Universidad de Buenos Aires

Facultad de Ingeniería

75.07/95.02 - Algoritmo y programación III

Curso 1
Segundo cuatrimestre de 2019
GRUPO: T14 - Corrector: Eugenia

Alejandro, Pablo Martín	98021	pabloale96@gmail.com
Huaman, Yonatan	98784	jonathanhc42@gmail.com
Chávez Cabanillas, José E.	96467	joseduardoc93@gmail.com
Rizzo, Gonzalo Gabriel	96772	gonzalorizzo95@gmail.com

5 de diciembre de 2019

Índice

1. Introducción

El siguiente trabajo practico consiste en realizar un juego interactivo entre 2 jugadores, donde el objetivo de cada uno es derrotar o lograr que el jugador contrario se quede sin unidades o se rinda. Para eso se uso una programación orientada a objeto, usando relaciones entre objetos (herencia, asociación, composición, etc) aprendidas en la clase.

2. Objetivo

El objetivo principal del trabajo es lograr diseñar una aplicación de manera grupal que logre cumplir las siguientes características:

- La aplicación consiste en un videojuego que tiene dos jugadores que disponen de su turno para jugar
- Cada jugador tiene unidades para colocar en un tablero, donde se le asigna una determinada zona del mismo (las unidades de cada jugador se colocan en el mismo tablero).
- El cada jugador dispone de puntos para comprar unidades los cuales le permite colocar una unidad sobre el tablero.
- Cada unidad presenta un ataque diferente, un alcance diferente y también un costo diferente.
- La aplicación debe tener un interfaz gráfica
- El objetivo principal del juego es lograr derrotar al jugador contrario, venciendo todas las unidades.
- El tablero se divide en dos sectores, sector aliado y sector enemigo. Cada jugador reconoce su sector y las unidades que se muevan al sector contrario recibirán un 5 % mas de daño.
- Cada jugador solo puede colocar las unidades sobre su sector del tablero.
- Las unidades tienen sus características, las mismas pueden ser soldado de infantería, jinete, curandero y catapulta.
- El soldado de infantería tiene un ataque de corta distancia (cuerpo a cuerpo), tiene un daño de 10 puntos de vida con una vida total de 100 puntos de salud. Además tiene la habilidad de moverse en grupo si se encuentran 3 o mas soldados juntos. El coste de esta unidad es de 1 punto.
- El jinete tiene un ataque de larga distancia (jinete con arco) y un ataque de corta distancia (caballero). El jinete tiene un daño de 15 si ataca a larga distancia y un daño de 5 si ataca a corta distancia, la unidad tiene 100 puntos de salud y un costo de 3 puntos. El jinete cambia su modo ataque dependiendo si tiene unidades aliadas cerca (unidades del mismo jugador) o no hay unidades enemigas (unidades del jugador contrario) cerca su modo de ataque termina siendo a larga distancia, si se encuentra con unidades enemigas cerca su modo de ataque es corta distancia.
- La unidad de curandero no presenta un daño de ataque en su lugar puede curar a las unidades cercanas (todas salvo por la catapulta) que le pertenezcan al mismo jugador. Presenta 100 puntos de salud y un costo de 2 puntos.
- La ultima unidad es la catapulta que presenta un ataque de larga distancia. El daño de la catapulta es de 20 y con puntos de salud de 50. El daño realizado por la catapulta a una unidad no depende si esa unidad es aliada o enemiga, a su vez si la unidad atacada tiene alrededor de ella, otras unidades el daño se propaga hacia ellas, y se repite sucesivamente(el daño es el mismo para todas las unidades). Su costo es de 5 puntos.

3. Supuestos

En esta sección explicaremos sobre los supuestos que consideramos para la implementación y desarrollo del programa:

- La inicialización del tablero tiene un agregado adicional sobre la asignación de cada casillero a un determinado jugador, lo cual permitiría asignar un campo perteneciente a cada jugador.
- Curandero puede realizar un ataque (no de sanación) a cualquier unidad sea aliada o enemiga, la unidad atacada no presenta daño alguno.
- Las excepciones manejadas para el Tablero se basaron en dos problemas, por un lado el estado en el que se encontraba un casillero (ocupado o no), por otro lado verificar la posición en la que se ingresaba la unidad, comprobando si la posición de ingreso estaba dentro de los límites del tablero, estas excepciones nos permitían controlar el correcto ingreso y posicionamiento de las unidades al tablero.
- Los tipos de ataque presentados las dividimos en tres clases las cuales representan los tipos disponibles para una determinada unidad, donde de las cuales el AtaqueDeSanacion es exclusivo de la Unidad Curandero, mientras las otras dos clases (o tipos) de Ataques son de un uso común en las restantes unidades.
- Las excepciones manejadas para el jugador están basadas primero en los puntos que el jugador dispone para comprar unidades, donde una de las excepciones creadas controla que el jugador tenga puntos disponibles para comprar una Unidad, y la segunda excepción controla que el jugador aun tenga Unidades disponibles para atacar o mover.

4. Modelo de Dominio

Para el presente trabajo, se busca aplicar las metodologías de diseño vistas en el curso. Entre ellas, se aplican XP (*eXtreme Programming*), TDD (*Test Driven Development*) e identificación de patrones.

Para la organización del trabajo, se separaron las distintas clases según sus funcionalidades y características. Se organizan los paquetes:

- *acciones*: Donde se encuentran las clases que facilitan las acciones que realizan las unidades en el juego, como pueden ser la movilidad o los ataques entre unidades.
- *entidades*: Donde se encuentran las clases que implementan los distintos tipos de unidades en el juego.
- *tablerosycasilleros*: Donde se encuentran las clases que determinan el posicionamiento de las entidades.
- *Aplicaciones*: Donde se encuentran las clases para la interfaz gráfica. Dentro de la misma, se encuentra dos carpetas que tienen el flujo del juego (la fases del juego en si) y las vistas (tableros, menú principal,etc).
- *Eventos*: Donde se encuentran las clases para la interacción con el usuario (como botones de la interfaz)

Luego, se encuentran en el paquete general *main/java/fiuba.Algo3.AlgoChess* serian las clases de control que manejan los grandes aspectos el programa en general, como la clase *AlgoChess* la cuál se encargará de crear los jugarodes y el tablero involucrado en el juego.

Para las implementaciones correspondientes, se crean las pruebas unitarias e integrales pertinentes a los aspectos interesantes del programa, y aquellas pruebas que son solicitadas en las entregas.

5. Diagrama de Clase

A continuación se mostraran los diagramas de clases que se utilizo del programa, en el mismo se observa las relaciones entre las clases.

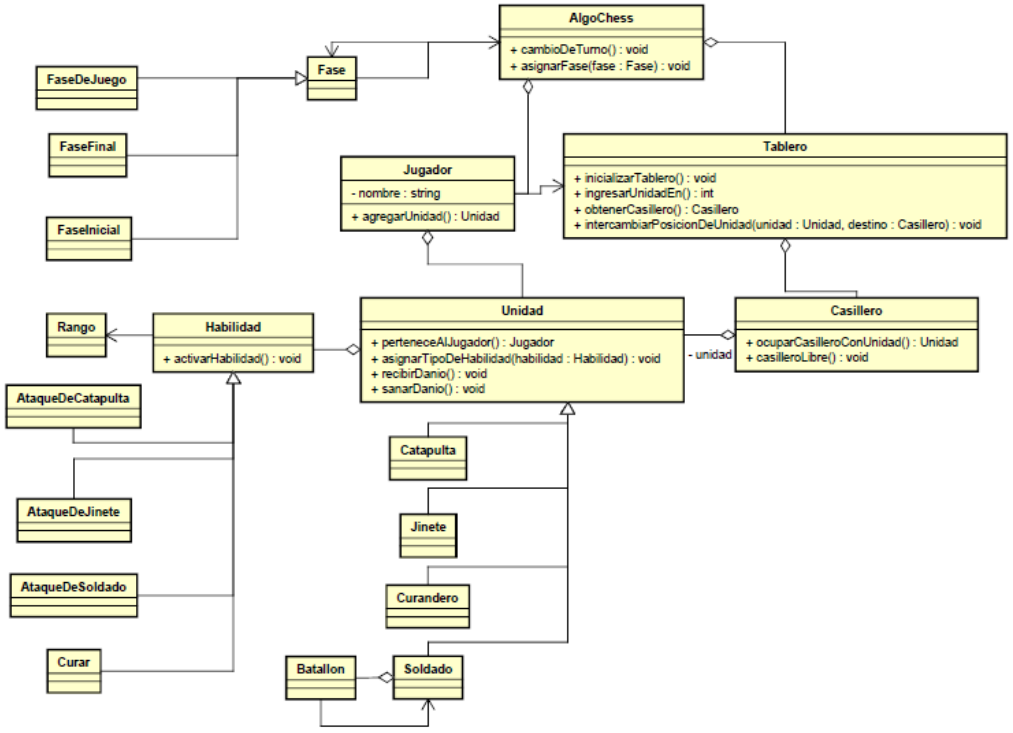

Figura 1: Diagrama de clases

6. Diagramas de Secuencia

En esta sección se mostraran los diagramas realizados, se intento hacer diagramas que abarquen la mayor cantidad de métodos. Se considero vital los métodos de movimiento, ataque y el cambio de fase o estado del juego. A partir de estas consideraciones, se realizaron diversos diagramas de secuencias para dejar prolijo el funcionamiento de cada método.

6.1. Diagramas de Secuencia- Ataque

En esta sección se muestra los diagramas de ataque, se decidió tomar el ataque del jinete que presenta un cambio dependiendo las unidades cercanas por la dificultad del mismo.

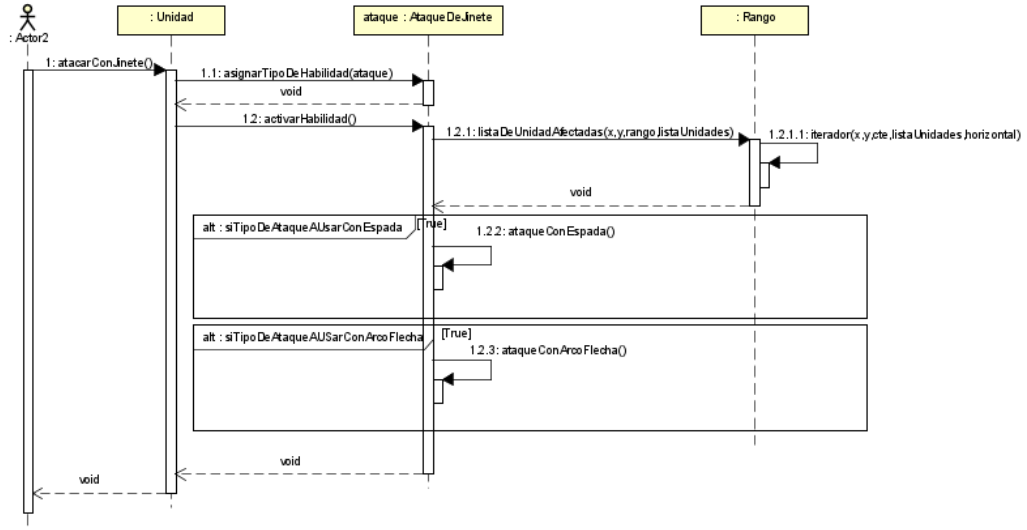

Figura 2: Diagrama de secuencia del ataque del jinete.

6.2. Diagramas de Secuencia- Movilidad

Se observa que la movilidad depende del tipo de unidades que se quiere mover, para eso, se muestra don casos dependiendo si la unidad es un soldado, ya que el mismo puede moverse como un batallón.

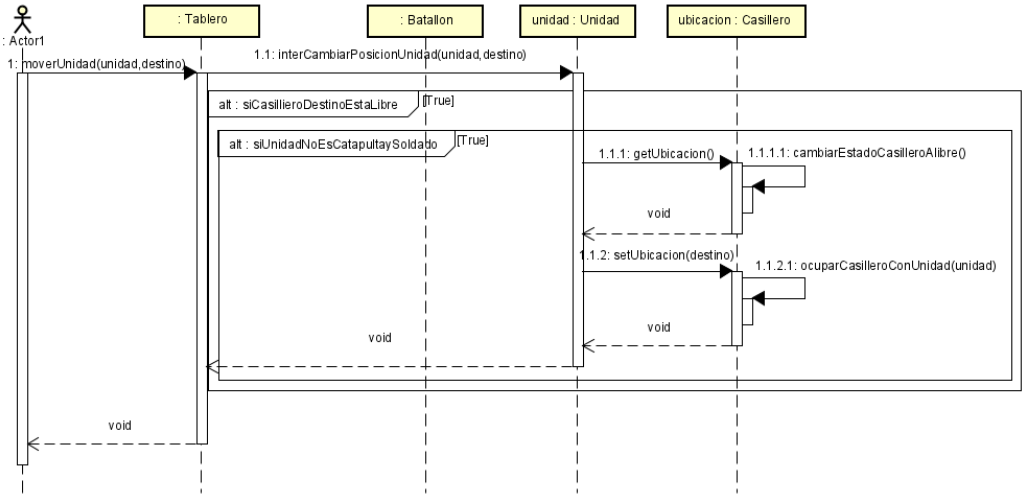

Figura 3: Diagrama de secuencia del movimiento sin soldado.

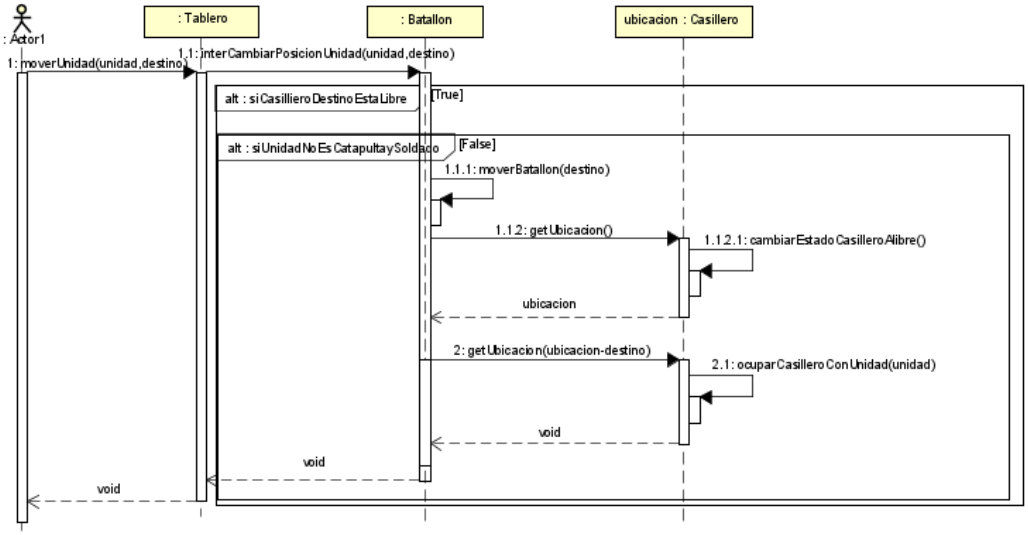

Figura 4: Diagrama de secuencia del movimiento con soldado.

6.3. Diagramas de Secuencia- Fases

En este caso, se mostraran distintos diagramas de secuencia para simplificar el diagrama de secuencia de activar una fase o el cambio de una fase. En la fase final el métodos de *activarFase()* muestra al ganador de la partida, y como no hay otra fase, el método *siguienteFase()* no realiza nada.

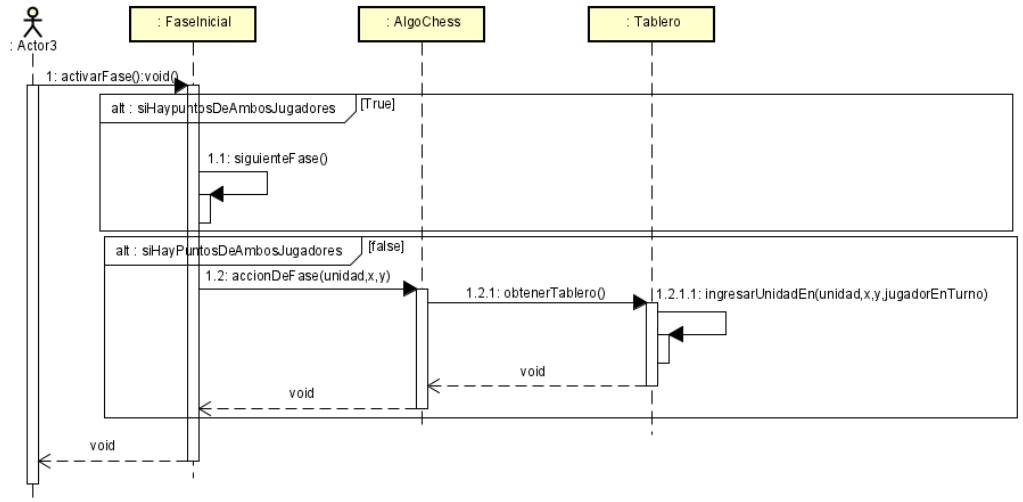

Figura 5: Diagrama de secuencia activar fase inicial.

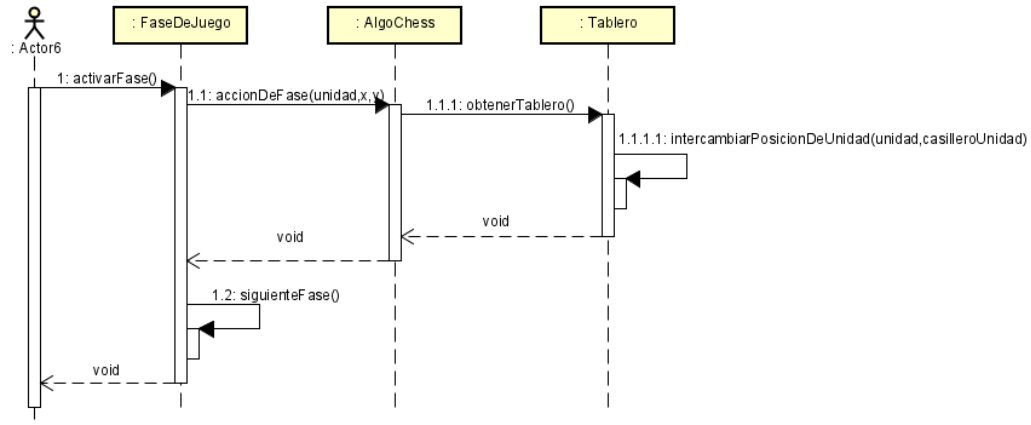

Figura 6: Diagrama de secuencia activar fase de juego.

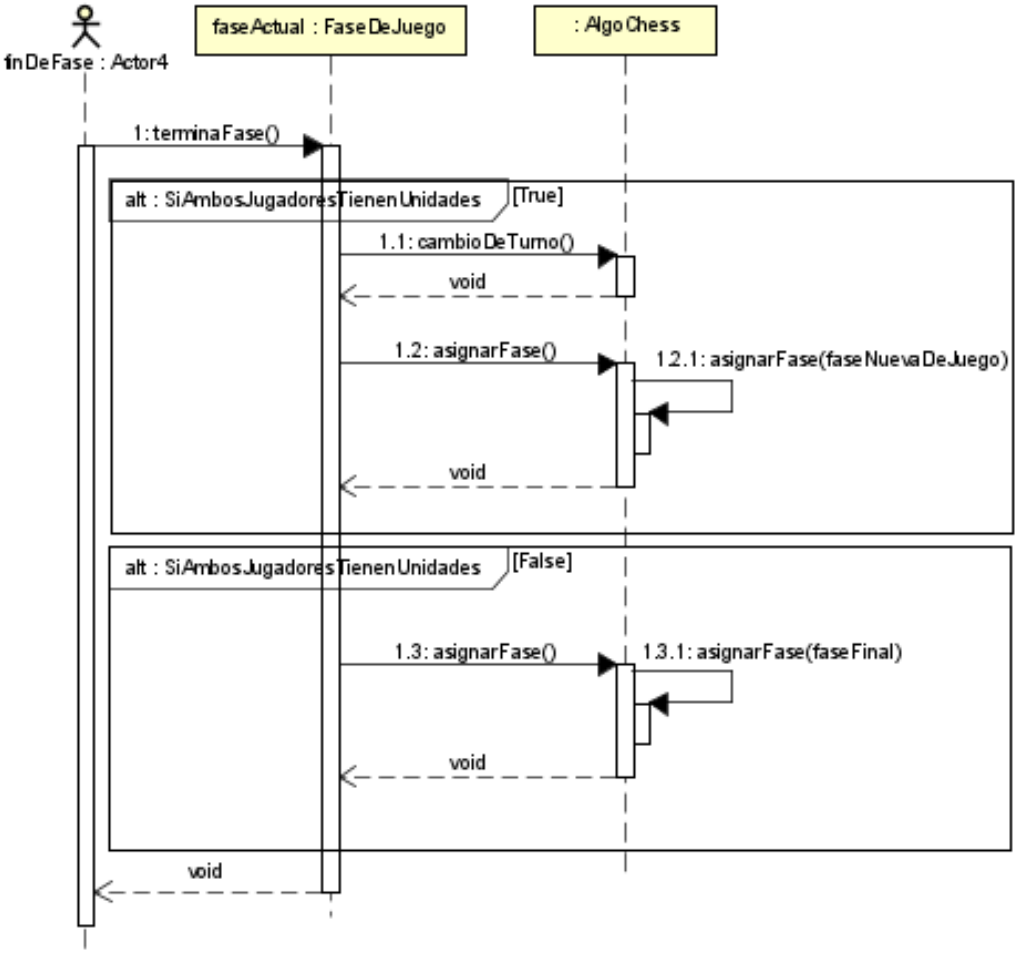

Figura 7: Diagrama de secuencia siguiente fase desde fase inicial.

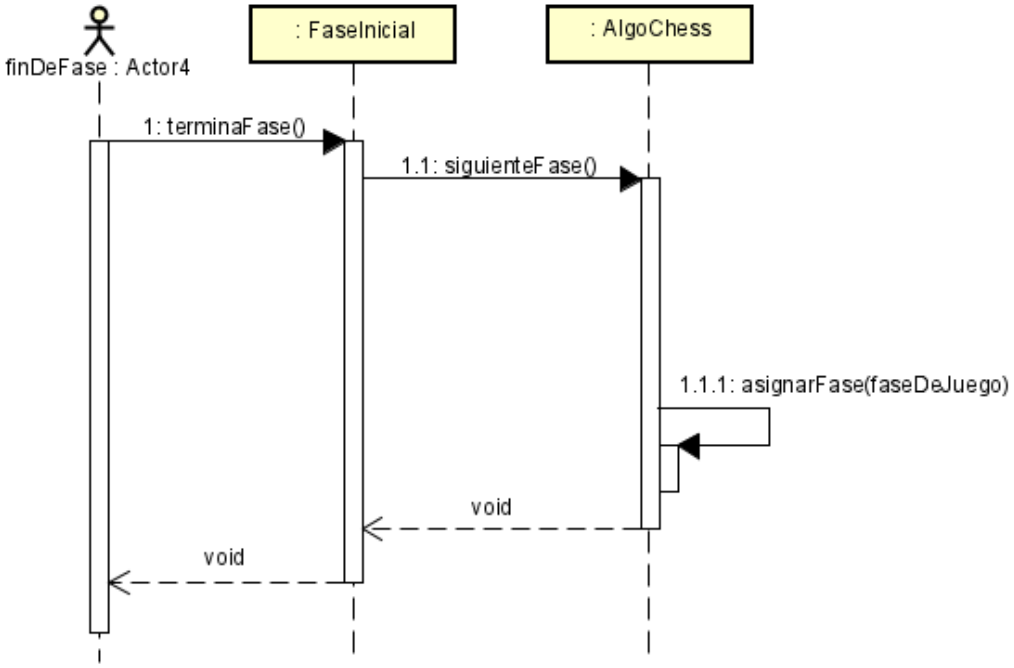

Figura 8: Diagrama de secuencia siguiente fase desde fase de juego.

A partir de ahora, los diagramas de secuencias de los siguientes métodos se realizaron para la mejora de la visualización de los métodos de anteriores.

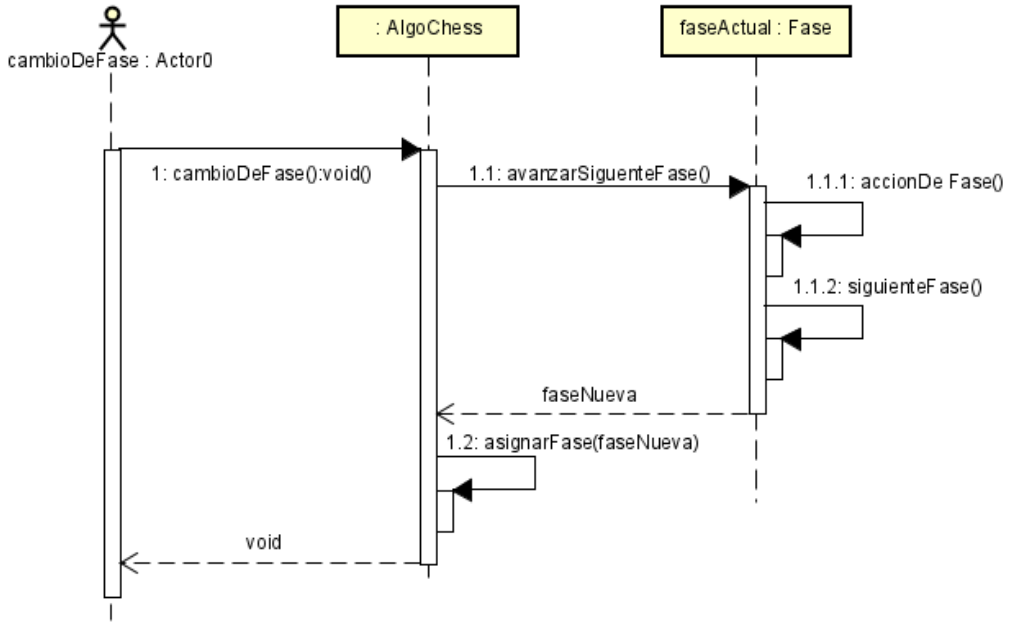

Figura 9: Diagrama de secuencia de cambio de fase.

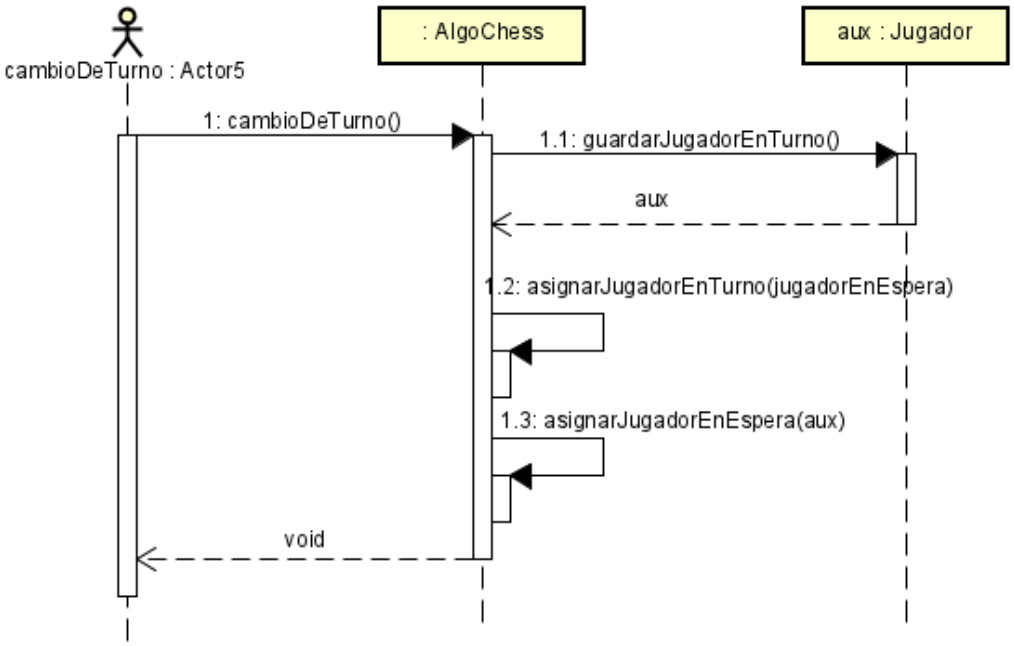

Figura 10: Diagrama de secuencia de cambio de turno.

7. Diagrama de Paquetes

Para mostrar la distribución de los paquetes usados para la programación de java. En el mismo, se muestra el paquete *src* que presenta el código de todas las clases y pruebas.

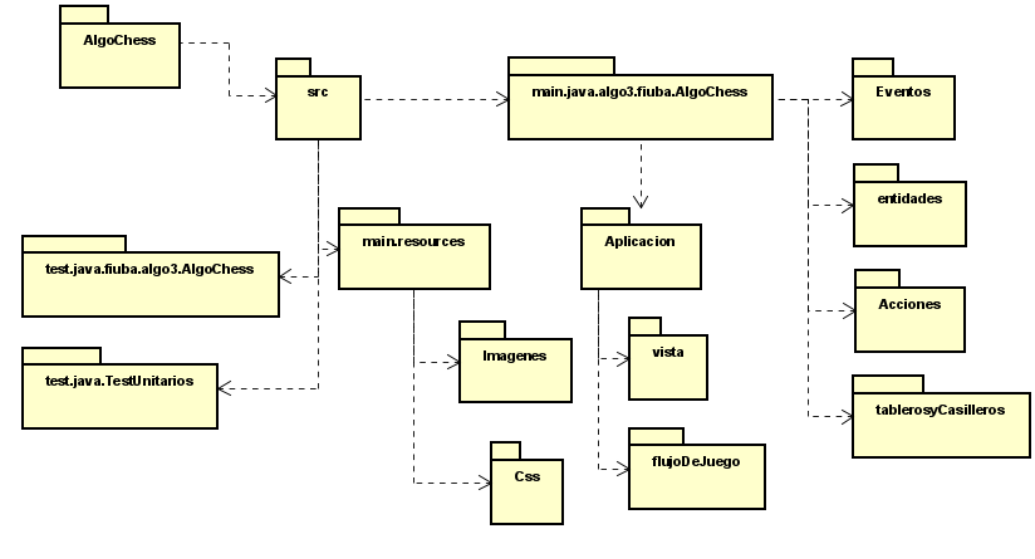

Figura 11: Diagrama de paquetes

8. Diagrama de Estados

El juego presenta 3 estados bien característicos. La fase inicial que se colocan las unidades sobre el tablero y termina dicha fase cuando ambos jugadores no tienen mas puntos, la fase de juego en donde se realiza el ataque y el movimiento de los unidades, el juego se quedara mayor parte del tiempo en esta fase, la única forma

de cambiar de fase se logra cuando unos de los jugadores se queda sin unidad (esta fase también se realiza la eliminación de unidades en el tablero) o cuando se reinicia el juego. Después de la fase de juego, viene la fase final que es la encargada de mostrar al ganador en pantalla.

Lo mencionado anteriormente se muestra en la figura (**??**), en donde se ve claramente que el programa pasa la mayor parte del tiempo en la fase de juego. El programa comienza en la fase inicial, esto es representado con el punto negro y termina en la fase final con la representación del punto negro con un circunferencia.

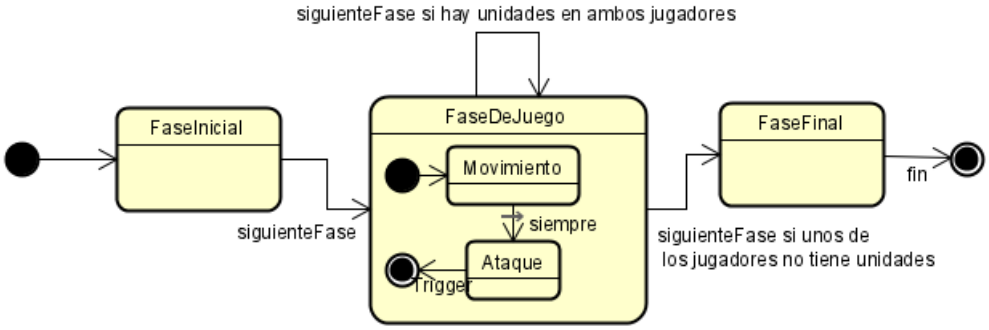

Figura 12: Diagrama de estado del programa.

9. Detalles de la implementación

El primer detalle de implementación es considerar que el tablero es un conjunto de casilleros. Cada casillero conoce a que jugador le pertenece, para la inicialización del tablero se determino mitad de tablero para un jugador y mitad del mismo para el jugador contrario, la ventaja de hacer que el casillero conozca a quien le pertenece son principalmente 2, si esa unidad es colocada por el jugador enemigo se logra mandar una excepción rápida sobre la clase casillero y no es necesario que tablero se entere de la misma. Segundo facilita la agregación de daño por estar en el sector enemigo.

Para la implementación de las unidades, se creo una clase general con el nombre unidad. Sobre ella se hace una herencia para cada tipo de unidad (soldado de infantería, jinete, curandero,etc), en la clase madre se obtiene los métodos principales como *recibirDanio* o *realizarAtaque*, estos métodos se realizan sobre todas las entidades por eso se realizo sobre la clase madre. El programa también controla que el ataque realizado es sobre unidades de diferentes jugadores (salvo por la catapulta que puede realizar daño sin importar el jugador).

10. Interfaz Gráfica

La interfaz gráfica se realizo con la librería *JavaFX*, para eso se considero un menú principal, créditos(menú final para la finalización del juego) y la pantalla de juego, que contiene un tablero con sus respectivos casilleros. A su vez, se coloco diferentes imágenes para indicar cada tipo de unidad insertada (soldado, jinete, catapulta y curandero).

En el menú principal, contiene tres botones. El primero activa la opción de comienzo del juego, el segundo sirve para la salida del juego y por ultimo se encuentra el botón de créditos, que muestra las nombres de los creadores del programa. La pantalla de juego tiene la representación gráfica de tableros, casilleros y unidades, se logra como un conjunto de imágenes que se inserta dentro de la imagen de tablero dividido por casillero, es decir, la imagen de las unidades se inserta en un casillero sobre una imagen del tablero (esta dividida por casilleros).