

Proyecto Final: El juego del ConectaN

1. Objetivos y Planteamiento	2
2. Clases	2
2.1 Clase MatrizEnteros	2
2.2 Clase Tablero	3
2.3 Clase Jugador	3
2.4 Clase ConjuntoJugadores	4
2.5 Partida	5
3. Programa main	7
4. Ejemplos de Ejecución	7

Autor: Carlos Fernández Arrabal

1.Objetivos y Planteamiento

Este proyecto consiste en la implementación de un juego de ConectaN capaz de gestionar una competición entre varios jugadores. Se tendrá que desarrollar un programa que permita llevar a cabo varias partidas entre dos jugadores a la vez, guardar sus resultados y poder manejar la información del campeonato.

En cuanto al funcionamiento detallado del proyecto, y tal como se menciona en la documentación proporcionada, el programa deberá:

- **Leer información del fichero de configuración.**
- **Gestionar los jugadores:** Se creará un objeto de la clase 'ConjuntoJugadores' para gestionar la información de los jugadores de la competición.
- **Configurar el Tablero:** Crear un objeto de la clase 'Tablero' para la partida.
- **Desarrollo de la partida:** Ambos jugadores jugarán turnándose hasta que alguno gane o se produzca un empate.
- **Actualizar la información:** Una vez finalizada la partida, se actualizará la información correspondiente a 'ConjuntoJugadores'.
- **Almacenar los datos:** La información se guardará en el fichero correspondiente.

Como base para realizar el planteamiento del proyecto se ha tenido como referencia las clases ya implementadas y la documentación proporcionada.

2.Clases

A continuación, se van a describir detalladamente todas las clases implementadas y que son necesarias para la resolución del proyecto que se aborda.

2.1 Clase MatrizEnteros

La implementación de esta clase se realizó en prácticas anteriores. Sin embargo, se han realizado una serie de modificaciones para poder adaptar la clase al proyecto e implementar una nueva funcionalidad.

La principal modificación ha sido aplicar memoria dinámica para la representación de la matriz. En base a esto, se han implementado nuevos métodos para reservar y liberar memoria. Debido a esta modificación, se han tenido que actualizar distintos métodos de la clase:

- Método `resize()`: Ahora actuará sobre la nueva variable `m`, reservando nuevo espacio y adaptando los datos.

- Constructor: Ahora se reserva memoria para inicializar la matriz.
- Destructor: Ahora se libera memoria.

A parte de estas modificaciones, se han implementado nuevas funciones necesarias para el programa: constructor de copia, operador de asignación = y el operador <<.

2.2 Clase Tablero

Al igual que la anterior, la implementación de esta clase se realizó en prácticas anteriores. En esta ocasión, se amplía la funcionalidad mediante la implementación de tres nuevos métodos:

- Método ganador(): Se encarga de analizar el tablero del juego y comprueba si alguno de los jugadores ha ganado. Para ello, se recorre el tablero al completo y se comprueba las diferentes formas de ganar (horizontal, vertical y diagonal). Se ve entonces si el número de fichas es igual a las necesarias para ganar. Si esto se cumple se imprime un mensaje de aviso junto con el id del jugador.
- Operador = : Se encarga de asignar objetos de la clase Tablero siempre y cuando no se trate del mismo objeto.
- Operador << : Se encarga de imprimir el tablero, de forma que se podrá ir viendo como evoluciona el tablero a lo largo de la partida.

A parte de esto, se hacen las correspondientes modificaciones debido al nuevo uso de memoria dinámica en la clase MatrizEnteros. Por último, cabe destacar que se hacen uso de algunas funciones relevantes ya implementadas como:

- Método IntroducirFicha() : Se encarga de ir recorriendo el tablero de abajo a arriba e introducir en la posición indicada por el usuario la ficha.
- Método tableroLleno(): Se encarga de comprobar si el tablero está lleno de fichas. Para ello comprueba si los valores del tablero son igual a 0. Con este método podemos definir si se ha producido un empate entre ambos jugadores.

2.3 Clase Jugador

Esta clase es la menos afectada ya que no requiere de ninguna modificación puesto que la estructura que tiene se adapta a la perfección para la realización del proyecto. Destacar algunos métodos fundamentales de los que se hacen uso:

- Métodos get(): Se utilizan para poder imprimir por pantalla los datos de los jugadores que participan en la competición.
- Métodos incrementar: Se encargan de aumentar el número de partidas ganadas y número de partidas jugadas.

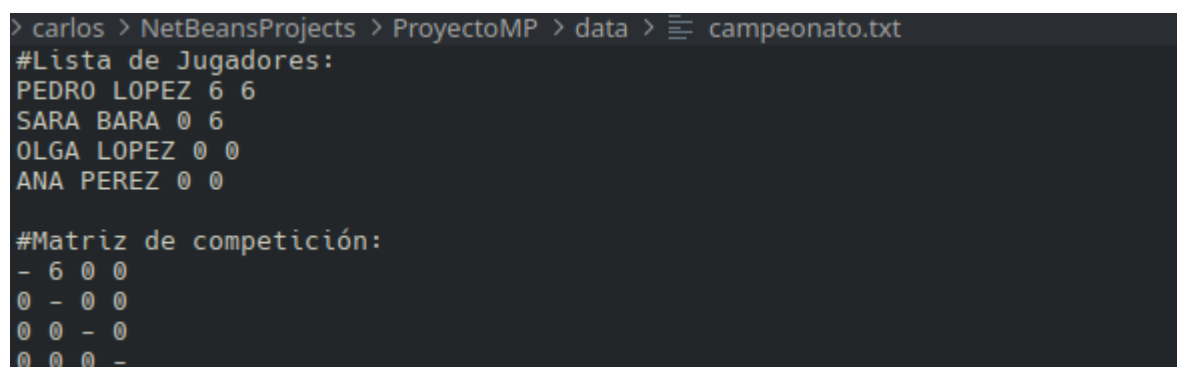
2.4 Clase ConjuntoJugadores

Para esta clase se realizarán varias modificaciones y se añadirán nuevos métodos para cumplir con la funcionalidad que se requiere. En primer lugar, se añade una nueva variable MatrizEnteros llamada competición, encargada de representar una matriz numjugadoresXnumjugadores que almacenará en la posición (i,j) el número de veces que el jugador 'i' ha ganado al jugador 'j'. Esto conlleva actualizar diferentes métodos (constructores) para que la variable esté correctamente inicializada.

A parte de esto, se crean dos nuevos métodos save y load encargados de almacenar y cargar los datos de la competición en ficheros. A continuación, se detalla cómo se han implementado cada uno de los métodos:

- Método save(): En primer lugar, se abre el fichero especificado como argumento. Mediante un bucle for, se escribe toda la información de cada jugador del vector. Por último, se escribe la matriz competición y se cierra el archivo.
- Método load(): En primer lugar, se abre el archivo especificado como argumento para su lectura. Se leen las líneas del archivo para obtener la información sobre los jugadores. La información leída se almacena en objetos de Jugador y se agregan al conjunto de jugadores. Por último, se lee y actualiza su matriz de competición con sus correspondientes valores y se cierra el archivo.

En cuanto al fichero donde se almacenan y cargan los datos, debe seguir el siguiente formato:



```
> carlos > NetBeansProjects > ProyectoMP > data > campeonato.txt
#Lista de Jugadores:
PEDRO LOPEZ 6 6
SARA BARRA 0 6
OLGA LOPEZ 0 0
ANA PEREZ 0 0

#Matriz de competición:
- 6 0 0
0 - 0 0
0 0 - 0
0 0 0 -
```

Figura 1: Campeonato.txt

Como se observa en la imagen, se listan todos los jugadores que van a participar en el campeonato con sus partidas ganadas y partidas jugadas respectivamente. También, se almacena la matriz competición que indica cuántas veces ha ganado un jugador a otro. Este fichero se irá actualizando según avance la partida.

A parte de estas funciones, se han añadido otras que cumplen con una funcionalidad muy importante en el programa:

- Método `esquemaCompeticion()`: Se encarga de mostrar la información de la matriz de competición. Las posiciones (i,i) están delimitadas por un guión ya que no tendría sentido que un jugador jugase contra él mismo.
- Método `apuntaPartida()`: Se encarga de actualizar la información en base a los resultados obtenidos de una partida realizada. El `jug1` representa al jugador ganador aumentándole en número de partidas ganadas y jugadas en uno. El `jug2` representa al jugador perdedor aumentando sólo el número de partidas jugadas. Por último se aumenta en uno la correspondiente posición en la matriz de competición.

Algunos métodos implementados con anterioridad pero que también se usan son `rankingJugadores()` que se encarga de imprimir la clasificación de todos los jugadores del campeonato, `buscarJugador()` que devuelve la posición en el vector del jugador a través de su id.

2.5 Partida

Esta clase será la encargada de la lógica detrás de una partida de conectaN, es decir, es la clase principal del proyecto. Se ha tenido que implementar desde cero y desarrollar toda su funcionalidad al completo. En cuanto a las variables y métodos implementados:

Variables

- Tablero `tab`: Objeto de tipo tablero donde jugarán los jugadores.
- Conjunto `jugs`: Objeto que contiene a todos los jugadores que participan y su respectiva información.
- `int jug1, jug2`: ids de los jugadores que participan en la partida.
- `int turnoactual`: id del jugador al que le toca jugar.

Métodos

- Constructor: Crea un nuevo objeto `partida` a partir de un objeto `Tablero` y un objeto `ConjuntoJugadores`.
- Constructor de copia.
- `turno()`: En primer lugar, imprime el estado del tablero y solicita al jugador introducir una ficha en una columna especificada mediante `cin`. A continuación, llama al método `introducirFicha` para colocar la ficha. Por último, actualiza la variable `turnoactual`.
- `inicializaPartida()`: Se encarga de inicializar las variables `jug1`, `jug2` y `turnoactual`.

- `save()` y `load()`: hacen una llamada al método `save()` y `load()` de la clase `ConjuntoJugadores` respectivamente.
- `realizaPartida()`: Llama de forma reiterada al método `turno` hasta que algunos de los jugadores ganen la partida. Entonces, llamará al método `apuntaPartida()` para apuntar los resultados.
- `muestraResultados()`: Se encarga de imprimir los resultados tras la partida, tanto la matriz de competición como el ranking de jugadores.

Por último, se piden implementar dos funciones fuera de la clase pero que forman parte de la partida del proyecto:

- `NuevaPartida()`: Se encarga de inicializar un nuevo campeonato entre jugadores. Posteriormente, inicia una partida entre dos jugadores específicos. En primer lugar, se crea un objeto `Tablero` y un objeto `ConjuntoJugadores`. Luego se inicializa un objeto `Partida` con el `Tablero` y `ConjuntoJugadores` creados, y se llama al método `inicializaPartida` con las IDs de los dos jugadores. Se lleva a cabo la partida y, al finalizar, los datos de la competición se almacenan en el fichero especificado.
- `CargaPartida()`: Se encarga de cargar un campeonato ya existente e inicia una nueva partida entre los dos jugadores especificados. Al igual que en el método anterior se crea un objeto `Tablero` y otro `ConjuntoJugadores` a partir de método `load`. Luego, se inicializa el objeto `Partida` y se llama al método `inicializaPartida` con las IDs de los dos jugadores. Se lleva a cabo la partida y, al finalizar, los datos de la competición se almacenan en el fichero especificado.

Con todo ello, se tendría implementada la clase `Partida` y podríamos pasar a la parte final del proyecto, el `main`.

3. Programa main y Fichero de configuración

El `main` deberá obtener como argumento el nombre del fichero de configuración. Se deberá comprobar si el parámetro se ha introducido correctamente. A continuación, se abrirá el fichero y se cargará la información básica. Según el contenido se darán dos casos:

- Caso 1: Si el fichero de configuración indica que no debe crear una nueva partida, se realiza una llamada a la función `CargaPartida`.
- Caso 2: Si el fichero de configuración indica que sí debe crear una nueva partida, se realiza una llamada a la función `NuevaPartida`.

En cuanto al fichero de configuración, se va a detallar cual es su estructura y que indica el contenido del mismo.

```

> carlos > NetBeansProjects > ProyectoMP > data > ≡ config.txt
#Fichero de configuracion de conectaN
NUEVAPARTIDA = no
FICHEROCAMPEONATO = campeonato.txt
JUGADORESID = 1 2
TAM = 7
NTWIN = 7
JUGADORES = 4
PEDRO LOPEZ
SARA BARA
OLGA LOPEZ
ANA PEREZ

```

Figura 2: Fichero de configuración.

En concreto, la variable 'NUEVAPARTIDA' se encargará de definir si hay que realizar un nuevo campeonato o cargar uno ya existente. Si después del igual aparece sí, se tendrá que crear un nuevo campeonato. Si por el contrario, se introduce no, entonces se carga y se continúa con el campeonato ya existente.

El resto de información define cómo se desarrollará el campeonato (tamaño del tablero, número de jugadores, número de fichas para ganar...) y son modificables según el usuario.

4. Ejemplos de Ejecución

4.1 Ejemplo 1

Configuramos un nuevo campeonato, establecemos un tablero 7x7 y el número de fichas para poder ganar será de 4. En total se definen 4 jugadores, aunque juegan sólo dos a la vez.

```

> carlos > NetBeansProjects > ProyectoMP > data > ≡ config.txt
#Fichero de configuracion de conectaN
NUEVAPARTIDA = sí
FICHEROCAMPEONATO = campeonato.txt
JUGADORESID = 1 2
TAM = 7
NTWIN = 4
JUGADORES = 4
PEDRO LOPEZ
SARA BARA
OLGA LOPEZ
ANA PEREZ

```

Figura 3: Config.txt para Ejemplo1

Estas imágenes muestran un ejemplo de una partida nueva entre dos jugadores:

```
Creando nueva partida...
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
Jugador 1, introduzca ficha (escoge columna): 1
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 1 0 0 0 0 0
Jugador 2, introduzca ficha (escoge columna): 2
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 1 2 0 0 0 0
```

```
Jugador 1, introduzca ficha (escoge columna): 1
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 1 0 0 0 0 0
0 1 2 0 0 0 0
Jugador 2, introduzca ficha (escoge columna): 3
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 1 0 0 0 0 0
0 1 2 2 0 0 0
Jugador 1, introduzca ficha (escoge columna): 2
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 1 1 0 0 0 0
0 1 2 2 0 0 0
```



```

Jugador 2, introduzca ficha (escoge columna): 3
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 1 1 2 0 0 0
0 1 2 2 0 0 0
Jugador 1, introduzca ficha (escoge columna): 3
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 1 1 2 0 0 0
0 1 2 2 0 0 0
Jugador 2, introduzca ficha (escoge columna): 4
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 1 1 2 0 0 0
0 1 2 2 2 0 0

```

```

0 0 0 1 0 0 0
0 1 1 2 1 0 0
0 1 2 2 2 0 0
Jugador 2, introduzca ficha (escoge columna): 4
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 1 2 0 0
0 1 1 2 1 0 0
0 1 2 2 2 0 0
Jugador 1, introduzca ficha (escoge columna): 4
El jugador 1 ha ganado la partida
Matriz De Competicion:
- 1 0 0
0 - 0 0
0 0 - 0
0 0 0 -
Ranking:
1,PEDRO,LOPEZ,1.000000
2,SARA,BARA,0.000000
3,OLGA,LOPEZ,0.000000
4,ANA,PEREZ,0.000000
RUN FINISHED; exit value 0; real time: 38s; user: 0ms; system: 0ms

```

En la última imagen se puede observar como el jugador 1 ha ganado la partida conectando las 4 fichas en diagonal, se muestra la matriz de competición actualizada y se actualiza y se muestra el ranking

4.2 Ejemplo 2

Ahora vamos a cargar una partida ya existente, en este caso la que se ha realizado anteriormente. Para ello se pone 'NUEVAPARTIDA' a no y se comienza a jugar.

```
> carlos > NetBeansProjects > ProyectoMP > data > config.txt
#Fichero de configuracion de conectaN
NUEVAPARTIDA = no
FICHEROCAMPEONATO = campeonato.txt
JUGADORESID = 1 2
TAM = 7
NTWIN = 4
JUGADORES = 4
PEDRO LOPEZ
SARA BARA
OLGA LOPEZ
ANA PEREZ
```

Figura 4: Config.txt para Ejemplo2

```
Jugador 1, introduzca ficha (escoge columna): 3
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 2 1 0 0 0
0 0 2 1 0 0 0
0 1 2 1 0 0 0
Jugador 2, introduzca ficha (escoge columna): 2
El jugador 2 ha ganado la partida
Matriz De Competicion:
- 1 0 0
1 - 0 0
0 0 - 0
0 0 0 -
Ranking:
1,PEDRO,LOPEZ,0.500000
2,SARA,BARA,0.500000
3,OLGA,LOPEZ,0.000000
4,ANA,PEREZ,0.000000
```

Ahora se ha hecho que el jugador 2 gane la partida. Así vemos cómo se actualiza el ranking y la matriz de competición.

Por último, se muestra el fichero campeonato.txt pudiendo ver como ha ido evolucionando el campeonato y ver los resultados obtenidos.

```
> carlos > NetBeansProjects > ProyectoMP > data > campeonato.txt
#Lista de Jugadores:
PEDRO LOPEZ 1 2
SARA BARA 1 2
OLGA LOPEZ 0 0
ANA PEREZ 0 0

#Matriz de competición:
- 1 0 0
1 - 0 0
0 0 - 0
0 0 0 -
```

Figura 5: Campeonato.txt