

OESCUELA COLOMBIANA DE INGENIERÍA

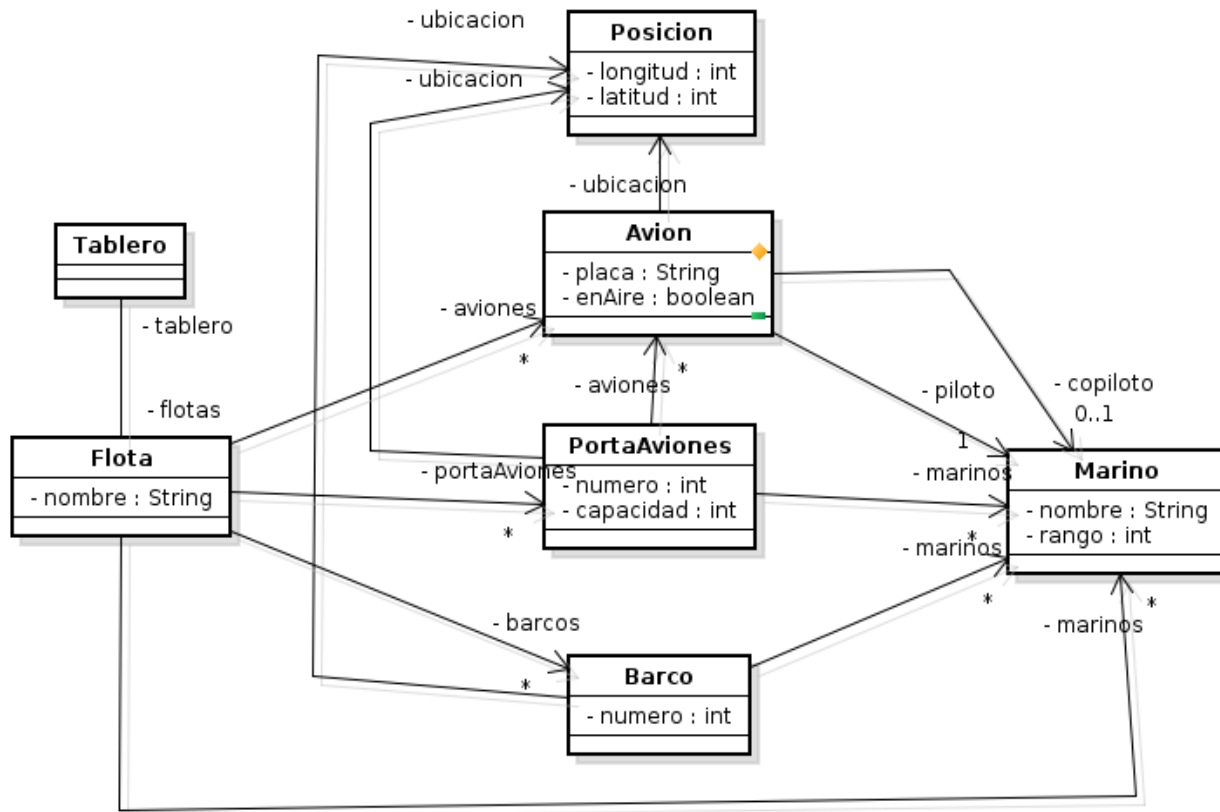
PROGRAMACIÓN ORIENTADA A OBJETOS

Memoria, diseño y construcción

S03-S04: 2022-1

BATALLA NAVAL

En el **juego batalla naval** una serie de flotas se pelean entre sí para tomar el control del tablero. Estas flotas se componen de barcos, portaaviones, aviones y marinos. De la totalidad de los marinos disponibles en la flota, sólo los que están en turno están asignados a una de estas naves. De las naves se conoce su ubicación dada en longitud y latitud. En una ubicación pueden estar muchas naves.



(Todos los contenedores son ArrayList)

I. MEMORIA

Presenten el mapa de memoria correspondiente a:

- En el juego participan dos flotas "LA GRAN FLOTA BLANCA" y "LA GRAN ARMADA DE CASTILLA"
- LA GRAN FLOTA BLANCA tiene un porta avión (número 100) que lleva dos aviones actualmente en misión de ataque (placas HR100 y JB100) piloteados por Henry Reuter Dahl y John Charles Roach respectivamente. Este portaaviones, cuyo capitán es el almirante Sperry, está al 50% de su capacidad.
- LA GRAN ARMADA DE CASTILLA tienen un barco (número 900) y un avión (placa PEACE). Pedrarias Dávila y Fernando Villamil son marinos de esta flota. Pedrarias está asignado al barco 900. Fernando está sin asignación.

II . ATRIBUTOS.

A. Los diseñados

Escriban el encabezado y los atributos de las clases: Flota y Tablero. (Clase = Atributos+ ..)

B. Nuevos atributos

En el juego se quiere incluir la siguiente información.

- Un código para identificar las flotas que no se puede modificar y puede ser consultado por todos.

- Los tripulantes mínimos que necesitan y los puntajes que otorgan las diferentes elementos de guerra: portaavión, barco y avión. Los puntajes se pueden cambiar durante el juego pero los tripulantes necesarios no.
 - El mundo tablero es cuadrado. La longitudes son -100 a 100 y las latitudes -100 .. 100
1. Adicionen los nuevos elementos en el diagrama de clases.
 2. Escriban el código necesario para implementarlos.
No olvide indicar la clase en la que está escribiendo y la documentación (si es necesaria).

II . MÉTODOS

A. Métodos de ArrayList

Consulten los siguientes métodos de la clase **Class ArrayList <E>** en el API de Java:

`add, contains, get, indexOf, isEmpty, size.`

B. Nuevos métodos

Desarrollen los métodos de la clase Flota seleccionados en clase considerando los siguientes pasos:

1. Adicionen los nuevos elementos en el diagrama de clases.
2. Diseñen el método usando un diagrama de secuencia y adicione los nuevos elementos en el diagrama de clases.
3. Escriban el código necesario para implementarlo.
No olvide indicar la clase en la que está escribiendo y la documentación (si no está documentado).
No construya los básicos (`get, set, is`)

Class Flota

alias()**

`public int alias(**)`

Consulta el número de flotas que tienen su mismo nombre

Returns:

numero de flotas con el mismo nombre

disponibilidadEnPortaaviones()**

`public int disponibilidadEnPortaaviones()`

Consulta la disponibilidad total del portaavion

Returns:

numero de aviones adicionales que podrían cargarse a los portaaviones

enAire(2)**

`public ArrayList<String> enAire()`

Consulta la placa de los aviones enemigos que están en el aire

Returns:

la placa de los aviones enemigos que están en el aire

esBuenAtaque (1*)**

`public boolean esBuenAtaque(int longitud, int latitud)`

Verifica si la ubicación para un ataque en agua es adecuado

(destruye elementos enemigos sin ocasionar bajas propias. Los aviones que están volando no se afectan.)

Parameters:

longitud - longitud de la explosion

latitud - latitud de la explosion

muevase()**

`public void muevase(int deltaLongitud,int deltaLatitud)`

Mueve todos los barcos la distancia definida, si es posible.

Parameters:

deltaLongitud - avance en longitud

deltaLatitud - avance en latitud

numeroMaquinas (*)

`public int numeroMaquinas()`

Consulta el numero de maquinas que tiene la flota

Returns:

numero de maquinas de la flota

problemaEnAire (2*)**

`public boolean problemaEnAire()`

Consulta si puede confundir sus aviones con aviones enemigos considerando las placas

Returns:

si hay problema en aire

suficientesMarinos(*)

`public boolean suficientesMarinos()`

Consulta si cuenta con suficientes marinos para conducir sus máquinas.

Un portaaviones requiere 5 marinos; un barco, 4; y un avión 2.

Returns:

si hay problema en aire

seranDestruidas (1)**

`public ArrayList<Object> seranDestruidas(int longitud,int latitud)`

Consulta las máquinas que pueden afectarse por una explosion en agua

Parameters:

longitud - longitud de la explosion

latitud - latitud de la explosion