

Informe Sprint 1

Ingeniería del Software



Mikel Abad

Andima Freire

Julen Mendiguren

Jon Ander González

26 de marzo de 2017

GRUPO 10 Abad y cia

Índice:

**Introducción2**

**Decisiones del juego3**

**Actas de reunión de grupo5**

Acta reunión 23 / 02 / 17. 5

Acta reunión 02 / 03 / 17. 6

Acta reunión 09 / 02 / 17. 7

Acta reunión 16 / 02 / 17. 8

Acta reunión 23 / 02 / 17. 9

**Reparto de tareas10**

**Diagrama de Clases11**

**Patrones de Diseño Implementados12**

**Diagramas de Secuencia 13**

Diagrama de secuencia “colocarBarco” 13

Diagrama de secuencia “ponerEscudo” 14

**Casos de Prueba 15**

# Introducción**.**

En la realización del proyecto, el juego de hundir la flota, utilizaremos la metodología SCRUM. Como parte de esta metodología el proyecto estará dividido en “Historias de Usuario y Sprints.

En este primer Sprint abordaremos las siguientes historias de usuario:

Decisiones del juego:

El jugador una vez inicie el juego dispondrá de un tablero con 100 casillas, organizadas en 10 filas y 10 columnas donde situar cada barco. Tendrá que colocar un total de 10 barcos y podrá controlar su orientación con el botón de orientación. Los barcos serán:

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

\*Para colocar los barcos tiene que respetar la distancia de una casilla entre los barcos.



Una vez colocados todos los barcos el usuario dispondrá de dos tableros, uno con sus barcos colocados y otro que mostrará el tablero del ordenador. También dispondrá de una tienda donde comprar armamento.

El dinero inicial que dispone para comprar es de 150 y podrá comprar el siguiente armamento en la tienda:

* Bomba. Gratuita y de uso ilimitado, que golpeará una única posición.
* Misil. De coste 10 y un máximo de 10 usos. El misil al golpear un barco sin escudo, destruirá el barco por completo.
* Misil Norte-Sur. Con un coste de 45 y un único uso en la partida. Destruirá todos los barcos sin escudo situados en la línea Norte-Sur, en caso de disponer de escudo se lo quitará.
* Misil Este-Oeste. Con un coste de 45 y un único uso en la partida. Destruirá todos los barcos sin escudo situados en la línea Este-Oeste, en caso de disponer de escudo se lo quitará.
* Misil BOOM. Con un coste de 90 y un único uso en la partida. Destruirá todos los barcos sin escudo situados en la línea Norte-Sur y Este-Oeste, en caso de disponer de escudo se lo quitará.
* Escudo de coste 5 y un máximo de 5 usos. Otorgará un escudo al barco entero. Si es alcanzado por un misil, el barco perderá su escudo, pero seguirá con vida.
* Reparación. Coste de 10 y un máximo de 5 usos. Reparará el barco entero siempre que el barco esté tocado. Si el barco es destruido el barco no podrá ser reparado.

|  |  |  |
| --- | --- | --- |
| **Armas** | **Cantidad** | **Precio** |
| Bomba | Ilimitada | 0 |
| Misil | 10 | 10 |
| Misil Norte-Sur | 1 | 45 |
| Misil Este-Oeste | 1 | 45 |
| Misil BOOM | 1 | 90 |
| Escudo | 5 | 25 |
| Reparación | 5 | 10 |

Actas de reunión del grupo:

## Acta reunión 23 / 02 / 17.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| blanco_medianoEUITI  INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA  ESCUELA UNIVERSITARIA DE INGENIERIA TECNICA INDUSTRIAL    ACTA DE REUNIÓN 1   |  |  |  |  | | --- | --- | --- | --- | | Fecha: *23 / 02 /2017* | Hora: *12:00* | Lugar: *Aula 3* | Duración: *3h* | | Trabajo:  *Ingeniería del software. Proyecto Hundir la flota.* | | | | | Personas Asistentes:  Mikel Abad, Andima Freire, Julen Mendiguren, Jon Ander González | | | | | **ASUNTOS TRATADOS**:  *Tareas de las HU 1 del primer sprint*  *Identificar clases necesarias* | | | | | **PRINCIPALES ACUERDOS ALCANZADOS**:  *Acordada forma de juego y estructuración de las pantallas.* | | | | | Fecha de la próxima reunión: *02 / 03 / 2017* | | | |   **FIRMAS** |

## Acta reunión 02 / 03 / 2017.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| blanco_medianoEUITI  INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA  ESCUELA UNIVERSITARIA DE INGENIERIA TECNICA INDUSTRIAL    ACTA DE REUNIÓN 2   |  |  |  |  | | --- | --- | --- | --- | | Fecha: *02 / 02 /2017* | Hora: *12:00* | Lugar: *Aula 3* | Duración: *3h* | | Trabajo:  *Ingeniería del software. Proyecto Hundir la flota.* | | | | | Personas Asistentes:  Mikel Abad, Andima Freire, Julen Mendiguren, Jon Ander González | | | | | **ASUNTOS TRATADOS**:  Diseñar clases necesarias | | | | | **PRINCIPALES ACUERDOS ALCANZADOS**:  Establecidos precios de armamento y numero de usos. | | | | | Fecha de la próxima reunión: *09 / 03 / 2017* | | | |   **FIRMAS** |

## Acta de reunión 09 / 03 / 2017

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| blanco_medianoEUITI  INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA  ESCUELA UNIVERSITARIA DE INGENIERIA TECNICA INDUSTRIAL    ACTA DE REUNIÓN 3   |  |  |  |  | | --- | --- | --- | --- | | Fecha: *09 / 03 /2017* | Hora: *12:00* | Lugar: *Aula 3* | Duración: *3h* | | Trabajo:  *Ingeniería del software. Proyecto Hundir la flota.* | | | | | Personas Asistentes:  Mikel Abad, Andima Freire, Julen Mendiguren, Jon Ander González | | | | | **ASUNTOS TRATADOS**:  Implementar y concretar clases | | | | | Fecha de la próxima reunión: *16 / 03 / 2017* | | | |   **FIRMAS** |

## Acta de reunión 16 / 03 / 2017.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| blanco_medianoEUITI  INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA  ESCUELA UNIVERSITARIA DE INGENIERIA TECNICA INDUSTRIAL    ACTA DE REUNIÓN 4   |  |  |  |  | | --- | --- | --- | --- | | Fecha: *16 / 03 /2017* | Hora: *12:00* | Lugar: *Aula 3* | Duración: *3h* | | Trabajo:  *Ingeniería del software. Proyecto Hundir la flota.* | | | | | Personas Asistentes:  Mikel Abad, Andima Freire, Julen Mendiguren, Jon Ander González | | | | | **ASUNTOS TRATADOS**:  *Implementar clases*  *Pruebas unitarias clases.* | | | | | Fecha de la próxima reunión: *23 / 03 / 2017* | | | |   **FIRMAS** |

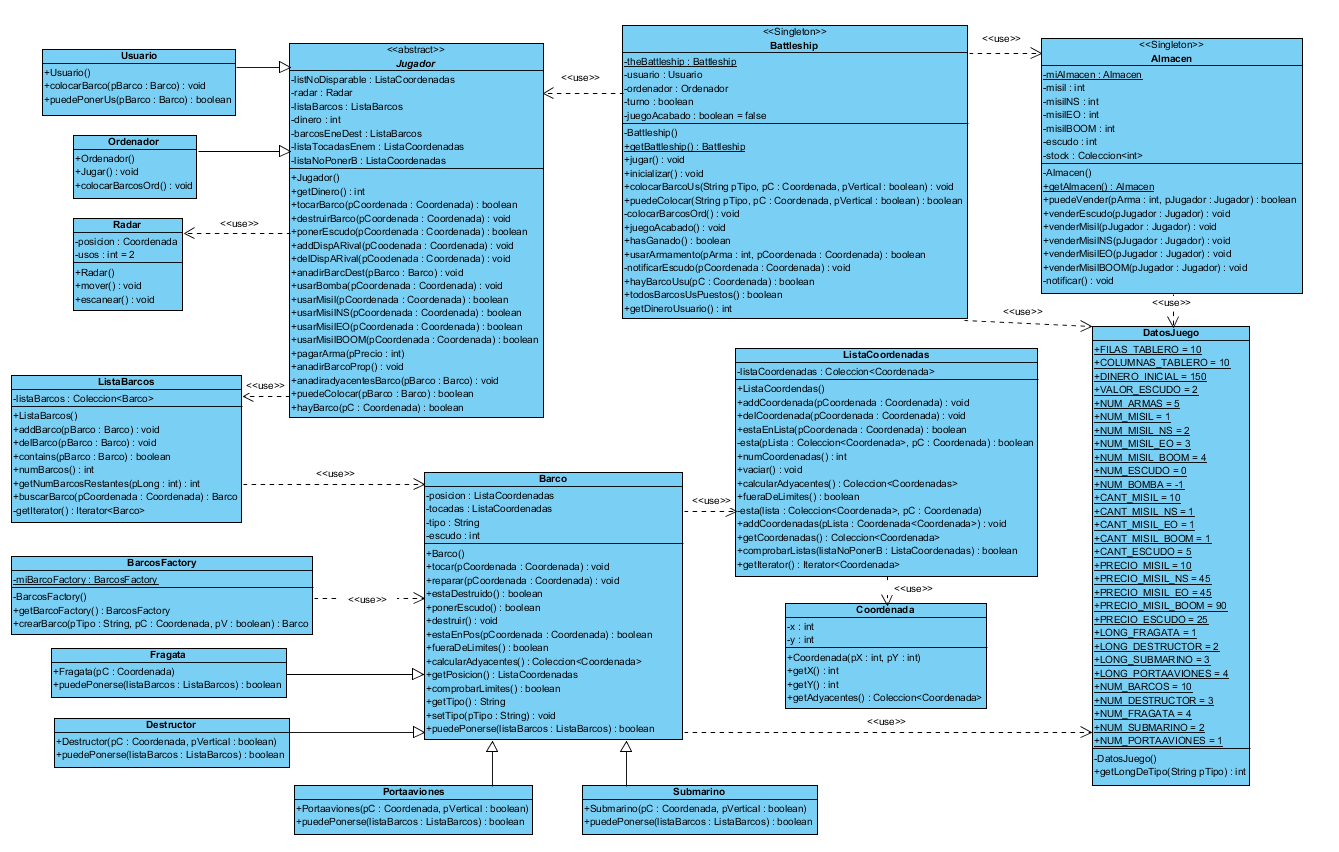
## Acta de reunión 23 / 03 / 2017

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| blanco_medianoEUITI  INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA  ESCUELA UNIVERSITARIA DE INGENIERIA TECNICA INDUSTRIAL    ACTA DE REUNIÓN 5   |  |  |  |  | | --- | --- | --- | --- | | Fecha: *23 / 03 /2017* | Hora: *12:00* | Lugar: *Aula 3* | Duración: *3h* | | Trabajo:  *Ingeniería del software. Proyecto Hundir la flota.* | | | | | Personas Asistentes:  Mikel Abad, Andima Freire, Julen Mendiguren, Jon Ander González | | | | | **ASUNTOS TRATADOS**:  *Solucionar conflictos implementación.* | | | | | Fecha de la próxima reunión: *30 / 03 / 2017* | | | |   **FIRMAS** |

Reparto de tareas:

* Documento con el reparto de tareas entre los miembros del grupo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sprint** | **Tarea** | **Responsable** | **Planif.** **Inicial** | **Planif. Real** | **Comentarios** |
| 1 | Identificar clases necesarias | Grupo | 1h | 1h |  |
| 1 | Diseñar clases necesarias | Grupo | 2h | 5h | Clases relacionadas con: |
| 1 | Implementar y concretar clases | Mikel Abad | 1.5h | 4h | Jugador/Ordenador |
| 1 | Implementar y concretar clases | Andima Freire | 1.5h | 4h | Barcos/Coordenadas |
| 1 | Implementar y concretar clases | Jon Ander González | 1.5h | 4h | Principal/Listas |
| 1 | Implementar y concretar clases | Julen Mendiguren | 1.5h | 4h | Armamento/Almacén |
| 1 | Solucionar conflictos implementación. | Grupo | 0.5h | 8h |  |
| 1 | Pruebas unitarias clases: | Mikel Abad | 2h | 2h | Barcos/Coordenadas |
| 1 | Pruebas unitarias clases: | Andima Freire | 2h | 2h | Jugador/Ordenador |
| 1 | Pruebas unitarias clases: | Jon Ander González | 2h | 2h | Armamento/Almacén |
| 1 | Pruebas unitarias clases: | Julen Mendiguren | 2h | 2h | Principal/Listas |
| 1 | Documentación del Sprint | Grupo | 5h | 5h |  |

Diagrama de clases: 

# Patrones de diseño Implementados

## Singleton:

En ingeniería de software, 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, Almacen y BarcosFactory del modelo, 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.

## Factory:

Este patrón nos permite que una clase, la factoría, se encargue de la creación de ciertos objetos. Nos aporta modularidad en la creación de objetos delegando en ella la tarea, así como facilidad de implementación de nuevos tipos de dicho objeto.

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:

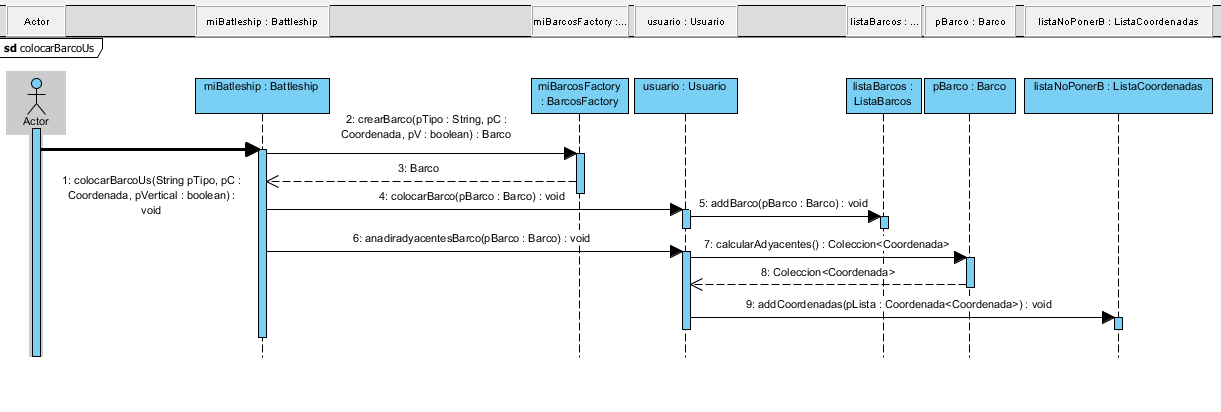
El patrón Observer es un patrón de diseño que define una dependencia del tipo *uno-a-muchos* entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes.

El patrón Observer es la clave del patrón de arquitectura Modelo Vista Controlador (MVC).

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, de un tablero, del dinero o de las armas cambia, se refleja en la vista. Este patrón también ayuda a mantener la independencia entre la vista y el modelo.

Diagramas de secuencias:

## Diagrama de secuencia “colocarBarco”.

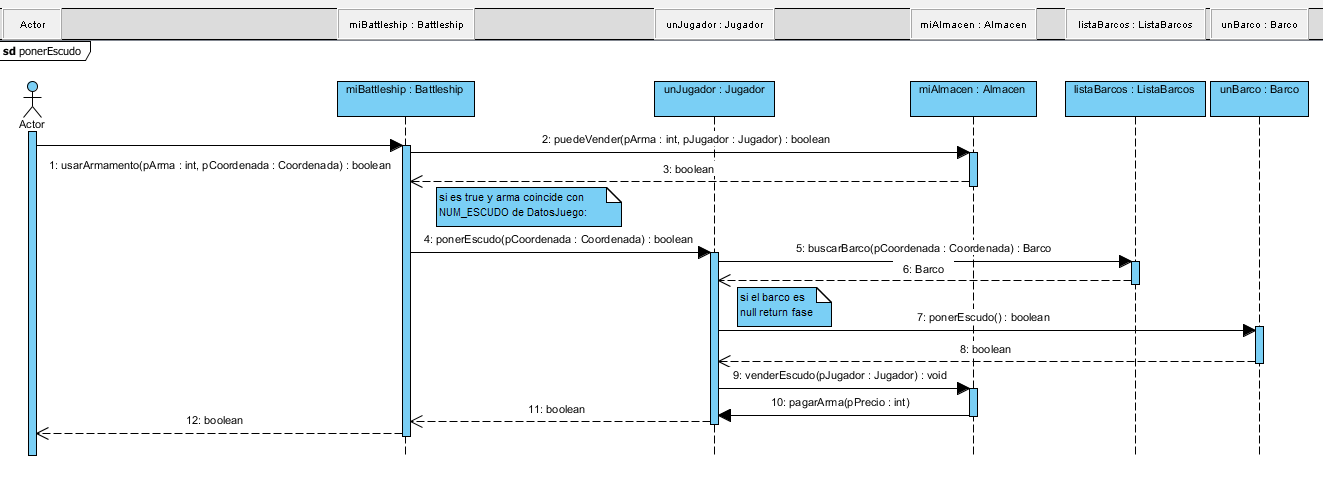


Este es el diagrama de colocarBarco. Se inicia con el método colocarBarco, al cual llamamos si puedeColocar ha devuelto un true, y le pasaremos el tipo de barco a colocar, la coordenada y su orientación, vertical u horizontal.

Con el método crearBarco solicitamos el nuevo barco al Factory con su tipo, coordenada y orientación. Después con el método colocarBarco añadiremos el barco ya creado a la lista de barcos.

Con el método calcularAdyacentes crearemos una colección de coordenadas, ya que en las casillas adyacentes a los barcos no podremos añadir nuevos barcos. Por último con el método addCoordenadas, pasaremos esa colección de coordenadas a las listaNoPonerB (lista de coordenadas donde no podemos colocar ningún barco).

## Diagrama de secuencia “usarArmamento”:



Este es el diagrama de secuencia para el método usarArmamento. En primer lugar, llamamos a Almacen mediante el método puedeVender y esté nos devolverá un boolean, ya que la cantidad de artículos es limitada e irá descendiendo con cada compra. Si nos devuelve un true el arma podrá ser vendida.

Para poner el escudo utilizaremos el método ponerEscudo, al que le pasaremos una coordenada. Con el método buscarBarco comprobará si en dicha coordenada existe un barco, si el barco no es válido devolverá null. Una vez tenemos el barco, con el método ponerEscudo, le asignará el escudo en caso de que no tenga ya.

Por último, el método venderEscudo al que le pasaremos el jugador, y pagarArma que descontará el precio del escudo al dinero del jugador.

Casos de prueba:

**Clase AlmacenTest**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id. De la prueba** | **Objetivo** | **Entrada** | **Condiciones de ejecución1** | **Resultado esperado** | **Resultado obtenido** | **Comentarios** |
| Test puedeVender 1 | devuelve true cuando el usuario tiene dinero y queda stock | El tipo de arma (escudo) y Usuario 1 | El usuario tiene dinero suficiente (100) y quedan escudos(5) | True | True |  |
| Test puedeVender 2 | devuelve true cuando el usuario tiene dinero y queda stock | El tipo de arma (escudo) y Usuario 2 | El usuario tiene dinero suficiente (100) y quedan escudos(4) | True | True |  |
| Test puedeVender 3 | devuelve true cuando el usuario tiene dinero y queda stock | El tipo de arma (escudo) y Usuario 1 | El usuario tiene dinero suficiente (75) y quedan escudos(3) | True | True |  |
| Test puedeVender 4 | devuelve true cuando el usuario tiene dinero y queda stock | El tipo de arma (escudo) y Usuario 1 | El usuario tiene dinero suficiente (50) y quedan escudos(2) | True | True |  |
| Test puedeVender 5 | devuelve true cuando el usuario tiene dinero y queda stock | El tipo de arma (escudo) y Usuario 1 | El usuario tiene dinero suficiente (25) y quedan escudos(1) | True | True |  |
| Test puedeVender 6 | devuelve true cuando el usuario tiene dinero y queda stock | El tipo de arma (escudo) y Usuario 2 | El usuario tiene dinero suficiente (25) y quedan escudos(1) | True | True |  |
| Test puedeVender 7 | devuelve true cuando el usuario tiene dinero y queda stock | El tipo de arma (escudo) y Usuario 1 | El usuario no tiene dinero suficiente (0) y no quedan escudos(0) | False | False |  |
| Test puedeVender 8 | devuelve true cuando el usuario tiene dinero y queda stock | El tipo de arma (escudo) y Usuario 2 | El usuario tiene dinero suficiente (50) y no quedan escudos(0) | False | False |  |
| Test puedeVender 9 | devuelve true cuando el usuario tiene dinero y queda stock | El tipo de arma (misil) y Usuario 1 | El usuario no tiene dinero suficiente (0) y quedan misiles | False | False |  |
| Test puedeVender 10 | devuelve true cuando el usuario tiene dinero y queda stock | El tipo de arma (misil) y Usuario 2 | El usuario tiene dinero suficiente (50) y quedan misiles(0) | True | True |  |
| Test venderMisil 1 | Vende un misil, lo resta del stock y le quita dinero al usuario | Un Usuario | puedeVender devuelve true | Dinero del usuario: 90  Stock: 9 | Dinero del usuario: 90  Stock: 9 |  |
| Test venderMisil 2 | Vende un misil, lo resta del stock y le quita dinero al usuario | Un Usuario | puedeVender devuelve true | Dinero del usuario: 80  Stock: 8 | Dinero del usuario: 80  Stock: 8 |  |
| Test venderEscudo 1 | Vende un escudo, lo resta del stock y le quita dinero al usuario | Un Usuario | puedeVender devuelve true | Dinero del usuario: 75  Stock: 4 | Dinero del usuario: 75  Stock: 4 |  |
| Test venderMisilNS 1 | Vende un misilNS, lo resta del stock y le quita dinero al usuario | Un Usuario | puedeVender devuelve true | Dinero del usuario: 55  Stock: 0 | Dinero del usuario: 55  Stock: 0 |  |
| Test venderMisilEO 1 | Vende un misilEO, lo resta del stock y le quita dinero al usuario | Un Usuario | puedeVender devuelve true | Dinero del usuario: 55  Stock: 0 | Dinero del usuario: 55  Stock: 0 |  |
| Test venderMisilBOOM 1 | Vende un misilBOOM, lo resta del stock y le quita dinero al usuario | Un Usuario | puedeVender devuelve true | Dinero del usuario: 10  Stock: 0 | Dinero del usuario: 10  Stock: 0 |  |

1 En qué estado se encuentra el sistema para que esa entrada cumpla el objetivo de la prueba

**Clase BarcoTest**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id. De la prueba** | **Objetivo** | **Entrada** | **Condiciones de ejecución1** | **Resultado esperado** | **Resultado obtenido** | **Comentarios** |
| Test estaEnPos 1 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (0,0) | Hemos creado un portaaviones en la posición (0,0) en vertical | True | True |  |
| Test estaEnPos 2 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (0,1) | Hemos creado un portaaviones en la posición (0,0) en vertical | True | True |  |
| Test estaEnPos 3 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (0,2) | Hemos creado un portaaviones en la posición (0,0) en vertical | True | True |  |
| Test estaEnPos 4 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (0,3) | Hemos creado un portaaviones en la posición (0,0) en vertical | True | True |  |
| Test estaEnPos 5 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (0,4) | Hemos creado un portaaviones en la posición (0,0) en vertical | False | False |  |
| Test estaEnPos 6 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (0,-1) | Hemos creado un portaaviones en la posición (0,0) en vertical | False | False |  |
| Test estaEnPos 7 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (5,7) | Hemos creado un Submarino en la posición (5,7) en horizontal | True | True |  |
| Test estaEnPos 8 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (6,7) | Hemos creado un Submarino en la posición (5,7) en horizontal | True | True |  |
| Test estaEnPos 9 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (7,7) | Hemos creado un Submarino en la posición (5,7) en horizontal | True | True |  |
| Test estaEnPos 10 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (8,7) | Hemos creado un Submarino en la posición (5,7) en horizontal | False | False |  |
| Test estaEnPos 11 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (4,7) | Hemos creado un Submarino en la posición (5,7) en horizontal | False | False |  |
| Test estaEnPos 12 | Devuelve true si la coordenada que entra pertenece al barco | Coordenada (5,6) | Hemos creado un Submarino en la posición (5,7) en horizontal | False | False |  |
| Test FueraDeLimites 1 | Devuelve true si el Barco se sale de los límites del tablero |  | Hemos creado un Portaaviones en la (0,0) en vertical | False | False |  |
| Test FueraDeLimites 2 | Devuelve true si el Barco se sale de los límites del tablero |  | Hemos creado un Portaaviones en la (7,0) en vertical | False | False |  |
| Test FueraDeLimites 3 | Devuelve true si el Barco se sale de los límites del tablero |  | Hemos creado una Fragata en la (10,0) | True | True |  |
| Test FueraDeLimites 4 | Devuelve true si el Barco se sale de los límites del tablero |  | Hemos creado un Submarino en la (5,9) en horizontal | False | False |  |
| Test FueraDeLimites 5 | Devuelve true si el Barco se sale de los límites del tablero |  | Hemos creado un Submarino en la (5,5) en horizontal | False | False |  |
| Test FueraDeLimites 6 | Devuelve true si el Barco se sale de los límites del tablero |  | Hemos creado un Submarino en la (-1,5) en horizontal | True | True |  |
| Test calcularAdyacentes | Devuelve una lista con las coordenadas del barco y sus adyacentes |  | Hemos creado un destructor en (0,0) en vertical | La lista de coordenadas de las adyacentes. | La lista de las coordenadas de las adyacentes  (Lo hace correctamente) | Para comprobar que este método funciona correctamente hemos creado una lista manualmente con las coordenadas adyacentes, y después comprobamos que ambas listas contienen los mismos elementos. |

**Clase BattleshipTest** (hemos inicializado el BattleShip antes de probar todos sus métodos)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id. De la prueba** | **Objetivo** | **Entrada** | **Condiciones de ejecución1** | **Resultado esperado** | **Resultado obtenido** | **Comentarios** |
| Test colocarBarcoUs 1 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un portaaviones en la posición (0,0) en vertical, ejecutamos hayBarcoUsu en la posición (0,0) | True | True |  |
| Test colocarBarcoUs 2 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un portaaviones en la posición (0,0) en vertical, ejecutamos hayBarcoUsu en la posición (0,1) | True | True |  |
| Test colocarBarcoUs 3 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un portaaviones en la posición (0,0) en vertical, ejecutamos hayBarcoUsu en la posición (0,2) | True | True |  |
| Test colocarBarcoUs 4 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un portaaviones en la posición (0,0) en vertical, ejecutamos hayBarcoUsu en la posición (0,3) | True | True |  |
| Test colocarBarcoUs 5 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un portaaviones en la posición (0,0) en vertical, ejecutamos hayBarcoUsu en la posición (0,4) | False | False |  |
| Test colocarBarcoUs 6 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un Submarino en la posición (4,4) en horizontal, ejecutamos hayBarcoUsu en la posición (4,4) | True | True |  |
| Test colocarBarcoUs 7 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un Submarino en la posición (4,4) en horizontal, ejecutamos hayBarcoUsu en la posición (5,4) | True | True |  |
| Test colocarBarcoUs 8 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un Submarino en la posición (4,4) en horizontal, ejecutamos hayBarcoUsu en la posición (6,4) | True | True |  |
| Test colocarBarcoUs 9 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un Submarino en la posición (4,4) en horizontal, ejecutamos hayBarcoUsu en la posición (3,4) | False | False |  |
| Test colocarBarcoUs 10 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un Submarino en la posición (4,4) en horizontal, ejecutamos hayBarcoUsu en la posición (7,4) | False | False |  |
| Test colocarBarcoUs 11 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un Destructor en la posición (6,6) en horizontal, ejecutamos hayBarcoUsu en la posición (6,6) | True | True |  |
| Test colocarBarcoUs 12 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un Destructor en la posición (6,6) en horizontal, ejecutamos hayBarcoUsu en la posición (7,6) | True | True |  |
| Test colocarBarcoUs 13 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un Destructor en la posición (6,6) en horizontal, ejecutamos hayBarcoUsu en la posición (5,6) | False | False |  |
| Test colocarBarcoUs 14 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado un Destructor en la posición (6,6) en horizontal, ejecutamos hayBarcoUsu en la posición (8,6) | False | False |  |
| Test colocarBarcoUs 15 | Coloca un barco en el tablero del usuario en la posición, y del tipo indicados | El tipo de barco en un String, la coordenada, y si es en vertical | Hemos creado una Fragata en la posición (9,9) en horizontal, ejecutamos hayBarcoUsu en la posición (9,9) | True | True |  |
| Test puedeColocar 1 | Devuelve true si el usuario puede colocar un barco en esa posición | El tipo de barco en un String, la coordenada, y si es en vertical | Le pasamos un portaaviones en la coordenada (0,0) en vertical | True | True |  |
| Test puedeColocar 2 | Devuelve true si el usuario puede colocar un barco en esa posición | El tipo de barco en un String, la coordenada, y si es en vertical | Le pasamos una Fragata en la coordenada (0,2), el portaaviones anterior está colocado | False | False |  |
| Test puedeColocar 3 | Devuelve true si el usuario puede colocar un barco en esa posición | El tipo de barco en un String, la coordenada, y si es en vertical | Le pasamos un submarino en la coordenada (4,4) en horizontal. El portaviones sigue colocado. | True | True |  |
| Test puedeColocar 4 | Devuelve true si el usuario puede colocar un barco en esa posición | El tipo de barco en un String, la coordenada, y si es en vertical | Le pasamos un destructor en la coordenada (6,6) en horizontal. El portaviones y el submarino están colocados | True | True |  |
| Test puedeColocar 5 | Devuelve true si el usuario puede colocar un barco en esa posición | El tipo de barco en un String, la coordenada, y si es en vertical | Le pasamos un Portaaviones en la coordenada (5,0) en horizontal. El portaviones, el destructor y el submarino están colocados | False | False |  |
| Test puedeColocar 6 | Devuelve true si el usuario puede colocar un barco en esa posición | El tipo de barco en un String, la coordenada, y si es en vertical | Le pasamos una fragata en la coordenada (9,9). El portaviones, el destructor y el submarino están colocados | True | True |  |

**Clase CoordenadaTest**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id. De la prueba** | **Objetivo** | **Entrada** | **Condiciones de ejecución1** | **Resultado esperado** | **Resultado obtenido** | **Comentarios** |
| Test getAdyacentes | Devuelve una lista con las coordenadas adyacentes a una coordenada y ella misma |  | Hemos creado una Coordenada (en nuestra prueba la 3,3) | Una lista con las 8 coordenadas adyacentes y ella misma | Una lista con las 8 coordenadas adyacentes y ella misma | Para comprobar hemos metido las coordenadas manualmente en otra lista y hemos comparado las dos listas |

**Clase ListaCoordenadasTest**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id. De la prueba** | **Objetivo** | **Entrada** | **Condiciones de ejecución1** | **Resultado esperado** | **Resultado obtenido** | **Comentarios** |
| Test estaEnLista 1 | Devuelve true si la coordenada que le entra está en la lista | Una coordenada | Hemos creado una Coordenada (en nuestra prueba la 3,3)  Y una lista vacía | False | False |  |
| Test estaEnLista 2 | Devuelve true si la coordenada que le entra está en la lista | Una coordenada | Hemos creado una Coordenada (en nuestra prueba la 3,3)  Y una lista a la que hemos añadido la coordenada | True | True |  |
| Test calcularAdyacentes 1 | Devuelve una lista con las coordenadas adyacentes a las coordenadas de la lista |  | La lista está vacía | Una lista vacía | Una lista vacía |  |
| Test calcularAdyacentes 2 | Devuelve una lista con las coordenadas adyacentes a las coordenadas de la lista |  | La lista tiene la coordenada (5,5) | Una lista con las adyacentes de la coordenada (5,5) | Una lista con las adyacentes de la coordenada (5,5) | Para comprobar creamos una lista con las coordenadas adyacentes manualmente, y las comparamos |
| Test calcularAdyacentes 3 | Devuelve una lista con las coordenadas adyacentes a las coordenadas de la lista |  | La lista tiene la coordenada (5,5) y sus adyacentes, le añadimos la coordenada (5,7) y lo ejecutamos de nuevo | Una lista con las adyacentes de la coordenada (5,5) y de (5,7) sin repetidos | Una lista con las adyacentes de la coordenada (5,5) y de (5,7) sin repetidos | Para comprobar creamos una lista con las coordenadas adyacentes manualmente, y las comparamos |
| Test calcularAdyacentes 4 | Devuelve una lista con las coordenadas adyacentes a las coordenadas de la lista |  | La lista tiene la coordenada (5,5) , (5,7) y sus adyacentes, le añadimos la coordenada (5,6) y lo ejecutamos de nuevo | Una lista con las adyacentes de la coordenada (5,5) , de (5,6) y de (5,7) sin repetidos | Una lista con las adyacentes de la coordenada (5,5) , de (5,6) y de (5,7) sin repetidos | Para comprobar creamos una lista con las coordenadas adyacentes manualmente, y las comparamos |
| Test fueraDeLimites 1 | Devuelve True si alguna de las coordenadas se sale de los límites |  | Tenemos una lista vacía | False | False |  |
| Test fueraDeLimites 2 | Devuelve True si alguna de las coordenadas se sale de los límites |  | Tenemos una lista con la coordenada (3,3) | False | False |  |
| Test fueraDeLimites 3 | Devuelve True si alguna de las coordenadas se sale de los límites |  | Tenemos una lista con las coordenadas (3,3) y (9,0) | False | False |  |
| Test fueraDeLimites 3 | Devuelve True si alguna de las coordenadas se sale de los límites |  | Tenemos una lista con las coordenadas (3,3), (9,0) y (32,3) | True | True |  |
| Test fueraDeLimites 3 | Devuelve True si alguna de las coordenadas se sale de los límites |  | Tenemos una lista con las coordenadas (3,3), (9,0), (32,3) y (-1,-13) | True | True |  |
| Test ComprobarListas 1 | Devuelve True si coincide alguna de las coordenadas de las listas | Una lista | Ambas listas están vacías | False | False |  |
| Test ComprobarListas 2 | Devuelve True si coincide alguna de las coordenadas de las listas | Una lista | La lista 1 tiene la coordenada (3,3) | False | False |  |
| Test ComprobarListas 3 | Devuelve True si coincide alguna de las coordenadas de las listas | Una lista | La lista 1 tiene: (3,3) (9,0)  La lista 2 tiene:  (4,5) (2,7) | False | False |  |
| Test ComprobarListas 4 | Devuelve True si coincide alguna de las coordenadas de las listas | Una lista | La lista 1 tiene: (3,3) (9,0) (4,5)  La lista 2 tiene:  (4,5) (2,7) | True | True |  |

**Clase JugadorTest** (hemos inicializado el BattleShip antes de probar todos los métodos)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id. De la prueba** | **Objetivo** | **Entrada** | **Condiciones de ejecución1** | **Resultado esperado** | **Resultado obtenido** | **Comentarios** |
| Test puedeColocar 1 | Devuelve true si jugador puede colocar ese barco | Un barco | Hemos creado un usuario y le hemos puesto un portaaviones en la (0,0) en horizontal. Intentamos poner un Submarino en vertical en la (0,0). | False | False |  |
| Test puedeColocar 2 | Devuelve true si jugador puede colocar ese barco | Un barco | Hemos creado un usuario y le hemos puesto un portaaviones en la (0,0) en horizontal. Intentamos poner un Submarino en vertical en la (1,1). | False | False |  |
| Test puedeColocar 3 | Devuelve true si jugador puede colocar ese barco | Un barco | Hemos creado un usuario y le hemos puesto un portaaviones en la (0,0) en horizontal. Intentamos poner una fragata en la (9,9). | True | True |  |
| Test puedeColocar 4 | Devuelve true si jugador puede colocar ese barco | Un barco | Hemos creado un usuario y le hemos puesto un portaaviones en la (0,0) en horizontal y una fragata en la (9,9). Intentamos poner un submarino en vertical en la (8,8) | False | False |  |
| Test puedeColocar 5 | Devuelve true si jugador puede colocar ese barco | Un barco | Hemos creado un usuario y le hemos puesto un portaaviones en la (0,0) en horizontal y una fragata en la (9,9). Intentamos poner un submarino en vertical en la (5,5) | True | True |  |
| Test puedeColocar 6 | Devuelve true si jugador puede colocar ese barco | Un barco | Hemos creado un usuario y le hemos puesto un portaaviones en la (0,0) en horizontal, un submarino en la (5,5) en vertical y una fragata en la (9,9). Intentamos poner un submarino en horizontal en la (3,6) | True | True |  |

**Clase JugadorTest** (hemos inicializado el BattleShip antes de probar todos los métodos)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id. De la prueba** | **Objetivo** | **Entrada** | **Condiciones de ejecución1** | **Resultado esperado** | **Resultado obtenido** | **Comentarios** |
| Test puedePonerUs 1 | Devuelve true si el usuario puede poner ese barco | Un barco | Usuario no tiene ningún barco. Al método le pasamos un portaaviones en vertical en la (0,0) | True | True | Colocamos el barco para las siguientes pruebas ya que devuelve true. |
| Test puedePonerUs 2 | Devuelve true si el usuario puede poner ese barco | Un barco | Intentamos poner otro portaaviones en horizontal en la (2,0) | False | False |  |
| Test puedePonerUs 3 | Devuelve true si el usuario puede poner ese barco | Un barco | Usuario no tiene ningún barco. Al método le pasamos un submarino en vertical en la (4,2) | True | True | Colocamos el barco para las siguientes pruebas ya que devuelve true. |
| Test puedePonerUs 4 | Devuelve true si el usuario puede poner ese barco | Un barco | Al método le pasamos un submarino en vertical en la (6,2) | True | True | Colocamos el barco para las siguientes pruebas ya que devuelve true. |
| Test puedePonerUs 5 | Devuelve true si el usuario puede poner ese barco | Un barco | Al método le pasamos un submarino en horizontal en la (0,0) | False | False |  |
| Test puedePonerUs 6 | Devuelve true si el usuario puede poner ese barco | Un barco | Usuario no tiene ningún barco. Al método le pasamos un destructor en vertical en la (0,0) | True | True | Colocamos el barco para las siguientes pruebas ya que devuelve true. |
| Test puedePonerUs 7 | Devuelve true si el usuario puede poner ese barco | Un barco | Al método le pasamos un destructor en vertical en la (2,0) | True | True | Colocamos el barco para las siguientes pruebas ya que devuelve true. |
| Test puedePonerUs 8 | Devuelve true si el usuario puede poner ese barco | Un barco | Al método le pasamos un destructor en vertical en la (4,0) | True | True | Colocamos el barco para las siguientes pruebas ya que devuelve true. |
| Test puedePonerUs 9 | Devuelve true si el usuario puede poner ese barco | Un barco | Al método le pasamos un destructor en vertical en la (6,0) | False | False |  |
| Test puedePonerUs 10 | Devuelve true si el usuario puede poner ese barco | Un barco | Usuario no tiene ningún barco. Al método le pasamos una fragata en la (0,0) | True | True | Colocamos el barco para las siguientes pruebas ya que devuelve true. |
| Test puedePonerUs 11 | Devuelve true si el usuario puede poner ese barco | Un barco | Al método le pasamos una fragata en la (2,0) | True | True | Colocamos el barco para las siguientes pruebas ya que devuelve true. |
| Test puedePonerUs 12 | Devuelve true si el usuario puede poner ese barco | Un barco | Al método le pasamos una fragata en la (2,0) | True | True | Colocamos el barco para las siguientes pruebas ya que devuelve true. |
| Test puedePonerUs 13 | Devuelve true si el usuario puede poner ese barco | Un barco | Al método le pasamos una fragata en la (4,0) | True | True | Colocamos el barco para las siguientes pruebas ya que devuelve true. |
| Test puedePonerUs 14 | Devuelve true si el usuario puede poner ese barco | Un barco | Al método le pasamos una fragata en la (6,0) | True | True | Colocamos el barco para las siguientes pruebas ya que devuelve true. |
| Test puedePonerUs 15 | Devuelve true si el usuario puede poner ese barco | Un barco | Al método le pasamos una fragata en la (8,0) | False | False |  |