eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Ingeniería del Software

Proyecto: Flota Hundida
2021-2022

Departamento:
Lenguajes y Sistemas Informáticos

Titulación:
Grado en Informática de Gestión y Sistemas de Información

2º Curso (2º Cuatrimestre)

Mayo de 2022

Velasco Prieto, Alvaro
Elorza Gabilondo, David
Bermudez Osaba, Unai
Castro Martínez De Mendibil, Unai
Martín Paniagua, Marcos

Índice

1. Introducción

Para el proyecto de la asignatura Ingeniería del Software se nos plantea desarrollar el juego *Hundir La Flota* aplicando diversos conceptos tratados en la asignatura y en anteriores asignaturas de programación. Esta aplicación será desarrollada en el lenguaje Java, lenguaje que se caracteriza por usar el paradigma de la programación orientada a objetos. De esta manera, podremos crear diferentes clases y aplicar diferentes patrones de diseño los cuales presentaremos más adelante.

Para desarrollar la interfaz gráfica del juego se emplea la biblioteca *Swing.* La interacción entre el usuario y el programa se gestiona implementando el patrón de diseño MVC (Modelo Vista Controlador) en el que se diferencian tres partes fundamentales del programa. El Modelo se encarga de la lógica de negocio y de gestionar los datos del programa, la vista que se encarga de presentar los datos de una manera adecuada, y el controlador que invoca los cambios en el modelo ante acciones de la vista. El documento anexo *Informe MVC* contiene una descripción más detallada de esto.

Para representar las clases del modelo y las relaciones entre ellas se emplean diagramas de clases y secuencia UML, desarrollados con la aplicación *Visual Paradigm*. Estos diagramas se presentan en el apartado de diseño.

Finalmente, cabe comentar que para el desarrollo de esta aplicación se ha empleado un marco de trabajo para desarrollo ágil de software SCRUM. Debido a esto, hemos desarrollado el proyecto a base de 3 SPRINTS, es decir, tres pasos en los que se desarrollan diferentes partes de la aplicación y al final de los cuales se realiza una reunión con el cliente para presentar el progreso, discutir las decisiones tomadas y plantear diferentes cambios. La gestión de esto se presenta en el siguiente apartado.

2. Gestión

En esta parte, explicaremos todas las reuniones que hemos tenido y cómo hemos dividido las tareas de cada sprint.

Reparto de Tareas

Sprint	Tarea	Responsables	Tiempo Planificado	Tiempo Real	Observaciones
1	HU1: Inicializar el juego	Marcos Martín Unai Castro	5h	8h	Problemas con casos críticos.
	HU6: Disparar el jugador	Álvaro Velasco David Elorza Unai Bermúdez	3h	4h	Problemas al cambiar de color en el tablero.
	HU7: Disparar el ordenador	Álvaro Velasco David Elorza Unai Bermúdez	2h	2h	
	Diagrama de clases y secuencia	Álvaro Velasco David Elorza Unai Bermúdez Marcos Martín Unai Castro	2h	3h	La organización en el diagrama de secuencias.

Sprint	Tarea	Responsables	Tiempo Planificado	Tiempo Real	Observaciones
2	HU5: Poner escudo a jugador e IA	Marcos Martín Unai Castro	3h	6h	Dificultades a la hora de definir el orden en el que se pintan los barcos, escudo, escudo tocado y agua para que todo se visualice.
	HU4: Radar del jugador y de la IA + diagrama de secuencia	David Elorza Unai Bermúdez	4h	7h	Para que la IA pueda ver utilizar el radar, a la hora de disparar se ha creado un Array con los avistamientos del radar el cual se tiene en cuenta a la hora de disparar

Sprint	Tarea	Responsables	Tiempo Planificado	Tiempo Real	Observaciones
3	HU8 y HU9: Reparar barco jugador y ordenador	Unai Castro Marcos Martín	3h	3h	Al principio pensamos en crear un botón como el de consultar radar, pero nos decantamos por el RadioButton ya que tienes que tener la opción de elegir el barco que reparas.
	HU10 y HU11: Comprar armamento jugador y ordenador	David Elorza Unai Bermúdez Álvaro Velasco	4h	4h	La distribución de la tienda en el aspecto de la interfaz se ha complicado un poco al tener tantos elementos.

Actas De Reunión

Las actas de reunión están adjuntas en la carpeta principal del proyecto, esa es la causa por la que aquí no pondremos todas las actas de reuniones. En este último sprint 3, nos hemos reunido los días 4, 8 y 15 de mayo. La primera reunión dividimos el sprint 3 en diferentes partes y ver quién hacía cada cosa. En la segunda reunión estuvimos poniendo todo el código en un mismo archivo para ver si funcionaba el juego. Al ver el éxito del juego, nos reunimos una vez más para ver si toda la documentación estaba finalizada correctamente.

Recursos

En este proyecto hemos usados tres herramientas clave para el desarrollo del mismo:

- Eclipse: Esta herramienta nos ha servido para poder crear el código del juego y hacer todas las pruebas. Eclipse es la herramienta más importante de las tres.
- GitHub: Nos ha ayudado a poder poner en común el código que habíamos actualizado cada uno de manera ordenada y detallada.
- Visual Paradigm: Con Visual Paradigm hemos podido hacer tanto el diagrama de clases como el diagrama de secuencia.

3. Diseño

3.1 Diagrama de clases

Como se comenta en la introducción para el diseño del juego se ha empleado el patrón de diseño MVC el cual se detalla en el documento anexo *Informe MVC.* En esta parte de la documentación nos centraremos en detallar las decisiones de diseño tomadas en la parte del Modelo. Para ello disponemos de los diagramas de clase y secuencia UML en los que podemos ver las diferentes clases, métodos y atributos implementados, junto a los diferentes patrones de diseño empleados.

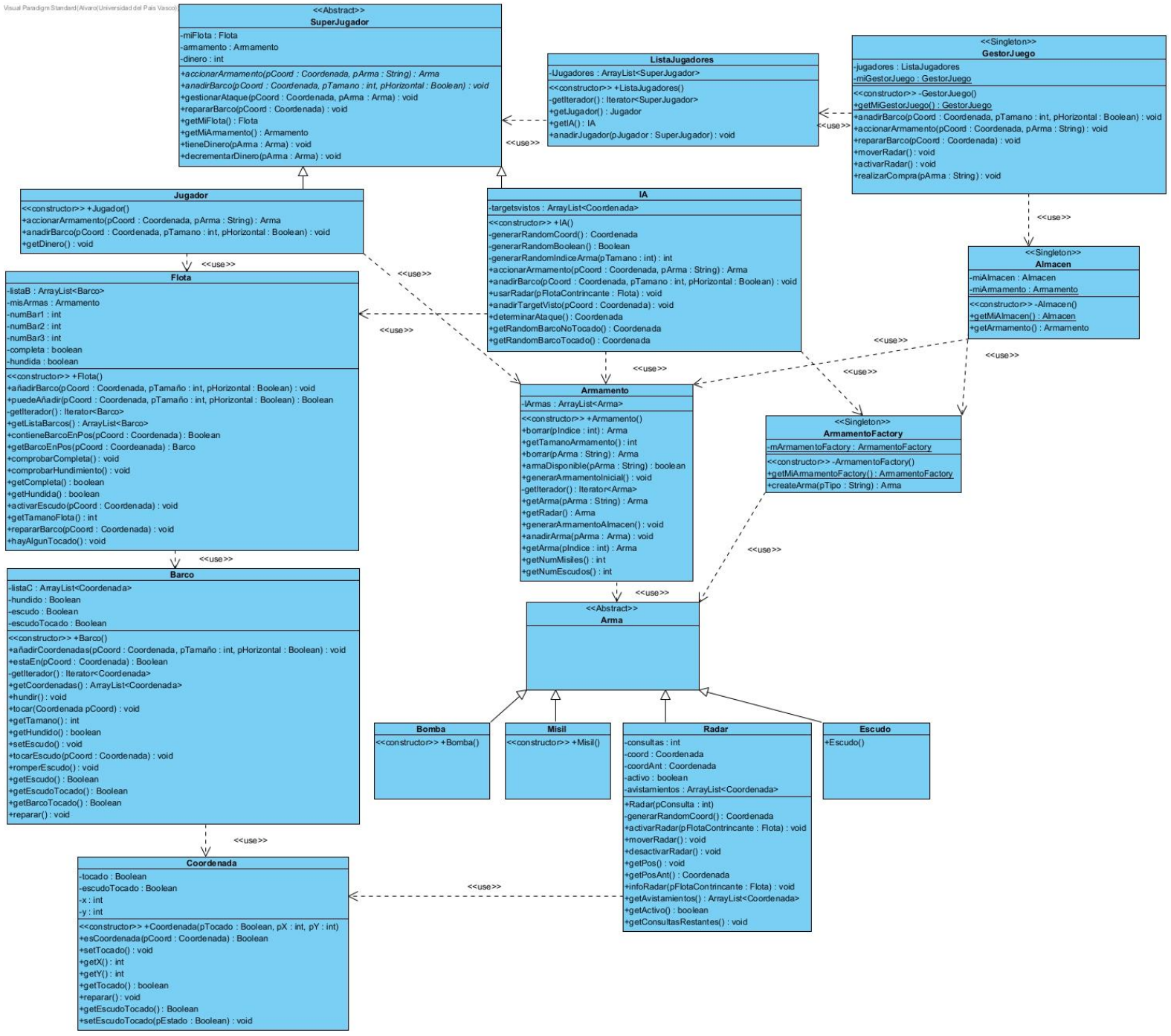

Figura 1: Diagrama de clases

En el diagrama de clases podemos ver que para el desarrollo del modelo se han empleado 15 clases diferentes. La clase GestorJuego es la encargada de gestionar las llamadas a los diferentes métodos del Jugador y la IA. Esta clase contiene una lista de SuperJugador que es la clase que tanto la IA como el Jugador extienden mediante herencia para implementar sus diferentes funcionalidades. Ambos contienen una Flota que a su vez contiene una lista de Barcos y cada Barco contiene una lista de Coordenadas. De esta manera cada jugador tiene unos barcos colocados en unas posiciones concretas.
Además de esto tanto el Jugador como la IA contienen un Armamento el cual contiene una lista de Arma, clase abstracta la cual extienden Bomba, Misil, Radar y Escudo. De esta manera, el jugador y la IA tienen unas armas con las que realizar ataques.
Finalmente, tenemos la clase Almacen que contiene un Armamento y de esta manera representamos las armas que contiene la tienda de la cual tanto la IA como el jugador pueden comprar.

En cuanto a los patrones de diseño, la aplicación incluye los siguientes:

- **Patron singleton:** Lo incluyen las clases GestorJuego y Almacen ya que en el modelo se quiere que solo exista una instancia de estas clases.
- **Patron Factory:** La clase ArmamentoFactory se encarga de generar los diferentes tipos de arma es decir se encarga de generar las diferentes clases que extienden a la clase abstracta Arma, es decir, Bomba, Misil, Radar y Escudo.

3.2 Diagrama de secuencia

Mediante el diagrama de secuencia podemos obtener una idea muy aproximada y clara de cualquier método. De forma que hasta gente ajena a la programación del código sepa de manera general como funciona el código. Además, una vez obtenido el diagrama de clases, si se hace el diagrama de secuencia con la funcionalidad del método a implementar ya pensada ahorra mucho trabajo a la hora de escribir el código para que el programa funcione. Por ello, concluímos que es una herramienta muy útil tanto para la presentación de la funcionalidad del proyecto como para la ejecución del mismo.

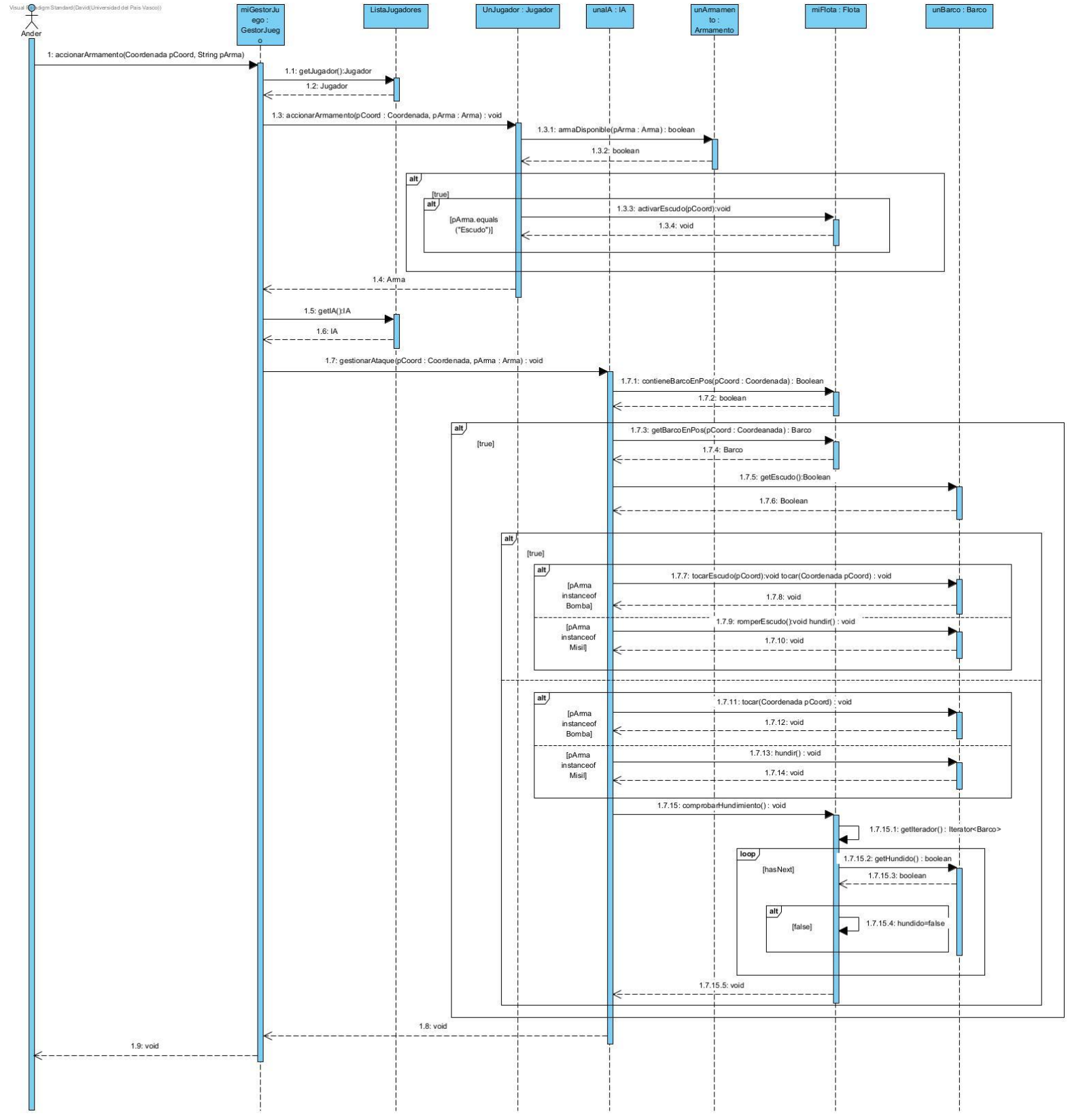

Figura 2: Diagrama de secuencia Disparar Jugador

La imagen superior, corresponde al diagrama de secuencia del método disparar del jugador, todo gestionado por la clase GestorJuego que llama al Jugador para que se elija un arma y se hagan todas las comprobaciones respectivas de que esta exista y se comprueba si es un escudo para activarlo. Posteriormente se envía el arma elegida a la IA para que gestione el ataque representando el mismo en las casillas gestionando el que sea una bomba o misil, si en la coordenada que recibe hay un barco y si tiene escudo para quitarselo o marcar esa casilla o la del barco entero dependiendo de si es un misil o una bomba.

Método consultar Radar IA

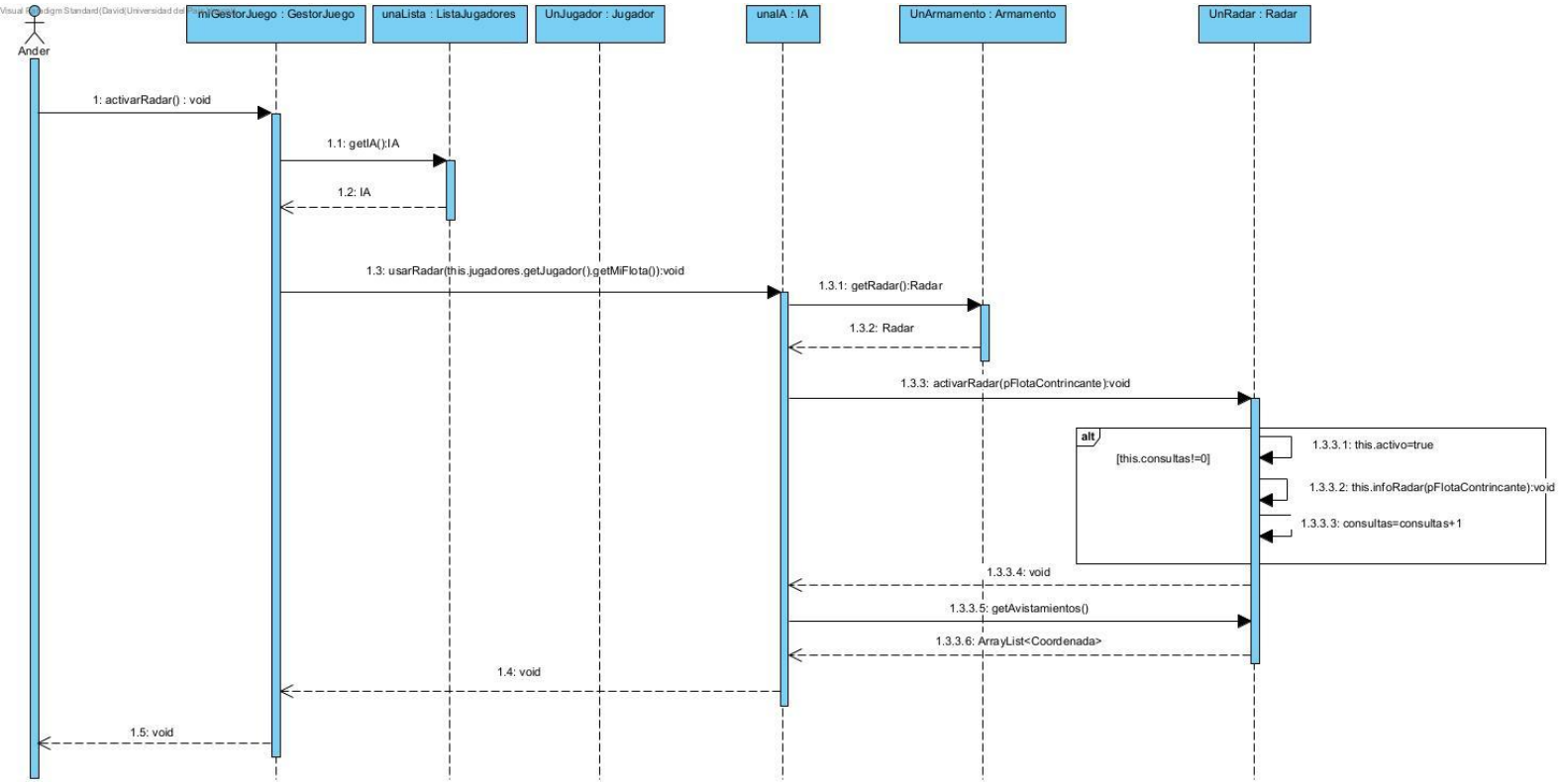

Figura 3: Diagrama de secuencia Activar Radar Ordenador

Esta segunda imagen, corresponde al método activar Radar del ordenador que es otra de las “armas” disponibles para su uso. Con el diagrama de secuencia, nos podemos hacer una primera idea de cómo funciona. De manera general, su funcionamiento es el siguiente: al igual que antes se gestiona desde el gestorJuego, que llama a la ListaJugadores para obtener la IA a la que se va a llamar para que use el radar, desde la IA se obtiene un radar para que posteriormente se le llame y que active él el radar y si no ha gastado todas sus consultas se activa el radar se reduce el número de consultas restantes y se obtiene la información del radar, es decir de las casillas a las que ha afectado para finalmente obtener las casillas de barcos que han sido avistadas.

4. Desarrollo

- Documentación de pruebas

Prueba id	Objetivo	Input	Condiciones de ejecución	Resultado esperado y obtenido	Comentarios
1	Colocación barco	Coordenada, tamaño, direccion	Tablero inicializado	Se coloca el barco si la posición está permitida	El método se asegura de que no haya ningún barco alrededor. La dirección se indica con un boolean que indica si es horizontal o no
2	Mover radar	-	Tablero inicializado y barcos colocados	El radar se mueve a una casilla aleatoria del oponente	
3	Activar radar	-	Tablero inicializado, barcos colocados y quedan consultas disponibles	Se muestra si hay algún barco alrededor del radar	
4	Disparar Bomba/Misil	Coordenada y Arma	Barcos colocados (en caso de misil tener misiles en el armamento)	Se dispara en la casilla indicada con el arma indicada	
5	Colocar escudo	Barco	Barcos colocados, barco no tocado y escudos disponibles	Se coloca el escudo en el barco indicado	
6	Reparar barco	Barco	Barcos colocados y dinero suficiente para reparación	Se reparan todas las casillas del barco	La reparación tiene un coste de 25
7	Comprar armamento	Arma	Barcos colocados, arma disponible en la tienda y dinero suficiente	Se mueve el arma de la tienda a tu armamento y se te resta el importe	Precio misil: 10 Precio escudo: 20

5. Conclusiones

Mediante este proyecto se han podido aplicar los conceptos aprendidos tanto en los laboratorios de la asignatura como en las clases magistrales. Se han aplicado diferentes patrones de diseño en un uso real y además se ha aprendido a desarrollar una interfaz gráfica usando una librería de Java. Así mismo, se ha aprendido a trabajar usando una dinámica de trabajo SCRUM, la cual nunca habíamos empleado. Esto último nos ha parecido un descubrimiento muy valioso ya que la comunicación y la dinámica del grupo ha sido muy buena durante el desarrollo de los SPRINTS y además se ha podido corregir diversos errores detectados en las reuniones con el cliente antes de la entrega final.

Como cambio futuro se plantea mejorar la interfaz gráfica haciendo uso de una librería moderna que contenga gráficos más actuales. Además, de una forma bastante asequible se podría implementar la funcionalidad de jugar contra otro jugador en vez de contra la IA. Finalmente, se podría mejorar la lógica de la IA implementando alguna técnica de inteligencia artificial, y se podrían crear diferentes niveles de dificultad.