UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

SCRABBLE

Projecte de Programació

Curs 2024-2025, QP

Versió 2.0
Grup 43.3
Escofet González, Gina (gina.escofet)
Gascón Moliné, Gerard (gerard.gascon)
Martínez Lassalle, Felipe (felipe.martinez.lassalle)
Pérez Silvestre, Biel (biel.perez)
Usero Martínez, Albert (albert.usero)

Índex

1. Decisions preses

1.1. Capa de persistència

En aquesta segona entrega, hem incorporat la capa de persistència al projecte. Aquesta capa té com a funció principal gestionar com es guarden i es carreguen les dades del sistema. Actua com a intermediari entre la representació en memòria de les dades i el sistema d'emmagatzematge persistent.

Concretament, s'encarrega de la serialització i deserialització dels objectes, facilitant la conversió entre els models del domini i un format d'emmagatzematge adequat, com ara JSON. Per a aquesta implementació, hem utilitzat la biblioteca **Gson**, que ofereix una manera eficient i senzilla de dur a terme aquestes operacions.

Amb aquesta estructura, la capa de persistència esdevé una solució neta, modular i fàcilment extensible, que permet modificar o substituir el mecanisme d'emmagatzematge sense afectar la resta de capes del sistema.

1.2. Capa de dades

En aquesta segona entrega, la **capa de dades** s'ha mantingut pràcticament igual que a la primera. Continua sent lleugera, testejable i independent de la resta del sistema. L'única ampliació significativa ha estat la incorporació de mètodes relacionats amb el **guardat de dades**, derivada de la necessitat d'introduir un mecanisme de persistència a l'aplicació.

Per implementar aquest mecanisme, s'ha utilitzat el component PersistentDictionary, que actua com a **capa d'abstracció** sobre el sistema d'emmagatzematge real (basat en fitxers JSON). Aquesta solució permet desar i recuperar l'estat dels objectes de manera senzilla i desacoblada.

Els objectes persistents fonamentals per poder restaurar correctament una partida són la **bossa de peces** (Bag), el **registre de puntuacions** (Leaderboard) i el **tauler de joc** (Board). Aquests elements encapsulen ara la lògica necessària per **serialitzar-se i desserialitzar-se**, assumint la responsabilitat de transformar-se en una representació emmagatzemable i de reconstruir-se a partir d'aquesta.

1.3. Capa de domini

En aquesta segona entrega, la capa de domini s'ha mantingut pràcticament igual que en la primera. Les novetats introduïdes, principalment relacionades amb la persistència de dades, s'han implementat de manera encapsulada a la capa de persistència, sense alterar el funcionament intern del model de domini. Aquest fet posa de manifest una bona aplicació del principi de **separació de responsabilitats**, ja que permet evolucionar el sistema afegint funcionalitats sense afectar el nucli de la lògica del joc.

1.4. Capa de presentació

Finalment, en aquesta segona entrega s'ha renovat completament la interfície d'usuari, adoptant la biblioteca **Swing** de Java per implementar una interfície gràfica. D'aquesta manera, el joc de Scrabble ofereix ara una experiència més visual i interactiva per a l'usuari.

La nova interfície permet interactuar amb diversos elements del joc, com el **tauler**, les **fitxes de la mà**, i diferents **botons** per dur a terme accions com jugar un torn, guardar la partida o sortir del joc. Els components gràfics, com GameScreen, MenuScreen i d'altres, s'han estructurat seguint el patró **Model-Vista-Presentador (MVP)**, en què aquestes classes actuen com a "vista", mantenint separada la lògica de presentació de la lògica de domini.

Un aspecte clau és que aquestes noves vistes gràfiques es comuniquen amb la capa de domini a través de **les mateixes interfícies** que ja es van definir a la **primera entrega**, quan la interfície era en mode **terminal**. Aquesta reutilització ha permès incorporar la nova capa gràfica sense haver de modificar la lògica del programa, afavorint el desacoblament entre capes i facilitant la mantenibilitat i l'escalabilitat del sistema.

2. Classes

A causa de l'aplicació del **Principi de Responsabilitat Única**, hem dissenyat el sistema amb múltiples classes, cadascuna encarregada d'una única responsabilitat. Aquesta decisió facilita el manteniment, la reutilització i la prova unitària del codi, ja que cada classe té un propòsit ben definit.

Com a conseqüència, el **diagrama de classes complet** pot resultar visualment complex i dens, dificultant la seva comprensió en una sola imatge. Per aquest motiu, i amb l'objectiu de millorar-ne la llegibilitat i claredat, hem optat per **dividir el diagrama en diferents seccions** corresponents a les capes i components del sistema:

- **Capa de persistència**
- **Capa de dades**
- **Capa de domini**
- **Capa de presentació**

Aquesta divisió ens permet presentar les relacions i responsabilitats de les classes dins del seu context específic, ajudant a entendre millor l'arquitectura modular del projecte.

Tot i això, per tal d'oferir una visió global del sistema, també s'adjunta una **imatge amb el diagrama de classes complet** en un arxiu a part, que mostra totes les dependències i interaccions entre els components.

2.1. Capa de persistència

Classe Deserializer

Classe abstracta que facilita la deserialització d'elements JSON en objectes de tipus PersistentObject.

Aquesta classe conté un mètode que processa recursivament diferents tipus d'elements JSON (primitives, arrays, objectes) i els converteix en tipus Java equivalents o en instàncies de PersistentObject.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Mètodes

protected Object parseJsonElement(JsonElement element, JsonDeserializationContext context)

Processa un element JSON i el converteix en un objecte Java equivalent.

Aquest mètode gestiona els següents casos:

Si l'element és null, retorna null.

Si és una primitive, retorna el valor corresponent (boolean, string, int, long o double).

Si és un array JSON, crea una llista d'objectes PersistentObject a partir dels elements de l'array.

Si és un objecte JSON, el deserialitza com un PersistentObject.

Retorna: Un objecte Java equivalent a l'element JSON, o null si no és possible.

Paràmetres

- **element**: L'element JSON a processar.
- **context**: El context de deserialització de Gson.

Classe GsonDeserializer

Implementació de la interfície IDeserializer utilitzant la biblioteca Gson per a la deserialització JSON.

Aquesta classe configura un objecte Gson amb deserialitzadors personalitzats per a PersistentObject i PersistentDictionary i proveeix un mètode per convertir cadenes JSON en objectes Java de tipus PersistentObject o subtipus.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Constructores

public GsonDeserializer()

Crea una instància de GsonDeserializer configurant Gson amb els deserialitzadors personalitzats.

Mètodes

public Tdeserialize(String data, Class clazz)

Deserialitza una cadena JSON en un objecte de la classe especificada que hereti de PersistentObject.

Retorna: L'objecte deserialitzat de tipus T.

Paràmetres

- **data**: La cadena JSON que conté la representació de l'objecte.
- **clazz**: La classe de l'objecte al qual es deserialitza, que ha de ser una subclasse de PersistentObject.

Classe PersistentDictionaryDeserializer

Deserialitzador personalitzat per a PersistentDictionary utilitzant Gson.

Aquesta classe implementa JsonDeserializer per deserialitzar objectes JSON que representen un PersistentDictionary, convertint-los en una instància de PersistentDictionary amb el nom i el mapa intern d'objectes PersistentObject.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Mètodes

public PersistentDictionary deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context) throws JsonParseException

Deserialitza un element JSON en un PersistentDictionary.

Retorna: Un PersistentDictionary amb el nom i els elements deserialitzats.

Llança JsonParseException si el JSON no conté exactament una entrada o si hi ha errors en la deserialització.

Paràmetres

- **json**: L'element JSON que conté la representació del diccionari.
- **typeOfT**: El tipus de l'objecte a deserialitzar.
- **context**: El context de deserialització de Gson.

Classe PersistentObjectDeserializer

Deserialitzador personalitzat per a PersistentObject utilitzant Gson.

Aquesta classe implementa JsonDeserializer per deserialitzar un element JSON que representa un PersistentObject, convertint-lo en una instància de PersistentObject amb un nom i un valor associat.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Mètodes

public PersistentObject deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context) throws JsonParseException

Deserialitza un element JSON en un PersistentObject.

Retorna: Un PersistentObject amb el nom i el valor deserialitzat.

Llança una JsonParseException Si el JSON no conté exactament una propietat o si hi ha errors en la deserialització.

Paràmetres

- **json**: L'element JSON que conté la representació de l'objecte persistent.
- **typeOfT**: El tipus de l'objecte a deserialitzar.
- **context**: El context de deserialització de Gson.

Classe GsonSerializer

Serialitzador que utilitza Gson per convertir PersistentObjects en JSON.

Aquesta classe configura un Gson personalitzat amb serialitzadors específics per a PersistentObject, PersistentDictionary i PersistentArray per gestionar la serialització d'aquests tipus de dades.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Constructores

public GsonSerializer()

Constructor que crea un objecte Gson configurat amb els serialitzadors personalitzats.

Mètodes

public String serialize(PersistentObject object)

Serialitza un PersistentObject a una cadena JSON.

Retorna: La representació JSON de l'objecte.

Paràmetres

- **object**: L'objecte persistent a serialitzar.

Classe PersistentArraySerializer

Serialitzador per a PersistentArray que utilitza Gson.

Aquesta classe converteix un PersistentArray en un objecte JSON on la clau és el nom de l'array i el valor és un array JSON amb els elements serialitzats.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Mètodes

public JsonElement serialize(PersistentArray src, Type typeOfSrc, JsonSerializationContext context)

Serialitza un PersistentArray a un JsonElement.

Retorna: Un JsonElement que representa el PersistentArray.

Paràmetres

- **src**: L'objecte PersistentArray a serialitzar.
- **typeOfSrc**: El tipus de l'objecte que s'està serialitzant.
- **context**: El context de serialització Gson per serialitzar elements interns.

Classe PersistentDictionarySerializer

Serialitzador per a PersistentDictionary que utilitza Gson.

Aquesta classe converteix un PersistentDictionary en un objecte JSON on la clau és el nom del diccionari i el valor és un objecte JSON amb les entrades del diccionari serialitzades.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Mètodes

public JsonElement serialize(PersistentDictionary src, Type typeOfSrc, JsonSerializationContext context)

Serialitza un PersistentDictionary a un JsonElement.

Retorna: Un JsonElement que representa el PersistentDictionary.

Paràmetres

- **src**: L'objecte PersistentDictionary a serialitzar.
- **typeOfSrc**: El tipus de l'objecte que s'està serialitzant.
- **context**: El context de serialització Gson per serialitzar elements internsClasse PersistentObjectSerializer

Serialitzador per a PersistentObject que utilitza Gson.

Aquesta classe converteix un PersistentObject en un objecte JSON amb una sola propietat, on la clau és el nom de l'objecte i el valor és el seu valor serialitzat. Si el valor és null, s'afegeix un JsonNull.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Mètodes

public JsonElement serialize(PersistentObject src, Type typeOfSrc, JsonSerializationContext context)

Serialitza un PersistentObject a un JsonElement.

Retorna: Un JsonElement que representa el PersistentObject.

Paràmetres

- **src**: L'objecte PersistentObject a serialitzar.
- **typeOfSrc**: El tipus de l'objecte que s'està serialitzant.
- **context**: El context de serialització Gson per serialitzar el valor intern.

Classe SaveDataStreamer

Classe abstracta encarregada de gestionar l'obtenció del camí absolut per guardar fitxers de dades de la partida.

Proporciona un mètode per obtenir la ruta absoluta a un fitxer dins del directori de guardat de dades del projecte.

Cardinalitat: Una instància en tot el programa

Autor: Gerard Gascón

Mètodes

protected File getAbsolutePath(String fileName)

Obté el camí absolut d'un fitxer de guardat dins del directori del projecte.

Construeix la ruta relativa al directori 'edu/upc/prop/scrabble/save' i la combina amb el nom del fitxer proporcionat.

Retorna: Un objecte File amb la ruta absoluta al fitxer de guardat.

Llança una RuntimeException si hi ha un error en obtenir l'URI del directori del programa.

Paràmetres

- **fileName**: Nom del fitxer del qual es vol obtenir la ruta absoluta.

Classe SaveReader

Classe encarregada de llegir dades des de fitxers de guardat de la partida.

Estén la classe abstracta SaveDataStreamer per obtenir la ruta absoluta i implementa la interfície ISaveReader per a la lectura de fitxers.

Cardinalitat: Una instància en tot el programa

Autor: Gerard Gascón

Mètodes

public String read(String fileName)

Llegeix el contingut d'un fitxer de guardat en format text.

Utilitza la codificació UTF-8 per llegir el fitxer. Si el fitxer no existeix, retorna null. En cas d'error durant la lectura, imprimeix un missatge d'error i retorna null.

Retorna: El contingut del fitxer com a cadena, o null si no existeix o hi ha un error.

Paràmetres

- **fileName**: Nom del fitxer que es vol llegir.

public boolean exists(String fileName)

Comprova que el fitxer de guardat existeixi.

Paràmetres

- **fileName**: Nom del fitxer de guardat.

Classe SaveWriter

Classe encarregada d'escriure dades en fitxers de guardat de la partida.

Estén la classe abstracta SaveDataStreamer per obtenir la ruta absoluta i implementa la interfície ISaveWriter per a l'escriptura de fitxers.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Mètodes

public void write(String data, String fileName)

Escriu el contingut proporcionat en un fitxer.

Si hi ha algun error durant l'escriptura, imprimeix un missatge d'error.

Paràmetres

- **data**: Dades a escriure en format cadena.
- **fileName**: Nom del fitxer on s'escriuran les dades.

Classe DataCollector

Classe encarregada de recopilar objectes persistibles i generar un diccionari persistent.

Permet afegir objectes que implementen la interfície IPersistableObject i, mitjançant el mètode run(), genera un PersistentDictionary que conté la representació persistent de tots aquests objectes.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Mètodes

public void addPersistableObject(final IPersistableObject persistableObject)

Afegeix un objecte persistible a la col·lecció.

Paràmetres

- **persistableObject**: Objecte que implementa IPersistableObject per afegir.

public void addPersistableObjects(final IPersistableObject… persistableObject)

Afegeix diversos objectes persistibles a la col·lecció.

Paràmetres

- **persistableObject**: Array d'objectes que implementen IPersistableObject per afegir.

public PersistentDictionary run()

Genera un diccionari persistent amb la representació de tots els objectes persistibles afegits.

Cada objecte persistent s'afegeix al diccionari amb el seu nom de classe com a clau.

Retorna: Un PersistentDictionary amb tots els objectes persistibles codificats.

Classe DataRestorer

Classe encarregada de restaurar l'estat dels objectes persistibles a partir d'un diccionari persistent.

Permet afegir objectes que implementen la interfície IPersistableObject i, mitjançant el mètode run(), restaura la seva informació codificada a partir d'un PersistentDictionary.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Mètodes

public void addPersistableObject(final IPersistableObject persistableObject)

Afegeix un objecte persistible a la col·lecció per a restaurar.

Paràmetres

- **persistableObject**: Objecte que implementa IPersistableObject per afegir.

public void addPersistableObjects(final IPersistableObject… persistableObject)

Afegeix diversos objectes persistibles a la col·lecció per a restaurar.

Paràmetres

- **persistableObject**: Array d'objectes que implementen IPersistableObject per afegir.

public void run(PersistentDictionary dictionary)

Restaura l'estat dels objectes persistibles a partir d'un diccionari persistent.

Cerca en el diccionari persistent la informació corresponent a cada objecte per nom de classe i crida el seu mètode decode() per carregar l'estat.

Paràmetres

- **dictionary**: El PersistentDictionary que conté l'estat codificat dels objectes.

Classe GameLoader

Classe responsable de carregar una partida desada.

Utilitza un DataRestorer per restaurar l'estat dels objectes, un IDeserializer per deserialitzar les dades guardades, i un ISaveReader per llegir les dades del fitxer de guardat.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Camps

- **dataRestorer**: Objecte encarregat de restaurar l'estat dels objectes persistibles
- **deserializer**: Objecte encarregat de deserialitzar les dades llegides
- **saveReader**: Objecte encarregat de llegir les dades desades des del fitxer
- **fileName**: Nom del fitxer on es troben les dades guardades

Constructores

public GameLoader(DataRestorer dataRestorer, IDeserializer deserializer, ISaveReader saveReader, String fileName)

Constructor de la classe GameLoader.

Paràmetres

- **dataRestorer**: Objecte encarregat de restaurar l'estat dels objectes persistibles.
- **deserializer**: Objecte encarregat de deserialitzar les dades llegides.
- **saveReader**: Objecte encarregat de llegir les dades desades des del fitxer.
- **fileName**: Nom del fitxer on es troben les dades guardades.

Mètodes

public void run()

Executa el procés de càrrega de la partida.

Llegeix les dades desades, les deserialitza a un PersistentDictionary, i restaura l'estat dels objectes utilitzant el DataRestorer.

Relacions

- Relació d'agregació amb DataRestorer
- Relació d'agregació amb IDeserializer
- Relació d'agregació amb ISaveReader

Classe GameSaver

Classe responsable de desar l'estat actual de la partida.

Utilitza un DataCollector per recollir l'estat dels objectes persistibles, un ISerializer per serialitzar aquestes dades, i un ISaveWriter per escriure les dades en un fitxer.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Camps

- **dataCollector**: Objecte encarregat de recollir l'estat dels objectes persistibles
- **serializer**: Objecte encarregat de serialitzar les dades abans de desar-les
- **saveWriter**: Objecte encarregat d'escriure les dades serialitzades al fitxer
- **fileName**: Nom del fitxer on es guardaran les dades de la partida

Constructores

public GameSaver(DataCollector dataCollector, ISerializer serializer, ISaveWriter saveWriter, String fileName)

Constructor de la classe GameSaver.

Paràmetres

- **dataCollector**: Objecte encarregat de recollir l'estat dels objectes persistibles.
- **serializer**: Objecte encarregat de serialitzar les dades.
- **saveWriter**: Objecte encarregat d'escriure les dades en el fitxer.
- **fileName**: Nom del fitxer on es guardaran les dades.

Mètodes

public void run()

Executa el procés de desament de la partida.

Recull l'estat dels objectes, els serialitza i escriu les dades al fitxer especificat.

Relacions

- Relació d'agregació amb DataCollector
- Relació d'agregació amb ISerializer
- Relació d'agregació amb ISaveWriter

Classe PersistentArray

Representa un objecte persistent que conté una llista d'altres objectes persistents.

Aquesta classe permet emmagatzemar un array d'objectes persistents sota un mateix nom, proporcionant accés a cada element i a la mida de l'array.

Autor: Gerard Gascón

Cardinalitat: Una instància per cada array present en un arxiu de guardat

Camps

- **elements**: Llista d'elements continguts en aquest array persistent

Constructores

public PersistentArray(String name)

Constructor que inicialitza l'array amb un nom i una llista buida d'elements.

Paràmetres

- **name**: Nom associat a aquest array persistent.

public PersistentArray(String name, List elements)

Constructor que inicialitza l'array amb un nom i una llista d'elements específica.

Paràmetres

- **name**: Nom associat a aquest array persistent.
- **elements**: Llista d'objectes persistents que contindrà l'array.

Mètodes

public int getLength()

Retorna la quantitat d'elements que conté l'array persistent.

Retorna: Nombre d'elements de l'array.

public Object getValue()

Retorna el valor associat a aquest objecte persistent. En aquest cas, retorna la llista d'elements.

Retorna: La llista d'objectes persistents que formen l'array.

public T parse(Class type) throws RuntimeException

Mètode no suportat per a aquest tipus d'objecte. Llança una excepció si es prova de convertir l'array a un altre tipus.

Llança una RuntimeException Sempre llança excepció perquè no es pot parsejar un array.

Paràmetres

- **type**: Tipus al qual es vol parsejar (no suportat).

public void add(PersistentObject value)

Afegeix un nou element a l'array persistent.

Paràmetres

- **value**: Objecte persistent a afegir.

public PersistentObject get(int index)

Retorna l'element situat en una posició concreta de l'array.

Retorna: Objecte persistent que es troba en la posició indicada.

Paràmetres

- **index**: Índex de l'element a retornar.

Relacions

- Relació d'agregació amb PersistentObject

Classe PersistentDictionary

Representa un objecte persistent que conté un diccionari d'objectes persistents.

Aquesta classe permet emmagatzemar un conjunt de parells clau-valor, on la clau és una cadena i el valor és un objecte persistent.

Autor: Gerard Gascón

Cardinalitat: Una instància per cada diccionari present en un arxiu de guardat

Camps

- **dictionary**: Diccionari intern que conté els objectes persistents associats a una clau

Constructores

public PersistentDictionary()

Constructor per defecte que inicialitza un diccionari sense nom i buit.

public PersistentDictionary(String name)

Constructor que inicialitza un diccionari buit amb un nom específic.

Paràmetres

- **name**: Nom associat a aquest diccionari persistent.

public PersistentDictionary(String name, Map<String, PersistentObject> dictionary)

Constructor que inicialitza un diccionari amb un nom i un mapa de contingut específic.

Paràmetres

- **name**: Nom associat a aquest diccionari persistent.
- **dictionary**: Mapa que conté els elements del diccionari.

Mètodes

public Map<String, PersistentObject> getDictionary()

Retorna el diccionari intern de parells clau-valor.

Retorna: El diccionari d'objectes persistents.

public T parse(Class type) throws RuntimeException

Mètode no suportat per a aquest tipus d'objecte. Llança una excepció si es prova de convertir el diccionari a un altre tipus.

Llança una RuntimeException Sempre llança excepció perquè no es pot parsejar un diccionari.

Paràmetres

- **type**: Tipus al qual es vol parsejar (no suportat).

public void add(PersistentObject value)

Afegeix un nou objecte persistent al diccionari, associat amb el seu nom.

Paràmetres

- **value**: Objecte persistent que s'afegirà al diccionari.

public PersistentObject get(String key)

Retorna l'objecte persistent associat a una clau determinada.

Retorna: Objecte persistent associat a la clau, o null si no existeix.

Paràmetres

- **key**: Clau que identifica l'objecte.

public PersistentDictionary toDictionary()

Converteix l'objecte en un diccionari.

Retorna: El mateix objecte, ja que aquest ja és un diccionari.

Relacions

- Relació d'agregació amb PersistentObject

Classe PersistentObject

Representa un objecte persistent genèric que conté un nom i un valor.

Aquesta classe serveix com a contenidor bàsic per a dades que es poden serialitzar i deserialitzar, associant un nom amb un valor genèric.

Autor: Gerard Gascón

Cardinalitat: Una instància per cada objecte present en l'arxiu de guardat.

Camps

- **name**: Nom identificador de l'objecte persistent
- **value**: Valor associat a l'objecte persistent, pot ser de qualsevol tipus

Constructores

public PersistentObject(String name, Object value)

Constructor que inicialitza l'objecte persistent amb un nom i un valor.

Paràmetres

- **name**: Nom de l'objecte persistent.
- **value**: Valor associat a l'objecte.

Mètodes

public void setName(String name)

Assigna un nou nom a l'objecte persistent.

Paràmetres

- **name**: Nou nom que es vol assignar.

public String getName()

Retorna el nom associat a aquest objecte persistent.

Retorna: El nom de l'objecte.

public Object getValue()

Retorna el valor associat a aquest objecte persistent.

Retorna: Valor de l'objecte.

public T parse(Class type) throws ClassCastException

Intenta convertir el valor a un tipus específic.

Retorna: El valor convertit al tipus especificat.

Llança una ClassCastException Si el valor no pot ser convertit al tipus especificat.

Paràmetres

- **type**: Classe del tipus al qual es vol fer el càsting.

public PersistentDictionary toDictionary()

Converteix l'objecte en un diccionari.

Retorna: Una versió del mateix objecte convertit en un diccionari.

Interfície IDeserializer

Interfície que defineix el contracte per a la deserialització de dades.

Implementacions d'aquesta interfície han de convertir una cadena de text (normalment JSON o similar) a un objecte PersistentObject o alguna subclasse d'aquest.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Mètodes

abstract T deserialize(String data, Class clazz)

Deserialitza una cadena de dades en un objecte PersistentObject del tipus especificat.

Retorna: Objecte deserialitzat de tipus T.

Paràmetres

- **data**: Dades en format de cadena (per exemple JSON) que es volen deserialitzar.
- **clazz**: Classe de l'objecte PersistentObject esperat.

Interfície IPersistableObject

Interfície que defineix un objecte persistible.

Aquest objecte pot codificar-se en un PersistentDictionary per a la seva serialització, i pot ser decodificat a partir d'un PersistentDictionary per restaurar el seu estat.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Mètodes

abstract PersistentDictionary encode()

Codifica l'objecte actual en un PersistentDictionary.

Retorna: PersistentDictionary que representa l'estat serialitzat de l'objecte.

abstract void decode(PersistentDictionary data)

Decodifica i restaura l'estat de l'objecte a partir del PersistentDictionary proporcionat.

Paràmetres

- **data**: PersistentDictionary que conté l'estat serialitzat per restaurar.

Interfície ISaveReader

Interfície per llegir dades desades des d'un fitxer.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Mètodes

abstract String read(String fileName)

Llegeix el fitxer de guardat i el converteix en una cadena.

Retorna: null si el fitxer no existeix, String en cas contrari.

Paràmetres

- **fileName**: Nom del fitxer a llegir.

abstract boolean exists(String fileName)

Comprova que el fitxer de guardat existeixi.

Retorna: true si el fitxer existeix.

Paràmetres

- **fileName**: Nom del fitxer de guardat.

Interfície ISaveWriter

Interfície per escriure dades a un fitxer de guardat.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Mètodes

abstract void write(String data, String fileName)

Escriu les dades al fitxer especificat.

Paràmetres

- **data**: Dades a escriure.
- **fileName**: Nom del fitxer on escriure les dades.

Interfície ISerializer

Interfície per serialitzar un PersistentObject a String.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Mètodes

abstract String serialize(PersistentObject object)

Serialitza un PersistentObject a una representació en String.

Retorna: La representació en String de l'objecte.

Paràmetres

- **object**: Objecte persistent a serialitzar.

2.2. Capa de dades

Classe Anchors

Classe que representa el conjunt de caselles *anchor* actives del tauler. Una casella *anchor* és una posició des de la qual es pot iniciar una jugada nova. Aquest conjunt s'utilitza per determinar els punts vàlids d'inici de paraules segons l'estat actual del joc.

Autor: Albert Usero

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa.

Camps

- **anchors**: Llista de coordenades de les caselles *anchor*.

Constructores

public Anchors()

Crea un conjunt buit de caselles *anchor*.

Mètodes

public void addAnchor(int x, int y)

Afegeix una nova casella *anchor* al conjunt.

Paràmetres

- **x**: Coordenada X de la casella a afegir.
- **y**: Coordenada Y de la casella a afegir.

public void removeAnchor(int x, int y)

Elimina una casella *anchor* del conjunt.

Paràmetres

- **x**: Coordenada X de la casella a eliminar.
- **y**: Coordenada Y de la casella a eliminar.

public int getSize()

Retorna: El nombre total de caselles anchor actuals.

public Vector2 getAnchor(int position)

Retorna: La casella anchor en una posició específica del conjunt. Llança IndexOutOfBoundsException si la posició és invàlida.

Paràmetres

- **position**: Índex dins la llista d'*anchors*.

public boolean exists(int x, int y)

Comprova si una casella determinada existeix com a anchor al conjunt.

Retorna: Cert si la casella donada és un *anchor*, fals en cas contrari.

Paràmetres

- **x**: Coordenada X de la casella a comprovar.
- **y**: Coordenada Y de la casella a comprovar.

public Anchors rotate(int boardSize)

Genera una nova instància de Anchors amb les caselles anchor rotades 90 graus en sentit horari, utilitzant la mida del tauler com a referència.

Retorna: Anchors del tauler rotat.

Paràmetres

- **boardSize**: Mida lateral del tauler (assumint tauler quadrat).

Classe Board

Representa el tauler del joc de Scrabble. Manté les fitxes col·locades i la informació de les caselles amb bonificació de puntuació. Aquesta classe proporciona funcionalitats bàsiques com col·locar fitxes, comprovar l'estat de les caselles i rotar el tauler.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa.

Camps

- **placedTiles**: Matriu de fitxes col·locades al tauler. Les caselles buides contenen null.
- **premiumTiles**: Matriu que representa les caselles de bonificació (doble/triple paraula o lletra).
- **empty**: Indica si el tauler està buit (sense cap fitxa col·locada).

Constructores

protected Board(int size)

Crea un tauler amb la mida donada.

Paràmetres

- **size**: la mida (amplada i alçada) del tauler quadrat.

Mètodes

public int getSize()

Retorna la mida d'un costat del tauler.

Retorna: la mida del tauler.

public void placePiece(Piece piece, int x, int y)

Col·loca una fitxa a les coordenades indicades del tauler. No fa res si la casella ja està ocupada.

Paràmetres

- **piece**: la fitxa a col·locar.
- **x**: índex de columna (comença a 0).
- **y**: índex de fila (comença a 0).

public boolean isCenter(int x, int y)

Comprova si les coordenades donades corresponen al centre del tauler. Normalment, la primera paraula s'ha de col·locar passant pel centre.

Retorna: true si la casella és el centre del tauler.

Paràmetres

- **x**: índex de columna.
- **y**: índex de fila.

public boolean isCellEmpty(int x, int y)

Comprova si una casella està buida i dins dels límits del tauler.

Retorna: true si la casella és vàlida i buida.

Paràmetres

- **x**: coordenada x.
- **y**: coordenada y.

public Piece getCellPiece(int x, int y)

Retorna la fitxa a la casella especificada.

Retorna: la fitxa en la casella, o null si està buida.

Paràmetres

- **x**: coordenada x.
- **y**: coordenada y.

public boolean isPremiumTile(int x, int y)

Comprova si una casella és una casella de bonificació.

Retorna: true si és una casella de bonificació.

Paràmetres

- **x**: coordenada x.
- **y**: coordenada y.

public PremiumTileType getPremiumTileType(int x, int y)

Retorna el tipus de bonificació d'una casella concreta.

Retorna: el tipus de casella de bonificació, o null si no n'és.

Paràmetres

- **x**: coordenada x.
- **y**: coordenada y.

protected void setPremiumTile(int x, int y, PremiumTileType type)

Assigna un tipus de casella de bonificació a una ubicació específica.

Paràmetres

- **x**: coordenada x.
- **y**: coordenada y.
- **type**: el tipus de bonificació a assignar.

public boolean isCellValid(int x, int y)

Comprova si les coordenades donades estan dins dels límits del tauler.

Retorna: true si les coordenades són vàlides

Paràmetres

- **x**: coordenada x
- **y**: coordenada y

public boolean isEmpty()

Retorna si el tauler està buit (sense cap fitxa col·locada).

Retorna: true si no hi ha cap fitxa, false altrament

public Board rotate()

Retorna una còpia del tauler rotat 90 graus en sentit antihorari.

Retorna: una còpia rotada del tauler

private Piece[][] getRotatedPieces()

Retorna una matriu de fitxes rotada 90 graus en sentit antihorari.

Retorna: la matriu de fitxes rotada

protected abstract Board copy()

Crea i retorna una còpia del tauler. Important: No inclou les fitxes col·locades.

Retorna: una còpia del tauler

public PersistentDictionary encode()

Codifica l'estat del tauler en un diccionari per persistència.

Retorna: el diccionari amb les dades del tauler

public void decode(PersistentDictionary data)

Decodifica les dades persistents per reconstruir l'estat del tauler.

Paràmetres

- **data**: diccionari amb les dades emmagatzemades

private void updateEmptyState()

Actualitza l'estat de "buit" del tauler en funció de si hi ha fitxes col·locades.

public abstract BoardType getType()

Retorna el tipus de tauler que és.

Retorna: BoardType indicant el tipus de tauler

Relacions

- Relació d'agregació amb Piece

Enumeració BoardType

Defineix els tipus de tauler disponibles per al joc de Scrabble. Cada tipus pot tenir una mida, disposició o conjunt de regles diferents

Camps

- **Junior**: Una versió simplificada de mida 11x11 per a jugadors més joves.
- **Standard**: El tauler oficial de mida 15x15 utilitzat en el Scrabble clàssic.
- **Super**: Una variant ampliada de mida 21x21.

Classe JuniorBoard

Representa un tauler de Scrabble configurat per a la variant Junior. Inicialitza un tauler d'11x11 amb les posicions específiques de les caselles especials.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Constructores

public JuniorBoard()

Crea una instància de JuniorBoard amb una disposició d'11x11 i configura totes les caselles especials (doble/multiplicador de paraula o lletra).

Mètodes

private void setDoubleWordTiles()

Estableix les caselles de doble paraula (Double Word) al tauler Junior.

private void setTripleWordTiles()

Estableix les caselles de triple paraula (Triple Word) al tauler Junior.

private void setDoubleLetterTiles()

Estableix les caselles de doble lletra (Double Letter) al tauler Junior.

private void setTripleLetterTiles()

Estableix les caselles de triple lletra (Triple Letter) al tauler Junior.

protected Board copy()

Crea i retorna una còpia del tauler JuniorBoard. Aquesta còpia manté la mateixa configuració però no les fitxes col·locades.

Retorna: una nova instància de JuniorBoard

public BoardType getType()

Retorna el tipus de tauler que és.

Retorna: BoardType.Junior

Enumeració PremiumTileType

Enum que representa els tipus de caselles especials en un tauler de Scrabble. Aquestes caselles multipliquen el valor de la lletra col·locada o de tota la paraula.

Autor: Gerard Gascón

Camps

- **QuadrupleWord**: Multiplica la puntuació de la paraula per 4.
- **TripleWord**: Multiplica la puntuació de la paraula per 3.
- **DoubleWord**: Multiplica la puntuació de la paraula per 2.
- **QuadrupleLetter**: Multiplica la puntuació de la lletra per 4.
- **TripleLetter**: Multiplica la puntuació de la lletra per 3.
- **DoubleLetter**: Multiplica la puntuació de la lletra per 2.

Classe StandardBoard

Representa un tauler de Scrabble configurat amb el disseny estàndard de 15x15. Inicialitza totes les caselles especials segons les regles oficials del joc.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Constructores

public StandardBoard()

Crea una instància de StandardBoard amb una disposició de 15x15 i assigna les posicions de totes les caselles especials del tauler estàndard.

Mètodes

private void setDoubleWordTiles()

Assigna les caselles que multipliquen la puntuació de la paraula per 2.

private void setTripleWordTiles()

Assigna les caselles que multipliquen la puntuació de la paraula per 3.

private void setDoubleLetterTiles()

Assigna les caselles que multipliquen la puntuació de la lletra per 2.

private void setTripleLetterTiles()

Assigna les caselles que multipliquen la puntuació de la lletra per 3.

protected Board copy()

Retorna una còpia del tauler estàndard amb la mateixa configuració.

Retorna: una nova instància de StandardBoard

public BoardType getType()

Retorna el tipus de tauler que és.

Retorna: BoardType.Standard

Classe SuperBoard

Representa un tauler de Scrabble de 21x21 utilitzat en modes de joc ampliats o personalitzats. Aquest tauler inclou caselles premium addicionals com les de paraula i lletra quadruplicada.

Autor: Gerard Gascón

Constructores

public SuperBoard()

Construeix un SuperBoard amb un disseny de 21x21 i estableix totes les posicions de caselles prèmium extenses.

Mètodes

private void setDoubleWordTiles()

Estableix les posicions de les caselles que dupliquen el valor de la paraula.

private void setTripleWordTiles()

Estableix les posicions de les caselles que tripliquen el valor de la paraula.

private void setQuadrupleWordTiles()

Estableix les posicions de les caselles que quadrupliquen el valor de la paraula.

private void setDoubleLetterTiles()

Estableix les posicions de les caselles que dupliquen el valor de la lletra.

private void setTripleLetterTiles()

Estableix les posicions de les caselles que tripliquen el valor de la lletra.

private void setQuadrupleLetterTiles()

Estableix les posicions de les caselles que quadrupliquen el valor de la lletra.

protected Board copy()

Crea una còpia d'aquest tauler.

Retorna: una nova instància de SuperBoard amb la mateixa configuració.

public BoardType getType()

Retorna el tipus de tauler que és.

Retorna: BoardType.Super

Classe CatalanCrossChecks

Implementació de CrossChecks pel català. Inclou les peces especials del català: Ç, L·L i NY.

Autor: Albert Usero

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa

Camps

- **letters**: Peces de l'anglès + posició 27 per la Ç, 28 per la L·L i 29 per NY. Faran referència a la seva posició als BitSets.

Constructores

public CatalanCrossChecks(int boardSize)

Crea una nova instància de CrossChecks pel català.

Paràmetres

- **boardSize**: Mida del tauler (assumint que es quadrat).

Mètodes

public String[] getLetters()

Obté el conjunt de peces vàlides pel català.

Retorna: Array de Strings amb totes les peces del català.

public int getNumberOfLetters()

Obté el nombre total de peces diferents del català

Retorna: Nombre total de peces del català.

public Boolean ableToPlace(int x, int y, String letter)

Determina si es pot col·locar una peça específica en una posició del tauler per peces en català.

Retorna: True si es pot col·locar la peça, False altrament

Paràmetres

- **x**: Coordenada x de la posició a comprovar
- **y**: Coordenada y de la posició a comprovar
- **letter**: Peça a verificar

private int getNumLetter(String letter)

Obté l'índex numèric corresponent a una peça que referència la seva posició als BitSets. Contempla les peces del català.

Retorna: Índex numèric de la peça.

Paràmetres

- **letter**: Peça a convertir

protected CrossChecks copy()

Crea una còpia dels Catalancrosschecks actuals.

Retorna: Nova instància de Catalancrosschecks amb la mateixa configuració que l'actual.

Classe CrossChecks

Classe que representa els crosschecks actuals del board. Un crosscheck és una casella que emmagatzema informació sobre quines peces no es poden col·locar a aquella casella tenint en compte els extrems de paraules al tauler i en els eixos horitzontal i vertical.

Autor: Albert Usero

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa

Camps

- **crossChecks**: Matriu de BitSets per emmagatzemar i representar els CrossChecks. Cada BitSet representa les peces permeses o no per a una posició. 0 permesa, 1 no permesa. La posició dels bits al BitSet la lletra: 0-a,1-b,2-c etc. Caràcters especials de certs idiomes al final del bit set.
- **boardSize**: Mida del tauler sobre el qual s'emmagatzemen els crosschecks

Constructores

public CrossChecks(int boardSize)

Crea uns nous CrossChecks per la mida de tauler donada. Inicialitzats perquè qualsevol lletra es pot col·locar a qualsevol posició.

Paràmetres

- **boardSize**: Mida del tauler del board sobre el qual s'han de crear els CrossChecks

Mètodes

public abstract String[] getLetters()

Obté el conjunt de peces vàlides per aquest idioma.

Retorna: Array de Strings amb totes les peces possibles

public abstract int getNumberOfLetters()

Obté el nombre total de peces diferents per a aquest idioma.

Retorna: Nombre total de peces

public abstract Boolean ableToPlace(int x, int y, String letter)

Determina si es pot col·locar una peça específica en una posició del tauler.

Retorna: True si es pot col·locar la peça, False altrament

Paràmetres

- **x**: Coordenada x de la posició a comprovar
- **y**: Coordenada y de la posició a comprovar
- **letter**: Peça a verificar

public BitSet getCrossCheck(int x, int y)

Obté el BitSet de crosschecks per a una posició específica del tauler.

Retorna: BitSet que representa les peces permeses/denegades

Paràmetres

- **x**: Coordenada x de la posició a accedir
- **y**: Coordenada y de la posició a accedir

public void setCrossCheck(int x, int y, int value)

Estableix a cert valor (1 no permès, 0 permès) un bit d'un crosscheck específic per a una posició del tauler.

Paràmetres

- **x**: Coordenada x de la posició al tauler
- **y**: Coordenada y de la posició al tauler
- **value**: Índex de la peça a marcar com a permès/denegat

private void setAllBits(int x, int y, BitSet bits)

Estableix tots els bits de crosscheck per a una posició específica.

Paràmetres

- **x**: Coordenada x de la posició al tauler
- **y**: Coordenada y de la posició al tauler
- **bits**: BitSet complet per a substituir els valors actuals

protected abstract CrossChecks copy()

Crea una còpia dels crosschecks actuals.

Retorna: Nova instància de CrossChecks amb la mateixa configuració que l'actual.

public CrossChecks rotate()

Rota els crosschecks 90 graus en sentit horari.

Retorna: Nova instància de CrossChecks amb la matriu rotada

Classe EnglishCrossChecks

Implementació de CrossChecks per l'anglès. Inclou les peces especials del català: Ç, L·L i NY.

Autor: Albert Usero

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa

Camps

- **letters**: Peces de l'anglès. Faran referència a la seva posició als BitSets.

Constructores

public EnglishCrossChecks(int boardSize)

Crea una nova instància de CrossChecks per l'anglès.

Paràmetres

- **boardSize**: Mida del tauler

Mètodes

public String[] getLetters()

Obté el conjunt de peces vàlides per l'anglès.

Retorna: Array de Strings amb totes les peces de l'anglès

public int getNumberOfLetters()

Obté el nombre total de peces diferents de l'anglès

Retorna: Nombre total de peces de l'anglès

public Boolean ableToPlace(int x, int y, String letter)

Determina si es pot col·locar una peça específica en una posició del tauler per peces en anglès.

Retorna: True si es pot col·locar la peça, False altrament

Paràmetres

- **x**: Coordenada x de la posició a comprovar
- **y**: Coordenada y de la posició a comprovar
- **letter**: Peça a verificar

private int getNumLetter(String letter)

Obté l'índex numèric corresponent a una peça que referència la seva posició als BitSets.

Retorna: Índex numèric de la peça.

Paràmetres

- **letter**: Peça a convertir

protected CrossChecks copy()

Crea una còpia dels EnglishCrosschecks actuals.

Retorna: Nova instància de EnglishCrosschecks amb la mateixa configuració que l'actual.

Classe SpanishCrossChecks

Implementació de CrossChecks pel castellà. Inclou les peçes especials del castellà: Ñ, RR, LL i CH.

Autor: Albert Usero

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa

Camps

- **letters**: Peces de l'anglès + posició 27 per la Ñ, 28 per la RR, 29 per LL i 30 per CH. Faran referència a la seva posició als BitSets.

Constructores

public SpanishCrossChecks(int boardSize)

Crea una nova instància de CrossChecks pel castellà.

Paràmetres

- **boardSize**: Mida del tauler

Mètodes

public String[] getLetters()

Obté el conjunt de peces vàlides pel castellà

Retorna: Array de Strings amb totes les peces del castellà

public int getNumberOfLetters()

Obté el nombre total de peces diferents del castellà

Retorna: Nombre total de peces del castellà

public Boolean ableToPlace(int x, int y, String letter)

Determina si es pot col·locar una peça específica en una posició del tauler per peces en castellà.

Retorna: True si es pot col·locar la peça, False altrament

Paràmetres

- **x**: Coordenada x de la posició a comprovar
- **y**: Coordenada y de la posició a comprovar
- **letter**: Peça a verificar

private int getNumLetter(String letter)

Obté l'índex numèric corresponent a una peça que referència la seva posició als BitSets. Contempla les peces del castellà.

Paràmetres

- **letter**: Peça a convertir

protected CrossChecks copy()

Crea una còpia dels SpanishCrosschecks actuals.

Classe DAWG

Classe per representar un DAWG (Directed Acyclic Word Graph)

Autor: Albert Usero

Cardinalitat: Una instància en tot el programa

Camps

- **root**: Node arrel
- **uniqueNodes**: Mapa que relaciona números de hash amb nodes per emmagatzemar nodes únics

Constructores

public DAWG()

Crea un DAWG buit

Mètodes

public Node getRoot()

Retorna: Node arrel

public Node getNode(int hash)

Retorna: Node al qual correspon el hash donat

Paràmetres

- **hash**: Número de hash que referència al node a retornar

public void addNode(Node node)

Afegeix un node al conjunt de nodes únics

Retorna: Índex numèric de la peça.

Paràmetres

- **node**: Node que s'ha d'afegir al conjunt de nodes únics

public void removeNode(int hash)

Elimina un node del conjunt de nodes únics

Paràmetres

- **hash**: Codi hash del node que s'ha d'eliminar del conjunt de nodes únics

Relacions:

- Relació d'agregació amb Node

Classe Node

Classe per representar nodes que formaran el DAWG. Els nodes representen caràcters no peces.

Autor: Albert Usero

Cardinalitat: Una instància per cada node del DAWG

Camps

- **isEndOfWord**: Indica si el node pot arribar a ser un final de paraula
- **character**: Emmagatzema el caràcter que representa el node
- **successors**: Mapa caràcter-Node per indicar els possibles caràcters successors del Node i en cas que existeixin, el Node al qual avançar.
- **hashCode**: Número creat a partir de les característiques del node que pretén representar-lo de manera única i inequívocament.
- **depth**: Indica a quants nodes de distància del node arrel a on està situat aquest node.
- **parent**: Indica el node predecessor al node actual.

Constructores

public Node(Character character, int depth, Node parent, boolean isEndOfWord)

Crea un node amb les característiques donades. Calcula el seu codi de hash actual.

Paràmetres

- **character**: Caràcter que representarà el node
- **depth**: Profunditat respete l'arrel del node
- **parent**: node predecessor al node creat
- **isEndOfWord**: Booleà que indica si el node pot ser o no final de paraula.

Mètodes

public Node getSuccessor(Character character)

Obtenim informació sobre els caràcters successors del node i si sí que existeixen retornem el node que els representa.En cas que no existeixi null.

Paràmetres

- **character**: Caràcter del qual volem veure si el node actual té successor.

Retorna: Cert si conté aquell successor, fals altrament

public Map<Character, Node> getSuccessors()

Retorna tots els successors del node actual

Retorna: Mapa caràcter-successors del node actual

public void addSuccessor(Character character, Node successor)

Afegeix un nou successor

Paràmetres

- **character**: Caràcter que representa el successor
- **successor**: Node successor a afegir

public int getDepth()

Retorna: Profunditat del node

public Node getParent()

Retorna: Node predecessor

public boolean isEndOfWord()

Retorna: True si el node és final de paraula, altrament False

public void setEndOfWord(boolean endOfWord)

Modifica el paràmetre endOfWord

Paràmetres

- **endOfWord**: Valor al qual volem establir end of word

public void calculateHashCode()

Calcula i actualitza codi de Hash del node segons els valors que té actualment.

public boolean equals(Object o)

Comparador

Retorna: True si és igual a l'objecte a comparar, False altrament

Paràmetres

- **o**: Objecte amb el qual el volem comparar

public int hashCode()

Retorna: Hash code del node

Classe PlayerDoesNotHavePieceException

Classe per tractar excepcións quan un jugador/a no té una peça en específic. Filla de ScrabbleException

Autor: Albert Usero

Cardinalitat: Una instància en tot el programa

Constructores

public PlayerDoesNotHavePieceException(String message)

Construiex una excepció amb el missatge especificat. Seran sobre la manca d'alguna peça a la mà.

Paràmetres

- **message**: Missatge explicant el motiu de l'excepció.

Classe ScrabbleException

Classe base abstracta per a totes les excepcions específiques del joc de Scrabble.

Aquesta classe és la mare de totes les excepcions que es llancen en el context d'una partida de Scrabble. Estén RuntimeException, cosa que permet utilitzar excepcions no comprovades que poden ser llançades durant el joc quan hi ha infraccions de les regles o accions de joc no vàlides.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Constructores

public ScrabbleException(String message)

Construeix una excepció de tipus Scrabble amb el missatge de detall especificat.

Paràmetres

- **message**: El missatge explicatiu de la causa de l'excepció.

Classe GameData

Representa l'estat de la partida en curs, incloent informació sobre el tauler, el llenguatge, el torn actual, els jugadors i el nombre de torns saltats. Aquesta classe també permet la persistència d'aquestes dades mitjançant encode/decode.

Autor: Biel

Cardinalitat: Una instància en tot el programa

Camps

- **skipCounter**: Nombre de torns consecutius que s'han saltat.
- **turnNumber**: Número del torn actual.
- **boardType**: Tipus de tauler de joc.
- **language**: Llengua utilitzada durant la partida.
- **players**: Jugadors participants en la partida.

Mètodes

public void setSkipCounter(int skipCounter)

Estableix el comptador de salts de torn.

Paràmetres

- **skipCounter**: nombre de torns consecutius saltats

public int getSkipCounter()

Retorna el nombre de torns consecutius saltats.

public void setTurnNumber(int turnNumber)

Estableix el número de torn actual.

Paràmetres

- **turnNumber**: número del torn

public int getTurnNumber()

Retorna el número de torn actual.

public void setLanguage(Language language)

Estableix l'idioma de la partida.

Paràmetres

- **language**: idioma seleccionat

public Language getLanguage()

Retorna l'idioma actual de la partida.

public void setPlayers(Player[] players)

Estableix els jugadors de la partida.

Paràmetres

- **players**: array de jugadors

public Player[] getPlayers()

Retorna l'array de jugadors participants.

public void setBoardType(BoardType boardType)

Estableix el tipus de tauler emprat.

Paràmetres

- **boardType**: tipus de tauler

public BoardType getBoardType()

Retorna el tipus de tauler emprat.

public PersistentDictionary encode()

Serialitza totes les dades del joc en un PersistentDictionary per a poder-les guardar.

public void decode(PersistentDictionary data)

Retorna: número del torn

Reconstrueix l'estat intern a partir d'un PersistentDictionary. Aquesta funció es fa servir per carregar una partida desada.

Paràmetres

- **data**: diccionari amb les dades persistides

Relacions

- Relació d'associació amb Player

Classe Leaderboard

Classe encarregada d'emmagatzemar els registres Score associats a les diferents partides jugades. Aquesta classe permet afegir, obtenir i serialitzar la informació dels resultats per a la seva persistència.

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa.

Relacions

- **Score**: relació d'agregació, indica quins Score pertanyen a la leaderboard.

Camps

- **leaderBoard**: Llista que conté tots els resultats Score registrats.

Mètodes

public Score[] getScoreArray()

Retorna els resultats en forma d'array per facilitar-ne la gestió i el seu ús des de capes superiors, com ara els controladors.

Retorna: Un array de Score amb tots els resultats emmagatzemats.

public void addScore(Score score)

Afegeix un nou resultat a la classificació.

Retorna: diccionari amb les dades persistents de la partida**.**

Paràmetres

- **score**: El nou Score a afegir.

public PersistentDictionary encode()

Codifica els resultats emmagatzemats per tal de permetre la seva persistència. Construeix un diccionari persistent amb la informació actual de la classificació.

Retorna: Un PersistentDictionary amb les dades codificades de la classificació.

public void decode(PersistentDictionary data)

Reconstrueix la classificació a partir de les dades desades en un diccionari persistent. Els resultats recuperats s'afegeixen a la llista interna leaderBoard.

Paràmetres

- **data**: El PersistentDictionary que conté les dades a deserialitzar.

Record Score

Representa les dades rellevants d'un jugador un cop finalitzada una partida. Aquest registre encapsula la puntuació total, si el jugador ha estat guanyador, i el seu nom identificador.

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància per jugador, per partida finalitzada.

Paràmetres

- **scoreValue**: Puntuació total obtinguda pel jugador en la partida.
- **isWinner**: Valor booleà que indica si el jugador ha estat el guanyador de la partida.
- **playerName**: Nom del jugador, utilitzat com a identificador.

Record Movement

Representa un moviment d'una paraula sobre el tauler de Scrabble, incloent-hi les coordenades d'inici (x, y), la direcció del moviment, i la paraula en si.

Nota: La paraula ha d'estar en majúscules. Qualsevol lletra en minúscula es considerarà com una fitxa en blanc al tauler.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Paràmetres

- **word**: La paraula que es col·loca al tauler. Ha d'estar en majúscules.
- **x**: La coordenada x (posició horitzontal) on comença la paraula.
- **y**: La coordenada y (posició vertical) on comença la paraula.
- **direction**: La direcció en què es col·loca la paraula (Vertical o Horitzontal).

Classe Bag

Representa una bossa de peces del joc de taula Scrabble.

Autor: Gina Escofet González

Cardinalitat: Una instància en tot el programa

Constructores

public Bag()

Creadora d'una bossa buida.

Mètodes

public boolean isEmpty()

Revisa l'estat de la bossa.

Retorna: True si està buida, False altrament.

public int getSize()

Revista l'estat de la bossa.

Retorna: La quantitat de peces que hi ha la bossa.

public Piece draw(int index)

Roba i elimina una peça de la bossa.

Retorna: La peça en la posició especificada.

Llança una IndexOutOfBoundsException si l'index no és vàlid. Llança una IllegalStateException si la bossa està buida.

Paràmetres

- **index**: La posició de la peça a agafar (ha de ser vàlida), començant des de zero.

public void add(Piece piece)

Afegeix una peça a la bossa. Llança una IllegalArgumentException si la peça és nul·la.

Paràmetres

- **piece**: La peça que s'afegirà a la bossa.

public PersistentDictionary encode()

Codifica l'estat actual de la bossa en un diccionari persistent.

public void decode(PersistentDictionary data)

Decodifica les dades d'un diccionari persistent per reconstruir l'estat de la bossa. Si les dades són invàlides o buides, inicialitza una bossa buida.

Paràmetres

- **data**: Diccionari persistent que conté les dades de la bossa serialitzades.

Relacions

- Relació d'agregació amb Piece

Classe Piece

Representa una fitxa del joc Scrabble amb una lletra, valor de puntuació i estat buit per fitxes escarrades.

Autor: Gina Escofet González

Cardinalitat: Una per cada piece determinada en el fitxer d'informació de l'idioma, normalment un total de 100 per partida.

Constructores

public Piece(String letter, int value)

Creadora d'una fitxa amb lletra i valor fixos.

Paràmetres

- **letter**: La lletra que es mostra a la fitxa.
- **value**: El valor en punts de la fitxa.

public Piece(String letter, int value, boolean isBlank)

Crea una fitxa escarràs amb lletra configurable.

Paràmetres

- **letter**: La lletra que es mostra a la fitxa.
- **value**: El valor en punts de la fitxa.
- **isBlank**: Indica si és una fitxa buida.

Mètodes

public String letter()

Obté la lletra que mostra la fitxa.

Retorna: La lletra de la fitxa actual.

public boolean isBlank()

Comprova si és una fitxa buida.

Retorna: Cert si és una fitxa buida, fals altrament.

public void setLetter(String letter)

Assigna una lletra a una fitxa buida.

Paràmetres

- **letter**: La nova lletra per a la fitxa buida.

public int value()

Obté el valor en punts de la fitxa.

Retorna: El valor de la fitxa actual.

public boolean equals(Object obj)

Compara si aquesta peça és igual a una altra.

Retorna: Cert si són iguals, fals altrament.

Paràmetres

- **obj**: L'objecte a comparar.

public String toString()

Retorna una representació en cadena de caràcters de la peça

Retorna: La lletra i el valor de la peça, amb indicació si és buida.

Classe Player

Classe que representa un jugador/a amb els seus atributs

Autor: Albert Usero

Cardinalitat: Quantitat de persones i CPU que participaran a una partida

Camps

- **name**: Nom del jugador/a
- **score**:Puntuació actual del jugador/a
- **CPU**: Indica si el jugador és controlat per la CPU (true) o per un humà (false)
- **hand**: Llista de fitxes que té el jugador/a a la mà (màxim 7)

Constructores

public Player(String name, boolean CPU)

Crea un nou jugador/a amb score 0 per defecte

Paràmetres

- **name**: Nom del jugador/a a crear
- **CPU**: Booleà que indica si el jugador/a a crear és controlat per un humà (false) o no (true)

Mètodes

public String getName()

Retorna: Nom del jugador/a

public boolean getCPU()

Retorna: Cert si és CPU, fals altrament

public int getScore()

Retorna: Puntuació actual del jugador/a

public void addScore(int scoretoadd)

Afegeix la quantitat de puntuació de l'input a la puntuació actual del jugador/a

Paràmetres

- **scoretoadd**: Puntuació que volem afegir

public void addPiece(Piece piece)

Afegeix la peça donada a la mà del jugador/a

Paràmetres

- **piece**: Peça que serà afegida

public Piece removePiece(Piece piece)

Elimina la peça donada de la mà del jugador/a

Paràmetres

- **piece**: Peça que serà eliminada

public Piece hasPiece(String piece)

Consulta si el jugador/a té certa peça a la mà i si és així la retorna. Si no tenim la peça, però sí que tenim comodí, es retornarà el comodí.

Paràmetres

- **piece**: Peça sobre la qual volem obtenir informació

Record GameProperties

Representa la configuració i informació bàsica d'una partida de Scrabble. Conté dades com l'idioma escollit, el tipus de tauler, els jugadors humans i les CPU participants.

Autor: Biel Pérez Silvestre

Cardinalitat: Una instància en tot el programa

Paràmetres

- **language**: Idioma de la partida (català, castellà, anglès).
- **boardType**: Tipus de tauler utilitzat (clàssic, duplicat, personalitzat).
- **players**: Llista amb els noms dels jugadors humans.
- **Cpus**: Llista amb els noms o identificadors dels jugadors controlats per la màquina.
- **reload**: Indica si la partida s'ha de carregar des de memòria.

Enumeració Language

Enum que representa els idiomes disponibles per al joc de Scrabble. Cada idioma pot afectar el diccionari i les regles específiques del joc.

Retorna: Peça en cas que sí que existeixi, null altrament

Autor: Biel Pérez

Camps

- **English**: Anglès (English).
- **Catalan**: Català (Catalan).
- **Spanish**: Espanyol (Spanish).

2.3. Capa de domini

Classe DrawActionMaker

Classe que gestiona l'intercanvi de fitxes entre la bossa del joc i la mà d'un jugador. Les fitxes a canviar es retornen a la bossa i es reparteixen noves fitxes a la mà del jugador.

Autor: Gina Escofet González

Cardinalitat: Una instància per cada Player

Constructores

public DrawActionMaker(Bag bag, Player player, IRand rand, IHandView handView, GameStepper stepper, PiecesConverter piecesConverter)

Constructor per defecte d'un DrawActionMaker amb bag, player, rand i handView.

Paràmetres

- **bag**: La bossa de fitxes actual del joc.
- **player**: El jugador que vol canviar fitxes.
- **rand**: Instància del generador de nombres aleatoris.
- **handView**: Interfície que representa la mà actual del jugador.

Mètodes

public void run(String[] word)

Procediment que gestiona l'intercanvi de fitxes entre la mà del jugador i la bossa de fitxes. Llança una IllegalArgumentException si PiecesToSwap és nul. Llança una NotEnoughPiecesInBagException si no hi ha prou fitxes a la bossa per completar l'intercanvi.

Paràmetres

- **word**: Paraula que es canvia per noves fitxes extretes de la bossa.

Relacions

- Relació d'associació amb Bag
- Relació d'associació amb Player
- Relació d'associació amb IHandView

Interfície IHandView

Interfície que defineix els mètodes necessaris per a la visualització i interacció amb les fitxes de la mà d'un jugador en un joc de scrabble.

Autor: Gina Escofet González

Cardinalitat: Una instància en tot el programa

Mètodes

abstract String[] getSelectedPiece()

Obté les fitxes actualment seleccionades per l'usuari.

Retorna: Array de Strings amb la informació de la fitxa seleccionada.

abstract int getSelectedPiecePoints()

Obté els punts associats a les fitxes actualment seleccionades per l'usuari.

Retorna: Puntuació de les fitxes actualment seleccionades segons les regles del joc.

abstract void showPieces(Piece[] pieces)

Actualitza la visualització de les fitxes disponibles.

Paràmetres

- **pieces**: Array de fitxes que s'han de mostrar a la mà de l'usuari.

Classe PlaceActionMaker

Gestiona els passos necessaris per realitzar una acció de col·locació de paraula en el joc de Scrabble. Això inclou validar el moviment, comprovar que la paraula existeix i col·locar la paraula al taulell amb les peces necessàries.

Autor: Gerard Gascón

Cardinalitat: Una instància per cada jugador**.**

Camps

- **movementBoundsChecker**: Component que comprova que el moviment es mantingui dins dels límits del taulell.
- **wordValidator**: Component que valida que una paraula existeixi al diccionari.
- **piecesInHandGetter**: Component que obté les peces necessàries que té el jugador a la mà.
- **movementCleaner**: Component que neteja el moviment, obtenint les peces i posicions necessàries.
- **wordPlacer**: Component que col·loca les peces al taulell.
- **presentPiecesWordCompleter**: Component que completa les paraules que es formen amb les peces presents al taulell.
- **crossCheckUpdater**: Component que actualitza les comprovacions creuades després d'un moviment.
- **stepper**: Component que gestiona el pas del torn i actualitza l'estat del joc.
- **piecesConverter**: Component que converteix paraules en conjunts de peces.
- **board**: El taulell actual del joc.
- **anchorUpdater**: Component que actualitza els punts d'ancoratge del taulell.

Constructores

public PlaceActionMaker(Player player, Bag bag, WordPlacer wordPlacer, GameStepper stepper, PiecesConverter piecesConverter, Board board, AnchorUpdater anchorUpdater, CrossChecks crossChecks, DAWG dawg, IRand rand)

Crea una instància de PlaceActionMaker que gestiona accions de col·locació de paraules.

Paràmetres

- **player**: El jugador que fa l'acció.
- **bag**: La bossa de peces del joc.
- **wordPlacer**: Component que col·loca la paraula al taulell.
- **stepper**: Controlador per avançar la lògica del joc després del moviment.
- **piecesConverter**: Utilitat per convertir paraules en conjunts de peces.
- **board**: El taulell actual del joc.
- **anchorUpdater**: Component que actualitza els punts d'ancoratge.
- **crossChecks**: Gestor de les comprovacions creuades.
- **dawg**: Diccionari DAWG per validar paraules.
- **rand**: Generador aleatori utilitzat per seleccionar peces.

Mètodes

public void run(Movement movement)

Executa una acció de col·locació de paraula al taulell. Aquest mètode fa totes les validacions necessàries i actualitza el taulell, les peces del jugador i l'estat del joc.

Llança una WordDoesNotExistException Si la paraula no existeix al diccionari. Llança una MovementOutsideOfBoardException Si el moviment està fora dels límits del taulell. Llança una PlayerDoesNotHavePieceException Si el jugador no té les peces necessàries.

Paràmetres

- **movement**: El moviment que conté la paraula i la seva posició.

private void assertWordIsConnected(Movement movement, Pair<Piece, Vector2>[] necessaryPiecesPositions)

Verifica que la paraula estigui connectada a altres paraules ja presents al taulell. Si no és així i el taulell no està buit, llança una excepció.

Paràmetres

- **movement**: El moviment realitzat.
- **necessaryPiecesPositions**: Les peces i posicions necessàries per al moviment.

private void assertInitialMoveInCenter(Pair<Piece, Vector2>[] necessaryPiecesPositions)

Comprova que el primer moviment passi pel centre del taulell. Si no és així, llança una excepció.

Paràmetres

- **necessaryPiecesPositions**: Les peces i posicions necessàries per al moviment.

private Piece[] extractNecessaryPieces(Pair<Piece, Vector2>[] necessaryPiecesPositions)

Extreu només les peces necessàries de la llista de parells (peca, posició).

Retorna: Array amb les peces necessàries.

Paràmetres

- **necessaryPiecesPositions**: Llista de parells (peca, posició).

private Vector2[] extractNecessaryPositions(Pair<Piece, Vector2>[] necessaryPiecesPositions)

Extreu només les posicions necessàries de la llista de parells (peca, posició).

Retorna: Array amb les posicions necessàries.

Paràmetres

- **necessaryPiecesPositions**: Llista de parells (peca, posició).

private void assertNewWordsExist(Piece[] pieces, Vector2[] positions)

Comprova que totes les noves paraules formades amb les peces noves existeixin al diccionari.

Paràmetres

- **pieces**: Les peces que s'han col·locat.
- **positions**: Les posicions on s'han col·locat les peces.

private void assertWordExists(String word)

Comprova si una paraula existeix al diccionari, i si no, llança una excepció.

Paràmetres

- **word**: La paraula a validar.

private void assertInsideOfBounds(Movement movement)

Comprova que el moviment estigui dins dels límits del taulell. Si està fora, llança una excepció.

Paràmetres

- **movement**: El moviment a validar.

Relacions

- Relació d'associació amb MovementBoundsChecker
- Relació d'associació amb WordValidator
- Relació d'agregació amb PiecesInHandGetter
- Relació d'associació amb MovementCleaner
- Relació d'agregació amb WordPlacer
- Relació d'associació amb PresentPiecesWordCompleter
- Relació d'associació amb CrossCheckUpdater
- Relació d'associació amb GameStepper
- Relació d'associació amb Board
- Relació d'associació amb AnchorUpdater

Classe SkipActionMaker

Classe que gestiona l'acció de passar el torn en el joc de Scrabble. Quan s'executa, simplement avança el joc al següent torn sense realitzar cap moviment.

Autor: Gerard Gascón

Camps

- **stepper**: Component que controla la lògica d'avançar el torn del joc.

Constructores

public SkipActionMaker(GameStepper stepper)

Crea una instància de SkipActionMaker amb el controlador de torns.

Paràmetres

- **stepper**: Component per avançar la lògica del joc.

Mètodes

public void run()

Executa l'acció de passar el torn, indicant al controlador que s'ha saltat el torn.

Classe AI

Intel·ligència artificial que computa el moviment a realitzar pel jugador controlat per la màquina

Autor: Albert Usero

Autor: Felipe Martínez

Cardinalitat: Una instància per jugador controlat per CPU en partida en joc.

Relacions

- **Player**: relació d'associació.
- **Board**: relació d'associació.
- **Anchors**: relació d'associació.
- **CrossChecks**: relació d'associació.
- **Movement**: relació d'associació.
- **PointCalculator**: relació d'associació.
- **PiecesConverter**: relació d'associació.
- **DAWG**: relació d'associació.

Camps

- **dawg**: Estructura que guarda les paraules vàlides per a la llengua amb la qual es juga.
- **bot**: Jugador al qual la intel·ligència artificial computarà el moviment.
- **board**: Board sobre el qual s'ha de buscar el moviment a realitzar.
- **anchors**: Anchors sobre els quals es pot iniciar un moviment.
- **crossChecks**: CrossChecks que senyalitzen les lletres vàlides a determinades caselles.
- **currentAnchor**: Anchor sobre el qual actualment es busca un moviment.
- **pointCalculator**: Encarregada de calcular els punts donats pels moviments obtinguts.
- **piecesConverter**: Encarregada de transformar les lletres dels moviments obtinguts en peces.
- **bestMove**: Millor moviment trobat.
- **bestScore**: Puntuació donada pel millor moviment trobat.
- **currentDirection**: Direcció sobre la qual es busca el moviment (Horizontal/Vertical).
- **initialPieces**: Número de peces que posseeix el jugador controlat.

Constructores

public AI(PiecesConverter piecesConverter, PointCalculator pointCalculator, DAWG dawg, Board board, Player bot, Anchors anchors, CrossChecks crossChecks)

Crea una nova intel·ligència artificial amb els paràmetres especificats. Per tenir un correcte funcionament cal que si els seus paràmetres poden adoptar diferents idiomes tots estiguin al mateix.

Paràmetres

- **piecesConverter**: Convertidor de peces.
- **pointCalculator**: Calculador de punts.
- **dawg**: DAWG que guarda els nodes que representen les possibles paraules.
- **board**: Tauler sobre el qual es calcularan els moviments.
- **bot**: Jugador al qual l'algorisme calcularà els moviments.
- **anchors**: Estructura de dades que emmagatzema caselles des de les quals es pot efectuar un moviment.
- **crossChecks**: Estructura de dades que senyalitza les lletres vàlides a determinades caselles.

Mètodes

public Movement run()

Computa i retorna el millor possible moviment que resulti en la quantitat més gran de punts.

private void checkVertical()

Actualitza la direcció actual a vertical i comprova possibles moviments en vertical. Això s'efectua comprovant moviments en horitzontal amb el tauler (i estructures de dades relacionades) amb una rotació en 90 del sentit horari. (Es desfà la rotació si el millor moviment resulta ser en vertical)

private void checkHorizontal()

Actualitza la direcció actual a horitzontal i comprova possibles moviments en horitzontal amb el tauler en la posició original.

private void resetGlobals()

Restableix els valors globals als valors inicials.

private void innerRun()

Comprova tots els moviments possibles vàlids amb les peces actuals del jugador en direcció horitzontal. Anirà actualitzant les dades globals bestMove i bestScore i el seu resultat quedarà emmagatzemat allà.

protected void LeftPart(String partialWord, Node node, int limit)

Funció de backtracking que itera sobre cada part esquerra ("prefixos") de les possibles paraules a formar a les condicions actuals (peces a la mà, peces en joc etc.)

Paràmetres

- **partialWord**: Tros de la paraula actual.
- **node**: Node del DAWG al qual s'arriba travessant-lo amb el tros de paraula actual.
- **limit**: Com de lluny podem anar.

protected abstract void processLeftPartSpecialPieces(String partialWord, int limit, Map.Entry<Character, Node> entry)

Tracta casos especials cap a l'esquerra (potser no fa res si no existeixen al llenguatge).

Paràmetres

- **partialWord**: Tros de la paraula actual
- **limit**: Com de lluny podem anar
- **entry**: Caràcter/node següent que estem comprovant

protected abstract void processNextLeftPiece(String partialWord, int limit, Map.Entry<Character, Node> entry, Piece usedPiece)

Comprova que no hi ha cap combinació il·legal (potser no fa res si no existeixen al llenguatge).

Paràmetres

- **partialWord**: Tros de la paraula actual.
- **limit**: Com de lluny podem anar.
- **entry**: Caràcter/node següent que estem comprovant.
- **usedPiece**: Peça utilitzada a la iteració actual.

protected void goToNextLeftPiece(String partialWord, Node node, int limit, Piece usedPiece)

Gestor de backtracking

Paràmetres

- **partialWord**: Tros de la paraula actual.
- **limit**: Com de lluny podem anar.
- **usedPiece**: Peça utilitzada a la iteració actual.

protected void ExtendRight(String partialWord, Node node, Vector2 cell)

Crida a estendre cap a la dreta si és possible. Té en compte possibles els possibles casos els quals ens podem trobar en realitzar la crida. Comprova el tros de paraula que portem per veure si és vàlid.

Paràmetres

- **partialWord**: Tros de paraula actual
- **node**: Node que representa el caràcter final del tros de paraula actual
- **cell**: Casella actual sobre la qual estem estenent

protected abstract void processRightPartSpecialPieces(String partialWord, Vector2 cell, Map.Entry<Character, Node> entry)

Tracta casos especials cap a la dreta (potser no fa res si no existeixen al llenguatge).

Paràmetres

- **partialWord**: Tros de la paraula actual
- **cell**: Posició a la qual ens trobem en expansió
- **entry**: Caràcter/node següent que estem comprovant

protected abstract void extendToNextNewPieceRight(String partialWord, Vector2 cell, Map.Entry<Character, Node> entry, Piece usedPiece)

Estén cap a la dreta fent servir una peça de la mà i comprova casos especials.

Paràmetres

- **partialWord**: Tros de paraula actual
- **cell**: Casella sobre la qual estem estenent
- **entry**: Caràcter/Node següent que estem comprovant
- **usedPiece**: Peça utilitzada

protected abstract void extendToNextExistingPieceRight(String partialWord, Node node, Vector2 cell, Piece placedPiece)

Estén cap a la dreta amb una peça ja col·locada al tauler i comprova casos especials.

Paràmetres

- **partialWord**: Tros de paraula actual
- **node**: Node que referencia a la peça ja col·locada
- **cell**: Casella sobre la qual estem estenent
- **placedPiece**: Peça ja al tauler

protected void goToNextRightPiece(String partialWord, Node nextNode, Vector2 cell, Piece usedPiece)

Gestor de backtracking.

Paràmetres

- **partialWord**: Tros de paraula actual
- **nextNode**: Caràcter/node següent que estem comprovant
- **cell**: Casella de la següent iteració
- **usedPiece**: Peça utilitzada a la iteració actual

protected Node getFinalNode(String word)

Retorna: Node que representa el caràcter final de la paraula donada.

Paràmetres

- **word**: Paraula sobre la qual volem obtenir el node final

protected abstract boolean validExistingWord(Vector2 cell, Board board, char c)

Validem si quan estenem la paraula actual no fem invàlides les paraules presents al board.

Retorna: Cert si la paraula és vàlida, fals en cas contrari.

Paràmetres

- **cell**: Casella on comença la paraula
- **board**: Tauler actual
- **c**: Caràcter a afegir

protected void checkWord(String word, Vector2 cell)

Comprovem si hem fet servir com a mínim una peça de la mà i si passem per algun anchor abans de confirmar que hem trobat una possible jugada. Tractem d'actualitzar la millor jugada i puntuació en cas que sigui correcte.

Paràmetres

- **word**: Paraula que estem comprovant
- **cell**: Casella d'inici del moviment (considerem direcció horitzontal)

private void recalculateMaxScoringWord(String word, Vector2[] posVector, Piece[] pieceArray)

Calcula la puntuació del nou moviment i en cas que sigui millor que l'actual actualitza les variables globals BestMove i BestScore.

Paràmetres

- **word**: Nova paraula trobada
- **posVector**: Posició ón s'efectua el nou moviment
- **pieceArray**: Peces que fem servir per realitzar el nou moviment

Relacions:

- Relació d'agregació amb Anchors
- Relació d'agregació amb Crosschecks
- Relació d'associació amb DAWG
- Relació d'associació amb Player
- Relació d'associació amb Board
- Relació d'associació amb Points calculator
- Relació d'associació amb Pieces Converter

Classe AnchorUpdater

Classe encarregada d'actualitzar els anchors.

Autor: Albert Usero

Autor: Felipe Martínez

Cardinalitat: Una instància al programa.

Relacions

- **board**: relació d'associació.
- **Anchors**: relació d'associació.

Camps

- **anchors**: Anchors a actualitzar.

Constructores

public AnchorUpdater(Anchors anchors, Board board, PiecesConverter piecesConverter)

Crea un nou actualitzador d'anchors. Afegeix l'anchor inicial al centre del tauler.

Paràmetres

- **anchors**: Anchors a actualitzar
- **board**: Tauler sobre el qual s'han creat els anchors
- **piecesConverter**: Convertidor de peces

Mètodes

public void run(Movement move)

Actualitza els anchors segons el moviment realitzat

Paràmetres

- **move**: Moviment realitzat sobre el qual s'han d'actualitzar el anchors.

private void updateHorizontalAnchors(int x, int y, int size)

Actualitza els anchors per un moviment que s'ha efectuat en horitzontal

Paràmetres

- **x**: Coordenada x del moviment
- **y**: Coordenada y del moviment
- **size**: Quantitat de peces utilitzades al moviment

private void updateVerticalAnchors(int x, int y, int size)

Actualitza els anchors per un moviment que s'ha efectuat en vertical

Paràmetres

- **x**: Coordenada x del moviment
- **y**: Coordenada y del moviment
- **size**: Quantitat de peces utilitzades al moviment

Classe CatalanAI

Intel·ligència artificial que computa el moviment a realitzar pel jugador controlat per la màquina i partides on la llengua seleccionada és el català.

Autor: Albert Usero

Autor: Felipe Martínez

Cardinalitat: Una instància en tot el programa

Mètodes

protected void processLeftPartSpecialPieces(String partialWord, int limit, Map.Entry<Character, Node> entry)

Tracta casos especials cap a l'esquerra (NY, L·L).

Paràmetres

- **partialWord**: Tros de la paraula actual
- **limit**: Com de lluny podem anar
- **entry**: Caràcter/node següent que estem comprovant

protected void processNextLeftPiece(String partialWord, int limit, Map.Entry<Character, Node> entry, Piece usedPiece)

Comprova que no hi ha cap combinació il·legal (NY, L·L).

Paràmetres

- **partialWord**: Tros de la paraula actual
- **limit**: Com de lluny podem anar
- **entry**: Caràcter/node següent que estem comprovant
- **usedPiece**: Peça utilitzada a la iteració actual

protected void extendToNextNewPieceRight(String partialWord, Vector2 cell, Map.Entry<Character, Node> entry, Piece usedPiece)

Estén cap a la dreta fent servir una peça de la mà i comprova casos especials (NY, L·L)

Paràmetres

- **partialWord**: Tros de paraula actual
- **cell**: Casella sobre la qual estem estenent
- **entry**: Caràcter/Node següent que estem comprovant
- **usedPiece**: Peça utilitzada

protected void extendToNextExistingPieceRight(String partialWord, Node node, Vector2 cell, Piece placedPiece)

Estén cap a la dreta amb una peça ja col·locada al tauler i comprova casos especials (NY, L·L).

Paràmetres

- **partialWord**: Tros de paraula actual
- **node**: Node que referencia a la peça ja col·locada
- **cell**: Casella sobre la qual estem estenent
- **placedPiece**: Peça ja al tauler

protected void processRightPartSpecialPieces(String partialWord, Vector2 cell, Map.Entry<Character, Node> entry)

Tracta casos especials cap a la dreta (NY, L·L).

Paràmetres

- **partialWord**: Tros de la paraula actual
- **cell**: Posició a la qual ens trobem en expansió
- **entry**: Caràcter/node següent que estem comprovant

protected boolean validExistingWord(Vector2 cell, Board board, char c)

Validem si quan estenem la paraula actual no fem invàlides les paraules presents al board.

Paràmetres

- **cell**: Casella on comença la paraula
- **board**: Tauler actual
- **c**: Caràcter a afegir

Classe CrossCheckUpdater

Classe per actualitzar els CrossChecks.

Autor: Albert Usero

Autor: Felipe Martínez

Cardinalitat: Una instància en tot el programa

Camps

- **crossChecks**: CrossChecks a actualitzar.

Constructores

public CrossCheckUpdater(PiecesConverter pc, CrossChecks crossChecks, Board board, DAWG dawg)

Crea un nou actualitzador de CrossChecks.

Paràmetres

- **pc**: Convertidor de peces.
- **crossChecks**: Crosschecks a actualitzar.
- **board**: Tauler sobre el qual s'han creat els crosschecks.
- **dawg**: DAWG necessari per crear el word validator.

Mètodes

public void run(Movement move)

Executa l'actualitzador de CrossChecks i s'actualitzen en funció del moviment realitzat.

Paràmetres

- **move**: Moviment realitzat sobre el qual s'han d'actualitzar els CrossChecks

private void calculateHorizontalCrossChecks(Movement move)

Calcula els CrossChecks que s'han creat horitzontalment en conseqüència d'un moviment en horitzontal.

Paràmetres

- **move**: Moviment realitzat amb direcció horitzontal.

private boolean canProcessCell(int x, int y)

Serveix per determinar quan s'ha de processar un crosscheck d'una casella per saber si és vàlida o si ja té una peça.

Retorna: True si la casella és vàlida i està buida, False altrament

Paràmetres

- **x**: Posició x de la casella
- **y**: Posició y de la casella

private void calculateHorizontalEndCrossCheck(Movement move, int endOfAddedWord)

Actualitza la casella (en cas que sigui vàlida) immediatament a la dreta del moviment realitzat.

Paràmetres

- **move**: Moviment realitzat en horitzontal.
- **endOfAddedWord**: Coordenada x final del moviment realitzat.

private void calculateHorizontalBeginCrossCheck(Movement move, int beginningOfAddedWord)

Actualitza la casella (en cas que sigui vàlida) immediatament a l'esquerra del moviment realitzat.

Paràmetres

- **move**: Moviment realitzat en horitzontal
- **beginningOfAddedWord**: Coordenada x inicial del moviment realitzat

private void calculateVerticalCrossChecks(Movement move)

Calcula els CrossChecks que s'han creat verticalment en conseqüència d'un moviment en vertical.

Paràmetres

- **move**: Moviment realitzat amb direcció vertical.

private void calculateVerticalEndCrossCheck(Movement move, int endOfAddedWord)

Actualitza la casella (en cas que sigui vàlida) immediatament a baix del moviment realitzat.

Paràmetres

- **move**: Moviment realitzat en vertical
- **endOfAddedWord**: Coordenada y final del moviment realitzat

private void calculateVerticalBeginCrossCheck(Movement move, int beginningOfAddedWord)

Actualitza la casella (en cas que sigui vàlida) immediatament a sobre del moviment realitzat.

Paràmetres

- **move**: Moviment realitzat en vertical
- **beginningOfAddedWord**: Coordenada y inicial del moviment realitzat

private Node getFinalNode(String word)

Retorna: Node que representa el caràcter final de la paraula donada.

Paràmetres

- **word**: Paraula sobre la qual volem obtenir el node final

private boolean isNextNodeTerminal(Node node, String piece)

Serveix per donat l'últim node d'una paraula la qual ja està al tauler comprovar si afegint una peça darrere la paraula continua sent vàlida.

Paràmetres

- **node**: Node actual sobre el qual volem fer l'extensió
- **piece**: Peça que afegirem

Relacions:

- Relació d'associació amb CrossChecks

Classe EnglishAI

Intel·ligència artificial que computa el moviment a realitzar pel jugador controlat per la màquina i partides on la llengua seleccionada és l'anglès.

Autor: Albert Usero

Autor: Felipe Martínez

Cardinalitat: Una instància per jugador per partida en lloc on la llengua seleccionada és l'anglesa.

Mètodes

protected void processLeftPartSpecialPieces(String partialWord, int limit, Map.Entry<Character, Node> entry)

Cap treball en aquesta subclasse.

Paràmetres

- **partialWord**: Tros de la paraula actual
- **limit**: Com de lluny podem anar
- **entry**: Caràcter/node següent que estem comprovant

protected void processNextLeftPiece(String partialWord, int limit, Map.Entry<Character, Node> entry, Piece usedPiece)

Comprova que no hi ha cap combinació il·legal i afegeix la lletra utilitzada.

Paràmetres

- **partialWord**: Tros de la paraula actual
- **limit**: Com de lluny podem anar
- **entry**: Caràcter/node següent que estem comprovant
- **usedPiece**: Peça utilitzada a la iteració actual

protected void extendToNextNewPieceRight(String partialWord, Vector2 cell, Map.Entry<Character, Node> entry, Piece usedPiece)

Estén cap a la dreta fent servir una peça de la mà.

Paràmetres

- **partialWord**: Tros de paraula actual.
- **cell**: Casella sobre la qual estem estenent.
- **entry**: Caràcter/Node següent que estem comprovant.
- **usedPiece**: Peça utilitzada.

protected void extendToNextExistingPieceRight(String partialWord, Node node, Vector2 cell, Piece placedPiece)

Estén cap a la dreta amb una peça ja col·locada al tauler.

Paràmetres

- **partialWord**: Tros de paraula actual.
- **node**: Node que referencia a la peça ja col·locada.
- **cell**: Casella sobre la qual estem estenent.
- **placedPiece**: Peça ja al tauler.

protected void processRightPartSpecialPieces(String partialWord, Vector2 cell, Map.Entry<Character, Node> entry)

Cap treball en aquesta subclasse.

Paràmetres

- **partialWord**: Tros de la paraula actual.
- **cell**: Posició a la qual ens trobem en expansió.
- **entry**: Caràcter/node següent que estem comprovant.

protected boolean validExistingWord(Vector2 cell, Board board, char c)

Validem si quan estenem la paraula actual no fem invàlides les paraules presents al board.

Paràmetres

- **cell**: Casella on comença la paraula.
- **board**: Tauler actual.
- **c**: Caràcter a afegir.

Classe SpanishAI

Intel·ligència artificial que computa el moviment a realitzar pel jugador controlat per la màquina i partides on la llengua seleccionada és el castellà.

Autor: Albert Usero

Autor: Felipe Martínez

Cardinalitat: Una instància per jugador per partida en joc on la llengua seleccionada és el castellà.

Mètodes

protected void processLeftPartSpecialPieces(String partialWord, int limit, Map.Entry<Character, Node> entry)

Tracta casos especials cap a l'esquerra (RR, LL, CH)

Paràmetres

- **partialWord**: Tros de la paraula actual
- **limit**: Com de lluny podem anar
- **entry**: Caràcter/node següent que estem comprovant

protected void processNextLeftPiece(String partialWord, int limit, Map.Entry<Character, Node> entry, Piece usedPiece)

Comprova que no hi ha cap combinació il·legal (RR, LL, CH).

Paràmetres

- **partialWord**: Tros de la paraula actual
- **limit**: Com de lluny podem anar
- **entry**: Caràcter/node següent que estem comprovant
- **usedPiece**: Peça utilitzada a la iteració actual

protected void extendToNextNewPieceRight(String partialWord, Vector2 cell, Map.Entry<Character, Node> entry, Piece usedPiece)

Estén cap a la dreta fent servir una peça de la mà i comprova casos especials (RR, LL, CH).

Paràmetres

- **partialWord**: Tros de paraula actual
- **cell**: Casella sobre la qual estem estenent
- **entry**: Caràcter/Node següent que estem comprovant
- **usedPiece**: Peça utilitzada

protected void extendToNextExistingPieceRight(String partialWord, Node node, Vector2 cell, Piece placedPiece)

Estén cap a la dreta amb una peça ja col·locada al tauler i comprova casos especials (RR, LL, CH)

Paràmetres

- **partialWord**: Tros de paraula actual
- **node**: Node que referencia a la peça ja col·locada
- **cell**: Casella sobre la qual estem estenent
- **placedPiece**: Peça ja al tauler

protected void processRightPartSpecialPieces(String partialWord, Vector2 cell, Map.Entry<Character, Node> entry)

Tracta casos especials cap a la dreta (RR, LL, CH)

Paràmetres

- **partialWord**: Tros de la paraula actual
- **cell**: Posició a la qual ens trobem en expansió
- **entry**: Caràcter/node següent que estem comprovant

protected boolean validExistingWord(Vector2 cell, Board board, char c)

Validem si quan estenem la paraula actual no fem invàlides les paraules presents al board.

Retorna: Cert en cas de ser vàlida, fals en cas contrari.

Paràmetres

- **cell**: Casella on comença la paraula
- **board**: Tauler actual
- **c**: Caràcter a afegir

Interfície IBoard

Interfície que facilita la comunicació entre la capa de domini i la capa de presentació per a les actualitzacions del tauler. Defineix els mètodes per actualitzar el tauler quan l'estat del joc canvia.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Mètodes

abstract void updateBoard()

Dispara una actualització del tauler a la capa de presentació per tal que es refresqui la vista del tauler. Aquest mètode s'invoca quan l'estat del joc o del tauler ha canviat i cal reflectir-ho en la UI.

abstract void updateCell(String piece, int points, int x, int y)

Actualitza una cel·la específica del tauler amb la peça i els punts corresponents.

Paràmetres

- **piece**: La lletra o peça que s'ha col·locat.
- **points**: Els punts assignats a aquesta peça.
- **x**: La coordenada horitzontal de la cel·la.
- **y**: La coordenada vertical de la cel·la.

abstract void setPremiumTile(PremiumTileType type, int x, int y)

Estableix el tipus de casella especial (premium) en una posició determinada del tauler.

Paràmetres

- **type**: El tipus de casella premium (doble lletra, triple paraula, etc.).
- **x**: La coordenada horitzontal de la cel·la.
- **y**: La coordenada vertical de la cel·la.

Classe PointCalculator

Classe responsable de calcular els punts associats a un moviment al tauler de Scrabble. Considera factors com el valor de cada peça, multiplicadors de paraula, bonificacions, i la interacció amb peces ja col·locades al tauler.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Camps

- **board**: El tauler actual del joc
- **wordGetter**: Component per obtenir paraules a partir de peces i posicions

Constructores

public PointCalculator(Board board)

Constructor que crea una instància de PointCalculator amb el tauler proporcionat.

Paràmetres

- **board**: El tauler on es juga la partida

Mètodes

public int run(Vector2[] positions, Piece[] pieces)

Calcula el total de punts d'un moviment tenint en compte les posicions i les peces col·locades. Es té en compte el valor de les peces, multiplicadors de paraula i bonificacions addicionals.

Retorna: El total de punts que es guanyarien amb aquest moviment

Paràmetres

- **positions**: Les posicions on es col·loquen les noves peces
- **pieces**: Les peces que es volen col·locar

private Direction getWordDirection(Vector2[] positions)

Determina la direcció de la paraula a partir de les posicions donades. Retorna Vertical, Horitzontal o null si només hi ha una peça.

Retorna: La direcció de la paraula

Paràmetres

- **positions**: Les posicions de les peces

private int getPresentWordPoints(Vector2[] positions, Piece[] pieces, Direction direction)

Calcula els punts de les paraules que es formen creuant la paraula principal.

Retorna: Punts totals de les paraules creuades

Paràmetres

- **positions**: Posicions de les peces noves
- **pieces**: Peces col·locades
- **direction**: Direcció de la paraula principal

private Piece[] getPresentPieces(Vector2 position, Piece piece, Direction direction)

Obté les peces presents en la direcció perpendicular per formar paraules creuades.

Retorna: Les peces que formen la paraula creuada

Paràmetres

- **position**: Posició de la peça nova
- **piece**: Peça nova
- **direction**: Direcció de la paraula principal

private int getPresentPoints(Vector2 position, Piece[] pieces)

Calcula els punts d'una paraula formada per les peces indicades.

Retorna: Punts totals de la paraula

Paràmetres

- **position**: Posició de la paraula
- **pieces**: Peces de la paraula

private int getAlreadyPresentWordPiecesPoints(Vector2[] positions, Piece[] pieces, Direction direction)

Calcula els punts de les peces que ja estaven presents a la paraula abans d'afegir les noves peces.

Retorna: Punts de les peces ja presents

Paràmetres

- **positions**: Posicions de les peces noves
- **pieces**: Peces noves
- **direction**: Direcció de la paraula

private int getPiecePoints(Vector2[] positions, Piece[] pieces)

Calcula la suma de punts de les peces en les posicions indicades.

Retorna: Punts totals de les peces

Paràmetres

- **positions**: Posicions de les peces
- **pieces**: Peces col·locades

private int getPiecePoints(Piece[] pieces)

Calcula la suma dels valors de les peces indicades.

Retorna: Suma dels punts de les peces

Paràmetres

- **pieces**: Peces de les quals es vol obtenir el valor

private static int getBonus(Vector2[] positions)

Calcula el bonus per fer un bingo (col·locar 7 peces en un sol torn).

Retorna: 50 punts si són 7 peces, 0 en cas contrari

Paràmetres

- **positions**: Posicions de les peces col·locades

private int getWordMultiplier(Vector2[] positions)

Calcula el multiplicador total de paraula a partir de les posicions.

Retorna: Multiplicador resultant

Paràmetres

- **positions**: Posicions de les peces noves

private int getPiecePoints(Vector2 position, Piece piece)

Calcula els punts d'una peça tenint en compte els multiplicadors de lletra.

Retorna: Punts totals de la peça amb multiplicadors

Paràmetres

- **position**: Posició de la peça
- **piece**: Peça que es vol puntuar

private int getLetterMultiplier(Vector2 position)

Obté el multiplicador de lletra segons el tipus de casella premium.

Retorna: Multiplicador de lletra (1 si no hi ha casella especial)

Paràmetres

- **position**: Posició al tauler

private int getWordMultiplier(Vector2 position)

Obté el multiplicador de paraula segons el tipus de casella premium. La casella central també multiplica per 2.

Retorna: Multiplicador de paraula (1 si no hi ha casella especial)

Paràmetres

- **position**: Posició al tauler

Relacions

- Relació d'associació amb Board
- Relació d'associació amb WordGetter

Classe PremiumTileTypeFiller

Classe que s'encarrega de "omplir" la vista del tauler amb la informació sobre les caselles premiades (premium) del tauler.

Aquesta classe recorre el tauler i actualitza la vista perquè mostri correctament els tipus de caselles prèmium, com doble paraula, triple lletra, etc.

També estableix la casella central com a DoubleWord (doble paraula).

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Camps

- **board**: El tauler del joc
- **view**: La interfície que representa la vista del tauler

Constructores

public PremiumTileTypeFiller(Board board, IBoard view)

Constructor que inicialitza la classe amb el tauler i la vista.

Paràmetres

- **board**: El tauler on es volen obtenir les caselles premium
- **view**: La vista que cal actualitzar amb la informació del tauler

Mètodes

public void run()

Recorre totes les posicions del tauler i actualitza la vista posant el tipus de casella premium corresponent a cada posició.

També estableix la casella central com a casella de doble paraula.

Relacions

- Relació d'associació amb Board
- Relació d'associació amb IBoard

Classe PresentPiecesWordCompleter

Aquesta classe s'encarrega d'identificar i completar les paraules que es formen amb les peces noves col·locades al tauler de Scrabble, així com les paraules adjacents que es formen per la interacció amb peces ja existents.

Permet obtenir totes les paraules resultants després d'una jugada.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Camps

- **wordGetter**: Instància que obté les peces que formen paraules al tauler

Constructores

public PresentPiecesWordCompleter(Board board)

Constructor que crea un PresentPiecesWordCompleter a partir del tauler on es juga.

Paràmetres

- **board**: El tauler on es vol detectar i completar les paraules

Mètodes

public String[] run(Vector2[] positions, Piece[] pieces)

Executa la lògica per obtenir totes les paraules formades amb les peces col·locades a les posicions indicades. Primer determina la direcció de la paraula (horitzontal o vertical) i després obté les paraules que s'han format tant en aquesta direcció com en la perpendicular.

Retorna: Un array de Strings amb totes les paraules formades (horitzontals i verticals)

Paràmetres

- **positions**: Les posicions de les peces noves col·locades al tauler
- **pieces**: Les peces noves col·locades

private String[] getPresentWords(Vector2[] positions, Piece[] pieces, Direction direction)

Retorna totes les paraules presents en la direcció indicada, o en ambdues direccions si la direcció és null (jugada amb una sola peça).

Retorna: Array de paraules formades

Paràmetres

- **positions**: Posicions de les peces col·locades
- **pieces**: Peces col·locades
- **direction**: Direcció de la paraula principal (horitzontal, vertical o null)

private String[] getWords(Vector2[] positions, Piece[] pieces, Direction direction, List words)

Obté la paraula principal formada a la direcció donada, i completa amb les paraules adjacents.

Retorna: Array de paraules formades

Paràmetres

- **positions**: Posicions de les peces col·locades
- **pieces**: Peces col·locades
- **direction**: Direcció de la paraula principal
- **words**: Llista on afegir les paraules trobades

private String[] completeAdjacentWords(Vector2[] positions, Piece[] pieces, Direction direction)

Completa i obté les paraules adjacents formades per la interacció amb les peces ja presents perpendicularment a la direcció de la paraula principal.

Retorna: Array de paraules adjacents formades

Paràmetres

- **positions**: Posicions de les peces col·locades
- **pieces**: Peces col·locades
- **direction**: Direcció de la paraula principal

private Piece[] getPresentPieces(Vector2 position, Piece piece, Direction direction)

Obté les peces que formen una paraula en la direcció perpendicular a la direcció principal, a partir de la peça i la seva posició.

Retorna: Array de peces que formen la paraula perpendicular

Paràmetres

- **position**: Posició de la peça
- **piece**: Peça col·locada
- **direction**: Direcció principal de la paraula

private String getWord(Vector2[] positions, Piece[] pieces, Direction direction)

Obté la paraula completa que formen les peces col·locades a les posicions indicades, en la direcció donada.

Retorna: La paraula formada, o null si només és una sola lletra (sense paraula)

Paràmetres

- **positions**: Posicions de les peces col·locades
- **pieces**: Peces col·locades
- **direction**: Direcció de la paraula

private Direction getWordDirection(Vector2[] positions)

Determina la direcció de la paraula a partir de les posicions de les peces. Si només hi ha una peça, retorna null. Si la coordenada x coincideix, la direcció és vertical, sinó horitzontal.

Retorna: Direcció de la paraula (Horizontal, Vertical o null)

Paràmetres

- **positions**: Posicions de les peces col·locades

Relacions

- Relació d'associació amb WordGetter

Classe WordGetter

Aquesta classe s'encarrega d'identificar i recuperar les peces que formen una nova paraula al tauler de Scrabble després de col·locar-hi peces noves. Gestiona els casos en què les peces noves estenen paraules ja existents o formen paraules noves.

Permet obtenir la paraula completa (com a peces) en una direcció determinada després de la jugada.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Camps

- **board**: Tauler on es col·loquen les peces i es formen les paraules

Constructores

public WordGetter(Board board)

Constructor que crea un WordGetter per un tauler determinat.

Paràmetres

- **board**: El tauler on es juguen les peces i es formen paraules

Mètodes

public Piece[] run(Piece[] newPieces, Vector2[] newPositions, Direction direction)

Recupera les peces que formen una nova paraula al tauler després de col·locar peces noves.

Aquest mètode és útil per quan es col·loquen peces que estenen una paraula existent o en formen de noves. Revisa tant les peces noves com les posicions al tauler i retorna totes les peces que formen una paraula contínua en la direcció indicada (horitzontal o vertical).

Retorna: Array de peces que formen la paraula resultant al tauler

Paràmetres

- **newPieces**: Array de peces noves que s'estan col·locant
- **newPositions**: Array de posicions on es col·loquen aquestes peces
- **direction**: Direcció horitzontal o vertical on es forma la paraula

private static int getYPosition(Vector2 position, Direction direction, int offset)

Calcula la coordenada Y segons la direcció de la paraula i un offset.

Retorna: Coordenada Y corresponent

Paràmetres

- **position**: Posició inicial de la paraula
- **direction**: Direcció horitzontal o vertical
- **offset**: Desplaçament per iterar per la paraula

private static int getXPosition(Vector2 position, Direction direction, int offset)

Calcula la coordenada X segons la direcció de la paraula i un offset.

Retorna: Coordenada X corresponent

Paràmetres

- **position**: Posició inicial de la paraula
- **direction**: Direcció horitzontal o vertical
- **offset**: Desplaçament per iterar per la paraula

Relacions

- Relació d'associació amb Board

Classe WordPlacer

La classe WordPlacer és responsable de col·locar paraules (peces) al tauler i actualitzar la puntuació. També notifica a la vista per refrescar el tauler després de col·locar la paraula.

Autor: Gerard Gascón

Cardinalitat: Una instància per cada jugador present a la partida.

Camps

- **board**: El tauler de joc on es col·loquen les peces
- **view**: La vista que s'actualitza després de cada jugada
- **pointCalculator**: Calculador de punts per a la jugada
- **player**: Jugador que fa la jugada

Constructores

public WordPlacer(Player player, Board board, IBoard view, PointCalculator pointCalculator)

Constructor que inicialitza el WordPlacer amb el jugador, tauler, vista i calculador de punts.

Paràmetres

- **player**: Jugador que realitza la jugada
- **board**: Tauler on es col·loquen les peces
- **view**: Vista que s'actualitzarà després de la jugada
- **pointCalculator**: Calculador de punts per a la jugada

Mètodes

public void run(Piece[] pieces, int x, int y, Direction direction)

Col·loca les peces al tauler segons la posició inicial i la direcció. Calcula la puntuació resultant i l'afegeix al jugador. Actualitza la vista per reflectir el canvi al tauler.

Paràmetres

- **pieces**: Array de peces noves a col·locar
- **x**: Posició X d'inici de la paraula
- **y**: Posició Y d'inici de la paraula
- **direction**: Direcció horitzontal o vertical per col·locar la paraula

private Vector2[] placeWordVertical(Piece[] pieces, int x, int y)

Col·loca les peces verticalment a partir de la posició (x, y). Ignora les cel·les ja ocupades.

Retorna: Array amb les posicions on s'han col·locat les peces

Paràmetres

- **pieces**: Peces a col·locar
- **x**: Posició X fixa (columnes)
- **y**: Posició Y inicial (files)

private Vector2[] placeWordHorizontal(Piece[] pieces, int x, int y)

Col·loca les peces horitzontalment a partir de la posició (x, y). Ignora les cel·les ja ocupades.

Retorna: Array amb les posicions on s'han col·locat les peces

Paràmetres

- **pieces**: Peces a col·locar
- **x**: Posició X inicial (columnes)
- **y**: Posició Y fixa (files)

Relacions

- Relació d'associació amb Board
- Relació d'associació amb IBoard
- Relació d'associació amb PointCalculator
- Relació d'associació amb Player

Classe WordAdder

Afegeix una nova paraula al DAWG. Les paraules s'han d'afegir en ordre alfabètic per un correcte funcionament.

Autor: Albert Usero

Cardinalitat: Una instància en tot el programa

Camps

- **dawg**: DAWG al qual afegim paraules.
- **lastWordAdded**: Última paraula emmagatzemada. Servirà per aprofitar que s'han d'afegir paraules en ordre alfabètic.

Constructores

public WordAdder(DAWG dawg)

Crea un nou WordAdder.

Paràmetres

- **dawg**: DAWG sobre el qual afegirem paraules.

Mètodes

public void run(String word)

Executa el WordAdder per afegir una paraula.

Paràmetres

- **word**: Paraula que afegirem al DAWG

private Node addNecessaryNodes(String word, int commonPrefix, Node startingPoint)

Gestiona al DAWG afegir els nodes necessaris per poder representar la paraula donada. Retorna l'últim node creat per representar la paraula i servirà perquè a 'run' es marqui com a final de paraula.

Paràmetres

- **word**: Paraula a afegir.
- **commonPrefix**: Prefix en comú de la paraula a afegir amb l'última afegida.
- **startingPoint**: Node en el qual cal començar a afegir nodes.

private void tryAddSuccessor(Node parent, char currentChar, Node nextNode)

Si el node predecessor no té com a successor un caràcter com el que volem afegir, afegim com a successor el caràcter i node donats.

Paràmetres

- **parent**: Node predecessor
- **currentChar**: Caràcter a considerar per crear o no el successor
- **nextNode**: Node el qual representaría el caràcter en cas que s'afegeixi.

private void setNodeAsEndOfWord(Node current)

Actualitza el paràmetre del node donat que indica si un node és final de paraula a True. També recalcula i propaga el seu codi hash.

Paràmetres

- **current**: Node a modificar.

private Node getStartingPoint(int commonPrefix)

Aprofita que les paraules s'afegeixen en ordre alfabètic i avança en el DAWG el prefix que tenen en comú.

Paràmetres

- **commonPrefix**: Prefix en comú que tenien l'última paraula afegida i la que volem afegir actualment.

private int findCommonPrefixIndex(String prevword, String actword)

Troba el prefix en comú de 2 paraules. Servirà per aprofitar la característica que les paraules s'han d'afegir en ordre alfabètic.

Retorna: Node final al recorrer el DAWG seguint els nodes que formen el prefix.

Paràmetres

- **prevword**: Última paraula introduïda.
- **actword**: Paraula que anem a introduir.

private Node findOrCreateNode(char character, boolean isEndOfWord, int depth, Node parent)

Tracta de trobar un node amb certes característiques. En cas que no existeixi en crea un i el retorna. Altrament retorna el prèviament existent.

Paràmetres

Retorna: Prefix en comú de les 2 paraules donades.

- **character**: Caràcter del node a trobar/crear.
- **isEndOfWord**: Característica és final de paraula o no del node a trobar/crear.
- **depth**: Profunditat del node a crear/trobar.
- **parent**: Node predecessor del node a crear/trobar.

Retorna: Node amb les característiques introduïdes.

private void updateHashCodeAndPropagate(Node node)

Recalcula el codi hash del node donat i propaga el canvi als seus predecessors.

Paràmetres

- **node**: Node el qual recalcularà el seu codi hash i propagarà.

Relacions:

- Relació d'associació DAWG

Classe WordValidator

Valida certa paraula donada mitjançant el DAWG. El DAWG representa un diccionari de paraules vàl·lides.

Autor: Albert Usero

Cardinalitat: Una instància en tot el programa

Camps

- **dawg**: DAWG el qual fem servir per validar.

Constructores

public WordValidator(DAWG dawg)

Crea un WordValidator

Paràmetres

- **dawg**: Dawg que representa les paraules vàl·lides

Mètodes

public boolean run(String word)

Executa el WordValidator per determinar si una paraula és vàlida o no.

Paràmetres

- **word**: Paraula la qual volem validar

Classe InitialMoveNotInCenterException

Excepció que es llença quan el primer moviment d'un jugador no es col·loca al centre del tauler. Aquesta excepció garanteix que la primera paraula jugada en la partida comenci a la casella central del tauler de Scrabble, que és una regla estàndard del joc.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Constructores

public InitialMoveNotInCenterException(String message)

Crea una nova excepció amb un missatge que explica la causa.

Paràmetres

- **message**: Missatge de detall que descriu el motiu de l'excepció.

Relacions:

- Relació d'associació DAWG

Classe MovementOutsideOfBoardException

Excepció que es llença quan s'intenta fer un moviment fora dels límits del tauler de Scrabble. Aquesta excepció garanteix que els jugadors no puguin col·locar fitxes fora de l'àrea jugable del tauler, evitant així jugades invàlides o errors durant la partida.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Constructores

public MovementOutsideOfBoardException(String message)

Crea una nova excepció amb un missatge que explica la causa.

Paràmetres

- **message**: Missatge detallat que descriu el motiu de l'excepció.

Classe NotEnoughPiecesInBagException

Excepció que es llença quan no hi ha prou fitxes restants a la bossa per completar un moviment. Aquesta excepció assegura que els jugadors no puguin fer un moviment que requereixi més fitxes de les que hi ha disponibles.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Constructores

public NotEnoughPiecesInBagException(String message)

Crea una nova excepció amb un missatge que explica la causa.

Paràmetres

- **message**: Missatge detallat que descriu el motiu de l'excepció.

Classe WordDoesNotExistException

Excepció que es llença quan una paraula no existeix al diccionari durant un moviment.

Aquesta excepció es llença quan un jugador intenta col·locar una paraula al tauler que no és vàlida segons el diccionari de paraules acceptades. Assegura que només es facin servir paraules vàlides en el joc.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Constructores

public WordDoesNotExistException(String message)

Crea una instància de WordDoesNotExistException amb un missatge detallat.

Paràmetres

- **message**: El missatge que explica la causa de l'excepció.

Classe WordNotConnectedToOtherWordsException

Excepció llençada quan una paraula no està connectada a cap altra paraula al tauler.

Aquesta excepció es llença quan un jugador intenta col·locar una paraula al tauler que no està connectada a cap paraula col·locada prèviament, violant la regla que totes les paraules noves han d'estar connectades a paraules ja existents al tauler.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Constructores

public WordNotConnectedToOtherWordsException(String message)

Crea una instància de WordNotConnectedToOtherWordsException amb un missatge detallat.

Paràmetres

- **message**: El missatge que explica la causa de l'excepció.

Classe GameStepper

Classe responsable d'actualitzar el leaderboard quan la partida finalitza. També gestiona l'execució dels torns i mostra la pantalla final.

Autor: Biel Pérez Silvestre

Cardinalitat: Una instància en tot el programa

Camps

- **turn**: Objecte que gestiona el flux dels torns de la partida.
- **leaderboard**: Objecte que representa el leaderboard on s'actualitzen les puntuacions.
- **players**: Array dels jugadors participants a la partida.
- **endScreen**: Objecte encarregat de mostrar la pantalla final de la partida.

Constructores

public GameStepper(Turn turn, Leaderboard leaderboard, Player[] players, IEndScreen endScreen)

Constructor de GameStepper.

Paràmetres

- **turn**: Objecte que gestiona els torns.
- **leaderboard**: Leaderboard on es registren les puntuacions.
- **players**: Jugadors de la partida.
- **endScreen**: Pantalla final a mostrar quan acaba la partida.

Mètodes

public void run(TurnResult result)

Executa el torn actual, actualitza el leaderboard si la partida ha acabat, i mostra la pantalla final amb els jugadors ordenats per puntuació.

Paràmetres

- **result**: Resultat del torn actual.

Relacions

- Relació d'agregació amb Leaderboard
- Relació d'associació amb Player
- Relació d'associació amb IEndScreen

Interfície IEndScreen

Interfície que defineix la pantalla de final de partida. Permet mostrar els resultats finals amb els jugadors ordenats segons la puntuació o un altre criteri.

Autor: Biel Pérez Silvestre

Mètodes

abstract void show(Player[] sortedPlayers)

Mostra la pantalla final amb els jugadors ordenats.

Paràmetres

- **sortedPlayers**: Array de jugadors ordenats segons la seva puntuació

Classe GamesPlayedLeaderboard

Controlador encarregat de generar una classificació de jugadors segons el nombre total de partides que han jugat. Processa un conjunt de resultats Score i agrupa les dades per nom de jugador, retornant una classificació ordenada per nombre de partides jugades (de més a menys).

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa.

Relacions

- Leaderboard: relació de composició.
- Score: relació d'associació.

Mètodes

public PlayerValuePair[] run(Score[] scores)

Executa el procés de classificació de jugadors segons el nombre de partides jugades.

Paràmetres

- **scores**: Array de resultats Score corresponents a partides prèvies.

Retorna: Un array de PlayerValuePair, ordenat descendentment pel nombre de partides jugades. Cada element representa un jugador i la seva quantitat total de partides.

Classe GamesWinsPair

Classe auxiliar utilitzada pels controladors de Leaderboard per representar les estadístiques d'un jugador en termes de partides jugades i partides guanyades. Aquesta estructura permet acumular resultats i calcular el percentatge de victòries.

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància per jugador present a la Leaderboard.

Camps

- **games**: Nombre total de partides jugades.
- **wins**: Nombre total de partides guanyades.

Constructores

public GamesWinsPair(boolean isWinner)

Creador de l'estructura a partir del resultat d'una partida.

Paràmetres

- **isWinner**: Cert si la partida va ser guanyada, fals altrament.

Mètodes

public GamesWinsPair addGame(boolean isWinner)

Afegeix una nova partida a l'estadística, i incrementa el nombre de victòries si escau.

Retorna: El mateix objecte GamesWinsPair, per permetre concatenació de crides.

Paràmetres

- **isWinner**: true si la nova partida va ser guanyada, false altrament.

public double getWinRate()

Calcula el percentatge de victòries respecte al total de partides jugades.

Retorna: Percentatge de victòries (de 0 a 100), com a valor double.

Classe GamesWonLeaderboard

Controlador de Leaderboard encarregat de generar una classificació dels jugadors segons el nombre total de partides guanyades. Processa un conjunt de resultats Score i compta les victòries per jugador, retornant una classificació ordenada descendentment.

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa.

Relacions

- Leaderboard: relació de composició.
- Score: relació d'associació.

Mètodes

public PlayerValuePair[] run(Score[] scores)

Executa el procés de classificació de jugadors segons el nombre de partides guanyades.

Retorna: Un array de PlayerValuePair, ordenat descendentment pel nombre de partides guanyades.

Cada element representa un jugador i la seva quantitat total de victòries.

Paràmetres

- **scores**: Array de resultats Score corresponents a partides prèvies.

Classe MaxScoreLeaderboard

Controlador de Leaderboard encarregat de generar una classificació dels jugadors basada en la puntuació més alta assolida en una partida. A partir d'un conjunt de resultats Score, es calcula la màxima puntuació per jugador i es retorna una classificació ordenada descendentment.

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa.

Relacions

- Leaderboard: relació de composició.
- Score: relació d'associació.

Mètodes

public PlayerValuePair[] run(Score[] scores)

Executa el procés de classificació de jugadors segons la seva puntuació més alta en una sola partida.

Retorna: Un array de PlayerValuePair, ordenat descendentment segons la puntuació màxima obtinguda per cada jugador.

Paràmetres

- **scores**: Array de resultats Score corresponents a partides jugades.

Record PlayerValuePair

Estructura de dades auxiliar utilitzada pels controladors de Leaderboard. Aquesta classe representa una parella formada pel nom d'un jugador i un valor numèric associat (com ara puntuació, partides guanyades o win rate), i és utilitzada per representar els resultats en les classificacions.

Autor: Felipe Martínez Lassalle

Paràmetres

- **playerName**: Nom del jugador.
- **value**: Valor associat al jugador (puntuació, nombre de partides, etc.)

Mètodes

public String playerName()

Obté el nom del jugador.

Retorna: El nom del jugador.

public double value()

Obté el valor associat al jugador.

Retorna: El valor numèric corresponent.

public static PlayerValuePair[] reverse(PlayerValuePair[] arr)

Inverteix l'ordre dels elements d'un array de PlayerValuePair.

Retorna: Un nou array amb els elements en ordre invers.

Paràmetres

- **arr**: Array d'objectes PlayerValuePair a invertir.

Classe TotalScoreLeaderboard

Controlador de Leaderboard encarregat de generar una classificació dels jugadors segons la puntuació total acumulada en totes les partides. Cada jugador és identificat pel seu nom, i la seva puntuació total es calcula sumant totes les puntuacions obtingudes a les diferents partides.

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa.

Relacions

- Leaderboard: relació de composició.
- Score: relació d'associació.

Mètodes

public PlayerValuePair[] run(Score[] scores)

Executa el controlador per generar la classificació de puntuació total.

Retorna: Un array de PlayerValuePair que conté el nom dels jugadors i la seva puntuació total, ordenats de major a menor valor.

Paràmetres

- **scores**: Array amb tots els objectes Score de les partides jugades.

Classe WinRateLeaderboard

Controlador de Leaderboard encarregat de generar una classificació dels jugadors segons el seu percentatge de victòries en les partides jugades. Calcula el ratio de partides guanyades respecte al total de partides jugades per cada jugador.

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa.

Relacions

- Leaderboard: relació de composició.
- Score: relació d'associació.

Mètodes

public PlayerValuePair[] run(Score[] scores)

Executa el controlador per calcular i ordenar el percentatge de victòries dels jugadors.

Retorna: Array de PlayerValuePair amb el nom del jugador i el seu percentatge de victòries,

ordenat de major a menor percentatge.

Paràmetres

- **scores**: Array amb tots els objectes Score que representen les partides jugades.

Classe MovementBoundsChecker

Classe per verificar si un moviment donat es troba dins dels límits d'un tauler. Gestiona tant l'orientació de paraules vertical com horitzontal.

Autor: Gina Escofet González

Cardinalitat: Una instància en tot el programa

Constructores

public MovementBoundsChecker(Board board, PiecesConverter piecesConverter)

Construeix una nova instància de MovementBoundsChecker.

Paràmetres

- **board**: El tauler de joc en el qual es verificaran els límits.
- **piecesConverter**: El convertidor utilitzat per transformar una paraula en un array de peces.

Mètodes

public boolean run(Movement movement)

Verifica si un moviment donat es troba dins dels límits del tauler de joc. i la direcció (vertical/horitzontal).

Paràmetres

- **movement**: El moviment a validar, que conté la paraula, la posició inicial (x, y),

Relacions

- Relació d'associació amb Board
- Relació d'associació amb PiecesConverter

Classe MovementCleaner

Filtra les peces d'una nova col·locació de paraula que se superposen amb les peces existents al tauler. Gestiona tant l'orientació vertical com l'horitzontal de la paraula.

Autor: Gina Escofet González

Cardinalitat: Una instància en tot el programa

Constructores

public MovementCleaner(Board board, PiecesConverter piecesConverter)

Construeix un MovementCleaner amb el board i el piecesConverter.

Llança una IllegalArgumentException si algun dels paràmetres és nul.

Paràmetres

- **board**: L'estat actual del tauler.
- **piecesConverter**: Converteix entre String a Piece[].

Mètodes

public Pair<Piece, Vector2>[] run(Movement movement)

Filtra i retorna només les noves peces necessàries per a la col·locació d'una paraula. la direcció de col·locació (horitzontal/vertical).

Llança una IllegalArgumentException si el moviment és nul.

Paràmetres

- **movement**: El moviment de paraula proposat que conté: la paraula a col·locar, les coordenades d'inici (x,y), i la direcció.

Relacions

- Relació d'associació amb Board
- Relació d'associació amb PiecesConverter

Classe BagFiller

Inicialitza i omple una bossa amb peces de Scrabble específiques de cada idioma.

Autor: Gina Escofet González

Cardinalitat: Una instància en tot el programa

Constructores

public BagFiller(PieceGenerator pieceGenerator)

Construeix un BagFiller amb el generador de peces especificat. Llança una IllegalArgumentException si el generador és nul.

Paràmetres

- **pieceGenerator**: El generador de peces que crea peces específiques de cada idioma.

Mètodes

public Bag run(String piecesTxt)

Omple una nova bossa amb peces segons la configuració de l'idioma especificat.

Llança una IllegalArgumentException si languageConfig és nul. Llança una IllegalStateException si la generació de peces falla.

Paràmetres

- **piecesTxt**: La cadena de configuració que defineix la distribució de les peces.

Relacions

- Relació d'associació amb PieceGenerator

Classe CatalanPiecesConverter

Classe que converteix una cadena de lletres en peces en català. Gestiona casos com L·L i NY.

Autor: Gina Escofet González

Cardinalitat: Una instància en tot el programa

Classe EnglishPiecesConverter

Classe que converteix una cadena de lletres en peces en anglès.

Autor: Gina Escofet González

Cardinalitat: Una instància en tot el programa

Classe HandFiller

Classe responsable d'omplir la mà de cada jugador amb fitxes de la bossa al començament de la partida.

La classe HandFiller assegura que cada jugador rep 7 fitxes a l'inici del joc, extretes de manera aleatòria de la Bag de fitxes disponibles. Utilitza una implementació de IRand per a la selecció aleatòria.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Camps

- **bag**: Bossa de fitxes d'on s'extreuen les fitxes per omplir les mans.
- **players**: Llista dels jugadors als quals cal omplir la mà.
- **random**: Generador de nombres aleatoris per seleccionar fitxes a l'atzar de la bossa.

Constructores

public HandFiller(Bag bag, Player[] players, IRand random)

Crea una nova instància de HandFiller.

Paràmetres

- **bag**: La bossa d'on s'extreuen les fitxes.
- **players**: Els jugadors als quals omplir la mà.
- **random**: El generador de nombres aleatoris usat per seleccionar les fitxes.

Mètodes

public void run()

Omple la mà de cada jugador amb 7 fitxes extretes de la bossa.

Aquest mètode assegura que cada jugador rebi 7 fitxes aleatòries de la bossa. Fa una comprovació que hi hagi prou fitxes a la bossa per a tots els jugadors.

private void fillPlayerHand(Player player)

Omple la mà d'un jugador amb 7 fitxes extretes de la bossa de forma aleatòria.

Paràmetres

- **player**: El jugador a qui s'omplirà la mà.

Relacions

- Relació d'associació amb Bag
- Relació d'associació amb Player

Interfície IFileReader

Interfície que defineix la lectura de fitxers amb dades relacionades amb el joc, com poden ser diccionaris o recursos segons l'idioma.

Aquesta interfície permet implementar diferents maneres de llegir fitxers segons el llenguatge o configuració local.

Autor: Gerard Gascón

Mètodes

abstract String run(Language locale)

Llegeix dades del fitxer associat a l'idioma especificat.

Retorna: El contingut del fitxer com a cadena de text.

Paràmetres

- **locale**: L'idioma o configuració regional per seleccionar el fitxer correcte.

Classe PieceDrawer

Gestiona l'intercanvi de peces entre la mà d'un jugador i la bossa del joc. Retorna les peces no desitjades a la bossa i treu noves peces aleatòries de la bossa.

Autor: Gina Escofet González

Cardinalitat: Una instància per cada jugador

Constructores

public PieceDrawer(Bag bag, Player player, IRand rand)

Crea un PieceDrawer per a la bossa de joc i el jugador especificats.

Paràmetres

- **bag**: La bossa de peces del joc.
- **player**: El jugador que realitza l'intercanvi.
- **rand**: La interfície per a la generació de nombres aleatoris.

Mètodes

public void run(Piece[] piecesToSwap)

Intercanvia peces entre la mà del jugador i la bossa.

Paràmetres

- **piecesToSwap**: Peces a retornar a la bossa.

Relacions

- Relació d'associació amb Bag
- Relació d'associació amb Player

Classe PieceGenerator

Classe responsable de generar objectes Piece a partir del contingut d'un fitxer de fitxes.

La classe PieceGenerator analitza el contingut d'un fitxer de fitxes (una cadena de text) per crear instàncies de Piece. Cada Piece representa una lletra, el seu valor i el nombre d'aparicions d'aquesta lletra. El resultat es retorna com un array de Pair<Piece, Integer>, on Piece representa la lletra i les seves propietats, i l'enter representa el nombre d'unitats d'aquesta fitxa.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Mètodes

public Pair<Piece, Integer>[] run(String pieces)

Converteix el contingut d'un fitxer de fitxes en un array d'objectes Piece i les seves quantitats.

El fitxer es processa línia per línia, on cada línia ha de contenir la informació sobre el caràcter, la quantitat i el valor de la fitxa. El resultat és un array de Pair<Piece, Integer>, on cada parella conté un objecte Piece i el nombre d'aparicions d'aquesta fitxa.

Retorna: Un array de parelles on cada parella conté una Piece i la seva quantitat.

Paràmetres

- **pieces**: El contingut del fitxer de fitxes com una cadena de text.

private Pair<Piece, Integer> generatePiece(String piece)

Genera una parella de Piece i el seu nombre d'aparicions a partir d'una línia de text.

Retorna: Una parella amb la Piece creada i la seva quantitat.

Paràmetres

- **piece**: La línia de text amb la informació de la fitxa.

private int parseCount(String piece)

Extreu la quantitat d'unitats d'una fitxa a partir de la línia.

Retorna: El nombre d'unitats d'aquesta fitxa.

Paràmetres

- **piece**: La línia de text amb la informació de la fitxa.

private int parseValue(String piece)

Extreu el valor de la fitxa a partir de la línia.

Retorna: El valor de la fitxa.

Paràmetres

- **piece**: La línia de text amb la informació de la fitxa.

private String parseCharacter(String piece)

Extreu el caràcter que representa la fitxa a partir de la línia.

Retorna: El caràcter de la fitxa.

Paràmetres

- **piece**: La línia de text amb la informació de la fitxa.

Classe PiecesConverter

Classe que converteix una cadena de lletres en peces.

Autor: Gina Escofet González

Constructores

public PiecesConverter()

Constructor per defecte de PiecesConverter.

public PiecesConverter(Piece[] pieces)

Construeix un PiecesConverter amb un conjunt inicial de peces.

Paràmetres

- **pieces**: Un array de peces per inicialitzar el convertidor.

Classe PiecesConverterFactory

Fàbrica encarregada de crear un convertidor de fitxes segons l'idioma seleccionat.

Aquesta classe utilitza un lector de fitxers per obtenir el contingut de les fitxes, genera les fitxes amb PieceGenerator i crea el convertidor adequat en funció de l'idioma.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Constructores

public PiecesConverterFactory(IFileReader piecesReader)

Constructor que rep un lector de fitxers de fitxes.

Paràmetres

- **piecesReader**: Lector encarregat de llegir el fitxer de fitxes segons l'idioma.

Mètodes

public PiecesConverter run(Language language)

Genera un convertidor de fitxes per l'idioma especificat.

Llegeix el fitxer de fitxes corresponent, genera les fitxes i retorna l'objecte convertidor específic per a l'idioma (Català, Castellà o Anglès).

Retorna: Un objecte PiecesConverter per l'idioma indicat.

Paràmetres

- **language**: L'idioma pel qual es vol generar el convertidor de fitxes.

Relacions

- Relació d'agregació amb IFileReader

Classe PiecesInHandGetter

Gestiona la recuperació i verificació de peces de la mà d'un jugador.

Autor: Gina Escofet González

Cardinalitat: Una instància per cada jugador

Constructores

public PiecesInHandGetter(Bag bag, Player player, IRand rand)

Construeix un nou PiecesInHandGetter amb la bossa, el jugador i el "piecePrinter" especificats.

Paràmetres

- **bag**: La bossa de peces del joc.
- **player**: El jugador objectiu.
- **rand**: La interfície per a la generació de nombres aleatoris.

Mètodes

public Piece[] run(Piece[] pieces)

Verifica i intercanvia peces de la mà del jugador.

Llança una IllegalStateException si el procés falla.

Paràmetres

- **pieces**: Peces necessàries per a la jugada actual.

Relacions

- Relació d'associació amb Bag
- Relació d'associació amb Player

Classe PiecesInHandVerifier

Verifica si un jugador té les peces necessàries a la seva mà per formar una paraula donada.

Autor: Gina Escofet González

Cardinalitat: Una instància per cada jugador

Constructores

public PiecesInHandVerifier(Player player, PiecesConverter piecesConverter)

Construeix un verificador per al jugador i el convertidor de peces especificats.

Llança una IllegalArgumentException si algun dels paràmetres és nul.

Paràmetres

- **player**: El jugador la mà del qual serà verificada.
- **piecesConverter**: Converteix paraules a arrays de peces.

Mètodes

public Piece[] run(String word)

Troba i retorna les peces de la mà del jugador que coincideixen amb una paraula donada.

Llança una IllegalArgumentException si la paraula és nul·la.

Paràmetres

- **word**: La paraula a verificar amb les peces del jugador.

Relacions

- Relació d'associació amb Player
- Relació d'associació amb PiecesConverter

Classe SpanishPiecesConverter

Classe que converteix una cadena de lletres en peces en castellà. Gestiona casos com RR, LL, CH.

Autor: Gina Escofet González

Cardinalitat: Una instància en tot el programa

Classe Endgame

Classe encarregada de determinar si la partida ha finalitzat. La partida acaba si tots els jugadors passen el torn diverses vegades consecutives o si algun jugador es queda sense fitxes.

Autor: Biel Pérez

Cardinalitat: Una instància en tot el programa

Camps

- **players**: Array de jugadors, utilitzat per consultar els seus atributs.

Constructores

public Endgame(Player[] players)

Constructor de la classe Endgame.

Paràmetres

Retorna: true si la partida ha acabat, false en cas contrari.

- **players**: Jugadors que participen a la partida.

Mètodes

public boolean run(int skipCounter)

Determina si la partida ha acabat. cas en què la partida finalitza.

Paràmetres

- **skipCounter**: Comptador per determinar si tots els jugadors han passat el torn tres vegades consecutives,

Relacions

- Relació d'associació amb Player

Interfície IGamePlayer

Interfície que representa l'objecte jugador des de la capa de presentació. Aquesta interfície defineix els mètodes per gestionar el torn d'un jugador dins del joc.

Autor: Gerard Gascón

Mètodes

abstract void startTurn()

Inicia el torn del jugador. Aquest mètode es crida quan és el torn del jugador per realitzar una acció (per exemple, col·locar una paraula al tauler). A partir d'aquí, el jugador pot dur a terme les seves accions.

abstract void endTurn()

Finalitza el torn del jugador. Aquest mètode es crida quan el jugador ha acabat les seves accions durant el torn (per exemple, ha col·locat una paraula). Retorna el resultat del torn, que pot incloure detalls com els punts obtinguts o la validesa del moviment.

abstract boolean isActive()

Comprova si el jugador està actualment jugant el seu torn. Aquest mètode s'utilitza per saber si el jugador està actiu en el torn o ja l'ha finalitzat.

Retorna: true si el jugador està actiu jugant el seu torn, false si ha acabat o encara no ha començat.

Classe Turn

Classe encarregada de gestionar el flux de torns en una partida. S'encarrega de controlar quin jugador finalitza i quin comença el torn, a més de portar el recompte de torns i comprovar si la partida ha acabat.

Autor: Biel Pérez

Cardinalitat: Una instància en tot el programa

Camps

- **endgame**: Atribut que s'encarrega d'avaluar si la partida ha acabat.

Constructores

public Turn(Endgame endgame, IGamePlayer[] players)

Constructora de la classe Turn.

Paràmetres

- **endgame**: Lògica per determinar si la partida ha finalitzat.
- **players**: Jugadors participants a la partida.

Mètodes

public boolean run(TurnResult result)

Gestiona el canvi de torns entre jugadors.

Finalitza el torn del jugador actual.

Inicia el torn del següent jugador.

Controla el nombre de torns consecutius que s'han passat.

Determina si la partida ha de finalitzar segons la lògica de l'Endgame.

Retorna: true si la partida ha acabat, false altrament.

Paràmetres

- **result**: Resultat del torn (ex: Skip si el jugador ha passat).

Relacions

- Relació d'associació amb EndScreen

Enumeració TurnResult

Enumeració que representa els possibles resultats d'un torn en el joc.

Autor: Gerard Gascón

Camps

- **Skip**: El jugador decideix passar el torn sense fer cap acció.
- **Place**: El jugador col·loca una o més fitxes al tauler.
- **Draw**: El jugador roba fitxes del sac (bag).

2.4. Capa de presentació

Classe DictionaryReader

Classe que s'encarrega de llegir fitxers de diccionaris per diferents idiomes.

La classe DictionaryReader és responsable de llegir els fitxers de diccionaris segons l'idioma (locale) seleccionat. Suporta la lectura de diccionaris en català, castellà i anglès.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Mètodes

public String run (Language locale)

Llegeix el fitxer de diccionari segons l'idioma especificat.

Aquest mètode comprova l'idioma donat i retorna el contingut del fitxer de diccionari corresponent com una cadena de text. Els idiomes suportats són català, castellà i anglès.

Retorna: Una cadena amb el contingut del diccionari en l'idioma seleccionat.

Paràmetres

- **locale**: L'idioma del diccionari que es vol llegir.

Classe LocaleReader

Classe abstracta que s'encarrega de localitzar i llegir fitxers de localització.

La classe LocaleReader proporciona funcionalitats per trobar i llegir fitxers de localització (per exemple, fitxers de diccionari). El mètode readFileToString llegeix el contingut d'un fitxer especificat del sistema de fitxers local, utilitzant una codificació concreta. Aquesta classe s'usa habitualment com a base per a altres classes que necessiten carregar recursos localitzats.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Mètodes

private File getAbsolutePath(String fileName)

Obtén la ruta absoluta al fitxer especificat dins del directori de localització.

Aquest mètode construeix el camí absolut a partir de la ubicació del programa i resol la ruta fins al directori edu/upc/prop/scrabble/locales, afegint el nom del fitxer.

Retorna: Un objecte File amb la ruta absoluta al fitxer.

Llança una RuntimeException Si no es pot resoldre l'URI de la ruta.

Paràmetres

- **fileName**: Nom del fitxer de localització.

protected String readFileToString(String filePath)

Llegeix el contingut d'un fitxer i el retorna com una cadena de text.

Aquest mètode llegeix el fitxer indicat, utilitzant la codificació UTF-8, i retorna tot el seu contingut com una cadena String. En cas d'error, retorna null i imprimeix un missatge d'error a la sortida d'errors.

Retorna: Contingut del fitxer en format cadena, o null si hi ha un error.

Paràmetres

- **filePath**: Nom del fitxer a llegir.

Classe PiecesReader

Classe per llegir fitxers de peces per diferents idiomes.

La classe PiecesReader estén LocaleReader i implementa la interfície IFileReader. S'encarrega de llegir els fitxers de peces (per exemple, lletres o fitxes) segons l'idioma proporcionat. Proporciona un mètode per obtenir el contingut del fitxer de peces corresponent a l'idioma seleccionat i retorna el seu contingut en forma de cadena.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Mètodes

public String run(Language locale)

Llegeix el fitxer de peces corresponent a l'idioma especificat.

Aquest mètode comprova l'idioma passat com a paràmetre i retorna el contingut del fitxer de peces associat a aquest idioma.

Retorna: El contingut del fitxer de peces seleccionat, en format cadena.

Paràmetres

- **locale**: L'idioma del fitxer de peces que es vol llegir.

Classe SceneObjectWithParametrizedConstructorException

Excepció llançada quan un SceneObject no pot tenir un constructor amb paràmetres.

Aquesta excepció s'utilitza quan es tracta de crear una instància de SceneObject amb un constructor que accepta paràmetres, fet que no està permès pel disseny o pel framework.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Constructores

public SceneObjectWithParametrizedConstructorException(String message)

Constructor de l'excepció amb un missatge personalitzat.

Paràmetres

- **message**: Missatge explicant la causa de l'excepció.

Classe GameLoop

Classe que gestiona el bucle principal del joc.

El GameLoop s'encarrega de controlar l'execució contínua del joc, actualitzant l'estat de l'escena activa a un ritme constant (FPS).

Autor: Gerard Gascón

Constructores

public GameLoop(Class<? extends Scene> initialScene, Object… dependencies)

Constructor que inicialitza el bucle carregant l'escena inicial.

Paràmetres

- **initialScene**: Classe de l'escena inicial que es vol carregar.
- **dependencies**: Dependències que pugui tenir l'escena inicial.

Mètodes

public void run()

Executa el bucle principal del joc, mantenint una actualització contínua de l'escena amb una velocitat fixa de fotogrames per segon (FPS).

El mètode calcula el temps transcorregut entre iteracions per passar-lo a l'escena, i dorm el fil per mantenir la taxa constant de fotogrames.

Relacions

- **Scene:** relació d'agregació

Classe Scene

Classe base per representar una escena dins del joc.

Aquesta classe proporciona la funcionalitat per gestionar i interactuar amb els objectes dins d'una escena, incloent-hi el processament d'actualitzacions, la creació d'objectes i la gestió de la destrucció d'objectes.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Camps

- **objects**: Llista d'objectes que pertanyen a aquesta escena.

Mètodes

void onProcess(float delta)

Actualitza contínuament els objectes presents a l'escena.

Aquest mètode s'anomena repetidament per processar les actualitzacions dels objectes.

Paràmetres

- **delta**: Temps transcorregut des de l'última actualització (delta time)

void onDetach()

Desenganxa (desconnecta) tots els objectes de l'escena actual.

Aquest mètode desactiva i desenganxa tots els objectes, eliminant-los efectivament de l'escena.

public T instantiate(Class o)

Instancia un nou objecte dins de l'escena.

Aquest mètode intenta crear una nova instància de la classe especificada i afegir-la a l'escena. Si la classe té un constructor amb paràmetres, llença una excepció, ja que no es donen suport constructors parametritzats.

Retorna: L'objecte instanciat

Llança una SceneObjectWithParametrizedConstructorException Si l'objecte té un constructor amb paràmetre. Llança una RuntimeException Si hi ha algun error durant la creació de l'objecte

Paràmetres

- **o**: La classe de l'objecte a instanciar

public void destroy(SceneObject o)

Destrueix un objecte de l'escena.

Aquest mètode elimina l'objecte especificat de l'escena, el desactiva i el desenganxa.

Llança una NullPointerException Si l'objecte a destruir és null

Paràmetres

- **o**: Instància de l'objecte a destruir

Relacions

- Relació d'agregació amb SceneObject

Classe SceneManager

Classe utilitzada per gestionar totes les escenes presents al joc i per canviar entre elles.

El SceneManager és responsable de carregar i gestionar diferents escenes durant el joc. Gestiona les transicions entre escenes, la destrucció de les escenes en fer el canvi i assegura que es passen les dependències correctes quan es carrega una nova escena.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Camps

- sceneLoaded: Indica si ja hi ha una escena carregada.
- scene: Escena actual activa.
- sceneToLoad: Escena pendent de carregar (classe i dependències).
- instance: Instància singleton de SceneManager.

Mètode

public static SceneManager getInstance()

Obté o crea la instància única de SceneManager.

Aquest mètode assegura que només existeixi una instància de SceneManager alhora.

Retorna: La instàsncia de SceneManager

public void load(Class<? extends Scene> sceneClass, Object… dependencies)

Carrega una nova escena. Si n'hi ha una activa, la destrueix abans de carregar la nova.

Aquest mètode s'encarrega de fer la transició entre escenes. Si ja hi ha una escena activa, la desenganxa i carrega la nova. Si no n'hi ha, simplement carrega la nova.

Paràmetres

- **sceneClass**: La classe de la nova escena a carregar
- **dependencies**: Les dependències que necessita la nova escena

private void loadScene(Class<? extends Scene> sceneClass, Object… dependencies)

Carrega la nova escena proporcionada, desenganxant l'escena actual si n'hi ha.

Paràmetres

- **sceneClass**: La classe de la nova escena
- **dependencies**: Les dependències per passar al constructor de l'escena

private Scene instantiateScene(Class<? extends Scene> sceneClass, Object… dependencies) throws InstantiationException, IllegalAccessException, InvocationTargetException

Instància l'escena amb el constructor que millor s'adapti a les dependències.

Retorna: La instància de l'escena creada

Paràmetres

- **sceneClass**: Classe de l'escena a instanciar
- **dependencies**: Paràmetres per passar al constructor

private static Constructor<?> getSceneConstructor(Class<? extends Scene> sceneClass)

Obté el constructor amb més paràmetres de la classe d'escena, que serà el que s'utilitzarà per instanciar.

Retorna: El constructor amb més paràmetres

Paràmetres

- **sceneClass**: La classe de l'escena

public void quit()

Tanca el joc i destrueix l'escena activa.

Aquest mètode s'encarrega de netejar recursos i tancar el joc, desenganxant l'escena activa abans de tancar.

public void process(float delta)

Actualitza l'escena activa i els seus objectes.

Aquest mètode s'executa contínuament per actualitzar l'escena actual. Si hi ha una escena pendent de carregar, la carrega. En cas contrari, processa l'escena activa.

Paràmetres

- **delta**: Temps transcorregut des de l'última crida

public boolean isRunning()

Comprova si hi ha una escena activa en execució.

Aquest mètode permet al joc o aplicació saber si hi ha una escena activa amb la qual interactuar.

Retorna: true si hi ha una escena activa, false si no n'hi ha cap

public Scene getActiveScene()

Obté l'escena activa actual.

Retorna: L'escena activa

Relacions

- Relació d'agregació amb Scene.
- Relació d'agregació amb SceneManager.

Classe SceneObject

Classe base abstracta que representa un objecte dins d'una escena.

Aquesta classe proporciona funcionalitats bàsiques per habilitar, deshabilitar i processar objectes dins l'escena. També gestiona la integració amb el sistema de gestió d'escenes, cosa que permet la instanciació i destrucció d'objectes segons sigui necessari.

Autor: Gerard Gascón

Cardinalitat: Una instància per cada vista dins del joc, sigui jugador, tauler o peces de la mà.

Camps

- enabled: Indica si l'objecte està habilitat (actiu) o no.
- scene: Escena a la qual pertany aquest objecte.

Mètodes

void setScene(Scene scene)

Assigna l'escena a la qual pertany aquest objecte.

Paràmetres

- scene: L'escena a associar amb aquest objecte

public final void disable()

Deshabilita l'objecte i executa la lògica associada a la deshabilitació. Aquest mètode es crida quan l'objecte és deshabilitat des de l'escena.

public final void enable()

Habilita l'objecte i executa la lògica associada a l'habilitació. Aquest mètode es crida quan l'objecte és habilitat dins l'escena.

public final boolean isEnabled()

Comprova si l'objecte està habilitat.

Retorna: true si l'objecte està habilitat, false en cas contrari

public void onProcess(float delta)

Mètode que es crida per processar l'objecte durant l'actualització de l'escena. Pot ser sobreescrit per les subclasses per implementar comportaments específics.

Paràmetres

- delta: Diferència de temps respecte a la darrera crida (en segons)

public void onDetach()

Mètode cridat quan l'objecte és desenganxat de l'escena. Les subclasses poden sobreescriure aquest mètode per gestionar el procés de desenganxament.

protected void onEnable()

Mètode cridat quan l'objecte és habilitat. Pot ser sobreescrit per implementar comportaments personalitzats quan l'objecte s'habilita dins l'escena.

protected void onDisable()

Mètode cridat quan l'objecte és deshabilitat. Pot ser sobreescrit per implementar comportaments personalitzats quan l'objecte es deshabilita dins l'escena.

protected T instantiate(Class o)

Instància un nou objecte d'escena dins de l'escena actual.

Retorna: L'objecte d'escena acabat d'instanciar

Paràmetres

- **o**: La classe de l'objecte d'escena a instanciar

protected final void destroy(SceneObject o)

Destrueix un objecte d'escena i l'elimina de l'escena actual.

Paràmetres

- **o**: L'objecte d'escena a destruir

Relacions

- Relació d'associació amb Scene

Classe Main

Punt d'entrada principal de l'aplicació Scrabble amb interfície Swing. Crea la finestra principal i mostra la pantalla de joc. Gestiona la configuració de la finestra segons el sistema operatiu.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Mètodes

public static void main(String[] args)

Mètode principal que inicia l'aplicació. Executa la creació de la finestra en el fil d'esdeveniments de Swing.

Paràmetres

- **args**: arguments de la línia de comandes (no utilitzats)

private static void disableTraversalKeys()

Deshabilita les tecles de navegació per defecte (TAB, SHIFT+TAB) per evitar que canviïn el focus.

private static JFrame createMainWindow()

Crea la finestra principal adaptada segons el sistema operatiu.

Retorna: JFrame configurat com a finestra principal

private static JFrame createMainWindow_Windows()

Crea la finestra principal per a sistemes Windows. La finestra és sense decoració, no redimensionable i a pantalla completa.

Retorna: JFrame configurat per a Windows

private static JFrame createMainWindow_Linux()

Crea la finestra principal per a sistemes Linux. Intenta posar la finestra en mode pantalla completa si és compatible. En cas contrari, estableix la mida completa de la pantalla.

Retorna: JFrame configurat per a Linux

Classe AIPlayerObject

Representa un objecte jugador controlat per intel·ligència artificial (IA). Estén la classe PlayerObject afegint funcionalitat específica per a jugadors IA.

Autor: Gerard Gascón

Cardinalitat: Una instància per cada jugador controlat per IA

Mètodes

public void configureAI(AI ai)

Configura l'objecte jugador amb una instància d'IA per controlar-lo.

Paràmetres

- **ai**: Instància de la intel·ligència artificial que gestionarà el jugador.

Classe HumanPlayerObject

Representa un objecte jugador controlat per un usuari humà. Estén la classe PlayerObject sense funcionalitat addicional específica.

Autor: Gerard Gascón

Cardinalitat: Una instància per cada jugador real

Classe PlayerObject

Classe abstracta que representa un objecte jugador dins de l'escena, integrant les accions de joc i la interfície amb la vista de la mà. Implementa la interfície IGamePlayer per a la gestió de torns.

Autor: Gerard Gascón

Cardinalitat: Una instància per cada jugador en la partida

Camps

- **playerIndex**: Índex que identifica el jugador dins de la partida.
- **onTurn**: Indica si el jugador té el torn actiu.
- **player**: Dades del jugador (nom, mà, etc.).
- **placeActionMaker**: Acció encarregada de col·locar fitxes al tauler.
- **drawActionMaker**: Acció encarregada de robar fitxes de la bossa.
- **skipActionMaker**: Acció encarregada de saltar el torn.
- **handView**: Vista que mostra les fitxes a la mà del jugador.

Mètodes

public final void configure(int playerIndex, PlaceActionMaker placeActionMaker, Player player, DrawActionMaker drawActionMaker, SkipActionMaker skipActionMaker, IHandView handView)

Configura el jugador amb les accions i dades necessàries.

Paràmetres

- **playerIndex**: Índex del jugador.
- **placeActionMaker**: Acció per col·locar fitxes.
- **player**: Dades del jugador.
- **drawActionMaker**: Acció per robar fitxes.
- **skipActionMaker**: Acció per saltar torn.
- **handView**: Vista de la mà del jugador.

public void startTurn()

Inicia el torn del jugador mostrant les fitxes a la vista.

protected final void placePiece(Movement movement)

Intenta col·locar una fitxa al tauler segons el moviment indicat. En cas d'error, salta automàticament el torn.

Paràmetres

- **movement**: Moviment que indica on col·locar la fitxa.

protected final void drawPieces(Piece[] piece)

Intenta robar les fitxes indicades. En cas d'error, salta automàticament el torn.

Paràmetres

- **piece**: Array de fitxes que es volen robar.

protected final void skipTurn()

Salta el torn actual.

public final void endTurn()

Finalitza el torn del jugador.

public final boolean isActive()

Indica si el jugador està actiu en el torn actual.

Retorna: true si el jugador té el torn, false en cas contrari.

Relacions

- Relació d'associació amb Player
- Relació d'agregació amb PlaceActionMaker
- Relació d'agregació amb DrawActionMaker
- Relació d'agregació amb SkipActionMaker
- Relació d'associació amb IHandView

Classe GameScene

Classe que representa l'escena principal del joc de Scrabble.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Camps

- **SAVE_FILE_NAME**: Nom del fitxer on es desa i es carrega la partida guardada.

Constructores

public GameScene(GameProperties properties, JFrame window)

Constructor que crea la escena del joc.

Paràmetres

- **properties**: Propietats que configuren el joc.
- **window**: Finestra principal del joc.

Mètodes

private void createNewGame(GameProperties properties, JFrame window, GameSaver saver, DataCollector dataCollector)

Crea una nova partida amb les propietats indicades.

Paràmetres

- **properties**: Propietats del joc.
- **window**: Finestra on es mostrarà la partida.
- **saver**: Objecte encarregat de guardar la partida.
- **dataCollector**: Objecte encarregat de recopilar dades persistents.

private void loadGame(JFrame window, DataRestorer dataRestorer, GameLoader loader, GameSaver saver, DataCollector dataCollector)

Carrega una partida desada prèviament.

Paràmetres

- **window**: Finestra on es mostrarà la partida.
- **dataRestorer**: Objecte encarregat de restaurar les dades.
- **loader**: Objecte encarregat de carregar la partida.
- **saver**: Objecte encarregat de guardar la partida.
- **dataCollector**: Objecte encarregat de recopilar dades persistents.

private void resolveDependencies(JFrame window, GameSaver saver, DataCollector dataCollector, Board board, Language language, Player[] playersData, int turnNumber, int skipCounter)

Resol i configura totes les dependències necessàries per al funcionament del joc.

Paràmetres

- **window**: Finestra principal.
- **saver**: Objecte encarregat de guardar la partida.
- **dataCollector**: Objecte encarregat de recopilar dades persistents.
- **board**: Tauler del joc.
- **language**: Idioma del joc.
- **playersData**: Dades dels jugadors.
- **turnNumber**: Número del torn actual.
- **skipCounter**: Comptador de salts de torn.

private static void generateWindow(JFrame window, BoardView boardView, SidePanel sidePanel, HandView handView, BlankPieceSelector blankPieceSelector, PauseMenu pauseMenu)

Genera i afegeix les diferents vistes i panells a la finestra del joc.

Paràmetres

- **window**: Finestra principal.
- **boardView**: Vista del tauler.
- **sidePanel**: Panell lateral amb informació dels jugadors.
- **handView**: Vista de la mà del jugador.
- **blankPieceSelector**: Selector de fitxes en blanc.
- **pauseMenu**: Menú de pausa.

private static void fillDAWG(DAWG dawg, Language language)

Omple l'estructura DAWG amb paraules segons l'idioma del joc.

Paràmetres

- **dawg**: Estructura DAWG on es guarden les paraules.
- **language**: Idioma del joc.

private static Bag generateBag(Language language)

Genera una bossa de fitxes segons l'idioma del joc.

Retorna: Bossa amb les fitxes configurades.

Paràmetres

- **language**: Idioma del joc.

private Player[] createPlayersData(GameProperties properties)

Crea un array de jugadors (dades bàsiques) a partir de les propietats del joc.

Retorna: Array de jugadors.

Paràmetres

- **properties**: Propietats del joc.

private PlayerObject[] instantiatePlayers(Player[] players, Language language, PiecesConverter piecesConverter, PointCalculator pointCalculator, DAWG dawg, Board board, Anchors anchors, CrossChecks crossChecks)

Instancia els objectes de jugadors, diferenciant entre jugadors humans i IA, segons l'idioma.

Retorna: Array d'objectes PlayerObject instanciats.

Paràmetres

- **players**: Array de jugadors amb dades bàsiques.
- **language**: Idioma del joc.
- **piecesConverter**: Convertidor de fitxes segons l'idioma.
- **pointCalculator**: Calculadora de punts.
- **dawg**: Estructura DAWG amb paraules.
- **board**: Tauler del joc.
- **anchors**: Ancoratges per a jugades.
- **crossChecks**: Objecte de validació de paraules creuades.

private void configurePlayers(PlayerObject[] playerObjects, Player[] players, GameStepper stepper, Board board, BoardView boardView, PointCalculator pointCalculator, Bag bag, PiecesConverter piecesConverter, AnchorUpdater anchorUpdater, DAWG dawg, CrossChecks crossChecks, IHandView handView)

Configura les propietats i dependències de cada jugador per al desenvolupament correcte del joc.

Paràmetres

- **playerObjects**: Objectes PlayerObject.
- **players**: Dades bàsiques dels jugadors.
- **stepper**: Gestor de torns.
- **board**: Tauler del joc.
- **boardView**: Vista del tauler.
- **pointCalculator**: Calculadora de punts.
- **bag**: Bossa de fitxes.
- **piecesConverter**: Convertidor de fitxes.
- **anchorUpdater**: Actualitzador d'ancoratges.
- **dawg**: Estructura DAWG.
- **crossChecks**: Validació de paraules creuades.
- **handView**: Vista de la mà del jugador.

private void configurePlayer(int index, PlayerObject playerObject, Player player, GameStepper stepper, Board board, BoardView boardView, PointCalculator pointCalculator, Bag bag, PiecesConverter piecesConverter, AnchorUpdater anchorUpdater, DAWG dawg, CrossChecks crossChecks, IHandView handView)

Configura un jugador amb totes les seves dependències i accions.

Paràmetres

- **index**: Índex del jugador a configurar.
- **playerObject**: Objecte que representa el jugador (humà o IA).
- **player**: Dades bàsiques del jugador.
- **stepper**: Gestor del torn de joc.
- **board**: Tauler del joc.
- **boardView**: Vista del tauler.
- **pointCalculator**: Calculadora de punts per a la partida.
- **bag**: Bossa de fitxes disponible.
- **piecesConverter**: Convertidor de fitxes segons idioma.
- **anchorUpdater**: Actualitzador d'ancoratges per a jugades.
- **dawg**: Estructura DAWG que conté el diccionari.
- **crossChecks**: Objecte per a la validació de paraules creuades.
- **handView**: Vista de la mà del jugador.

private Board getBoard(GameProperties properties)

Retorna el tauler segons la configuració especificada a les propietats.

Retorna: Tauler configurat.

Paràmetres

- **properties**: Propietats del joc que defineixen el tipus de tauler.

Classe MenuScene

Escena encarregada del menú principal

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa.

Relacions

- Relació de composició amb Scene.

Constructores

public MenuScene(JFrame window)

Creadora i inicialitzadora de tots els elements presents al menú

Paràmetres

- **window**: Finestra sobre la qual es mostrarà el menú

Mètodes

private void handlePlay()

Acció cridada al pulsar el botó Jugar

private void handleContinue()

Acció cridada al pulsar el botó Continuar

private void handleRanking()

Acció cridada al pulsar el botó Rànquing

Classe BoardView

Vista del tauler de Scrabble.

Gestiona la graella de fitxes, les peces temporals i les fitxes prèmium.

Autor: Gerard Gascón

Cardinalitat: Una instància en tot el programa

Camps

- **size**: Mida (nombre de caselles per fila i columna).
- **handView**: Vista de la mà per gestionar la selecció de peces.
- **blankPieceSelector**: Selector per fitxes en blanc.
- **temporalPieces**: Llista de peces temporals col·locades al tauler però no confirmades.

Constructores

public BoardView(int size, IHandView handView, IBlankPieceSelector blankPieceSelector)

Constructor del tauler.

Paràmetres

- **size**: mida del tauler
- **handView**: vista de la mà
- **blankPieceSelector**: selector per fitxes en blanc

Mètodes

private void generateEmptyTiles(int size, IHandView handView)

Genera les caselles buides i les caselles de coordenades del tauler. Les caselles de coordenades mostren la fila i columna amb lletres o números.

Paràmetres

- **size**: mida del tauler
- **handView**: vista de la mà

protected void paintComponent(Graphics g)

Personalitza el pintat del tauler.

Paràmetres

- **g**: context gràfic

private String getBorderRow(int row)

Retorna la cadena que representa la fila per la coordenada.

Retorna: cadena que representa la fila (o buit si és bord)

Paràmetres

- **row**: número de fila

private String getBorderCol(int col)

Retorna la cadena que representa la columna per la coordenada.

Retorna: cadena amb lletra (o buit si és bord)

Paràmetres

- **col**: número de columna

public void changeTile(BoardTile tile, int x, int y)

Canvia la fitxa en la posició (x,y) per la nova BoardTile proporcionada.

Paràmetres

- **tile**: nova fitxa
- **x**: coordenada x
- **y**: coordenada y

public BoardCell getCell(int x, int y)

Retorna la cel·la corresponent a la posició (x,y).

Retorna: BoardCell a la posició indicada

Paràmetres

- **x**: coordenada x
- **y**: coordenada y

public TComponent getComponentOfType(Class clazz, int index)

Retorna el component de tipus clazz en la posició index, filtrant només components de tipus clazz.

Retorna: component sol·licitat

Llança una RuntimeException si no es troba cap component del tipus i índex indicats

Paràmetres

- **clazz**: classe del component
- **index**: índex dins dels components d'aquest tipus

public void placeTemporalPiece(String piece, int points, int x, int y)

Col·loca una peça temporal al tauler a la posició (x,y), si la cel·la està disponible i la posició és vàlida segons les regles.

Paràmetres

- **piece**: lletra de la peça
- **points**: punts de la peça
- **x**: coordenada x
- **y**: coordenada y

public String getTemporalWord()

Retorna la paraula temporal formada per les peces col·locades temporalment.

Retorna: paraula formada

Llança una RuntimeException si el cas no està suportat

private String getHorizontalTemporalWord()

Retorna la paraula formada horitzontalment per les peces temporals i fixes.

Retorna: paraula horitzontal

private String getVerticalTemporalWord()

Retorna la paraula formada verticalment per les peces temporals i fixes.

Retorna: paraula vertical

private BoardTile[] getHorizontalTiles()

Retorna les fitxes horitzontals adjacents a la peça temporal.

Retorna: array de fitxes horitzontals

private BoardTile[] getVerticalTiles()

Retorna les fitxes verticals adjacents a la peça temporal.

Retorna: array de fitxes verticals

private List getSortedTiles(boolean horizontal)

Retorna la llista de fitxes ordenades horitzontalment o verticalment.

Retorna: llista ordenada de fitxes

Paràmetres

- **horizontal**: true si és horitzontal, false si és vertical

private boolean isPositionValid(int x, int y)

Comprova si la posició (x,y) és vàlida per col·locar-hi una peça temporal. Si no hi ha peces temporals, qualsevol posició és vàlida; en cas contrari, ha de ser adjacent i alineada amb les peces ja col·locades temporalment.

Retorna: true si la posició és vàlida

Paràmetres

- **x**: coordenada x
- **y**: coordenada y

private boolean isAdjacentToOtherPlacedWords(int x, int y)

Comprova si la posició (x,y) és adjacent a altres peces ja col·locades temporalment.

Retorna: true si és adjacent

Paràmetres

- **x**: coordenada x
- **y**: coordenada y

private boolean isInLineWithAlreadyPlacedWords(int x, int y)

Comprova si la posició (x,y) està alineada amb les peces ja col·locades temporalment.

Retorna: true si està alineada

Paràmetres

- **x**: coordenada x
- **y**: coordenada y

private static boolean isCellAvailable(BoardCell cell)

Comprova si una cel·la està disponible per col·locar-hi una peça (no ocupada per peces ja col·locades).

Retorna: true si està disponible

Paràmetres

- **cell**: cel·la a comprovar

public void updateBoard()

Actualitza el tauler (mètode pendent d'implementar).

public void updateCell(String piece, int points, int x, int y)

Actualitza la cel·la (x,y) amb una peça definitiva.

Paràmetres

- **piece**: lletra de la peça
- **points**: punts de la peça
- **x**: coordenada x
- **y**: coordenada y

public void setPremiumTile(PremiumTileType type, int x, int y)

Assigna una fitxa prèmium a la posició (x,y). La fitxa central té un tractament especial.

Paràmetres

- **type**: tipus de fitxa prèmium
- **x**: coordenada x
- **y**: coordenada y

Relacions

- relació d'associació amb handView
- relació d'associació amb blankPieceSelector
- relació d'agregació amb BoardTemporalPieceTile

Interfície IBlankPieceSelector

Interfície per gestionar la selecció d'una fitxa en blanc. Quan s'obre el selector, es passa un callback que rebrà la fitxa seleccionada.

Autor: Gerard Gascón

Mètodes

abstract void openSelectorPopUp(Consumer selectPieceCallback)

Obre el popup per seleccionar la fitxa en blanc. Un cop seleccionada la fitxa, es crida el callback amb la lletra escollida.

Paràmetres

- **selectPieceCallback**: funció que rep la fitxa seleccionada (ex: "A", "E", etc.)

Classe PlayerHighlight

Component gràfic que s'encarrega de resaltar visualment el jugador actual.

Aquesta classe dibuixa un marc rodó de color al voltant del jugador que té el torn, utilitzant colors diferents per a cada jugador per facilitar la identificació visual.

Autor: Albert Usero

Cardinalitat: Una instància en tot el programa

Mètodes

public void drawPlayerHighlight(Graphics g, int panelHeight, int rectangleWidth, int selectedPlayer, int numberOfPlayers)

Dibuixa el resaltat del jugador actual al panell.

Paràmetres

- **g**:Context gràfic on es dibuixarà el resaltat
- **panelHeight**: Alçada total del panell contenidor
- **rectangleWidth**: Amplada disponible per al resaltat
- **selectedPlayer**: Índex del jugador seleccionat (0-3)
- **numberOfPlayers**: Nombre total de jugadors participants

private void setPlayerColor(Graphics g, int selectedPlayer)

Assigna el color del resaltat segons el jugador seleccionat.

Colors assignats:

- Jugador 0: Vermell (#f52e2e)
- Jugador 1: Blau (#5463ff)
- Jugador 2: Groc (#ffc717)
- Jugador 3: Verd (#43e81f)

Paràmetres

- **g**: Context gràfic on s'aplicarà el color
- **selectedPlayer**:Índex del jugador seleccionat (0-3)

Classe PlayerInfo

Component gràfic que mostra la informació dels jugadors durant la partida. Aquesta classe s'encarrega de renderitzar visualment: El nom de cada jugador, la seva puntuació actual, un indicador de color que identifica a cada jugador. Els elements es mostren en targetes individuals amb disseny arrodonit.

Autor: Albert Usero

Cardinalitat: Una instància en tot el programa

Mètodes

public void drawPlayerInfo(Graphics g, int panelHeight, int rectangleWidth, Player[] players)

Dibuixa la informació de tots els jugadors al panell.

Paràmetres

- **g**: Context gràfic on es dibuixarà la informació
- **panelHeight**: Alçada total del panell contenidor
- **rectangleWidth**: Amplada disponible per a les targetes
- **players**: Llista de jugadors amb la seva informació

private void setPlayerColor(Graphics g, int selectedPlayer)

Assigna el color identificatiu per a cada jugador.

Els colors segueixen el mateix patró que en PlayerHighlight per consistència:

- Jugador 0: Vermell (#f52e2e)
- Jugador 1: Blau (#5463ff)
- Jugador 2: Groc (#ffc717)
- Jugador 3: Verd (#43e81f)

Paràmetres

- **g**: Context gràfic on s'aplicarà el color
- **selectedPlayer**: Índex del jugador (0-3)

Classe SidePanel

Panell lateral que mostra informació dels jugadors durant la partida. Aquest component gràfic s'encarrega de visualitzar: El jugador actual (resaltat), les puntuacions i noms dels jugadors, utilitza PlayerHighlight i PlayerInfo per renderitzar els elements.

Autor: Albert Usero

Cardinalitat: Una instància en tot el programa

Camps

- **playerHighlight**: Component gràfic que s'encarrega de resaltar visualment el jugador actual.
- **playerInfo**: Component gràfic que mostra la informació dels jugadors.
- **players**: Llista immutable dels jugadors participants.

S'inicialitza al constructor i no pot modificar-se posteriorment. Cada element conté les dades d'un jugador (nom, puntuació, etc.).

Constructores

public SidePanel(Player[] players)

Constructor que inicialitza el panell amb la llista de jugadors.

Paràmetres

- **players**: Llista de jugadors que participen en la partida.

Mètodes

public void setCurrentPlayer(int player)

Canvia el jugador actual i actualitza la visualització.

Si el valor és invàlid, no es realitza cap canvi.

Paràmetres

- **player**: Índex del nou jugador actual (entre 0 i 3).

protected void paintComponent(Graphics g)

Sobreescriptura del mètode de pintat per personalitzar l'aparença del panell.

S'encarrega de:

1. Pintar el fons amb un color fosc.
2. Renderitzar el resaltat del jugador actual.
3. Mostrar la informació dels jugadors.

Paràmetres

- **g**: Context gràfic utilitzat per al pintat.

Relacions

- Relació d'agregació amb PlayerInfo
- Relació d'agregació amb PlayerHighlight

Classe BoardCell

Casella del tauler que conté una fitxa (BoardTile).

Aquesta classe gestiona una cel·la del tauler on es pot afegir o substituir una fitxa visual del joc.

Autor: Gerard Gascón

Camps

- **tile**: Fitxa que conté la cel·la.

Constructores

public BoardCell()

Crea una nova cel·la del tauler sense fitxa i amb fons transparent.

Mètodes

public void setTile(BoardTile tile)

Assigna una fitxa a la cel·la. Si ja hi havia una fitxa, la substitueix.

Paràmetres

- **tile**: la fitxa que s'afegirà a la cel·la

public BoardTile getTile()

Retorna la fitxa que conté la cel·la.

Retorna: la fitxa actual de la cel·la, o null si no en té

Relacions

- relació d'associació amb playerHighlight
- relació d'associació amb playerInfo
- relació d'associació amb Player

Classe BoardCoordinateTile

Fitxa que mostra una coordenada al tauler (lletra o número).

Aquesta classe representa una cel·la que conté una lletra de coordenada i que es dibuixa centrada dins la cel·la.

Autor: Gerard Gascón

Camps

- **letter**: Lletra o símbol que representa la coordenada.

Constructores

public BoardCoordinateTile(String coordinate)

Crea una nova fitxa de coordenada amb la lletra especificada.

Paràmetres

- **coordinate**: La lletra o símbol de la coordenada

Mètodes

protected void paintComponent(Graphics g)

Pinta el component amb antialiasing per a millorar la qualitat i dibuixa la lletra de la coordenada.

public boolean contains(int x, int y)

Comprova si un punt està dins la forma rectangular de la fitxa.

Retorna: true si el punt està dins la fitxa, false altrament

Paràmetres

- x:coordenada x del punt
- y: coordenada y del punt

protected void drawTile(Graphics2D g, Color bg)

Dibuixa la fitxa. Per defecte, només dibuixa la lletra.

Paràmetres

- **g**: elcontext gràfic
- **bg**: el color de fons (no s'utilitza aquí)

private void drawLetter(Graphics2D g)

Dibuixa la lletra de la coordenada centrada dins la fitxa.

Paràmetres

- **g**: el context gràfic

Relacions

- Relació d'associació amb Letter.

Classe BoardEmptyTile

Fitxa buida del tauler.

Representa una cel·la del tauler sense cap premi ni peça especial, simplement un espai disponible per posar una fitxa.

Autor: Gerard Gascón

Constructores

public BoardEmptyTile(int x, int y, IHandView handView, BoardView boardView, IBlankPieceSelector blankPieceSelector)

Crea una fitxa buida a la posició (x,y).

Paràmetres

- **x**: coordenada x al tauler
- **y**: coordenada y al tauler
- **handView**: vista de la mà de fitxes
- **boardView**: vista del tauler
- **blankPieceSelector**: selector per fitxes en blanc

Mètodes

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa la fitxa buida com un rectangle arrodonit amb el color de fons.

Paràmetres

- **g**: context gràfic 2D
- **bg**: color de fons
- **radius**: radi de les cantonades arrodonides

Classe BoardPieceTile

Fitxa amb una lletra i puntuació del tauler.

Representa una peça que es pot col·locar sobre el tauler, amb una lletra i els punts associats a aquesta.

Autor: Gerard Gascón

Camps

- **letter**: Lletra representada per aquesta fitxa.
- **points**: Punts associats a la lletra.

Constructores

public BoardPieceTile(String letter, int points, int x, int y, IHandView handView, BoardView boardView, IBlankPieceSelector blankPieceSelector)

Crea una fitxa amb lletra i punts a la posició (x, y).

Paràmetres

- **letter**: lletra de la fitxa
- **points**: punts de la fitxa
- **x**: coordenada x al tauler
- **y**: coordenada y al tauler
- **handView**: vista de la mà de fitxes
- **boardView**: vista del tauler
- **blankPieceSelector**: selector per fitxes en blanc

Mètodes

public String getLetter()

Retorna la lletra de la fitxa.

Retorna: lletra de la fitxa

public int getPoints()

Retorna els punts de la fitxa.

Retorna: punts de la fitxa

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa la fitxa amb el seu fons, lletra i punts.

Paràmetres

- **g**: context gràfic 2D
- **bg**: color de fons
- **radius**: radi de cantonades arrodonides

private void drawLetter(Graphics2D g)

Dibuixa la lletra centrada a la fitxa. Si la lletra té més d'un caràcter, aplica un escalat horitzontal.

Paràmetres

- **g**: context gràfic 2D

protected void clicked(ActionEvent actionEvent)

Override del mètode per evitar que es cridi la funcionalitat de clic del pare.

Paràmetres

- **actionEvent**: esdeveniment d'acció

private void drawScore(Graphics2D g)

Dibuixa la puntuació a la cantonada inferior dreta de la fitxa.

Paràmetres

- **g**: context gràfic 2D

Relacions

- Relació d'associació amb Letter.

Classe BoardTemporalPieceTile

Fitxa temporal amb lletra i punts del tauler.

Representa una peça que s'ha col·locat temporalment sobre el tauler, mostrant la lletra i la seva puntuació amb transparència.

Autor: Gerard Gascón

Cardinalitat: Una instància per cada peça temporal sobre el tauler

Camps

- **letter**: Lletra representada per aquesta fitxa temporal.
- **points**: Punts associats a la lletra.

Constructores

public BoardTemporalPieceTile(String letter, int points, int x, int y, IHandView handView, BoardView boardView, IBlankPieceSelector blankPieceSelector)

Crea una fitxa temporal amb lletra i punts a la posició (x, y).

Paràmetres

- **letter**: lletra de la fitxa
- **points**: punts de la fitxa
- **x**: coordenada x al tauler
- **y**: coordenada y al tauler
- **handView**: vista de la mà de fitxes
- **boardView**: vista del tauler
- **blankPieceSelector**: selector per fitxes en blanc

Mètodes

public String getLetter()

Retorna la lletra de la fitxa temporal.

Retorna: lletra de la fitxa

public int getPoints()

Retorna els punts de la fitxa temporal.

Retorna: punts de la fitxa

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa la fitxa amb fons transparent, lletra i punts.

Paràmetres

- **g**: context gràfic 2D
- **bg**: color de fons (amb transparència)
- **radius**: radi de cantonades arrodonides

protected void clicked(ActionEvent actionEvent)

Override per evitar cridar el mètode clic del pare.

Paràmetres

- **actionEvent**: esdeveniment d'acció

private void drawLetter(Graphics2D g)

Dibuixa la lletra centrada a la fitxa temporal. Aplica un escalat horitzontal si la lletra té més d'un caràcter.

Paràmetres

- **g**: context gràfic 2D

private void drawScore(Graphics2D g)

Dibuixa la puntuació a la cantonada inferior dreta de la fitxa temporal.

Paràmetres

- **g**: context gràfic 2D

Relacions

- Relació d'associació amb Letter.

Classe BoardTile

Classe abstracta que representa una fitxa del tauler.

És un botó personalitzat amb posició al tauler, que pot contenir una peça i permet interaccions amb la mà i la selecció de fitxes en blanc.

Autor: Gerard Gascón

Camps

- **handView**: Vista de la mà de fitxes de l'usuari.
- **blankPieceSelector**: Selector per fitxes en blanc (wildcards).
- **position**: Posició de la fitxa al tauler.
- **board**: Vista del tauler on es col·loca la fitxa.
- **tooltip**: Text d'ajuda que surt en algunes caselles quan els hi passes el ratolí per sobre.

Constructores

public BoardTile(int x, int y, IHandView handView, BoardView board, IBlankPieceSelector blankPieceSelector)

Constructor amb posició, vista de la mà, tauler i selector fitxa en blanc.

Paràmetres

- **x:** coordenada x al tauler
- **y:** coordenada y al tauler
- **handView**: vista de la mà de fitxes
- **board**: vista del tauler
- **blankPieceSelector**: selector per fitxes en blanc

Mètodes

public Vector2 getPosition()

Retorna la posició (x,y) de la fitxa.

Retorna: vector de la posició

private void disableKeyboardInput()

Deshabilita la resposta a les tecles SPACE i ENTER per evitar activacions accidentals.

protected void paintComponent(Graphics g)

Pinta la fitxa amb un fons que canvia segons l'estat del botó (normal, ressaltat o premut).

Paràmetres

- **g**: context gràfic

protected abstract void drawTile(Graphics2D g, Color bg, int radius)

Mètode abstracte per pintar la fitxa, ha de ser implementat per les subclasses.

Paràmetres

- **g**: context gràfic 2D
- **bg**: color de fons
- **radius**: radi de cantonades arrodonides

public boolean contains(int x, int y)

Sobrescriu el mètode contains per fer clics dins una figura de cantonades arrodonides.

Retorna: true si el punt està dins la fitxa, false en cas contrari

Paràmetres

- **x**: coordenada x del punt a testear
- **y**: coordenada y del punt a testear

private int getCornerRadius()

Retorna el radi de les cantonades arrodonides basat en l'alçada.

Retorna: radi en píxels

protected void clicked(ActionEvent actionEvent)

Mètode invocat quan es fa clic sobre la fitxa. Col·loca la peça seleccionada temporalment al tauler, i si és una fitxa en blanc, obre un selector de lletra.

Paràmetres

- **actionEvent**: esdeveniment d'acció

private void openSelectBlankPieceLetterPopup()

Obre el popup per seleccionar la lletra d'una fitxa en blanc.

private void placeBlankPiece(String letter)

Col·loca la fitxa en blanc amb la lletra seleccionada i 0 punts.

Paràmetres

- **letter**: lletra seleccionada per la fitxa en blanc

private void placePiece(String piece, int points)

Col·loca temporalment una peça al tauler a la posició d'aquesta fitxa.

Paràmetres

- **piece**: lletra de la peça
- **points**: punts de la peça

protected final void createTooltip(String text)

Crea un tooltip personalitzat que apareix quan es mou el cursor sobre la fitxa.

Paràmetres

- **text**: text del tooltip

Relacions

- Relació d'associació amb HandView.
- Relació d'associació amb BlankPieceSelector.
- Relació d'associació amb BoardView.
- Relació d'associació amb ToolTip.

Classe BoardCenterTile

Classe que representa la casella central del tauler de Scrabble.

Aquesta casella marca el punt inicial del joc i també és una casella de puntuació doble de paraula. Es dibuixa amb un fons rosa i una estrella central.

Autor: Gerard Gascón

Constructores

public BoardCenterTile(int x, int y, IHandView handView, BoardView boardView, IBlankPieceSelector blankPieceSelector)

Constructor per a la casella central.
Paràmetres

- **x:** coordenada X de la casella.
- **y**: coordenada Y de la casella.
- **handView**: vista de la mà del jugador.
- **boardView**: vista general del tauler.
- **blankPieceSelector**: selector de fitxes en blanc.

Mètodes

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa la casella amb un rectangle arrodonit i una estrella central.

Paràmetres

- **g:** objecte gràfic
- **bg**: color de fons
- **radius**: radi de les vores arrodonides

Classe BoardDoubleLetterTile

Casella de Doble Lletra del tauler de joc.

Aquesta casella proporciona una bonificació de puntuació per doble lletra quan s'hi col·loca una fitxa. Es representa amb un color blau clar i mostra un missatge emergent explicatiu.

Autor: Gerard Gascón

Constructores

public BoardDoubleLetterTile(int x, int y, IHandView handView, BoardView boardView, IBlankPieceSelector blankPieceSelector)

Crea una nova instància de la casella de doble lletra.

Paràmetres

- **x**: coordenada horitzontal al tauler
- **y:** coordenada vertical al tauler
- **handView**: vista de la mà del jugador
- **boardView**: vista del tauler
- **blankPieceSelector**: selector de fitxes blanques

Mètodes

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa la casella amb el color de fons corresponent.

Paràmetres

- **g:** context gràfic utilitzat per dibuixar
- **bg**: color de fons de la casella
- **radius**: radi de les cantonades arrodonides

Classe BoardDoubleWordTile

Casella de Doble Paraula del tauler de joc.

Aquesta casella aplica una bonificació de puntuació per doble paraula quan s'hi col·loca una fitxa. Es representa amb un color rosa i mostra una descripció emergent informativa.

Autor: Gerard Gascón

Constructores

public BoardDoubleWordTile(int x, int y, IHandView handView, BoardView boardView, IBlankPieceSelector blankPieceSelector)

Crea una nova instància de la casella de doble paraula.

Paràmetres

- **x**: coordenada horitzontal al tauler
- **y**: coordenada vertical al tauler
- **handView**: vista de la mà del jugador
- **boardView**: vista del tauler
- **blankPieceSelector**: selector de fitxes blanques

Mètodes

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa la casella amb el color de fons i forma arrodonida.

Paràmetres

- **g:** context gràfic per dibuixar
- **bg**: color de fons
- **radius**: radi de les cantonades

Classe BoardQuadrupleLetterTile

Casella de Lletra Quàdruple del tauler de joc.

Aquesta casella aplica una bonificació de puntuació per lletra quàdruple quan s'hi col·loca una fitxa. Es representa amb un color blau fosc i mostra una descripció emergent informativa.

Autor: Gerard Gascón

Constructores

public BoardQuadrupleLetterTile(int x, int y, IHandView handView, BoardView boardView, IBlankPieceSelector blankPieceSelector)

Crea una nova instància de la casella de lletra quàdruple.

Paràmetres

- **x**: coordenada horitzontal al tauler
- **y**: coordenada vertical al tauler
- **handView**: vista de la mà del jugador
- **boardView**: vista del tauler
- **blankPieceSelector**: selector de fitxes blanques

Mètodes

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa la casella amb el color de fons i forma arrodonida.

Paràmetres

- **g:** context gràfic per dibuixar
- **bg**: color de fons
- **radius**: radi de les cantonades

Classe BoardQuadrupleWordTile

Casella de Paraula Quàdruple del tauler de joc.

Aquesta casella aplica una bonificació de puntuació per paraula quàdruple quan una paraula la travessa. Es representa amb un color vermell fosc i mostra una descripció emergent informativa.

Autor: Gerard Gascón

Constructores

public BoardQuadrupleWordTile(int x, int y, IHandView handView, BoardView boardView, IBlankPieceSelector blankPieceSelector)

Crea una nova instància de la casella de paraula quàdruple.

Paràmetres

- **x**: coordenada horitzontal al tauler
- **y**: coordenada vertical al tauler
- **handView**: vista de la mà del jugador
- **boardView**: vista del tauler
- **blankPieceSelector**: selector de fitxes blanques

Mètodes

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa la casella amb el color de fons i forma arrodonida.

Paràmetres

- **g:** context gràfic per dibuixar
- **bg**: color de fons
- **radius**: radi de les cantonades

Classe BoardTripleLetterTile

Casella de Lletra Triple del tauler de joc.

Aquesta casella aplica una bonificació de puntuació per lletra triple quan una lletra és col·locada en aquesta posició. Es representa amb un color blau intens i mostra una descripció emergent informativa.

Autor: Gerard Gascón

Constructores

public BoardTripleLetterTile(int x, int y, IHandView handView, BoardView boardView, IBlankPieceSelector blankPieceSelector)

Crea una nova instància de la casella de lletra triple.

Paràmetres

- **x**: coordenada horitzontal al tauler
- **y**: coordenada vertical al tauler
- **handView**: vista de la mà del jugador
- **boardView**: vista del tauler
- **blankPieceSelector**: selector de fitxes blanques

Mètodes

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa la casella amb el color de fons i forma arrodonida.

Paràmetres

- **g**: context gràfic per dibuixar
- **bg**: color de fons
- **radius**: radi de les cantonades

Classe BoardTripleWordTile

Casella de Paraula Triple del tauler de joc.

Aquesta casella aplica una bonificació de puntuació per paraula triple quan una paraula és col·locada en aquesta posició. Es representa amb un color vermell fosc i mostra una descripció emergent informativa.

Autor: Gerard Gascón

Constructores

public BoardTripleWordTile(int x, int y, IHandView handView, BoardView boardView, IBlankPieceSelector blankPieceSelector)

Crea una nova instància de la casella de paraula triple.

Paràmetres

- **x**: coordenada horitzontal al tauler
- **y**: coordenada vertical al tauler
- **handView**:vista de la mà del jugador
- **boardView**: vista del tauler
- **blankPieceSelector**: selector de fitxes blanques

Mètodes

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa la casella amb el color de fons i forma arrodonida.

Paràmetres

- **g**: context gràfic per dibuixar
- **bg**: color de fons
- **radius**: radi de les cantonades

Classe EndScreen

EndScreen és un JPanel que mostra els resultats finals d'una partida de Scrabble. Mostra un podi amb els 3 primers jugadors, una llista completa de classificació, i proporciona botons per iniciar una nova partida, tornar al menú, o sortir de l'aplicació.

La pantalla presenta un fons amb degradat amb una visualització de podi i targetes de jugadors codificades per colors que mostren puntuacions i posicions.

Autor: Albert Usero

Cardinalitat: Una instància en tot el programa

Camps

- **SIDE_PANEL_WIDTH_PERCENT**: Percentatge d'amplada de pantalla ocupat pel panell lateral
- **PODIUM_HEIGHT_PERCENT**: Percentatge d'alçada de pantalla ocupat per l'àrea del podi
- **PLAYER_CARD_HEIGHT**: Alçada en píxels de cada targeta de jugador a la llista de classificació
- **PLAYER_CARD_MARGIN**: Marge en píxels entre targetes de jugadors
- **players**: Array de jugadors de la partida
- **sortedPlayers**: Array de jugadors ordenats per puntuació (ordre descendent)
- **buttonPanel**: Panell que conté els botons d'acció
- **onMenuCallback**: Callback per tornar al menú principal
- **onNewGameCallback**: Callback per iniciar una nova partida

Constructores

public EndScreen()

Constructor per defecte que crea un EndScreen sense jugadors. Inicialitza el component amb l'estil per defecte i arrays de jugadors buits.

public EndScreen(Player[] players)

Constructor que crea un EndScreen amb els jugadors especificats.

Paràmetres

- **players**: Array de jugadors per mostrar a la pantalla final

Mètodes

public void show(Player[] sortedPlayers)

Mostra la pantalla final amb l'array de jugadors ordenats proporcionat. Fa visible la finestra, la porta al davant i demana el focus.

Paràmetres

- **sortedPlayers**: Array de jugadors ordenats per puntuació final

public void setPlayers(Player[] players)

Estableix l'array de jugadors per a la pantalla final. Els jugadors ja haurien d'estar ordenats per puntuació en ordre descendent.

Paràmetres

- **players**: Array de jugadors ordenats per puntuació, o null per array buit

public void setOnMenuCallback(ActionListener callback)

Estableix la funció callback que s'executarà quan es cliqui el botó "Menú Principal".

Paràmetres

- **callback**: ActionListener per gestionar els clics del botó menú

public void setOnNewGameCallback(ActionListener callback)

Estableix la funció callback que s'executarà quan es cliqui el botó "Nova Partida".

Paràmetres

- **callback**: ActionListener per gestionar els clics del botó nova partida

private void initializeButtons()

Inicialitza i configura el panell de botons amb els botons d'acció. Crea botons per "Nova Partida", "Menú Principal" i "Sortir" amb els seus respectius gestors.

protected void paintComponent(Graphics g)

Mètode de pintat personalitzat que renderitza la interfície completa de la pantalla final. Dibuixa el fons, títol i resultats amb antialiasing activat.

Paràmetres

- **g**: Context gràfic per dibuixar

private void drawBackground(Graphics2D g2, int width, int height)

Dibuixa el fons dividit amb un panell lateral fosc i àrea principal amb degradat.

Paràmetres

- **g2**: Context Graphics2D per dibuixar
- **width**: Amplada del component
- **height**: Alçada del component

private void drawTitle(Graphics2D g2, int width, int height)

Dibuixa el títol i l'anunci del guanyador a la part superior de la pantalla.

Paràmetres

- **g2**: Context Graphics2D per dibuixar
- **width**: Amplada del component
- **height**: Alçada del component

private void drawResults(Graphics2D g2, int width, int height)

Dibuixa la secció de resultats incloent tant el podi com la llista completa de jugadors.

Paràmetres

- **g2**: Context Graphics2D per dibuixar
- **width**: Amplada del component
- **height**: Alçada del component

private void drawPodium(Graphics2D g2, int x, int y, int width, int height)

Dibuixa la visualització del podi per als 3 primers jugadors. El podi mostra les posicions 1a, 2a i 3a amb diferents alçades i mostra noms de jugadors i puntuacions.

Paràmetres

- **g2**: Context Graphics2D per dibuixar
- **x**: Coordenada X de l'àrea del podi
- **y**: Coordenada Y de l'àrea del podi
- **width**: Amplada de l'àrea del podi
- **height**: Alçada de l'àrea del podi

private void drawPlayerList(Graphics2D g2, int x, int y, int width)

Dibuixa la llista completa de classificació de tots els jugadors en format desplaçable. Cada jugador es mostra en format targeta amb posició, nom i puntuació.

Paràmetres

- **g2**: Context Graphics2D per dibuixar
- **x**: Coordenada X de l'àrea de la llista de jugadors
- **y**: Coordenada Y de l'àrea de la llista de jugadors
- **width**: Amplada de l'àrea de la llista de jugadors

private void drawPlayerCard(Graphics2D g2, Player player, int x, int y, int width, int height, int position)

Dibuixa una targeta individual de jugador mostrant el seu rang, nom, puntuació i indicador de CPU. Cada targeta està codificada per colors basant-se en la posició del jugador.

Paràmetres

- **g2**: Context Graphics2D per dibuixar
- **player**: Objecte Player que conté la informació del jugador
- **x**: Coordenada X de la targeta
- **y**: Coordenada Y de la targeta
- **width**: Amplada de la targeta
- **height**: Alçada de la targeta
- **position**: Posició de classificació del jugador (1r, 2n, etc.)

private void setPlayerColor(Graphics g, int playerIndex)

Estableix el color gràfic basat en l'índex del jugador utilitzant un esquema de colors predefinit. Els colors van rotant entre vermell, blau, groc i verd per fins a 4 jugadors.

Paràmetres

- **g**: Context gràfic on establir el color
- **playerIndex**:Índex del jugador (basat en 0)

public void doLayout()

Posiciona el panell de botons a la cantonada inferior esquerra de la pantalla. Es crida automàticament durant les actualitzacions del layout.

Relacions

- relació d'associació amb Player

Classe GameScreen

Pantalla principal del joc Scrabble. Gestiona la disposició visual del tauler, la mà del jugador, el panell lateral, el selector de fitxes en blanc i el menú de pausa.

Aquesta classe s'encarrega de col·locar dinàmicament els components en funció de les dimensions de la finestra.

Autor: Gerard Gascón

Camps

- **SIDE_PANEL_WIDTH_PERCENTAGE**: Percentatge d'amplada reservada per al panell lateral.
- **BOARD_VERTICAL_SIZE_PERCENTAGE**: Percentatge d'alçada destinat al tauler del joc.
- **BOARD_HORIZONTAL_OFFSET_PERCENTAGE**: Percentatge de desplaçament horitzontal des de l'esquerra per col·locar el tauler.
- **boardView**: Component que representa la vista gràfica del tauler de Scrabble.
- **sidePanel**: Component lateral amb informació del joc, com puntuacions o controls.
- **handView**: Component que representa la mà del jugador amb les fitxes disponibles.
- **blankPieceSelector**:Component emergent que permet seleccionar una lletra per a una fitxa en blanc.
- **pauseMenu**: Component de menú de pausa que permet interrompre el joc.

Constructores

public GameScreen()

Constructor que inicialitza la pantalla amb un disseny BorderLayout i un color de fons verdós.

Mètodes

public void addBoard(BoardView boardView)

Afegeix el tauler de joc a la pantalla.

Paràmetres

- boardView: la vista del tauler.

public void addSidePanel(SidePanel sidePanel)

Afegeix el panell lateral a la pantalla.

Paràmetres

- **sidePanel:** el panell lateral amb informació addicional.

public void addHandView(HandView handView)

Afegeix la vista de la mà del jugador a la pantalla.

Paràmetres

- **handView**: la vista de la mà amb les fitxes del jugador.

public void addBlankPieceSelector(BlankPieceSelector blankPieceSelector)

Afegeix el selector de fitxes en blanc a la pantalla i el posa al davant.

Paràmetres

- **blankPieceSelector**: el selector de fitxes en blanc.

public void addPauseButton(PauseMenu pauseMenu)

Afegeix el menú de pausa a la pantalla i el posa al davant.

Paràmetres

- **pauseMenu**: el menú de pausa del joc.

public void doLayout()

Sobreescriu el mètode de disposició dels components per organitzar-los manualment a la pantalla segons les proporcions establertes.

private void putBoard()

Col·loca el tauler en la posició i mida adequada dins la pantalla.

private void putSidePanel()

Col·loca el panell lateral a l'esquerra de la pantalla.

private void putHandView()

Col·loca la mà del jugador sota el tauler, centrada horitzontalment.

private void putBlankPieceSelector()

Estableix la mida i posició del selector de fitxes en blanc (ocupa tota la pantalla).

private void putPauseButton()

Estableix la mida i posició del menú de pausa (ocupa tota la pantalla).

Relacions

- Relació d'associació amb BoardView.
- Relació d'associació amb SidePanel.
- Relació d'associació amb HandView.
- Relació d'associació amb BlankPieceSelector.
- Relació d'associació amb PauseMenu.

Classe HandPieceButton

Botó personalitzat que representa una peça de lletra del jugador a la seva mà. Mostra la lletra i la seva puntuació, amb estil gràfic personalitzat.

Autor: Gina Escofet González

Camps

- **letter**: Lletra que representa la peça (pot ser una lletra simple o buida en cas de peça blanca).
- **points**: Puntuació associada a la peça. Normalment va de 0 a 10 segons el valor de la lletra.

Constructores

public HandPieceButton(String letter, int points)

Constructor del botó de peça de mà.

Paràmetres

Retorna: Radi en píxels

- **letter**: Lletra que representa la peça
- **points**: Puntuació de la lletra

Mètodes

private void disableKeyboardInput()

Inhabilita l'entrada per teclat com l'espai o enter per evitar selecció accidental del botó via teclat.

private int getCornerRadius()

Calcula el radi de les cantonades arrodonides basat en l'alçada del botó.

protected void paintComponent(Graphics g)

Personalitza el renderitzat del botó per mostrar la peça estilitzada.

public String getLetter()

Retorna la lletra associada al botó.

public int getPoints()

Retorna la puntuació de la lletra.

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa el rectangle arrodonit del botó i el seu contingut.

Paràmetres

- **g**: Gràfics
- **bg**: Color de fons
- **radius**: Radi de les cantonades

private void drawLetter(Graphics2D g)

Dibuixa la lletra centrada al botó. Fa un escalat si la lletra és múltiple (ex: "QU").

private void drawScore(Graphics2D g)

Dibuixa la puntuació en petit a la cantonada inferior dreta.

Relacions

- Relació d'associació amb Letter.

Classe HandView

Vista gràfica de la mà del jugador, que mostra les peces disponibles i permet seleccionar-ne una per jugar-la. Implementa la interfície IHandView per interactuar amb la lògica del joc.

Autor: Gina Escofet Gonzalez

Camps

- **pieceButtons**: Llista de botons que representen les peces de la mà del jugador.
- **selectedPieceButton**: Referència a la peça seleccionada actualment (si n'hi ha una). Només es pot seleccionar una peça a la vegada.
- **HAND_PIECES_SPACING**: Espai horitzontal (en píxels) entre les peces de la mà del jugador.

Constructores

public HandView()

Constructor. Configura el layout i la transparència del panell.

Mètodes

public String[] getSelectedPiece()

Retorna la lletra de la peça seleccionada actualment, si n'hi ha.

public int getSelectedPiecePoints()

Retorna els punts de la peça seleccionada actualment, o 0 si no n'hi ha.

public void showPieces(Piece[] pieces)

Mostra les peces de la mà. Crea botons per cada peça i gestiona la selecció.

Paràmetres

- **pieces**: Array de peces a mostrar

public void piecePlaced()

Elimina la peça seleccionada de la mà (després que hagi estat jugada).

protected void paintComponent(Graphics g)

Pinta el fons arrodonit de la mà del jugador.

public void setPieceSize(int handPieceSize)

Estableix la mida de totes les peces visibles a la mà.

Paràmetres

- **handPieceSize**: mida (amplada i alçada) de cada peça.

Relacions

- Relació d'associació amb HandPieceButton.

Classe BlankPieceSelector

Panell per seleccionar la fitxa blanca (comodí) en el joc. Implementa la interfície IBlankPieceSelector.

Autor: Gerard Gascón

Camps

- **overlay**: Panell superposat que conté el selector de fitxes.

Constructores

public BlankPieceSelector()

Constructor que inicialitza el panell amb layout nul i fons transparent.

Mètodes

public void openSelectorPopUp(Consumer selectPieceCallback)

Mostra un pop-up per seleccionar una fitxa, amb un callback que rep la fitxa escollida.

Paràmetres

- **selectPieceCallback**: Funció que s'executa quan es selecciona una fitxa.

Classe PieceSelectorButton

Botó personalitzat per al selector de peces amb estil propi i sense interacció per teclat. Aplica un fons rodó amb colors que canvien segons l'estat (pressionat, passant el ratolí).

Autor: Gerard Gascón

Constructores

public PieceSelectorButton(String text)

Crea un botó amb el text indicat i aplica l'estil personalitzat.

Paràmetres

- **text**: Text que es mostrarà al botó

Mètodes

private void disableKeyboardInput()

Desactiva la interacció del botó amb les tecles Espai i Enter per evitar activacions no desitjades via teclat.

protected void paintComponent(Graphics g)

Pinta el botó amb un fons rodó i colors adaptats segons l'estat.

Paràmetres

- **g**: Context gràfic per pintar

public boolean isOpaque()

Indica que el botó no és opac per permetre pintar fons personalitzat.

Retorna: false sempre

Classe PieceSelectorPanel

Panell personalitzat amb fons rodó i estil específic per al selector de peces. Aquest panell no és opac i pinta un fons amb cantonades arrodonides i color personalitzat.

Autor: Gerard Gascón

Constructores

public PieceSelectorPanel(LayoutManager layout)

Crea un panell amb el LayoutManager indicat.

Paràmetres

- **layout**: LayoutManager que s'aplicarà al panell; pot ser null

Mètodes

protected void paintComponent(Graphics g)

Pinta el panell amb un fons de cantonades arrodonides i color personalitzat.

Paràmetres

- **g**: Context gràfic per pintar

public boolean isOpaque()

Indica que aquest panell no és opac per permetre pintar fons personalitzats.

Retorna: false sempre

Classe UppercaseDocument

Document personalitzat que limita la longitud del text i converteix totes les lletres a majúscules.

Constructores

public UppercaseDocument(int maxChars)

Constructor que inicialitza el límit de caràcters.

Paràmetres

- **maxChars**: nombre màxim de caràcters

Mètodes

public void insertString(int offs, String str, AttributeSet a) throws BadLocationException

Insereix text convertit a majúscules, sempre que no excedeixi el límit màxim de caràcters.

Llança una BadLocationException si la posició és invàlida

Paràmetres

- **offs**: posició on inserir
- **str**: cadena a inserir
- **a**: atributs associats

public void remove(int offs, int len) throws BadLocationException

Elimina una part del text.

Llança una BadLocationException si la posició és invàlida

Paràmetres

- **offs**: posició inicial d'eliminació
- **len**: longitud a eliminar

Classe PieceWriter

Camp de text personalitzat per a l'entrada de peces, que limita el nombre de caràcters i converteix automàticament el text a majúscules. També personalitza la pintura per tenir un fons blanc amb cantonades arrodonides.

Autor: Gerard Gascón

Camps

- **MAX_CHARACTERS**: Màxim nombre de caràcters permesos.

Constructores

public PieceWriter()

Constructor que inicialitza el camp de text amb un document personalitzat, alineació centrada i estil de font.

Mètodes

public void setText(String text)

Estableix el text convertint-lo automàticament a majúscules.

Paràmetres

- **text**: text a establir

public boolean isOpaque()

Indica que el component no és opac per permetre la personalització del fons.

Retorna: false sempre

public void doLayout()

Ajusta la mida de la font en funció de l'altura del component.

protected void paintComponent(Graphics g)

Pinta el fons blanc amb cantonades arrodonides i després el contingut del JTextField.

Paràmetres

- **g**: context gràfic

private Font calculateFont(int height)

Calcula una mida de font basada en una proporció de l'altura del component.

Retorna: font ajustada a l'altura

Paràmetres

- **height**: altura del component

Classe ActionButtonPanel

Classe per gestionar el panell que conté tots els botons d'accions de torn: Robar (Draw), Col·locar (Place) i Passar (Skip).

Autor: Gina Escofet González

Constructores

public ActionButtonPanel(DrawAction drawAction, PlaceAction placeAction, SkipAction skipAction)

Construeix un nou ActionButtonPanel amb les accions de torn especificades. Aquest panell utilitza un FlowLayout per disposar els botons.

Paràmetres

- **drawAction**: L'objecte DrawAction que gestiona la lògica del botó de robar.
- **placeAction**: L'objecte PlaceAction que gestiona la lògica del botó de col·locar.
- **skipAction**: L'objecte SkipAction que gestiona la lògica del botó de passar torn.

Mètodes

protected void paintComponent(Graphics g)

S'encarrega de pintar el component amb un fons arrodonit de color negre, si ho desitges per aquest panell contenidor. Si no vols un fons per a aquest panell, pots eliminar o comentar aquest mètode.

Paràmetres

- **g**: L'objecte gràfic proporcionat pel sistema Swing per pintar.

public boolean isOpaque()

Indica que el panell no és opac per permetre transparència.

Classe DrawAction

Classe per gestionar la lògica del botó de Draw (Robar) en una partida.

Autor: Gina Escofet González

Camps

- **parent**: Panell contenidor on s'afegeixen i treuen els botons (Draw / Confirm).
- **drawBtn**: Botó principal que permet iniciar l'acció de robar peces.
- **drawActionMaker**:Gestor de la lògica que executa l'acció de robar peces un cop confirmada.
- **handView**: Vista de la mà del jugador, des d'on es recuperen les peces seleccionades per intercanviar.

Constructores

public DrawAction(JPanel parent, DrawActionMaker drawActionMaker, HandView handView)

Construeix un objecte DrawAction.

Paràmetres

- **parent**: El panell pare on s'afegirà aquest component.
- **drawActionMaker**: L'objecte responsable de gestionar la lògica de robar peces.
- **handView**: La vista de la mà del jugador per interactuar amb les peces seleccionades.

Mètodes

private void createDrawButton()

Crea el botó de "Draw" i configura el seu comportament. Quan es prem, el botó canvia a un botó "Confirm" per executar l'acció real.

public boolean isOpaque()

Indica si aquest panell és opac. Retorna sempre fals per assegurar que el fons sigui transparent, evitant que el panell dibuixi un fons sòlid que pugui tapar altres components visuals.

Relacions

- Relació d'associació amb DrawActionMaker.
- Relació d'associació amb HandView.

Classe DrawActionButton

Classe per gestionar el botó de Draw (Robar) en una partida.

Autor: Gina Escofet González

Constructores

public DrawActionButton(String text)

Construeix un botó personalitzat per l'acció de "Draw" (Robar).

Paràmetres

- **text**: El text que es mostrarà al botó.

Mètodes

private void disableKeyboardInput()

Deshabilita l'ús de les tecles Espai i Enter quan el botó està en focus, evitant així que es disparin accions no desitjades amb aquestes tecles.

protected void paintComponent(Graphics g)

Pinta el botó amb cantonades arrodonides. El color de fons varia segons l'estat: més fosc quan està premut i més clar quan el cursor està sobre el botó.

Paràmetres

- **g**: L'objecte gràfic usat per pintar el component.

public boolean isOpaque()

Indica que el botó no és opac per permetre efectes de transparència.

Retorna: fals per permetre transparència.

Relacions

- relació d'associació amb DrawActionMaker.
- relació d'associació amb HandView.

Classe DrawActionPanel

Classe per gestionar el disseny del botó de Draw (Robar) en una partida.

Autor: Gina Escofet González

Constructores

public DrawActionPanel(LayoutManager layout)

Retorna: fals, per permetre transparència visual.

Construeix un nou DrawActionPanel amb el gestor de disseny especificat.

Paràmetres

- **layout**: El LayoutManager que s'utilitzarà per a aquest panell.

Mètodes

protected void paintComponent(Graphics g)

S'encarrega de pintar el component amb un fons arrodonit de color negre. També aplica suavitzat per millorar la qualitat visual dels gràfics.

Paràmetres

- **g**: L'objecte gràfic proporcionat pel sistema Swing per pintar.

public boolean isOpaque()

Indica que el panell no és opac. Això permet que el fons del panell sigui transparent i no tapi altres components.

Classe PlaceAction

Classe per gestionar la lògica del botó de Place (Posar) en una partida.

Autor: Gina Escofet González

Camps

- **parent**: Panell pare on s'insereix aquest component i on es gestionen els botons.
- **placeBtn**: Botó per iniciar l'acció de posar peces al tauler.
- **confirmBtn**:Botó per confirmar la col·locació de peces al tauler.
- **cancelBtn**: Botó per cancel·lar la col·locació i retornar les peces a la mà.
- **placeActionMaker**: Lògica que gestiona l'acció de posar peces al tauler.
- **boardView**:Vista del tauler de joc, per accedir a la col·locació actual.
- **handView**:Vista de la mà del jugador, per gestionar les peces seleccionades o retornades.

Constructores

public PlaceAction(JPanel parent, PlaceActionMaker placeActionMaker, BoardView boardView, HandView handView)

Construeix un objecte PlaceAction.

Paràmetres

- **parent**: El panell pare on s'afegirà aquest component.
- **placeActionMaker**: L'objecte responsable de gestionar la lògica de robar peces.
- **boardView**: La vista del tauler de la partida per interactuar amb les peces seleccionades.
- **handView**: La vista de la mà del jugador per interactuar amb les peces seleccionades.

Mètodes

private void createPlaceButton()

Crea el botó “Place” que inicia el procés de col·locació. Quan es prem, elimina aquest botó i mostra els botons “Confirm” i “Cancel”.

public boolean isOpaque()

Indica que aquest panell no és opac per permetre efectes de transparència.

Retorna: fals per indicar que el component no és opac.

Relacions

- relació d’associació amb PlaceActionMaker.
- relació d’associació amb BoardView.
- relació d’associació amb HandView.

Classe PlaceActionButton

Classe per gestionar el botó de Place (Posar) en una partida.

Autor: Gina Escofet González

Constructores

public PlaceActionButton(String text)

Construeix un botó personalitzat per l’acció de “Place” (Posar).

Retorna: fals sempre.

Paràmetres

- **text**: El text que es mostrarà al botó.

Mètodes

private void disableKeyboardInput()

Desactiva les tecles ESPAI i ENTER perquè no activin el botó.

protected void paintComponent(Graphics g)

Dibuixa el botó amb un fons rodó que canvia de tonalitat segons l’estat.

Paràmetres

- **g**: Context gràfic per dibuixar el component.

public boolean isOpaque()

Indica que el botó no és opac, cosa que permet efectes de transparència.

Classe PlaceActionPanel

Classe per gestionar el disseny del botó de Place (Posar) en una partida.

Autor: Gina Escofet González

Constructores

public PlaceActionPanel(LayoutManager layout)

Construeix un nou PlaceActionPanel amb el gestor de disseny especificat.

Paràmetres

- **layout:** El LayoutManager que s'utilitzarà per a aquest panell.

Mètodes

protected void paintComponent(Graphics g)

Personalitza el dibuix del component pintant un rectangle rodó negre amb qualitat antialiasing per suavitzar les vores i el text.

Paràmetres

- **g**: El context gràfic on es fa el dibuix.

public boolean isOpaque()

Indica que el panell no és opac per permetre transparències o dibuixos sota seu.

Classe SkipAction

Classe per gestionar la lògica del botó de Skip (Saltar) en una partida.

Autor: Gina Escofet González

Camps

- **parent**: Panell pare on s'afegirà el botó Skip.
- **skipBtn**: Botó que representa l'acció de saltar el torn.
- **skipActionMaker**: Objecte que gestiona la lògica associada a l'acció de Skip.

Constructores

public SkipAction(SkipActionMaker skipActionMaker, JPanel parent)

Construeix un objecte SkipAction.

Paràmetres

- **parent**: El panell pare on s'afegirà aquest component.
- **skipActionMaker**: L'objecte responsable de gestionar la lògica de robar peces.

Mètodes

private void createSkipButton()

Crea el botó Skip, li assigna posició fixa i el listener per executar l'acció. El botó s'afegeix directament al panell pare.

public boolean isOpaque()

Indica que el panell no és opac per permetre transparències o dibuixos sota seu.

Relacions

- relació d'associació amb SkipActionMaker.

Classe SkipActionButton

Classe per gestionar el botó de Skip (Saltar) en una partida.

Autor: Gina Escofet González

Constructores

public SkipActionButton(String text)

Construeix un botó personalitzat per l'acció de "Skip" (Saltar).

Paràmetres

- **text**: El text que es mostrarà al botó.

Mètodes

private void disableKeyboardInput()

Desactiva les tecles espai i enter per evitar que activin el botó per teclat.

protected void paintComponent(Graphics g)

Pinta el botó amb esquemes de color i cantonades arrodonides personalitzades. Utilitza efectes de sombrejat segons l'estat del botó (clicat, sobrevolat, normal).

public boolean isOpaque()

Indica que el panell no és opac per permetre transparències o dibuixos sota seu.

Retorna: fals sempre.

Classe SkipActionPanel

Classe per gestionar el disseny del botó de Skip (Saltar) en una partida.

Autor: Gina Escofet González

Constructores

public SkipActionPanel(LayoutManager layout)

Construeix un nou SkipActionPanel amb el gestor de disseny especificat.

Paràmetres

- **layout**: El LayoutManager que s'utilitzarà per a aquest panell.

Mètodes

protected void paintComponent(Graphics g)

Pinta el component amb un fons negre i cantonades arrodonides per a un estil personalitzat.

Paràmetres

- **g:** L'objecte Graphics utilitzat per dibuixar el component.

public boolean isOpaque()

Indica que el panell no és opac per permetre transparències o dibuixos sota seu.

Classe Tooltip

Finestra emergent personalitzada per mostrar textos d'ajuda (tooltip) amb estil. La finestra es crea transparent i amb cantonades arrodonides.

Autor: Gerard Gascón

Constructores

public Tooltip(Component owner, String text)

Crea un tooltip vinculat a un component propietari amb un text específic.

Paràmetres

- **owner**: component propietari sobre el qual es posicionarà el tooltip
- **text**: text a mostrar dins del tooltip

Mètodes

private static JLabel createTooltipLabel(String text)

Crea una etiqueta per al tooltip amb el text especificat.

Retorna: JLabel configurat amb el text i estil

Paràmetres

- **text**: text que mostrarà l'etiqueta

private static JPanel createTooltipPanel()

Crea un panell personalitzat amb cantonades arrodonides i fons blanc translúcid, que s'utilitza com a fons del tooltip.

Retorna: JPanel configurat per a contenir el tooltip

public void showAt(int x, int y)

Mostra el tooltip a la posició (x, y) especificada a la pantalla.

Paràmetres

- **x**: posició horitzontal on mostrar el tooltip
- **y**: posició vertical on mostrar el tooltip

Classe MenuButton

Classe personalitzada de botó Swing que representa un botó de menú estilitzat. Dibuixa el botó amb cantonades arrodonides i una lletra gran en estil Scrabble. També desactiva la interacció amb el teclat per evitar activacions accidentals amb ESPAI o ENTER.

Autor: Gerard Gascón

Camps

- **text**: Text que es mostra damunt del botó.

Constructores

public MenuButton(String text)

Crea un nou botó de menú amb el text especificat.

Paràmetres

- **text**: el text que es mostrarà al botó.

Mètodes

private void disableKeyboardInput()

Desactiva la interacció per teclat (ESPAI i ENTER) per evitar activacions no desitjades.

protected void paintComponent(Graphics g)

Pinta el component del botó amb efectes d'estil personalitzat, com ara colors segons l'estat (normal, sobresortit, premut) i forma arrodonida.

Paràmetres

- **g**: l'objecte Graphics que s'utilitza per dibuixar el botó.

protected void drawTile(Graphics2D g, Color bg, int radius)

Dibuixa la forma del botó com una fitxa amb cantonades arrodonides i el text centrat verticalment.

Paràmetres

- **g**: l'objecte Graphics2D utilitzat per al dibuix.
- **bg**: el color de fons del botó.
- **radius**: el radi de les cantonades arrodonides.

private void drawLetter(Graphics2D g)

Dibuixa el text del botó amb una font gran i negra, alineada verticalment al centre.

Paràmetres

- **g**: l'objecte Graphics2D utilitzat per al dibuix del text.

public boolean contains(int x, int y)

Comprova si el punt especificat es troba dins de la forma arrodonida del botó.

Retorna: true si el punt està dins del botó, false altrament.

Paràmetres

- **x**: la coordenada x del punt.
- **y**: la coordenada y del punt.

private int getCornerRadius()

Calcula el radi de les cantonades arrodonides basant-se en l'altura del botó.

Retorna: el radi de les cantonades.

Classe MenuScreen

Panel amb els elements a mostrar al menú principal

Autor: Felipe Martínez Lassalle

Cardinalitat: Una instància en tot el programa.

Relacions

- **Jpanel**: relació de composició.

Constructores

public MenuScreen()

Creadora i inicialitzadora

Mètodes

public JugarButton getPlayButton()

Consultora

public ContinueButton getContinueButton()

Consultora

public RanquingButton getRankingButton()

Consultora

public MenuButton getQuitButton()

Consultora

protected void paintComponent(Graphics g)

Dibuixa el component personalitzat, incloent el panell lateral fosc i el títol “Scrabble”. Aquesta funció s’encarrega de la renderització gràfica del component, utilitzant antialiasing per millorar la qualitat visual del text i les formes.

Paràmetres

- **g**: L’objecte Graphics proporcionat pel sistema per dibuixar.

public void doLayout()

Reorganitza la disposició del panell de botons dins del component. Calcula i assigna les dimensions i posició del primer component fill (suposadament un JPanel amb botons), col·locant-lo a la part inferior del panell lateral.

Classe PauseMenu

Panell del menú de pausa, que detecta la tecla ESC per obrir un panell d’opcions (overlay), com guardar o continuar la partida.

Autor: Biel Pérez Silvestre

Camps

- **gameSaver**: Objecte encarregat de gestionar el guardat de la partida.
- **overlay**: Panell superposat (overlay) que es mostra quan s’activa el menú de pausa.

Constructores

public PauseMenu(GameSaver gameSaver)

Constructor del menú de pausa. Configura l’input ESC per obrir el panell de pausa i assigna el gestor de guardat.

Paràmetres

- **gameSaver**: objecte encarregat de guardar l’estat actual de la partida

Mètodes

private void openPausePanel()

Obre el panell de pausa si no està ja actiu. Afegeix l’overlay a la vista, li dona el focus i actualitza l’escena.

public void closePauseMenu()

Tanca el menú de pausa eliminant la referència a l’overlay actiu.

Relacions

- relació d’associació amb GameSaver.

Classe PauseOverlay

Panell superposat (overlay) que enfosqueix la pantalla i mostra el menú de pausa. Gestiona la tecla ESC per tancar-lo.

Autor: Biel Pérez Silvestre

Camps

- **parent**: Panell sobre el ho apliquem tot

Constructores

public PauseOverlay(PauseMenu parent, GameSaver gameSaver)

Creadora del panell de pausa (overlay), que es mostra a sobre de la pantalla actual.

Paràmetres

Retorna: fals sempre, ja que es vol un fons translúcid

- **parent**: panell sobre el qual es mostra l'overlay
- **gameSaver**: objecte responsable de gestionar el guardat de la partida

Mètodes

public void closePanel()

Tanca l'overlay eliminant-lo del panell pare i actualitza la vista.

private void layoutComponents(GameSaver gameSaver)

Inicialitza i afegeix el panell de pausa a l'overlay.

Paràmetres

- **gameSaver**: objecte utilitzat per gestionar el guardat de partida

protected void paintComponent(Graphics g)

Pinta el fons translúcid per donar un efecte de popup sobre el contingut principal.

Paràmetres

- **g**: Context gràfic per pintar

public boolean isOpaque()

Indica que el panell no és completament opac.

2.5. Paquet d'utilitats

Enumeració Direction

Enumeració que representa les dues direccions possibles en què es pot col·locar una paraula en un tauler de Scrabble.

Autor: Gerard Gascón

Camps

- **Vertical**: La paraula es col·loca verticalment al tauler.
- **Horizontal**:La paraula es col·loca horitzontalment al tauler.

Interfície IRand

Interfície que defineix mètodes per generar enters aleatoris. És útil per aleatoritzar elements del joc, com ara la selecció o col·locació de fitxes.

Autor: Gerard Gascón

Mètodes

abstract int nextInt()

Genera un enter aleatori.

Retorna: Un enter aleatori.

abstract int nextInt(int bound)

Genera un enter aleatori dins del rang especificat.

Retorna: Un enter aleatori entre 0 (inclusiu) i el límit donat (exclusiu).

Paràmetres

- **bound**: El límit superior (exclusiu) per al nombre aleatori.

abstract int nextInt(int origin, int bound)

Genera un enter aleatori dins del rang especificat (inclusiu per l'origen, exclusiu pel límit).

Retorna: Un enter aleatori entre l'origen (inclusiu) i el límit (exclusiu).

Paràmetres

- **origin**: El límit inferior (inclusiu) per al nombre aleatori.
- **bound**: El límit superior (exclusiu) per al nombre aleatori.

Record Pair

Una classe genèrica simple per emmagatzemar un parell de valors de tipus potencialment diferents.

Autor: Gerard Gascón

Mètodes

public X first()

Obté el primer element del parell.

Retorna: El primer element de tipus X.

public Y second()

Obté el segon element del parell.

Retorna: El segon element de tipus Y.

Classe Rand

Rand és una implementació concreta de la interfície IRand. Proporciona mètodes per generar nombres aleatoris utilitzant la classe java.util.Random.

Autor: Gerard Gascón

Mètodes

public int nextInt()

Retorna un enter aleatori. Aquest mètode genera enters aleatoris dins de tot el rang possible d'enters.

Retorna: un enter aleatori.

public int nextInt(int bound)

Retorna un enter aleatori des de 0 (inclusiu) fins al límit especificat (exclusiu).

Retorna: un enter aleatori dins del rang [0, bound).

Paràmetres

- **bound**: el límit superior (exclusiu)per al nombre aleatori.

public int nextInt(int origin, int bound)

Retorna un enter aleatori entre l'origen especificat (inclusiu) i el límit (exclusiu).

Retorna: un enter aleatori dins del rang [origin, bound).

Paràmetres

- **origin**: el límit inferior (inclusiu) per al nombre aleatori.
- **bound**: el límit superior (exclusiu) per al nombre aleatori.

Classe Vector2

Classe que representa un vector o punt 2D amb coordenades enteres. Aquesta classe proporciona mètodes per a operacions vectorials bàsiques, com la suma, així com la redefinició de mètodes per a la igualtat i la representació en forma de cadena.

Autor: Gerard Gascón

Camps

- **x**: La coordenada x del vector.
- **y**: La coordenada y del vector.

Constructores

public Vector2()

Construeix un Vector2 amb x i y inicialitzats a 0.

public Vector2(int x, int y)

Construeix un Vector2 amb les coordenades x i y especificades.

Paràmetres

- **x**: la coordenada x del vector.
- **y**: la coordenada y del vector.

Mètodes

public Vector2 add(Vector2 other)

Suma el vector donat a aquest vector i retorna un nou Vector2 com a resultat.

Retorna: un nou Vector2 que representa la suma d'aquest vector amb el vector donat.

Paràmetres

- **other**: el vector a sumar a aquest vector.

public boolean equals(Object obj)

Compara aquest vector amb un altre objecte per veure si són iguals. Els vectors es consideren iguals si tenen les mateixes coordenades x i y.

Retorna: true si l'objecte donat és igual a aquest vector, false altrament.

Paràmetres

- **obj**: l'objecte amb el qual comparar aquest vector.

public int hashCode()

Retorna el codi hash d'aquest vector basat en les seves coordenades x i y.

Retorna: el codi hash d'aquest vector.

public String toString()

Retorna una representació en forma de cadena del vector en el format: Vector2[x=, y=].

Retorna: una representació en forma de cadena d'aquest vector.

3. Algorismes utilitzats

3.1. DAWG

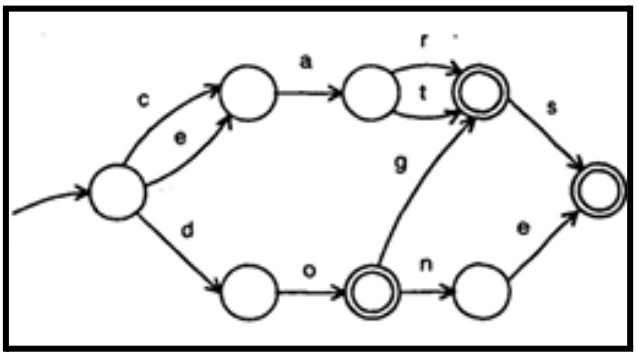
Seguint el paper ***The World's Fastest Scrabble Program***[1] per emmagatzemar de manera eficient els diccionaris vam fer servir un *Directed Acyclic Word Graph*[2] (DAWG). El principal avantatge que presenta aquesta estructura de dades respecte a un *Trie* és que redueix significativament l'ús de memòria mitjançant la compartició de nodes idèntics, evitant redundàncies.

Cada node del DAWG conté:

- **Caràcter**: El símbol que representa. Per representar peces que tinguin 2 o més caràcters (ex. l·l) caldrà més d'un node.
- **Successors**: HashMap<Character, Node>. Representa els nodes fills. Permet fer un accés ràpid al node successor de cert caràcter si el té.
- **Pare**: Referència al node pare.
- **Profunditat**: Distància des de l'arrel.
- **Hash**: Codi únic calculat en funció dels altres atributs del node (incloent-hi successors). El propòsit d'aquest codi hash és fer la comparació de si 2 nodes són iguals molt més ràpida, comparant només els seus hash en comptes de tots els seus valors.

El DAWG és, per tant:

- **Node arrel**, sense pare ni caràcter inicial
- **Nodes únics**: HashMap<Integer, Node> Per identificar i reutilitzar de manera eficient nodes iguals fent servir el seu hash.



Imatge de representació d'un DAWG de *The World's Fastest Scrabble Program*[3]

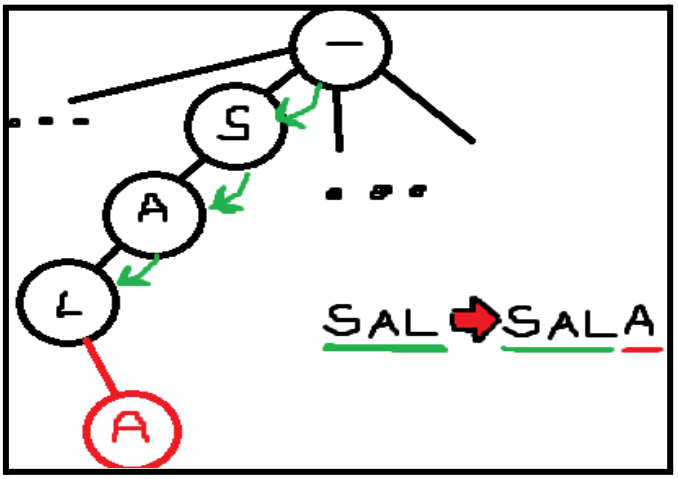
Hi ha 2 maneres de mantenir poc espai quan creem un DAWG (i no fer un *Trie*), anar mirant cada vegada que afegim una nova paraula de no crear un node repetit, o després d'afegir tot el diccionari pronar els nodes repetits. Ens vam decantar per la primera opció.

[1] Appel, A. W., & Jacobson, G. J. (1988, 1 maig). The World's Fastest Scrabble Program.

[2] Daciuk, J., Watson, B. W., Mihov, S., & Watson, R. E. (2000, 6 juliol). Incremental Construction of Minimal Acyclic Finite-State Automata.

[3] Appel, A. W., & Jacobson, G. J. (1988, 1 maig). The World's Fastest Scrabble Program.

Per afegir el nostre diccionari al DAWG ens aprofitem del fet que les paraules venen en ordre. És molt probable que 2 paraules consecutives comparteixin alguna part prefixa entre si, així que recorrem el DAWG fins a arribar a on difereix la nova respecte a l'anterior i comencem a afegir els nous nodes successors des d'allà.



DAWG ón s'acaba d'afegir la paraula SAL i s'aprofita per afegir posteriorment SALA

En aquell moment creem un nou node amb els atributs necessaris que ens farien falta per afegir-lo als successors del node actual (i després continuem per aquest nou creat). Abans d'afegir-lo fem una consulta al HashMap de nodes únics amb el hash del node que hàgim creat per veure si hi havia algun amb el mateix hash, és a dir exactament igual. En cas que si el reutilitzem i no n'afegim un de nou.

En aquell moment com el codi hash dels nodes depèn dels seus successors i hem alterat els successors del node actual en afegir un de nou, així que actualitzem el seu codi hash i propaguem aquesta actualització recursivament al seu pare avançant ascendentment fins a arribar a l'arrel.

Tot i que aquesta propagació i constant actualització dels codis hash sembla molt costosa té el principal *tradeoff* de després poder consultar molt eficientment al HashMap.

Una vegada hem arribat al final de la paraula a afegir al DAWG cal no oblidar-se de marcar l'últim node que sigui final de paraula.

La principal, tot i que no única, funcionalitat del DAWG és verificar si una paraula existeix o no. Per això començant des de l'arrel tractem de navegar tot el DAWG caràcter per caràcter fins a arribar al final. Si no aconseguim arribar al final o l'últim node no és un final de paraula la paraula no existeix.

Anàlisi de costos

El diccionari vindrà ordenat, per tant, no cal considerar una ordenació.

- **Cost d'espai del DAWG:**

Segons ***The World's Fastest Scrabble Program***, la implementació d'un DAWG amb eliminació de nodes redundants permet reduir l'ús de memòria fins a 6 vegades respecte a utilitzar Tries. Donar un cost asimptòtic precís és difícil, ja que depèn dels nodes únics. Aquests variariaran segons quants prefixos té en comú l'idioma a emmagatzemar, així com caràcters únics quant a posició a la paraula i postfixos possibles.

- **Temps de construcció del DAWG: O (n * m * h)**

n = nombre de paraules al diccionari

m = longitud mitjana de paraula

h = altura mitjana del DAWG

Processarem cada caràcter, de cada paraula, amb una actualització de hash recursiva fins a l'arrel.

- **Validar paraula: O(m)** *m = longitud de la paraula*

Cal recórrer els nodes del DAWG i verificar que l'últim sigui final de paraula. En cas que una paraula no sigui válida com no podrem seguir avançant ho detectarem abans.

- **Actualització del hash recursiva: O(h)** h = distància del node modificat fins a l'arrel
- **Consulta HashMap nodes únics: O(1)** Accés a un map

Cal observar que els valors m i h estan acotats per la paraula amb més caràcters de l'idioma actual. Realitzar les actualitzacions de hash recursives és molt més eficient que no pas comparar amb tots els nodes únics actuals. **O(k) > O(h)** k = nodes únics

3.2. Scrabble

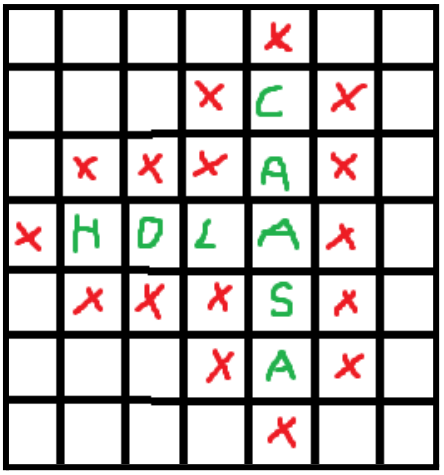
Seguint el paper ***The World's Fastest Scrabble Program***[4] ara explicarem l'algorisme creat als anys 80 pel joc de taula Scrabble i com l'hem implementat.

Abans d'explicar el seu funcionament caldrà explicar les estructures de dades auxiliars que requereix aquest algorisme:

- **DAWG**: Explicat al punt 4.1.
- **Anchors**: Consisteix en un ArrayList<Vector2>. El primer i segon valor d'aquest Vector 2 fan referència a posicions x e y del *Board*.

  La principal funcionalitat aquesta classe és tenir emmagatzemades les posicions sobre les quals la IA pot començar a crear paraules, les caselles anchor. Inicialment, comença sent la casella central del *Board* (inicialitza en el seu updater). Cada vegada que es col·loca un moviment s'actualitza, eliminant les posicions sobre les quals s'ha efectuat el moviment i afegint les que l'envolten.

[4] Appel, A. W., & Jacobson, G. J. (1988, 1 maig). The World's Fastest Scrabble Program.



Les creus vermelles representen els anchors que hi hauria en aquell moment amb aquestes 2 paraules

Tenir aquesta informació emmagatzemada i actualitzada en tot moment és molt més eficient que recalcular-la iterant tot el *Board* cada vegada que la consultem.

Anàlisi de costos

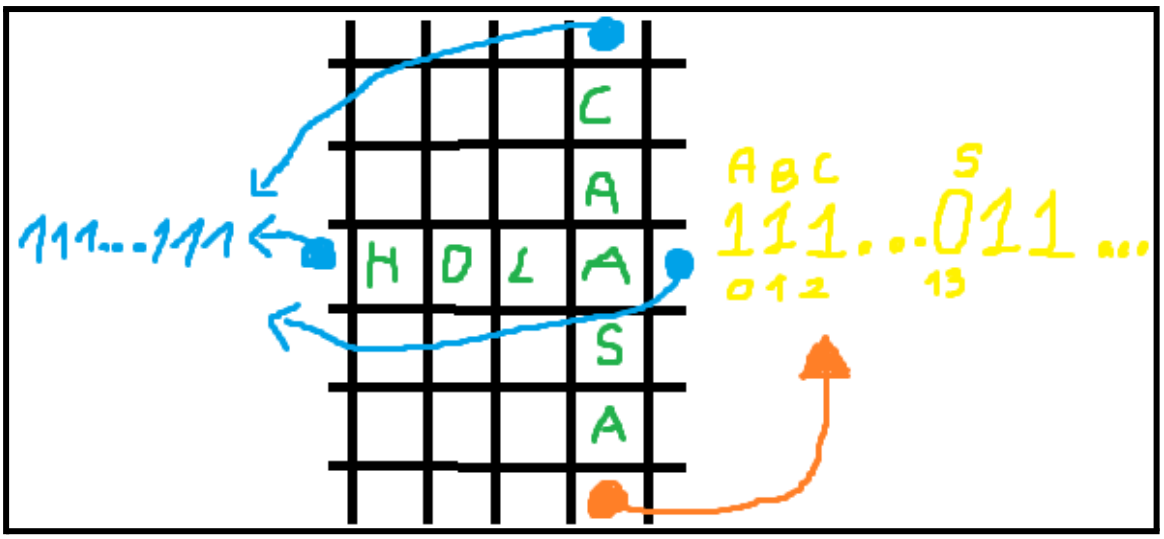
- **Espai: O(s²)** *s = mida tauler*
- **Actualització per moviment: O(c)** *c = nombre de caselles adjacents*

  Aquestes caselles adjacets serán, si totes poden ser considerades (no estan ocupades o fora del tauler) el doble de la mida de les peces col·locades + 2.

- **Consulta conté: O(n)** accés a array list *n = mida array list, n <= s²*

- **CrossChecks**: Consisteix en una matriu de BitSets del size del *Board*. Cada bit representa una *piece*. La mida d'aquest bit set i la *piece* a la qual fan referència depèn de l'idioma seleccionat, per això tenim 3 tipus de crosschecks. Aquesta divisió per idiomes també és útil per mirar els casos de caràcters especials propis de cada idioma.

  El propòsit dels CrossChecks és evitar que la IA iteri sobre moviments que acabin deixant paraules invàlides al *Board* afegint peces al voltant de les ja formades. Al principi tots els BitSets comencen a 0. Cada vegada que es col·loca un moviment s'actualitzen els BitSets de les posicions afectades, posant a 1 els bits els quals fan referència a la peça que, col·locant-la a aquella posició invalidaria a alguna paraula ja formada al *Board*.



En un diccionari ón només tenim HOLA, CASA, CASAS els BitSets dels CrossChecks actuals serien per totes les caselles d'abans/després de cada paraula seran tot 1 a excepció de la S després de CASA ja que es pot formar CASAS

Els CrossChecks no s'actualitzen al voltant de cada paraula, a la casella d'abans de l'inici i després del final (si són vàlides). Les posicions paral·leles a la paraula col·locada, les files de dadalt i a baix (després veurem a la IA perquè 'només col·loquem paraules en horitzontal') ja es tracten a l'algorisme de la IA.

Una possible millora de la implementació perquè l'algorisme sigui una mica més ràpid sería fer que CrossChecks també tinguin emmagatzemada aquesta informació de les files paralel·les.

Tot i poder ser més extensa, igual que els anchors tenir aquesta informació emmagatzemada i actualitzada en tot moment és molt més eficient que recalcular-la iterant tot el *Board* cada vegada que la volem consultar.

Anàlisi de costos

- **Espai: O(s² * b)** *b = logitud bitsets s = mida tauler*
- **Actualització per moviment: O(m)** *m = longitud de paraula formada*

  Només actualitzem anchors a la casella prèvia/següent de paraula (les altres adjacents es comproven dinàmicament a l'algorisme de la IA) i la seva val·lidació dependrà de la mida.

- **Consulta: O(1)** accés directe a matriu → BitSet

- ***The World's Fastest Scrabble Program*:** Una vegada explicades les estructures de dades utilitzades l'algorisme és de backtracking. El seu propòsit és donat l'estat del *Board* actual trobar el moviment que més punts aconsegueixi.

  Els moviments que es poden efectuar essencialment són en horitzontal o vertical. La IA primerament farà una cerca de tots els moviments en horitzontal. Una vegada fet això aplicarem una rotació al *Board* i estructures de dades auxiliars, per fer una segona cerca, novament en horitzontal, tot i que ara realment gràcies a la rotació estarem considerant els moviments en vertical a l'orientació original.

Ens ha semblat més adient realitzar aquesta rotació que no modificar l'algorisme per tenir una còpia pràcticament idèntica, però que es realitzi en vertical. Considerem que els costos de realitzar les rotacions és negligible.

A aquesta segona passada emmagatzemem el millor moviment amb la seva component de direcció com a vertical. Si finalment el millor moviment s'efectua en vertical, aplicarem una transformació a les seves coordenades d'origen, per desfer que es van emmagatzemar com les horitzontals a l'estat amb rotació.

Tot i que no representi un gran impacte globalment, és millor que fer la rotació cada vegada que passi un moviment vertical a ser el que tingui millor puntuació, ja que només ho farem una vegada.

Les cerques amb backtracking consisteixen en, per cada casella anchor disponible tractar d'estendre cap a esquerra i dreta la possible paraula a formar fent servir paraules de la mà i existents al *Board*. Amb l'extensió cap a l'esquerra només es tindran en compte paraules a la *Board* o a la mà, però no les dues a la vegada.

Aquest algorisme és eficient, ja que, tracta d'evitar no començar a provar combinacions que no arribin a una paraula possible. Quan es realitza una extensió com es fa travessant el DAWG podem identificar ràpidament casos ón no val la pena continuar provant combinacions, ja que no hi ha cap manera de continuar avançant a través d'ell i, per tant, aconseguir una paraula vàlida. (també no provem a les cel·les amb CrossChecks les peces que tinguin el seu bit a 1).

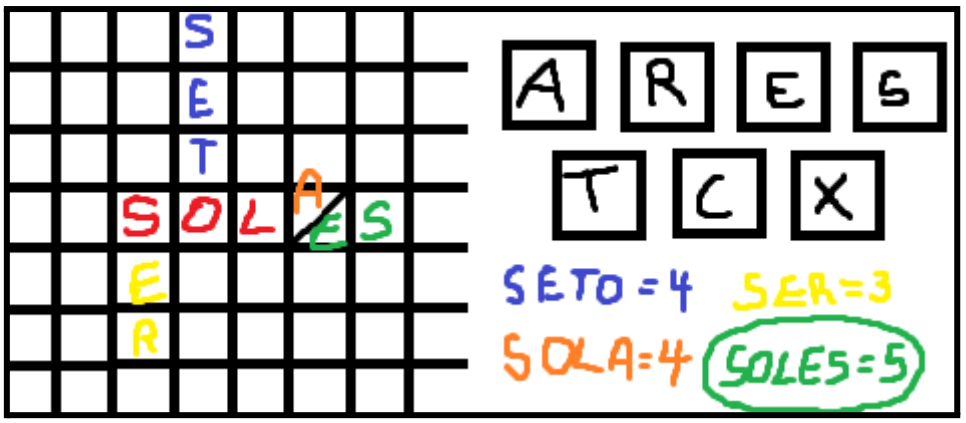
Aquesta manera és molt més ràpida que, per exemple, fer totes les permutacions possibles de paraules que es poguessin formar i després comprovar-les una, ja que quasi totes no ens serviran per a res.

D'aquesta manera també podem anar comprovant si els nodes que travessem quan fem l'extensió cap a la dreta són finals de paraula per veure si és un moviment vàlid i anar actualitzant el millor moviment en cas que sí i aconsegueixi millor puntuació que l'actual.

Al dur a terme proves tot i els nostres tests vam tenir alguns problemes a solucionar. El primer va ser que es realitzaven moviments paral·lels a altres paraules ja col·locades que deixaven paraules invàlides. Com ja s'ha mencionat a CrossChecks ara comprovem manualment aquest fenomen.

Es podria millorar, ja que, ara per ara als idiomes amb peces especials es podria fer millor la comprovació d'aquestes paraules paral·leles, ja que, al cas concret de detectar que podrien causar descartem completament el moviment actual, perdent alguns possibles moviments.

L'altre va ser que la IA retornava moviments que eren tot paraules ja col·locades al *Board*. Una solució ràpida i poc costosa va ser guardar-se quantes peces tenia el jugador bot a l'inici del seu torn (no podia ser 7, ja que, als torns finals potser no hi ha prou peces i tenim menys de les habituals), i mirar al tractar de validar el moviment si havia baixat les actuals de la quantitat original.



Exemple de quines paraulas considearaaria la IA amb un diccionari en castellà i sense considerar caselles especials que millorin els punts. Amb SOL al tauler i A,R,E,S,T,C,X peces a la mà. Acabarà escollint la jugada que crearà SOLES perquè és la que genera més punts.

Una observació a realitzar sobre el comportament de la IA és que per la seva naturalesa *greedy* només tindrà en compte la millor jugada al seu torn sense pensar una estratègia defensiva o que tingui en compte als altres jugadors. Un exemple d'això seria que no tindria cap dubte en col·locar una jugada que generi la paraula helicòpter, sense tenir en compte que un jugador amb la lletra s al següent torn podria formar helicòpters.

Anàlisi de costos

Aquest anàlisi és pot realitzar de manera molt més detallada. El propòsit actual és donar una visió generalitzada.

- **Extensió esquerra: O(e ^ r)**

  e = caselles disponibles per estendre a l'esquerra

  r = rack peces disponibles (comodí compta com totes)

- **Extensió dreta: O(d ^ r)**

  d = caselles disponibles per estendre a la dreta

  r = rack peces disponibles (comodí compta com totes)

- **Validació de paraula: O(n*m)**

  m = longitud de paraules afectades al realitzar el moviment

  n = número de paraules afectades al realitzar el moviment

  Aquest cost acaba sent poc rellevant respecte el total.

- **Cerca completa: O(a*(e*d)^r)** *a = anchors disponibles*

El cost temporal serà imperceptible en temps d'execució, a excepció de quan al rack de peces de la mà tinguem 2 comodins i una quantitat anchors superior a l'inici de la partida.

4. Estructures de dades

4.1. Gestió d'escenes: SceneManager, Scene i SceneObject

Per gestionar l'estat del joc, hem organitzat les diferents parts del codi en escenes separades. En aquesta primera entrega, el joc disposa de dues escenes principals: l'escena de **menú** i l'escena de **joc**.

4.1.1. SceneManager

La classe *SceneManager* s'encarrega de controlar el bucle principal del programa, així com de gestionar el canvi d'escenes. Això inclou carregar noves escenes i descarregar les anteriors. Per fer-ho possible, *SceneManager* fa ús de dues classes fonamentals: *Scene* i *SceneObject*.

4.1.2. Scene

La classe *Scene* és una **classe abstracta** que proporciona la funcionalitat bàsica per gestionar els objectes d'una escena, com la seva **creació**, **actualització** i **destrucció**.

Cada nova escena ha d'heretar de *Scene*, i dins del seu constructor s'instancien tots els objectes que han d'estar presents mitjançant el mètode *instantiate()*, proporcionat per la mateixa classe base.

4.1.3. SceneObject

La classe *SceneObject* és també una **classe abstracta** que representa un objecte actiu dins d'una escena. Cada objecte pot implementar diversos callbacks per respondre als canvis d'estat i a l'evolució del joc. Els mètodes més rellevants són:

- **constructor()**: No pot tenir paràmetres. Es crida durant la instanciació de l'objecte.
- **onProcess(delta)**: S'executa en cada iteració del bucle principal. El paràmetre *delta* indica la diferència de temps respecte a l'última execució.
- **onEnable()**: S'executa quan l'objecte és activat. A partir d'aquest moment, rebrà actualitzacions a través de *onProcess(delta)*.
- **onDisable()**: Es crida quan l'objecte és desactivat.
- **onDetach()**: S'executa quan l'objecte és eliminat de l'escena.

A més, SceneObject ofereix mètodes com *instantiate()* i *destroy()* per facilitar la creació i eliminació d'objectes dins de l'escena.

4.2. Lectura de diccionaris i fitxes: DictionaryReader i PiecesReader

Per a la lectura dels diccionaris i peces hem decidit fer-ho en temps d'execució. Per gestionar aquesta funcionalitat de manera organitzada, s'han definit tres classes. Aquestes s'encarreguen de localitzar i llegir els fitxers corresponents segons l'idioma seleccionat.

4.2.1. LocaleReader

Aquesta és la classe base de la qual hereten les altres. El seu propòsit és proporcionar una funcionalitat genèrica per localitzar i llegir fitxers del sistema, situats dins d'un directori específic del projecte (*locales*).

Conté dos mètodes principals:

- **getAbsolutePath(String fileName)**: Construeix la ruta absoluta cap a un fitxer donat, assumint que es troba dins de la jerarquia del paquet *edu/upc/prop/scrabble/locales*.
- **readFileToString(String filePath)**: Llegeix i retorna el contingut del fitxer en format String amb codificació UTF-8. Si hi ha un error, retorna null.

Aquesta classe no s'instancia directament, sinó que serveix com a base perquè altres classes en puguin reutilitzar el comportament de lectura.

4.2.2. PiecesReader

Aquesta classe hereta de *LocaleReader* i s'encarrega de llegir els fitxers que contenen les peces (fitxes amb lletres i punts) que s'utilitzen en el joc.

Implementa un mètode públic *run(Language locale)* que selecciona el fitxer corresponent segons l'idioma:

- **Catalan** → letrasCAT.txt
- **Spanish** → letrasCAST.txt
- **English** → letrasENG.txt

Amb això, permet obtenir fàcilment les peces apropiades per a cada idioma, garantint que el joc pugui adaptar-se correctament a diferents localitzacions.

4.2.3. Pieces

També derivada de *LocaleReader*, aquesta classe està enfocada a llegir els fitxers de diccionaris, que contenen les paraules vàlides per a cada idioma del joc.

Té el mètode *run(Language locale)*, que retorna el contingut del fitxer corresponent:

- **Catalan** → catalan.txt
- **Spanish** → castellano.txt
- **English** → english.txt

4.3. Emmagatzematge i gestió de puntuacions: Score, Leaderboard i controladors

Per guardar tota la informació relacionada amb la puntuació i resultats de les partides i gestionar com mostrar-la disposem de dues classes a la capa de dades i diversos controladors a la capa de domini.

4.3.1. Score

Score és la classe més bàsica en aquest àmbit. Aquesta és un *public record* que guarda la puntuació obtinguda, si el jugador ha guanyat la partida i el seu nom.

4.3.2. Leaderboard

El següent esglaó en aquesta jerarquia és la classe leaderboard. Encarregada de guardar *Score* de diverses partides. Per aquest emmagatzematge hem utilitzat una ArrayList, que permet el creixement del nombre d'entrades gràcies a la seva gestió de memòria dinàmica.

4.3.3. Controladors

Per la gestió de com mostrar aquesta informació a l'usuari teníem diverses vies. Podíem emmagatzemar en diferents classes els *Score* amb els criteris adients. Això faria que cada vegada que es completa una partida s'hagués d'actualitzar cadascuna de les diferents classes. A més d'això molta de la informació emmagatzemada estaria duplicada. Com el volum de dades a processar mai serà molt gran vam optar per la via de crear controladors pels diferents mètodes de filtratge que processen les dades emmagatzemades a la *Leaderboard* i les retorni de la forma desitjada. D'aquesta manera podem guardar la informació relacionada a cada partida de forma senzilla i només operar quan es demana. A més aquesta metodologia facilita la inclusió de filtratges extra o la introducció de variables extra al record *Score,* el que va molt agafats de la mà amb la filosofia amb la qual afrontem aquest projecte. Els mètodes de filtratge actualment són:

- **GamesPlayedLeaderboard:** retorna una array dels noms dels jugadors amb el seu nombre de partides jugades ordenada de forma descendent per aquesta quantitat.
- **GamesWinLeaderboard:** retorna una array dels noms dels jugadors amb el seu nombre de partides guanyades ordenada de forma descendent per aquesta quantitat.
- **MaxScoreLeaderboard:** retorna una array dels noms dels jugadors amb la seva maxima puntuació obtinguda ordenada de forma descendent per aquesta quantitat.
- **TotalScoreLeaderboard:** retorna una array dels noms dels jugadors amb la suma de les puntuacions obtingudes ordenada de forma descendent per aquesta quantitat.
- **WinRateLeaderboard:** retorna una array dels noms dels jugadors amb el seu percentatge de victòries ordenada de forma descendent per aquest percentatge.

Per ajudar l'execució d'aquest controlador disposem de dues classes auxiliars:

- **PlayerValuePair:** classe que guarda el nom del jugador i la variable a gestionar en format double. L'array que retornen els controladors és d'objectes d'aquesta classe.
- **GamesWinsPair:** classe que guarda el nombre de partides jugades i guanyades. Sempre està relacionada amb el nom d'un jugador i només s'utilitza per al controlador de percentatge de victòries.

4.4. Emmagatzematge dels moviments

Per emmagatzemar les dades més rellevants d'un moviment seguim el format estàndard abreujat del Scrabble[5]. Per a poder entendre millor aquest format, donem un parell d'exemples:

- **HOLa 2A**: En aquest cas, es col·locarà la paraula "HOLA" a la posició (2, A) en horitzontal. La lletra "A" d'aquest moviment és un escarràs.
- **ADEU F4**: En aquest cas, es col·locarà la paraula "ADEU" a la posició (F, 4) en vertical. No s'utilitza cap escarràs en aquest moviment.

Tal com es pot veure quan la primera coordenada del moviment és un número, aquest s'interpretarà com a moviment en horitzontal, en canvi, si la primera coordenada és una lletra, aquest s'interpretarà com un en vertical.

Els punts més importants d'aquest format són que els **escarrassos s'escriuen en minúscula** i que en la paraula del moviment **es representa la paraula sencera** i no les peces individuals a col·locar. En el moment d'efectuar el moviment, aquests dos factors es tenen en compte per seleccionar les peces que utilitza el jugador.

Internament, els moviments s'emmagatzemen en el record *Movement*. Aquest record conté la següent informació:

- **word**: String amb la paraula seguint el format explicat anteriorment.
- **x**: Enter amb la coordenada X de l'inici del moviment.
- **y**: Enter amb la coordenada Y de l'inici del moviment.
- **direction**: Direcció en la qual s'efectua el moviment (Horitzontal o Vertical)

Tal com es pot veure, aquí no utilitzem les coordenades amb el format abreujat del Scrabble, aquest format d'entrada únicament s'utilitza en la versió de la **vista per terminal**. Això és degut al fet que aquest format d'entrada de les coordenades únicament ens és útil en aquesta primera entrega.

4.5. Persistència de dades

Per a implementar la capa de persistència, hem dissenyat un sistema completament **aïllat de la lògica de negoci** del projecte de Scrabble. En altres paraules, aquest sistema s'ha concebut de manera **genèrica i reutilitzable**, de manera que podria integrar-se en altres projectes sense necessitat de dependre de cap funcionalitat específica del domini del Scrabble.

Aquesta independència s'ha pogut assolir gràcies a la **separació de responsabilitats** establerta des de la primera entrega. En aquell moment, ja vam definir una divisió clara entre la **capa de domini**, on resideixen els controladors i la lògica de negoci, i la **capa de dades**, encarregada de representar l'estat actual del sistema. Aquesta estructura ens ha permès introduir la persistència sense trencar l'arquitectura ja existent.

El nucli del sistema de persistència es basa en una interfície anomenada **IPersistableObject**, que defineix el contracte que han de complir tots els objectes que poden ser persistits. Aquesta interfície conté dues operacions fonamentals:

[5] *Scrabble/Rules*. (2024, 22 agost). Wikibooks. https://en.wikibooks.org/wiki/Scrabble/Rules#Example

- **PersistentDictionary encode()**
  Aquest mètode ha de ser implementat per cada classe persistible i s'encarrega de transformar l'estat intern de l'objecte en un PersistentDictionary, una estructura clau-valor dissenyada per representar dades de manera serialitzable.
- **void decode(PersistentDictionary data)**
  Aquest mètode fa la inversa: reconstrueix l'estat intern de l'objecte a partir d'un PersistentDictionary prèviament generat pel mètode encode().

Amb aquest disseny, el funcionament del sistema de persistència es pot resumir de la següent manera:
Una classe central s'encarrega de coordinar el procés de **guardat**, demanant a tots els objectes que implementen IPersistableObject la seva representació serialitzada mitjançant el mètode encode(). Un cop obtingudes, aquestes dades es guarden a un arxiu mitjançant l'ús de la biblioteca Gson. En el procés de **càrrega**, aquest mateix sistema llegeix el fitxer emmagatzemat, reconstrueix els PersistentDictionary, i invoca el mètode decode() en cada objecte per restaurar-ne l'estat original.

Aquest enfocament ens permet mantenir un sistema **modular, desacoblat i fàcilment testejable**, en què les classes persistibles tenen autonomia per definir com han de guardar i recuperar el seu estat, mentre que la capa de persistència gestiona el procés global de forma genèrica.