# IMPLEMENTACIÓN DEL JUEGO HUNDIR BARCOS

Un jugador compite contra la máquina. Sabemos que muchos han trabajado en este mismo proyecto, así que, en lugar de explicar las reglas básicas del juego, nos centraremos en cómo lo hemos desarrollado, destacando las funciones, clases y lógica que hemos implementado.

## Variables, funciones y su propósito

Para estructurar el código de manera clara y reutilizable, hemos definido en primer lugar las variables y dos funciones clave que permiten modularizar la lógica del juego. Estas son:

**Variables**: dificultas, size, eslora, celdas (vacia, barco, tocado, agua)

* Función Translate coord: las usamos para pasar las coordenadas de los barcos
* Función Translate xy: convierte dos índices numéricos en una coordenada letra, numero.

## Clases y su propósito

Además de las funciones, decidimos estructurar el proyecto utilizando clases, (gran parte del código son las clases y sus atributos) para mantener el código organizado. Definimos:

1. Batlleship: donde se crea tanto el tablero de la maquina como el del jugador teniendo en cuenta el máximo número de letras del abecedario y de espacio para imprimirlo por pantalla (son dos instancias de la misma clase) tiene un print para imprimir ambos tableros.
2. Clase tablero: Alojara un Fata Frame en su atributo grit, también albergar un bolleano en el atributo otro\_turno, para saber si hay que volver a disparar y posiciona todos los barcos definidos en la constante eslora, llamando a su método posicionar aleatorio, finalmente guarda una lista de posiciones disparadas Aquí definimos varios constructores dentro de la clase y métodos de métodos para el funcionamiento de ambos tableros como:

* Def\_\_getitem\_\_ para el formato de la coordenada
* Def\_\_setitem\_\_ para la coordenada de letra y numero (devuelve el atributo grid)
* Def\_\_str\_\_ representación de string del tablero
* Posicionar. Método que capa el tamaño y dimensiones del barco, así como su posición si ya existe alguno
* Posicionar aleatorio: método que da unas coordenadas aleatorias para posicionarse
* Disparar: método para los disparos
* Disparo maquina: método para los disparos de la maquina

1. Clase Barco: representa un barco individual con sus diferentes eslora y jugador, y con una clase anidada subarco, con las diferentes posiciones dentro del barco para determinar cuándo esta hundido., para ello utilizan un constructor de casilla barco.

## Demostración del juego

Vamos a mostrar cómo funciona nuestra implementación:

1. Inicio del juego:

Se generan dos tableros, uno para el jugador y otro para la máquina. Los barcos se colocan automáticamente.

2. Turnos alternos:

Primero, el jugador introduce las coordenadas para disparar, si fallas toca el turno de la máquina, luego, la máquina realiza su disparo, siguiendo una lógica de selección aleatoria o dirigida según aciertos previos.

Si el disparo es acertado vuelves a disparar, si no pasa el turno al otro jugador.

3. Fin del juego:

Una vez se hayan hundido todos los barcos de un mismo jugador se acaba el juego.

## Explicación del código

1. Inicialización del juego:

Primero importamos todas las librerías necesarias, el código se encuentra dentro de un while que se rompe cuando se acaba el juego porque todos los barcos estén hundidos, dentro de este while tenemos otro, que no se rompe cuando fallemos si no que contiene un try except para que nos vuelvan a preguntar coordenada

1. Modularidad:

Dividimos claramente la lógica en funciones y métodos dentro de clases. Esto nos permitió trabajar en paralelo y mantener el código legible, hay dos partes de código uno para la maquina y otro para el jugador, sin embargo, son muy similares

1. Gestión de errores:

Validamos los inputs para evitar coordenadas inválidas o duplicadas.

Utilizamos manejo de excepciones para garantizar que el programa no se bloquee.