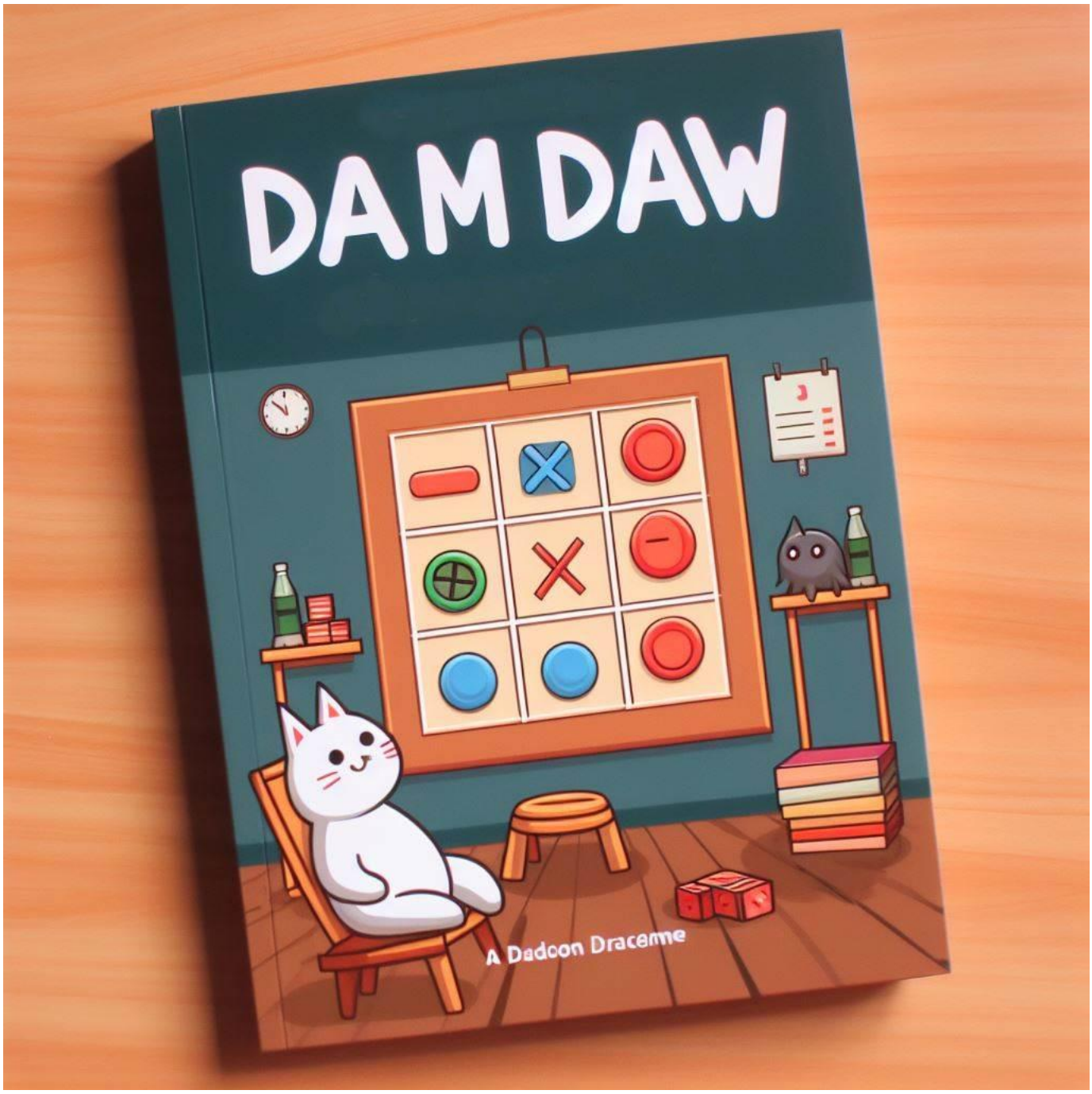
C/ Societat Unió Musical, 8 - 03802 Alcoi
Tel. 966 52 76 60 - Fax 966 52 76 61
03012165.secret@gva.es
www.cipfpbatoi.es



ACTIVIDAD 6.18

(hasta 8 puntos)

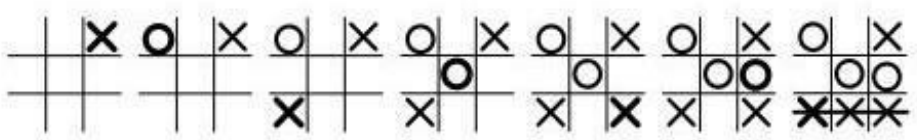
TRES EN RAYA

ÍNDICE

ACTIVIDAD 6.18 - Tres en raya 1/2 (Hasta 8 puntos)

Enunciado

Vamos a desarrollar el famoso juego del **tres en raya**. Se trata de un juego entre 2 jugadores, **O** y **X**, los cuales van introduciendo sus fichas en un tablero de **3x3.** El **jugador ganador** será aquel que consiga colocar **3 fichas en línea** ya sea en **posición horizontal**, **vertical,** o en **diagonal**. La siguiente imagen muestra una partida en la que el **jugador X** ha ganado.

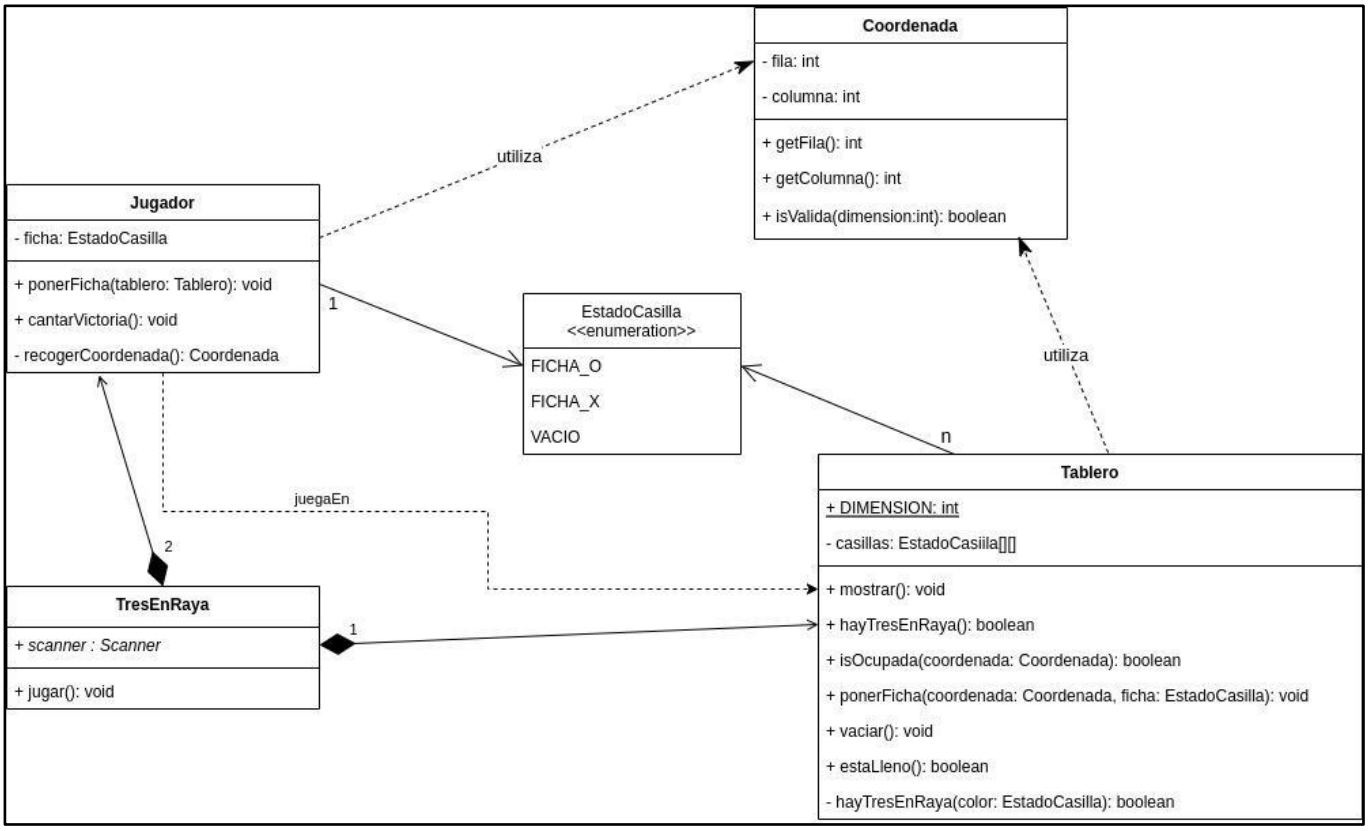


Por otro lado, la partida también puede **finalizar en empate** si se completa todo el tablero y ningún jugador ha conseguido poner 3 de sus fichas en línea.

Se pide desarrollar un programa que nos permita jugar al "**tres en raya**" utilizando la metodología de programación orientada a objetos (**POO**).

A continuación, se establece el **diagrama de clases** en el que se deberá apoyar tu

sistema:



⚠ Solo se definen las **clases (y enumerados) mínimas** que deben existir y su responsabilidad en la aplicación, así como cada uno de **sus métodos mínimos.** El alumno deberá decidir si implementar nuevas clases o métodos tanto públicos como privados para las clases propuestas o incluso incluir clases adicionales.

⚠ Por convenio, **deberás definir la clase Scanner** como un atributo `public` y `static` en la clase `TresEnRaya` para poder tener acceso al objeto `Scanner` desde cualquier clase del sistema. La creación del objeto de esta clase se deberá realizar en la primera línea del método `main()`. Esto implica que podrás acceder al objeto de `Scanner` creado desde cualquier método de cualquier clase del sistema. Debes llamar a la variable que hace referencia al objeto "`scanner`".

Está **terminante prohibido** crear **clases estáticas**, esto es, clases que sólo contienen métodos estáticos, salvo en el caso de que decidas crear una clase que esté exclusivamente

⚠ dedicada a hacer uso del objeto Scanner (que habrás creado en la clase principal `TresEnRaya`, tal y como se ha indicado en el tip anterior).

Especificación de clases

❖ Coordenada

Esta clase será la encargada de manejar las posibles ubicaciones de cada una de las casillas del tablero y, por tanto, de las posibles tiradas del usuario. Por tanto, almacenará la fila y la columna de las posibles casillas del tablero. Deberás implementar los siguientes métodos:

- Un **constructor** que reciba como argumento todos los atributos de la clase.
- Los **getters** (consultores) de cada uno de sus atributos.
- `public boolean isValida(int dimension):` Determina si una coordenada es válida, aceptando como válida una coordenada si sus valores de fila y columna se encuentran dentro de la dimensión recibida como parámetro. Por ejemplo, para una dimensión de 3, se daría por válida la coordenada si su fila y su columna están comprendidos entre 1 y 3, siendo incorrectos los valores de 0 hacia atrás y de 4 en adelante. Es decir, para el caso de una dimensión 3, una coordenada en la fila 1 y columna 3 sería una coordenada válida (el método devolvería true), mientras que una coordenada en una fila 0 y columna 2, no sería una coordenada válida (el método devolvería false).

❖ Tablero

Esta clase representará al tablero y deberá disponer de los métodos necesarios para que los jugadores puedan insertar su ficha, así como saber si caben más fichas en el tablero o si la partida ha finalizado. Cada una de las casillas vendrá representada por una matriz de **N x N** elementos del **tipo enumerado** `EstadoCasilla` (que deberás definir en un fichero java aparte).

Define también **N** como una constante llamada `DIMENSION` de tipo entero, de forma que, si cambiamos su valor a 5, el juego se convertirá en el 5 `en raya` jugando a partir de

⚠ ese momento en un tablero de **5 x 5 (5 filas y 5 columnas)** y con el **objetivo** de colocar **5 fichas en línea**.

A continuación, se definen los métodos que deberemos implementar en la clase `Tablero`:

- Un **constructor** sin parámetros en el que se inicien las casillas al estado vacío. Para ello, hará uso del método `vaciar()`.
- `public void mostrar()`: Dibuja el estado actual del tablero en la pantalla siguiend el formato mostrado en el ejemplo de ejecución.
- `public boolean hayTresEnRaya()`: Indica si existe 3 en raya (o la dimensión que esté establecida en la constante DIMENSION) en el tablero, es decir, indica si alguno de los 2 jugadores ha realizado 3 en raya y, por tanto, la partida ha finalizado. Para ello se valdrá del siguiente método privado:
  - `private boolean hayTresEnRaya(EstadoCasilla color)`: Indica si existe 3 en raya para la ficha del color pasado como argumento (será invocado por el método anteriormente descrito)
- `public boolean isOcupada(Coordenada coordenada)`: Retorna un booleano indicando si la casilla de la posición `coordenada` está ocupada.
- `public void ponerFicha(Coordenada coordenada, EstadoCasilla color)`: Inserta una ficha del color `@color` en la casilla de la posición `@coordenada` del tablero.
- `public void vaciar()`: Este método nos servirá tanto para inicializar el tablero como para vaciarlo para que pueda ejecutarse una nueva partida. Para ello establecerá todas las casillas del tablero como `VACIO` (uno de los estados de `EstadoCasilla`).
- `public boolean estaLleno()`: Devuelve un booleano indicando si el tablero está lleno, es decir, si se pueden insertar más fichas en él. Un tablero estará lleno, cuando no exista ninguna posición vacía.

❖ Jugador

Esta clase representará a cada uno de los jugadores de la partida, deberá disponer del

tipo de ficha con el que el jugador está jugando (`FICHA_X` o `FICHA_0`) y de los siguientes métodos **públicos/privados**:

- Un **constructor** que reciba como argumento el **color de la ficha** con la que jugará la partida.
- `public void ponerFicha(Tablero tablero)`: Deberá preguntar al usuario en qué coordenada quiere insertar la ficha **siguiendo el formato presentado en el ejemplo de ejecución**. El tablero, que recibe como argumento, se utilizará para comprobar si la posición introducida por el usuario está libre y colocar la ficha, si no, se mostrará un mensaje de error, solicitando una nueva posición para la ficha. Deberás crear e invocar desde el cuerpo de este método al **siguiente método privado** que se encargará de lo siguiente:
  - `private Coordenada recogerCoordenada()`: Obtiene, preguntando al usuario (siguiendo el formato del ejemplo de ejecución) en qué coordenada desea colocar la ficha. Dentro de este método, se debe validar la coordenada (**debe ser compatible con la dimensión establecida para el tablero**). Este método volverá a solicitar los datos en caso de recibir valores incorrectos por parte del usuario (mostrando el mensaje de error correspondiente). Sigue el formato de los textos a mostrar teniendo en cuenta el ejemplo de ejecución.
- `public void cantarVictoria()`: Muestra por pantalla el mensaje de que ha ganado la partida **siguiendo el formato presentado en el ejemplo de ejecución**

❖ TresEnRaya

Esta clase contendrá el método `main()` y será la encargada de controlar **el flujo de ejecución** de toda la aplicación.

Como **atributos (o variables de instancia)** declararemos una variable de tipo `Tablero` y 1 array de longitud 2 en el que guardaremos los objetos de tipo `Jugador` que van a jugar la partida.

- Un **constructor sin parámetros** que inicialice las variables de instancia de `TresEnRaya`.

- `public void jugar()`: de forma alternativa, irá llamando a los métodos `ponerFicha` de cada jugador hasta que exista un **"3 en raya"** o un **"empate".**

Ten en cuenta que la clase `TresEnRaya` es una clase que forma parte del sistema y, por tanto, se debe crear un objeto de esta clase también (en este caso en el método `main()`). El método `main`, por tanto, tendría este aspecto:

```
public static void main(String[] args){
    scanner = new Scanner(System.in);
    TresEnRaya tresEnRaya = new TresEnRaya();
    tresEnRaya.jugar();
}
```

Observa también que existe una **relación de composición** (*esta relación la vimos en la actividad por parejas realizada días atrás*) entre esta clase y las clases `Jugador` y `Tablero`.

A continuación, se presenta un ejemplo de la utilización conjunta de algunos métodos de los objetos de las clases descritas (ten en cuenta que este ejemplo **NO es el método** `jugar()`) de esta clase:

```
// Creamos el tablero del juego
Tablero tablero = new Tablero();

// Creamos los 2 jugadores
Jugador jugador1 = new Jugador(EstadoCasilla.FICHA_X);

Jugador jugador2 = new Jugador(EstadoCasilla.FICHA_O);

// El jugador 1 inserta una ficha en el tablero,
jugador1.ponerFicha(tablero);

// Una vez insertada mostraremos el tablero
tablero.mostrar();
if (tablero.estaLleno()){
    System.out.println("La partida ha finalizado en empate");
}
if (tablero.hayTresEnRaya()){
 System.out.println("Ha ganado la partida el jugador X");
```

```
}
// El jugador 2 inserta una ficha en el tablero,
jugador2.ponerFicha(tablero);
tablero.mostrar();
```

Ejemplo de ejecución (un jugador gana la partida)

```
Vamos a jugar al "Tres en Raya"

| |1|2|3|
|1| | | |
|2| | | |
|3| | | |

Jugador con 0

Introduce fila [1-3]: a

¡Error! Debe introducir un número entero

Introduce fila [1-3]: 1

Introduce columna [1-3]: 4

¡Error! Introduce una coordenada válida

Jugador con 0

Introduce fila [1-3]: 4

Introduce columna [1-3]: 1

¡Error! Introduce una coordenada válida

Jugador con 0

Introduce fila [1-3]: 1

Introduce columna [1-3]: 2

| |1|2|3|
|1| |0| |
|2| | | |
|3| | | |

Jugador con X

Introduce fila [1-3]: 1

Introduce columna [1-3]: 2

¡Error! Coordenada ocupada en el tablero

Jugador con X
```

```
Introduce fila [1-3]: 1

Introduce columna [1-3]: 1

|  |1|2|3|
|1|X|0|  |
|2|  |  |  |
|3|  |  |  |

Jugador con 0

Introduce fila [1-3]: 1

Introduce columna [1-3]: 3

|  |1|2|3|
|1|X|0|0|
|2|  |  |  |
|3|  |  |  |

Jugador con X

Introduce fila [1-3]: 2

Introduce columna [1-3]: 1

|  |1|2|3|
|1|X|0|0|
|2|X|  |  |
|3|  |  |  |

Jugador con 0

Introduce fila [1-3]: 2

Introduce columna [1-3]: 3

|  |1|2|3|
|1|X|0|0|
|2|X|  |0|
|3|  |  |  |
```

```
Jugador con X

Introduce fila [1-3]: 3

Introduce columna [1-3]: 1

¡El jugador X es el ganador!

|  |1|2|3|
|1|X|0|0|
|2|X|  |0|
|3|X|  |  |

¿Quieres volver a jugar? [S/N]: hola

¡Error! Debes introducir S o N

¿Quieres volver a jugar? [S/N]: N
```

Ejemplo de ejecución (empate)

Exactamente igual al ejemplo de ejecución anterior, salvo que cuando la partida acaba, antes de solicitar si se desea una nueva partida, se imprime por pantalla el texto "`Empate`"