FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico II — Java

[7507/9502] Algoritmos y Programación III
Curso 1
Primer cuatrimestre de 2023

Nombre	Padrón	Email
Lionel Maydana	106512	lmaydana@fi.uba.ar
Ramiro Mantero	107162	rmantero@fi.uba.ar
Lautaro Torraca	108813	ltorraca@fi.uba.ar

Índice

1. Introducción

El presente informe reúne la documentación de la solución del segundo trabajo practico de la materia Algoritmos y Programación III. Dicha solución consiste en desarrollar una aplicación de manera grupal aplicando todos los conceptos vistos en el curso, utilizando un lenguaje de tipado estático (Java) con un diseño del modelo orientado a objetos y trabajando con las técnicas de TDD e Integración Continua.

2. Especificaciones de la aplicación a desarrollar

AlgoDefense es un juego similar al famoso TowerDefense pero no es en tiempo real sino por turnos. El mismo es un juego de estrategia y se basa en la construcción de defensas que impidan que los enemigos lleguen hasta el jugador.

Durante el desarrollo del juego el jugador podrá elegir que defensa construir o si quiere avanzar el turno para que los enemigos avancen, sus defensas se construyan y obtener mas créditos para seguir construyendo torres y evitar perder el juego

3. Supuestos

Recorrido del Búho Se supuso que el búho con mas del 50 por ciento de vida baja en forma vertical hasta la coordenada Y de la meta y luego avanza sobre la coordenada X hasta llegar a la meta.

4. Diagramas de clase

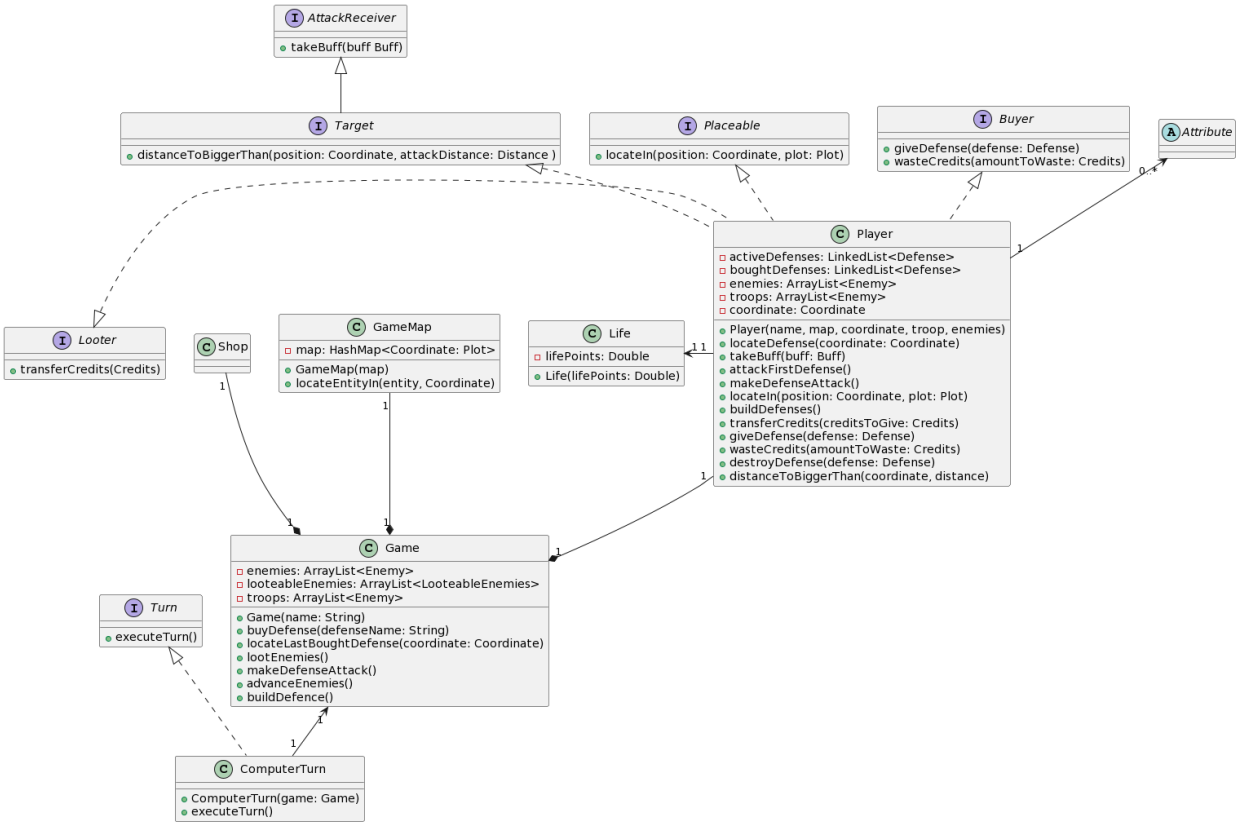

Figura 1: Visión de las clases principales a la hora de jugar.

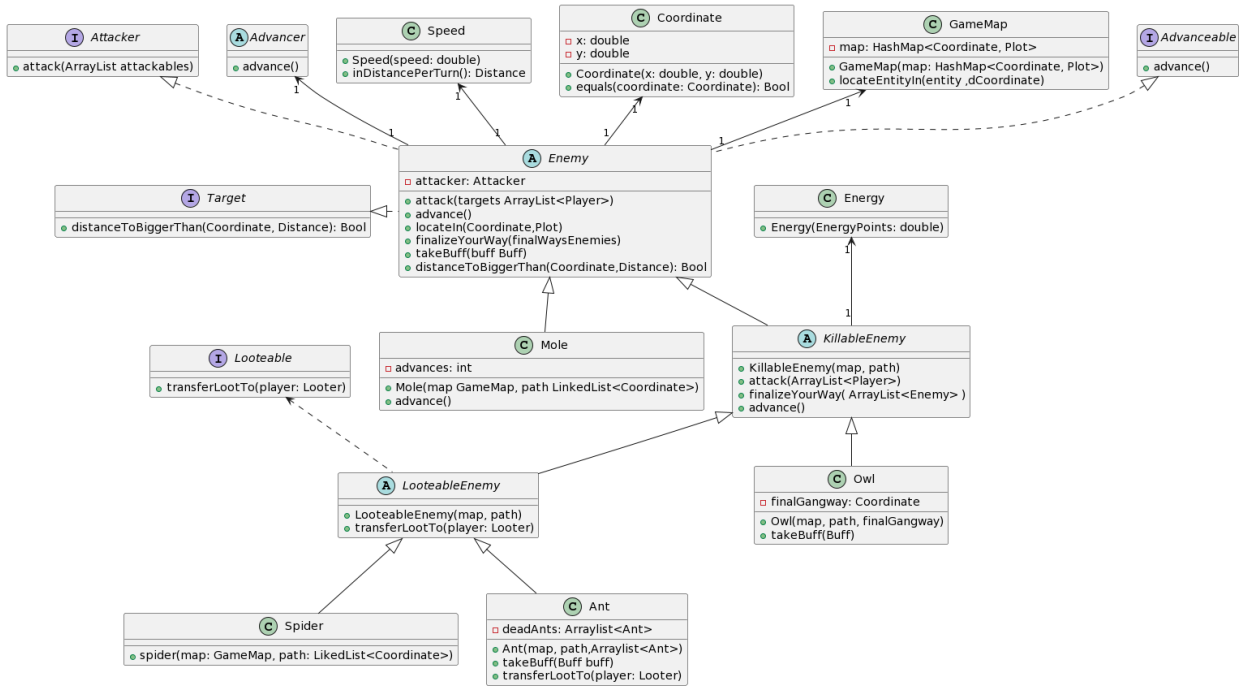

Figura 2: Diagrama de la clase Enemy, sus clases hijas y atributos.

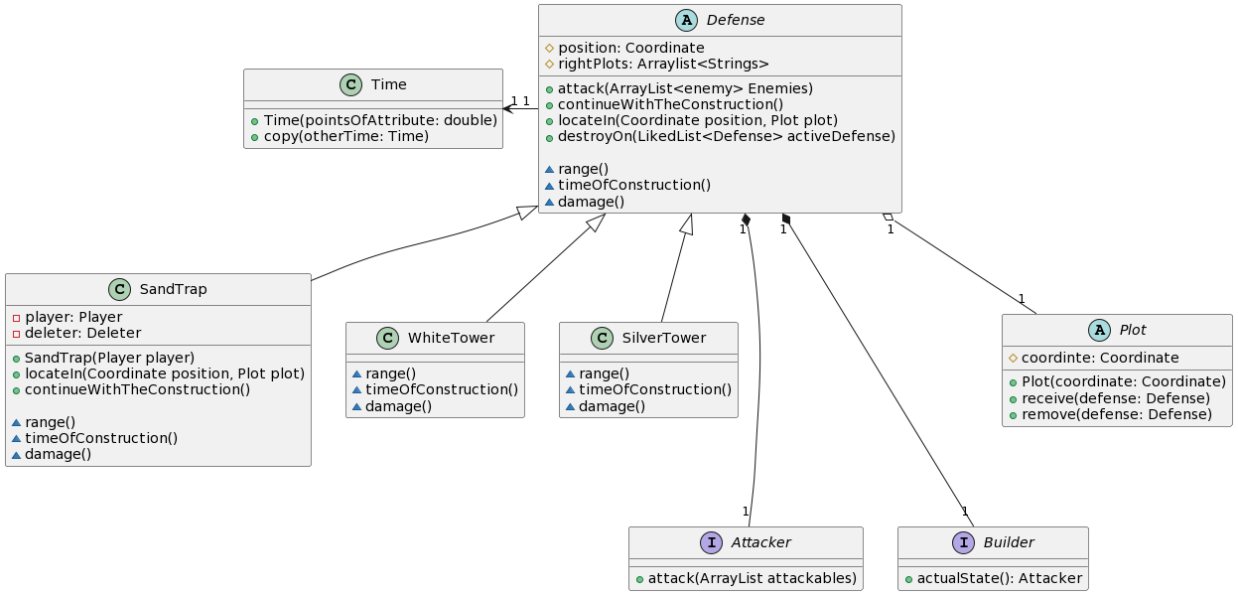

Figura 3: Diagrama de la clase Defense, sus clases hijas y atributos.

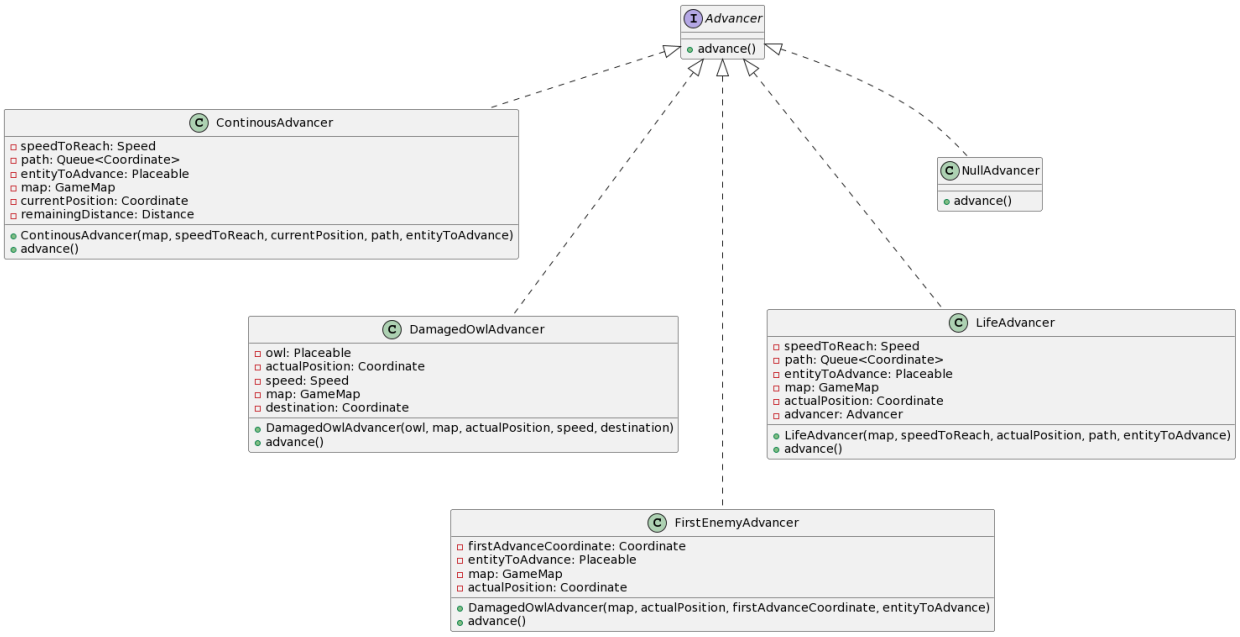

Figura 4: Diagrama de las clases Advancer y sus clases hijas

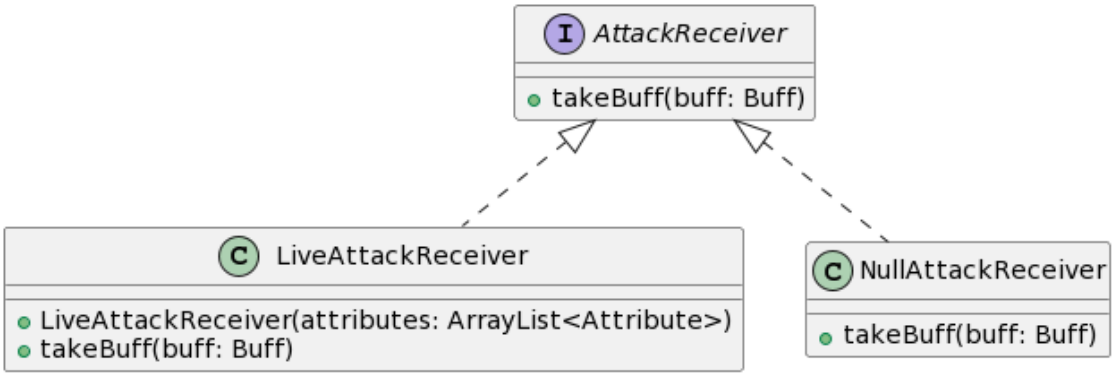

Figura 5: Diagrama de las clases AttackReceiver y sus clases hijas

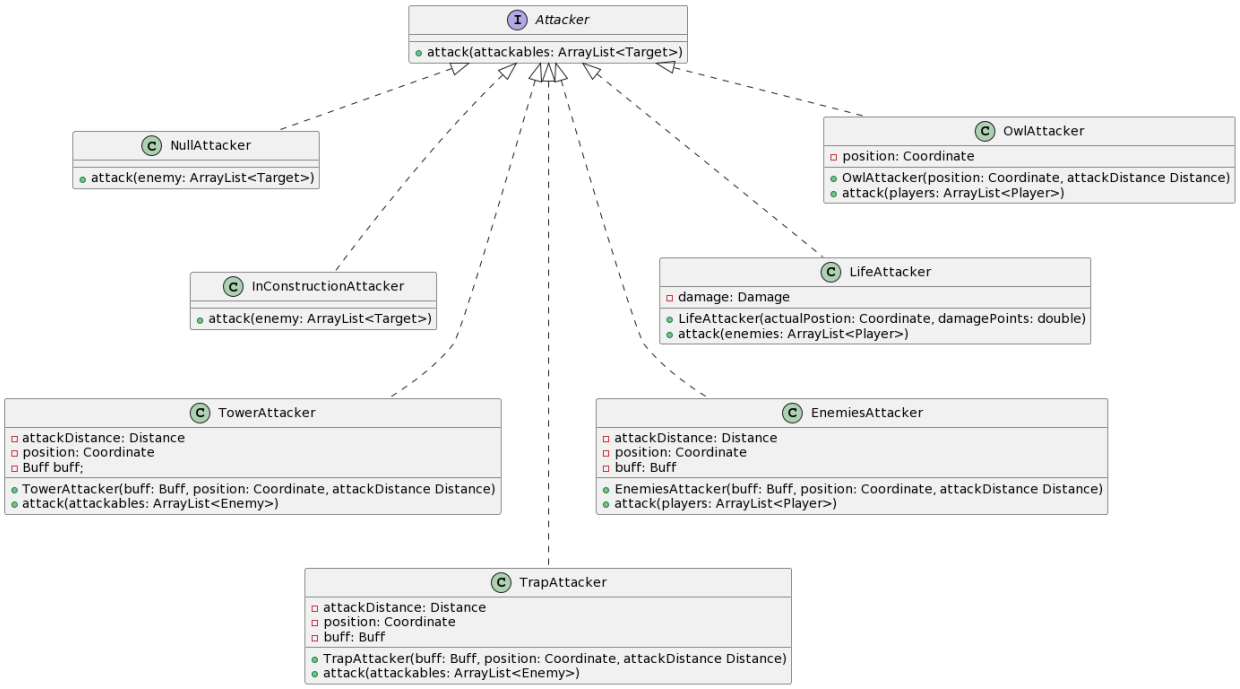

Figura 6: Diagrama de las clases Attacker y sus clases hijas

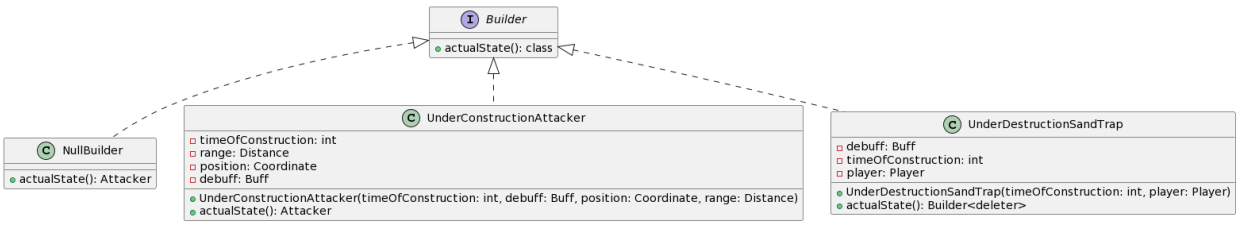

Figura 7: Diagrama de las clases Builder y sus clases hijas

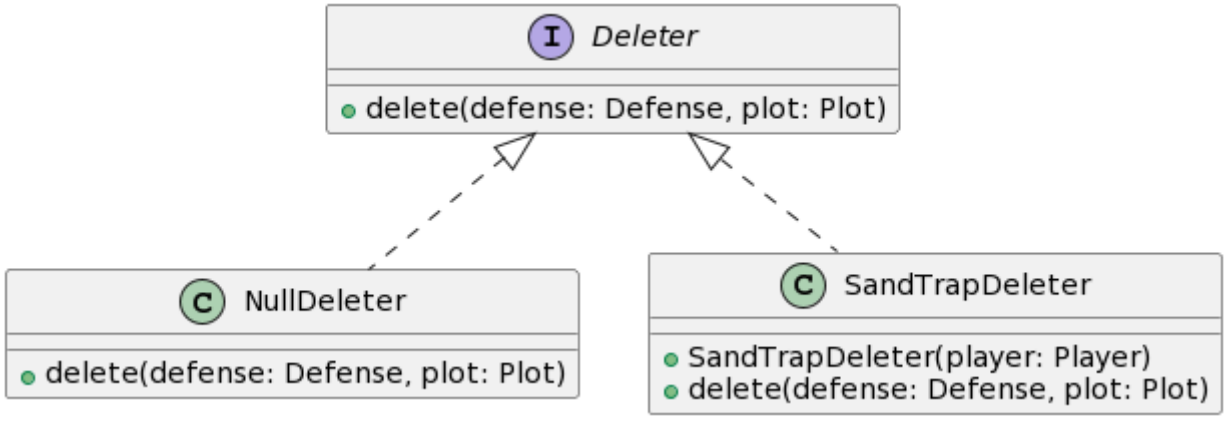

Figura 8: Diagrama de las clases Deleter y sus clases hijas

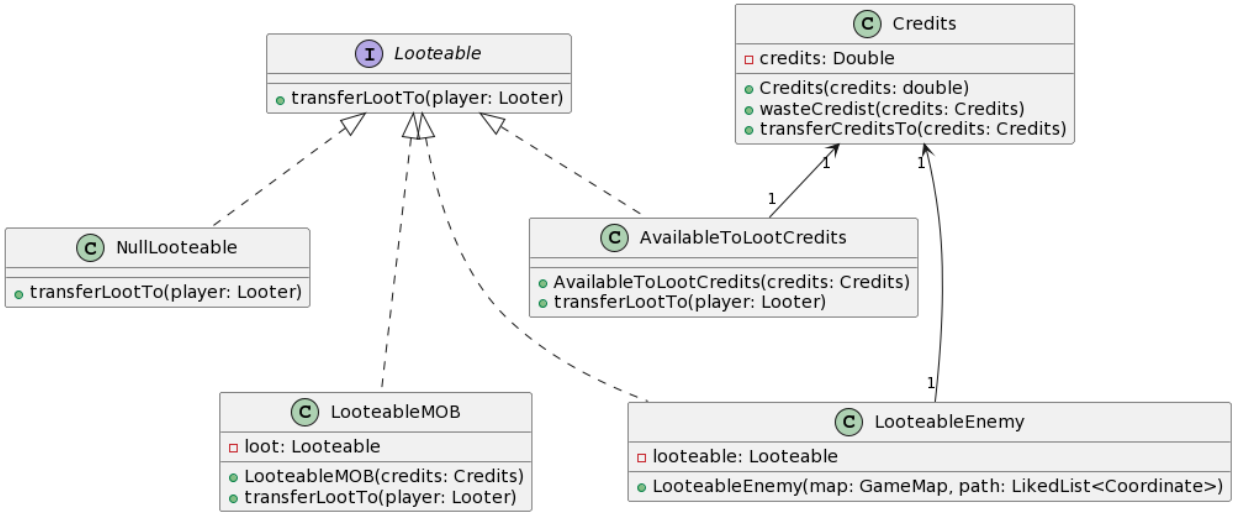

Figura 9: Diagrama de las clases Loot y sus clases hijas

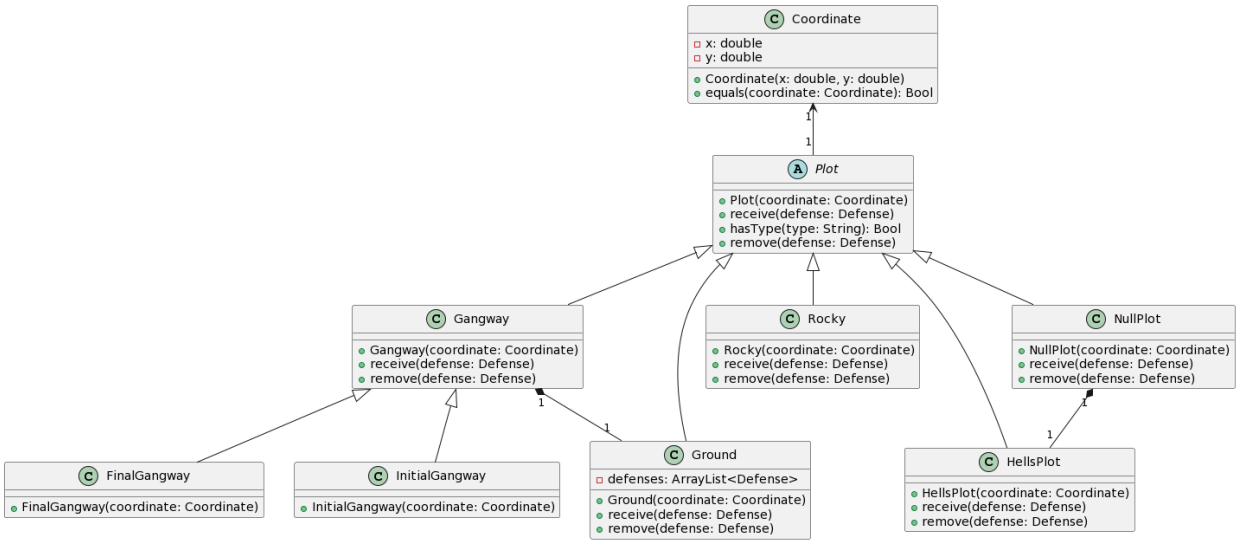

Figura 10: Diagrama de las clases Plot y sus clases hijas

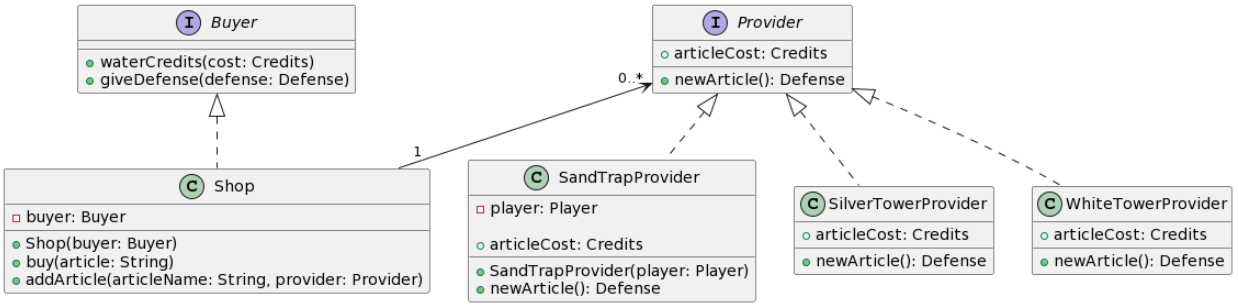

Figura 11: Diagrama de las clases Shop, Provider y sus clases hijas

5. Detalles de implementación

5.1. Game

La clase Game es la clase encargada del manejo general del juego. Todas las acciones relacionadas al usuario interactúan internamente con Game y luego esta delega las responsabilidades a otras clases.

5.2. Patrón Strategy

Se utilizo el patron Strategy para poder cambiar el estado de las torres a medida que van pasando los turnos y acciones del jugador. Por ejemplo: Cuando un torre pasa del estado de en construcción a construida y puede funcionar ya como defensa del jugador. Otro ejemplo puede ser el caso en que un enemigo es afectado por una trampa de arena reduciendo su velocidad.

5.3. Patrón Facade para la lectura de los JSon

Se utilizo este patrón para eliminar complejidad a la hora de la lectura y parseo de los JSon además que permite en un futuro, si se desea, cambiar el archivo de los enemigos o mapa sin requerir muchos cambios en el código

6. Diagramas de Estado

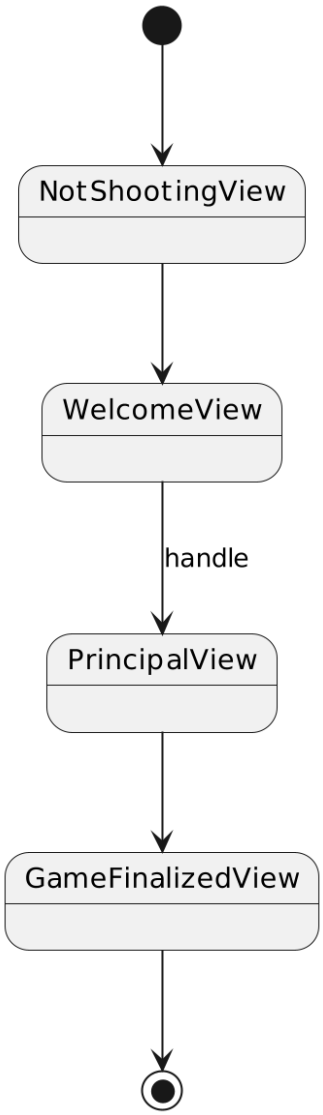

Figura 12: Diagrama de estado la vista

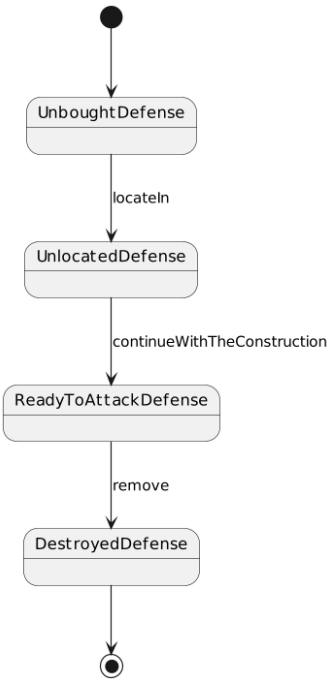

Figura 13: Diagrama de estado de una defensa al ser comprada

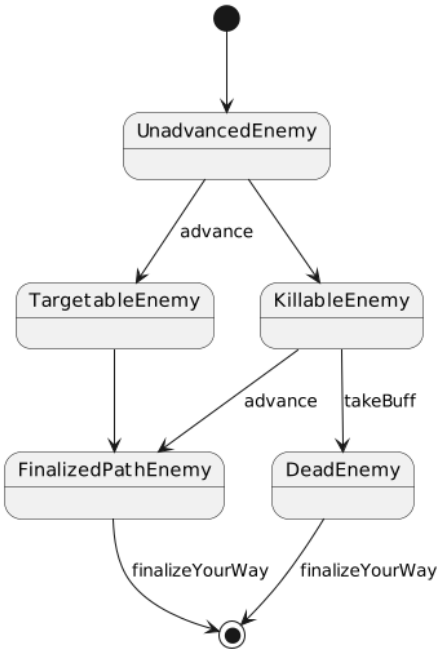

Figura 14: Diagrama estado de un enemigo

7. Diagramas de Paquete

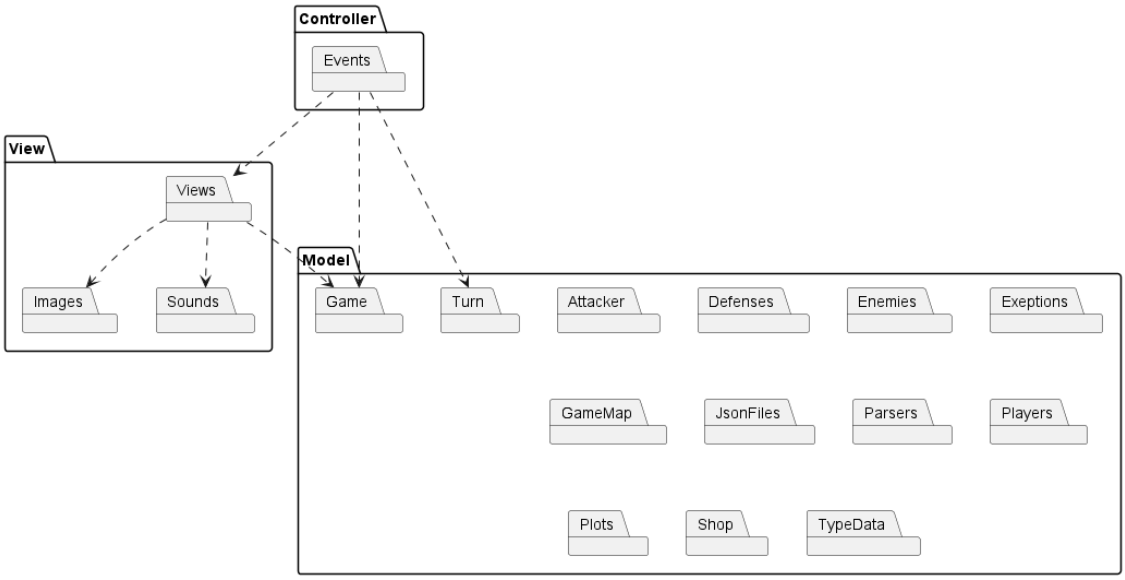

Figura 15: Diagrama de paquete general

8. Excepciones

InsuficientCredits Esta excepción es lanzada cuando el jugador quiere comprar una defenza y no cuenta con los créditos suficientes.

InvalidEnemy Esta excepción fue creada con el objetivo de evitar la creación de enemigos desconocidos en la parseo del JSon.

InvalidJson Esta excepción fue creada para informar al usuario que el JSon que envió no es de la forma correcta.

NonExistentArticle Esta excepción se lanza cuando un jugador quiere comprar un objeto que no existen en la tienda.

WrongPlace Esta excepción avisa al jugador que el lugar que quiere colocar la defensa no es el indicado

WrongPlayerName Esta excepción informa la usuario que el nombre que eligió para registrarse es invalido

9. Diagramas de secuencia

A continuación se muestran algunos diagramas de secuencias de casos de usos del trabajo practico. En el repositorio de GitHub hay otros diagramas que no se logran ver bien en el informe como por ejemplo:

- Como aumenta la velocidad del topo según el turno.
- Como se compra, construye y se hace atacar una defensa.
- Como la Sand Trap afecta a los enemigos que pasan sobre ella.

- Como el búho hace que el jugador pierda una torre al ser atacado.

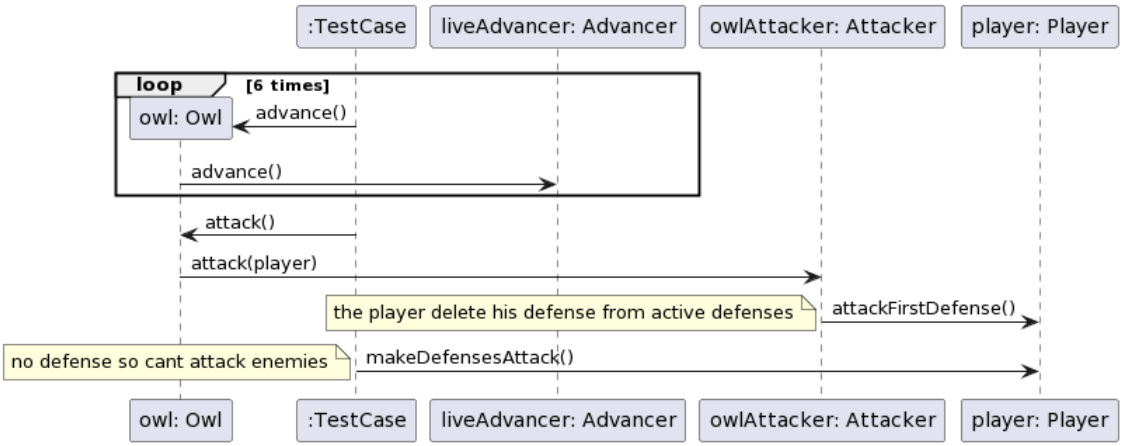

Figura 16: Secuencia de como hace una defensa para atacar a un enemigo.

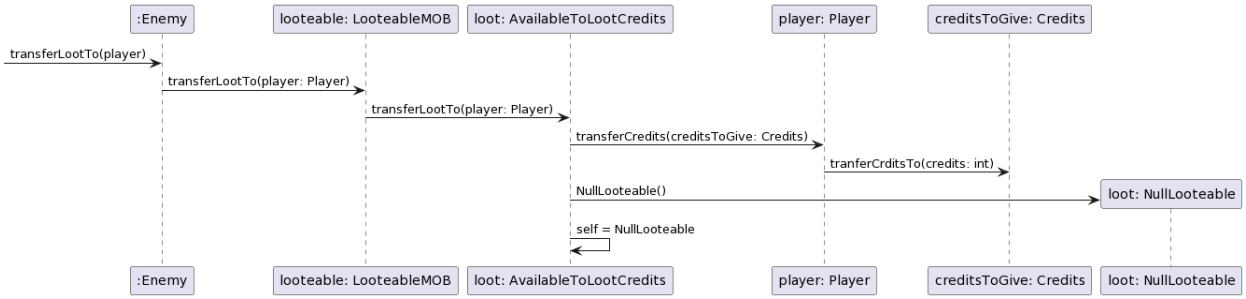

Figura 17: Secuencia de como se otorgan los créditos.

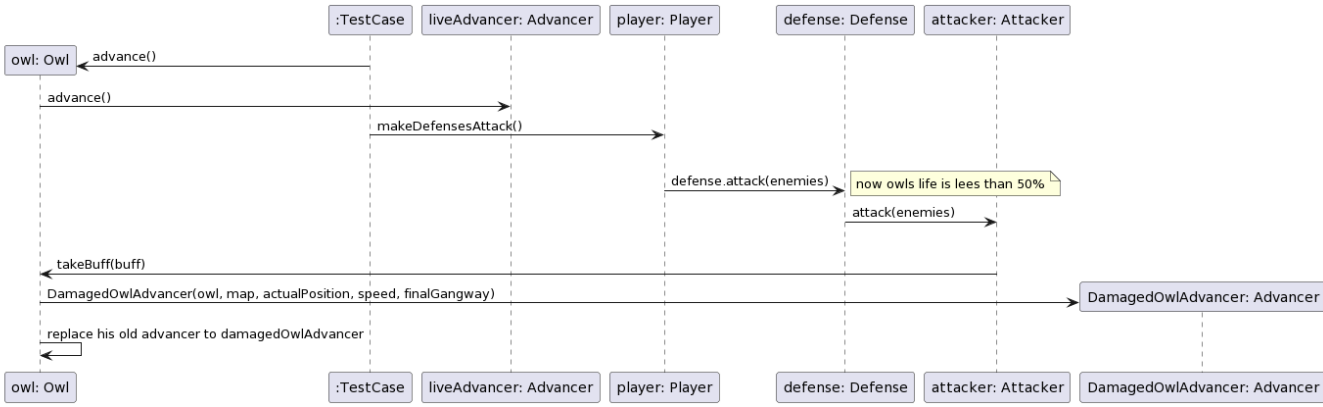

Figura 18: Secuencia de como el recorrido el buho cambia segun su vida.

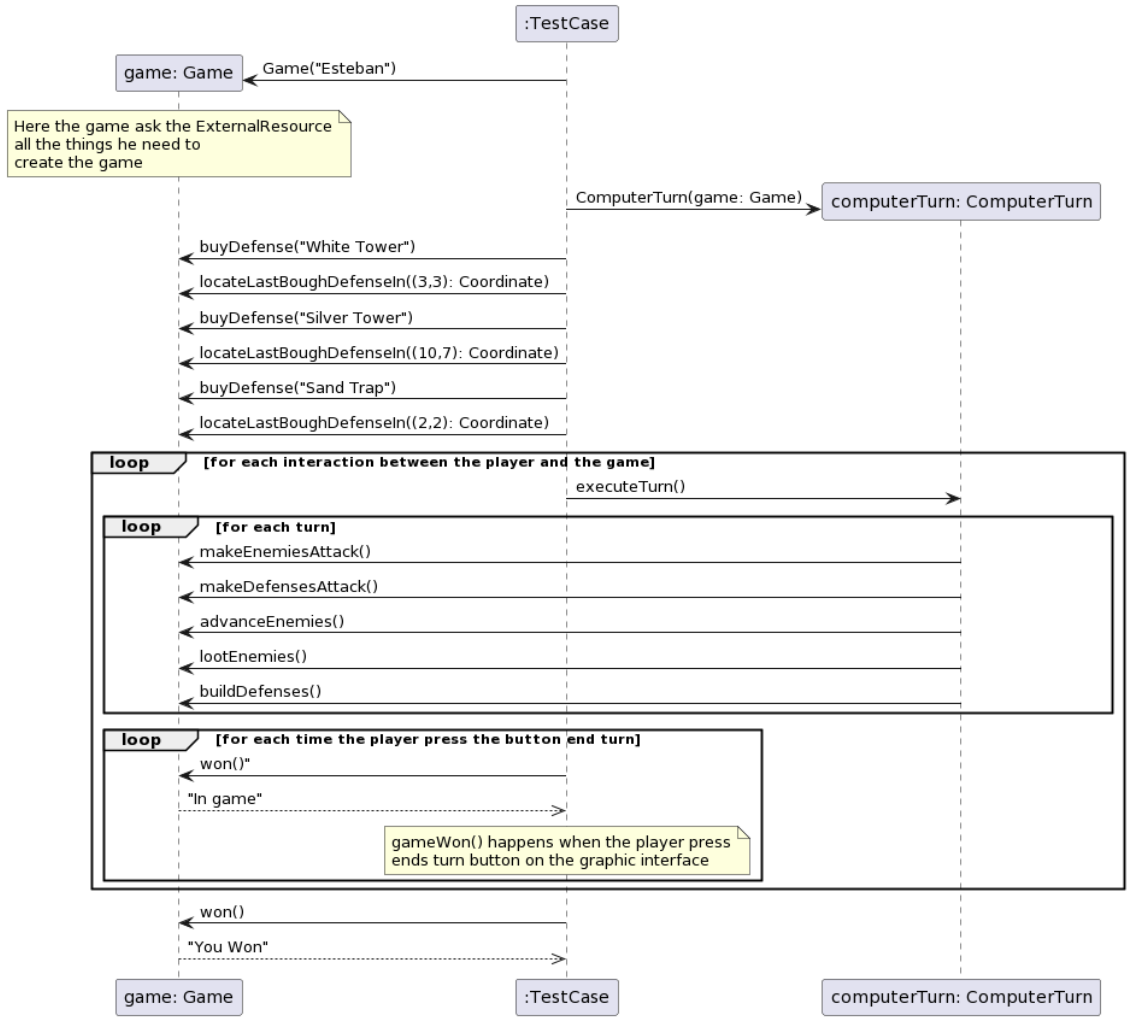

Figura 19: Secuencia de como es el funcionamiento general del juego

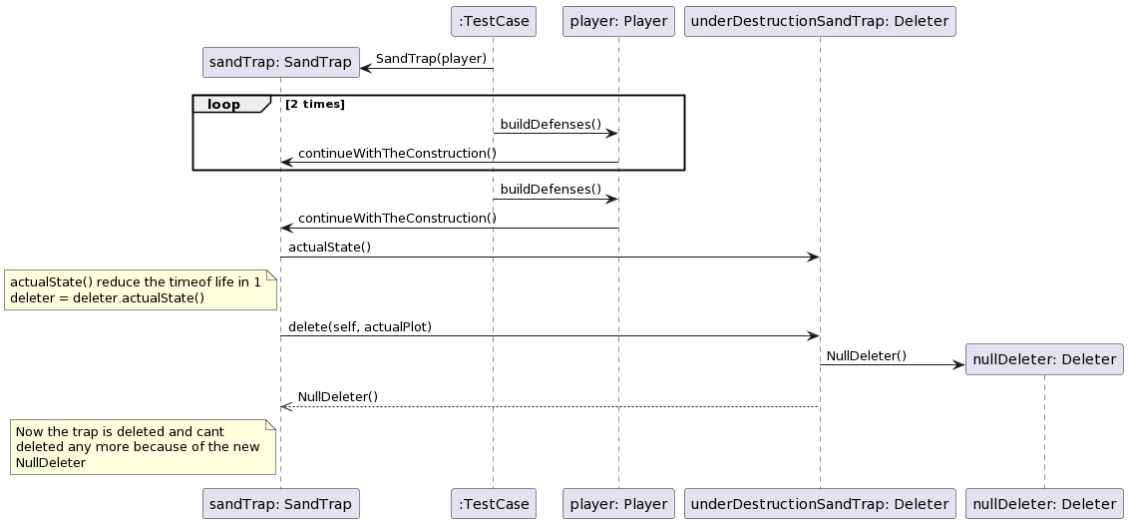

Figura 20: Secuencia de como una sand trap luego de 3 turnos desaparece