Apellidos: | **Nombre:**

Equipo: | **CRC-32:**

Práctica POE. Actualización del Juego de Barcos

En la carpeta $Examen\_POE$ que se encuentra en el escritorio, está disponible una versión de la práctica que es prácticamente idéntica a la solución que se proporcionó en su momento para la práctica del "Juego de barcos" realizada durante el curso.

Además de la solución que se proporciona, se incluye una nueva versión de la biblioteca $componentes.jar$ para manipular los objetos del juego: los *tableros*, las *celdas* de éstos¸ los *barcos* y sus posiciones en el tablero. La nueva versión, $componentes\text{-}2.jar$, es casi idéntica a la previa e incorpora un único cambio: *el método* $alcanzado(row, col)$ *de la clase* $Barco$ *ahora retorna* $null$ *si el tiro de coordenadas especificadas ya se había realizado previamente.* Como consecuencia de este pequeño cambio, ha sido necesario retocar el código de la operación del servicio $coordenadasTiro(str)$ para que sea correcta.

Partiendo de la solución proporcionada y la biblioteca $componentes\text{-}2.jar$, realiza los cambios que se solicitan en los ejercicios que se plantean a continuación.

Ejercicio 1 *[3 puntos]*

Modifica la especificación de la operación $coordenadasTiro(tiro)$ tal y como se indica a continuación y realiza los cambios que sean necesarios para que ésta se cumpla.

```
/**
 * Coordenadas del tablero océano del oponente donde se ataca. Si el
 * estado del OOS no es el esperado, se ignorará el tiro realizado.
 * @param tiro las coordenadas. Éstas serán siempre correctas
 */
void coordenadasTiro(String tiro);
```

Ahora esta operación ya no retornará lo que ocurre con el tiro realizado para mostrárselo al usuario porque el efecto de éste ya se refleja en el tablero de tiros del jugador. Además, ahora la operación no lanzará excepciones. Esto es debido a que, por un lado, si el estado no es el adecuado, el tiro no tendrá efecto alguno (se ignorará) y a que, por otro lado, **las coordenadas de tiro dadas serán ahora validadas en el cliente** y serán siempre válidas.

Ejercicio 2 *[6 puntos]*

En la nueva versión del juego, se han de tener en cuenta las REGLAS ADICIONALES siguientes:

a) Los jugadores solo podrán disparar, durante el juego, un número máximo de veces al océano del oponente. Una vez que uno de los dos jugadores alcance dicho máximo el juego finalizará. (Nótese que, al igual que antes, también finalizará si se hunden todos los barcos de un jugador).

b) En caso de que el juego finalice por alcanzar el máximo de tiradas, lo ganará el jugador que haya conseguido más dianas (sólo habrán de contabilizarse dianas no repetidas), pudiendo darse el caso de empate si ambos jugadores tienen las mismas.

Se pide:

1. *[0,6 puntos]* Declarar una nueva constante de nombre $MAXIMO\_TIROS$ de valor el número máximo de tiros que podrán realizar cada jugador durante la partida y añadir a la interfaz la operación que se especifica a continuación:

```
/**
 * Retorna {@code <0}, {@code 0} o {@code >0} según el jugador haya
 * perdido, empatado o ganado el juego, respectivamente.
 * @return {@code <0}, {@code 0} o {@code >0} según el jugador haya
 * perdido, empatado o ganado el juego, respectivamente
 * @throws AccionNoPermitida si el juego no ha finalizado
 */
int ganador() throws AccionNoPermitida;
```

2. *[1,2 puntos]* Añade e inicializa la información adicional requerida para que el juego se desarrolle de acuerdo con las reglas adicionales dadas.
3. *[3 puntos]* Implementa la nueva operación del servicio: $ganador()$ y realiza los cambios adicionales que se requieran en el servidor.
4. *[1,2 puntos]* Modifica el cliente para informar del resultado del juego cuando este finaliza, indicando si el jugador pierde, gana o empata.

Nota: en este ejercicio no se valorará el realizar la exclusión mutua de la información compartida que se añada para solucionarlo, pues es lo que se solicita en el siguiente ejercicio.

Ejercicio 3 *[1 punto]*

Realiza la exclusión mutua de las nuevas secciones críticas que haya en el servicio, debido a la información compartida que haya sido necesario añadir para solucionar el ejercicio previo.