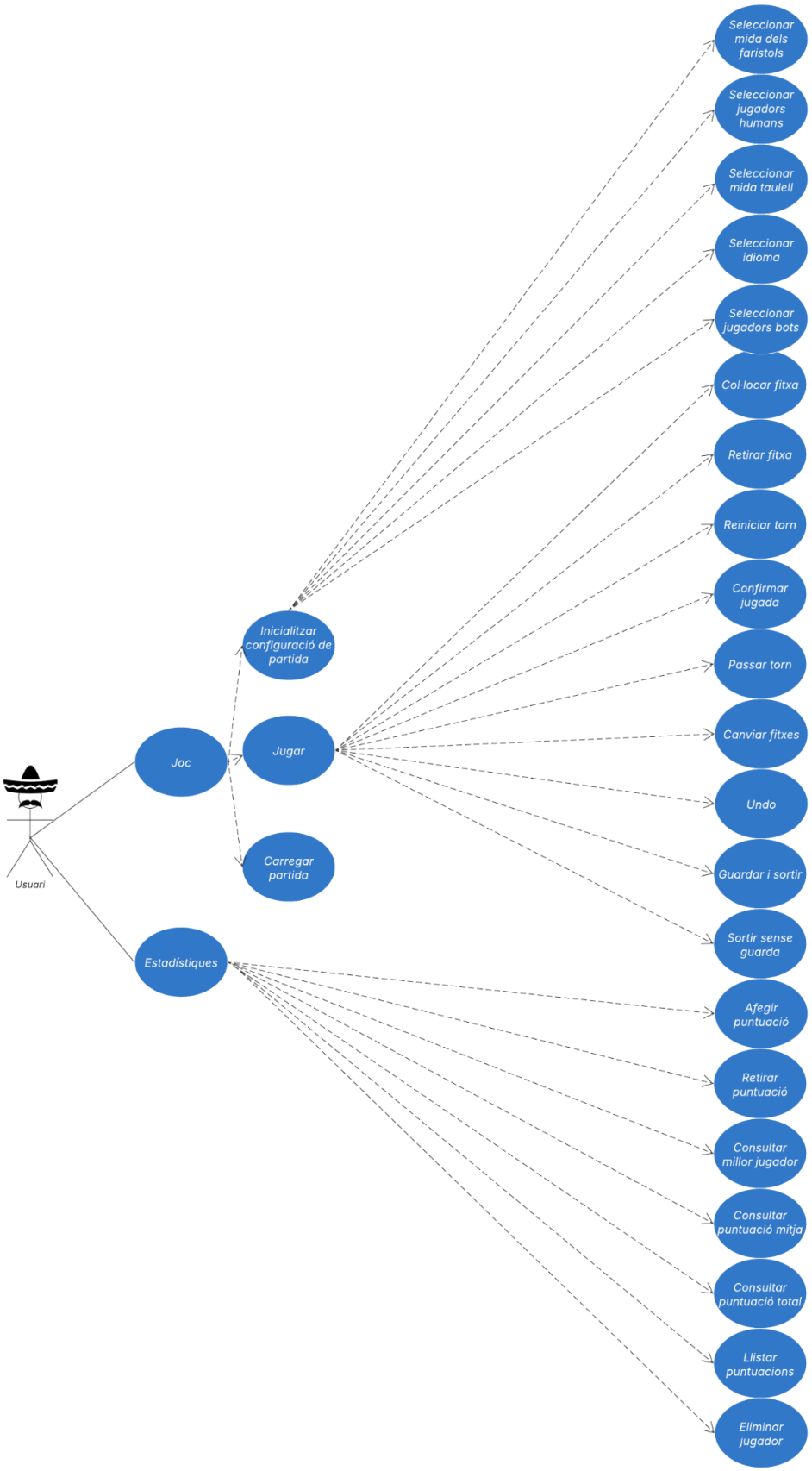
PROP: Scrabble

Grup 13.4

Entrega 1.0
Quadrimestre de primavera, curs 24/25

Toni Gratacós Miralles | **ToniGratacosMiralles**
Jan Pruneda Marcet | **JanPru**
Camila Valeria Santos Cabrera | **camivsantos**
Roger Bitlloch Galceran | **MUX-enjoyer**

1. Casos d'ús


Usuari
Joc
Estadístiques
Inicialitzar configuració de partida
Jugar
Carregar partida
Seleccionar mida dels faristols
Seleccionar jugadors humans
Seleccionar mida taulell
Seleccionar idioma
Seleccionar jugadors bots
Col·locar fitxa
Retirar fitxa
Reiniciar torn
Confirmar jugada
Passar torn
Canviar fitxes
Undo
Guardar i sortir
Sortir sense guarda
Afegir puntuació
Retirar puntuació
Consultar millor jugador
Consultar puntuació mitja
Consultar puntuació total
Llistar puntuacions
Eliminar jugador

1.1. Partida

Nom	Inicialitzar configuració de partida
Actor	Usuari
Comportament	1) L'usuari selecciona "Nova Partida" en el menu principal 2) El sistema mostra una pestanya amb totes les opcions de configuració pas a pas
Errors possibles i cursos alternatius	1) L'usuari en qualsevol moment pot cancelar tota la operació de creació de partida i s'el retorna a la pàgina principal

Nom	Seleccionar mida del taulell
Actor	Usuari
Comportament	1) El sistema demana la mida del taulell 2) L'usuari introdueix un valor numèric 3) El sistema valida la mida i l'accepta
Errors possibles i cursos alternatius	3) Si la mida es invalida(massa petita / negativa ...) es mostra un missatge d'error i es demana repetir

Nom	Seleccionar mida del faristol
Actor	Usuari
Comportament	1) El sistema demana la mida del faristol 2) L'usuari introdueix un valor numèric 3) El sistema valida la mida i l'accepta
Errors possibles i cursos alternatius	3) Si la mida es invalida(massa petita / negativa ...) es mostra un missatge d'error i es demana repetir

Nom	Seleccionar idioma del diccionari
Actor	Usuari
Comportament	1) El sistema mostra els idiomes disponibles (català, castellà, anglés) 2) L'usuari selecciona un d'ells 3) El sistema valida i guarda l'acció
Errors possibles i cursos alternatius	3) Si l'idioma no és vàlid, es mostra un missatge d'error i es demana repetir

Nom	Seleccionar jugadors humans
Actor	Usuari
Comportament	1) El sistema solicita el número de jugadors humans 2) L'usuari introdueix un número 3) El sistema demana el nom de cada jugador
Errors possibles i cursos alternatius	2) Si el número es zero o negatiu es mostra un missatge d'error i es demana repetir 3) Si el nom és buit, es mostra un missatge d'error i es demana repetir

Nom	Seleccionar jugadors bots
Actor	Usuari
Comportament	1) El sistema solicita el número de jugadors bots 2) L'usuari introdueix un número 3) El sistema demana la dificultat de cada bot - Fàcil - Nomal - Difícil
Errors possibles i cursos alternatius	2) Si el número es zero o negatiu, es mostra un missatge d'error i es demana repetir 3) Si la dificultat no és vàlida, es mostra un missatge d'error i es demana repetir

Nom	Carregar partida
Actor	Usuari
Comportament	1) L'usuari selecciona "Carregar Partida" en el menú principal 2) El sistema mostra les diferents partides guardades amb detalls com: - Data i hora de creació - Jugadors - Estat actual (torn, puntuacions...) 3) L'usuari selecciona la partida que vol continuar 4) El sistema redirigeix a l'usuari a la partida seleccionada
Errors possibles i cursos alternatius	1) Si no hi ha partides guardades, el sistema mostrarà un missatge d'error informant d'això. 2) L'usuari en qualsevol moment pot cancelar tota la operació de carregar partida i s'el retorna a la pàgina principal

Nom	Jugar una partida
Actor	Usuari
Comportament	1) El sistema mostra de qui és el torn actual 2) Si es un bot realitza una jugada automàtica 3) Si es un humà, se li mostren totes les opcions d'acció: - Jugar - Passar Torn - Canviar Fitxes - Undo - Guardar i Sortir - Sortir sense guardar
Errors possibles i cursos alternatius	3) Si es realitza una acció no vàlida, es mostra un missatge d'error i es demana repetir

Nom	Col·locar fitxa
Actor	Usuari
Comportament	1) L'usuari indica la lletra de la fitxa que vol jugar 2) El sistema solicita la fila i la columna on col·locar-la 3) Si es un comodín, el sistema demana per quina lletra es vol substituir 4) El sistema actualitza el taulell
Errors possibles i cursos alternatius	1) Si la fitxa no és vàlida, es mostra un missatge d'error i es demana repetir 2) Si la posició no és vàlida, es mostra un missatge d'error i es demana repetir

Nom	Retirar fitxa
Actor	Usuari
Comportament	1) L'usuari indica la fila i la columna de la fitxa que vol retirar 2) El sistema retira la fitxa i retorna el taulell
Errors possibles i cursos alternatius	1) Si la posició no és vàlida (Ex: La casella està buida), es mostra un missatge d'error i es demana repetir

Nom	Reiniciar torn
Actor	Usuari
Comportament	1) L'usuari indica l'opció de reiniciar el torn 2) El sistema desfà totes les accions realitzades durant el torn en qüestió

Errors possibles i cursos alternatius	1) Si no es pot reiniciar el torn, es mostra un missatge d'error i es demana repetir

Nom	Confirmar jugada
Actor	Usuari
Comportament	1) L'usuari col·loca les fitxes desitjades 2) El sistema valida la jugada, mostra la paraula formada i la puntuació 3) L'usuari selecciona l'opció de confirmar la jugada 4) El sistema assigna els punts i recarrega el taulell
Errors possibles i cursos alternatius	1) Si la jugada no és vàlida, no es pot confirmar

Nom	Passar torn
Actor	Usuari
Comportament	1) L'usuari decideix no jugar i selecciona l'opció de passar el torn 2) El sistema avança al següent jugador.
Errors possibles i cursos alternatius	Cap

Nom	Canviar fitxes
Actor	Usuari
Comportament	1) L'usuari indica quantes fitxes vol canviar 2) El sistema solicita les lletres de les fitxes a intercanviar 3) El sistema valida les fitxes i realitza el canvi amb fitxes aleatòries del sac
Errors possibles i cursos alternatius	3) Si alguna de les fitxes seleccionades no és vàlida (no existeix o no es troba el el faristol del jugador), es mostra un missatge d'error i es demana repetir

Nom	Undo
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció Undo (Desfer) 2) El sistema reverteix l'última acció realitzada
Errors possibles i cursos alternatius	1) Si no es pot desfer, es mostra un missatge d'error

Nom	Guardar i sortir
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de guardar i sortir 2) El sistema guarda el estat actual de la partida 3) El sistema tanca el joc
Errors possibles i cursos alternatius	3) Si hi ha algun error al guardar, es mostra un missatge d'error

Nom	Guardar sense sortir
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de sortir sense guardar 2) El sistema tanca la partida sense guardar els canvis
Errors possibles i cursos alternatius	Cap

1.2. Estadístiques

Els casos d'ús de Afegir puntuació i Retirar puntuació es canviaran per indicar que els fa el sistema automàticament, no obstant ara els fa l'usuari ja que no tenim capa de persistencia.

Nom	Afegir puntuació
Actor	Usuari
Comportament	1) L'usuari indica el nom del jugador i la puntuació 2) El sistema afegeix la puntuació al seu historial
Errors possibles i cursos alternatius	3) Si el nom està buit o la puntuació no es vàlida, es mostra un missatge d'error i es demana repetir

Nom	Retirar puntuació
Actor	Usuari
Comportament	1) L'usuari indica el nom del jugador i la puntuació a retirar 2) El sistema elimina aquella puntuació
Errors possibles i cursos alternatius	1) Si la puntuació no existeix, es mostra un missatge d'error i es demana repetir

Nom	Consultar millor jugador
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de veure el millor jugador 2) El sistema mostra el jugador amb major puntuació
Errors possibles i cursos alternatius	1) Si no hi ha puntuacions registrades, s'informa a l'usuari

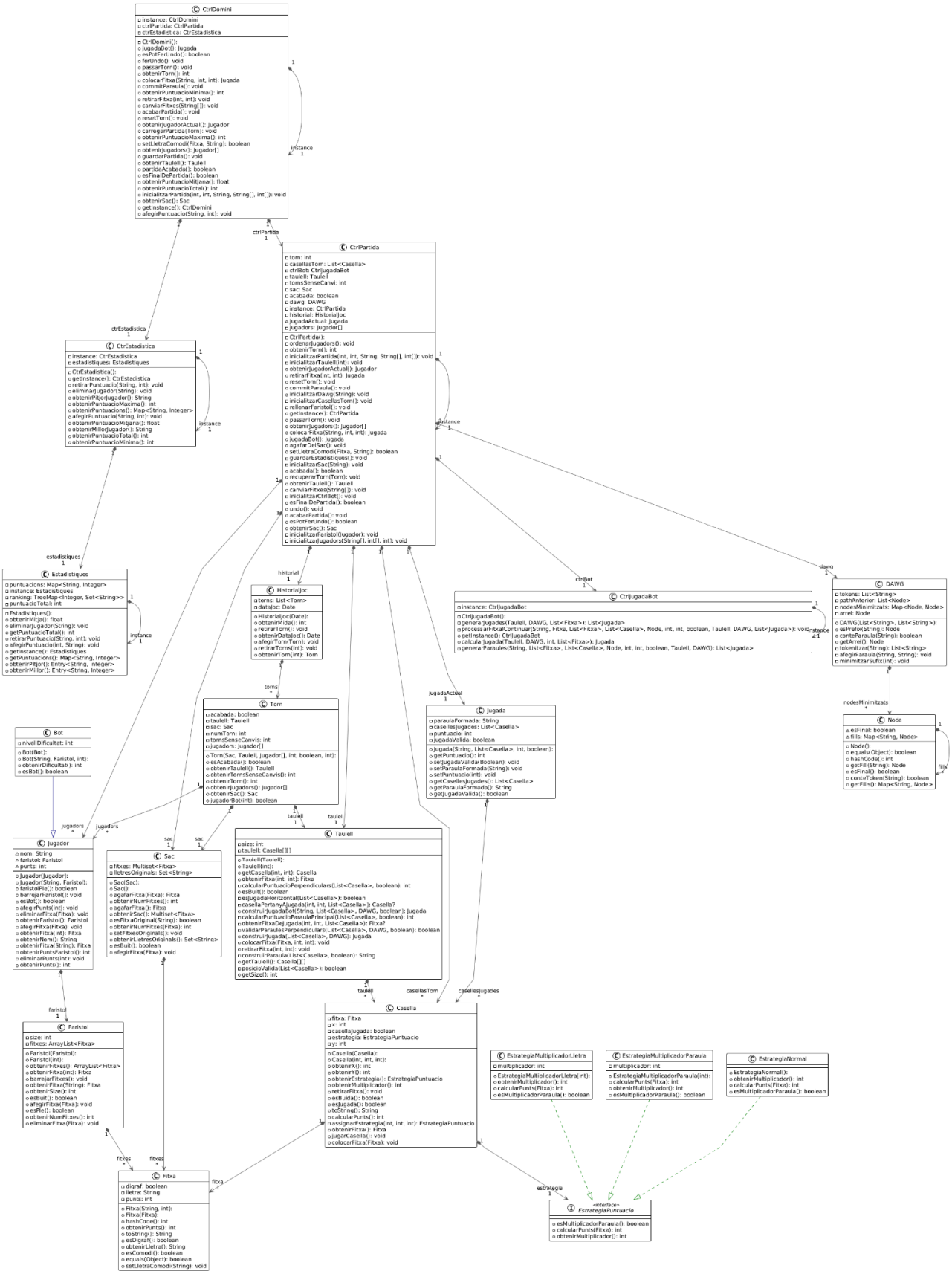
Nom	Consultar puntuació mitja
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de veure la puntuació mitja 2) El sistema mostra la mitja de totes les puntuacions de tots els jugadors registrats
Errors possibles i cursos alternatius	1) Si no hi ha puntuacions registrades, s'informa a l'usuari

Nom	Consultar puntuació total
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de veure la puntuació total 2) El sistema mostra la suma de totes les puntuacions de tots els jugadors registrats
Errors possibles i cursos alternatius	1) Si no hi ha puntuacions registrades, s'informa a l'usuari

Nom	Llistar puntuacions
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de veure les puntuacions 2) El sistema mostra totes les puntuacions, ordenades de major a menor
Errors possibles i cursos alternatius	1) Si no hi ha puntuacions registrades, s'informa a l'usuari

Nom	Eliminar jugador
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció d'eliminar un jugador 2) L'usuari indica el nom del jugador a eliminar 3) El sistema elimina totes les puntuacions registrades d'aquest jugador
Errors possibles i cursos alternatius	1) Si el jugador no existeix, es mostra un missatge d'error i es demana repetir

2. Diagrama de classes[1]



[1] La imatge també en bona qualitat es pot trobar a la carpeta DOCS de la entrega

2.1. Restriccions textuals

I. Els punts d'una fitxa han de ser major o igual que 0
II. No es pot canviar la lletra d'una fitxa que no es comodí
III. La mida d'un faristol ha de ser estrictament positiu
IV. Les fitxes de faristol han de ser un subset de les fitxesOriginals del Sac
V. Les posicions x i y de la casella han de ser més petites que la size del Taulell
VI. Els multiplicadors de una estratègia han de ser positius
VII. La mida del tauler ha de ser imparella per garantir simetria
VIII. La mida del tauler ha de ser mínim 5
IX. Els punts del jugador han de ser majors o iguals a 0
X. No poden haver-hi dos jugadors amb el mateix nom a la mateixa partida
XI. El nivell de dificultat de bot ha de estar entre els valors [1, 2, 3]
XII. El numTorn de la classe Torn no pot ser negatiu
XIII. El tornsSenseCanvis de Torn no potser negatius ni majors que numTorn
XIV. Tots els torns dins de HistorialJoc han de pertànyer a una mateixa partida
XV. La puntuació total de les estadístiques no pot ser negativa

2.2. Descripció de les classes

Nom	Fitxa
Descripció	Representa una fitxa del joc, amb una lletra, un valor en punts i un indicador de si és un dígraf.
Atributs	-String lletra -int punts -boolean dígraf
Mètodes	+Fitxa(String lletra, int punts) : Fitxa Crea una nova fitxa amb la lletra i els punts indicats. +Fitxa(Fitxa copiaFitxa) : Fitxa Inicialitza una fitxa com a còpia d'una altra. +esDigraf(): boolean Indica si la fitxa representa un dígraf. +setLletraComodi(String lletra) Assigna una nova lletra a una fitxa comodí. +esComodi() : boolean Comprova si una fitxa és un comodí. Un comodí és una fitxa amb un valor de punts igual a 0.

Nom	Sac
Descripció	Representa el sac de fitxes del joc. Conté totes les fitxes que encara no s'han utilitzat i la quantitat disponible de cadascuna. El sac permet extreure fitxes de manera aleatòria i controlar quantes queden.
Atributs	-Multiset<Fitxa> fitxes -Set<String> lletresOriginals;
Mètodes	+Sac() : Sac Crea un nou sac buit de fitxes. +Sac(Sac copiaSac) : Sac Inicialitza un sac com a còpia d'un altre. +otenirLletresOriginals(): Set<String> Retorna les lletres inicials del sac. +setFitxesOriginals() Inicialitza les lletres originals del sac. +esFitxaOriginal(String lletra): boolean Comprova si una lletra és un fitxa original. +afegirFitxa(Fitxa f) Afegeix una fitxa al sac. Si la fitxa ja hi és, incrementa la seva quantitat. +agafarFitxa() : Fitxa Extreu una fitxa aleatòria del sac. Un cop seleccionada, la seva quantitat es redueix en una unitat. ExcepcioSacBuit +agafarFitxa(Fitxa fitxa) : Fitxa Extreu una instància concreta d'una fitxa del sac. Redueix en una unitat la seva quantitat disponible. ExcepcioSacNoConteLaFitxa +obtenirNumFitxes() : int Retorna el nombre total de fitxes disponibles al sac, sumant totes les quantitats de cada tipus de fitxa. +obtenirNumFitxes(Fitxa fitxa) : int Retorna la quantitat disponible d'una fitxa concreta al sac. +esBuit() : boolean Comprova si el sac està buit. +obtenirSac(): Multiset<Fitxes> Retorna el contingut intern del sac.

Nom	Casella
Descripció	Representa una casella del tauler de joc. Cada casella té una posició definida per coordenades (x,y), pot contenir una fitxa i té associada una estratègia de puntuació.
Atributs	-int x -int y -Fitxa fitxa -boolean casellaJugada -EstrategiaPuntuació estrategia
Mètodes	+Casella(int x, int y, int size) Crea una nova casella amb els paràmetres indicats. +Casella(Casella copia) Inicialitza una casella com a còpia d'una altra. +calcularPunts() : int Calcula els punts que aporta la fitxa en aquesta casella segons l'estratègia assignada. +jugarCasella() Marca la casella com a jugada. +esBuida() : booelan Comprova si la casella està buida (no té fitxa). +colocarFitxa(Fitxa fitxa) Col·loca una fitxa a la casella. ExcepcioCasellaOcupada si la casella ja conté una fitxa. +retirarFitxa() Retira la fitxa de la casella. ExcepcioCasellaBuida -assignarEstratègia(int i, int j, int size): EstrategiaPuntuacio Assigna una estratègia de puntuació a la casella indicada segons la seva posició

Nom	Taulell
Descripció	Representa el taulell de joc de Scrabble com una matriu de caselles. Cada casella pot contenir una fitxa i tenir una estratègia de puntuació associada.
Atributs	-int size -Casella[][] taulell
Mètodes	+Taulell(int size) Crea un nou taulell amb la mida especificada +Taulell(Taulell copiaTaulell)

	Inicialtiza aquest taulell com a copia d'un altre. +colocarFitxa(Fitxa fitxa, int fila, int columna) Mètode per assignar una fitxa a una casella del taulell. +retirarFitxa(int fila, int columna) Mètode per desassignar un fitxa d'una casella del taulell. +esBuit(): boolean Comprova si totes les caselles del taulell estan buides. -esJugadaHoritzontal(List<Casella> caselles): boolean Identifica si una jugada es juga en direcció horitzontal. -casellaPertanyAJugada(int x, int y, List<Casella> jugada): Casella Ens diu si una casella ha estat utilitzada en una jugada. -obtenirFitxaDeJugada(int x, int y, List<Casella> jugada): Fitxa Obté la fitxa jugada d'una casella utilitzada a la jugada. -posicioValida(List<Casella> casellesJugades): boolean Verifica si la posició de les caselles jugades és vàlida. -construirParaula(List<Casella> casellesJugades, boolean horitzontal): String Construeix la paraula principal formada per les caselles jugades. -validarParaulesPerpendiculars(List<Casella> casellesJugades, DAWG dawg, boolean horitzontal) : boolean Valida les paraules perpendiculars formades per les caselles jugades. -calcularPuntuacioParaulaPrincipal(List<Casella> casellesJugades, boolean horitzontal): int Calcula la puntuació de la paraula principal. -calcularPuntuacioPerpendiculars(List<Casella> casellesJugades, boolean horitzontal): int Calcula la puntuació de les paraules perpendiculars formades. +construirJugada(List<Casella> casellesJugades, DAWG dawg): Jugada Construeix una instància de la classe jugada, corresponent a la jugada del jugador. +construirJugadaBot(String paraulaFormada, List<Casella> casellesJugades, DAWG dawg, boolean horitzontal): Jugada Construeix una instància de la classe jugada, corresponent a la jugada del bot.

Nom	Jugador
Descripció	Representa un jugador de la partida. Conté el nom del jugador, la seva puntuació acumulada i el seu faristol amb fitxes disponibles.
Atributs	-String nom -int punts -Faristol faristol
Mètodes	+Jugador(String nom, Faristol faristol) Crea un nou jugador amb el nom especificat i un faristol assignat. +Jugador(Jugador copiaJugador) Inicialitza un jugador com a còpia d'un altre. +faristolPle() : boolean Comprova si el faristol del jugador està ple. +barrejarFaristol() Barreja aleatòriament les fitxes del faristol del jugador. +afegirPunts(int nousPunts) Afegeix punts a la puntuació del jugador. +eliminarPunts(int nousPunts) Resta punts de la puntuació del jugador. +afegirFitxa(Fitxa fitxa) Afegeix una fitxa al faristol del jugador. +eliminarFitxa() Elimina una fitxa del faristol del jugador. +esBot() : boolean Indica si el jugador és un bot. Per defecte, retorna fals.

Nom	Faristol
Descripció	Representa el faristol d'un jugador, on es col·loquen les fitxes disponibles durant la partida. El faristol té una mida fixa i pot contenir diverses fitxes que el jugador pot utilitzar.
Atributs	-int size -ArrayList<Fitxa> fitxes
Mètodes	+Faristol(int size) Crea un nou faristol amb una mida especificada. +Faristol(Faristol copiaFaristol) Inicialitza aquest faristol com la copia d'un altre.

	+afegirFitxa(Fitxa fitxa) Afegeix una fitxa al faristol. ExcepcioFaristolPle si el faristol ja ha arribat a la seva capacitat màxima +eliminarFitxa(Fitxa fitxa) Elimina una fitxa concreta del faristol. ExcepcioFaristolNoConteLaFitxa si la fitxa no és present al faristol +barrejarFitxes() Barreja aleatòriament les fitxes del faristol. +esPle() : boolean Comprova si el faristol està ple +esBuit() : boolean Comprova si el faristol és buit

Nom	Bot
Descripció	Jugador però controlat per la màquina seguint un algorisme. Hereta de Jugador i té un nivell de dificultat del 1 al 3.
Atributs	-int nivellDificultat
Mètodes	+Bot(String nom, Faristol faristol, int nivellDificultat) Crea un bot amb els paràmetres indicats. +Bot(Bot copiaBot) Inicialitza aquest bot com a còpia d'un altre. +esBot() : boolean Retorna vertader.

Nom	Jugada
Descripció	Classe que representa una jugada al joc de Scrabble. Emmagatzema la paraula formada, les caselles on s'han col·locat fitxes noves i la puntuació total.
Atributs	-String paraulaFormada -List<Casella> casellesJugades -int puntuacio -boolean jugadaValida
Mètodes	+Jugada(String paraulaFormada, List<Casella> casellesJugades, int puntuacio, boolean jugadaValida)

Nom	Torn
Descripció	Representa un torn dins d'una partida de Scrabble. Conté la informació necessària per gestionar el torn actual, incloent el sac de fitxes, el taulell, els jugadors, el número de torn i si el torn ha acabat.
Atributs	-Sac sac -Taulell taulell -Jugador[] jugadors -int numTorn -boolean acabada -int tornsSenseCanvis
Mètodes	+Torn(Sac sac, Taulell taulell, Jugador[] jugadors, int torn, boolean acabada, int int tornsSenseCanvis) Crea un nou torn amb els paràmetres indicats. +jugadorBot(int torn) : boolean Retorna *true* si el jugador del torn és un bot. +esAcabada() : boolean Indica si és l'últim torn de la partida

Nom	HistorialJoc
Descripció	Representa l'historial d'una partida de joc. Emmagatzema la llista de torns realitzats durant la partida, així com la data en què es va jugar.
Atributs	-List<Torn> torns -Date dataJoc
Mètodes	+HistorialJoc(Date date) Crea un nou historial de joc amb la data especificada. +afegirTorn(Torn torn) Afegeix un torn a l'historial de joc +retirarTorn() Elimina l'últim torn afegit a l'historial de joc. +retrirarTorns(int torn) Elimina els torns posteriors a un torn especific. +obtenirTorn(int index) : Torn Retorna el torn corresponent a una posició concreta de l'historial. La numeració dels torns comença per 1 (no per 0). +obtenirMida() : int Retorna el nombre total de torns registrats a l'historial.

Nom	EstrategiaPuntuacio [INTERFÍCIE]
Descripció	Interfície que defineix una estratègia de puntuació per a una casella del tauler. Les estratègies poden aplicar multiplicadors a lletres o paraules senceres, segons el tipus de casella.
Mètodes	+calcularPunts() : int Calcula els punts aportats per una fitxa segons l'estratègia de puntuació. +esMultiplicadorParaula() : boolean Indica si l'estratègia és de multiplicador de paraula.

Nom	EstrategiaNormal/ EstrategiaMultiplicadorLletra / EstrategiaMultiplicadorParaula
Descripció	Classes que implementen EstrategiaPuntuacio Estratègia normal (sense multiplicadors) Multiplicador de lletra (multiplica el valor de la lletra) Multiplicador de paraula (multiplica el valor de la paraula)
Atributs	-int multiplicador
Mètodes	+calcularPunts() : int Calcula els punts aportats per una fitxa segons l'estratègia de puntuació. +esMultiplicadorParaula() : boolean Indica si l'estratègia és de multiplicador de paraula.

Nom	Estadistiques [SINGLETON]
Descripció	Classe que gestiona les estadístiques d'una partida o conjunt de partides. Permet emmagatzemar les puntuacions dels jugadors, així com calcular la puntuació mínima, la puntuació total i la mitjana.
Atributs	-int puntuacioMinima -int puntuacioTotal -int puntuacioMitjana
Mètodes	+Estadistiques() Constructor per defecte que inicialitza les estadístiques. Estableix la puntuació total i mitjana a 0, i la mínima al valor màxim d'enter. Crea una cua de prioritat per emmagatzemar les puntuacions, ordenades de major a menor valor. +afegirPuntuacio(int puntuacio, String jugador) Afegeix una puntuació per a un jugador determinat. Si la puntuació és vàlida (no negativa), s'actualitzen la puntuació total, la mínima i s'afegeix la nova entrada a la cua de prioritats.

	+calcularPuntuacioMitjana() Calcula la puntuació mitjana dels jugadors i l'emmagatzema. La mitjana es calcula dividint la puntuació total entre el nombre de jugadors. Si no hi ha cap puntuació registrada, no es fa cap càlcul. +retirarPuntuacio(String jugador, int punts) Retira una quantitat de punts a un jugador i actualitza les estadístiques globals. Si la nova puntuació es menor o igual a 0, s'elimina al jugador del registre. +eliminarJugador(String jugador) Elimina un determinat jugador de totes les estadístiques.

Nom	Node
Descripció	És una classe interna de DAWG i representa cada un dels punts del graf. Cada node pot contenir diversos arcs sortints cap a altres nodes, identificats per tokens (lletres o dígrafs) i marca si és el final o no d'algun prefix.
Atributs	-Map<String, Node> fills -boolean esFinal
Mètodes	+getFill(String token) : Node +getFills() : Map<String, Node> +conteToken(String token) : boolean

Nom	DAWG
Descripció	DAWG (Directed Acyclic Word Graph) que representa un diccionari de paraules a on cada node té un conjunt de transicions etiquetades amb tokens (caràcters o dígrafs) cap a altres nodes tot formant paraules.
Atributs	-Node arrel -Map<Node, Node> nodesMinimitzats -List<String> tokens -List<Node> pathAnterior
Mètodes	+DAWG(List<String> tokens, List<String> paraules) Constructor. Crea i construeix el DAWG a partir de tokens i paraules donades. -afegirParaula(String paraula, String paraulaAnterior) Afegeix una nova paraula al DAWG comparant-la amb la paraula anterior per detectar el prefix comú i reutilitzar-lo. En cas que sigui diferent el prefix crearà una nova ruta amb un node diferent +conteParaula(String paraula) : boolean Comprova si una paraula completa existeix dins del DAWG

	-minimitzarSufix(int desDe) Minimitza els nodes de la ruta afegida recentment (pathAnterior) començant des de la posició 'desDe'. Aquest mètode substitueix subgràfics equivalents per versions ja existents usant el mapa de nodes minimitzats. Això assegura que el DAWG sigui minimal, reutilitzant subestructures comunes. -tokenitzar(String paraula) : List<String> Divideix una paraula en una llista de tokens (Caràcters/Dígrafs) segons els tokens càrregats al DAWG. +esPrefix(String prefix) : Node Comprova si un prefix donat existeix en l'estructura DAWG i retorna el node corresponent al final del prefix si existeix.

3. Relació implementació membres de l'equip

Toni	Roger	Camila	Jan
Jugador i Bot Estadístiques Casella Colors Estrategia Puntuació	Ctrl Jugada Bot Taulell DAWG Jugada	Ctrl Partida Driver Partida Historial Joc Sac Torn	Ctrl Domini Ctrl Estadística Driver Estadística Faristol Fitxa

4. Estructures de dades i algorismes

4.1. Preàmbul

Per desenvolupar el nostre projecte de Scrabble, hem pensat en una estructura basada en classes que representin els diferents elements reals del joc. L'objectiu ha estat mantenir una separació clara de responsabilitats i reflectir de la millor manera possible com funciona un Scrabble real. Hem volgut que el codi fos entenedor, fàcil de modificar i ampliable en cas que en el futur vulguem afegir més funcionalitats o millorar la intel·ligència del bot.

Per representar les fitxes, hem creat la classe Fitxa, que conté la informació essencial de cada peça del joc: la lletra (o dígraf, com ara "NY") i la puntuació que li correspon. Aquesta classe és simple però imprescindible, ja que les fitxes són els elements que es col·loquen al taulell i que acaben definint les jugades i els punts.

El taulell del nostre Scrabble està format per una graella de caselles, com en la versió clàssica del joc. Ara bé, no totes les caselles són iguals: algunes tenen efectes especials que afecten la puntuació, com per exemple multiplicar el valor d'una lletra o d'una paraula. Per gestionar aquesta varietat de comportaments de manera clara i flexible, hem fet servir una interfície anomenada Multiplicador.

La interfície EstratègiaMultiplicador defineix el comportament comú que poden tenir totes les caselles que modifiquen la puntuació. Les diferents classes de casella implementen aquesta interfície segons l'efecte que han de tenir. Així doncs, hem creat tres tipus concrets de caselles: CasellaNormal, MultiplicadorLletra i MultiplicadorParaula.

La classe CasellaNormal representa les caselles estàndard, que no apliquen cap modificador, i per tant implementen EstratègiaMultiplicador retornant sempre un valor neutre (és a dir, multiplicador 1). Les caselles de tipus MultiplicadorLletra implementen la interfície per retornar el factor pel qual s'ha de multiplicar la puntuació de la lletra col·locada (per exemple, x2 o x3). De manera similar, les caselles de tipus MultiplicadorParaula apliquen un multiplicador a tota la paraula formada quan una de les seves fitxes cau en aquesta casella.

Un aspecte a tenir en compte és que si la casella és de tipus multiplicador de paraula, aquest efecte no s'aplica en el mètode calcularPunts(), ja que no té sentit considerar-lo de manera aïllada. El multiplicador de paraula només entra en joc quan s'ha completat tota una paraula: és en aquest moment que es calcula la puntuació total sumant les aportacions de cada casella i, si alguna d'aquestes és un multiplicador de paraula, s'hi aplica el factor corresponent al resultat global.

El fet que cada casella tingui una sola estratègia ben definida ens ha permès mantenir el codi net, fàcil de llegir i molt extensible. En lloc de tenir una lògica complexa dins d'una única classe amb molts condicionals, hem separat el comportament en diferents classes que comparteixen una interfície comuna. Aquesta estratègia s'ha demostrat molt útil a l'hora de calcular la puntuació de les jugades, ja que només cal demanar a la casella quin multiplicador aplica segons el tipus que és.

Aquest enfocament també ens facilita molt les possibles ampliacions del joc. Si en el futur volguéssim afegir nous tipus de casella amb comportaments més avançats, només caldria crear una nova classe que implementi la interfície Multiplicador amb la lògica corresponent.

Els Jugadors són una part clau del joc, i per això tenen la seva pròpia classe. Cada jugador té un nom, una puntuació i un faristol, que és el conjunt de fitxes que pot utilitzar en cada torn. El jugador ha de poder veure les seves fitxes, col·locar paraules, passar torns o intercanviar fitxes. Per representar el faristol hem fet una classe pròpia, que facilita molt la gestió de les fitxes del jugador.

Un cas particular de jugador és el Bot, que hem implementat com una subclasse de Jugador. Això ens permet reutilitzar bona part del codi del jugador humà, però redefinint certs comportaments, com ara la manera de decidir quina paraula col·locar. El bot pot tenir una lògica senzilla o més avançada segons l'atribut del nivell de dificultat.

Finalment, hi ha el Sac de fitxes, que conté totes les fitxes disponibles al joc. Aquest sac s'utilitza per repartir les fitxes inicials als jugadors i per omplir els faristols després de cada torn. La classe gestiona quantes fitxes queden, i retorna fitxes de manera aleatòria, tal com passa en el joc real. També té un conjut de Strings amb les lletres "originals" del sac. Això ens permet saber si una lletra o dígraf pertany al sac o no. Per exemple, si el jugador inicialitza el sac en català quan vulgui intercanviar un comodí amb el dígraf CH se li prohibirà, ja que aquest no pertany a les lletres originals del sac.

Amb aquest conjunt de classes hem aconseguit una estructura clara i lògica que reflecteix molt bé les mecàniques del joc de Scrabble. Cada peça del joc està representada per un objecte, i el comportament es reparteix entre les classes de forma coherent. Això ens ha ajudat molt tant en el procés de desenvolupament com en les proves i l'extensió del codi.

La classe Estadístiques s'encarrega de gestionar les estadístiques globals del sistema de joc, com les puntuacions dels jugadors, el rànquing ordenat i la puntuació total acumulada. Aquesta classe segueix el patró de disseny Singleton, de manera que només existeix una única instància durant tota l'execució del programa. Aquesta decisió es justifica pel fet que les estadístiques han de ser globals i consistents: no tindria sentit tenir estadístiques duplicades o disperses, i el Singleton permet accedir fàcilment a la instància única des de qualsevol part del codi.

4.2. Estructures de les classes

Sac com a Multiset<Fitxa>

Per representar el sac de fitxes, hem optat per utilitzar un Multiset<Fitxa>, una estructura que ens permet tenir múltiples còpies de la mateixa fitxa i fer operacions sobre elles de manera eficient i natural. Aquesta tria és coherent tant amb la naturalesa del joc com amb els costos i el tipus d'operacions que hi fem habitualment.

En primer lloc, cal tenir en compte que el sac no és un conjunt de fitxes úniques, sinó que conté diverses còpies d'una mateixa lletra amb puntuacions associades. Un Multiset ens permet modelar això de forma directa, ja que internament manté un recompte de quantes vegades apareix cada element.

Pel que fa als costos, amb un Multiset, afegir o treure fitxes és una operació d'ordre O(1) en la mitjana, igual que consultar la quantitat d'un element concret. A més, iterar sobre totes les fitxes (tenint en compte les repeticions) és lineal respecte al total d'elements, no només al nombre de claus diferents.

Si en lloc d'un Multiset haguéssim utilitzat un Map<Fitxa, Integer>, també podríem haver representat la quantitat de cada fitxa, però la gestió hauria estat més feixuga i menys intuïtiva. Tot i que operacions com afegir o consultar una fitxa concreta serien igualment ràpides (O(1) en mitjana amb un HashMap), eliminar una fitxa o reduir-ne el comptador implicaria escriure més lògica manual. A més, hauríem d'estar pendents de quan una fitxa arriba a zero per eliminar-la explícitament del mapa.

Per tant, el Multiset és no només més natural des del punt de vista conceptual, sinó també més eficient i més senzill de gestionar pel tipus d'operacions que fem: seleccions aleatòries, insercions, eliminacions i consultes sobre quantes fitxes queden.

Taulell com a matriu de Caselles

La representació del taulell com una matriu de caselles és una elecció molt natural i coherent, tant des del punt de vista conceptual com pel que fa a l'eficiència i simplicitat del codi.

D'entrada, el taulell del Scrabble té una estructura clarament bidimensional i de mida fixa on cada posició és una casella que pot contenir o no una fitxa, i que pot tenir propietats especials (com multiplicadors de lletra o de paraula). Utilitzar una matriu ens permet reflectir fidelment aquesta organització: cada casella queda identificada per unes coordenades (fila, columna) que corresponen de manera directa a la seva posició real al taulell.

Des del punt de vista d'implementació, accedir a una casella concreta amb una matriu és extremadament eficient: tenim accés constant O(1) a qualsevol posició, la qual cosa és important quan es vol comprovar ràpidament si una casella està buida, aplicar una jugada, o calcular la puntuació d'una paraula col·locada al taulell. Aquest accés directe també facilita la validació de moviments, la cerca de paraules formades, i la visualització general del taulell.

A més, com que la mida del taulell és coneguda i fixa, no cal cap estructura dinàmica ni cap optimització especial per gestionar l'espai. La matriu cobreix totes les necessitats de manera simple i clara, i facilita molt la comprensió del codi i el seu manteniment.

Estadísiques com a Map i TreeMap

Pel que fa al cost computacional de les operacions més habituals, afegir o retirar puntuació d'un jugador té un cost de O(log n), degut al fet que cal modificar el TreeMap que manté l'ordre del rànquing. Consultar la puntuació mitjana o accedir a la millor o pitjor puntuació és molt eficient, ja que es pot fer en temps constant o logarítmic segons el cas. Aquesta gestió eficient i ordenada de les estadístiques, facilita el manteniment del codi i assegura una bona escalabilitat si el nombre de jugadors augmenta.

La separació entre dues estructures —el HashMap<String, Integer> puntuacions i el TreeMap<Integer, Set<String>> ranking— és totalment intencionada i té diversos motius importants, tant pel rendiment com per la claredat del codi.

D'una banda, el HashMap permet consultar ràpidament la puntuació d'un jugador concret a partir del seu nom. Aquesta operació és molt freqüent (per exemple, quan vols afegir punts o restar-ne a un jugador concret), i per això interessa que sigui molt eficient: el HashMap ofereix accés en temps constant, O(1) de mitjana.

D'altra banda, el TreeMap manté el rànquing ordenat de puntuacions. És a dir, ens permet saber en tot moment quin jugador (o jugadors) tenen la puntuació més alta o més baixa, sense haver de recórrer tota la col·lecció. El TreeMap, en ordenar automàticament les claus, permet accedir directament al millor (firstEntry()) o pitjor (lastEntry()) en temps logarítmic, O(log n).

També es manté una variable sencera (int puntuacioTotal) per acumular la suma de totes les puntuacions. Això evita haver de recórrer el mapa sencer cada vegada que es vol obtenir la mitjana, cosa que milloraria el rendiment especialment en sistemes amb molts jugadors.

Quan s'afegeix una puntuació amb el mètode afegirPuntuacio, si el jugador ja existeix, primer s'elimina la seva puntuació antiga del rànquing i es resta de la puntuació total. A continuació, es calcula la nova puntuació, s'actualitza al HashMap i es torna a inserir al TreeMap segons la seva nova puntuació. Si el jugador no existia, simplement s'afegeix amb la seva puntuació inicial a ambdues estructures i s'actualitza la puntuació total. Així, les dues estructures es mantenen sincronitzades.

Quan s'elimina un jugador amb eliminarJugador, es comprova si existeix; si és així, es treu del HashMap, es resta la seva puntuació de la total i també s'elimina del conjunt corresponent del TreeMap. Si aquell conjunt de jugadors queda buit, es retira també la clau corresponent. Tot plegat assegura que les estadístiques siguin precises i que no quedin dades residuals.

Estructura DAWG

El DAWG (Directed Acyclic Word Graph) és una estructura de graf dirigit acíclic dissenyada per representar un conjunt de paraules vàlides de manera eficient. En aquest projecte, hem dividit la seva implementació en dues parts principals: la classe externa DAWG, que gestiona la construcció i navegació del graf, i la classe interna Node, que representa els estats individuals dins d'aquest graf.

Classe Node:

Cada instància de Node representa un estat parcial d'una paraula. Conté les següents estructures:

- **Map<String, Node> fills**
  Aquest mapa defineix les transicions sortints des del node actual. Cada clau és un token (pot ser una lletra o un dígraf), i el valor associat és el node fill corresponent. Hem utilitzat un HashMap ja que ens cal garantir un accés ràpid (O(1) de mitjana) a les transicions, ja que cal consultar amb freqüència si existeix una connexió concreta a partir d'un token determinat.
- **boolean esFinal**
  Indica si el camí des de l'arrel fins aquest node forma una paraula vàlida completa del diccionari. Aquesta marca és fonamental per la validació de jugades.

DAWG:

La classe externa DAWG s'encarrega de construir i gestionar tot el graf. Les seves estructures principals són:

- **Node arrel**
  Representa el node inicial del graf. Totes les cerques i insercions comencen a partir d'aquest node.
- **Map<Node, Node> nodesMinimitzats**
  Aquest mapa s'utilitza durant la construcció del DAWG per detectar i reutilitzar subgràfics equivalents ja existents.

Usem un Map (i no un Set) perquè, un cop detectat un node equivalent, cal recuperar-lo explícitament per substituir l'original. Un Set en canvi permetria saber si ja existeix, però no accedir-hi de manera directa.

- **List<String> tokens**
  Conté els tokens vàlids (lletres o dígrafs) per a la tokenització de paraules.
  Hem optat per una List en lloc d'un Set perquè l'ordre d'aquests tokens és rellevant. Durant la tokenització, s'intenta trobar el primer token que coincideixi amb l'inici de la cadena, i per això cal preservar un ordre personalitzat (per exemple, prioritzar "ny" abans que "n").
- **List<Node> pathAnterior**
  Guarda el camí complet (els nodes) corresponent a l'última paraula afegida al graf.
  Aquest camí és necessari per a la minimització de sufixos, ja que permet identificar el punt de divergència entre dues paraules consecutives i aplicar l'optimització només on cal.

Torn i HistorialJoc

La classe Torn modela l'estat complet d'una partida de Scrabble en un moment determinat. Emmagatzema còpies independents del sac de fitxes, el taulell, els jugadors i informació com el número del torn, si la partida ha acabat i el nombre de torns consecutius sense canvis. Això permet preservar l'estat exacte de la partida en aquell instant, evitant efectes col·laterals, i facilita operacions com desfer accions o analitzar el desenvolupament del joc. La classe també permet consultar si el jugador actual és un bot i accedir als elements clau del torn.

La classe HistorialJoc serveix per registrar l'evolució cronològica d'una partida mitjançant una llista de torns. A cada acció del jugador, es pot afegir un nou torn, eliminant-se o modificant-se posteriorment si cal. També inclou la data de la partida per a finalitats de registre o visualització. Aquesta estructura és essencial per recuperar l'estat del joc en un punt concret, permetent funcionalitats com desfer accions, revisar partides jugades o implementar modes d'anàlisi i repetició.

Jugada

La classe Jugada concentra tota la informació rellevant d'una acció al taulell en un únic objecte, evitant haver de passar múltiples paràmetres entre mètodes i facilitant tant la llegibilitat com el manteniment del codi.

Conté la paraula formada, per mostrar-la a l'usuari per pantalla, una llista amb les caselles jugades, la seva puntuació corresponent i finalment un booleà que indica si la jugada és vàlida.

4.3. Algorisme creació DAWG

4.3.1. Avantatges d'usar un DAWG

Per tal d'emmagatzemar totes les paraules vàlides d'un diccionari o temàtica, de forma compacta i eficient hem utilitzat un **DAWG** (Directed Acyclic Word Graph), tal i com marcava l'enunciat del projecte.

Un DAWG està format per nodes connectats mitjançant arcs dirigits, on cada camí des de l'arrel fins a un node final forma una paraula del diccionari. L'objectiu del DAWG és aprofitar aquelles paraules que comparteixen una part comuna perquè comparteixin nodes dins del graf i optimitzar tant l'espai com el temps de consulta.

A diferència d'un trie (arbre de caràcters, que forma paraules), un DAWG aplica un procés de minimització per detectar i reutilitzar subgràfics idèntics. Això permet evitar duplicació de sufixos, reduint dràsticament la memòria necessària i accelerant les operacions de cerca.

El DAWG també ens permet saber donat un prefix, totes aquelles lletres que al ajuntar-les amb el prefix formen nous prefixos de paraules vàlides, d'aquesta manera el bot en comptes de generar totes les combinacions possibles de lletres, pot navegar pel DAWG i només explorar aquells camins que porten a una paraula vàlida.

4.3.2. Construcció del DAWG

El procés de construcció es duu a terme de manera directa, és a dir, no es genera un trie per després minimitzar-lo, sinó que el graf es construeix ja minimitzat en temps real.

Per construir el DAWG tan sols necessitem:

- Una llista de **tokens** (caràcters o dígrafs vàlids).
- Una llista de **paraules vàlides**, **ordenades alfabèticament** com a precondició.

Per cada una de les paraules vàlides farem els següents passos:

1. **Tokenitzar la paraula:** Es divideix la paraula en tokens utilitzant la llista de dígrafs/caràcters.
    - Tokens: ["NY", "L·L", "A", "B", "C", ...]
    - Paraula: "NYANYO" → ["NY", "A", "NY", "O"]

    L'algorisme intenta emparellar primer els tokens més llargs, per assegurar que es prioritzen els dígrafs.

2. **Buscar el node divergent:** Es compara la paraula actual amb la paraula anterior per trobar el punt en què difereixen.

Paraula Completa		Token 0	Token 1	Token 2
Paraula anterior:	NYAUFO	NY	A	U
Nova paraula:	NYANYO	NY	A	NY
Iguals		✅	✅	❌

No ens cal revisar altres tokens amb paraules anteriors ja que sabem que cap paraula reusarà sufixos d'una que no sigui l'anterior si la llista de paraules està ordenada.

3. **Minimitzar el sufix:** A partir del punt de divergència, es minimitzen els nodes anteriors per identificar subgràfics equivalents.

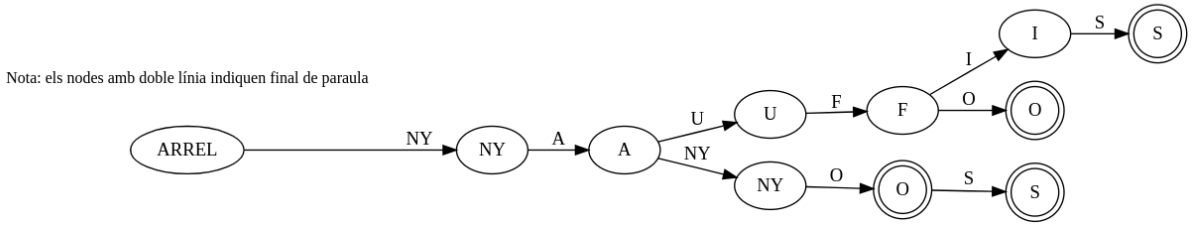
   Si ja existeix un node equivalent, (mateixa estructura i esFinal) es reutilitza. Això evita crear sufixos redundants i garanteix que el DAWG és minimal.

4. **Afegir els nous nodes**: A partir del punt de divergència, es crea un nou camí de nodes, un per cada token restant de la paraula. Aquests nodes s'enllacen amb l'anterior i s'afegeixen al camí actual.

5. **Marcar el node final:** L'últim node afegit es marca com a final. Això indica que el camí des de l'arrel fins aquest node representa una paraula vàlida del diccionari.

DAWG format amb les paraules "NYAUFIS", "NYAUFO", "NYANYO" i "NYANYOS"

Nota: els nodes amb doble línia indiquen final de paraula



4.3.3. Vida útil del DAWG

El DAWG tan sols triga entre 5 i 10 segons a generar-se (depenent de la quantitat de paraules i tokens, i de la màquina que executi el programa).
Només s'usa en la generació de paraules del bot i en la validació de jugades dels usuaris.

Per aquests motius i que el diccionari pot canviar segons la partida, hem decidit que el DAWG es generi un cop s'inicia la partida i que es mantindrà a memòria fins que: o bé s'acabi la partida, o bé es tanqui l'aplicació.

4.4. Algorisme Jugades Bot

Per a implementar la intel·ligència del jugador automàtic (bot), s'ha dissenyat un sistema que pot adaptar-se a diferents nivells de dificultat. El bot pot escollir la millor jugada possible (nivell 3), una jugada intermedia (nivell 2) o la pitjor jugada entre les vàlides (nivell 1). La selecció es basa en la puntuació obtinguda d'entre totes les jugades vàlides possibles.

4.4.1. Algorisme de Backtracking

Per tal de generar totes les jugades possibles, hem utilitzat un algorisme de **backtracking**. Aquest algorisme consisteix en cercar totes les combinacions possibles que compleixin certes condicions. En el nostre cas, l'objectiu serà cercar totes les **jugades vàlides** que pot fer un jugador sabent l'estat actual del taulell i les fitxes que té a la seva disposició per jugar.
L'algorisme de backtracking recorre el taulell casella per casella, intentant construir totes les paraules (primer horitzontalment i després verticalment).

Per cada posició inicial construeix el prefix existent a partir de les fitxes que ja es troben col·locades en el taulell abans de la posició actual.
A continuació, explora totes les combinacions possibles de fitxes del faristol que poden continuar aquest prefix.

Si la casella està buida, per a cada fitxa disponible al faristol s'hi col·loca una fitxa temporalment. Es construeix un nou prefix a partir de:

- Si la casella està buida amb una fitxa disponible del faristol.
- Si la casella està ocupada amb s'afegeix la fitxa que l'ocupa

Es comprova si aquest nou prefix pot conduir a una paraula vàlida fent ús del DAWG (vegeu secció següent).
Si el prefix és vàlid, es continua recursivament col·locant més fitxes.

Quan s'acaba l'exploració d'aquella branca, es retira la fitxa del faristol i es restaura l'estat anterior per poder provar amb una altra fitxa.

4.4.2. Ús del DAWG per Podar la Recerca

Per evitar generar combinacions que no poden formar cap paraula vàlida, entra en joc el nostre DAWG. Com hem explicat anteriorment aquest conté el conjunt de paraules vàlides del diccionari i ens permet comprovar totes les ramificacions possibles per cada prefix.

Cada vegada que s'afegeix una nova fitxa a una combinació parcial, es comprova si el prefix resultant existeix al DAWG. Si no és així, es **poda** (és a dir, s'abandona aquella possible combinació i es va a provar una altre), estalviant temps i evitant recorreguts inútils. Això millora significativament l'eficiència de l'algorisme.

4.4.3. Validació i Classificació de Jugades

Un cop es genera una combinació que forma una paraula present al DAWG, es valida que compleixi amb les regles del joc Scrabble:

- Estar col·locada al centre si és la primera jugada i estar connectada amb alguna fitxa existent del taulell si ja hi han paraules jugades.
- Formar paraules vàlides amb tots els seus encreuaments tant verticals com horitzontals.
- Si la jugada és vàlida, se'n calcula la puntuació tenint en compte els multiplicadors del taulell i bonificacions com jugar totes les fitxes.

Finalment totes les jugades vàlides es recullen en una llista i es classifiquen per puntuació on segons el nivell de dificultat, se selecciona la jugada adecuada.

5. Controladors

5.1. Preàmbul

L'arquitectura de controladors del joc d'Scrabble que hem implementat segueix un disseny ben estructurat que separa les responsabilitats en components especialitzats, facilitant així la mantenibilitat i escalabilitat del sistema. Amb aquesta organització els controladors actuen com a intermediaris entre la lògica del domini i la interfície d'usuari.

El nucli d'aquesta estructura és el Controlador de Domini (CtrDomini), que funciona com a punt central de coordinació, canalitzant la comunicació entre els diferents components del sistema i proporcionant un punt de comunicació unificado per a la capa de presentació. Aquest controlador delega responsabilitats específiques a controladors més especialitzats mentre manté la cohesió del sistema en el seu conjunt.

La gestió de partides s'encapsula en el Controlador de Partida (CtrlPartida), que administra tot el cicle de vida d'una partida des de la seva inicialització fins a la seva finalització. S'ocupa de gestionar l'estat del joc, incloent el tauler, els jugadors, el sac de fitxes, i l'historial de jugades, a més d'implementar les regles del joc i validar les jugades mitjançant el diccionari DAWG.

L'aspecte estadístic del joc se separa en el Controlador d'Estadística (CtrEstadistica), que manté un registre històric del rendiment dels jugadors al llarg de múltiples partides. Aquesta separació permet que les estadístiques persisteixin independentment de l'estat de les partides actives.

Finalment, la intel·ligència artificial del joc s'aïlla en el Controlador de Jugada Bot (CtrlJugadaBot), que implementa l'algorisme per calcular les millors jugades possibles segons diferents nivells de dificultat. Aquesta separació permet modificar o millorar la IA sense afectar la resta del sistema.

Tots aquests controladors implementen el patró Singleton, assegurant que existeixi una única instància de cadascun al llarg de l'execució del programa, la qual cosa facilita la gestió de l'estat i evita inconsistències. L'estructura general reflecteix una aplicació efectiva del principi de responsabilitat única, on cada component té una funció clarament definida dins del sistema, cosa que contribueix a un codi més net, més fàcil d'entendre i més senzill de mantenir o estendre en el futur.

5.2. Controlador Domini

5.2.1. Descripció

Classe principal del domini que actua com a controlador general per gestionar les partides, els bots i les estadístiques del Scrabble. Coordina els tres controladors principals (CtrlEstadistica, CtrlPartida, CtrlJugadaBot) i ofereix els mètodes necessaris per a poder jugar una partida.

5.2.2. Atributs

- **ctrlEstadistica** (CtrlEstadistica): Controlador que gestiona totes les estadístiques del joc.
- **ctrlPartida** (CtrlPartida): Controlador que gestiona tot el funcionament d'una partida.
- **ctrlJugadaBot** (CtrlJugadaBot): Controlador que gestiona les jugades que efectuarà un bot.

5.2.3. Mètodes

- **obtenirPuntuacioMitjana**(): Retorna la puntuació mitjana de tots els jugadors registrats.
- **obtenirPuntuacioTotal**(): Retorna la suma de les puntuacions de tots els jugadors.
- **obtenirPuntuacioMaxima**(): Retorna la puntuació més alta registrada.
- **obtenirPuntuacioMinima**(): Retorna la puntuació més baixa registrada.
- **obtenirPuntuacions**(): Retorna un Mapa amb les puntuacions i els noms de tots els jugadors.
- **inicialitzarPartida**(int midaTaulell, int midaFaristol, String idioma, String[] nomsJugadors, int[] dificultatsBots): Inicialitza una nova partida amb els paràmetres indicats.
- **carregarPartida**(Torn torn): Carrega una partida a partir d'un estat de torn guardat. D'aquesta manera es pot reprendre una partida ja iniciada.
- **obtenirTorn**(): Retorna el número del torn actual.
- **obtenirTaulell**(): Retorna el taulell del joc actual.
- **obtenirJugadors**(): Retorna els jugadors participants de la partida.
- **obtenirJugadorActual**(): Retorna el jugador que ha de jugar el torn acutal.
- **obtenirSac**(): Retorna el sac de fitxes actual.
- **colocarFitxa**(String lletra, int i, int j): Coloca una fitxa del faristol del jugador actual en una posició indicada en els paràmetres.
- **retirarFitxa**(int i, int j): Retira la fitxa situada en la posició indicada.
- **ferUndo**(): Desfà l'última acció realitzada.

- **acabarPartida**(): Acaba la partida actual.
- **passarTorn**(): Passa el torn al següent jugador.
- **canviarFitxes**(String[] fitxesCanviades): Canvia un conjunt de fitxes del faristol del jugador actual per noves del sac.
- **commitParaula**(): Confirma la paraula formada en el taulell.
- **partidaAcabada**(): Comprova si la partida ha acabat.
- **esFinalDePartida**(): Comprova si es el final de la partida.
- **jugadaBot**(): Realitza la jugada del bot.
- **resetTorn**(): Es torna a l'estat inicial del torn.
- **setLletraComodi**(Fitxa fitxa, String lletra): Canvia la lletra comodí per la indicada en el paràmetre lletra.

5.3. Controlador Partida

4.3.1 Descripció

Controlador principal encarregat de gestionar el desenvolupament d'una partida de Scrabble. S'encarrega de la creació i inicialització del taulell, el sac de fitxes, el conjunt de jugadors (incloent bots), l'historial de jugades i el diccionari (DAWG). També gestiona els torns, controla l'estat del joc, permet desfer accions, i determina quan la partida ha d'acabar.

4.3.2 Atributs

- **instance** (CtrlPartida): Instància singleton del controlador de partida.
- **ctrlBot** (CtrlJugadaBot): Controlador encarregat de gestionar les jugades dels bots.
- **historial** (HistorialJoc): Historial de tots els torns jugats durant la partida.
- **sac** (Sac)**:** Sac de fitxes utilitzat per distribuir lletres als jugadors.
- **taulell** (Taulell): Taulell on es desenvolupen les jugades.
- **jugadors** (Jugador[]): Array de jugadors que participen en la partida.
- **dawg** (DAWG): Diccionari per validar paraules segons l'idioma.
- **acabada** (boolean): Indica si la partida ha finalitzat.
- **torn** (int): Número de torn actual.
- **casellesTorn** (List<Casella>): Caselles modificades durant el torn actual.
- **jugadaActual** (Jugada): Jugada en curs del torn actual.
- **tornsSenseCanvi** (int): Comptador de torns consecutius sense jugades que modifiquin el taulell.

4.3.3 Mètodes

- **getInstance**(): Retorna la instància singleton del controlador de partida.
- **inicialitzarPartida**(int midaTaulell, int midaFaristol, String idioma, String[] nomsJugadors, int[] dificultatsBots): Inicialitza una nova partida de Scrabble configurant tots els components essencials: reinicia l'estat del joc, crea el taulell i el sac de fitxes segons l'idioma, carrega el diccionari, genera els jugadors (incloent bots), assigna els faristols, crea l'historial de jugades i estableix el primer torn.
- **inicialitzarTaulell**(int midaTaulell): Crea i inicialitza el taulell amb la mida indicada.
- **inicialitzarSac**(String idioma): Llegeix el fitxer corresponent a l'idioma i omple el sac amb les fitxes configurades.
- **inicialitzarJugadors**(String[] nomsJugadors, int[] dificultatsBots, int midaFaristol): Crea els jugadors (humans i bots) i assigna els seus faristols inicials.
- **inicialitzarCtrlBot**(): Inicialitza el controlador de jugades dels bots.
- **inicialitzarDawg**(String idioma): Llegeix i carrega les paraules vàlides i símbols de l´idioma seleccionat per la generació del DAWG.
- **inicialitzarFaristol**(Jugador jugador): Omple el faristol del jugador amb fitxes del sac fins que estigui ple.
- **recuperarTorn**(Torn nouTorn): Recupera l'estat del joc a partir d'un torn prèviament guardat.
- **resetTorn**(): Restaura l'estat del torn actual a com estava a l'inici.
- **acabarPartida**(): Finalitza la partida i ordena els jugadors segons la seva puntuació.
- **passarTorn**(): Passa el torn al jugador següent i actualitza l'historial.
- **undo**(): Desfà l'últim torn fet per un jugador humà, si és possible.
- **esPotFerUndo**(): Comprova si es pot fer undo segons l'estat de la partida i el tipus de jugadors.
- **esFinalDePartida**(): Determina si es compleixen les condicions per finalitzar la partida.
- **acabada**(): Retorna si la partida ha finalitzat o no.
- **obtenirSac**(): Retorna el sac de fitxes actual.
- **obtenirTaulell**(): Retorna el taulell actual.
- **obtenirJugadors**(): Retorna els jugadors participants.
- **obtenirJugadorActual**(): Retorna el jugador al qual li toca jugar.
- **obtenirTorn**(): Retorna el número de torn actual.
- **agafarDelSac**(): Agafa una fitxa del sac i l'afegeix al faristol del jugador en curs.
- **ordenarJugadors**(): Ordena els jugadors de la partida en funció de la seva puntuació, de major a menor.

- **setLletraComodi**(Fitxa fitxa, String lletraComodi): Permet al jugador canviar la lletra d'una fitxa comodí per la que hagi escollit, sempre que sigui vàlida.
- **canviarFitxes**(String[] fitxesCanviades): Canvia les fitxes que el jugador ha escollit per noves fitxes aleatòries del sac, i passa el torn al següent jugador.
- **colocarFitxa**(String fitxa, int fila, int columna): Coloca una fitxa en una casella del taulell i registra la jugada.
- **retirarFitxa**(int fila, int columna): Retira una fitxa de la casella especificada del taulell, la retorna al faristol del jugador i registra la jugada .
- **rellenarFaristol**(): Reomple el faristol del jugador actual amb fitxes del sac fins que torni a estar ple o no quedin més fitxes disponibles.
- **commitParaula**(): Registra la jugada com a vàlida, suma els punts al jugador (incloent-hi 50 punts extres si ha utilitzat totes les fitxes), reomple el faristol, reinicia el comptador de torns sense canvi i avança al següent torn.
- **jugadaBot**(): Executa la jugada del bot en funció del seu nivell de dificultat, col·loca les fitxes al taulell, les elimina del faristol, i registra la jugada com a vàlida.

5.4. Controlador JugadaBot

5.4.1. Descripció

El controlador JugadaBot és l'encarregat de generar automàticament les jugades del jugador bot en el joc de Scrabble. El seu objectiu és calcular la millor/intermèdia/pitjor jugada possible a partir de l'estat actual del taulell i les fitxes disponibles d'un bot, tenint en compte el nivell de dificultat establert.

L'algorisme implementat es basa en backtracking amb poda mitjançant l'estructura DAWG. Per més informació vegeu algorisme Bot.

5.4.2. Mètodes

- **calcularJugada**(Taulell taulell, DAWG dawg, int nivellDificultat, List<Fitxa> fitxes)
  Calcula i retorna una jugada vàlida segons el nivell de dificultat.

- **generarJugades**(Taulell taulell, DAWG dawg, List<Fitxa> fitxes)
  Genera totes les jugades vàlides possibles des de qualsevol posició del taulell.

- **generarParaules**(String prefix, List<Fitxa> fitxes, List<Casella> caselles, DAWG.Node node, int fila, int col, boolean horitzontal, Taulell taulell, DAWG dawg)
  Genera recursivament paraules vàlides a partir d'un prefix.

- **processarFitxaIContinuar**(String prefix, Fitxa fitxa, List<Fitxa> fitxes, List<Casella> caselles, DAWG.Node node, int fila, int col, boolean horitzontal, Taulell taulell, DAWG dawg, List<Jugada> resultats)
  Afegeix una fitxa i continua generant paraules amb recursió.

5.5. Controlador Estadístiques

5.5.1. Descripció

Aquest és el controlador encarregat de gestionar les operacions relacionades amb les estadístiques del sistema. Permet donar d'alta, eliminar i consultar jugadors i les seves respectives puntuacions, així com consultar estadístiques globals. Tot això es fa mitjançant el maneig dels atributs de la classe Estadistiques.

5.5.2. Atributs

- **estadistiques**(Estadistiques): Objecte que encapsula i gestiona les estadístiques del sistema.

5.5.3. Mètodes

- **afegirPuntuacio**(String jugador, int puntuacio): Afegeix una puntuació al jugador especificat i actualitza les estadístiques globals.
- **retirarPuntuacio**(String jugador, int punts): Resta una quantitat de punts a un jugador i actualitza les estadístiques globals.
- **eliminarJugador**(String jugador): Elimina totes les estadístiques associades al jugador especificat.
- **obtenirMillorJugador**(): Retorna el nom del jugador amb la puntuació més alta.
- **obtenirPitjorJugador**(): Retorna el nom del jugador amb la puntuació més baixa.
- **obtenirPuntuacioTotal**(): Retorna la suma de les puntuacions de tots els jugadors.
- **obtenirPuntuacioMaxima**(): Retorna la puntuació més alta registrada.
- **obtenirPuntuacioMinima**(): Retorna la puntuació més baixa registrada.
- **obtenirPuntuacions**(): Retorna un Mapa amb les puntuacions i els noms de tots els jugadors.

6. Recursos

La carpeta de resources/ conté tots els arxius de dades externs necessaris per el correcte funcionament del nostre joc de Scrabble. Està organitzada amb una carpeta per idioma i una per el driver de prova, facilitant així la gestió, ampliació i manteniment dels recursos segons l'idioma seleccionat.

Les carpetes de castellano, catalan i english inclouen els recursos específics per el idioma corresponent. La seva estructura interna es la següent:

- <idioma>.txt: Conté el diccionari de paraules valides en aquell idioma. Cada línea representa una paraula que pot jugar-se durant la partida.
- fitxes<idioma>.txt: Defineix la distribució de fitxes per lletra, indicant quantes fitxes n'hi ha de cada una i quants punts val. Aquesta informació és necessària per a poder inicialitzar el sac de la partida. El format de cada línea és: LLETRA-QUANTITAT-VALOR.

Finalment el directori driverFitxa conté l'arxiu fitxesdriverFitxa.txt. Aquest document és l'arxiu utilitzat per inicialitzar el sac en els drivers de prova. L'hem creat per poder provar totes les funcionalitats d'una manera més fàcil i directa.

7. Drivers

7.1. Preàmbul

Durant el desenvolupament del nostre sistema de joc Scrabble, hem creat dos drivers principals per verificar el correcte funcionament de components crítics: el DriverPartida i el DriverEstadística.

El nostre objectiu amb aquests drivers ha estat garantir que els components fonamentals del joc funcionin correctament de manera independent i sota diferents escenaris d'ús. Aquesta aproximació ens permet detectar i corregir errors en etapes primerenques del desenvolupament, quan encara són més fàcils de solucionar.

Pel que fa al DriverPartida, l'hem dissenyat per simular una partida completa de Scrabble, des de la configuració inicial fins al final. Aquest driver ens permet verificar tot el procés de configuració dels paràmetres inicials, com ara la mida del taulell, el faristol i l'idioma escollit. També comprova la correcta gestió dels torns entre jugadors humans i bots de diferents nivells de dificultat, així com tota la mecànica de col·locació i retirada de fitxes, la validació de les jugades i el càlcul de les puntuacions tenint en compte els multiplicadors. A més, ens permet comprovar la visualització adequada del taulell i els faristols, i funcionalitats addicionals com desfer jugades o guardar partides.

Pel que fa al DriverEstadística, ens hem centrat específicament en el sistema d'estadístiques del joc. Aquest driver comprova l'administració correcta de les puntuacions dels jugadors, permetent

operacions com afegir o restar punts a jugadors concrets, així com la consulta adequada dels rànquings i l'obtenció d'informació sobre els millors jugadors.

Hem optat per aquesta metodologia de testing mitjançant drivers perquè ens ofereix avantatges significatius. Ens permet provar components complexos amb interacció manual, sense necessitat d'implementar tots els tests unitaris més granulars. També facilita la detecció d'errors d'integració entre components que podrien passar desapercebuts en tests individuals. A més, ens proporciona una visió completa del funcionament del sistema des de la perspectiva de l'usuari final i ens permet verificar aspectes visuals i d'interacció que serien difícils d'avaluar amb tests automatitzats.

En resum, aquests drivers són eines essencials en aquesta fase del nostre desenvolupament, ja que ens permeten validar el funcionament correcte tant del joc com del sistema d'estadístiques, assegurant que tots els components interactuen adequadament abans de procedir amb la implementació de la interfície gràfica i la capa de persistència completa.

7.2. DriverEstadistica

L'archiu DriverEstadisitca.java conte un programa principal dissenyat per provar manualment les funcionalitats del controlador de les estadístiques. Aquest driver simula un menú interactiu que permet a l'usuari executar diverses operacions relacionades amb la gestió de puntuacions en el joc de Scrabble.

Aquest driver es útil en aquestes primeres etapes del desenvolupament, ja que encara no disposem d'una capa de persistencia que permet guardar les puntuacions entre partides. Per aquest motiu s'ofereix la possibilitat de afegir i manipular puntuacions manualment, amb el fi de verificar que aquestes funcions del sistema de estadistiques funcionen correctament.

El menú implementat ens permet fer les seguenta accions:

- Afegir una puntuació a un jugador
- Retirar punts a un jugador
- Consultar el millor jugador
- Obtenir la puntuació mitja entre tots els jugadors
- Obtenir la puntuació total acumulada
- Llistar totes les puntuacions registrades, ordenades de major a menor
- Eliminar un jugador i totes les seves estadístiques associades
- Sortir del programa

7.3. DriverPartida

El DriverPartida conté un programa principal dissenyat per simular i gestionar una partida completa del joc de Scrabble (és una simulació del que seria la capa de presentació, ja que no s'ha implementat encara). Aquest controlador permet interactuar amb el sistema de joc des de la configuració inicial fins a la finalització de la partida, oferint una interfície per línia de comandes que facilita la prova de totes les funcionalitats del joc.

6.3.1 Estructura General del Driver

El driver s'estructura en dues fases principals ben diferenciades:

1. Configuració inicial de la partida (‘inicialitzarPartida’).
2. Gestió dels torns de joc (‘jugarTorns’).

6.3.2 Fase de Configuració

Durant la fase de configuració, el driver guia l'usuari a través d'una sèrie de paràmetres que definen la partida:

- Mida del taulell: Sol·licita i valida una dimensió adequada pel taulell de joc.
- Mida del faristol: Determina quantes fitxes pot tenir cada jugador simultàniament.
- Idioma/Temàtica: Permet escollir entre català castellà i anglès, cosa que determina el diccionari que s'utilitzarà
- Bots: Configura el nombre de jugadors controlats per la màquina i la seva dificultat (Fàcil, Normal, Difícil).
- Jugadors humans: Estableix el nombre de jugadors humans i sol·licita els seus noms.

Després de recollir tota aquesta informació, el sistema mostra un resum de la configuració i demana confirmació abans d'inicialitzar la partida. Si l'usuari rebutja la configuració, es reinicia el procés.

6.3.3 Fase de Gestió dels Torns

Un cop iniciada la partida, el driver gestiona els torns dels jugadors, alternant entre:

6.3.3.1 Torns dels Bots

Per als jugadors controlats per la màquina, el sistema:

- Executa automàticament la jugada del bot
- Mostra la jugada realitzada i el faristol actualitzat

6.3.3.2 Torns dels Jugadors Humans

Per als jugadors humans, s'ofereix un menú amb les següents opcions:

1. Jugar paraula: Permet col·locar fitxes al taulell. Dins d'aquesta opció, es pot:
    - Col·locar una fitxa: Seleccionant la lletra i la posició
    - Retirar una fitxa: Eliminant una fitxa prèviament col·locada
    - Reiniciar torn: Desfent les fitxes col·locades en el torn actual
2. Passar torn: Renunciar a jugar en aquest torn.
3. Canviar fitxes: Retornar fitxes del faristol i obtenir-ne de noves.
4. Undo: Desfer l'última jugada (si és possible).

5. Guardar i sortir: Desar l'estat actual de la partida i sortir.
6. Sortir sense guardar: Finalitzar la partida sense desar els canvis.

6.3.3.3 Gestió de Fitxes i Jugades

El driver implementa funcionalitats específiques com:

- Col·locació de fitxes: Validació i confirmació de les jugades realitzades.
- Gestió de comodins: Permet substituir un comodí (#) per qualsevol lletra vàlida.
- Visualització del taulell: Mostra el taulell amb colors diferenciats segons els multiplicadors.
- Visualització del faristol: Presenta les fitxes disponibles amb la seva puntuació.

6.3.3.4 Característiques de la Interfície

El driver utilitza colors i formats en la sortida per consola per millorar la llegibilitat:

- Codis de color: Diferents colors per identificar els tipus de caselles (multiplicadors de paraula x2/x3, multiplicadors de lletra x2/x3, caselles normals).
- Separadors visuals: Línies que delimiten seccions de la interfície.
- Informació destacada: Dades del jugador actual, puntuació, estat del taulell i faristol.

6.3.3.5 Cicle de Vida de la Partida

1. **Inicialització**: Configuració dels paràmetres de la partida.
2. **Execució dels torns**: Alternança entre jugadors humans i bots.
3. **Final de partida**: La partida finalitza quan es compleixen les condicions de finalització (determinades pel controlador de domini).
4. **Resultat**: Presentació dels resultats finals.

Aquest driver és una eina essencial per provar manualment totes les funcionalitats del joc de Scrabble, permetent simular partides completes i verificar el comportament del sistema en diferents situacions i configuracions.

8. Jocs de prova dels drivers

8.1. Driver partida

Les proves realitzades sobre el projecte DriverToni tenen com a objectiu garantir que el joc funcioni correctament en diversos escenaris d'entrada per part de l'usuari. A continuació, es detallen els casos de prova, amb les configuracions utilitzades i els passos seguits per verificar el correcte funcionament de cada aspecte del joc.

Nom	Configuració del Joc
Configuració	**Taulell**: 15 **Faristol**: 7 **Idioma**: castellano **Bot**: 2 (dificultat 2 i 3) **Jugadors humans**: 1
Passos realitzat	**Mida del Taulell:** La configuració del taulell és de **15x15** caselles. Es verifica que el taulell sigui de la mida correcta, amb un total de 225 caselles disponibles per col·locar les fitxes. **Mida del Faristol:** Cada jugador comença la partida amb un **faristol de 7 fitxes**. Es comprova que aquesta configuració es compleixi, amb el faristol inicialitzat amb 7 fitxes disponibles per al jugador. El sistema també ha de comprovar que el faristol no permeti que el jugador tingués més o menys fitxes en qualsevol moment. Quan el jugador col·loca una fitxa sobre el taulell, el faristol es redueix en 1 fitxa; si intenta col·locar una fitxa quan no té cap fitxa disponible, el sistema ha de mostrar un missatge indicant que no pot jugar més. **Idioma:** El sistema s'ha configurat en **castellà** per a aquest conjunt de proves. Fa servir el diccionari castellà atorgat pels professors. **Comprovació del nombre de jugadors humans i bots:** Es comprova que el sistema permet configurar correctament un jugador humà i un bot. El sistema accepta la configuració de 1 jugador humà i 2 bots de dificultat diferent sense errors. Es comprova també que els jugadors humans poden realitzar les seves jugades en els torns establerts, mentre que el bot fa les seves pròpies jugades en el seu torn.

Nom	Errors en col·locar o retirar paraules
Configuració	**Taulell**: 15 **Faristol**: 7

	Idioma: castellano **Bot**: 1 **Jugadors humans**: 1
Passos realitzat	**Col·locar una fitxa en una casella buida**: Primer, es comprova que el jugador pot col·locar una fitxa en qualsevol casella que estigui buida. Es verifica que el sistema permet col·locar la fitxa en la posició seleccionada, sempre que aquesta sigui dins dels límits del taulell i no s'hi hagi col·locat prèviament una altra fitxa. **Col·locar una fitxa fora de rang o en una posició ocupada**: A continuació, es comprova que el jugador no pot col·locar una fitxa en una casella fora del rang del taulell o en una posició on ja hi hagi una fitxa col·locada. Si el jugador intenta col·locar la fitxa en una posició ocupada, el sistema emet un missatge d'error indicant que la casella ja està ocupada. Si intenta col·locar una fitxa fora del taulell, el sistema també mostra un missatge d'error indicant que la posició no és vàlida. **Retirar una fitxa**: Es comprova que el jugador només pot retirar una fitxa si ha estat col·locada prèviament per ell al mateix torn. Si el jugador intenta retirar una fitxa d'una posició on no hi ha, el sistema emet un missatge d'error indicant que no es pot retirar una fitxa que no ha estat col·locada.

Nom	Validació de les paraules jugades
Configuració	**Taulell**: 15 **Faristol**: 7 **Idioma**: castellano **Bot**: 1 **Jugadors humans**: 1
Passos realitzat	**Introducció de paraules vàlides**: El sistema verifica que quan el jugador col·loca una paraula, aquesta sigui vàlida segons el diccionari del joc. Per provar-ho, es col·loquen diverses fitxes en el taulell que sabem que són vàlides, com ara "LAXITUD" o "CHACHALAQUEABAMOS". El sistema ha de verificar que aquestes paraules són acceptades sense errors i que ens donen l'opció de fer el "*commit*" de la paraula. **Introducció de paraules no vàlides**: A continuació, es comprova que el sistema no permet la col·locació de paraules que no són vàlides, per exemple "dsafasd" o "supercalifragilisticoespialidoso" (si no forma part del diccionari). Quan es col·loca una paraula no vàlida, el sistema

	ha no deixa al usuari poder validar la paraula.

Nom	Gestió dels torns i interacció amb els bots
Configuració	**Taulell**: 15 **Faristol**: 7 **Idioma**: castellano **Bot**: 1 (dficultat 3) **Jugadors humans**: 1
Passos realitzat	**Canvi de torns entre el jugador humà i el bot:** En aquest cas, el jugador humà comença la partida. Es comprova que després de cada jugada, el sistema canvia correctament el torn entre el jugador humà i el bot. El bot realitza una jugada automàtica basada en la dificultat configurada (en aquest cas, dificultat 3). Es verifica que el bot realitza jugades coherents i que el torn es canvia de manera seqüencial sense que hi hagi errors.

Nom	Dificultat del Bot
Configuració	**Taulell**: 15 **Faristol**: 7 **Idioma**: castellano **Bot**: 3 (un de cada dificultat) **Jugadors humans**: 1
Passos realitzat	**Dificultat 1 (Fàcil):** Es configura el bot a dificultat 1 per observar el seu comportament. El bot, a la dificultat més baixa, fa moviments molt simples i previsibles, centrant-se en col·locar paraules curtes i de poc valor. Aquest Es verifica que el bot tria paraules bàsiques i curtes utilitzant només les fitxes que té disponibles. **Dificultat 2 (Mitjana):** Es configura el bot a dificultat 2 per a aquesta fase de les proves. A dificultat mitjana, el bot ha de mostrar un comportament no tant fàcil (paraula itermedia). Es verifica que el bot intenti formar paraules més llargues i de major puntuació. **Dificultat 3 (Difícil):** Es configura el bot a dificultat 3 per comprovar que la seva estratègia sigui més avançada i difícil de superar (la millor paraula).

Nom	Finalització de la Partida amb Faristol Buit
Configuració	**Taulell**: 15 **Faristol**: 7 **Idioma**: castellano **Bot**: 3 (en dificultat màxima, paraules llargues) **Jugadors humans**: 0
Passos realitzat	**Comprovació de la Finalització de la Partida:** Si un jugador queden sense fitxes (tant en el faristol com en el sac), la partida ha de finalitzar automàticament.

Nom	Jugadors que passen durant dos cicles seguits
Configuració	**Taulell**: 15 **Faristol**: 7 **Idioma**: castellano **Bot**: 0 **Jugadors humans**: 4
Passos realitzat	**Passar:** Es comprova que quan un jugador passa el seu torn, aquest es registra correctament en el sistema. La situació es dona quan el jugador decideix no col·locar cap fitxa i usa l'opció de passar (2). Si el jugador passa, el sistema ha de mantenir un compte dels torns passats. **Passar durant dos cicles seguits:** Si els jugadors passen durant dos cicles consecutius això vol dir que la partida ha d'entrar en la fase de finalització. Quan els dos jugadors passen dues vegades consecutives es comprova que el sistema finalitzi la partida correctament. En aquest cas, quan hi ha 8 passades seguides (2*nombreJugadors).

Nom	Punts finalització partida
Configuració	**Taulell**: 15 **Faristol**: 7 **Idioma**: castellano **Bot**: 3 (indiferent) **Jugadors humans**: 0
Passos realitzat	Aprofitem que tenim els bots per simular un joc i veure que els bots acaben si ja no tenen espai per col·locar paraules o fer jugades. Ho detectem i ho marquem com a “passar” del bot. Així, quan ja tots els bots passen (test anterior), s’acaba la partida. **Restar la Puntuació de les Fitxes Restants:** Quan la partida finalitza, el sistema ha de restar la puntuació de les fitxes que el jugador tingui al faristol. Es comprova que qualsevol fitxa restant en el faristol del jugador s'ha de restar de la seva puntuació total. **Suma jugador faristol buit**: Es comprova que si un jugador acaba la partida amb el faristol buit, se li sumen els punts descomptats dels altres jugadors per excès de fitxes als seus faristols. **Puntuació Final i Comparació:** Un cop totes les paraules han estat puntuades i les fitxes restants han estat descomptades, el sistema ha de mostrar el resultat final de la partida. Aquest inclou la puntuació total de cada jugador i el guanyador, qui serà el jugador amb la puntuació més alta després de les deduccions de les fitxes restants.

8.2. Driver estadística

L'hem provat amb un fitxer de prova (provaStats.txt). L'hem preparat per comprovar que totes les opcions del menú del DriverEstadistica funcionen correctament. Comencem afegint algunes puntuacions a jugadors com en Joan i la Maria, per veure si el sistema les guarda i les suma bé quan un jugador rep més d'una puntuació. Un cop fet això, demanem que ens mostri totes les puntuacions i mirem qui és el millor jugador, per comprovar que ho calcula bé.

També revisem la puntuació mitjana i la total, per assegurar-nos que fa bé els càlculs. Després retirem punts a en Joan i tornem a mostrar les puntuacions, per veure si s'ha actualitzat correctament. A continuació, eliminem la Maria i ens assegurem que ja no aparegui. Per acabar, afegim una nova jugadora, la Clara, i tornem a consultar les dades per veure que tot segueix funcionant com toca.

Finalment, tanquem el programa amb l'opció de sortir. Amb aquest recorregut, passem per totes les funcionalitats disponibles i ens assegurem que el controlador d'estadístiques reacciona correctament a cada cas.

Arxiu de text *provaStats.txt*

```
Unset
1
Joan
10
1
Maria
30
1
Joan
20
6
3
4
5
2
Joan
10
6
7
Maria
6
1
Clara
25
3
6
0
```

9. Testos unitaris

9.1. Preàmbul

En el desenvolupament del projecte de Scrabble, hem implementat una sèrie de proves unitaries orientades específicament a testar el comportament individual de cada classe que forma part del sistema. L'objectiu principal ha estat assegurar que cada component compleix correctament amb la seva funcionalitat, de forma independent i en diferents situacions.

Aquest apartat recull i comenta els tests dissenyats per a cadascuna de les classes del projecte, com ara la gestió del taulell, la lògica de puntuació, la manipulació de fitxes o la validació de paraules. Les proves ens han permès identificar i corregir errors, així com garantir l'estabilitat i coherència del comportament intern del joc.

A més de validar el funcionament inicial de cada classe, els tests han resultat especialment útils durant el manteniment del codi. Quan s'ha realitzat algun canvi o refactorització en una classe, l'execució dels tests ens ha permès comprovar ràpidament que les funcionalitats continuen funcionant correctament, evitant la introducció de regressions o errors inesperats.

9.2. Test Fitxa

Els tests cobreixen diferents aspectes de la classe:

- **Creació de fitxes normals i dígrafs:** Es verifica que les fitxes s'instanciïn correctament amb la lletra i puntuació corresponents, incloent-hi fitxes especials com els dígrafs (p. ex. "CH").

- **Funcionalitats bàsiques:** Es comprova el funcionament dels mètodes obtenirLletra, obtenirPunts, toString, i esDigraf, així com la seva coherència amb el tipus de fitxa.

- **Igualtat de fitxes:** S'han fet proves per assegurar que dues fitxes amb la mateixa lletra i punts siguin considerades iguals, i que diferències en qualsevol d'aquests camps impliquin desigualtat.

- **Gestió d'errors:** S'inclou un test que comprova que no es poden crear fitxes amb valors invàlids (com punts negatius), llançant l'excepció corresponent.

- **Comodins:** Una part important dels tests es dedica a la nova funcionalitat de les fitxes comodí (#), que tenen un comportament especial. Es verifica que només aquestes fitxes poden canviar la seva lletra mitjançant el mètode setLletraComodi, i que es reconeguin correctament amb el mètode esComodi.

9.3. Test Faristol

Els tests cobreixen els següents aspectes principals:

- **Inicialització**: Es comprova que un faristol buit es comporti com s'espera (sense fitxes, no ple).

- **Afegir fitxes**: S'ha validat que les fitxes es poden afegir correctament fins al límit, i que en intentar afegir-ne més del màxim permès, es llença l'excepció ExcepcioFaristolPle.

- **Accés per índex**: Es testa la funcionalitat d'obtenir una fitxa concreta segons la seva posició, incloent també la gestió d'errors si es demana un índex invàlid.

- **Eliminació de fitxes**: S'assegura que es poden eliminar fitxes que hi són, i que si s'intenta eliminar una fitxa que no està present, es llença l'excepció ExcepcioFaristolNoConteLaFitxa.

- **Barrejar fitxes**: S'ha implementat un test que verifica que la funció de barrejar no elimini cap fitxa i mantingui la integritat del conjunt. Tot i que no es pot garantir un canvi d'ordre (ja que és aleatori), es comprova que les fitxes segueixin presents i que l'operació no falli.

9.4. Test Jugador

Els tests cobreixen els següents aspectes:

- **Inicialització i obtenció de dades bàsiques:** Es comprova que el jugador es crea amb el nom correcte i amb una puntuació inicial de zero, així com que el faristol assignat s'obté correctament.

- **Gestió de punts:** Es valida que la puntuació del jugador augmenti o disminueixi segons correspongui, reflectint els resultats de la partida o penalitzacions.

- **Delegació d'operacions al faristol:** Una part important de la lògica de la classe Jugador és que les operacions relacionades amb la gestió de fitxes (com afegir o eliminar fitxes) es deleguen correctament al Faristol. Per comprovar-ho, s'utilitzen mocks per verificar que aquestes operacions es transmeten com s'esperava.

9.5. Test Bot

Els tests cobreixen els següents aspectes:

1. **Inicialització i creació de bots**: Es comprova que els bots es creen correctament amb el nom assignat i amb un nivell de dificultat vàlid (entre 1 i 3). S'assegura que els noms dels bots coincideixen amb els valors esperats en el constructor.

2. **Validació del nivell de dificultat**: Es verifica que si es passa un valor incorrecte per al nivell de dificultat (per exemple, 0), es llanci una excepció IllegalArgumentException. Això garanteix que els nivells de dificultat siguin dins de l'interval permès (1-3).

3. **Obtenint la dificultat dels bots**: Es valida que els bots tinguin el nivell de dificultat correctament assignat, comprovant que el mètode obtenirDificultat retorna els valors esperats per a cada bot (fàcil, mitjà i difícil).

4. **Còpia d'un bot**: Es comprova que un bot es pot copiar correctament utilitzant el mètode de còpia. La còpia manté el mateix nom i nivell de dificultat que l'original, el que garanteix que la lògica de còpia funciona com s'espera.

5. **Verificació que els bots són realment bots**: Es comprova que el mètode esBot retorna true per a tots els objectes de tipus Bot, confirmant que la classe es comporta com un bot, tal com es defineix en la seva implementació.

9.6. Test Casella

Aquestes proves cobreixen diversos escenaris essencials:

- **Estat inicial i col·locació de fitxes**: Es comprova que la casella inicialment està buida, i que en col·locar una fitxa l'estat canvia adequadament. També es verifica que no es pot col·locar una nova fitxa si ja n'hi ha una (es llença ExcepcioCasellaOcupada) ni accedir a una fitxa inexistent (es llença ExcepcioCasellaBuida).

- **Gestió de fitxes**: Es valida que es poden retirar fitxes correctament i que aquesta acció només es pot fer si realment hi ha una fitxa a la casella.

- **Estat de "jugada":** Es comprova que una casella pot marcar-se com a jugada, i que aquest estat es conserva.

- **Representació gràfica** (toString): S'han implementat diversos tests per assegurar que la representació textual de la casella reflecteix correctament el seu contingut (buit, fitxa simple o dígraf).

- **Estrategies de puntuació**: Es comprova que cada casella aplica l'estratègia de puntuació correcta segons la seva posició. Això inclou multiplicadors de paraula (doble o triple), multiplicadors de lletra, o simplement l'estratègia normal. Aquest apartat és fonamental per assegurar que el taulell de joc aplica correctament les regles especials de puntuació.

9.7. Test Taulell

Els casos de prova cobreixen:

- **Validació del constructor:** Es comprova que només es permeten mides imparelles per assegurar que hi ha una casella central (condició necessària per a la primera jugada).

- **Estat inicial:** Es valida que el taulell es crea buit i es detecta correctament quan s'hi col·loca la primera fitxa.

- **Gestió de fitxes:** Es proven operacions bàsiques de col·locació i retirada de fitxes, incloent-hi control d'errors per coordenades fora de límits.

- **Accés a caselles:** Es verifica l'accés correcte a les caselles internes, especialment important per a la manipulació directa des de components com DAWG o la classe Jugada.

- **Validació de jugades:** S'inclouen casos que comproven si les jugades són vàlides segons les regles del joc:

    - La **primera jugada** ha de passar pel centre.

    - Jugades posteriors han de **connectar amb paraules existents**.

- **Càlcul de puntuació:** S'avalua la puntuació assignada a una jugada:

    - Tant per a la **paraula principal**, com per a paraules **perpendiculares** generades durant la jugada.

    - També es comprova l'efecte d'**estratègies de puntuació**, com multiplicadors de paraula (x2, etc.).

9.8. Test Sac

Els tests validen:

- **Afegir fitxes:** Es comprova que el sac augmenta correctament la quantitat de fitxes, incloent-hi casos amb duplicats (mateixa lletra i puntuació).

- **Obtenir el nombre de fitxes:** Tant pel total com per una fitxa específica, es valida que el recompte sigui exacte.

- **Extreure fitxes:**

    - Es prova l'extracció aleatòria.

    - L'extracció per lletra específica, assegurant que es redueix el recompte correctament.

    - També es valida que el comportament davant errors sigui correcte (com intentar treure fitxes d'un sac buit o d'una fitxa que no existeix).

- **Comprovació de buit:** Es verifica l'estat del sac abans i després de treure o afegir fitxes.

- **Gestió de comodins:**

    - Es comprova que les fitxes amb puntuació zero són reconegudes com a **comodins**.

    - S'avalua que aquests comodins es poden afegir i treure com qualsevol altra fitxa.

9.9. Test Jugada

Aspectes que es validen en els tests:

- **Constructor i getters:**

    - Es comprova que el constructor de la classe inicialitza correctament els atributs paraulaFormada, puntuacio, casellesJugades i jugadaValida.

    - La prova valida que els getters retornen els valors correctes i que la jugada és marcada com a vàlida.

- **Metodes setters:**
    - **setParaulaFormada():** Es valida que el canvi del valor de la paraula formada s'apliqui correctament.

    - **setPuntuacio():** Es comprova que la puntuació s'actualitzi correctament en la jugada.

- **setJugadaValida():** Es valida el canvi de l'estat de validesa de la jugada, passant de vàlida a no vàlida.

- **Immutabilitat de les caselles jugades:**

  - Es comprova que les caselles implicades en una jugada no es poden modificar després de la creació de la jugada.

  - Les caselles jugades s'han d'afegir de manera que no es poden eliminar ni modificar un cop establertes en el moment de la creació.

9.10. Test DAWG

Els tests cobreixen diversos aspectes fonamentals de la classe, com ara la validació de paraules, la comprovació de prefixes vàlids i la gestió de dígrafs. Al before inicialitzem un dawg, amb certs tokens i paraules personalitzats i després revisem si existeixen.

- Proves de paraules vàlides:

  - **testParaulaSimpleValida**: Aquesta prova verifica que les paraules presents a la llista de paraules de la DAWG siguin reconegudes com a vàlides. Es comprova que les paraules "CASA", "CAS" i "SABATA" estan correctament emmagatzemades i retornen true quan es consulta la seva existència.

  - **testParaulaInexistent**: Es comprova que les paraules que no existeixen en la DAWG, com "CASO", "SABA" i "CHICO", retornen false quan es consulta la seva existència.

- Proves de paraules amb dígrafs:

  - **testParaulaAmbDigraf**: Es verifica el comportament de la DAWG en relació amb les paraules que contenen dígrafs. Per exemple, "CHIC" i "CH" són paraules vàlides, mentre que "C" no ho és, ja que "C" no es considera una paraula en la DAWG.

- Proves de prefixes:

  - **testPrefixValid**: Aquesta prova valida que la DAWG reconeix prefixos vàlids de paraules emmagatzemades. Per exemple, "CAS" és un prefix vàlid de "CASA" i "CAS", i "CHI" és un prefix vàlid de "CHIC" i "CHIP".

  - **testPrefixInvalid**: Es comprova que els prefixes que no existeixen, com "XO", "BA" i "XCH", retornen null, indicant que no són prefixos vàlids en la DAWG.

  - **testPrefixAmbDigraf**: Aquesta prova assegura que la DAWG gestiona correctament els dígrafs com prefixos. Per exemple, "CH" i "CHI" són prefixes vàlids, mentre que "CI" no ho és.

10. Funcionalitats extres Implementades

A més de les funcionalitats requerides per l'enunciat del projecte, hem desenvolupat una sèrie de millores i extensions que aporten, personalització i intel·ligència al joc. Aquestes funcionalitats extres són:

10.1. Partides personalitzables:

- **Mida del taulell configurable**: Es pot triar qualsevol mida imparella (per garantir la simetria del joc) i jugar amb un taulell d'aquella mida. Les caselles amb bonificacions s'adaptaran automàticament a noves caselles.
- **Mida del faristol variable**: L'usuari pot definir quantes fitxes tindran al faristol els jugadors.
- **Nombre flexible de jugadors i bots**: Es permet configurar partides amb qualsevol nombre de jugadors i bots.

10.2. Nivells de dificultat del Bot:

Cadascun dels bots pot operar amb tres nivells de dificultat, aquest serà escollit a l'inici de la partida

- **(1)** ***Fàcil***: Escull la jugada vàlida que otorga menys puntuació
- **(2)** ***Mitjà***: Escull una jugada intermedia.
- **(3) Difícil**: Escull la jugada vàlida que otorga més punts.

10.3. Desfer moviment **(Undo)**

Hem implementat una funció que durant la partida permet retornar a l'estat del joc abans de l'últim torn realitzat per un jugador humà.

Aquesta funcionalitat es gestiona mitjançant un historial complet de torns que clona l'estat de cada moment de la partida.

11. Futures millores a implementar

Tot i les funcionalitats i millores actuals del sistema, considerem que hi ha àrees amb potencial per a una futura ampliació o optimització. Aquestes millores afegirien encara més flexibilitat i intel·ligència al joc, així com enriquirien l'experiència de l'usuari:

- **Oferir nous diccionaris**
  Tenir preparats diversos jocs de diccionaris temàtics que permetin jugar amb diferents paraules diferents.

- **Personalització del Sac de Fitxes**
  Donar a l'usuari la capacitat de personalitzar el contingut del sac de fitxes. Això inclouria:
  - Definir la quantitat de cada lletra o dígraf.
  - Assignar valors de puntuació personalitzats per cada lletra o dígrafs.
  - Crear distribucions adaptades a idiomes minoritaris o a regles de jocs alternatius.

- **Millorar el sistema de rànquing:**
  - Punts totals
  - Paraules més llargues formades
  - Nombre de partides jugades

- **Optimització de l'algorisme del bot:**
  Una futura millora podria consistir en millorar l'algorisme de Backtracking, aplicant tècniques de poda intel·ligent (heurístiques que descarten combinacions ineficients abans d'explorar-les) i així evitar continuar explorant combinacions quan ja se sap que no podran superar la millor jugada trobada fins al moment.
  Aquesta optimització suposaria una gran millora en eficiència i rendiment, especialment en taulells grans o amb moltes fitxes disponibles.

- **Fer més intel·ligent l'algorisme de jugada del Bot:**
  Una altra línia de millora és dotar el bot d'una intel·ligència contextual, que li permeti prendre decisions estratègiques més enllà de la simple maximització de punts. Això inclouria:
  - Conservar lletres valuoses per a futurs torns en lloc de gastar-les immediatament.
  - Bloquejar jugades potencials de l'oponent, ocupant posicions estratègiques del taulell.
  - Escollir entre jugades amb la mateixa puntuació tenint en compte el faristol resultant o la probabilitat de millors opcions futures.

- **Sistema d'ajuda:**
  Possibilitat de tenir un mode de joc on poder demanar una pista, i que el programa et digui algunes paraules que puguis jugar amb les fitxes disponibles del faristol.