Mastermind

Lliurament 1

Grup 11.3

Joel Corredor Casares
Rosa Hortelano Bronchal
Susanna Jané Ramón
Nicola Scognamillo

Índex

Casos d'ús

Diagrama

Descripció

Jugar

Nom: Jugar

Actor: Jugador

Comportament:

- El jugador indica que vol començar una partida.
- El sistema pregunta si vol continuar la partida pausada o si vol crear una nova.

Errors possibles i cursos alternatius:

- En el cas que no hi hagi una partida pausada, el sistema, crea una nova directament.

include: Escollir rol

Nom: Escollir rol

Actor: Jugador

Comportament:

- El jugador escull l'opció jugar com a *Codemaker* o *Codebreaker*, que són jocs diferents.
- El sistema prepara el tipus de joc acord amb el rol escollit.

Errors possibles i cursos alternatius:

- (no hi ha)

include: Configurar partida

Nom: Configurar partida

Actor: Jugador

Comportament:

- El jugador indica que vol modificar la configuració per defecte del mode de la partida.
- El sistema entra en el menú de configuració de partida.

Errors possibles i cursos alternatius:

- (no hi ha)

include: Canviar nombre de boles

Nom: Canviar nombre de boles

Actor: Jugador

Comportament:

- El jugador indica el nombre de boles amb el que voldrà jugar.
- El sistema selecciona el nombre de boles que tindrà el patró escollit per l'usuari.

Errors possibles i cursos alternatius:

- (no hi ha)

include: Toggle colors repetits

Nom: Toggle colors repetits

Actor: Jugador

Comportament:

- El jugador indica si vol jugar o no amb l'opció de colors repetits. Aquesta permet que els patrons es formin amb colors repetits.
- El sistema guarda l'opció colors repetits.

Errors possibles i cursos alternatius:

- (no hi ha)

include: Mode ajuda

Nom: Mode ajuda

Actor: Jugador

Comportament:

- El jugador indica que vol activar el mode ajuda. Aquest et diu quina bola és correcta i et permet eliminar automàticament, de la llista, les boles que no són.
- El sistema escull el mode ajuda.

Errors possibles i cursos alternatius:

- Tot i que s'activi el mode ajuda sent codemaker, quan es comenci la partida es tornarà a desactivar.

include: Seleccionar mode de joc

Nom: Seleccionar mode joc

Actor: Jugador

Comportament:

- El jugador escull entre mode normal i mode crono de tipus de partida.
- El sistema selecciona el mode de joc escollit.

Errors possibles i cursos alternatius:

- (no hi ha)

include: Fer torn

Nom: Fer torn

Actor: Jugador

Comportament:

- Si el jugador és Codemaker, aquest ha definit una combinació de colors solució que intentarà endevinar la màquina. També ha d'anar corregint les combinacions que va introduint la màquina. Un cop la màquina introdueix un patró, l'usuari ha de comparar-lo amb la solució primerament introduïda. La normativa per qualificar-les és la següent: si el color és correcte i està en la posició correcta es posarà una bola negra, si el color està bé, però en una posició incorrecta posarà una bola negra.
- Si el jugador és Codebreaker, es crearà un patró solució de colors aleatori, respectant la configuració introduïda, i haurà d'anar introduït diferents intents de patró per tal d'endevinar la combinació correcta.
- El sistema en cas de Codemaker guardarà el patró com a solució de la partida i comprovarà que la correcció introduïda pel jugador sigui correcta, per evitar que el jugador faci trampes.
- En cas de Codebreaker un cop confirmat l'intent, el sistema calcularà les boles blanques i negres corresponents de la correcció de linea.

Errors possibles i cursos alternatius:

- La combinació pot estar incompleta (patró de colors incomplet). En aquest cas apareixerà una finestra amb un avís.
- Colors repetits en un mode de joc que no ho permet. Igual que en el cas anterior apareixerà un avís.
- Possible error del jugador en introduir una correcció incorrecta. En aquest cas s'obrirà una finestra amb un avís del fet que aquesta correcció ha estat incorrecte.

include: Guardar i sortir

Nom: Guardar i sortir

Actor: Jugador

Comportament:

- El jugador indica que vol sortir al menú, guardant prèviament la partida iniciada.
- El sistema, pausa el temps i guarda la partida en l'estat actual.

Errors possibles i cursos alternatius:

- (no hi ha)

include: Restart

Nom: Restart

Actor: Jugador

Comportament:

- El jugador indica que vol reiniciar la partida.
- El sistema torna a crear una partida amb les mateixes característiques de la que acaba de finalitzar.

Errors possibles i cursos alternatius:

- (no hi ha)

include: Sortir Menú

Nom: Sortir Menú

Actor: Jugador

Comportament:

- El jugador indica que vol sortir al menú i tancar la partida iniciada.
- El sistema tanca la partida començada sense guardar i surt al menú.

Errors possibles i cursos alternatius:

- (no hi ha)

Carregar partida

Nom: Carregar partida

Actor: Jugador

Comportament:

- El jugador indica que vol carregar una partida guardada.
- El sistema carrega la partida prèviament guardada.

Errors possibles i cursos alternatius:

- Si no hi ha cap partida desada encara, apareix un pop-up avisant-ho.

Consultar rànquing

Nom: Consultar rànquing

Actor: Jugador

Comportament:

- El jugador vol consultar el rànquing.
- El sistema porta a una pantalla amb els rànquings (el crono i el normal) on surten les partides guanyades ordenades.

Errors possibles i cursos alternatius:

- (no hi ha)

Consultar com jugar

Nom: Consultar com jugar

Actor: Jugador

Comportament:

- El jugador indica que vol consultar la normativa per a jugar en l'idioma preferit.
- El sistema ensenya el manual de com jugar, on s'expliquen totes les regles: els tipus de joc, com jugar, el mode ajuda…

Errors possibles i cursos alternatius:

- (no hi ha)

Sortir

Nom: Sortir

Actor: Jugador

Comportament:

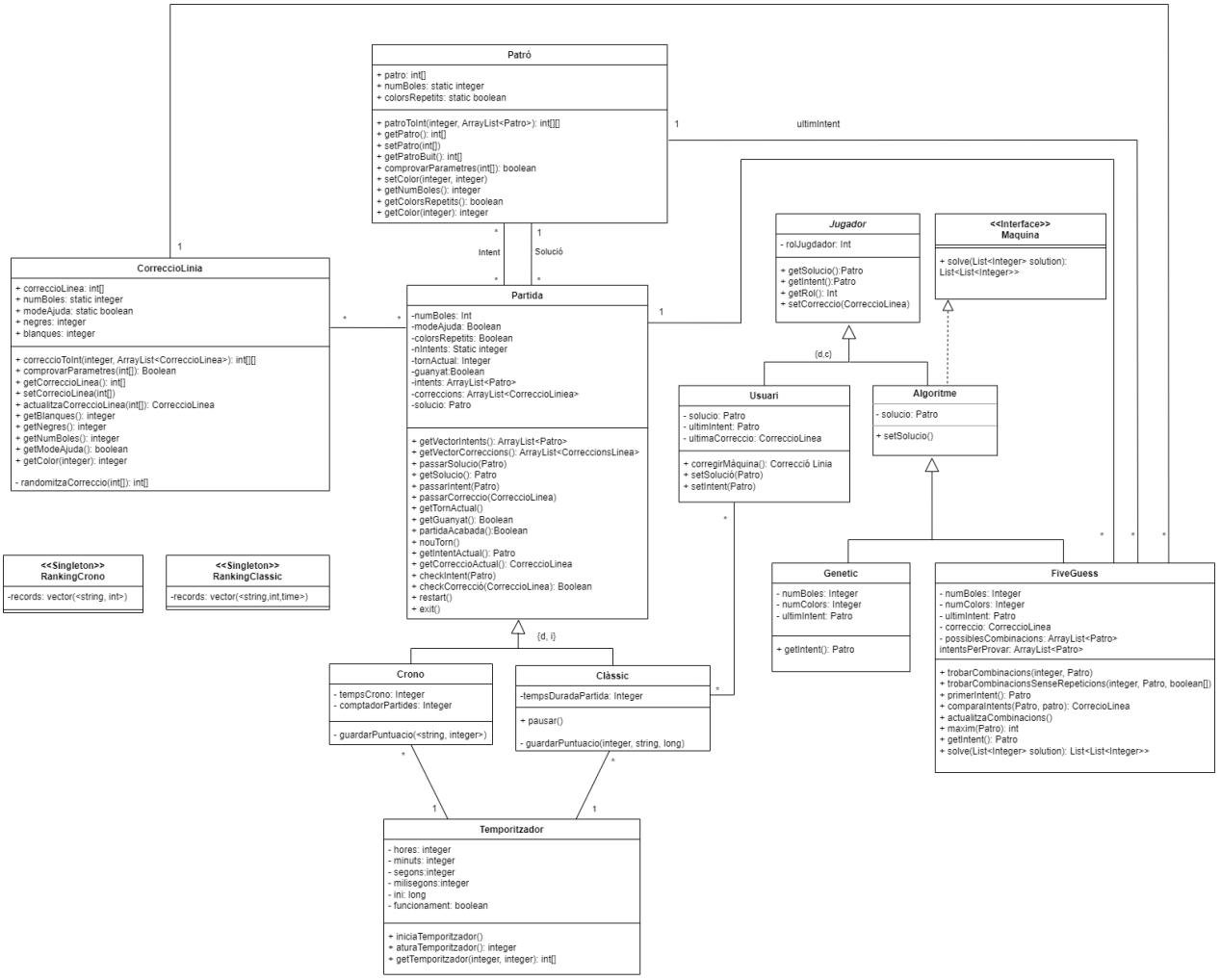
- El jugador indica que vol sortir i tancar el joc.
- El sistema tanca el joc.

Errors possibles i cursos alternatius:

- (no hi ha)

Model

Diagrama



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

Descripció

Drivers

Nom de la classe: Drivers

Descripció:
Drivers per a veure el funcionament del joc.

Descripció atributs:

- **integer NumBoles:** És el número de boles en que es juga la partida.
- **boolean ColorsRepetits:** Si es true, es juga amb colors repetits, si no, no.
- **boolean ModeAjuda:** Si es true, esta el mode ajuda activat, si no, no.

- **integer Rol:** És l'atribut que indica si es juga com a Codemaker o com a Codebreaker, els valors són 0 i 1 respectivament.
- **Partida partida:** És l'atribut de la classe Partida.

Descripció mètodes:

- **main(String[] args):** És el main de la classe.
- **Benvinguda():** Mostra el menú principal amb les funcions implementades i per implementar a futur.
- **Play():** És la funció per a gestionar la primera opció del main.
- **MenuPlay():** Mostra el menú dins de l'apartat de jugar amb varies opcions més.
- **Regles():** És la funció per a gestionar la cuarta opció del main.
- **Error():** És la funció que mostra un missatge d'error en cas d'escollir una opció inexsitent.
- **NoImplementat():** És la funció que mostra un missatge d'error en cas d'escollir una opció no implementada.
- **ReglesAngles():** Mostra les regles del joc en anglès.
- **ReglesCastella():** Mostra les regles del joc en Castellà.
- **PartidaPerduda():** Mostra un missatge de derrota.
- **PartidaGuanyada():** Mostra un missatge de victoria.
- **ComencarPartida():** És la funció per a gestionar la nova partida.
- **MenuPartida():** Mostra el menú de la partida.
- **Configuració():** És la funció per a gestionar la configuració de la partida.
- **MenuConfiguracio():** Mostra el menú de la configuració.
- **EscollirHelpMode():** És la funció per a gestionar el mode ajuda.
- **MenuHelpMode():** Mostra el menú per a escollir el mode ajuda.
- **EscollirNumBoles():** És la funció per a gestionar el número de boles.
- **MenuNumBoles():** Mostra el menú per a escollir el número de boles.
- **EscollirColorsRepetits():** És la funció per a gestioanr el mode de colores repetits.
- **MenuColorsRepetits():** Mostra el menú per a escollir el mode de colors repetits.
- **EscollirRol():** És la funció per a gestionar el rol de l'usuari.
- **MenuRol():** Mostra el menú de selecció de rol.

CtrlDomini

Nom de la classe: CtrlDomini

Descripció:
És l'encarregat de comunicarse entre la capa de presentació (en aquesta entrega, amb els drivers) i les altres classes.

Descripció atributs:

- **Partida partida:** És l'atribut de la classe Partida.
- **Jugador maquina:** És l'atribut de la classe maquina.
- **Jugador 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.
- **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.

Descripció mètodes:

- **inicialitzarJoc(int NumBoles, boolean ColorsRepetits, boolean ModeAjuda, int Rol):** És la funció encarregada de inicialitzar el joc, inicialitzarà atributs i crearà classes.
- **marcarSolucio(int[] Solucio):** Agafa la solució donada per l'usuari, comprova que sigui correcte i l'envia a la partida i al jugador.
- **getVectorIntents():** Agafa l'Array d'intents de partida i la transforma a matriu de ints per a pasarla a presentació.
- **getVectorCorreccions():** Agafa l'Array de correccions de partida i la transforma a matriu de ints per a pasarla a presentació.
- **ferTorn():** Agafa l'intent que ha fet la maquina i l'envia a partida.
- **ferCorreccio(int[] Correccio):** Comprova que la correcció donada per l'usuari sigui correcte i comprova que no fagi trampes, i en aquest cas, l'envia a la maquina.
- **partidaAcabada():** Pregunta a Partida si la partida s'ha acabat, en cas negatiu, fa un altre torn.
- **partidaGuanyada():** Retorna true si la partida es una victoria o un false si es una derrota.
- **establirSolució():** Agafa la solució donada per la maquina i l'envia a partida.
- **ferIntent(int[] Intent):** Comprova que l'intent donat per l'usuari sigui correcte i en aquest cas, l'envia a la partida. En cas de no ser-ho, retorna un error.
- **getTornActual():** Retorna un int amb el torn actual de la partida.

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.

Descripció atributs:

- **integer[] correcciólinea:** És el vector amb la correcció.
- **static integer 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 boles negres de la correcció (color correcte, posició incorrecte).
- **integer blanques:** És el número de boles blanques de la correcció (color correcte, posició correcte).

Descripció mètodes:

- **correccioToInt(int torn, ArrayList<CorreccioLinea> MatArrayCorr):** Tradueix un ArrayList de CorreccioLinea a una matriu de ints.

- **comprovarParametres(int[] p):** Comprova que el vector de pius sigui correcte, en cas de ser-ho retorna un true, en cas contrari, retorna un false.
- **getNumBoles():** Retorna el número de boles.
- **getModeAjuda():** Retorna si està activat el mode ajuda.
- **getColor(int i):** Retorna el color de la correcció de la posició pasada com atribut.
- **getNegres():** Retorna el número de pius negres de la correcció.
- **getBlanques():** Retorna el número de pius blancs de la correcció.
- **getCorreccioLinea():** Retorna la correcció guardada.
- **setCorrecioLinea(int[] p):** Se l'hi pasa un vector de pius i el guarda.
- **actualitzaCorreccióLinea(int[] Correccio):** Si esta el mode ajuda activat, retorna el vector de pius ordenat, si no, crida a RandomitzaCorreccio(p).
- **randomitzaCorreccio(int[] p):** En cas que el mode ajuda estigui desactivat, aquesta funció desordena el vector de pius de forma aleatoria.

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.

Descripció atributs:
- **int[] patro:** És el vector de pius de la solució.
- **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(int torn, ArrayList<Patro> MatArrayInt):** Tradueix un ArrayList de Patro a una matriu de ints.
- **comprovarParametres(int[] p):** Comprova que el vector patró sigui correcte, en cas de ser-ho retorna un true, en cas contrari, retorna un false.
- **setPatro(int[] p):** Se l'hi pasa un vector patró i el guarda.
- **getPatroBuit():** Retorna un patró buit.
- **getPatro():** Retorna el patró guardat.
- **setColor(int pos, int color):** Estableix en la posició un color.
- **getNumBoles():** Retorna el número de boles.
- **getColorsRepetits():** Retorna si la partida està en mode colors repetits o no.
- **getColor(int pos):** Retorna el color de la posició demanada.

Partida

Nom de la classe: Partida

Descripció:
Classe que representa una partida de mastermind.

Cardinalitat:

- Cada partida té una solució.
- Cada partida té molts intents.
- 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.
- **integer tornActual:** Indica el torn actual que està jugant el usuari.
- **static integer nIntents:** Indica el nombre d'intents que té una partida. Es fixe, sempre és 10.
- **bool 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.

Descripció mètodes:

- **getVectorIntents():** Retorna l'atribut intents.
- **getVectorCorreccions():**Retorna l'atribut correccions.
- **passarSolucio(Patro solucio):** Fica valor a l'atribut solucio.
- **getSolucio():**
- **passarIntent(Patro intent):** Afegeix un intent a l'atribut intents.
- **passarCorreccio(CorreccioLinea correccio):** Afegeix una correcció a l'atribut correccions.
- **getTornActual():** Retorna l'atribut tornActual.
- **getGuanyat():** Retorna l'atribut guanyat.
- **partidaAcabada():** Diu si la partida actual ha sigut acabada o no.
- **nouTorn():** Augmenta el torn actual.
- **getIntentActual():** retorna l'intent actual.
- **getCorreccioActual():** retorna la correcció actual.
- **CheckIntent(Patro intent):** Permet comprovar si el patró intent coincideix amb el patró solució.
- **CheckCorreccio(CorreccioLinea correccio):** Permet comprovar si la correcció donada es correcte.
- **Restart():** Permet començar una nova partida amb la configuració actual. (No implementat per aquesta entrega)
- **Exit():** Permet sortir de la partida.(No implementat per aquesta entrega)

Clàssic

Nom de la classe: Clàssic

Descripció:
Classe que representa una partida on el jugador fa el rol de codebreaker jugada de manera clàssica.

Cardinalitat:

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

Descripció atributs:

- **integer tempsDuradaPartida:** el temps que ha durat la partida.

Descripció mètodes:

- **Pausar():** Pausa la partida(pausa el temporitzador).
- **GuardarPuntuacio():** (no implementada per aquesta entrega).

Crono

Nom de la classe: Crono

Descripció:
Classe que representa una partida on el jugador fa el rol de codebreaker jugada de manera crono. (Aquesta classe s'implementarà més endavant, després de la primera entrega)

Cardinalitat:

- És el fill de la superclasse Partida.
- Una Partida Crono té un Temporitzador.

Descripció atributs:

- **Integer tempsCrono:** el temps que té l'usuari per jugar la partida.
- **Integer ComptadorPartides:** les partides guanyades seguidament.

Temporitzador

Nom de la classe: Temporitzador

Descripció:
Classe que serveix per al temporitzador de la partida.

Cardinalitat:

- Cada Temporitzador té moltes partides Classic
- Cada Temporitzador té moltes partides Crono

Descripció atributs:

- **integer hores, minuts, segons, milisegons:** Són els atributs que guarden el temps.
- **long ini:** És l'atribut que guarda el temps inicial del cronòmetre, es un long perquè la funció System.currentTimeMillis() retorna un long.
- **boolean funcionament:** És true si el cronòmetre esta en funcionament, false en cas contrari.

Descripció mètodes:

- **iniciaTemporitzador():** Fica el cronòmetre en funcionament.
- **aturaTemporitzador():** Atura el cronòmetre per si s'acaba la partida o es vol pausar.
- **getTemporitzador():** Retorna el temps en diferents formats depenent del 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.

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.

FiveGuess

Nom de la classe: FiveGuess

Descripció:

Classe que representa l'algoritme 5 Guesses.

Cardinalitat:

- És el fill de la superclasse Algoritme.

Descripció atributs:

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

Descripció mètodes:

- **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.
- **comparaIntents(Patro solucio, Patro intent):** Compara els patrons i retorna una correcció linea.
- **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> solution):** 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.

Descripció atributs:

- **integer numBoles:** integer amb el número de boles de la partida.
- **integer numColors:** integer amb el número de colors de la partida.
- **Patro ultimIntent:** últim patró que s'ha intentat.

Descripció mètodes:
- **getIntent():** Retorna els intents a provar.

Ranking Crono

Nom de la classe: rankingCrono

Descripció:
Rànquing on s'ensenyen el rècords en el mode crono, organitzats per nombre de codis trencats. (Aquesta classe s'implementarà més endavant, després de la primera entrega)

Descripció atributs:
- **Vector<string, integer, time> records**: La puntuació del jugador.

Ranking Clàssic

Nom de la classe: rankingClassic

Descripció:
Rànquing on s'ensenyen el rècords en el mode clàssic, organitzats per nombre d'intents emprats i temps trigat. (Aquesta classe s'implementarà més endavant, després de la primera entrega)

Descripció atributs:
- **Vector<string, integer, time> records**: La puntuació del jugador.

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> solution):** Retorna una llista dels intents que fa l'algoritme per arribar a la solució solution.

ED&ALG. Funcionalitat Ppal

Estructures de dades

Hem fet servir com a estructures de dades les nostres pròpies classes de Patro i CorreccioLinea. Aquestes utilitzen arrays per guardar les diferents combinacions possibles.

La classe Patro s'utilitza en les classe Partida, Usuari, Algoritme i FiveGuess. Mentre que correccioLinea s'utilitza a Partida i FiveGuess.

També hem utilitzat un stub que s'utilitza per el test de la classe Jugador.

Per últim hem utilitzat HashMap per l'algoritme de la classe FiveGuess. D'aquesta forma hem millorat l'eficiència de l'algoritme.

Five-guess algorithm

Que és?

L'algorisme Five Guess es tracta d'una cerca exhaustiva, mitjançant la tècnica de minimax. Endevina la combinació proposada pel Codebreaker en una mitjana de 4.3411 torns pel cas habitual de 4 posicions i 6 colors, en el pitjor cas serien 6 voltes. Aquest algorisme va ser demostrat per Donald Knuth en 1977.

La tècnica minimax és un algorisme recursiu per a triar el següent moviment en un joc de n-jugadors, normalment en un joc de 2 jugadors. Un valor es associat a cada posició o estat del joc. Aquest valor esta decidit per una funció que avalua les posicions i indica que tan bona pot ser per al jugador arribar a aquesta posició. El jugador després fa un moviment que maximitza el mínim valor de les possibles posicions on podrà moure el rival.

Com l'hem implementat?

Per implementar aquest algoritme, hem creat una classe anomenada "*FiveGuess*". Aquesta és una extensió de la classe "*Algoritme*".

Funcions principals:

Hem creat 9 funcions diferents per a poder implementar l'algoritme. Aquestes funcions són les següents *setCorrecció(), trobarCombinacions(), trobarCombinacionsSenseRepeticions(), primerIntent(), comparaIntents(), actualitzaCombinacions(), maxim(), getIntent() i solve().*

L'algoritme se centra en la funció *getIntent()*, que va cridant a la resta. En crear una partida nova com a *codemaker* es fa la crida a la funció per primer cop. Si aquesta té la llista de possibilitats buida, crida a una funció per trobar totes les combinacions. Depenent si la partida permet colors repetits o no, es cridarà una funció o un altre. Ho hem fet així per reduir la quantitat de possibilitats i fer l'algoritme més eficient. Les operacions encarregades de fer això són *trobarCombinacions()* i *trobarCombinacionsSenseRepeticions(),* que es basen a fer una cerca exhaustiva (algoritme de cerca utilitzant la força bruta). Les combinacions generades es guarden en un vector de patrons (ArrayList). Fem una copia de la llista i la guardem com a intents per provar.

Després es crida a la funció primerIntent() que retorna un patró específic segons el nombre de boles de la partida i si es poden repetir colors o no. En el cas de 4 boles amb repeticions serà el patró {0, 0, 1, 1}, si són 6 o 8 boles es genera una combinació aleatoria. D'aquesta forma eliminen la màxima quantitat de combinacions possibles i les següents iteracions són més fàcils. Sense repeticions serien les següents: 4 boles {0, 1, 2, 3}, 6 {0, 1, 2, 3, 4, 5} i 8 {0, 1, 2, 3, 4, 5, 6, 7}.

Un cop provat aquest intent, la partida retorna la correcció del patró enviat i s'actualitzen les possibles combinacions amb la funció *actualitzaCombinacions()*. Aquesta operació s'encarrega de comparar les diferents possibilitats de la llista amb l'últim intent (combinació provada) i si són pitjors que la solució proporcionada pel jugador descartar-les. Això ho fa cridant a les funcions *setCorreccio()* i *comparaIntents()*. A través de la classe algoritme s'agafa la correcció feta per l'usuari utilitzant *setCorreccio()*. Amb *comparaIntents()* es recorren els patrons per obtenir la seva correcció (quantitat de pius negres i blancs) i es compara amb la correcció donada pel jugador.

Després d'actualitzar les possibilitats, s'avaluarà quina combinació serà la que elimini més patrons en la següent iteració. Per fer això s'ha de buscar per cada combinació possible el mínim de patrons que s'eliminarien per després obtenir el màxim entre els mínims. Per a fer això, hem de calcular el màxim d'encerts de cada patró, amb l'operació *maxim()*. Aquesta compara (amb *comaparaIntent())* el patró passat amb la resta de combinacions de la llista, i calcula el màxim d'encerts d'aquesta.

Ja que per cada parella de patrons només pot haver-hi una correcció, hem decidit utilitzar un HashMap. D'aquesta forma no hem de recórrer totes les correccions. Aquest HashMap té com a clau la correcció línia i el nombre d'encerts pel patró amb la resta. Cada cop que per dos patrons es dona un control específic, se suma en 1 el valor d'aquest. Mentre es fan aquests càlculs, es va guardant el màxim.

Un cop tenim aquest valor calculem el mínim de patrons eliminats (mínim = possibles combinacions - màxim d'encerts). Amb aquest mínim, el comparem amb el màxim mínim d'encerts de la iteració anterior, si aquest nou mínim és major que l'anterior se substitueix. Si són iguals i l'anterior combinació ja formava part de les possibles combinacions, no es fa res, ja que s'haurà d'agafar el patró més petit. Si el patró anterior no es troba entre les possibles combinacions i l'actual si, es canviarà.

El pròxim intent es decidirà segons la combinació amb el màxim de patrons eliminats. Aquesta combinació l'eliminarem de les combinacions per provar i serà retornada a la classe algoritme per a després continuar fins a arribar a la solució.

Tot el que hem explicat s'ha de fer per cada intent, a partir d'actualitzar les combinacions. En el cas que la combinació sigui de 4 boles s'arribarà a la solució en 5 iteracions. En els altres casos té un màxim de 10 intents per endevinar la combinació, si no ho aconsegueix perdrà la partida.

Relació classes per membre grup

Repartició de classes

Joel Corredor Casares:

- CorreccioLinia
- CorreccioLineaTest
- Patro
- Temporitzador
- TemporitzadorTest
- CtrlDomini
- RankingClassic
- RankingCrono
- Excepcions de les meves classes

Rosa Hortelano Bronchal:

- Five Guess
- FiveGuessTest
- Genètic

Susanna Jané Ramón:

- Partida
- PartidaTest
- Clàssic
- Crono
- Excepcions de les meves classes
- PatroTest

Nicola Scognamillo:

- Usuari
- Jugador
- Algoritme
- Maquina
- DriverJocComplet
- DriverFiveGuesses
- Makefile