsuari()**: Guarda l'usuari.

- **Void guardarNouUsuari()**: Crea y guarda un nou usuari.

- **List<Integer> generadorCodiResoldre(Integer min, Integer max)**: Funció que permet generar la seqüència a resoldre pel CodeBreaker de manera random.

- **Void creaNovaPartida(Integer dificultat, String rol)**: Funció que permet crear una nova partida.

- **Void continuaPartida(Integer idPartida)**: continua la partida amb idPartida.

- **Boolean checkValida()**: Funció que permet comprovar que la seqüència introduïda utilitza colors vàlids.
- **Integer mostraColorExistent()**:Mostra un color de la seqüència a resoldre que no te la seqüència introduïda pel CodeBreaker.

- **Void ajutUsuari()**: Crida a mostraColorExistent() i fa comprovacions.

- **Boolean partidaGuanyada(String rol)**: Comprova si el CodeBreaker a resolt la seqüència proposada pel CodeMaker .

- **Void actualitzaPuntuacio(Integer puntuacioCM, Integer puntuacioCB)**: Funció que actualitza la puntuació del CodeMaker i CodeBreaker de la partida actual del sistema.

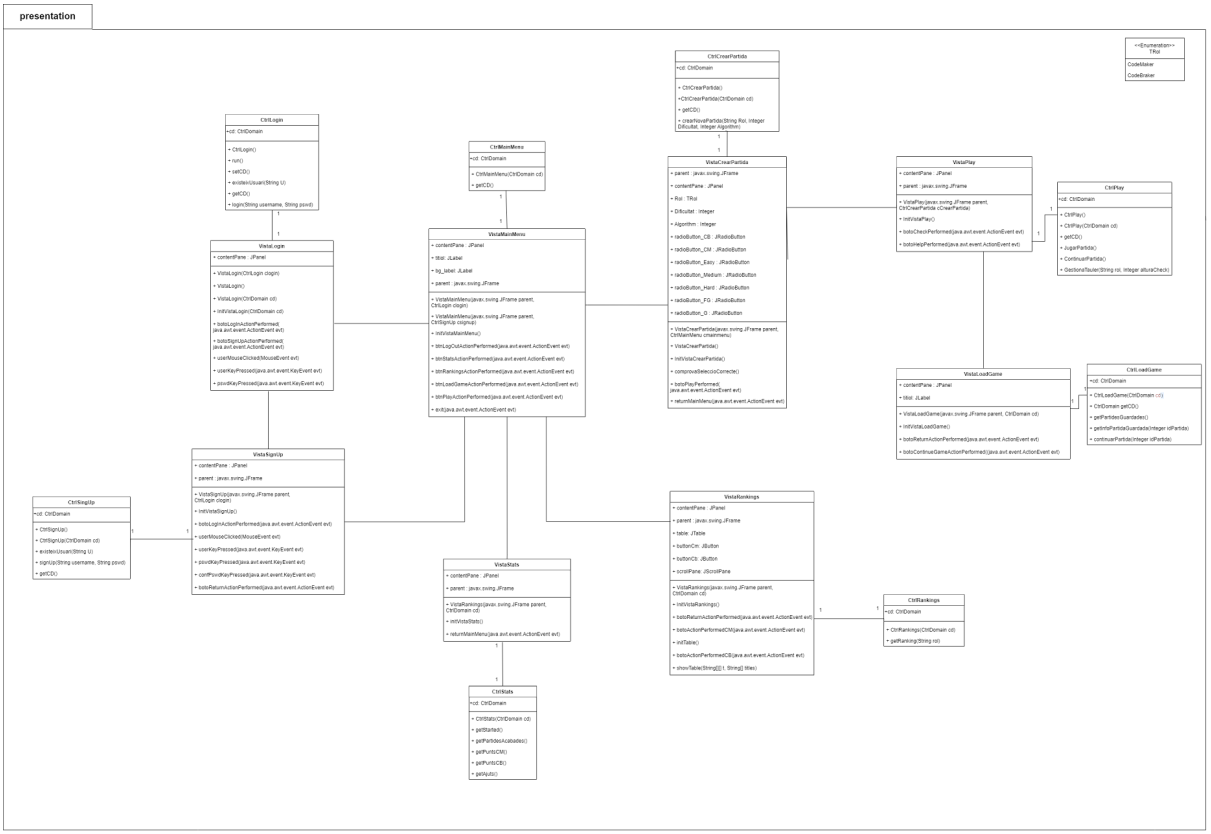
- **Void procedimentsFinalPartida()**: Funció que borra de les partides guardades la partida actual si hi està i actualitza stats.

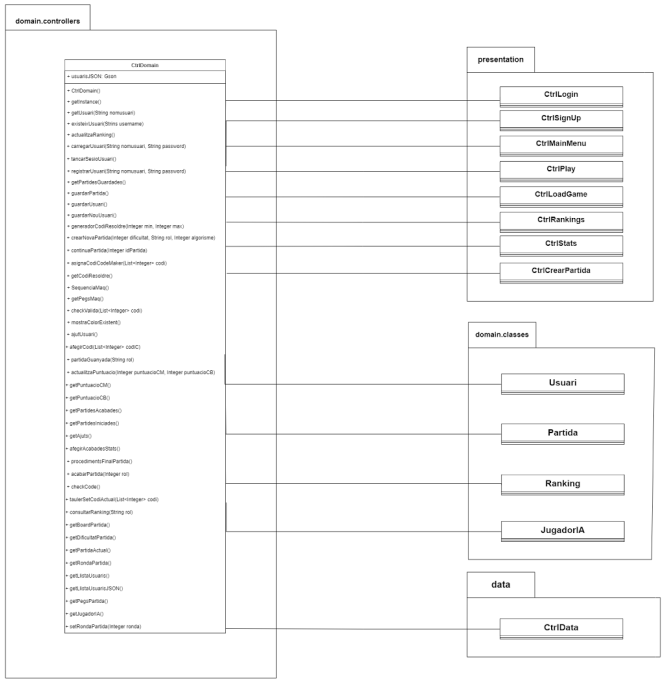
- **Void acabarPartida(Integer rol)**: Acaba la partida actual.

- **String checkCode()**: funció que retorna els feedback després de comparar el codi que l'usuari dona amb el codi correcte.

- **String[][] ConsultarRanking(String rol)**: Funció que mostra el ranking segons el rol.

3. Diagrama de la capa de presentació





4. Descripció de les vistes de la presentació

Vista Login

Aquesta vista mostra el primer que es veu quan s'inicia el joc. La seva funció és mostrar a l'usuari una pantalla per a poder iniciar sessió o mostrar la vista SignUp en cas d'interactuar amb el botó de crear un compte si és que no està registrat.

Els mètodes rellevants de la classe són:

- **Void InitVistaLogin()**: Funció que crea la interfície gràfica de la finestra. El més rellevant d'aquesta funció són el JTextField i JPasswordField dels que disposa per a poder iniciar sessió amb el username i la contrasenya, respectivament. També hi ha un JButton per a crear un compte si és que l'usuari no està registrat i un JLabel per a mostrar missatges d'error.

- **Void botoLogInActionPerformed(java.awt.event.ActionEvent evt):** Funció que comprova que els camps introduïts d'usuari i contrasenya siguin correctes. En cas que aquesta situació es doni, es crea i mostra la vista MainMenu i en cas contrari, es mostra un missatge d'error corresponent amb el tipus d'error comès.

- **Void botoSignUpActionPerformed(java.awt.event.ActionEvent evt):** Funció que crea i mostra la vista SignUp en cas de prèmer el botó de crear un compte.

Vista SignUp

La funció d'aquesta vista és mostrar a l'usuari una pantalla per a poder registrar-se. El registre consta de la definició d'un username i d'un password que per seguretat, s'ha de repetir.

Els mètodes rellevants de la classe són:

- **Void InitVistaSignUp()**: Funció que crea la interfície gràfica de la finestra. El més rellevant d'aquesta funció són el JTextField i els dos JPasswordField dels que

disposa per a poder dur a terme el registre. Al igual que amb la vista Login, també hi ha un JLabel que mostra els missatges d'error. Per últim, disposa d'un botó JButton per a tornar a la vista Login.

- **Void botoLogInActionPerformed(java.awt.event.ActionEvent evt):** Funció que comprova que el camp d'usuari i els camps de contrasenya siguin correctes. En cas que aquesta situació es doni, es crea i mostra la vista MainMenu i en cas contrari, es mostra un missatge d'error corresponent amb el tipus d'error comès.

- **Void botoReturnActionPerformed(java.awt.event.ActionEvent evt):** Funció que mostra la vista "parent", en aquest cas la vista Login.

Vista MainMenu

Aquesta vista mostra el menú principal del joc un cop l'usuari ha iniciat sessió. La seva funció és mostrar a l'usuari una pantalla amb les diferents interaccions que aquest pot dur a terme (detallades en la explicació dels mètodes).

Els mètodes rellevants de la classe són:

- **Void InitVistaMainMenu():** Funció que crea la interfície gràfica de la finestra. El més rellevant d'aquesta funció són els botons JButton dels que disposa per a que l'usuari pugui crear una partida, carregar una partida, consultar el ranking, consultar les estadístiques, tancar sessió i tancar l'aplicació.

- **Void btnPlayActionPerformed(java.awt.event.ActionEvent evt):** Funció que crea i mostra la vista CrearPartida.

- **Void btnLoadGameActionPerformed(java.awt.event.ActionEvent evt):** Funció que crea i mostra la vista LoadGame.

- **Void btnRankingsActionPerformed(java.awt.event.ActionEvent evt):** Funció que crea i mostra la vista Rankings.

- **Void btnStatsActionPerformed(java.awt.event.ActionEvent evt):** Funció que crea i mostra la vista Stats.

- **Void btnLogOutActionPerformed(java.awt.event.ActionEvent evt):** Funció que crea i mostra la vista Login.

- **Void exit(java.awt.event.ActionEvent evt):** Funció que acaba l'execució del programa.

Vista Rankigs

Aquesta vista mostra el ranking global de punts del joc i hi ha un per a cada rol. La seva funció és mostrar a l'usuari una pantalla per a que aquest consulti el ranking CodeMaker o el ranking CodeBreaker. Els rankings mostren la posició de l'usuari en aquest, el nom d'usuari i la seva puntuació màxima en el rol seleccionat.

Els mètodes rellevants de la classe són:

- **Void InitVistaRankings():** Funció que crea la interfície gràfica de la finestra. El més rellevant d'aquesta funció són els dos botons JButton per a triar quin ranking consultar i una JTable per a mostrar-lo. Aquesta es mostra dins d'un JScrollPane.

- **Void botoReturnActionPerformed(java.awt.event.ActionEvent evt):** Funció que mostra la vista "parent", en aquest cas la vista MainMenu.

- **Void btnActionPerformedCM(java.awt.event.ActionEvent evt):** Funció que consulta i mostra el ranking CodeMaker quan l'usuari interacciona amb el botó de CodeMaker.

- **Void btnActionPerformedCB(java.awt.event.ActionEvent evt):** Funció que consulta i mostra el ranking CodeBreaker quan l'usuari interacciona amb el botó de CodeBreaker.

- **Void initTable():** Funció que mostra el ranking CodeMaker un cop creada i mostrada la vista.

- **Void showTable(String[][] t, String[] titles):** Funció que mostra el ranking “t” amb els títols “titles” a les columnes amb el format escollit pels membres del grup. “t” pot ser el ranking CodeMaker o el ranking CodeBreaker.

Vista Stats

Aquesta vista mostra les estadístiques de l’usuari. La seva funció és mostrar a l’usuari una pantalla per a poder consultar les partides que ha iniciat, les que ha acabat, la puntuació total com a CodeMaker, la puntuació total com a CodeBreaker i les ajudes demanades.

Els mètodes rellevants de la classe són:

- **Void iniVistaStats()**: Funció que crea la interfície gràfica de la finestra. El més rellevant d’aquesta funció són els cinc JLabel dels que disposa per a poder actualitzar els valors de cada camp mitjançant la crida a getters del controlador.

- **Void returnMainMenu(java.awt.event.ActionEvent evt):** Funció que mostra la vista “parent”, en aquest cas la vista MainMenu.

Vista LoadGame

Aquesta vista mostra les partides guardades que té l’usuari. La seva funció és mostrar a l’usuari una pantalla per a poder consultar les partides guardades i carregar la que aquest triï. La informació que es mostra de cada partida guardada és el seu identificador, el rol de l’usuari en la partida i el número de ronda on aquest es trobava.

Els mètodes rellevants de la classe són:

- **Void InitVistaLoadGame()**: Funció que crea la interfície gràfica de la finestra. El més rellevant d’aquesta funció són el botó JButton per a continuar la partida, que

es troba seleccionada en la llista JList que conté la informació de les partides guardades i el botó per a tornar a la vista MainMenu.

- **Void botoReturnActionPerformed(java.awt.event.ActionEvent evt):** Funció que mostra la vista “parent”, en aquest cas la vista MainMenu.

- **Void botoContinueGameActionPerformed((java.awt.event.ActionEvent evt):** Funció que al interactuar amb el botó JButton de continuar la partida, continúa la partida seleccionada en la llista JList.

Vista CrearPartida

Aquesta vista mostra el menú de creació de partida del joc. La seva funció és mostrar a l’usuari una pantalla per a poder triar el rol en la partida, la dificultat del tauler i l’algoritme que utilitzarà la interfície màquina per a resoldre la seqüència.

Els mètodes rellevants de la classe són:

- **Void InitVistaCrearPartida()**: Funció que crea la interfície gràfica de la finestra. El més rellevant d’aquesta funció són els radio button JRadioButton per a seleccionar la configuració de partida que es vol crear. Disposa de dos per a triar el rol, tres per a triar la dificultat i dos per a triar l’algorisme. Hi ha un label JLabel per a mostrar missatges d’error i un botó JButton per a tornar a la vista MainMenu

- **Void botoPlayPerformed(java.awt.event.ActionEvent evt):** Funció que crea una partida amb la selecció de l’usuari i crea i mostra la vista Play.

- **Boolean comprovaSeleccioCorrecte():** Funció que comprova que la selecció dels elements radio button (JRadioButton) de la partida s’han dut a terme correctament.

- **Void returnMainMenu(java.awt.event.ActionEvent evt):** Funció que mostra la vista “parent”, en aquest cas la vista MainMenu.

Vista Play

Aquesta vista mostra la interfície creada per a poder jugar. La seva funció és mostrar a l'usuari una pantalla amb els colors disponibles, un tauler amb deu files de n "forats", on n canvia per cada dificultat i per últim, al costat de cada fila i fora del tauler, n "forats" més per a fer les comprovacions mitjançant un botó.
Finalment, disposa d'un altre botó per a demanar ajuda.

Els mètodes rellevants de la classe són:

- **Void InitVistaPlay(String dificulty, String rol, Integer algorithm)**: Funció que crea la interfície gràfica de la finestra. El més rellevant d'aquesta funció són els botons JButton per a representar cada color i per a "crear" el tauler de n "forats", juntament amb el botó Check i el botó per a demanar ajuda. Segons el paràmetre dificulty, es crearà un tauler o un altre, amb més o menys n "forats".

- **Void botoCheckPerformed(java.awt.event.ActionEvent evt):** Funció que comprova els botons JButton pegs o la seqüència introduïda a cada ronda (depenent del rol de l'usuari en la partida) i ho compara amb els pegs correctes o la seqüència correcte, respectivament.

- **Void botoHelpPerformed(java.awt.event.ActionEvent evt):** Funció que es mostra en cas de que l'usuari jugui com a CodeBreaker. Omple d'un color un botó JButton del "tauler" de la seqüència correcte.

Cada vista té associada un controlador que s'encarrega de la comunicació amb el controlador de domini per a poder fer crides dels mètodes que més ens interessen. A més a més, els controladors de les vistes que creen i mostren altres vistes tenen un mètode anomenat getCD() que retorna el controlador de domini, per no crear així un nou cada vegada que es crea la vista corresponent.
Cada mètode del controlador de domini es troba explicat en l'apartat 2 d'aquest document.

Classe CtrlLogin:

Els mètodes rellevants de la classe són:

- **Void run():** Funció que crea i mostra la vista Login (recordem que és la primera que es veu en executar el programa).

- **Boolean existeixUsuari(String U):** Funció que crida a la funció existeixUsuari(U) del controlador de domini.

- **Void login(String username, String pswd):** Funció que crida a la funció carregarUsuari(username, pswd) del controlador de domini.

Classe CtrlSignUp:

Els mètodes rellevants de la classe són:

- **Boolean existeixUsuari(String U):** Funció que crida a la funció existeixUsuari(U) del controlador de domini.

- **Void signUp(String username, String pswd):** Funció que crida a la funció registrarUsuari(username, pswd) del controlador de domini.

Classe CtrlMainMenu:

L'únic mètode de la classe és getCD().

Classe CtrlRankings:

Els mètodes rellevants de la classe són:

- **String[][] getRanking(String rol):** Funció que crida a la funció consultarRanking(rol) del controlador de domini.

Classe CtrlStats:

Els mètodes rellevants de la classe són:

- **Integer getStarted():** Funció que crida a la funció getPartidesIniciades() del controlador de domini.

- **Integer getPartidesAcabades():** Funció que crida a la funció getPartidesAcabades() del controlador de domini.

- **Integer getPuntsCM():** Funció que crida a la funció getPuntuacioCM() del controlador de domini.

- **Integer getPuntsCB():** Funció que crida a la funció getPuntuacioCB() del controlador de domini.

- **Integer getAjuts():** Funció que crida a la funció getAjuts() del controlador de domini.

Classe CtrlLoadGame:

Els mètodes rellevants de la classe són:

- **List<Partida> getPartidesGuardades():** Funció que crida a la funció getPartidesGuardades() del controlador de domini.

- **String[] getInfoPartidaGuardada(Integer idPartida):** Funció que guarda en un vector la informació de la partida guardada. Crida a les funcions getIdPartida(), getRolPartida() i getRondesPartida() del controlador de domini.

- **Void continuarPartida(Integer idPartida)**: Funció que crida a la funció continuaPartida() del controlador de domini.

Classe CtrlCrearPartida:

Els mètodes rellevants de la classe són:

- **Void crearNovaPartida(String Rol, Integer Dificultat, Integer Algorithm):** Funció que crida a la funció creaNovaPartida(Dificultat, Rol, Algorithm) del controlador de domini.

Classe CtrlPlay:

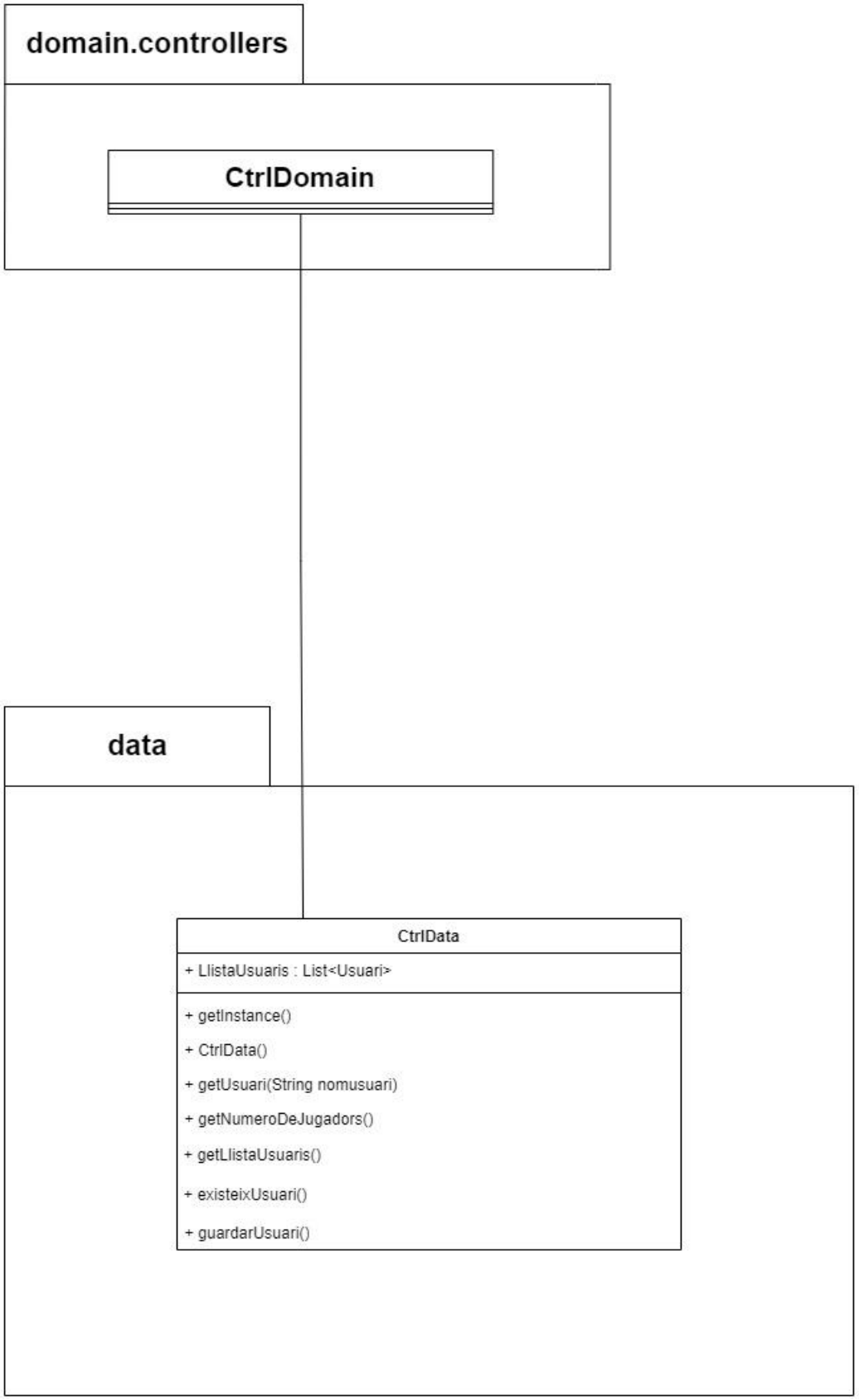
Els mètodes rellevants de la classe són:

- **Void JugarPartida():** Funció que gestiona el funcionament correcte de la partida, usant els paràmetres de la funció que crea la partida de la vista CrearPartida. També s'encarrega d'actualitzar les estadístiques de l'usuari i del sistema de puntuació. Es dur a terme mitjançant les crides pertinents al controlador de domini.

- **Void ContinuarPartida():** Funció que fa el mateix que l'anterior però es crida quan es ve desde la vista LoadGame.

- **Void GestionaTauler(String rol, Integer alturaCheck):** Funció que gestiona l'estat dels botons JButton segons el rol de l'usuari en la partida i la ronda en la que aquest es troba.

5. Diagrama de la capa de persistència



6. Descripció dels gestors de disc

Classe CtrlData

Aquesta classe s'encarrega de la comunicació amb el controlador de domini. El que permet aquesta classe és llegir, en el nostre cas, fitxers JSON, que guardarem al disc com a fitxers.
Gràcies a això, podem tenir usuaris registrats en el disc, és a dir, que quan s'allibera la memòria del programa aquests usuaris queden guardats i es poden carregar quan es torna a iniciar el programa.
Per últim, mitjançant el controlador de domini, podem gestionar, carregar i actualitzar els valors d'aquests usuaris.

Els mètodes de la classe són (obviant setters i getters):

- **getUsuari(String nomusuari):** Funció que retorna en format JSON l'usuari amb nom nomusuari del disc.

- **getNumeroDeJugadors():** Funció que retorna el número d'usuaris guardats en el disc.

- **existeixUsuari(String nomusuari):** Funció que comprova que el fitxer usuari amb nom nomusuari existeix al disc.

- **guardarUsuari(String nomusuari, String usuariJson):** Funció que crea un fitxer JSON amb nom nomusuari en cas de que no existeixi. En cas que si, actualitza el fitxer JSON.

7. Estructures de dades i algorismes utilitzats

Per a implementar les funcionalitats principals del projecte, hem decidit treballar amb les següents estructures de dades:

- **Usuari:**

  Com ja hem comentat, un usuari ha de poder guardar partides i més endavant carregar-les. Per això, hem decidit utilitzar una llista (List) de tipus Partida ja que la interfície "List" ens permet representar una col·lecció ordenada d'elements.

  D'aquesta manera, l'hem implementat per a poder controlar la posició on inserim cada element de la llista i a més a més, accedir a aquests mitjançant l'índex que ocupa dins d'ella. El seu cost és O (1)**.**

- **Tauler:**

  Per a implementar el tauler, hem decidit representar-lo amb una llista de llistes d'enters. List<List<Integer>>, ja que igual que en el cas d'usuari, hem implementat la interfície List de Java per a que ens permeti accedir als índex en tot moment de la llista. El cost d'aquesta implementació és O (n*m).

- **Ranking:**

  El ranking, per a mostrar-lo, ens hem decantat per a fer servir una matriu de n*3 que realitza operacions d'ordre constant, on n són el número d'usuaris registrats al sistema. El seu cost és O(n log n), ja que per a mostrar el ranking, abans ordenem els usuaris amb el mètode Collections.sort de Java, que té un cost algorítmic de O(n log n).

- **FiveGuess:**

  El codi d'aquesta classe és depenent de dos variables, la mida del codi i la quantitat de colors.

  Totes les funcions exceptuant backtrackingCombinations, pruneCandidates i findMinMax, tenen n accesos a llista els quals són constants O(1) gràcies a les llibreries de java, per tant, el cost d'aquestes seria O(n).

La generació de codis es realitza utilitzant l'algoritme de backtracking ve donada per la mida i pels colors totals, n. Si prenem mida = m el cost de la funció seria O(n^m).

A la funció de pruneCandidates, en el pitjor cas, s'ha de comparar totes les combinacions generades al backtracking amb la actual, per tant el cost total seria O(n^2).

En el pitjor cas de findMinMax() ocorre el mateix que al pruneCandidates, s'ha de comparar cada combinació restant amb els candidats, és a dir, O(n^2).

Sabent que totes les altres funcions tenen cost O(n), per tant, el cost total de la classe és: O(n^m + n^2 + n^2 + n) = O(n^m).

- **Genetic:**

  En aquesta classe el cost és semblant al del FiveGuess, és també dependent de les variables de mida i la quantitat de colors.

  La majoria de funcions són un bucle amb accesos amb cost constant gràcies a les llibreries de java, per tant, el seu cost és de O(n), exceptuant backtrackingCombinations i evaluatePopulation.

  En la classe Genetic trobem la funció backtrackingCombinations que té el mateix cost que al FiveGuess donat per la mida i els colors, O(n^m).

  La funció evaluatePopulation avalua la puntuació d'un codi concret amb el codi secret recorrent els dos amb un doble bucle, per tant, el cost total de la funció seria O(n^2).

  El valor final del cost ve donat per la suma de totes les funcions, per tant, el cost de la classe és: O(n^m + n^2 + n) = O(n^m).