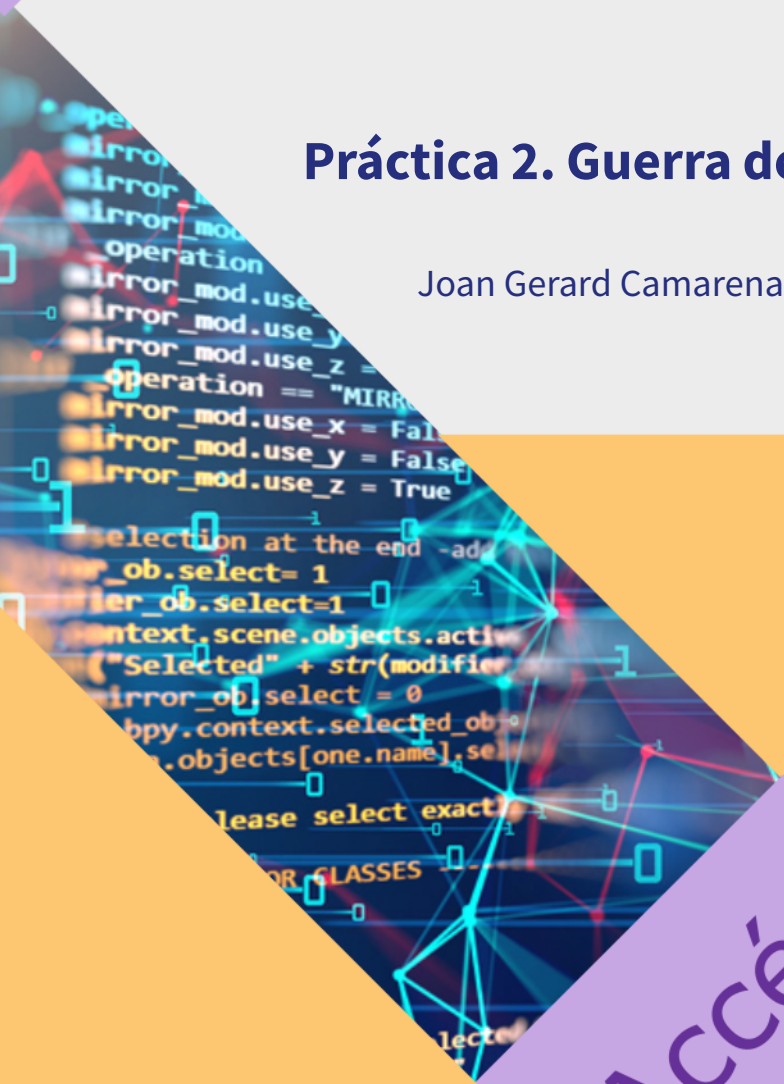
Curs 2020-2021



Pràctica 2. Guerra de Barcos con Objetos.

Joan Gerard Camarena Estruch

Accés a Dades

Pràctiques

IES Jaume II El Just
Tavernes de la Valldigna

1. La guerra de barcos

Se pide una implementación más OO que el ejercicio visto en clase. Las clases que participaran en ela són:

- `Cell` → Clase que controla el contenido de cada una de las celdas del tablero.
- `Boat` → Clase que controla la lógica de los barcos.
- `Board` → Clase que controla la lógica del tablero.

Antes de implementar se deberia hacer:

- Un pequeño esquema en UML de las relaciones entre la clases
- Un diagrama de objectos viendo las referencias que hay entre los objetos y cómo se relacionan entre ellos.

2. `Cell`

Esta clase representará una celda del tablero. Para cada celda:

- Constantes:
  - Los posibles estados de la celda:
    - WATER
    - BOAT
    - TOUCH
    - SUNKEN
    - NOT_INIT
  - Las representaciónes en formato texto:
    - (blanco como que no hay nada)
    - `X` Barco
    - `T` Tocado
    - `H` Hundido
- Atributos:
  - Las coordenadas que ocupa dicha celda dentro del `Board`
  - El tipo de contenido que tiene, segun los propios estados
  - Una referencia al `Boat` que ocupa dicha celda si la celda contiene parte de un bote o **`null`** caso de no tener un barco
- Mètodos:

- constructor(es)
- `setBoat(Boat)` → asigna una porción de barco a dicha celda
- `setTouch()` → cuando se ha disparado en esa celda y estaba ocupada
- `setSunken()` → cuando se ha disparado en todas las casillas que ocupaba un barco
- `getters` → sólo los necesarios

3. Boat

La clase que representa a un barco. Contendrá:

- Constantes:
  - Estado general del barco, pudiendo ser:
    - OK
    - TOUCHED
    - SUNKEN
  - Horientación del barco dentro del tablero:
    - HOR
    - VER
- Atributos:
  - Dimensión y Orientación del barco
  - Estado del mismo, segun las constantes anteriores
  - Array con todas las `Cell` que ocupa el barco
- Métodos:
  - Constuctor(es)
  - `boolean boatFit(int f, int c, Board)` → comprueba si cabe el barco a partir de la posición indicada en el tablero pasado. También comprueba que no hay ningun barco adyacente a ninguna de las posiciones que ocupará el barco.
  - `void setBoat(dim,Board)` → asigna al barco la dimensión indicada y lo posiciona en una posición aleatoria del tablero, en la horientación que tiene el barco, comprobando que cabe y no hay adyacentes con el método anterior
  - `int touchBoat(int f, int c)` → cae la bomba sobre el barco en la posición indicada. Retornará el estado en que ha quedado el barco. Notar que como mínimo será tocado, pero puede pasar a hundido.

4. Board

Controla la lógica de todo el juego. Contendrá:

- Constantes:
  - Las dimensiones del tablero: 10 x 10
  - El número de barcos: 5
  - La longitud de cada uno de los barcos: `{5,4,3,2,2}`
- Atributos:
  - El tablero del juego: será una matriz de las dimensiones definidas como constantes. Los elementos de la matriz seran de tipo `Cell`, que representan cadauna de las celdas del juego
  - Un array de `Boat`, para tener acceso directo a todos los barcos que se han colocado en el tablero.
- Mètodos:
  - constructor(es)
  - `initBoats()` → crea els `Boats` y los posiciona de manera aleatórea
  - `getters` → sólo los necesarios
  - `shot(int fila,int columna)` → dispara sobre la casilla indicada. Retornará la información de la `Cell` de dicha possición.
  - `shot(i,j)` → se dispara una bomba sobre la posición indicada. Retornara la inforamción de la celda o del barco que está en dicha celda
  - `paint` → pinta el tablero por pantalla. Haremos la práctica mostrando la información por consola.

5. main

Debe implementarse este main de manera que funcione.

```
public class WarShip {

    final static int MAX_JUGADAS = 50;

    public static void main(String[] args) {
        // TODO code application logic here
        Board board = new Board();
        board.initBoats();
        board.paint();
```

```
        // Vamos a realizar 50 jugadas aleatorias ...
        for (int i = 0; i < MAX_JUGADAS; i++){
            System.out.println("JUGADA: " +  i);
            int fila = new Double((Math.random()*(Board.BOARD_DIM))).
               intValue();
            int columna = new Double(Math.random()*(Board.BOARD_DIM)).
               intValue();
            if (board.shot(fila, columna) != Cell.CELL_WATER)
                board.paint();
            else
               System.out.println("(" + fila + "," + columna +") --> AGUA"
                  );
        }
    }
}
```

5.1 Ampliaciones:

1. Crear un menú de manera que ofrezca la posibilidad que:
    1. Juege el ordenador (el main anterior). Modificar el main anterior para que se evite la repetición de lanzamientos, mostrando en cada jugada el tablero y marcando con una `o` las casillas donde el ordenador ya ha lanzado una bomba.
    2. Es el usuario quien juega. El jugador irá dando posiciones por teclado (A1, B8 . . .) como en el típico juego, donde la letra es la fila y el número la columna. El juego acaba cuando el jugador se rinde, mediante el input `00` ('cero cero'), cuando se hunden todos los barcos o cuando se llega a un máximo de 50 jugadas.