FACULTAD DE INGENIERIA
Universidad de Buenos Aires

FIUBA - 75.07

Algoritmos y Programación III

Trabajo práctico 2: Al-Go-Oh!
1er cuatrimestre, 2018
Primer cuatrimestre de 2018

Alumnos:

Nombre	Padrón	Mail
MARCÓ DEL PONT, Matías	101302	matiasmdelp@gmail.com
ILLESCAS, Geronimo	102071	gero17illescas@gmail.com
DEL CARRIL, Manuel	100772	manueldelcarril@gmail.com
ROMERO VÁZQUEZ, Maximiliano	99118	maxi9614@gmail.com

Fecha de entrega final:

Tutor: Tomás Bustamante

Comentarios:

Índice

1. Introducción

El presente informe reune la documentación de la solución del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación que implemente un juego basado en el juego de mesa presente en el manga (historieta japonesa) Yu-Gi-Oh!. Utilizando los conceptos del paradigma de la orientación a objetos y de los patrones de diseño vistos hasta ahora en el curso.

2. Supuestos

Un supuesto tomado, es que el Jugador no puede elegir en que lugar específico colocar sus cartas, es decir, dentro de la "zona de monstruos"no puede elegir si colocarlo en el medio del campo o en un extremo por ejemplo. Se ubican siempre de izquierda a derecha.

3. Modelo de dominio

El programa consta de las siguientes clases:

- **Campo** : es el área de juego de cada jugador, donde ubicará sus cartas. Además contiene al Mazo de dicho jugador [1].
- **Carta** : representa a una Carta del juego Al-Go-Oh; no es una clase abstracta. Es capaz de realizar su efecto o su efecto de volteo. Es instanciada para las cartas magicas o trampas del juego; para los monstruos se utiliza su clase hija, Monstruo.
  - **Monstruo** : es el tipo de carta monstruo. Puede atacar a otros monstruos o defenderse, y al igual que su madre, realizar efectos o no.
  - **DeCampo** : representa a otro tipo de carta que está presente en el juego. Se ubican en un sector especial del tablero. Constan de un efecto que abarca a todo el campo, y aplica a todas las cartas en el.
- **Efecto** : es la interfaz que implementan los efectos o poderes especiales que posee cualquier Carta. Tiene los métodos realizarEfecto y realizarEfectoDeVolteo, cuya diferencia es el momento en que son llamados: el realizarEfectoDeVolteo se llama durante el ataque del oponente, mientras que el otro no (Sus implementaciones dependerán de las relgas de la carta de juego).
  - **EfectoAgujeroOscuro** : es el efecto que posee la carta magica .ªgujero Oscuro", la cual destruye todos los monstruos del campo del jugador y el oponente. No implementa el efecto de volteo.
  - **EfectoAumentar500Ataque** : Este efecto aumenta el ataque del monstruo atacado en 500 puntos antes del calculo de daño; por ello, implementa el efecto de volteo.
  - **EfectoCilindroMagico** : corresponde al efecto de la carta mágica Çilindro Mágico", negando el ataque del monstruo atacante, e inflige el mismo daño directamente a los puntos de vida del oponente.
  - **EfectoDestruirMonstruoAtacante** : hace referencia al efecto que tiene la carta de tipo monstruo llamada Ïnsecto come-hombres". Dicha carta destruye un monstruo en el campo, y únicamente puede activarse cuando pasa de estar boca abajo a boca arriba. Puede ser activada en el momento en que el monstruo es atacado.
  - **EfectoFisura** : representa al efecto que posee la carta mágica "Fisura", que le permite al jugador destruir al monstruo boca arriba con menor ataque en el campo del oponente (en caso de haber empate, elige al azar).

[1] Más adelante se mostrará dicha relación en su respectivo diagrama.

  - **EfectoJinzo7** : es el efecto que contiene la carta Monstruo denominada "Jinzo 7", el cual puede atacar directamente a los puntos de vida del oponente.
  - **EfectoOllaDeLaCodicia** : éste efecto le permite al jugador tomar 2 cartas del mazo. Se encuentra en la carta Mágica .ºlla de la Codicia".
  - **EfectoSogen** : éste efecto aumenta en 500 los puntos de defensa de tus monstruos, y 200 los puntos de ataque de los monstruos de tu oponente. Se encuentra en la carta De Campo "Sogen".
  - **EfectoVacio** : hace referencia a los monstruos que no tienen efecto.
  - **EfectoWasteland** : dicho efecto aumenta en 200 puntos el ataque de tus monstruos, y 300 puntos la defensa de los monstruos de tu oponente, y corresponde a la carta De Campo "Wasteland".

- **Invocacion** : es una interfaz y representa a la forma que se debe invocar un Monstruo. Varia según su cantidad de estrellas u otro hecho en particular de cada carta. Monstruo delegará la realización de los sacrificios en alguno de los objetos que implementan esta interfaz.

  - **Invocacion1Sacrificio** : éste tipo de invocación requiere realizar el sacrificio de un monstruo del campo. Corresponde a aquellas cartas monstruo que tienen 5 ó 6 estrellas.
  - **Invocacion2Sacrificio** : dicha forma de invocar a un monstruo, requiere realizar el sacrificio de dos monstruos del campo. Corresponde a aquellos monstruos cuyo cantidad de estrellas es 7 ó más.
  - **InvocacionDragonDefinitivoDeOjosAzules** : representa a la única forma de invocar al "Dragon definitivo de ojos azules"que es sacrificando 3 dragones azules, que se encuentren en el campo de juego.
  - **InvocacionNormal** : éste tipo de invocación no requiere ningún sacrificio, y se encuentran en las cartas monstruos cuyas estrellas no superan las 4.

- **Jugador** : representa al jugador. Ataca a los Monstruos del Campo del oponente con sus Monstrous que colocó sobre su propio Campo. También coloca cartas maágicas o trampas sobre el campo y es capaz de activarlas.

- **Estado** : representa el estado del juego (ganador o no). Es la clase que decide si un Jugador gano o no.

- **Mazo** : representa una baraja de cartas del juego en cuestión, conteniendo todas las cartas que un jugador puede tomar.

- **AlGoOh** : Representa el ”organizador” o ”referi” del juego. Es quien se encarga de llevar la cuenta de los turnos y fases de cada Jugador.

- **Fase** : Es la interfaz que implementan las distintas fases del juego. Jugador siempre antes de realizar una acción llama al método correspondiente en la fase, generando así que se lance una excepción cuando la acción es inválida. Las distintas fases implementarán estos métodos de manera distinta dependiendo de las reglas del juego. Éstas son: **FasePreparacion**, **FaseAtaque** y **FaseFinal**.

- **Posicion** : es una 'interface' y representa las posiciones que puede tener una Carta del tipo Monstruo.

  - **PosicionAtaque** : es la implementación de la posicion de ataque del Monstruo y es en quien Monstruo delega el cálculo de daño.
  - **PosicionDefensa** : es la implementación de la posición de defensa del Monstruo y su responsabilidad es la misma que la de PosicionAtaque.

4. Diagramas de clase

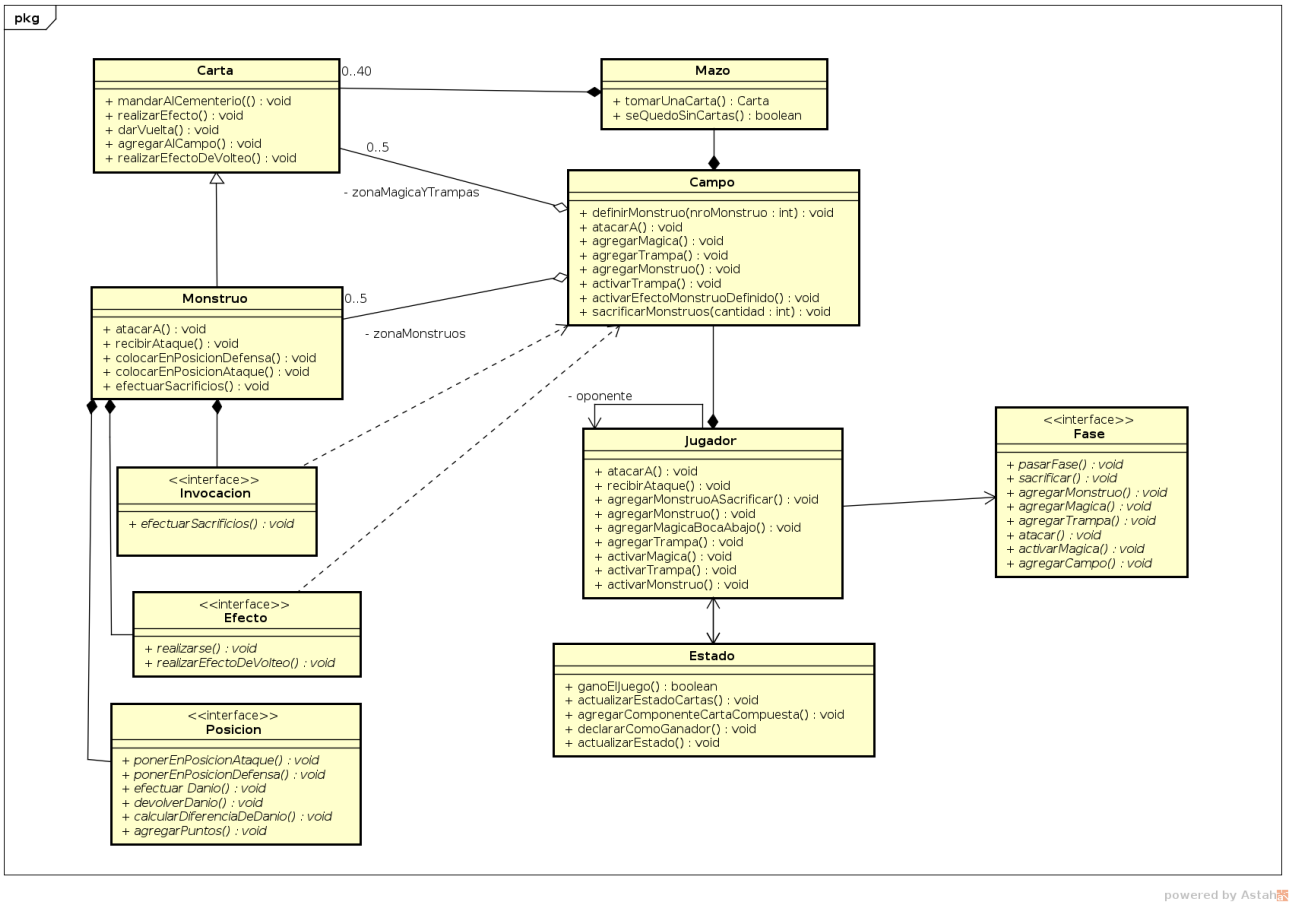


Figura 1: Diagrama de Clase principal.

En este diagrama se pueden observar las clases mas abarcativas del programa y sus relaciones.

Tenemos en primer lugar a la clase Jugador quien como dicho anteriormente coloca cartas y comanda ataques. Para ello utiliza a Campo sobre quien delega la gran mayoría de estas responsabilidades. Posee también un estado, quien se encarga de saber cuando el jugador es ganador o no.

Jugador tiene como atributo una referencia a un objeto que implementa la interfaz de Fase. Cada vez que Jugador desea realizar una acción, llama al método correspondiente de Fase. El objeto de acuerdo a su implementación, levantará una excepción o no (según las reglas del juego). El Jugador re lanzará esa excepción a la clase cliente de Jugador para notificarle de la imposibilidad de usar el método en cuestión en ese momento.

Campo posee colecciones de Cartas y Monstruos para almacenar las colocaciones de Jugador. Cada vez que Jugador realiza un ataque o activa un efecto, Campo es el encargado de localizar a la Carta en cuestión y notificarle de la acción. Monstruo a su vez vuelve a delegar el ataque en Posicion (ver más adelante).

A la hora de realizar efectos, la secuencia de mensajes es similar (Jugador le avisa a Campo de activar el efecto de un Monstruo; Campo le avisa a Monstruo y Monstruo llama al método de Efecto) , exceptuando que cuando Monstruo delega en Efecto, este último vuelve a delegar en Campo (en la mayoría de los casos). Por eso la flecha de ”Usage” en el diagrama.

Lo mismo vuelve a suceder con las invocaciones, donde al ir delegando llega finalmente a Invocacion. Quien vuelve a delegar en campo. Por eso nuevamente tenemos la flecha de Üsage".

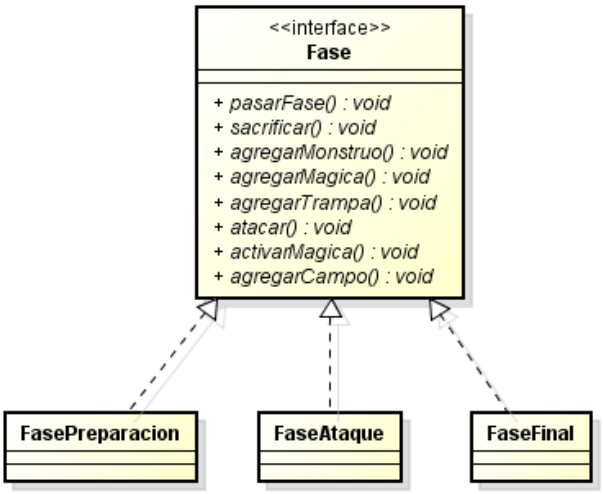


Figura 2: Diagrama de Clase de implementación de la interfaz Fase.

Aquí se muestran las tres fases del juego que implementan a la *interface* Fase. Como se dijo en la sección 3, cada una implementa los métodos correspondientes de una forma distinta según el juego en cuestión.

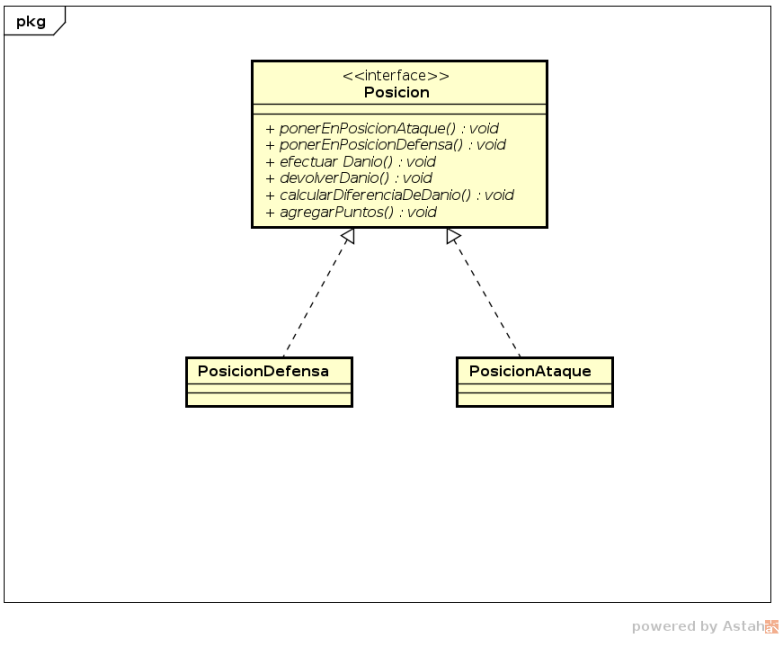


Figura 3: Diagrama de Clase de implementación de la interfaz Posicion.

En este diagrama podemos ver todas las clases que implementan Posicion. Sobre estas clases, Monstruo delegará la responsabilidad de la realizacion de daño y defensa de él mismo. Estas clases diferirán en cuanto a la implementación de los distintos métodos de acuerdo a las reglas del juego.

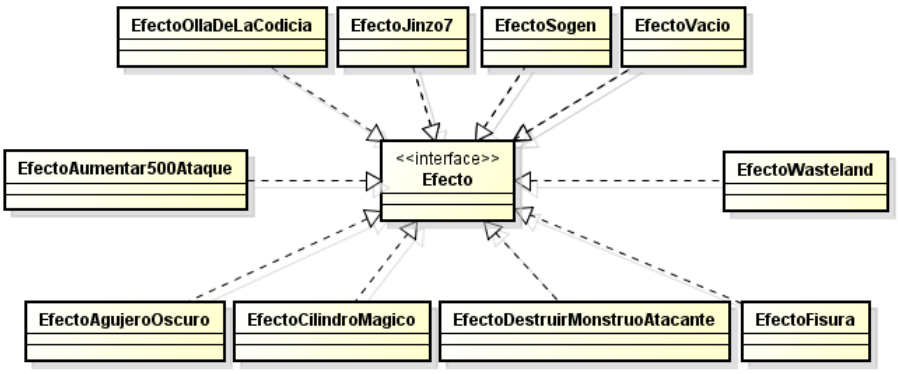


Figura 4: Diagrama de Clase de implementación de la interfaz Efecto.

En dicho diagrama muestra en donde se implementa la *interface* Efecto. Se puede ver que los diversos efectos corresponden a una respectiva carta, como se explica en el sección 3. En el caso de una carta Monstruo que no tiene ningún efecto, se le asigna EfectoVacio, que como su nombre lo indica, no realiza ninguna acción.

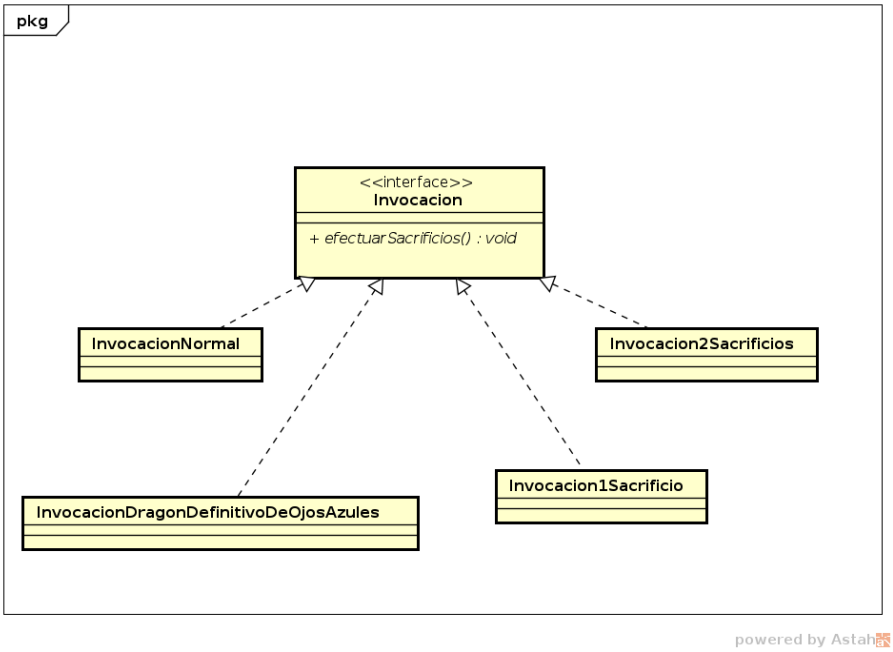


Figura 5: Diagrama de Clase de implementación de la interfaz Invocacion.

En este diagrama se pueden ver todas las clases que implementan la interfaz de Invocacion. Todas las invocaciones implementan el método de acuerdo a las reglas del juego, exceptuando InvocacionNormal cuyo método no hace nada puesto que dicha invocación no presenta requisito alguno.

5. Diagramas de secuencia

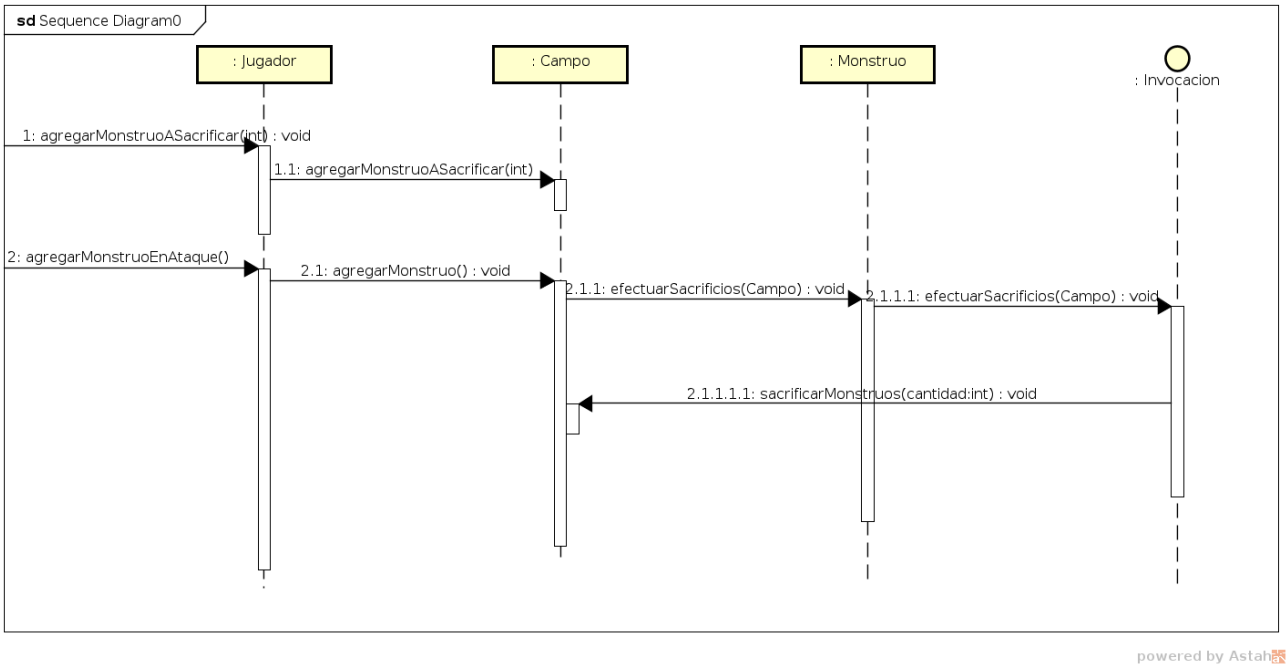


Figura 6: Diagrama de Secuencia del proceso de invocar un Monstruo.

Como podemos observar, éste diagrama de secuencia expone el proceso que ocurre cuando un Jugador quiere invocar un Monstruo. Lo primero que hace el jugador es agregar monstruos que va a ofrecer como sacrificio en su proxima invocacion. Para ello le envía un mensaje a Campo.

Luego, agrega el monstruo que desea al Campo. Campo antes de agregarlo, se asegura que cumpla los requisitos de invocacion llamando al método del propio Monstruo efectuarSacrificios y pasándose como parámetro a sí mismo. Monstruo a su vez, delega en su Invocación, quien dependiendo de su implementación llama al método correspondiente de Campo para que realice los sacrificios.

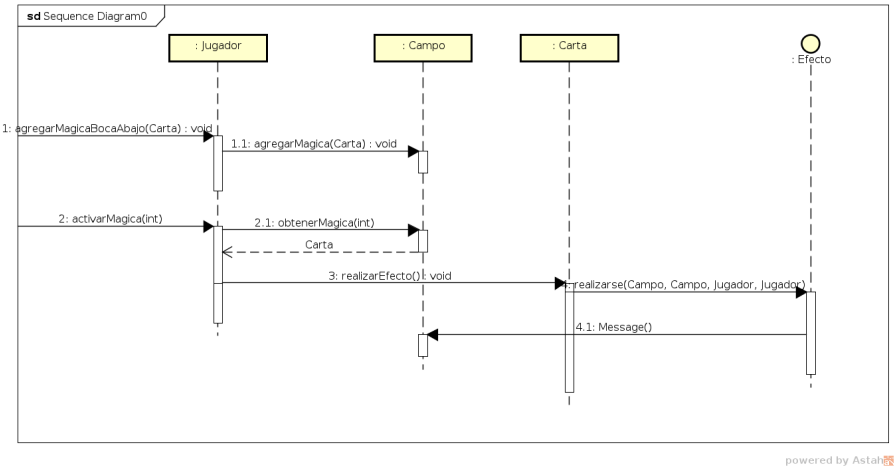


Figura 7: Diagrama de Secuencia del proceso de un efecto de la carta Magica.

La figura 7 muesta el diagrama de secuencia del proceso en el cual una carta del tipo Magica realizar su efecto.

El primer paso lo realiza la clase Jugador agregando la carta magica en cuestión. El Campo la agrega.

Luego, Jugador le pide la carta mágica que desea activar y Campo se la devuelve, permitiéndole a Jugador enviarle el mensaje de realizarEfecto a la carta.

A continuación, la Carta le manda el mensaje de realizarse a Efecto quién de acuerdo a qué implementación sea, mandará los mensajes que tenga que mandar los cuales en este caso, desconocemos (suponemos que le enviará alguno a Campo).

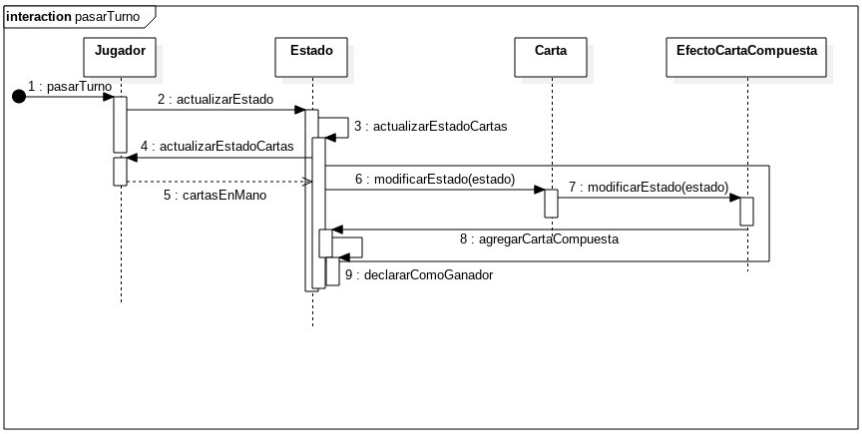


Figura 8: Diagrama de Secuencia del proceso de pasar el turno de un jugador a otro.

La imagen muestra como interactua una instancia Jugador con su propio estado actualizandolo cada vez que se llama el metodo de pasarTurno en la situacion de que Jugador tenga las 5 cartas de Exodia. El estado lo que hace pedirle las cartas en mano al jugador e iterarlas de manera que llama al metodo de modificarEstado de cada carta. Carta a su vez delega el comportamiento a su efecto, que en el caso de las cartas Exodia es el de cartaCompuesta. Este metodo le avisa al estado de que hay cartas de Exodia.

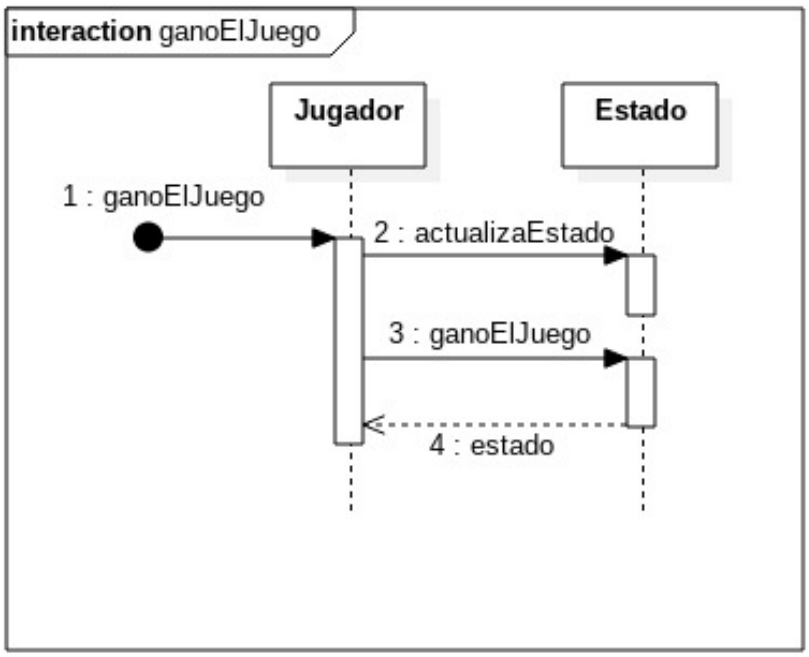
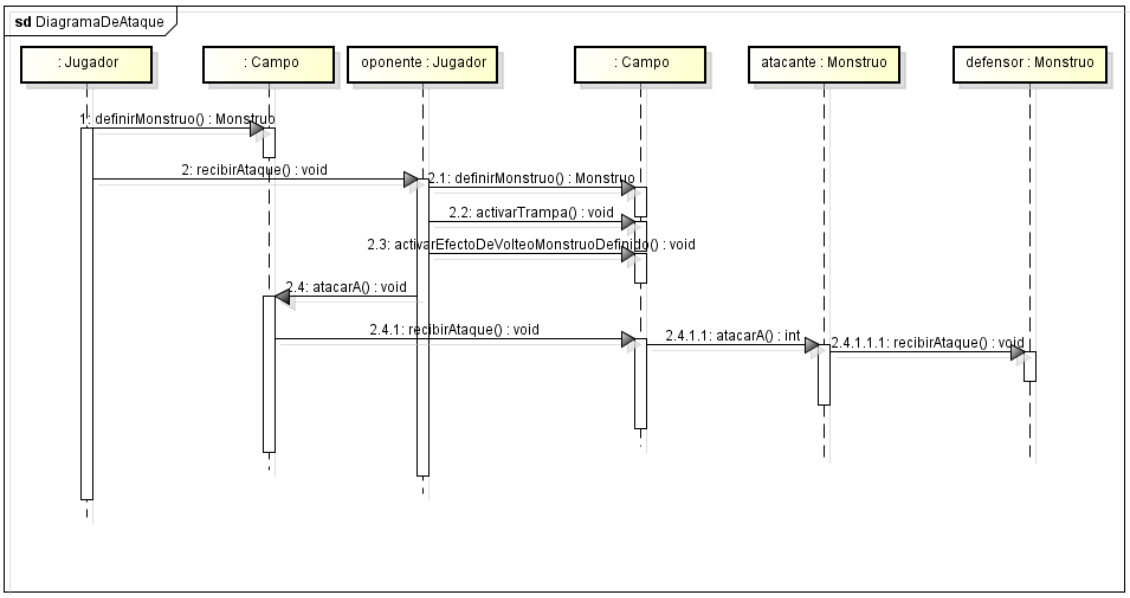


Figura 9: Diagrama de Secuencia del ver si un juagdor gano.

Se ve que es una secuencia simple en la que antes de preguntarle al estado si el jugador gano, se actualiza el estado.

6. Detalles de implementación



En este diagrama podemos ver la secuencia de mensajes que se suceden al invocarse el método de atacarA de Jugador. Primero se obtiene el monstruo atacante, y luego se delega el calculo

de daño en el oponente. Así el oponente busca el monstruo atacado en su campo y deja que se "peleen"los monstruos.

En esta última parte, podemos ver un Double Dispatch donde el atacante le manda el mensaje con sus puntos de ataque al atacado y este último es quien termina realizando el cálculo de daño.

Así se desencadena la devolución de mensajes, donde cada objeto hará los ajustes necesarios a sus respectivos estados de acuerdo al resultado.