INFORME BATTLESHIP

Ingeniería del Software

Mikel Abad

Andima Freire

Julen Mendiguren

Índice

Introducción

Este proyecto consiste en la realización del conocido juego Battleship o Hundir la flota con modificaciones en el comportamiento del mismo, la inclusión de nuevo armamento, un almacén, una tienda y la posibilidad de reparar tus propios barcos.

Los objetivos del proyecto son entre otros: aprender a trabajar en un grupo con técnicas de organización como las metodologías ágiles, aprender a incorporar patrones de diseño en nuestras aplicaciones de manera eficiente y profundizar en la importancia del diseño de un modelo previo a su implementación.

En el proyecto se ha utilizado la metodología SCRUM, dividiendo el proyecto en diferentes historias de usuario y sprints:

Sprint 1: 26 de Marzo
HU1: Inicializar el juego - Colocar los barcos de "flota jugador" solicitando al jugador las posiciones en las que desea situar sus barcos. Es necesario que las posiciones en las que se sitúen los barcos respeten los dos requisitos especificados en el enunciado general. - Colocar los barcos de "flota ordenador" en posiciones obtenidas aleatoriamente. Es necesario que las posiciones en las que se sitúen los barcos respeten los dos requisitos especificados en el enunciado general. - Asociar a ambas flotas el armamento y el dinero inicial. - Establecer el número de consultas del radar - Inicializar la información de "flota adversario" del jugador y del ordenador - Inicializar el almacén con los distintos tipos de armamento que tiene, las unidades disponibles y su precio unitario. - Establecer el precio de las reparaciones de los barcos.
HU2: Activar escudo jugador Si el jugador dispone de algún escudo, se activa sobre el barco que indique el jugador.
HU3: Activar escudo ordenador Si el ordenador dispone de algún escudo, se activa sobre el barco que indique el ordenador.

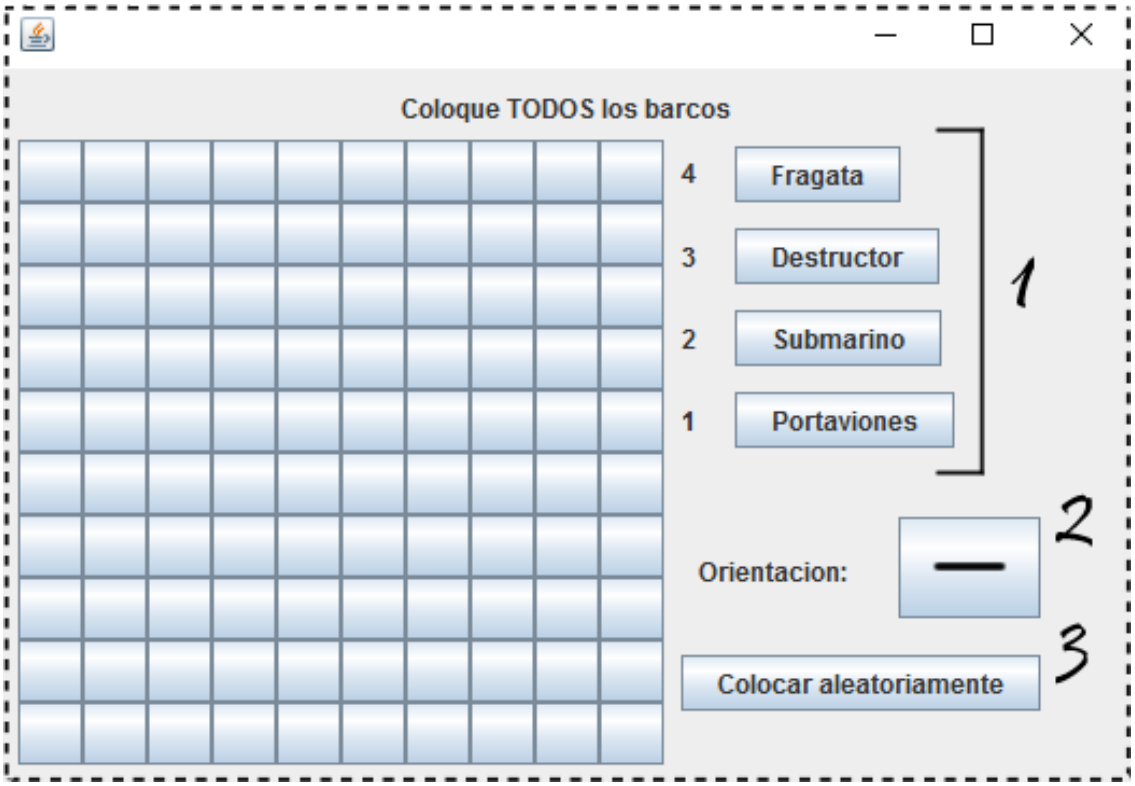
Sprint 2: 22 de Abril
HU4: Consultar radar jugador El jugador consulta su radar para comprobar si hay algún barco de la "flota ordenador" en las posiciones que rodean al radar de jugador. En caso afirmativo, devolverá una posición del barco detectado. Antes de hacer la consulta puede desplazar el radar a otra posición del tablero.
HU5: Consultar radar ordenador El ordenador consulta su radar para comprobar si hay algún barco de la "flota jugador" en las posiciones que rodean al radar de ordenador. En caso afirmativo, devolverá una posición del barco detectado. Antes de hacer la consulta puede desplazar el radar a otra posición del tablero.
HU6: Disparar el jugador El jugador indica las coordenadas sobre las que desea disparar y el armamento que desea utilizar; el gestor del juego determina el efecto del disparo sobre la "flota del ordenador". Además muestra en pantalla su resultado.
HU7: Disparar el ordenador El ordenador decide las coordenadas sobre las que va disparar y el armamento que desea utilizar; el gestor del juego determina el efecto del disparo sobre la "flota del jugador". Además, se muestra en pantalla su resultado. En una primera versión del juego, es posible establecer que el ordenador elige aleatoriamente las posiciones sobre las que realiza el disparo. Después se puede ir refinando esta estrategia de juego para añadirle algún tipo de conocimiento y razonamiento sobre estrategias de juego.

Sprint 3: 14 de Mayo
HU8: Reparar barco jugador Si el jugador dispone de dinero suficiente, se realiza la reparación del barco que indique, disminuyendo la cantidad de dinero del jugador.
HU9: Reparar barco ordenador Si el ordenador dispone de dinero suficiente, se realiza la reparación del barco que indique, disminuyendo la cantidad de dinero del ordenador.
HU10: Comprar armamento jugador Si el jugador desea comprar algún armamento y dispone de dinero suficiente, pasa a tener el armamento comprado disminuyendo la cantidad de dinero del jugador.
HU11: Comprar armamento ordenador Si el ordenador desea comprar algún armamento y dispone de dinero suficiente, pasa a tener el armamento comprado disminuyendo la cantidad de dinero del ordenador.

Descripción y requisitos del juego

En este juego el usuario se enfrentará a nuestra IA (Inteligencia Artificial) en una batalla naval. El objetivo del juego es destruir la flota enemiga antes de que la propia sea hundida.

El usuario debe colocar sus barcos de uno en uno en su tablero seleccionando el tipo de barco [1] y la orientación que desee [2], o también puede optar por una colocación aleatoria por parte de la aplicación [3].



Las dimensiones de nuestro tablero actual son 10x10, haciendo un total de 100 casillas disponibles. El número total de barcos a colocar es 10. Los barcos son los siguientes:

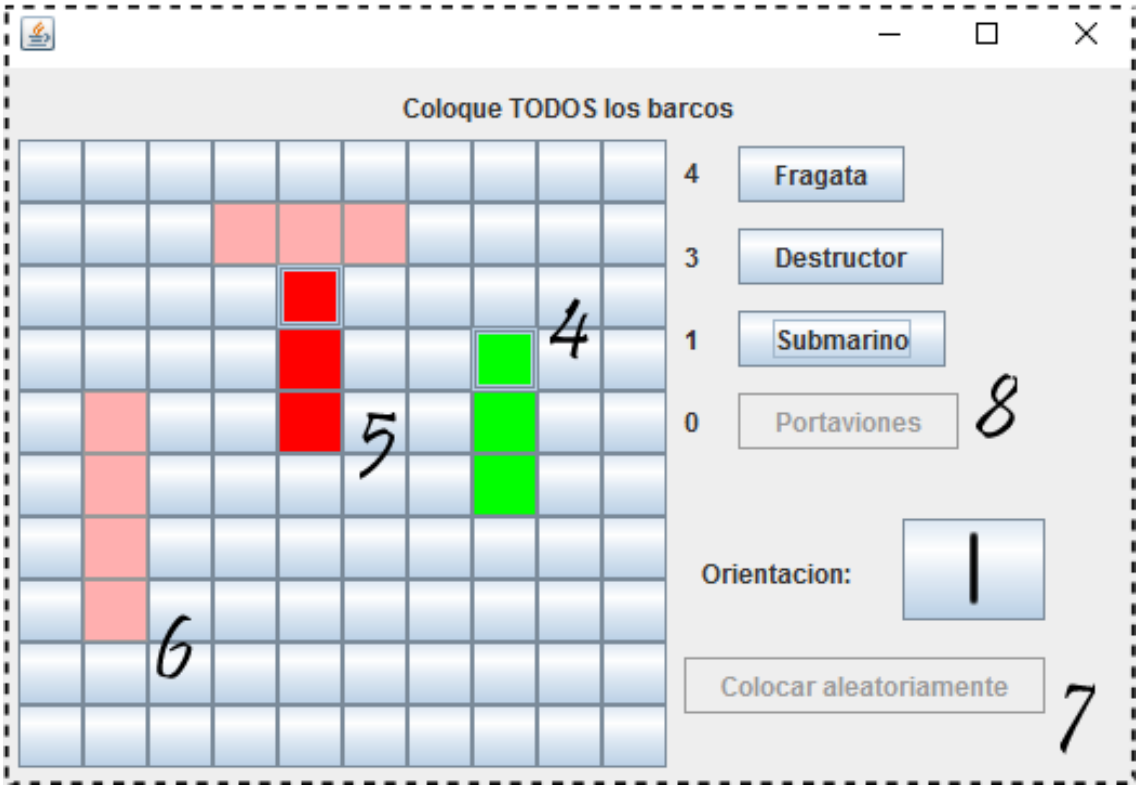
- 4 fragatas de 1 casilla.
- 3 destructores de 2 casillas.
- 2 submarinos de 3 casillas.
- 1 portaviones de 4 casillas.

Cuando se hayan seleccionado ambas componentes la aplicación iluminará, mientras nos desplazamos por el tablero, las casillas que se corresponden al barco que vamos a colocar. Si éste está en verde significa que la posición es correcta y puede ser colocado [4]. En cambio, si es rojo, significa que la posición no está disponible [5] e intentar colocarlo dará un mensaje de error.

Un barco no podrá ser colocado bajo las siguientes circunstancias:

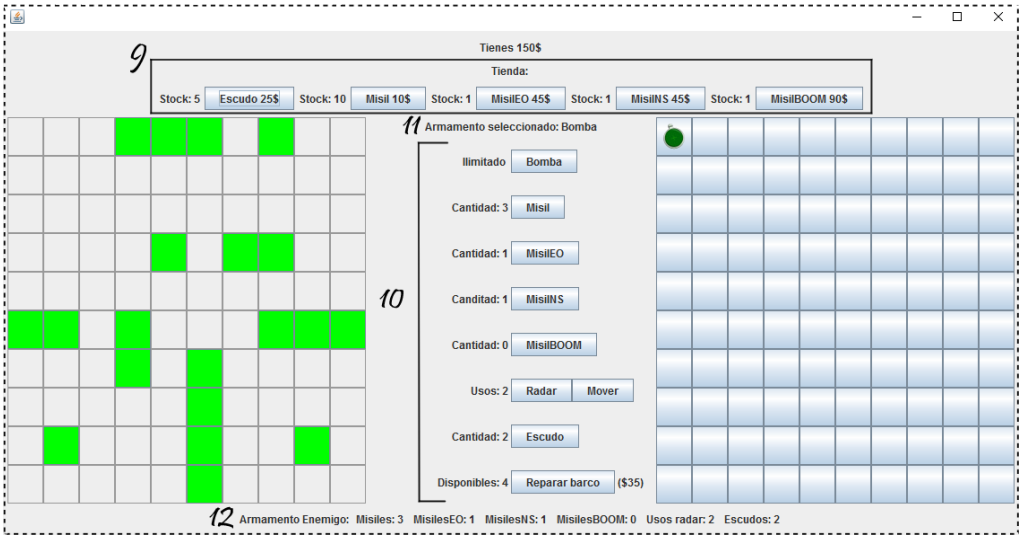
- Alguna de sus casillas se encuentra fuera de los límites del tablero.
- Alguna de sus casillas se encuentra entre las adyacentes de otro barco.

Cuando un barco ha sido colocado sus casillas se mostrarán en rosa [6] y afectarán a la colocación de los siguientes barcos. Una vez se coloca el primer barco, el botón de colocación aleatoria queda desactivado [7], ya que es una opción únicamente para colocar la flota completa. Una vez colocado el número máximo de un tipo de barco su botón correspondiente quedará desactivado [8].

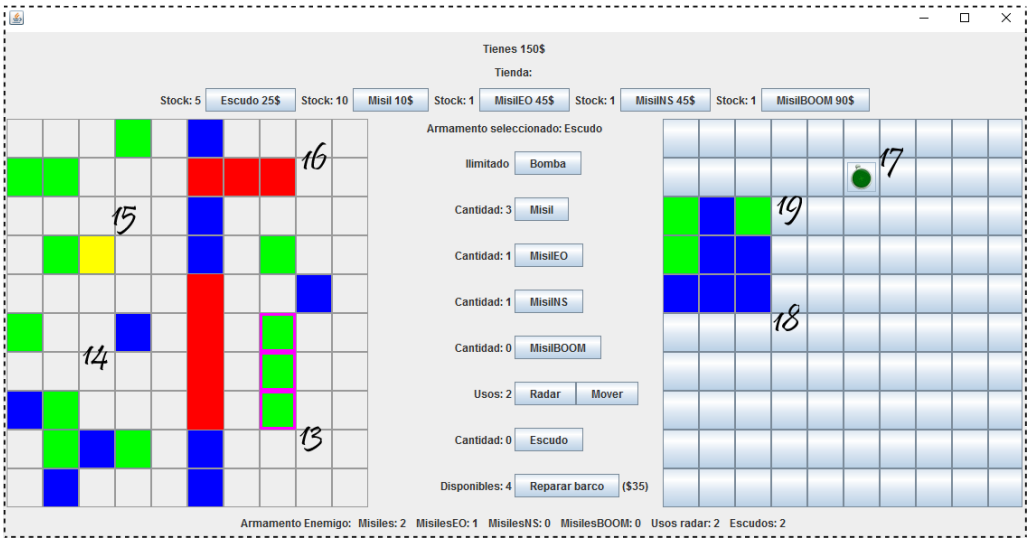


Una vez se hayan colocado todos los barcos el ordenador colocará los suyos automáticamente y pasaremos a la siguiente ventana en la que jugaremos la partida. En ella se encuentran ambos tableros desde el punto de vista del usuario, la tienda donde comprar munición, sus botones para utilizar el armamento, su dinero actual y la cantidad de munición de la que dispone el enemigo.

En esta nueva ventana, la tienda se encuentra en la parte superior [9]. Contiene información del stock restante y del precio de cada arma. En vertical se encuentran los botones que el usuario utilizará para seleccionar la acción que desee realizar [10]. Cuando uno de estos botones es presionado, el texto que indica la opción seleccionada cambiará [11]. De esta manera el usuario sabe en todo momento qué tiene seleccionado, ya que se mantiene después de utilizarlo. En la parte inferior se encuentran las unidades de las que dispone el enemigo [12].



El tablero de la izquierda es el del usuario, donde se observa el estado de los barcos. Los barcos protegidos por un escudo estarán rodeadas por un borde morado [13] que se mantendrá hasta que el escudo sea destruido. Si un disparo enemigo cae sobre una posición vacía, se pintará de color azul [14]. Cuando una posición de un barco es tocada, ésta cambia a color amarillo [15]. En caso de que todas las posiciones de un barco sean tocadas, el barco será destruido y se pondrá de color rojo [16]. Es lo mismo que ocurre si un misil alcanza alguna de las posiciones de un barco.



Por ende, el tablero de la derecha se corresponde a la información que nosotros tenemos sobre el enemigo. Los colores siguen el mismo esquema que

el anterior, sólo que al no disponer de tanta información hay algunos casos que se reducen. Por ejemplo, si disparamos a una casilla y se desvela en verde en vez de amarillo, significa que ese barco está, o estaba antes de nuestro disparo, protegido por un escudo.

La otra diferencia es la funcionalidad de nuestro radar [17]. Cuando lo utilizamos para escanear, desvela todas las posiciones adyacentes como azules [18], en caso de estar vacías, o como verdes [19], en caso de haber un barco oculto.

El dinero inicial del que se dispone para realizar las compras es de 150$ y las opciones disponibles en el juego son las siguientes, en orden descendente de aparición en la interfaz de usuario:

- **Bomba:** Gratuita y de uso ilimitado.
  Afecta a una única posición y tiene un valor de daño de uno, lo que tocará la casilla si está desprotegida o restará un punto de vida al escudo si es que el barco objetivo tiene uno.
- **Misil:** Coste 10$, munición inicial 3 u. y stock en tienda 10 u.
  El misil destruye automáticamente el barco objetivo o, si tiene un escudo, su protección completa.
- **Misil Norte-Sur:** Coste 45$, munición inicial 1 u. y stock en tienda 1 u.
  Dispara un Misil por cada casilla disponible en la columna donde sea disparado. Si atraviesa más de una casilla de un mismo barco y éste tiene escudo, solamente le destruirá la protección y no el barco.
- **Misil Este-Oeste:** Coste 45$, munición inicial 1 u. y stock en tienda 1 u.
  Tiene el mismo funcionamiento que el Misil Norte-Sur pero a lo largo de una fila, es decir, en horizontal.
- **Misil BOOM:** Coste 90$, munición inicial 0 u. y stock en tienda 1 u.
  Su efecto es el equivalente a disparar un Misil Este-Oeste y un Misil Norte-Sur en la misma casilla.
- **Escudo:** Coste 25$, munición inicial 2 u. y stock en tienda 5 u.
  Protege el barco objetivo con un escudo que resiste un impacto de cualquier armamento potente o dos impactos de bomba.
- **Reparación:** Coste 35$, usos disponibles hasta falta de efectivo.
  Repara el barco objetivo eliminando de cada casilla el estado de "tocada". No se puede reparar un barco destruido.
- **Mover Radar:** Gratuito y de uso ilimitado.
  Desplaza el radar a la posición seleccionada.
- **Usar Radar:** Gratuito, dos usos disponibles sin posibilidad de compra.
  Escanea las casillas adyacentes a la posición del radar, incluida la propia, para desvelar los barcos enemigos en el área.

Desarrollo de una partida

La partida se desarrolla por turnos, comenzando siempre el usuario. Al comenzar su turno se tienen diferentes posibilidades:

- **Comprar:** Cada jugador puede realizar todas las compras que desee, dentro de los límites de stock y dinero, durante su turno sin que éste cambie. Los botones de compra restan una unidad al stock del almacén, añaden esa unidad a las reservas del comprador y le restará el dinero pertinente.
- **Disparar:** Cualquier tipo de disparo al tablero enemigo consumirá el turno del atacante una vez se haya desvelado el resultado de dicho disparo.
- **Colocar escudo:** Colocar un escudo a un barco propio para protegerlo. Esta acción acaba con su turno.
- **Reparar:** Reparar un barco dañado también acaba el turno. El enemigo sigue sabiendo la posición del barco después de repararlo.
- **Radar:** El radar puede ser movido o utilizado sin pasar turno. Es posible mover el radar de manera ilimitada durante tu turno.

Tras realizar las acciones pertinentes el ordenador jugará su turno. Nuestra Inteligencia Artificial está basada principalmente en la aleatoriedad. Lo primero que hace al comenzar su turno es decidir su acción. Hay dos procesos principales que se repiten: comprobar si puede utilizar y decidir dónde utilizar.

En el proceso de comprobación, el ordenador mira si tiene suficiente munición del tipo que esté intentando usar. En caso negativo, intentará comprar una unidad de ese tipo. Si tiene suficiente dinero y hay stock en el almacén, comprará y utilizará dicho elemento. En caso negativo simplemente decidirá utilizar una bomba común. En ambas situaciones pasará al proceso de decisión con el armamento seleccionado.

El proceso de decisión varía en función del armamento seleccionado:

En el caso de las armas ofensivas, el ordenador comprobará una lista dónde guarda coordenadas relevantes a las que debe disparar, como por ejemplo una coordenada donde haya disparado pero tuviera escudo, una coordenada que haya tocado pero ha sido reparada o una coordenada detectada por el radar. En el caso de que dicha lista contuviera alguna coordenada, escogerá aleatoriamente una de ellas donde utilizar su arma. Si por el contrario ésta está vacía, el ordenador escogerá una coordenada aleatoria dentro de los límites del tablero y que no se encuentre en su lista de coordenadas a las que no debe disparar, como coordenadas de barcos destruidos y sus adyacentes o coordenadas donde ya ha detectado que no hay nada.

En el caso de que quiera mover su radar, símplemente escogerá una coordenada aleatoria del tablero enemigo. Para utilizar el radar comprueba que aún tenga usos y lo utiliza en caso afirmativo. Independientemente de si lo usa o no, vuelve a decidir otra acción, ya que su turno no finaliza desplazando o utilizando el radar.

Si lo que desea es poner un escudo o reparar un barco, buscará posiciones

aleatorias en su tablero donde pueda proteger o reparar un barco. Si tras 10 coordenadas revisadas no encuentra ninguna en la que pueda hacerlo, ya sea por la coordenada en sí o porque no dispone de munición o dinero, pasará a disparar una bomba.

La decisión del ordenador se decidirá en base a las siguiente tabla de probabilidades:

Disparar una bomba	0.65
Disparar un Misil normal	0.14
Disparar un Misil NS	0.0525
Disparar un Misil EO	0.035
Poner un escudo	0.035
Mover su radar	0.0175
Reparar un barco	0.0175
Disparar un Misil BOOM	0.0175
Utilizar el radar	0.0175

Patrones de diseño

Singleton:

El patrón singleton (*instancia única* en inglés) es un patrón de diseño creado para restringir la creación de objetos de una clase.

Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

En nuestro proyecto, este patrón se ha usado para las clases Battleship y Almacén, que gestionan todas las acciones y hacen de puente entre el modelo y la vista de forma que éstas sean independientes, ya que la vista solo puede acceder al juego mediante las instancias únicas, y BarcosFactory para la creación de los barcos.

Factory:

Este patrón nos permite que una clase, la factoría, se encargue de la creación de ciertas instancias. Nos aporta modularidad en la creación de objetos delegando en ella la tarea, así como facilidad a la hora de añadir nuevos tipos del mismo.
En el proyecto, éste patrón lo hemos utilizado a la hora de crear los diferentes tipos de barco. De esta manera, para introducir un nuevo tipo de barco, solo habría que crear una clase que extienda la clase abstracta barco y añadirlo en el Factory.

Observer:

El patrón Observer es un patrón de diseño que define una dependencia del tipo *uno-a-muchos* entre objetos, de manera que cuando uno de los objetos cambia, notifica este cambio a todos los observadores.

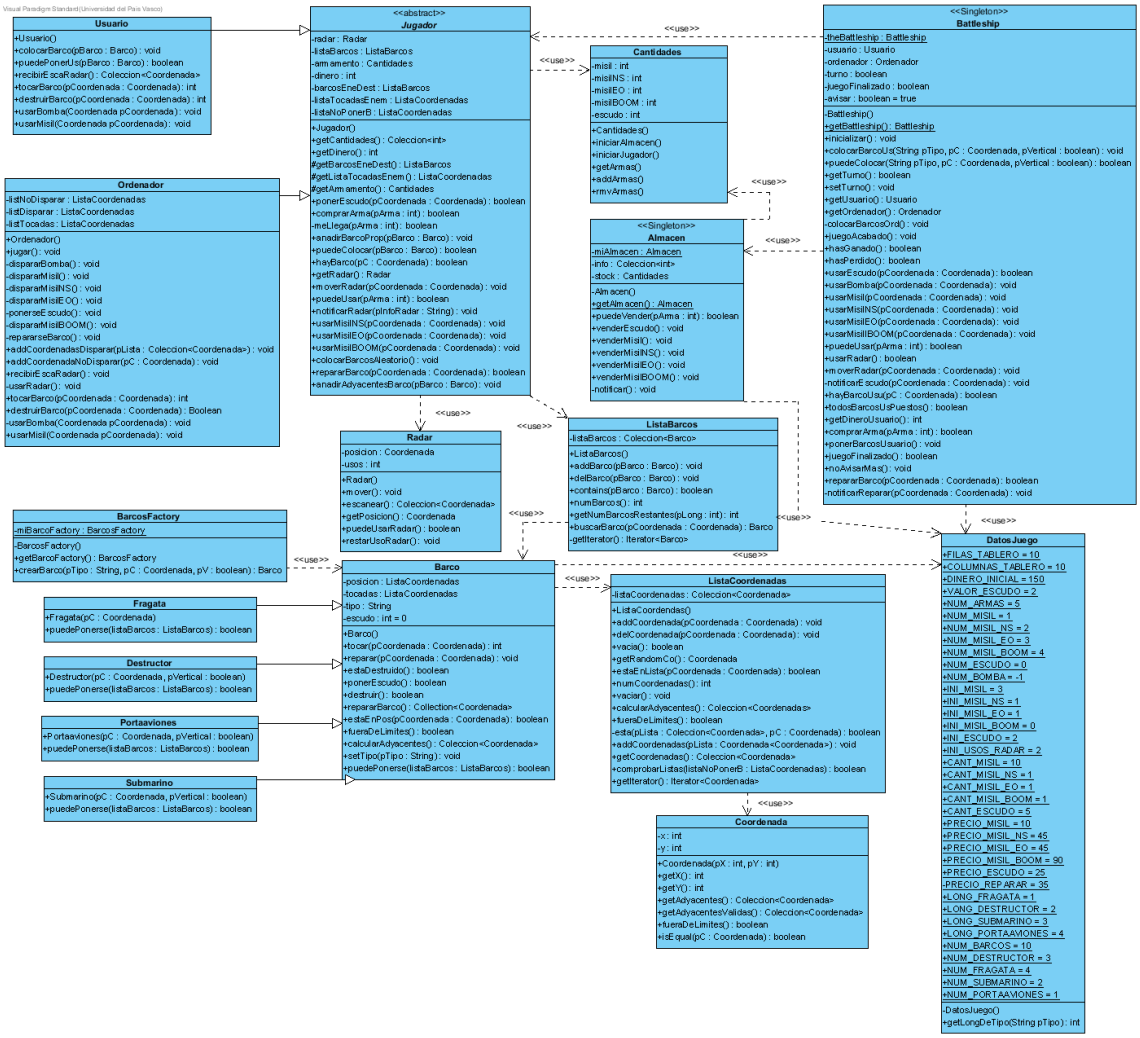
El patrón Observer es la clave del patrón de arquitectura Modelo Vista Controlador (MVC) que hemos utilizado en nuestro proyecto.

En nuestro proyecto, el patrón observer se ha usado para que cada vez que el modelo cambie, cambie la vista acorde a él, de manera que cada vez que el estado de un barco, del tablero, del dinero, del almacén o de las armas cambia, se refleja en la vista. Este patrón también ayuda a mantener la independencia entre la vista y el modelo. **BarcoNoEncException**

BarcoNoEncException:

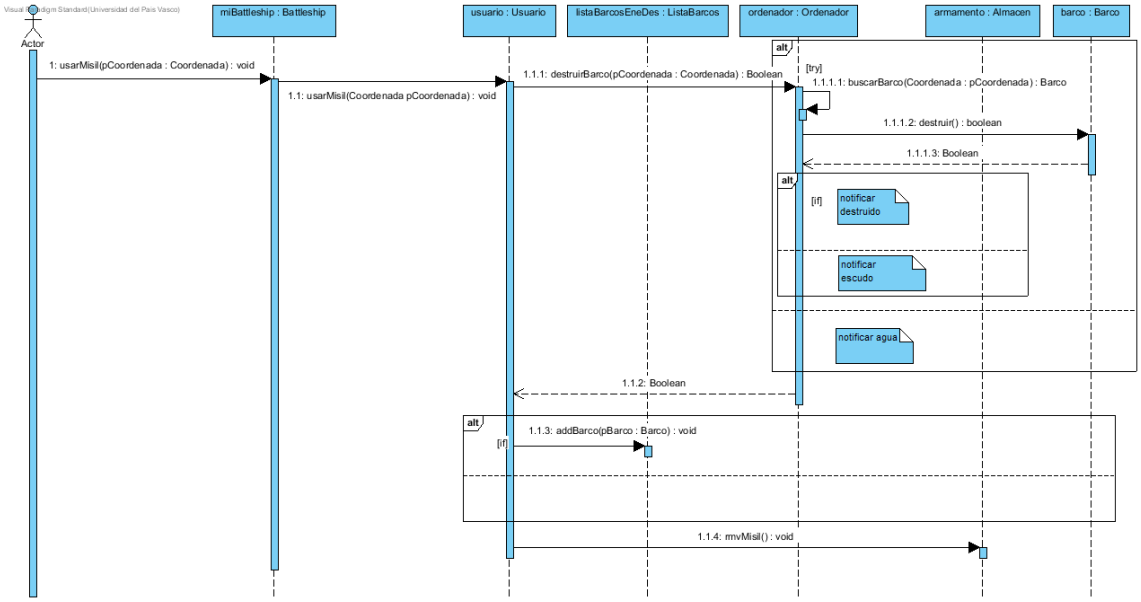
Ésta es la excepción que hemos incorporado en nuestro proyecto y que lanzamos cuando algún método que busca un barco no es capaz de encontrar uno.

Diagrama de clases

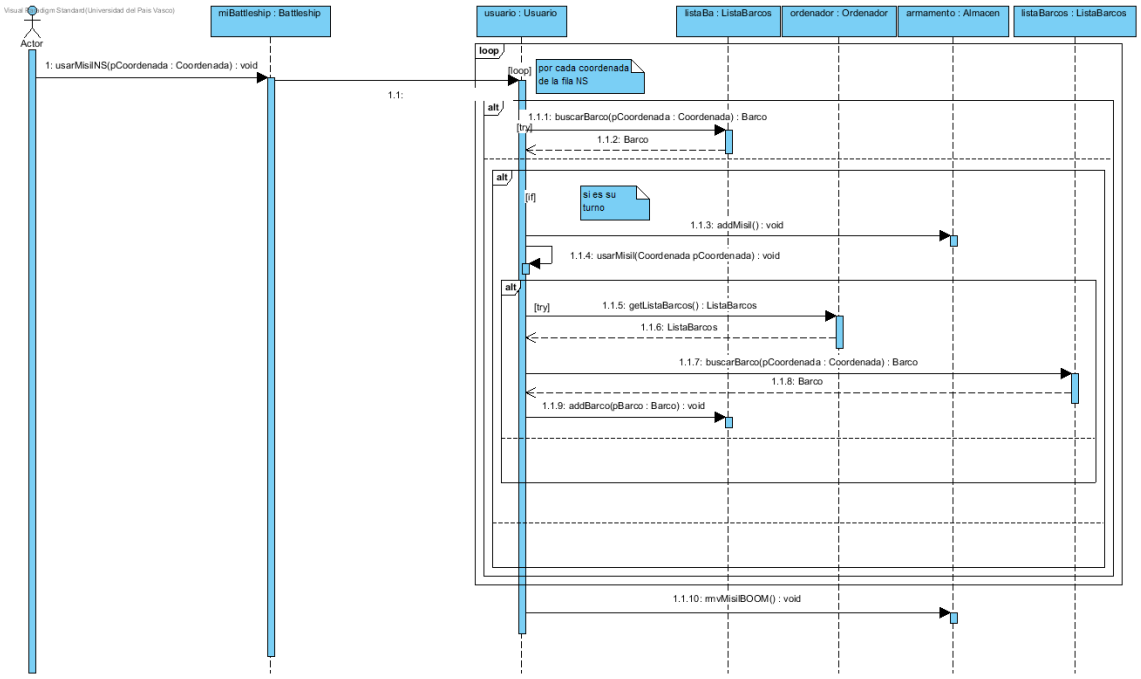


Diagramas de secuencias

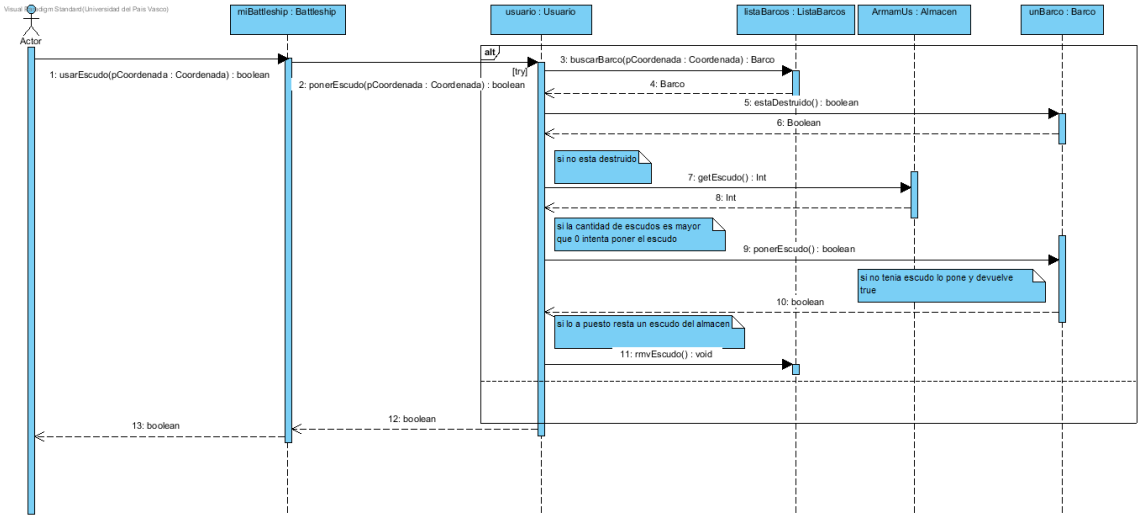
- Diagrama de secuencia usarMisil



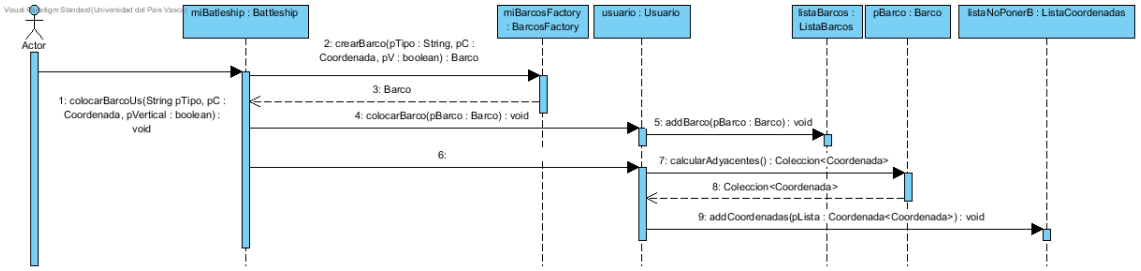
- Diagrama de secuencia usarMisilNS



- Diagrama de secuencia usarEscudo



- Diagrama de secuencia colocarBarcoUs



Pruebas

Para comprobar el correcto funcionamiento del proyecto hemos realizado una serie de testeos utilizando las pruebas unitarias JUnit. Hemos hecho una clase JUnit para testar cada una de las siguientes clases:

- Almacen
- Barco
- Battleship
- Coordenadas
- ListaCoordenadas
- Jugador
- Usuario

Almacen: Hemos comprobado el correcto funcionamiento del método puedeVender, para ello lo hemos ejecutado seleccionando distintas armas y con distintas cantidades de stock.

Barco: Probamos que el método estaEnPos funciona correctamente, utilizando distintas coordenadas que forman o no parte del barco, o incluso que pertenecen a otro barco. También hemos comprobado el método fueraDeLimites, con coordenadas que pertenecen o no al tablero. Por último probamos el método calcularAdyacentes, para ello comparamos las coordenadas del barco y las de alrededor con las que nos devuelve el método.

Battleship: De esta clase probamos dos métodos referentes a la colocación de barcos por parte del usuario, el puedeColocar y colocarBarcoUs. Para ello hemos probado diversas configuraciones, barcos que entran en las coordenadas adyacentes de otros, que se salen de los límites, que se solapan, etc.

Coordenada: En esta clase probamos el método de getAdyacentes, comparando manualmente las coordenadas que debería devolver con las que devuelve.

ListaCoordenadas: Comprobamos que estaEnLista funciona correctamente, con coordenadas tanto que están como que no. También comprobamos que calcularAdyacentes nos devuelve las coordenadas adyacentes de las coordenadas de la lista, sin repetir ninguna coordenada. Probamos también fueraDeLimites, con distintos tipos de listas de coordenadas: vacías, con un elemento fuera de los límites, varios, ninguno… Por último comprobamos que comprobarListas devuelve true si y sólo si alguna de las coordenadas de las listas está en ambas, también comprobamos con todo tipo de listas.

Jugador: De la clase jugador comprobamos los métodos: puedeColocar y comprarArma, que son los métodos genéricos para comprobar si un jugador puede colocar un barco en su tablero y comprar un arma. puedeColocar lo comprobamos con distintas situaciones de tablero y de barcos (Barcos fuera de límites, se superan el número de barcos de un tipo permitido, se solapan coordenadas de barcos…). Para comprar arma probamos también distintas situaciones de stock y de dinero.

Usuario: Probamos puedeColocarUs comprobando si el usuario puede colocar un barco en diferentes situaciones del tablero y de los barcos colocados. También comprobamos los métodos de usar armamento usarMisil, usarMisilNS, usarMisilEO y usarMisilBOOM. Los disparamos con distintas situaciones del tablero de la máquina, como barcos con escudos, tocados, ya hundidos, etc. y contrastados los resultados con los que esperamos de cada disparo de los misiles.

Herramientas adicionales

GitHub:

Para la sincronización del código entre los diferentes miembros del grupo hemos utilizado la plataforma GitHub, donde hemos creado un proyecto que hemos utilizado a través del programa SourceTree.

Link del proyecto: https://github.com/MikelAbad/Battleship

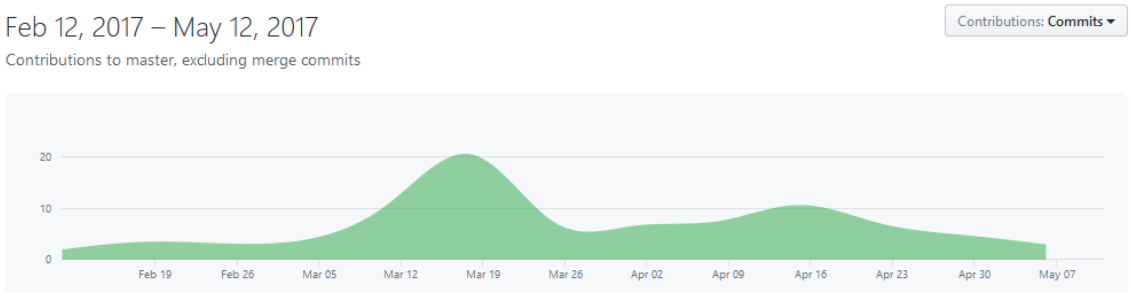
Miembros del grupo:

https://github.com/MikelAbad

https://github.com/andimafreire

https://github.com/JulenMendiguren

Actividad del ultimo mes:



Código añadido y eliminado por semana: