Práctica final: Mini Scrabble

Programación - Informática I

Juan Antonio Montes de Oca Durán

Antoni Frau Gordiola

Alejandro Sánchez Sayes

Introducción

La práctica final Mini Scrabble se trata de crear un programa en java con el entorno de desarrollo Netbeans. Siempre siguiendo los estándares enseñados del paradigma de Programación Orientada a Objetos y el uso del diseño descendente.

El programa debe dar la opción de empezar una nueva partida, visualizar el registro histórico de partidas anteriores y la opción de finalizar el programa. Cabe destacar que la forma de realizar las partidas podrá ser de forma que solo el jugador realice sus turnos o que juegue también la CPU o máquina, nombrada cerebro superior y simulador. El primer caso daría la opción de una evaluación de 0 a 7 y el segundo de 0 a 10, que es el escogido en esta práctica.

La forma de realizar las partidas debe ser con un nombre escogido por el usuario y un número de turnos a realizar también indicado, la repartición de 11 fichas, mediante las características de fichas del tipo scrabble proporcionada por el docente, para formar una palabra con dichas fichas. Esta palabra debe encontrarse en un diccionario también proporcioando por el docente, y al comprobarse su validez, sumar la puntuación de la palabra dada por las propiedades de las fichas a un contador. Al tratarse del modo de juego contra el cerebro superior, ya que el simulador no está implementado en esta práctica, la máquina en su turno deberá componer la palabra con mayor puntuación que pueda formar.

Al pasar los turnos, se indicará la puntuación final y en caso de jugar contra la máquina, también su puntuación, para poder realizar el registro histórico con la fecha y hora del momento, el nombre del jugador y su puntuación de cada partida.

Diseño

Para proporcionar el diseño final que conlleva a la estructura de la práctica, hemos pensado en crear una clase principal llamada ClasePrincipal en la que se llevan a cabo todos los procesos de la forma más general.

ClasePrincipal

El desarrollo de la clase principal cuenta con el metodoPrincipal(), en el que se lleva a cabo la elección del menú al usuario siempre que no llegue a finalizar el programa mediante un bucle del tipo while. Al escoger una opción, '1' para iniciar una partida y '2' para visualizar los registros, se llegan a llamar diferentes métodos más concretos como obtenerNombre() y obtenerTurnos(), para la introducción del nombre en una variable del tipo AccionesPartida y la cantidad de turnos en una variable entera en esta clase. Posteriormente se puede llamar a partida() para realizar las acciones de la partida como los turnos o la lectura de las fichas con sus características y visualizarRegistros() para mostrar por pantalla los registros de partidas anteriores.

Específicamente, el método de la partida llama a la generación de fichas base de la clase AccionesFichas, y mediante un bucle del tipo for hasta los turnos guardados previamente da paso al turno del jugador mediante un objeto de la clase AccionesPartida y si el modo de juego no es solitario también se ejecuta el mismo método pero para otro objeto que representa al cerebro superior. Finalizado el bucle muestra los puntos obtenidos y los guarda en el registro histórico llamando a guardarRegistros(), que hace uso de un objeto de clase FicheroControlHistorico para guardar los datos necesarios.

En cambio, el método de visualizar los registros hace uso de la clase FicheroControlHistorico para leer los registros y mostrarlos por pantalla.

AccionesFichas

El motivo de esta clase es el poder hacer las acciones que son necesarias con las fichas, haciéndola prácticamente una clase con muchos atributos y métodos estáticos, como generacionFichas(), que mediante un objeto de tipo FicheroFichasLectura lee el fichero de fichas para poder generar un array de objetos de tipo Ficha. Esto es mediante un bucle del tipo for para cada una de las 26 fichas posibles que se encuentran en el fichero, en el que se usa el objeto anteriormente mencionado para la búsqueda y lectura de los atributos de las fichas, como su carácter, cantidad y puntuación, que al final de cada iteración de registran en el array de fichas.

La clase también cuenta con el método sortearFichas(), que devuelve un array de 11 elementos Ficha escogidos aleatoriamente con el objeto del tipo Random y otro array de enteros para indicar la cantidad restante de fichas por repartir, el cual irá disminuyendo a cada ficha repartida hasta la 11ª.

Otros métodos que tiene son sePuedeFormar(), para la validación de una palabra con un array de fichas, de forma que devuelve true si esta primera tiene unas letras contenidas en el array y false si no y también el método obtenerPuntuación(). Este último recibe una palabra y usa

las fichas base para identificar los puntos de cada letra y devolver así la puntuación de la palabra.

AccionesPartida

Esta clase realiza las acciones relacionadas con los hechos en una partida, así que sirve de jugador, pero también de máquina. Para poder instanciarse tiene su constructor al que se le indica si es o no del tipo máquina.

Cuenta con métodos que modifican sus atributos como guardarNombre(), getNombre(), aplicarPuntos() o getPuntos(), que aplican y devuleven el nombre y los puntos actuales respectivamente.

También cuenta con los métodos principales durante el turno, recibirTurno() que se encarga de obtener las fichas sorteadas por la clase AccionesFichas, mostrarlas por pantalla y llamar a recibirPalabra(). Esta segunda se encarga de obtener una palabra introducida por el usuario si el modo es jugador o de obtener una palabra según las características si el modo es máquina. Posteriormente continúa a finalizarTurno() que comprueba si la palabra obtenida se puede formar por la clase AccionesFichas y si está en el diccionario con el método palabraEnDiccionario() que usa la clase FicheroDiccionarioLectura para leer el fichero del diccionario. Al hacer la comprobación aplica los puntos de la palabra por la clase AccionesFichas si esta es válida o resta 10 si no lo es.

Cuenta además con el método mostrarFichas() que imprime por pantalla las fichas actuales que tenga.

Ficha

Esta clase sirve como base para el manejo de fichas, por lo que cuenta con sus atributos necesarios y métodos como su respectivo constructor que inicializa los atributos o métodos getLetra(), getCantidad() y getValor() para recuperarlos.

Cuenta también con copiar(), para poder realizar copias de las fichas y toString() devuelve el carácter o letra y su valor, usado principalmente para mostrar las fichas disponibles a la hora de dar el turno al jugador.

FicheroControlHistorico

Es una clase dedicada a la gestión del fichero de registros históricos de partidas anteriores. Cuenta con el método constructor que recibe un booleano para escoger si inicializar la conexión con el fichero de registros destinada a la lectura o a la escritura y la ruta del fichero para realizar la conexión.

Cuenta, al estar destinada a la escritura, con métodos como guardarRegistro(), que al ser llamado por la ClasePrincipal, recibe el nombre del usuario, sus puntos totales y el número de turnos que ha durado la partida para, junto a la fecha y hora, poder escribirlos de forma que no sobreescriba el contenido anterior y aplique una nueva línea al final de cada escritura, para separar diferentes registros. A la hora de escribir los números, hace uso de su método separarNumero() ya que al ser posible que los números dados a escribir tengan más de una vocal y por tanto deban separarse para ser escritos.

Cuenta, al estar destinada a la lectura, con el método recuperarRegistros(), que lee mediante un bucle de tipo while hasta la finalización de la lectura del fichero todo su contenido y lo muestra por pantalla.

Para cerrar la conexión con el fichero tiene cerrarEnlace().

FicheroDiccionarioLectura

Esta clase está destinada a la lectura del fichero del diccionario que al estar formado por palabras, se trabaja a este nivel, de forma que cuenta con su respectivo constructor para poder instanciarlo como objeto con la ruta del fichero.

Cuenta con métodos destinados a la lectura como hayPalabras(), buscarPalabra() y lectura() para realizar la lectura del fichero hasta que encuentre una palabra y realizar la lectura de esta.

Cuenta también con métodos como estaPalabra() para, dada una palabra, recorrer el fichero hasta encontrarla, de forma que devuelve true si la encuentra y false si no. Así como también está el método palabraCerebroSuperior() que recorre recibe un array de fichas con las que recorre el fichero y comprueba si cada palabra de este puede ser formada con el array. De esta forma, al guardar las palabras válidas con mayor puntuación, llega a devolver la palabra con mayor valor.

Contiene también el método cerrarEnlace() para cerrar el enlace con el fichero.

FicheroFichasLectura

Esta clase se encarga de la lectura del fichero de fichas para manejar la obtención de las fichas base con sus propiedades iniciales. Cuenta con su respectivo constructor con la ruta del fichero y métodos dedicados a la lectura del mismo. Estos son encontrarLetra() y encontrarNumero() que hacen uso de esLetra() y esNumero(), de los cuales los dos últimos indican si un carácter es una letra del alfabeto británico y un carácter numérico respectivamente. Los dos primeros realizan la lectura mientras no encuentren sus objetivos.

Cuenta también con leerLetra() y leerNumero() para realizar la lectura de estos al ser encontrados y devolverlos, ayudado el segundo por juntarDigitos(), ya que algunos valores numéricos contienen más de un dígito y son inicialmente leídos por separado.

Tiene también el método cerrarEnlace() para cerrar el enlace con el fichero.

Palabra

Esta clase sirve para manejar secuencias de caracteres a nivel de palabras, por lo que cuenta con su constructor y métodos como buscarPalabra() para realizar la lectua por teclado de caracteres hasta encontrar un palabra y lectura() para realizar la misma.

Cuenta además con toString() para devolver sus caracteres, esLetra() para realiza comprobaciones al buscar las palabras, getNumeroCaracteres() para obtener el número de caracteres de la misma. Tiene también adicionCaracter() para añadir caracteres a la palabra y obtenerCaracter() para obtener el carácter de la palabra de la posición indicada por

parámetro, copiar() para realizar una copia de la palabra e iguales() para la comprobación de si dos palabras son iguales.

LT

La existencia de esta clase proporcionada por el docente sirve para la entrada de datos por teclado.

Conclusión

Al realizar la práctica, hemos visto la importancia y complejidad principalmente de usar de forma práctica la lógica y estructa de tratar datos secuencialmente leyendo y escribiendo en ficheros, al tener que hacer acciones varias durante esta, como la comprobación de palabras válidas y su posibilidad de ser formadas al leer el fichero de diccionario para formar una palabra para el cerebro superior.

Aún viendo complicaciones en apartados como el anterior, ya que la lógica era compleja, hemos visto que el uso de objetos simplifica y concreta más las funciones del proyecto, de forma que se puedan llamar más “masivamente” al realizar otras acciones que requieran de sus posibilidades, como el poder obtener e imprimir por pantall los datos de las fichas que tiene un objeto de tipo AccionesPartida, a través del método toString() de objetos de tipo Ficha.

Es de esta forma que hemos visto sobre todo complicaciones en la lógica de tratamientos de búsqueda y recorrido pero se han podido simplificar por la estructua general del proyecto basada en la orientación a objetos y al diseño descendente.