TEC | Tecnológico de Costa Rica

INSTITUTO TECNOLÓGICO DE COSTA RICA

PROGRAMACIÓN ORIENTADA A OBJETOS

SCRABBLE

Por:
Danilo Duque Gómez
Pablo Pérez Delgado
Jose Emmanuel Rojas Soto

Profesor:
Andrés Víquez Víquez

23 de noviembre de 2023

1. Introducción

En el contexto del curso de Programación Orientada a Objetos, se planteó la tarea de llevar a cabo la implementación completa del reconocido juego de mesa Scrabble. Este juego, centrado en la formación de palabras en un tablero mediante fichas con letras y valores específicos, demanda una combinación de destreza lingüística y estrategia.

Desarrollado por Alfred Butts en la década de 1930, el Scrabble ha alcanzado el estatus de clásico, desafiando la creatividad y el conocimiento del vocabulario de los participantes. La premisa básica consiste en formar palabras en el tablero utilizando las letras de las fichas, cada una con un valor asignado.

La implementación de esta solución se abordó desde la perspectiva de la Programación Orientada a Objetos (POO). Este paradigma, basado en la encapsulación de datos y funciones en objetos interconectados que se comunican mediante mensajes, proporciona una estructura modular y favorece la reutilización de código. La elección de la POO busca optimizar el desarrollo, haciéndolo más eficiente y sostenible a largo plazo.

No obstante, cabe preguntarse: ¿por qué se optó específicamente por la Programación Orientada a Objetos para la resolución de este proyecto?

2. Estrategia de la solución

La solución implementada para el juego de mesa Scrabble se diseñó siguiendo el paradigma de programación orientada a objetos. La elección de este enfoque se basa en su capacidad para representar de manera efectiva los elementos del mundo real inherentes al juego. Este método resulta especialmente adecuado para modelar las complejas interacciones y características del Scrabble, facilitando así la resolución lógica del programa.

Al adoptar objetos que representan las distintas partes del juego en la realidad, se logra una simplificación significativa en la implementación. Cada componente del juego se aborda como una entidad independiente, mejorando la estructura y organización del código. Esta modularidad no solo facilita el mantenimiento, sino que también permite una escalabilidad más eficiente del sistema al introducir nuevas funcionalidades o realizar modificaciones en el futuro.

En consonancia con esta estrategia, se implementó una distribución de tareas dentro del equipo de tres personas. Un miembro se encargó del diseño del software y la gestión del equipo, otro se centró en la lógica del programa, y el tercero se ocupó de la interfaz de usuario. Esta división especializada maximizó la eficiencia y calidad del trabajo, fomentando una clara comunicación y colaboración entre los integrantes. Cada miembro comprendió su responsabilidad y contribución al objetivo general del proyecto.

Además, se incorporaron controladores de versión, como GitHub, para facilitar la gestión y seguimiento de los cambios en el código. Este enfoque proporcionó un mejor control y organización del desarrollo del programa, permitiendo la colaboración simultánea en diferentes aspectos del proyecto sin interferencias. La implementación de controladores de versión también ofreció la capacidad de realizar rollbacks en caso de errores, garantizando la estabilidad del proyecto y permitiendo un desarrollo seguro y confiable. En conjunto, estas estrategias posibilitaron un entorno de trabajo más eficaz y resistente, minimizando la pérdida de trabajo debido a posibles contratiempos durante el desarrollo.

3. Detalles de la implementación

La fase de diseño, programación e implementación consto de dos partes importantes:

3.1. Implementación de Lógica

Con el propósito de alcanzar el objetivo global, se inició la tarea de investigar la implementación del diccionario durante el juego. Diversas propuestas fueron consideradas en este proceso, incluyendo la creación de una lista de cadenas (Strings) que abarcara todas las palabras, la opción de acceder directamente al archivo .txt para buscar las palabras, y, finalmente, se examinó una idea particular que consistía en incorporar el diccionario en una estructura de datos conocida como Árbol Trie.

El uso del árbol Trie se presenta como la solución óptima para gestionar diccionarios de considerable magnitud, gracias a su eficacia en la búsqueda de palabras. Para resaltar la importancia del Trie, es fundamental comprender el funcionamiento del algoritmo encargado de encontrar una palabra en una lista de cadenas de texto. El mencionado algoritmo opera de la siguiente manera:

```
1
2  public boolean search(ArrayList<String> lista,String palabra) {
3
4      for (String s : lista) {
5          boolean encontrada = true;
6          for (int i = 0; i < palabra.length(); ++i) {
7              if (i>=s.length() || palabra.charAt(i)!=s.charAt(i))
8              {
9                  encontrada = false;
10                 break;
11             }
12
13         }if (encontrada) return true;
14
15     }return false;
16 }
```

Listing 1: Algoritmo para encontrar una palabra en una lista

Al examinar la complejidad del algoritmo mencionado anteriormente, se evidencia que en el peor de los casos, el programa recorrerá las n palabras del diccionario. Suponiendo que cada palabra tiene una longitud de m letras, el número total de operaciones en el peor escenario se aproxima a $O(n \cdot m)$.

La incorporación del Trie se presenta altamente beneficiosa en este contexto, ya que esta estructura permite determinar en $O(n)$ operaciones, donde n representa la longitud de la palabra, si dicha palabra está presente en el diccionario. En resumen, el Trie optimiza significativamente la eficiencia de la búsqueda de palabras en comparación con el algoritmo previo, agilizando el proceso de manera considerable. Este rendimiento se debe a la capacidad del Trie para organizar y estructurar la información de manera eficiente, facilitando la búsqueda y recuperación de datos de manera más rápida y efectiva.

Después de realizar este análisis, el siguiente paso fue incorporar el Trie en el código para que pudiera almacenar el diccionario. Para lograr esto, se crearon tres clases, las cuales se describirán a continuación:

- Clase Node:

  La clase 'Node' fue diseñada para la construcción del Trie, ya que cada nodo del árbol requiere ser representado por una clase. Esta clase incluye atributos esenciales como un carácter que representa la letra asociada al nodo, y un arreglo de nodos (de tamaño 27) que representa los posibles 27 descendientes desde el nodo actual.

- Clase TrieTree

  La clase 'TrieTree' se encarga de gestionar la búsqueda e inserción de datos en el árbol Trie. Su utilidad radica en encapsular todo el funcionamiento del Trie, simplificando la visibilidad y comprensión desde el exterior. Esta clase ofrece dos métodos públicos, 'buscar' e 'insertar', que facilitan la interacción y operación con la estructura del Trie de manera sencilla.

- Clase DictionaryReader

  La clase DictionaryReader desempeña un papel fundamental en la lectura del archivo dictionary.txt y la creación de un árbol Trie con todas las pa-

labras insertadas. Esta clase encapsula de manera eficiente el proceso de construcción del Trie a partir del diccionario, proporcionando una interfaz simplificada para el manejo de esta operación.

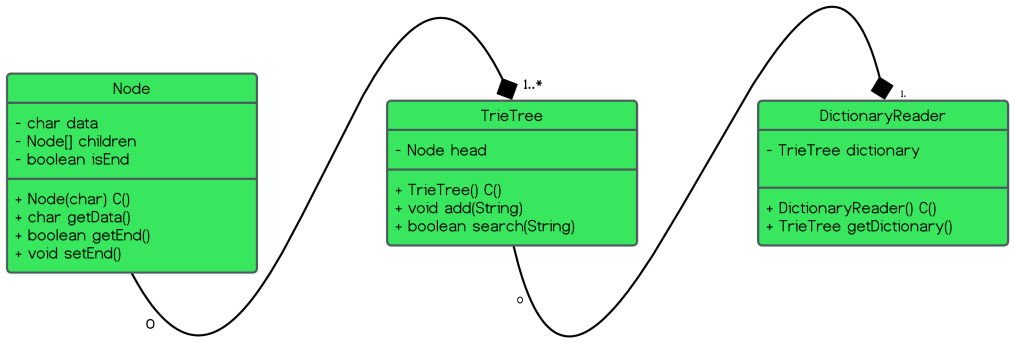

Figura 1: Diagrama que representa la interacción de las clases Node, TrieTree y DictionaryReader

Tras el estudio de la implementación del diccionario, se procede con la siguiente etapa de la lógica del programa: la modelización del juego Scrabble utilizando Programación Orientada a Objetos.

En una primera instancia, para lograr una representación abstracta de todos los elementos del juego Scrabble, se comienza abstrayendo la información más elemental del juego: las fichas utilizadas en el mismo. La representación completa de estas fichas en el software se logra mediante la inclusión de tres atributos esenciales: un carácter que refleja la letra contenida en la ficha, un entero que representa el puntaje asociado a la ficha y, finalmente, un valor booleano que indica si la ficha en cuestión es un comodín o no.

Con esta información elemental del programa, se inicia la construcción de los componentes fundamentales sobre los cuales operará el software. El primer bloque abordado y implementado es el tablero de juego. Para facilitar la implementación y promover la reutilización del código, se ha creado el tablero utilizando 225 objetos de la clase Casilla. Esta clase se encarga de gestionar los datos de cada posición del tablero, incluyendo la información sobre si hay una ficha en una casilla determinada. Además, esta clase desempeña un papel crucial al asignar puntajes a

distintas casillas, considerando las reglas del juego que implican multiplicadores en ciertas posiciones del tablero.

La clase Casilla está compuesta, entre otros elementos, por la clase Ficha. Uno de sus atributos es la ficha que ocupa la casilla en cuestión. Además, el tablero cuenta con métodos encargados de verificar si una jugada es válida.

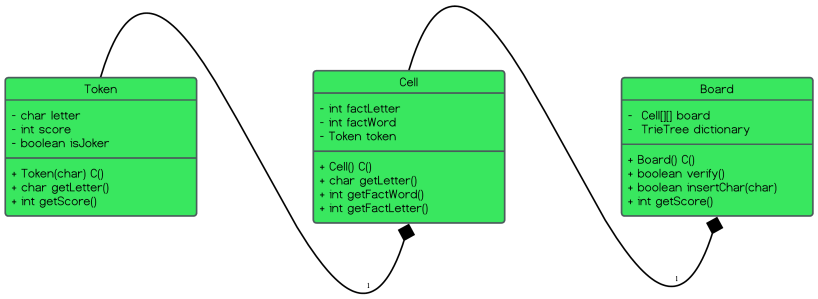

Figura 2: Diagrama que representa la interacción de las clases Token(Ficha), Cell(Casilla) y Board(Tablero)

Con las clases que representan de manera integral un tablero, se avanza en la creación de las clases que definirán el comportamiento de un jugador. Para lograr esto, se han desarrollado tres clases:

- Clase Sack:

  La clase 'Saco' está compuesta por 100 objetos Ficha y tiene la función de almacenar todas las fichas del juego. Como se detallará más adelante, es accedida por los objetos de la clase 'Player' al construirse, ya que cada jugador toma 7 fichas del saco al inicio de la partida. Además, esta clase se encarga de rellenar los atriles de los jugadores después de cada jugada, así como de barajarse cuando sea necesario.

- Clase Deck:

  Los objetos representados por la clase 'Atril' desempeñan la función de almacenar las fichas asignadas a cada jugador. Además, actúan como intermediarios en la comunicación entre el jugador y el saco de fichas. Cuando

un jugador desea cambiar ciertas fichas, solicita al atril que realice la operación correspondiente con el saco. Esta interacción se diseñó de manera que el jugador se mantenga lo más alejado posible del saco de fichas, garantizando así la aleatoriedad en cada acceso al mismo.

- Clase Player:

  La clase 'Player' en el contexto del juego Scrabble se define como una entidad que representa a un participante en la partida. Posee tres atributos fundamentales para su representación completa: un String llamado 'name' que almacena el nombre del jugador, un entero 'score' que registra el puntaje acumulado durante la partida y un Deck denominado 'atril', que funciona como el conjunto de fichas asignadas al jugador. La interacción del jugador con el saco de fichas se lleva a cabo a través del 'atril', permitiendo al jugador modificar su conjunto de fichas según las reglas del juego. Este diseño asegura una gestión eficiente y coherente de la información esencial de cada jugador en el desarrollo del juego.

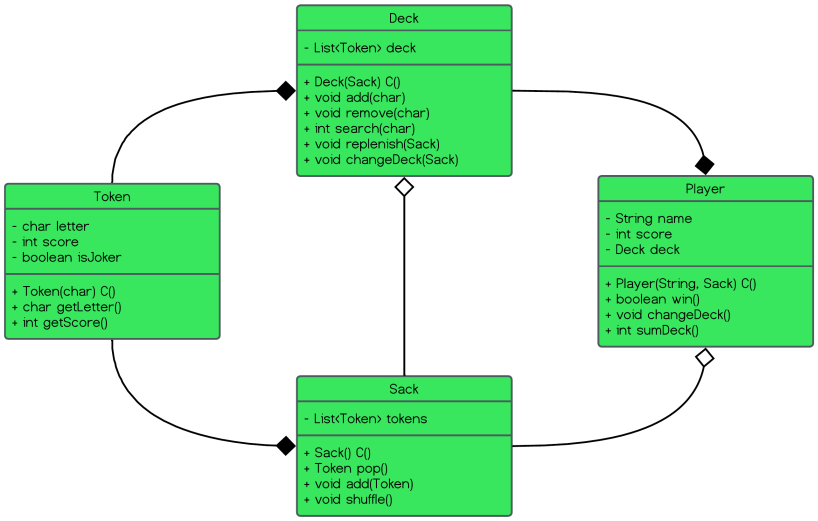

Figura 3: Diagrama que representa la interacción de las clases Token(Ficha), Sack(Saco), Deck(Atril), y Player(jugador)

Con la totalidad de la lógica implementada, se introduce la clase 'Game' para coordinar las interacciones entre las demás clases y seguir una secuencia de juego coherente en el contexto de Scrabble. Esta clase posee tres atributos clave: una lista de jugadores que contiene los n participantes en la partida actual, un objeto 'tablero' responsable de rastrear las posiciones de las fichas colocadas por los jugadores y, finalmente, un atributo de tipo 'Saco' encargado de gestionar el almacenamiento de fichas para que los jugadores puedan tomarlas cuando sea necesario.

Esta clase será únicamente gestionada por el 'Main', ya que actúa como la entidad principal que dirige el flujo del juego. El 'Main' inicializa una instancia de 'Game', configura el número de jugadores, y establece las conexiones con el tablero y el saco de fichas. Una vez que 'Game' está instanciada, se encarga de orquestar el desarrollo de la partida, gestionando el turno de cada jugador, verificando la validez de las jugadas, actualizando el estado del tablero y el saco de fichas, y llevando un registro de los puntajes acumulados.

Finalmente, se implementó una clase dedicada a la gestión de los puntajes de los jugadores al concluir una partida. Para esta funcionalidad, se diseñó la clase 'Archive', basada en un HashMap. La elección de esta estructura de datos se fundamenta en su eficiencia para manipular información asociativa, ya que permite almacenar pares clave-valor de manera rápida y acceder a ellos de forma eficiente.

La clase 'Archive' se encarga de mantener un registro actualizado de los puntajes acumulados por cada jugador a lo largo de múltiples partidas. Al concluir una partida, se realiza una actualización mediante el método 'update', que suma los puntajes obtenidos por los jugadores en dicha partida a sus puntuaciones totales. Este enfoque asegura la persistencia de los datos y proporciona un historial completo de desempeño para cada jugador.

Además, el uso de un HashMap permite una flexibilidad significativa, ya que se pueden manejar fácilmente nombres de jugadores dinámicamente, y se proporciona una interfaz eficaz para consultar y modificar los puntajes acumulados en cualquier momento.

3.2. Implementación de la Interfaz Gráfica

La fase de implementación de la interfaz gráfica del juego Scrabble se llevó a cabo utilizando la biblioteca Java Swing como base. Esta elección se fundamentó en la versatilidad y robustez que Swing proporciona para la construcción de interfaces de usuario. Cada componente visual, desde el tablero de juego hasta los atriles de los jugadores, se representó mediante objetos Swing, con un énfasis particular en el uso de JButtons y JLabels para interactuar con los usuarios.

En el diseño visual, se hizo hincapié en la disposición estratégica de los elementos dentro de la ventana del juego para asegurar una experiencia de usuario intuitiva y estéticamente agradable. Se utilizaron paneles y layouts de Swing para organizar eficientemente los componentes, logrando así una interfaz coherente y fácil de seguir. La elección de colores y la inclusión de iconos específicos se realizaron con el objetivo de mejorar la identificación visual y la estética general del juego.

La gestión de eventos fue fundamental en la implementación. Cada elemento interactivo se asoció a un manejador de eventos que respondía a acciones específicas del usuario. Por ejemplo, al hacer clic en un JButton en la cuadrícula del tablero, se activaba un evento que verificaba la legalidad de la jugada y actualizaba la interfaz en consecuencia.

La actualización dinámica de la interfaz fue una característica crucial. Después de cada jugada, la interfaz gráfica se actualizaba para reflejar los cambios en el estado del juego. Los métodos desarrollados para este propósito garantizaron una representación visual precisa y en tiempo real del tablero y de los atriles de los jugadores.

El diseño de clases adoptó un enfoque de Programación Orientada a Objetos (POO). Se crearon clases que representaban las entidades principales del juego, como Jugador, Tablero y Bolsa de Fichas, estableciendo relaciones claras entre ellas. Este diseño facilitó la modificación y expansión del juego en etapas posteriores del desarrollo.

4. Diagrama de clases

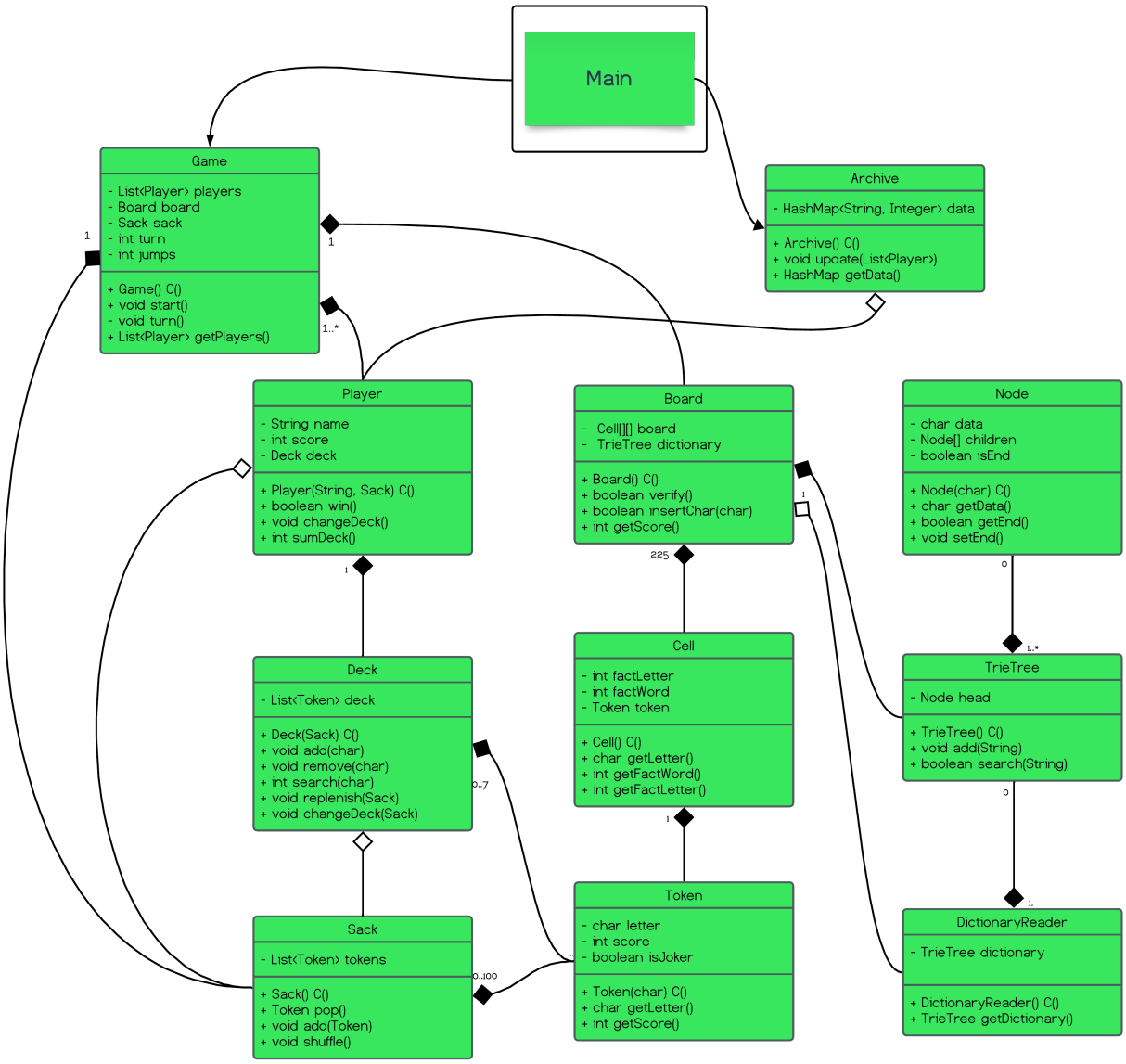

Figura 4: Diagrama de clases, usando POO para la realización del Scrabble

5. Restricciones y suposiciones

Al desarrollar el programa para el juego de Scrabble, se han tenido en cuenta diversas restricciones y suposiciones para garantizar un diseño coherente y eficiente. Estas consideraciones son esenciales para el correcto funcionamiento del programa y la experiencia del usuario. A continuación, se detallan algunas de las restricciones y suposiciones adoptadas:

5.1. Restricciones

- Tamaño del Tablero: El tablero de juego se limita a una dimensión específica, en este caso, se ha considerado un tablero estándar de dimensiones 15x15, de acuerdo con las normas del Scrabble.
- Número de Jugadores: Se establece un límite en el número de jugadores para simplificar la interacción y la dinámica del juego. En este caso, se asume que el juego admitirá de 2 a 4 jugadores.
- Palabras Válidas: El programa se basa en un diccionario predefinido que limita las palabras válidas en el juego. Solo se considerarán palabras presentes en este diccionario.

5.2. Suposiciones

- Interfaz de Usuario: Se supone que el programa contará con una interfaz gráfica intuitiva y fácil de usar para mejorar la experiencia del usuario. Se asume que los jugadores interactuarán principalmente a través de esta interfaz.
- Inicio de Partida: El juego comienza con la distribución aleatoria de fichas a cada jugador desde un saco central. Se asume que al inicio de la partida, todos los jugadores tendrán sus atriles llenos con las fichas correspondientes.
- Puntajes y Fin de Partida: El juego seguirá un sistema de puntuación estándar del Scrabble, y se considera que la partida concluye cuando se cumplen ciertas condiciones específicas, como cuando se agotan las fichas en el saco central o cuando un jugador alcanza un puntaje máximo predeterminado.

6. Problemas encontrados

Durante el proceso de diseño y codificación del juego Scrabble, se abordaron dos desafíos esenciales que exigieron una atención detenida para lograr la plenitud del programa.

El primer desafío, vinculado a la gestión del diccionario, presentó una complejidad adicional debido a la presencia de caracteres especiales. La fase de limpieza del diccionario se reveló como un paso crítico para que el programa pudiera reconocer únicamente palabras válidas. La elección de emplear un árbol Trie para optimizar los tiempos de búsqueda fue producto de una investigación exhaustiva, considerando otras alternativas antes de determinar la solución más eficaz. La implementación de esta estructura demostró ser esencial para el manejo eficiente del diccionario, destacando la importancia de la investigación y la elección de estructuras de datos adecuadas en el desarrollo de software.

En la fase de diseño y codificación, el segundo desafío se centró en los métodos de verificación de jugadas válidas, que, a pesar de su aparente simplicidad, requerían un estudio profundo para garantizar su confiabilidad, especialmente al incorporar el manejo de comodines. El análisis detallado de estrategias permitió encontrar una solución que prescindiera de almacenar el primer carácter de la palabra en memoria. Esta solución, que verifica todas las casillas del tablero retrocediendo hasta el inicio de la palabra, se destacó por su eficacia en pruebas con diversas palabras y comodines. Resalta la importancia de la flexibilidad en el diseño para adaptarse a la libertad del usuario al insertar palabras en el orden deseado.

En conclusión, estos desafíos ilustran la complejidad inherente a la implementación de un juego como el Scrabble, subrayando la importancia de la investigación, la elección de estructuras de datos adecuadas y la adaptabilidad del diseño para garantizar la funcionalidad y confiabilidad del programa. Este proceso evidencia la necesidad de considerar cuidadosamente cada componente del juego, desde la gestión del diccionario hasta la validación de jugadas, para lograr un producto final coherente y eficiente.

7. Conclusiones y recomendaciones

El desarrollo del juego Scrabble bajo el paradigma de Programación Orientada a Objetos (POO) ha permitido crear una implementación modular, eficiente y estructurada. La elección de la POO facilitó la representación de los elementos del juego de manera coherente, mejorando la mantenibilidad y escalabilidad del código. La división de tareas dentro del equipo, especializándose en el diseño del software, la lógica del programa y la interfaz de usuario, contribuyó a una colaboración eficaz y a la calidad del trabajo.

La gestión del diccionario a través de un árbol Trie demostró ser una solución óptima, mejorando significativamente los tiempos de búsqueda y la eficiencia en la identificación de palabras válidas. La implementación de estructuras como el Trie y el uso de controladores de versión contribuyeron a un desarrollo más robusto y controlado.

En cuanto a la lógica del juego, la representación de fichas, casillas y jugadores como objetos permitió un modelado fiel del Scrabble. La interacción entre estas clases se diseñó cuidadosamente para garantizar un flujo de juego coherente y mantener la integridad de las reglas del juego.

Asimismo, se ha constatado la imperante necesidad de una interfaz gráfica optimizada, dado que esta sección del software representa el principal punto de interacción con el usuario. En el caso de no proporcionar una interfaz intuitiva y estéticamente agradable, podría generar la percepción en el usuario de que el programa aún no ha alcanzado su totalidad. La impresión inicial que un usuario obtiene al interactuar con la interfaz gráfica es crucial para la aceptación general del software, ya que influye directamente en la percepción de su completitud y usabilidad. Por ende, la continua mejora y refinamiento de la interfaz gráfica resulta esencial para garantizar una experiencia positiva y satisfactoria para los usuarios, fortaleciendo así la impresión de un producto final completo y bien desarrollado.

Algunas de las recomendaciones que el equipo podría dar son las siguientes:

- **Documentación del Código:**
  - Incluir comentarios claros y ejemplos de uso en el código para facilitar la comprensión.
  - Utilizar herramientas de documentación automática como Javadoc o Doxygen.
- **Eficiencia del Programa:**
  - Realizar análisis de rendimiento y optimizar algoritmos y estructuras de datos.
  - Considerar técnicas de programación concurrente si es necesario.
- **Intuitividad de la Interfaz:**
  - Conducir pruebas de usabilidad y mejorar elementos visuales y de diseño.
  - Optimizar el flujo de trabajo del usuario para una experiencia más intuitiva.
- **Estudio Exhaustivo de Componentes del Software:**
  - Realizar revisiones detalladas de cada componente para identificar dependencias y áreas propensas a errores.
  - Documentar las relaciones entre módulos y componentes, implementando pruebas unitarias específicas.

Referencias

[1] Hasbro, 'Scrabble Game Rules and Instructions' , Hasbro Instructions. [Online]. Available: https://instructions.hasbro.com/en-us/instruction/Scrabble-Game. [Accessed: Nov. 18, 2023].

[2] Oracle, 'Java Software,' Oracle.com, 2019. [Online]. Available: https://www.oracle.com/java/.

[3] Oracle, 'javax.swing (Java Platform SE 8),' docs.oracle.com. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html.

[4] sambhav228, 'Introduction of Object Oriented Programming,' GeeksforGeeks, Sep. 05, 2020. [Online]. Available: https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/.

8. Repositorio en Línea

El código fuente y la documentación completa del proyecto Scrabble están disponibles en un repositorio público en GitHub. Este repositorio proporciona acceso abierto y transparente al código del proyecto, permitiendo a los desarrolladores, colaboradores y usuarios interesados explorar, contribuir y comprender la implementación del juego.

Repositorio en Línea:
https://github.com/Scrabble-OOP/Scrabble