

Tarea DWES05 - 25/26

Vamos a desarrollar un programa para simular partidas de ajedrez. Crearemos varias clases:

1. Clase **Pieza**, que tendrá los siguientes atributos y métodos:
 - a. \$posición(Atributo). Definida mediante una letra entre A y H (que indica la columna), y un número entre 1 y 8 (que indica la fila). Si la pieza ha sido capturada el valor de la posición será "CAPTURADA". Ejemplos de posiciones: A1, C4, H4...
 - b. \$valor(Atributo). De tipo numérico, quedará establecido con los diversos constructores de las subclases.
 - c. Métodos getter, setter y constructor. Estos métodos deben crearse en las sucesivas clases.
2. Después tendremos unas subclases que heredan de pieza. Todas tendrán un método en común, que será *movimiento()*, que recibe como parámetro una cadena de texto con la nueva posición y el propio método tendrá que verificar si el movimiento es válido según el tipo de pieza. Las subclases serán:
 - a. **Caballo**, que se mueve dos escaques en dirección vertical u horizontal y otra en ángulo recto. Es la única pieza que puede saltar casillas. Su valor es 3.
 - b. **Alfil**, se mueve en líneas diagonales. Con valor igual a 3.
 - c. **Torre**, con movimientos en líneas horizontales o verticales. Su valor es 5.
 - d. **Dama**, que se mueve en líneas horizontales, verticales y diagonales. Tiene un valor de 9.
 - e. **Rey**, que se puede mover en cualquier dirección pero solo una casilla. No tiene valor calculado, pues su pérdida supone también la pérdida de la partida.
 - f. **Peón**, que puede avanzar en dirección vertical una casilla, avanzar una casilla en diagonal si es para hacer una captura o avanzar dos casillas en su primer movimiento.

El atributo con la posición de la pieza (de la superclase) se actualizará si el movimiento es legal. Si no conocéis las reglas del juego podéis consultarlas en la Wikipedia: https://es.wikipedia.org/wiki/Leyes_del_ajedrez#Movimiento.

Además, todas estas subclases tendrán en común otro método, *simulaMovimiento()*. Al igual que movimiento recibe un parámetro con la nueva posición. Este método verifica si el movimiento es legal y si es así devuelve un array con todas las posiciones por las que ha pasado la pieza desde su posición original hasta la posición del supuesto movimiento (el caso del caballo es la excepción, pues es la única pieza que puede saltar casillas). Este método no modifica el valor de ningún atributo.

3. Clase **Jugador**, que tendrá los siguientes atributos, además de constructor, setters y getters:
- a. \$nombre, simplemente para establecer el nombre del jugador.
 - b. \$color, que podrá tener los valores “blancas” o “negras”.
 - c. \$piezas, un array con objetos de las subclases de Pieza, inicialmente 16.
 - d. En el constructor habrá que tener en cuenta el color para colocar cada una de las piezas en su posición inicial del tablero.
4. Clase **Partida**, con los siguientes atributos y métodos:
- a. \$jugadores, un array de dos jugadores, uno con piezas blancas y otro con negras.
 - b. \$turno, establece a quién le toca mover, de inicio será el jugador que juega con piezas blancas.
 - c. jugada(), este método recibe dos parámetros que son la posición de origen y la posición de destino. Debe comprobar que existe una pieza en la posición de origen, que corresponde al jugador al que le toca mover y que esa pieza puede moverse a la posición de destino. No debe haber ninguna pieza en las posiciones intermedias y si en la posición de destino hay una pieza, esta debe ser del color contrario, además, tras el movimiento de la pieza que estaba en la posición de destino debe ser eliminada.
 - d. marcador(), método que no recibe ningún parámetro y que nos devuelve un array con dos enteros, correspondientes a los puntos (suma de los valores de todas las piezas), que tiene cada jugador.
 - e. muestraTablero(): puede ser en modo texto, mostrando una letra por cada pieza, usando mayúsculas y minúsculas para distinguir entre blancas y negras. Si no hay pieza en la casilla se pone una x. Por ejemplo el comienzo de la partida sería así:

```
tcadract
pppppppp
xxxxxxxx
xxxxxxxx
xxxxxxxx
xxxxxxxx
PPPPPPPP
TCADRACD
```

En todo caso, tenéis libertad en este aspecto para hacerlo de un modo más gráfico, si así os gusta.

5. Para ver el funcionamiento de las clases hay que crear un código de prueba que permita crear un objeto de tipo Partida y mediante un formulario enviarle jugadas. Tras cada jugada se mostrará el estado del tablero y el marcador. No será necesario tener en cuenta reglas como captura al paso, coronación de peón o enroque.

Indicaciones de evaluación y envío

No es necesario adjuntar ningún documento explicativo, pero sí añadir comentarios que ayuden a la comprensión del programa, especialmente en los métodos de las clases. Cada uno de los siguientes ítems de evaluación tendrán un valor de 2 dos puntos:

- Los métodos *movimiento()* de las subclases de *Pieza*.
- Los métodos *simulaMovimiento()*.
- Método *jugada()* de la clase *Partida*.
- Métodos *muestraTablero()* y *marcador()* de la clase *Partida*, 1,5 puntos y 0,5 puntos respectivamente.
- Programa de prueba.

En la evaluación de cada ítem se tendrá en cuenta, además de su correcto funcionamiento, su claridad y el uso de comentarios. En caso de añadir un documento explicativo debe ser en formato PDF.

La tarea se puede realizar creando uno o varios archivos (se recomienda esto último siguiendo los conceptos vistos de POO). Todos ellos deben ser comprimidos en un archivo .RAR o .ZIP y el archivo comprimido debe ser enviado a la tarea de la plataforma según la nomenclatura típica:

Apellido1_Apellido2_Nombre_DWES05-TAREA