|  |
| --- |
| Acerca del Diseño |
| En cuanto al diseño de la práctica se ha mantenido la línea de la práctica anterior, por lo que sólo destacaremos las novedades con respecto a ella:   * La clase UIConsola, ahora extiende a una clase abstracta nueva llamada UI. Hemos elegido clase abstracta, porque la consideramos más apropiada, ya que en ella tenemos los atributos \_debug (que indica si el modo de ejecución es en debug o no) y el \_tableroInicial, con la disposición de los barcos. Además, algunos métodos (los más generales) los hemos implementado en la claseUI, para poder hacer herencia de código en las clases hijas. * La clase JugadorHumano, ahora forma parte de un árbol de clases encabezado por una nueva clase abstracta Jugador. De nuevo, hemos elegido una clase abstracta, porque aunque en este caso no tenemos atributos en la clase madre, si que implementamos los métodos más generales, que se repiten en las clases hijas (el método posValida), así que como en el caso anterior hacemos herencia de código. Además se ha introducido una clase abstracta intermedia, llamada JugadorSobreIA, que implementa un método que tienen, las clases hijas (JugadorIA, y JugadorIAMejor) que es el método rellenaPosAleatoriamente. De esta forma evitamos que la clase Jugador herede esta ultima función ya que no la necesita. * Para comprobar el entorno despejado, simplemente se ha añadido una función nueva en la clase Tablero y se ha colocado la comprobación en el lugar pertinente. |
| Parte Opcional |
| Hemos implementado la parte opcional de hacer una IA más inteligente en la clase llamada JugadorIAMejor. Hemos intentado que la IA, pensara de manera parecida a como lo haría un humano, de tal forma, que cuando se toca un barco, en lugar de directamente solicitar las casillas que ocupa el barco e ir tocándolas una a una, hacemos que busqué aleatoriamente hacia donde puede estar colocado el barco. Así cuando identifica la dirección que parece seguir, dispara en esa dirección hasta hundir el barco o hasta darse cuenta de que pudo haber tocado el barco en una posición intermedia, y debe invertir la dirección para hundirlo.  Básicamente hemos tratado de diferenciar los posibles casos en los que tiene que “pensar” la IA de la siguiente manera:   * Si es el primer disparo o en el último disparo ha hecho agua, y no tiene ningún barco tocado, dispara aleatoriamente. * Si por el contrario ya ha hundido un barco en el anterior disparo, también generará un nuevo disparo aleatoriamente.   \*Las posiciones aleatorias se generan teniendo en cuenta que no haya ningún barco ya visible (o sea hundido) adyacente, que la casilla no se haya disparado todavía, y que esté dentro del tablero)   * Y si estamos en un momento en el que tenemos tocado un barco, distinguiremos el disparo. Si tocamos un barco por primera vez, es tras un disparo aleatorio, así que como uno no sabría en qué dirección seguir disparando generamos una posición adyacente (norte, sur, este u oeste) aleatoriamente, para que la IA pruebe a ver si el barco va por ahí. Si en ese disparo se vuelve a tocar, se continúa disparando en esa dirección hasta hundirlo, y si se hace agua, se generará una adyacente distinta hasta dar con la dirección correcta, y una vez encontrada, se irá disparando a la casilla siguiente hasta hundirlo. Si en este proceso se identifica que siguiendo las casillas siguientes en esa dirección se hace un agua, y el barco todavía no está hundido, significa que el primer disparo se realizo en una posición del barco intermedia, así que se continua disparando en la dirección opuesta a la casilla siguiente de la que se toco en primer lugar, para hundir definitivamente el barco. |