FIUBA - 75.07

Algoritmos y programación III

Trabajo práctico 2: Al-Go-Oh!

1er cuatrimestre, 2018

(trabajo grupal de 4 integrantes)

Alumnos:

Nombre	Padrón	Mail
Bellaera, Leonardo	100973	leobellaera@gmail.com
Gómez Hobbs, Fernando	94811	fgomezhobbs@gmail.com
Kler, Alejandro	100596	alejandrokler@gmail.com
Rodríguez Peluffo, Pedro	99883	ppedro.rod@gmail.com

Fecha de entrega final: 06/07/2018

Tutor:Diego Sánchez

Comentarios:

- **Supuestos:**
    - Consideramos que un Monstruo en Modo Defensa no puede realizar un ataque. No está aclarado en la consigna explícitamente.
    - Por default, los monstruos se inicializan en Modo Defensa. Puede cambiarse apenas se crea llamando al método colocarEnModoAtaque().
    - Las cartas se inicializan BocaAbajo por defecto, esto también nos facilita no aplicar efectos de volteo en momentos incorrectos.
    - Si un Monstruo ataca a otro Monstruo que se encuentre en Modo Defensa, éste último no infligirá daño a su Jugador dueño pero sí puede infligir daño al Jugador atacante (por estar en Modo Ataque).
    - Cuando se seleccionan los Monstruos a sacrificar, se evalúa solamente que la cantidad sea la necesaria o que estén los Monstruos correctos. Si esto se cumple pero hay más Monstruos seleccionados, se sacrificarán los necesarios y se destruirán los demás.

- **Modelo de dominio:**
  AlGoOh: Utilizo la clase como interfaz entre el usuario y el modelo, pensándola como la capa exterior del programa que tiene la responsabilidad de realizar un intercambio directo de mensajes (sin tener en cuenta ninguna interfaz gráfica o menú con facilidades) que permitirá al jugador realizar acciones sobre el juego cuando corresponda.

  Fase: Es la clase que se encarga de permitir al Jugador realizar acciones según sea el momento.

  Jugador: Representa a uno de los jugadores que interactuará con el programa permitiéndole realizar cambios en el juego, y tendrá su tablero con zonas y su mano con cartas. Conociendo a su oponente, realizará ataques en su tablero.

  Monstruo: Es una clase abstracta que define algunos métodos comunes a todos los monstruos y permite a sus subclases sobrecargarlos para realizar las acciones de manera distinta según corresponda a su comportamiento. Al estar colocadas en ZonaMonstruos, pueden atacar a otros Monstruos en la FaseAtaque y destruirlos, dañando o no al Jugador que lo posea.

  Magica: Es una clase abstracta que funciona de igual manera que la anterior, declarando métodos para cartas de tipo mágico. Se pueden aplicar en la FaseMagicas para causar efectos que dependen de cada carta.

  Trampa: Es, también, una clase abstracta que declara métodos para cartas de tipo trampa. Si se juegan en la ZonaMagicasYTrampas de un Jugador, serán seleccionadas en orden para aplicarse en los ataques de un Monstruo a otro. Estas primero aplican su efecto trampa y luego, según corresponda, dan lugar al ataque o no.

Campo: También clase abstracta, si es jugada en la ZonaCampo puede aplicar su efecto previo a un ataque para modificar los valores con que se producirá el enfrentamiento. Una vez realizado, se desactiva. De esta forma, evitamos aplicar Campo a las cartas que se juegan durante la FasePreparacion y simplemente aplicamos su efecto cuando es necesario. Además, evita que ante un cambio de Campo la carta continúe con el efecto anterior.

- **Diagramas de clases:**

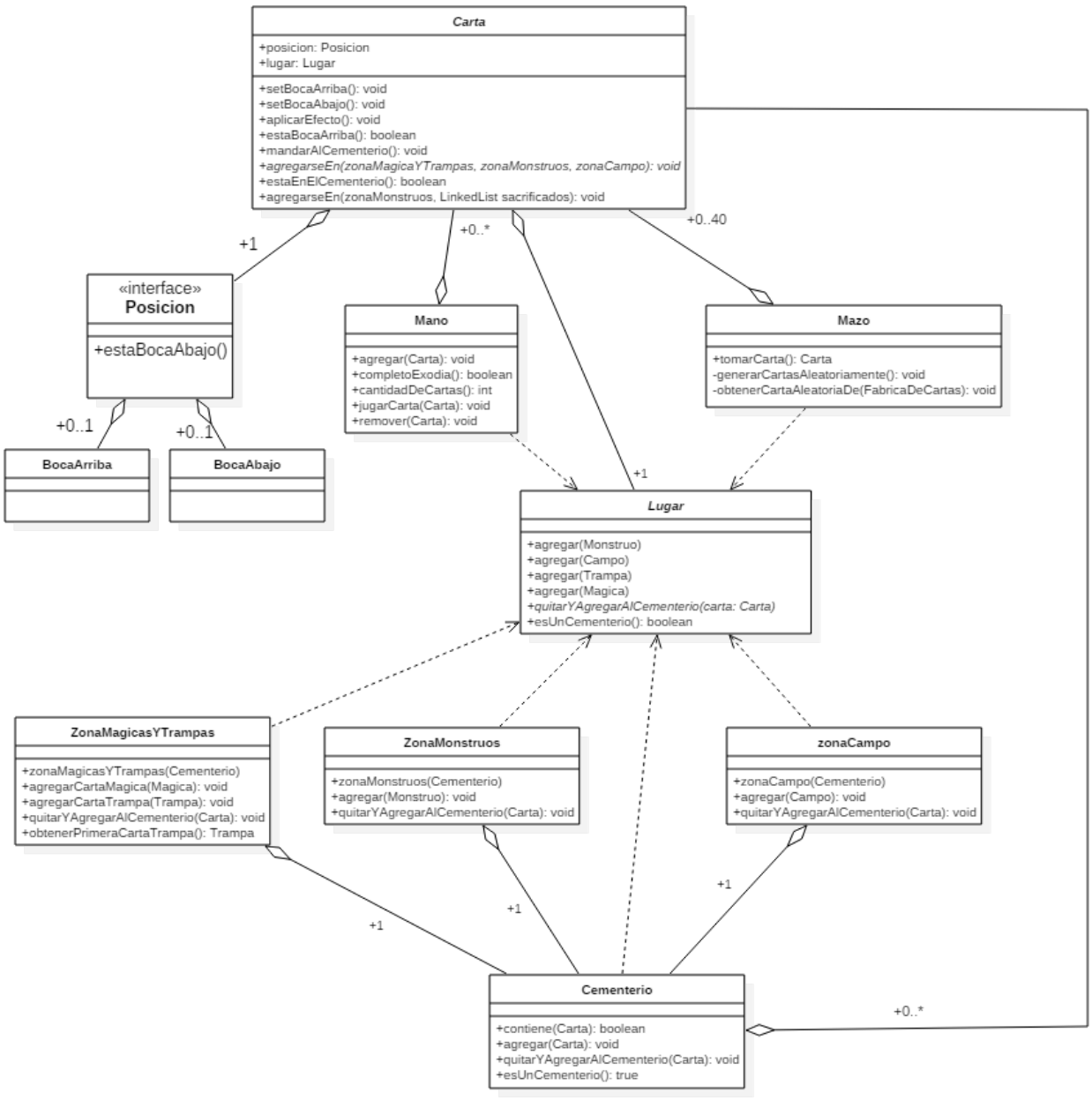


Diagrama de las clases relacionadas con Carta y las zonas donde puede estar (Lugar).

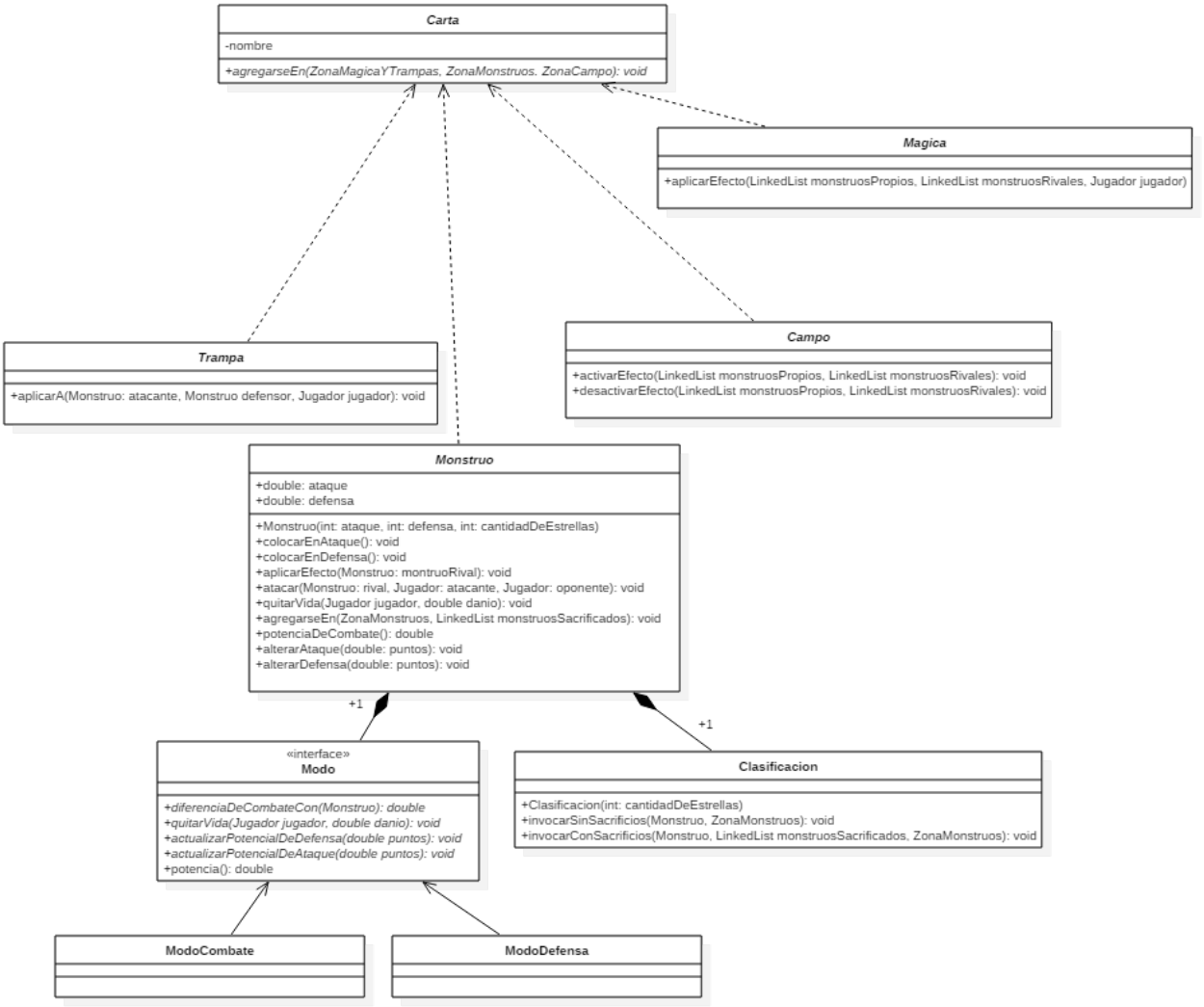


Diagrama de las clases que modelan cada tipo de Carta.

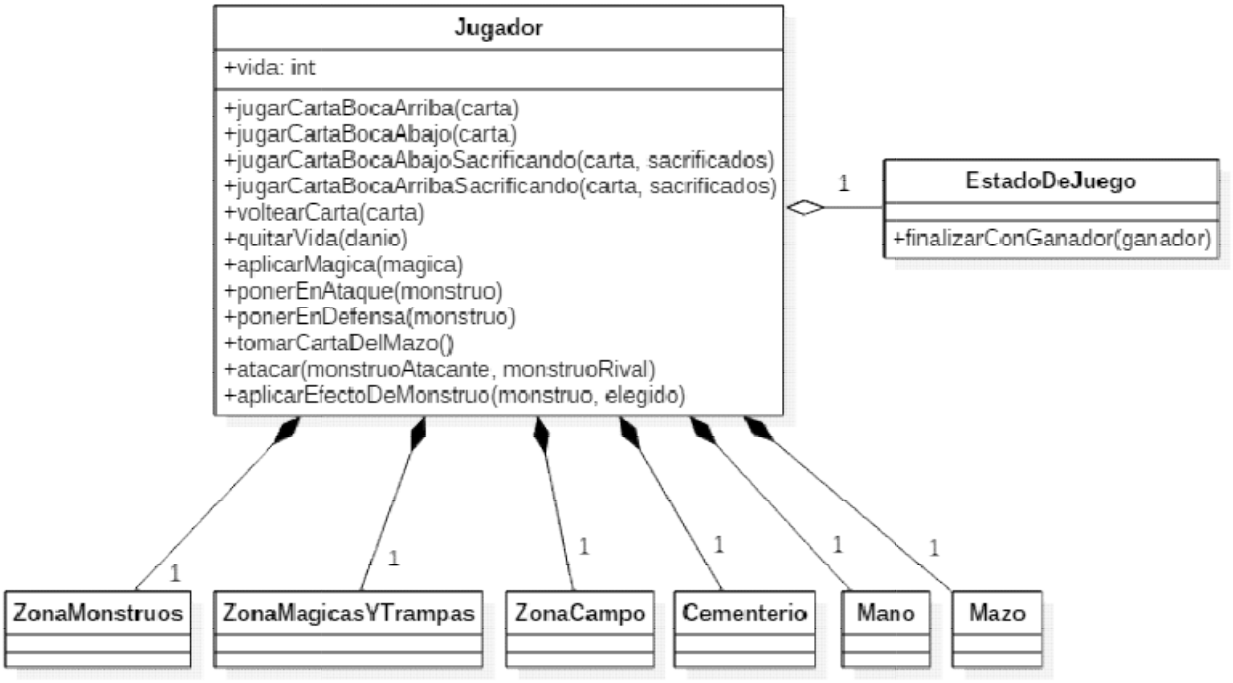


Diagrama de la clase Jugador y las zonas donde mueve sus cartas. Tiene una referencia al EstadoDeJuego para terminar el juego cuando corresponde.

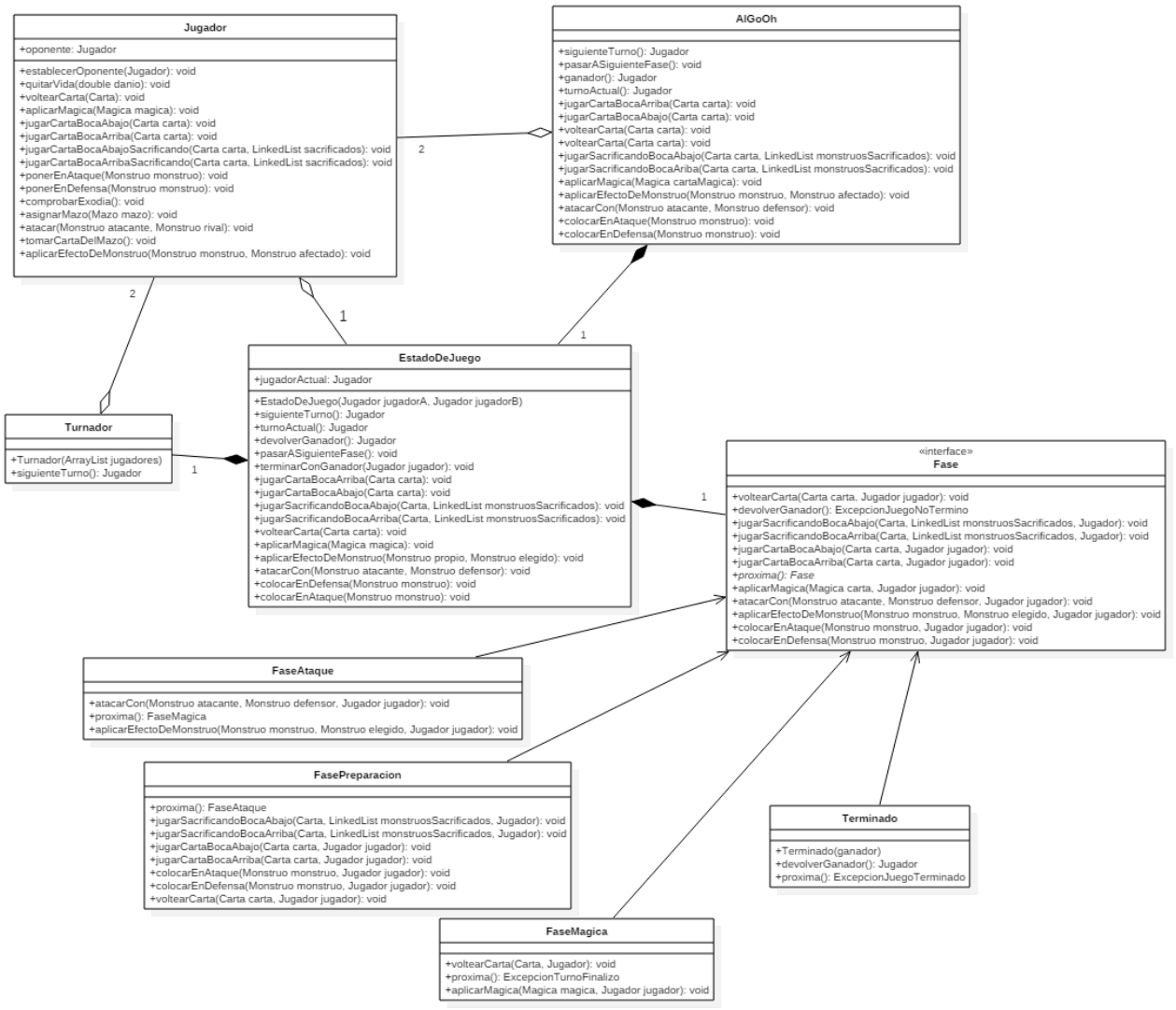


Diagrama de clases del juego y sus fases.

- **Detalles de implementación:**

  Utilizamos los patrones Null Pattern, Strategy y State para modelar distintos comportamientos de una misma clase.

  Utilizamos el patrón Double Dispatch para métodos que realizaban una misma tarea con una pequeña diferencia en los parámetros necesarios.

  Utilizamos los patrones MVC y Observer para el diseño de la interfaz gráfica.

  Utilizamos Factory Method para la creación de cartas y las tomamos cuasi-aleatoriamente para generar un Mazo.

  No le delegamos la responsabilidad de atacar con Monstruos y Trampas al Campo ya que son 2 Campos los que entran en juego y deben activarse para distintos Jugadores y Monstruos, y luego desactivarse.

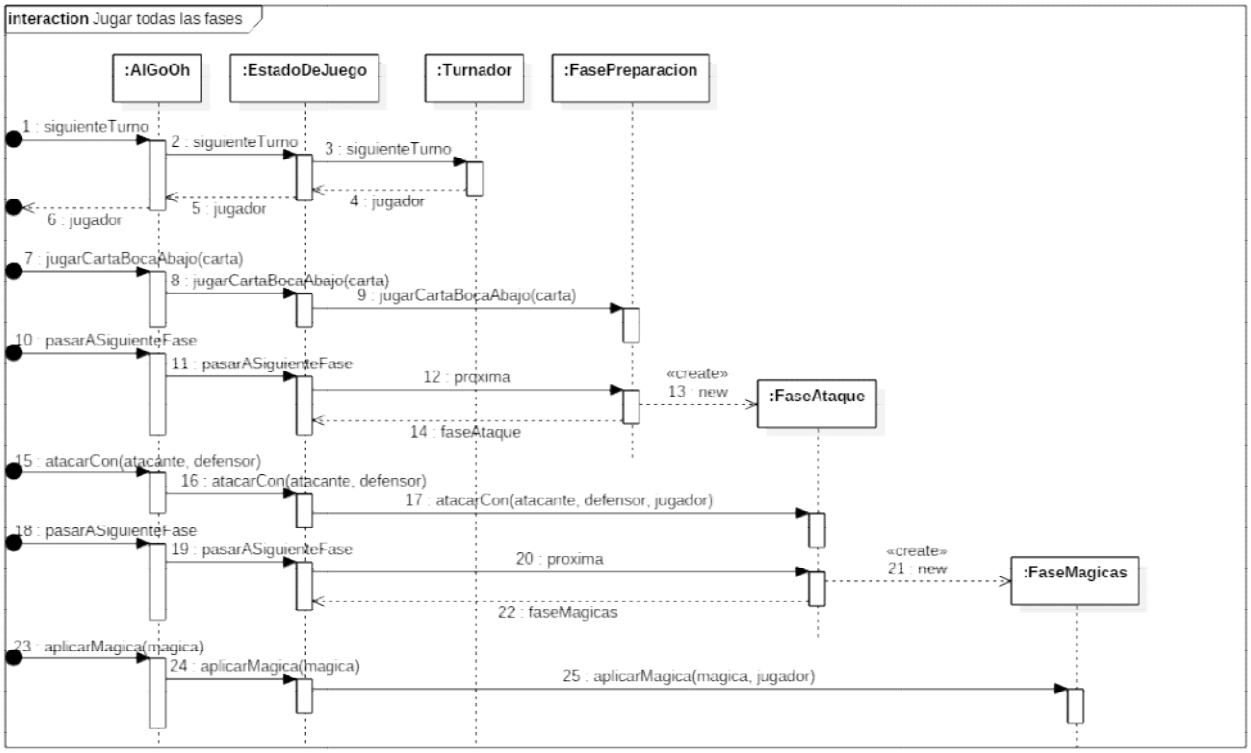
  A Monstruo le paso Jugador (y no un objeto Vida) para notificar al Jugador que está siendo atacado y éste poder cambiar el EstadoDeJuego a Terminado declarando ganador al oponente (finalizarConGanador(ganador)).

  AlGoOh se inicializa con un turno al azar sin ser solicitado (sin llamar al método siguienteJugador()) para evitar NullPointers.

- **Excepciones:**
  1. *ExcepcionCartaBocaAbajo:* Se lanza cuando se intenta aplicar un efecto o atacar con una carta que se encuentra boca abajo.
  2. *ExcepcionCartaNoNecesitaSacrificios:* Se lanza cuando se intenta sacrificar monstruos para jugar una carta que no lo necesita.
  3. *ExcepcionEfectoSoloAplicableEnVolteo:* Se lanza cuando se intenta aplicar un efecto que sólo se aplica cuando se pone la carta boca arriba.
  4. *ExcepcionFaseIncorrecta:* Se lanza cuando se intenta llamar a un método determinado del jugador en una fase del juego que no corresponde.
  5. *ExcepcionJuegoNoTermino:* Se lanza cuando se intenta devolver un ganador y el juego está en proceso.
  6. *ExcepcionJuegoTerminado:* Se lanza cuando se intenta pasar a siguiente fase y el juego cortó su proceso.
  7. *ExcepcionMazoVacio:* Se lanza cuando se intenta agarrar una carta del mazo y este ya quedó vacío. Ésta es atrapada por el jugador para finalizar el juego con el oponente como ganador.
  8. *ExcepcionMonstruoNoPuedeAtacar:* Se lanza cuando se intenta atacar con un monstruo que está en modo defensa.
  9. *ExcepcionMonstruoYaAtaco:* Se lanza cuando se intenta atacar nuevamente con un monstruo en un mismo turno.
  10. *ExcepcionNoEsPosibleAgregarAlCementerio:* Se lanza cuando se intenta mover una carta al cementerio pero ésta ya está ahí, o se encuentra en la mano ó en el mazo.

11. ExcepcionSacrificiosInsuficientes: Se lanza cuando se intenta invocar un monstruo de 5 o más estrellas y no se envían los sacrificios necesarios.
12. ExcepcionTurnoFinalizo: Se lanza cuando se intenta pasar a siguiente fase y ésta fue la última del turno.
13. ExcepcionZonaCompleta: Se lanza cuando se intenta jugar una carta en una zona que ya tiene el máximo permitido de cartas.
14. ExcepcionZonaIncorrecta: Se lanza cuando se intenta jugar una carta de un tipo que no corresponde al tipo de la zona.

- **Diagramas de secuencia:**



*Para comenzar un turno, se llama al método **siguienteTurno()** que devuelve el Jugador al cual le toca jugar y queda almacenado en EstadoDeJuego.*

*Para avanzar en las fases, se llama al método **pasarASiguienteFase()** que permite al Jugador realizar determinadas acciones que correspondan a la Fase próxima.*

*Para llamar métodos del jugador, se pueden utilizar **varios métodos de AlGoOh** que a través del EstadoDeJuego y sus Fases controlará en qué momento se pueden utilizar.*

De no ser posible realizar una acción durante determinada Fase, se lanzará ExcepcionFaseIncorrecta.

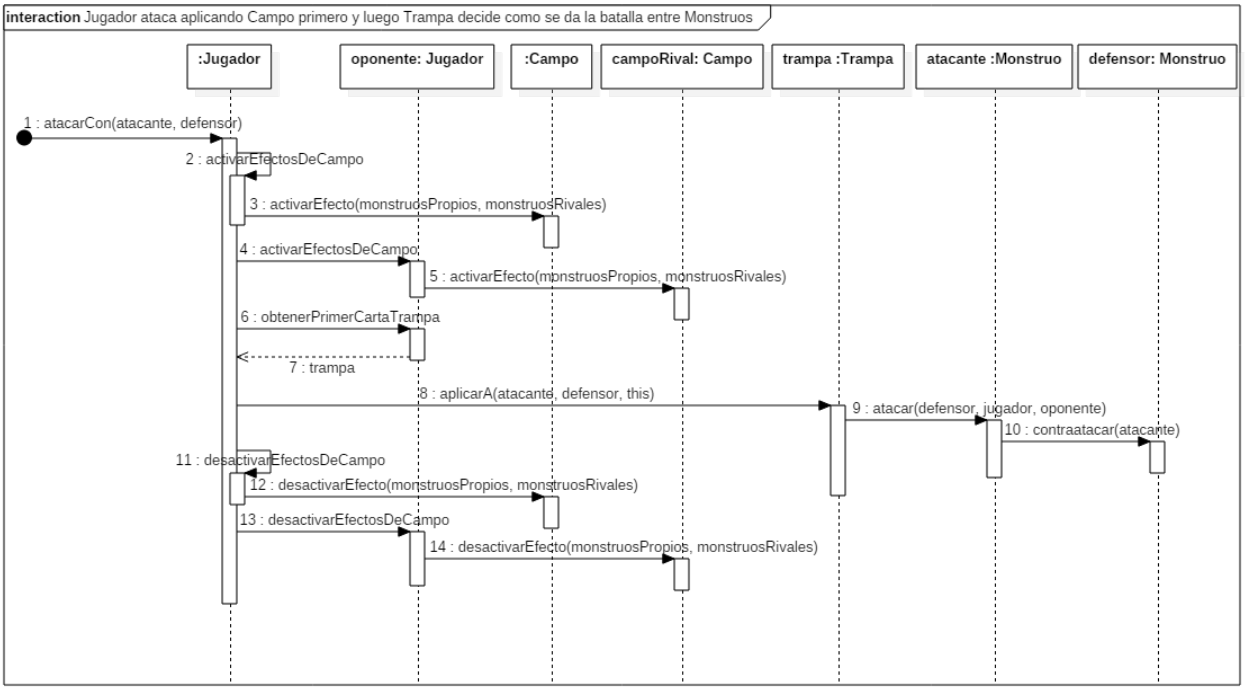


Diagrama de secuencia del ataque entre monstruos.

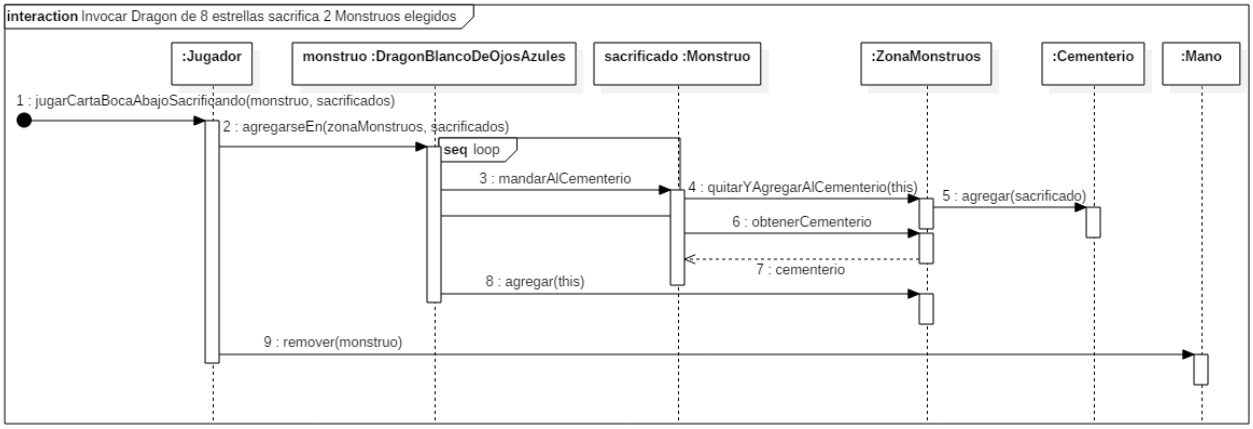


Diagrama de secuencia de la invocación de monstruos que requieren sacrificios.

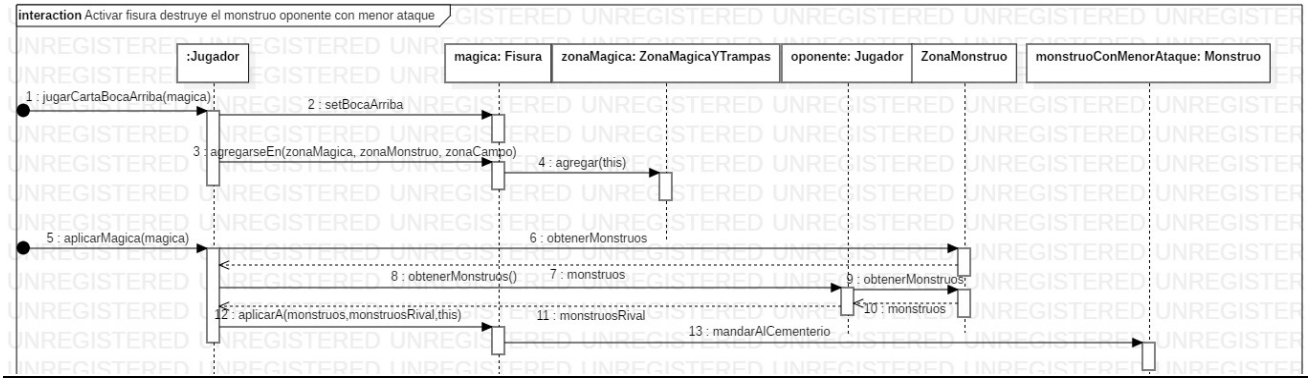


Diagrama de secuencia de la aplicación de un efecto mágico.