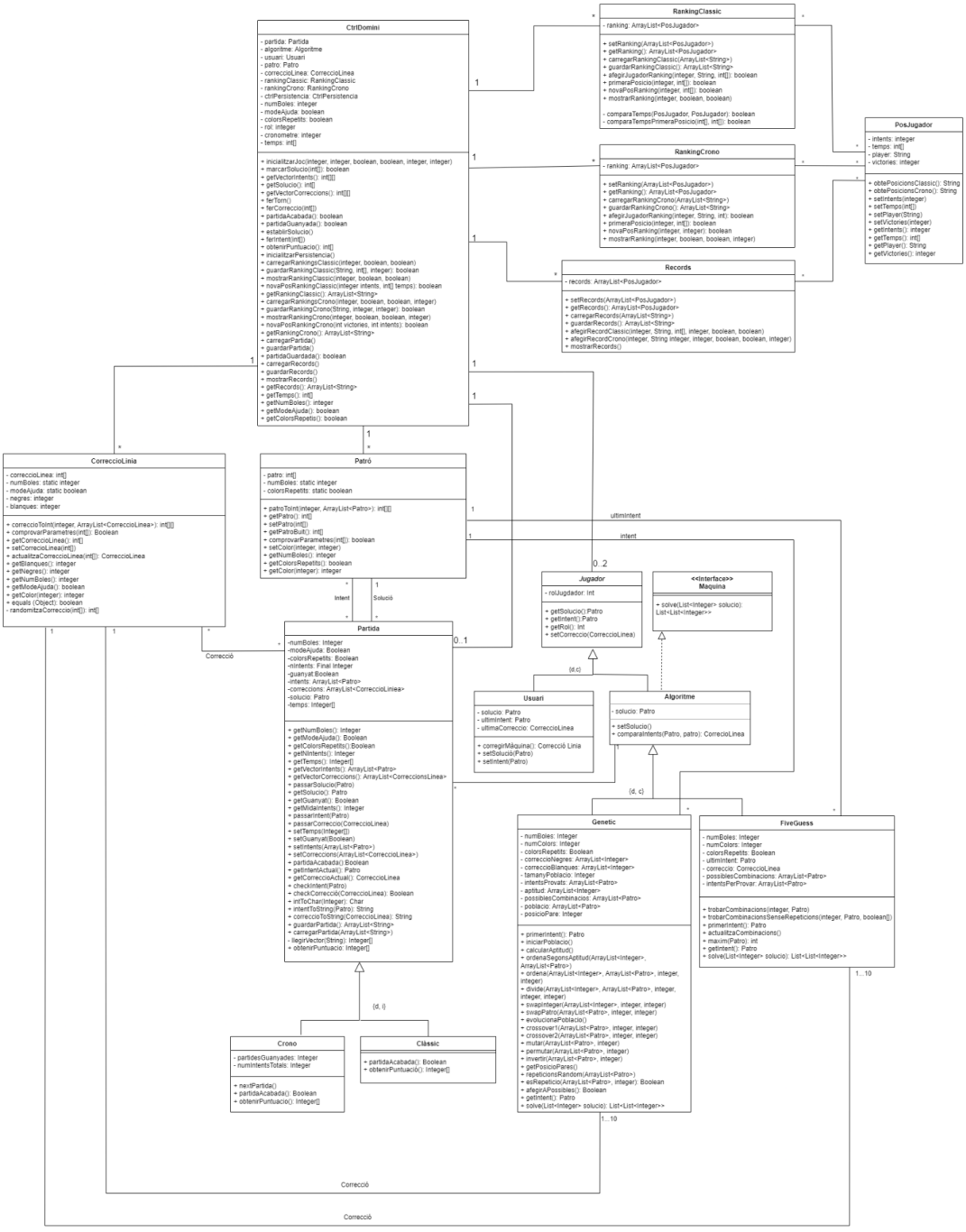
Diagrama classes Domini



Com no es pot visualitzar bé en tamany A4, hem adjuntat aquest diagrama en el document "*Diagrama de classes Domini.pdf*".

Descripció

CtrlDomini

Nom de la classe: CtrlDomini

Descripció:
Classe que serveix per a coordinar el funcionament de totes les classes del domini.

Cardinalitat:
- Cada CtrlDomini o no té Partida o en té una.
- Cada CtrlDomini o no té Jugador o en té 2.
- Cada CtrlDomini té molts Patro.
- Cada CtrlDomini té moltes CorreccionsLinea.
- Cada CtrlDomini té molts RankingClassic.
- Cada CtrlDomini té molts RankingCrono.
- Cada CtrlDomini té molts Records.

Descripció atributs:
- **Partida partida:** És l'atribut de la classe Partida.
- **Algoritme algoritme:** És l'atribut de la classe algoritme.
- **Usuari usuari:** És l'atribut de la classe usuari.
- **Patro patro:** És l'atribut de la classe Patro.
- **CorreccioLinea correccioLinea:** És l'atribut de la classe CorreccioLinea.
- **RankingClassic rankingClassic:** És l'atribut de la classe RankingClassic.
- **RankingCrono rankingCrono:** És l'atribut de la classe RankingCrono.
- **Records records:** És l'atribut de la classe Records.
- **CtrlPersistencia ctrlPersistencia:** És l'atribut de la classe CtrlPersistencia.
- **static int numBoles:** És el número de boles en que es juga la partida.
- **static boolean colorsRepetits:** Si es true, es juga amb colors repetits, si no, no.
- **static boolean modeAjuda:** Si es true, esta el mode ajuda activat, si no, no.
- **static int rol:** És l'atribut que indica si es juga com a Codemaker o com a Codebreaker, els valors són 0 i 1 respectivament.
- **static int cronometre:** És l'atribut que indica el temps del cronòmetre en cas de jugar en mode crono. Altrament, aquest atribut es 0.
- **static int[] temps:** És l'atribut que indica el temps de la partida, el [0] són els minuts i [1] els segons.

Descripció mètodes:
- **InicialitzarJoc(int rol, int numBoles, boolean colorsRepetits, boolean modeAjuda, int cronometre):** Aquest mètode s'encarrega d'inicialitzar un joc i tot el que comporta aquesta acció amb les variables donades.
- **marcarSolucio(int[] solucioV):** Aquest mètode s'encarrega de donar la solució del joc per al mode codemaker. Retorna un booleà: true en cas que calgui posar una altre solució i false en cas contrari.
- **getVectorIntents():** Aquest mètode s'encarrega de retornar una matriu amb els intents que es porten a la partida. Retorna una matriu d'enters amb els intents. Cada fila és un intent i cada columna un valor de l'intent.

- **getSolucio():** Aquest mètode s'encarrega de retornar la solució guardada. Retorna un vector d'enters que representa la solució guardada.
- **getVectorCorreccions():** Aquest mètode s'encarrega de retornar una matriu amb les correccions que es porten a la partida. Retorna una matriu d'enters amb les correccions. Cada fila és una correcció i cada columna un valor de la correcció.
- **ferTorn():** Aquest mètode s'encarrega de demanar-l'hi a la màquina l'intent.
- **ferCorreccio(int[] Correccio):** Aquest mètode s'encarrega d'enviar a la partida la correcció feta pel jugador. Retorna un booleà: true en cas que la correcció sigui correcte, false en cas contrari.
- **partidaAcabada():** Aquest mètode s'encarrega de comprovar si la partida s'ha acabat o no. Retorna un booleà: true en cas afirmatiu, false en cas contrari.
- **partidaGuanyada():** Aquest mètode s'encarrega de comprovar si la partida jugada s'ha guanyat o no. Retorna un booleà: true en cas afirmatiu, false en cas contrari.
- **establirSolució():** Aquest mètode s'encarrega que la màquina estableixi una solució per a la partida.
- **ferIntent(int[] Intent):** Aquest mètode s'encarrega d'enviar a la partida l'intent fet pel jugador.
- **obtenirPuntuacio():** Aquest mètode s'encarrega d'obtenir la puntuació de la partida. Retorna un vector d'enters. En cas que el mode de joc sigui el classic, un vector amb el nombre d'intents. En cas contrari, amb el nombre de victories i d'intents totals.
- **inicialitzarPersistencia()**: Aquest mètode s'encarrega d'inicialitzar la persistencia de l'aplicació.
- **carregaRankingsClassic(int numBoles, boolean modeAjuda, boolean colorsRepetits):** Aquest mètode s'encarrega de carregar els rankings classic al joc.
- **carregaRankingsCrono(int numBoles, boolean modeAjuda, boolean colorsRepetits, int cronometre):** Aquest mètode s'encarrega de carregar els rankings crono al joc.
- **mostrarRankingsClassic(int numBoles, boolean modeAjuda, boolean colorsRepetits):** Aquest mètode s'encarrega de mostrar el ranking classic seleccionat.
- **mostrarRankingsCrono(int numBoles, boolean modeAjuda, boolean colorsRepetits, int cronometre)**: Aquest mètode s'encarrega de mostrar el ranking crono seleccionat.
- **getRankingClassic():** Aquest mètode s'encarrega de retornar el ranking classic perquè es pugui veure a la presentació. Retorna un ArrayList de Strings que representa el ranking.
- **getRankingCrono():** Aquest mètode s'encarrega de retornar el ranking crono perquè es pugui veure a la presentació. Retorna un ArrayList de Strings que representa el ranking.
- **getRecords():** Aquest mètode s'encarrega de retornar la llista de records perquè es puguin veure a la presentació. Retorna un ArrayList de Strings que representa la llista de records.
- **guardarRankingClassic(String player, int[] temps, int intents):** Aquest mètode s'encarrega de guardar el ranking classic a l'arxiu. Retorna un booleà: true en cas que la nova posició sigui la primera, false en cas contrari.
- **guardarRankingCrono(String player, int victories, int intents):** Aquest mètode s'encarrega de guardar el ranking crono a l'arxiu. Retorna un booleà: true en cas que la nova posició sigui la primera, false en cas contrari.

- **guardarPartida():** Aquest mètode s'encarrega de guardar la partida a l'arxiu.
- **carregarPartida():** Aquest mètode s'encarrega de carregar la partida al joc.
- **getTemps():** Aquest mètode s'encarrega de retornar el temps. Retorna un vector d'enters que representa els minuts i els segons.
- **getModeAjuda():** Aquest mètode s'encarrega de retornar si la partida s'està jugant amb el mode ajuda o no. Retorna un booleà que representa si la partida té el mode ajuda activat.
- **getNumBoles():** Aquest mètode s'encarrega de retornar el nombre de boles guardat. Retorna un enter amb el nombre de boles amb la que s'està jugant la partida.
- **getColorsRepetits():** Aquest mètode s'encarrega de retornar si la partida s'està jugant en colors repetits o no. Retorna un booleà que representa si la partida té els colors repetits activats
- **novaPouRankingClassic(int intents, int[] temps):** Aquest mètode s'encarrega de comprovar si el jugador accedirà al ranking classic. Retorna un boolèa: true en cas que el jugador accedeixi al ranking, false en cas contrari.
- **novaPosRankingCrono(int victories, int intents):** Aquest mètode s'encarrega de comprovar si el jugador accedirà al ranking crono. Retorna un boolèa: true en cas que el jugador accedeixi al ranking, false en cas contrari.
- **partidaGuardada():** Aquest mètode s'encarrega de comprovar si hi ha una partida guardada en els arxius. Retorna un booleà: true en cas que existeixi, false en cas contrari.
- **guardarRecords():** Aquest mètode s'encarrega de guardar els records a l'arxiu.
- **carregarRecords():** Aquest mètode s'encarrega de carregar els records al joc.
- **mostrarRecords():** Aquest mètode s'encarrega de mostrar el llistat de records.

CorreccioLinea

Nom de la classe: CorreccioLinea

Descripció:
Classe que serveix per a les correccions de un intent, tant amb el mode ajuda activat o desactivat.

Cardinalitat:
- Cada CorreccioLinea té moltes partides.
- Cada CorreccioLinea té molts FiveGuess.
- Cada CorreccioLinea té un CtrlDomini.

Descripció atributs:
- **int[] correccioLinea:** És un vector que representa la correcció.
- **static int numBoles:** És el número de boles en que es juga la partida.
- **static boolean modeAjuda:** Si es true, esta el mode ajuda activat, si no, no.
- **integer negres:** És el número de pius negres de la solució (color correcte, posició correcte).
- **integer blanques:** És el número de pius blancs de la solució (color correcte, posició incorrecte).

Descripció mètodes:

- **correccioToInt(ArrayList<CorreccioLinea> MatArrayCorr):** Aquest mètode s'encarrega de traduir un ArrayList de CorreccioLinea a una matriu d'enters. Retorna una matriu de ints que és igual que l'ArrayList de CorreccioLinea.
- **comprovarParametres(int[] p):** Aquest mètode s'encarrega de comprovar si els paràmetres donats són correctes.
- **getNumBoles():** Aquest mètode s'encarrega de retornar el nombre de boles guardat. Retorna un enter amb el nombre de boles amb la que s'està jugant la partida.
- **getModeAjuda():** Aquest mètode s'encarrega de retornar si la partida s'està jugant en mode ajuda o no. Retorna un booleà que representa si la partida té el mode ajuda activat.
- **getCorreccioLinea():** Aquest mètode s'encarrega de retornar la correcció guardada. Retorna un vector d'enters que representa una correcció.
- **getColor(int pos):** Aquest mètode s'encarrega d'agafar el color d'una posició de la correcció guardada. Retorna un enter que és el color que volem agafar.
- **actualitzaCorreccióLinea(int[] Correccio):** Aquest mètode s'encarrega de retornar la CorreccióLinea passada com a paràmetre, pero de dos possibles maneres Si esta el mode ajuda activat, retorna la correcció tal qual, en cas contrari, randomitza la correcció. Retorna una CorreccioLinea randomitzada o no.
- **randomitzaCorreccio(int[] p):** Aquest mètode s'encarrega de randomitzar la correcció que es passa com a paràmetre. Retorna el mateix vector d'enters però randomitzat.
- **getNegres():** Aquest mètode s'encarrega d'agafar el nombre de pius negres de la correcció. Retorna un enter que representa el nombre de pius negres.
- **getBlanques():** Aquest mètode s'encarrega d'agafar el nombre de pius blancs de la correcció. Retorna un enter que representa el nombre de pius blancs.
- **equals(Object o):** Aquest mètode s'encarrega de comprovar si el paràmetre és igual a la correcció. Retorna un booleà: true en el cas que sigui igual, false en cas contrari.

Patro

Nom de la classe: Patro

Descripció:
Classe que serveix tant per al vector solució com per al vector intent.

Cardinalitat:

- Cada Patro intent té moltes partides.
- Cada Patro solució té moltes partides.
- Cada Patro té molts FiveGuess.
- Cada Patro té un CtrlDomini.

Descripció atributs:

- **int[] patro:** És un vector d'enters que representa una combinació de colors.
- **static int numBoles:** És el número de boles en que es juga la partida.
- **static boolean colorsRepetits:** Si es true, es juga amb colors repetits, si no, no.

Descripció mètodes:

- **patroToInt(ArrayList<Patro> MatArrayInt):** Aquest mètode s'encarrega de traduir un ArrayList de patrons a una matriu d'enters. Retorna una matriu de ints que és igual que l'ArrayList de Patrons.
- **comprovarParametres(int[] p):** Aquest mètode s'encarrega de comprovar si els paràmetres donats són correctes.
- **setPatro(int[] p):** Aquest mètode s'encarrega de guardar un patro en la classe.
- **getPatro():** Aquest mètode s'encarrega de retornar el patro guardat. Retorna un vector d'enters que representa un torn.
- **setColor(int pos, int color):** Aquest mètode s'encarrega de guardar els colors en un patro.
- **getNumBoles():** Aquest mètode s'encarrega de retornar el nombre de boles guardat. Retorna un enter amb el nombre de boles amb la que s'està jugant la partida.
- **getColorsRepetits():** Aquest mètode s'encarrega de retornar si la partida s'està jugant en colors repetits o no. Retorna un booleà que representa si la partida té els colors repetits activats
- **getColor(int pos):** Aquest mètode s'encarrega d'agafar el color d'una posició del patro guardat. Retorna un enter que és el color que volem agafar.

Partida

Nom de la classe: Partida

Descripció:
Classe que representa una partida de mastermind.

Cardinalitat:

- Cada partida té un Ctrldomini.
- Cada partida té moltes correccions.
- Cada partida té una solució.
- Cada partida té molts intents.
- Cada partida té un Algoritme.
- És la superclasse de Crono i Clàssic.

Descripció atributs:

- **integer numBoles:** El nombre de boles que conté el codi.
- **boolean modeAjuda:** Indica si el mode ajuda està activat.
- **boolean colorsRepetits:** Indica si en la partida es permeten colors repetits.
- **final integer nIntents:** Indica el nombre d'intents que té una partida. Es fixe, sempre és 10.
- **boolean guanyat:** indica si s'ha guanyat la partida.
- **Patro[ ] intents:** Tots els intents de la partida actual ordenats per el torn. ex: patro[0] -> intent del torn 0.
- **CorreccioLinea[ ] correccions:** Correccions dels intents ordenats per torn.
- **Patro solucio:** La solució de la partida actual.
- **Int[] temps:** EL temps de la partida actual

Descripció mètodes:

- **getNumBoles():** Retorna el nombre de boles de la Partida.
- **getModeAjuda():** Retorna si el modeAjuda ha sigut activat o no
- **getColorsRepetits():** Retorna si s'acepten colors repetits en la Partida.
- **getNIntents():** Retorna el nombre d'intents màxims que té la Partida.
- **getTemps():** Retorna el temps de la Partida.
- **getVectorIntents():** Retorna l'atribut intents.
- **getVectorCorreccions():**Retorna l'atribut correccions.
- **passarSolucio():** Fica valor a l'atribut solucio.
- **getSolucio():** Retorna la solució actual de la Partida.
- **getGuanyat():** Retorna si la partida ha sigut guanyada o no.
- **getMidaIntents():** Retorna el nombre d'intents fets pel jugador en la Partida.
- **passarIntent():** Afegeix un intent a l'atribut intents.
- **passarCorreccio():** Afegeix una correcció a l'atribut correccions.
- **setTemps():** Actualitza l'atribut temps.
- **setGuanyat():** Actualitza l'atribut guanyat.
- **setIntents():** Actualitza l'atribut intents.
- **setCorreccions():** Actualitza l'atribut correccions.
- **partidaAcabada():** Diu si la partida actual ha sigut acabada o no.
- **getIntentActual():** Retorna l'intent actual.
- **getCorreccioActual():** Retorna la correcció actual.
- **CheckIntent():** Permet comprovar si el patró intent coincideix amb el patró solució.
- **CheckCorreccio():** Permet comprovar si la correcció donada és correcte.
- **intToChar():** Passa el paràmetre de integer a char.
- **intentToString():** Passa el paràmetre de Patro a String.
- **correccioToString():** Passa el paràmetre de CorreccioLinea a String.
- **guardarPartida():** Guarda la Partida en un ArrayList de Strings.
- **carregarPartida():** Transforma el ArrayList passat com a paràmetre en una Partida.
- **llegirVector():** Llegeix el vector en forma de String i el converteix en un vector de integers.
- **obtenirPuntuacio():** Retorna la informació valuable de la Partida.

Clàssic

Nom de la classe: Clàssic

Descripció:
Classe que representa una partida jugada amb les regles clàssiques de Mastermind.

Cardinalitat:

- És el fill de la superclasse Partida.
- Una Partida Clàssic l'està jugant un Usuari.

Descripció mètodes:

- **partidaAcabada():** retorna si la Partida ha sigut acabada o no.
- **obtenirPuntuacio():** retorna la puntuació corresponent amb el mode de joc.

Crono

Nom de la classe: Crono

Descripció:
Classe que representa una partida on el jugador fa el rol de codebreaker jugada de manera que guanya totes les partides possibles en un temps determinat.

Cardinalitat:
- És el fill de la superclasse Partida.

Descripció atributs:
- **Int partidesGuanyades:** nombre de partides guanyades durant el modo crono
- **Int numIntentsTotals:** el nombre d'intents durant tota la Partida

Descripció mètodes:
- **nextPartida():** fa totes les preparacions per poder fer una nova Partida sense sortir del modo Crono.
- **partidaAcabada():** retorna si la Partida ha sigut acabada o no.
- **obtenirPuntuacio():** retorna la puntuació corresponent amb el mode de joc.

Jugador

Nom de la classe: Jugador

Descripció:
Classe que representa al jugador, tant d'usuari com d'algoritme.

Cardinalitat:
- Cada jugador com a Codebrearker pot fer moltes partides.
- Cada jugador com a Codemaker pot fer moltes partides.
- És una superclasse d'Usuari i Algoritme.

Descripció atributs:
- **integer Rol:** Representa el rol (0:Codemaker, 1:Codebreaker) del jugador.

Descripció mètodes:
- **getSolucio():** Permet consultar el valor del patró solució.
- **getIntent():** Permet consultar el valor de l'últim intent.
- **getRol():** Permet consultar el rol del jugador.
- **setCorreccio(CorreccioLinea):** Permet determinar el valor de la última correcció feta a un intent del jugador.

Usuari

Nom de la classe: Usuari

Descripció:
Classe que representa al usuari.

Cardinalitat:
- És el fill del supeclasse Jugador.

Descripció atributs:

- **Patro solucio:** Representa la solució establerta per l'usuari.
- **Patro ultimIntent:** Representa l'últim intent establert per l'usuari.
- **Patro ultimaCorreccio:** Representa la última correcció establerta per l'usuari.

Descripció mètodes:

- **setSolucio(Patro):** Permet determinar el valor del patró solució.
- **setIntent(Patro):** Permet determinar el valor de l'últim intent.
- **corregirMaquina():** Permet consultar el valor de la última correcció.

Algoritme

Nom de la classe: Algoritme

Descripció:
Classe que representa al propi ordinador, que serà l'encarregat de dur a terme els algoritmes.

Cardinalitat:

- És el fill de la superclasse Jugador.
- És una superclasse de Genètic i FiveGuess.
- Un algoritme pot estar en moltes partides.

Descripció atributs:

- **Patro solucio:** Representa la solució establerta per l'usuari.

Descripció mètodes:

- **setSolucio():** Permet generar una solució aleatòria que s'adapta a les regles del joc.
- **getSolucio():** Permet consultar el valor del patró solució.
- **getIntent():** Funció ganxo per consultar l'intent de la màquina, està implementada a les subclasses genètic i 5 guesses
- **solve(List<Integer> solution):** Funció ganxo que retorna una llista buida.
- **comparaIntents(Patro solucio, Patro intent):** Compara els patrons i retorna una correcció linea.

Màquina

Nom de la classe: <<interface>> Màquina

Descripció:
Interfície màquina que utilitza els algoritmes (five guess o genètic) per crear una llista de les combinacions que et porten a la solució.

Descripció mètodes:

- **solve(List<Integer> solucio):** Retorna una llista dels intents que fa l'algoritme per arribar a la solució solution.

FiveGuess

Nom de la classe: FiveGuess

Descripció:
Classe que representa l'algoritme 5 Guesses.

Cardinalitat:
- És el fill de la superclasse Algoritme.
- Cada FiveGuess té molts patrons.
- Cada FiveGuess té com a màxim 10 correccioLinea i com a mínim 1.

Descripció atributs:
- **final integer numBoles:** integer amb el número de boles de la partida.
- **final integer numColors:** integer amb el número de colors de la partida.
- **final boolean colorsRepetits:** boolean que indica si es permeten colors repetits.
- **Patro ultimIntent:** últim patró que s'ha intentat.
- **CorreccioLinea correccio:** CorreccioLinea on es guarden les correccions de l'usuari en cada torn.
- **final ArrayList<Patro> possiblesCombinacions:** vector amb totes les possibles combinacions.
- **ArrayList<Patro> intentsPerProvar:** vector amb les combinacions que encara falten per provar.

Descripció mètodes:
- **setCorreccio(CorreccioLinea solucio):** Guarda la correcció feta per l'usuari.
- **trobarCombinacions(int posicio, Patro combinacio):** Genera totes les possibles combinacions amb els 8 colors i el número de boles donat.
- **trobarCombinacionsSenseRepeticions(int position, Patro combinacio, boolean[] usat):** Genera totes les combinacions possibles, sense repeticions de colors.
- **primerIntent():** Retorna un patró que serà la primera combinació de colors que es provarà.
- **actualitzaCombinacions():** Elimina de l'array "possiblesCombinacions" les combinacions que tenen correccions pitjors a la donada pel jugador.
- **maxim(Patro seguent)** Calcula el màxim número d'encerts de les combinacions i retorna aquest com un integer.
- **getIntent():** Retorna els intents a provar. En el cas de que sigui la primera vegada que es crida la funció, genera totes les combinacions possibles, sinó actualitza les combinacions i comprova quina és la millor solució comparant els mínims i màxims calculats.
- **solve(List<Integer> solucio):** Retorna una llista dels intents que fa l'algoritme per arribar a la solució solution.

Genetic

Nom de la classe: Genètic

Descripció:
Classe que representa l'algoritme de Genètic.

Cardinalitat:
- És el fill de la superclasse Algoritme.
- Cada Genetic té molts patrons.
- Cada Genetic té com a màxim 10 correccioLinea i com a mínim 1.

Descripció atributs:

- **final integer numBoles:** integer amb el número de boles de la partida.
- **final integer numColors:** integer amb el número de colors de la partida.
- **final boolean colorsRepetits:** boolean que indica si es permeten colors repetits.
- **final ArrayList<Integer> correccioNegres:** array amb les correccions negres.
- **final ArrayList<Integer> correccioBlanques:** array amb les correcions blanques.
- **static final integer tamanyPoblacio:** integer que representa la mida de les poblacions.
- **final ArrayList<Patro> intentsProvats:** array amb els patrons corresponents als intents provats.
- **final ArrayList<Integer> aptitud:** array amb l'aptitud de cada població.
- **final ArrayList<Patro> possiblesCombinacions:** llista amb totes les possibles combinacions.
- **ArrayList<Patro> poblacio:** llista on es guarden tots els patrons de la població.
- **integer posicioPare:** integer que serveix per escollir que 2 patrons s'utilitzaran per les funcions crossover1 i crossover2.

Descripció mètodes:
- **setCorreccio(CorreccioLinea solucio):** Guarda la correcció feta per l'usuari en dues llistes de boles negres i blanques.
- **primerIntent():** Retorna un patró que serà la primera combinació de colors que es provarà.
- **iniciarPoblacio():** Crea una població aleatòria i s'encarrega de buidar les llistes.
- **calcularAptitud():** Calcula l'aptitud de cada patró de la població. Compara tots els patrons de la població amb els intents provats.
- **ordenaSegonsAptitud(ArrayList<Integer> apt, Arraylist<Patro> intents):** Ordena els patrons de la població segons les aptituds.
- **ordena(ArrayList<Integer> apt, Arraylist<Patro> intents, int petit, int gran):** Ordena segons l'aptitud, divideix per la meitat i va ordenant.
- **divide(Arraylist<Integer> apt, Arraylist<Patro> intents, int petit, int gran, int pivot):** Decideix la posició en la qual es tallarà el patró i mou tant les aptituds com els patrons.
- **swapInteger(Arraylist<Integer> apt, int pos1, int pos2):** Intercanvia integers de la llista d'aptituds passada per paràmetre.
- **swapPatro(Arraylist<Patro> patrons, int pos1, int pos2):** Intercanvia de posicions patrons de la llista passada per paràmetre.
- **evolucionaPoblacio():** Modifica, aleatòriament, la població de diferents formes.

- **crossover1(Arraylist<Patro> novaPoblacio, int fill1, int fill2):** Agafa dos patrons aleatoris d'una llista i fa un crossover de les dues parts de patrons.
- **crossover2(Arraylist<Patro> novaPoblacio, int fill1, int fill2):** Agafa dos patrons aleatoris d'una llista i fa un crossover de les tres parts dels patrons.
- **mutar(Arraylist<Patro> novaPoblacio, int posicio):** Modifica el color d'una posició per un color aleatori.
- **permutar(Arraylist<Patro> novaPoblacio, int posicio):** Permuta els colors d'un patró entre dues posicions random.
- **invertir(Arraylist<Patro> novaPoblacio, int posicio):** Inverteix els colors entre dues posicions aleatòries.
- **getPosicioPares():** Retorna un integer que es genera de forma random, aquest representa la posició per escollir patrons de la població.
- **repeticionsRandom(Arraylist<Patro> novaPoblacio):** Modifica els patrons que es repeteixen entre la nova població i la guardada a la variable població. Si es repeteixen genera una nova combinació de forma random.
- **esRepeticio(Arraylist<Patro> novaPoblacio, int posicio):** Compara un patró de la població passada amb la població guardada a la variable població. Retorna un boolea: true si aquest patró es repeteix, false en cas contrari.
- **afegirAPossibles():** Compara els intents provats i els patrons de la població per saber si s'han d'afegir a les possibles solucions. Retorna un boolea: true si es pot afegir a la llista de possibles combinacions, false en cas contrari.
- **getIntent():** Decideix el pròxim intent que es provarà. En el cas de que sigui la primera vegada que es crida la funció, retorna un primer intent ja predefinit, sinó crea la població, calcula les aptituds i les ordedna. Després comprova quina és la millor solució segons les aptituds calculades.
- **solve(List<Integer> solucio):** Retorna una llista dels intents que fa l'algoritme per arribar a la solució solution.

RankingCrono

Nom de la classe: RankingCrono

Descripció:
Classe que serveix per als rankings del mode crono. S'encarrega també de traduir els rankings a un arxiu de text per a guardar-ho a la persistencia.

Cardinalitat:
- Cada RankingCrono té un CtrlDomini.
- Cada RankingCrono té moltes PosJugador.

Descripció atributs:
- **ArrayList<String> ranking:** És un Array que conté les posicions del ranking en mode crono.

Descripció mètodes:
- **setRanking(ArrayList<PosJugador> ranking):** Aquest mètode s'encarrega de guardar un rankingCrono.
- **getRanking():** Aquest mètode s'encarrega de carregar un rankingCrono.
  Retorna un ArrayList de PosJugador que representa el ranking.

- **carregarRankingCrono(ArrayList<String> ranking):** Aquest mètode s'encarrega de traduir un ArrayList de strings a un ranking
  Retorna un ArrayList de strings on cada string és una posició del ranking.
- **guardarRankingCrono():** Aquest mètode s'encarrega de traduir un ranking a un ArrayList de strings.
  Retorna un ArrayList de string que és el contingut que després anirà guardat a un arxiu.
- **afegirJugadorRanking(int intents, String player, int victories):** Aquest mètode s'encarrega de comparar els resultats de la nova partida amb les partidas guardades al ranking per veure quina posició és millor.
  Retorna un boolèa. En cas que la nova posició sigui millor que les anteriors retorna un true, en cas contrari, un false.
- **primeraPosicio(int intents, int victories):** Aquest mètode s'encarrega de comprovar si la nova posició és un record.
  Retorna un booleà. Retorna un true en cas que la nova posició sigui un record, un false en cas contrari.
- **novaPosRanking(int victories, int intents):** Aquest mètode s'encarrega de comprovar si el jugador accedirà al ranking.
  Retorna un booleà. Retorna true en cas que el jugador accederixi al ranking, un false en cas contrari.
- **mostrarRanking(int numBoles, boolean modeAjuda, boolean colorsRepetits, int cronometre):** Aquest mètode s'encarrega de mostrar el ranking per pantalla.

RankingClassic

Nom de la classe: rankingClassic

Descripció:

Classe que serveix per als rankings del mode clàssic. S'encarrega també de traduir els rankings a un arxiu de text per a guardar-ho a la persistencia.

Cardinalitat:

- Cada RankingClassic té un CtrlDomini.
- Cada RankingClassic té moltes PosJugador.

Descripció atributs:

- **ArrayList<String> ranking:** És un Array que conté les posicions del ranking en mode classic.

Descripció mètodes:

- **setRanking(ArrayList<PosJugador> ranking):** Aquest mètode s'encarrega de guardar un rankingClassic.
- **getRanking():** Aquest mètode s'encarrega de carregar un rankingClassic.
  Retorna un ArrayList de PosJugador que representa el ranking.
- **carregarRankingClassic(ArrayList<String> ranking):** Aquest mètode s'encarrega de traduir un ArrayList de strings a un ranking
  Retorna un ArrayList de strings on cada string és una posició del ranking.
- **guardarRankingClassic():** Aquest mètode s'encarrega de traduir un ranking a un ArrayList de strings.

Retorna un ArrayList de string que és el contingut que després anirà guardat a un arxiu.

- **comparaTemps(PosJugador novaPosicio, PosJugador posicionsGuardades):** Aquest mètode s'encarrega de comparar quin temps en inferior per veure quin jugador té una millor posició en el ranking.
  Retorna un booleà. En cas que el temps de la nova posició sigui menor es retorna un true, en cas contrari, un false.
- **afegirJugadorRanking(int intents, String player, int[] temps):** Aquest mètode s'encarrega de comparar els resultats de la nova partida amb les partidas guardades al ranking per veure quina posició és millor.
  Retorna un boolèa. En cas que la nova posició sigui millor que les anteriors retorna un true, en cas contrari, un false.
- **comparaTempsPrimeraPosicio(int[] tempsNou, int[] tempsPrimeraPos):** Aquest mètode s'encarrega de comparar el temps de la partida amb el millor temps per veure si és un record.
  Retorna un booleà. En cas que el temps de la nova posició sigui menor es retorna un true, en cas contrari, un false.
- **primeraPosicio(int intents, int[] temps):** Aquest mètode s'encarrega de comprovar si la nova posició és un record.
  Retorna un booleà. Retorna un true en cas que la nova posició sigui un record, un false en cas contrari.
- **novaPosRanking(int intents, int[] temps):** Aquest mètode s'encarrega de comprovar si el jugador accedirà al ranking.
  Retorna un booleà. Retorna true en cas que el jugador accederixi al ranking, un false en cas contrari.
- **mostrarRanking(int numBoles, boolean modeAjuda, boolean colorsRepetits):** Aquest mètode s'encarrega de mostrar el ranking per pantalla.

Records

Nom de la classe: Records

Descripció:

Classe que serveix per als records. S'encarrega també de traduir el llistat de records a un arxiu de text per a guardar-ho a la persistencia.

Cardinalitat:

- Cada Records té un CtrlDomini.
- Cada Records té moltes PosJugador.

Descripció atributs:

- **ArrayList<String> ranking:** És un Array que conté les posicions del llistat de records.

Descripció mètodes:

- **setRecords(ArrayList<PosJugador> records):** Aquest mètode s'encarrega de guardar uns records.
- **getRecords():** Aquest mètode s'encarrega de carregar uns records.
  Retorna un ArrayList de PosJugador que representa la llista de records.

- **carregarRecords(ArrayList<String> records):** Aquest mètode s'encarrega de traduir un ArrayList de strings a un ranking
  Retorna un ArrayList de strings on cada string és una posició del ranking.
- **guardarRecords():** Aquest mètode s'encarrega de traduir un ranking a un ArrayList de strings.
  Retorna un ArrayList de string que és el contingut que després anirà guardat a un arxiu.
- **afegirRecordClassic(int intents, String player, int[] temps, int numBoles, boolean modeAjuda, boolean colorsRepetits):** Aquest mètode s'encarrega d'afegir un nou record en el mode classic
- **afegirRecordCrono(int intents, String player, int victories, int numBoles, boolean modeAjuda, boolean colorsRepetits, int cronometre):** Aquest mètode s'encarrega d'afegir un nou record en el mode crono.
- **mostrarRecords():** Aquest mètode s'encarrega de mostrar els records per pantalla

PosJugador

Nom de la classe: PosJugador

Descripció:
Classe que fa la funció de tupla. Serveix per a emmagatzemar les dades de les posicions dels jugadors en els RankingsClassic, RankingCrono i els Records.

Cardinalitat:
- Cada Records té molts RankingClassic.
- Cada Records té molts RankingCrono.
- Cada Records té molts Records.

Descripció atributs:
- **int intents:** Depèn del mode de joc. En el mode classic, és el nombre d'intents fets a la partida, mentre que en el mode crono, es el nombre total d'intents de totes les partides guanyades.
- **int[] temps:** És el temps trigat a la posició, es divideix el vector en minuts i segons.
- **String player:** És el nom del jugador de la posició.
- **int victories:** És el nombre de victòries de la posició.

Descripció mètodes:
- **obtePosicionsClassic()**: Aquest mètode s'encarrega de traduir els atributs de la classe en un string per al ranking classic.
  Retorna un string que conté la informació dels atributs de la classe.
- **obtePosicionsCrono():** Aquest mètode s'encarrega de traduir els atributs de la classe en un string per al ranking crono.
  Retorna un string que conté la informació dels atributs de la classe.
- **setIntents(int intents):** Aquest mètode s'encarrega guardar els intents en la classe.
- **setTemps(int[] temps):** Aquest mètode s'encarrega de guardar els intents en la classe.
- **setPlayer(String player):** Aquest mètode s'encarrega de guardar el jugador en la classe.

- **setVictories(int victories):** Aquest mètode s'encarrega guardar les victories en la classe.
- **getIntents():** Aquest mètode s'encarrega de retornar els intents.
  Retorna un enter que és el nombre d'intents.
- **getTemps()**: Aquest mètode s'encarrega de retornar el temps.
  Retorna un vector d'enters que representa els minuts i els segons.
- **getPlayer():** Aquest mètode s'encarrega de retornar el jugador.
  Retorna un string que és el nom del jugador.
- **getVictories():** Aquest mètode s'encarrega de retornar les victòries.
  Retorna un enter que és el nombre de victòries.