FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico - Entrega final

[7507/9502] Algoritmos y Programación III
Curso 1 - Grupo 3
Segundo cuatrimestre de 2019

Alumno:	BENITEZ, Juan Severiano
Número de padrón:	96069
Email:	juanwbenitez@gmail.com

Alumno:	MASSONE, Mario Bernardo
Número de padrón:	102141
Email:	bernardomassone@hotmail.com

Alumno:	RODRIGUEZ DALA, Tomás
Número de padrón:	102361
Email:	chubyrd1@gmail.com

Alumno:	ZABALA, Andoni Felix
Número de padrón:	101005
Email:	andonifzabala@gmail.com

Índice

1. Introducción

El presente informe reúne la documentación de la solución de la primera entrega del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación de un vídeo juego en Java utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

-En el movimiento de un batallón de más de tres infanterías se moverán aquellos de prioridad en sentido horario.

Ejemplo: Al tener 4 soldados en las posiciones: [(0.0) , (0.1) , (1.0) , (1.1)] y se quiere mover el soldado en la posición (0.1) entonces el batallón estará formado por los soldados en las posiciones : [(0.0) , (0.1) , (1.0)].

-Si una unidad ataca a un enemigo fuera de su rango le ocasiona 0 de daño.

-Si un curandero cura a una unidad aliada en una distancia mediana o distancia lejana,esta unidad aumentaría en 0 su vida.

-No se pueden pasar turnos,en cada turno necesariamente se hace una acción.

3. Diagramas de clase

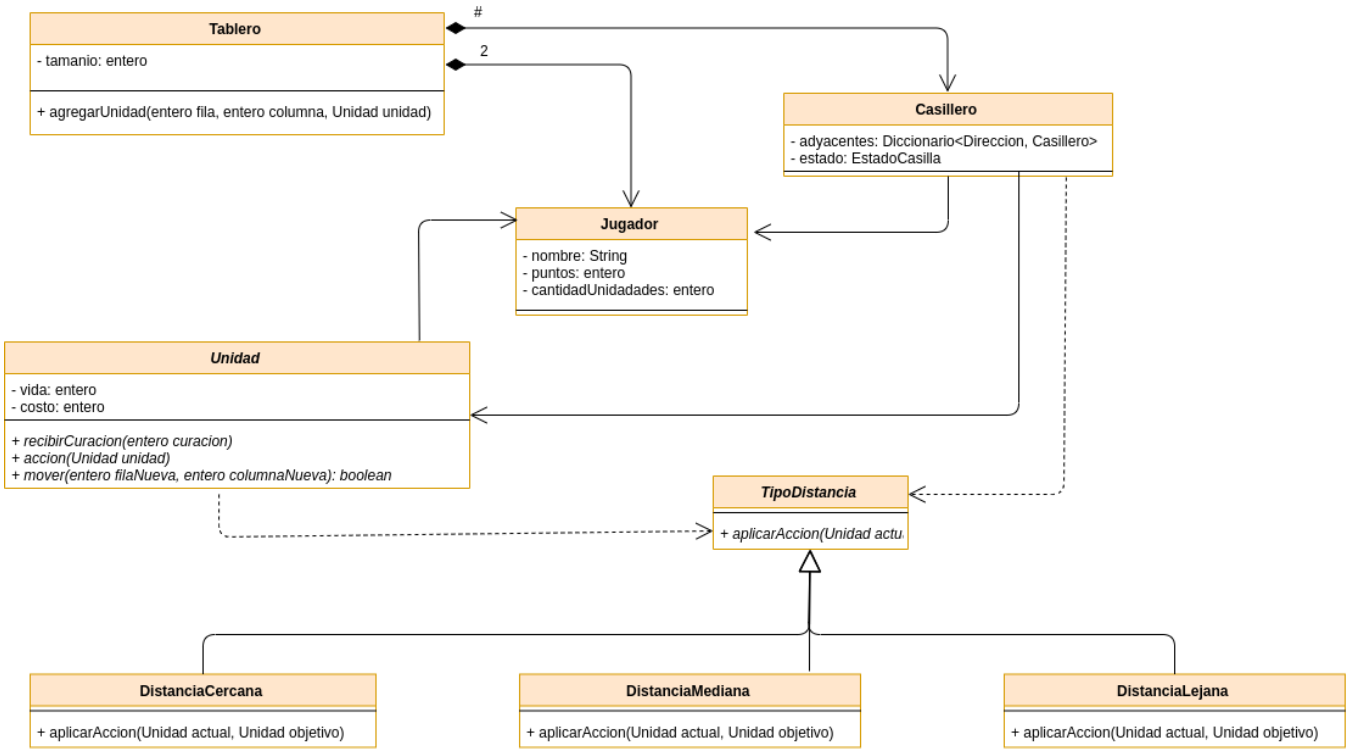

Figura 1: Diagrama de clase general.

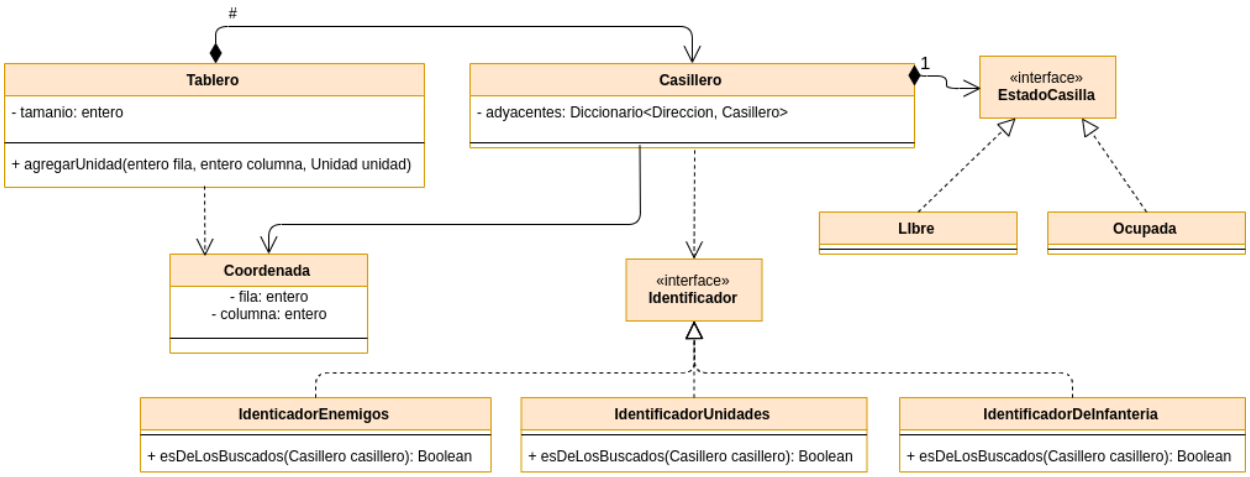

Figura 2: Diagrama de clase Casillero.

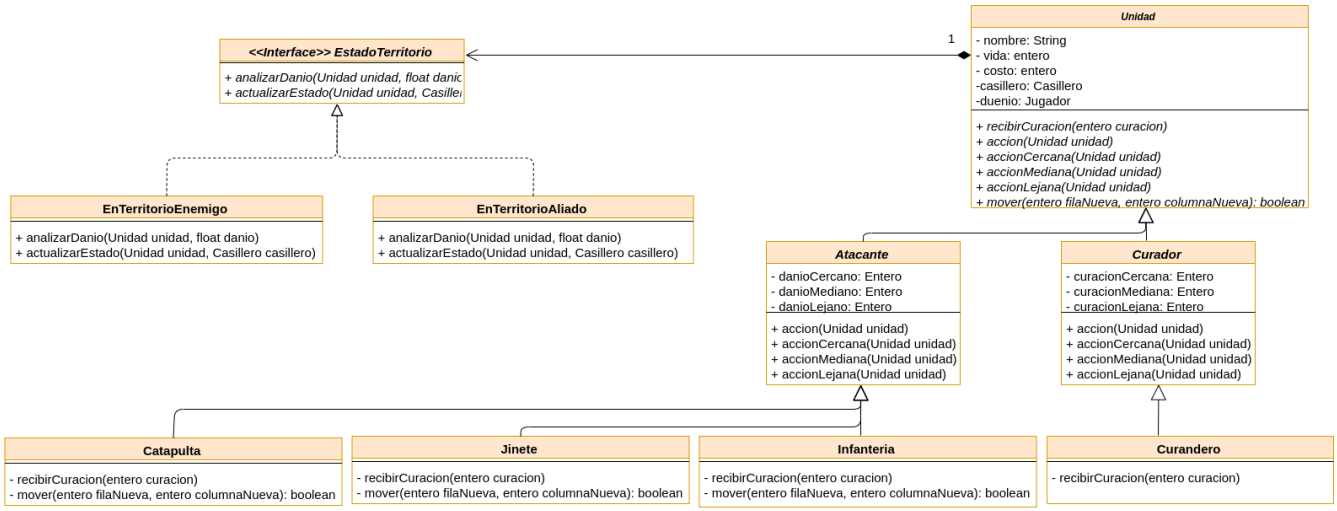

Figura 3: Diagrama de clase Unidad.

4. Diagramas de secuencia

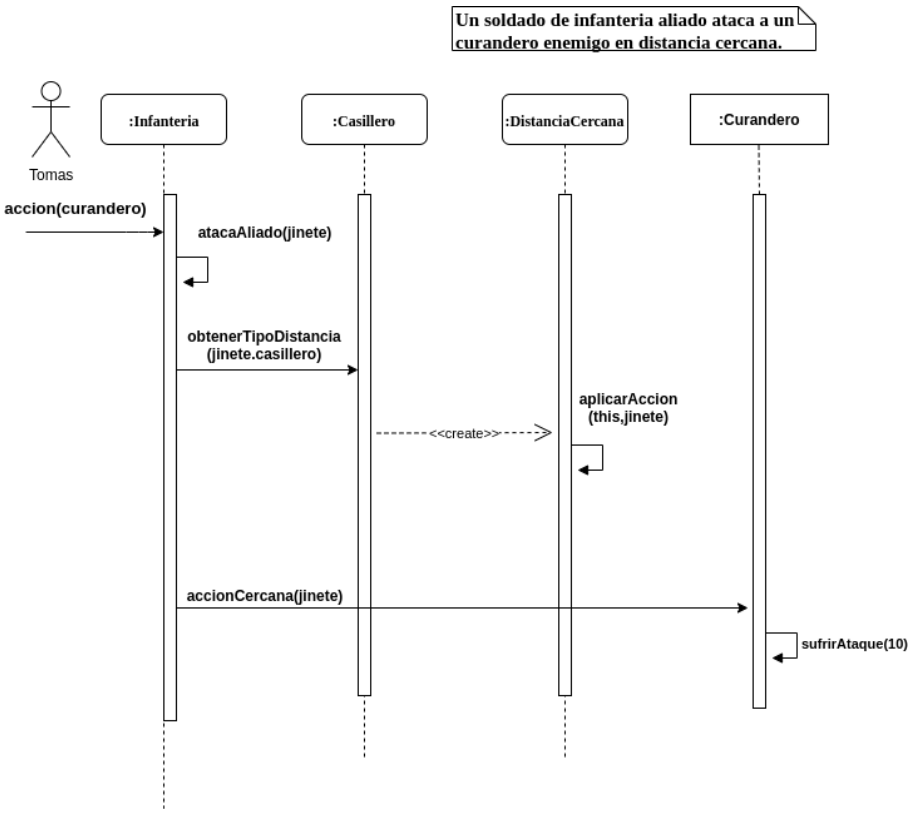

Figura 4: Diagrama de secuencia de un soldado atacando a enemigo en distanca cercana.

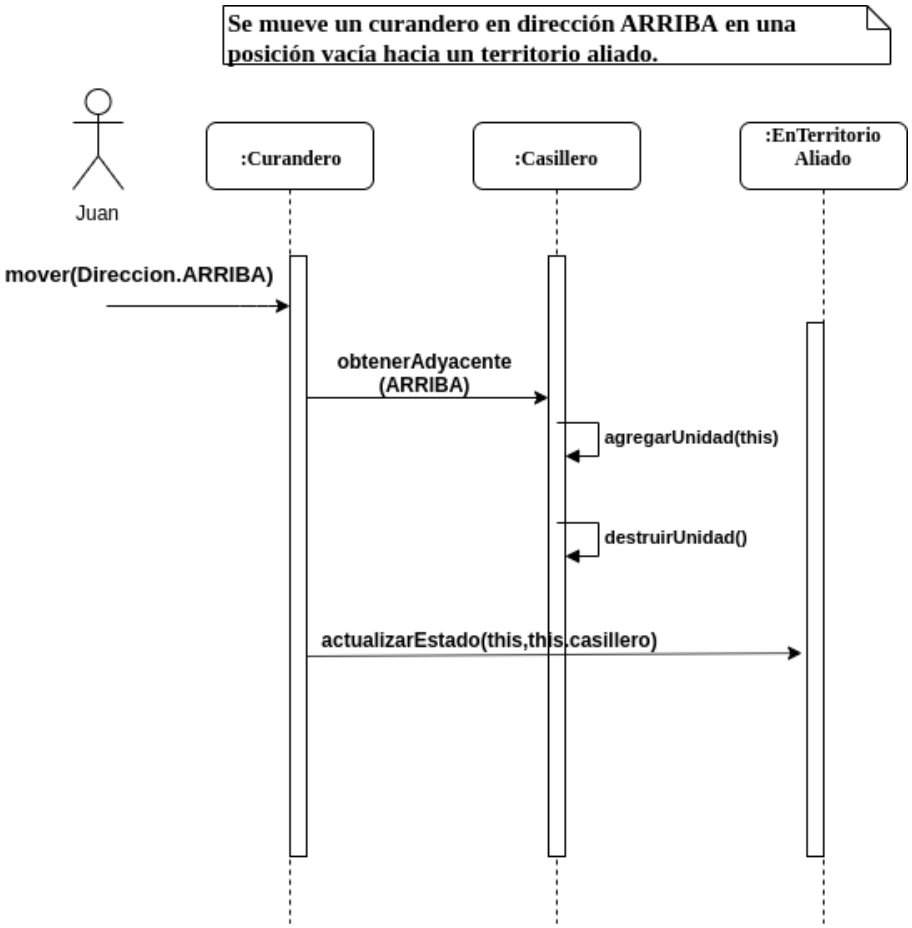

Figura 5: Movimiento de un curandero a una posición vacía.

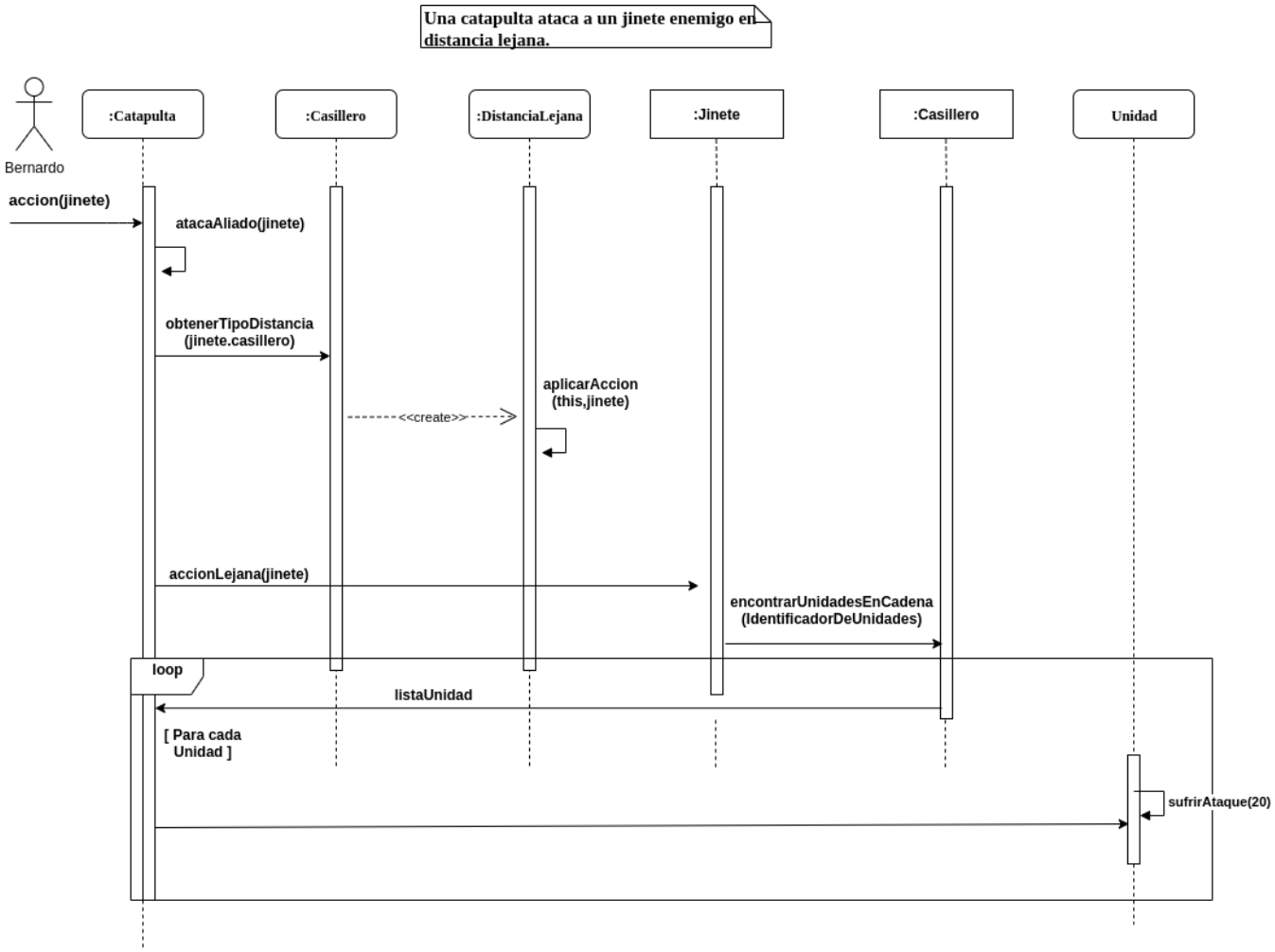

Figura 6: Una catapulta ataca en distancia lejana y por ende ataca a todos los adyacentes (y a los adyacentes de estos).

5. Diagramas de paquetes

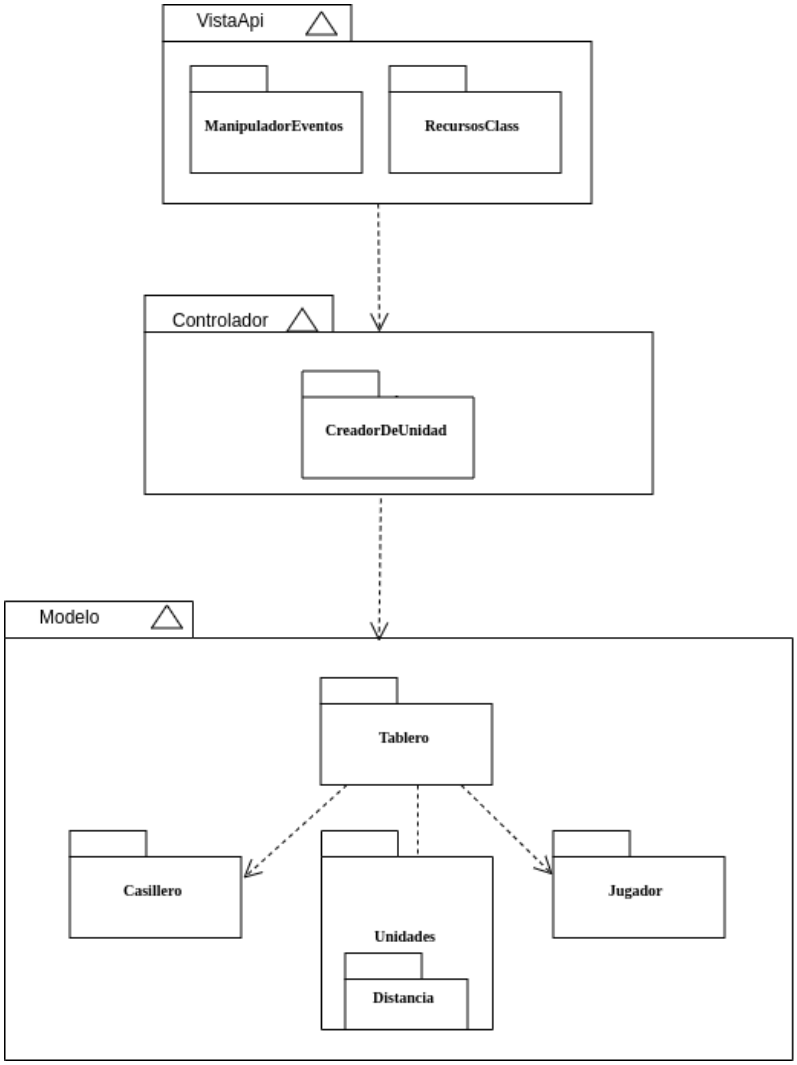

Figura 7: Diagrama de paquetes del proyecto.

6. Diagramas de estados

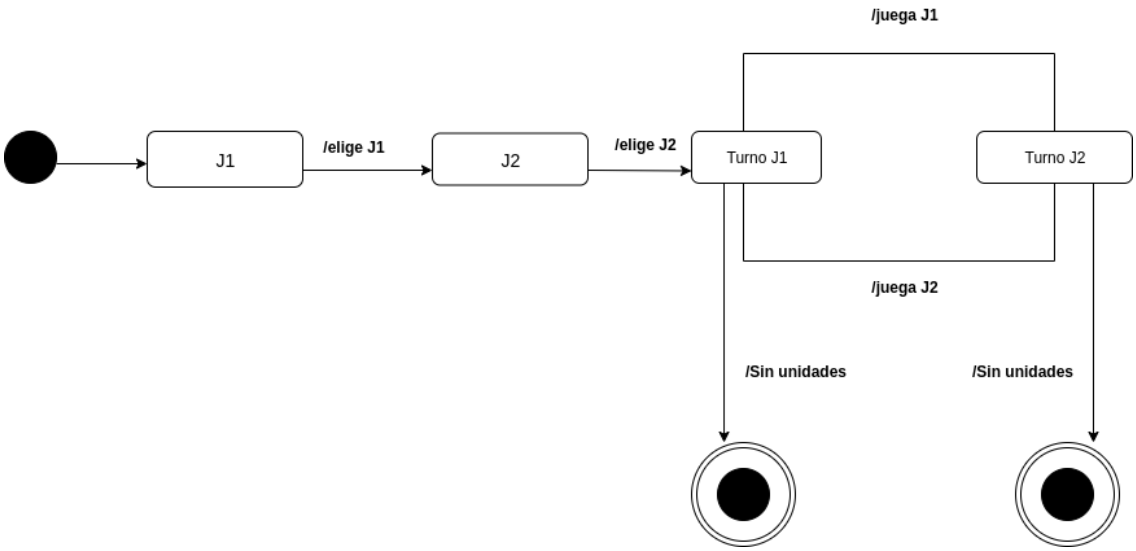

Figura 8: Diagrama de estado de los turnos de jugadores.

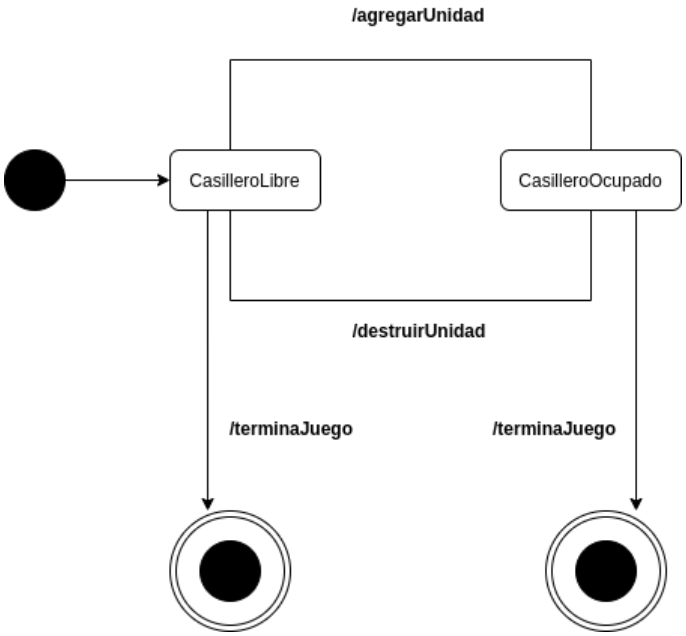

Figura 9: Diagrama de estado de los casilleros.

7. Detalles de implementación

Unidad:

Definimos las unidades con tres tipos de acciones de acuerdo a las distancias: accionCercana,accionMediana y accionLejana (se pueden apreciar en la clase abstracta TipoDistancia) lo que,si

bien, por un lado tuvimos que definir comportamiento sin resultado final (por ejemplo: un jinete ataca con un daño igual a 0 en distancia lejana) por el otro logramos una mayor generalización de comportamiento y entonces aplicamos polimorfismo con el uso de las clases DistanciaCercana,DistanciaMediana y DistanciaLejana.También permitió encapsular las constantes que caracterizan a los distintos tipos de distancia en Casillero con el metodo obtenerTipoDistancia(casillero) y no en las unidades dado que esto no es una responsabilidad de la unidad.

Además,permite que el programa sea mas flexible dado que si en un futuro se desea que una Unidad pueda realizar una acción a una distancia en la que antes no podía (en nuestro modelo seria un daño o curación con valor nulo) bastaría con cambiar el valor del atributo correspondiente a esa distancia.

Casillero:

Nos encontramos con el problema de que el Casillero cambiaba su comportamiento a medida que el juego avanzaba.En consecuencia, decidimos aplicar el patrón de diseño Strategy para cambiar su estado en tiempo de ejecución a medida que las unidades ocupaban o liberaban casilleros.

MVC:

Todas las acciones principales del juego son manejadas por dos controladores diferentes: ControladorDeAgregarUnidad y ControladorDeMovimiento los cuales se comunican con el modelo y la vista todo el tiempo.

De esta forma se busco separar la lógica y el estado entre el modelo y la vista haciendo que estas se conozcan lo menor posible con el beneficio de que si en un futuro cambia el modelo no se deberían hacer cambios directos en la vista.

8. Excepciones

CasilleroFueraDeRangoExcepcion: Excepción creada cuando se intenta mover una unidad fuera del rango del campo de juego.
Esta es atrapada por el controladorDeMovimiento y la resuelve sin mover la unidad a dicha posición e informándole a la vista que esa posición es invalida.

CasilleroNoEsAdyacenteExcepcion: Excepción creada cuando se intenta mover una Unidad a un casillero el cual no es adyacente.
Esta es atrapada por el controladorDeMovimiento y la resuelve sin mover la Unidad e informándole a la vista que el casillero no es adyacente.

CasilleroOcupadoExcepcion: Excepción creada cuando se intenta mover una Unidad a un casillero el cual esta ocupado.
Esta es atrapada por el controladorDeMovimiento y la resuelve sin mover la Unidad e informándole a la vista que el casillero esta ocupado.

AtacaAliadoExcepcion: Excepción creada cuando se intenta atacar a una unidad aliada. Esta es atrapada por el controladorDeMovimiento y la resuelve sin atacar e informándole a la vista que no puede atacar a un aliado.

CatapultaCuracionExcepcion: Excepción creada cuando se intenta curar a una catapulta. Esta es atrapada por el controladorDeMovimiento y la resuelve sin curar e informándole a la vista que las catapultas no pueden ser curadas.

CatapultaNoSePuedeMoverExcepcion: Excepción creada cuando se intenta mover a una catapulta. Esta es atrapada por el controladorDeMovimiento y la resuelve sin mover la catapulta e informándole a la vista que no puede mover catapultas.

CurarEnemigoExcepcion: Excepción creada cuando se intenta curar a un enemigo. Esta es atrapada por el controladorDeMovimiento y la resuelve sin curar e informándole a la vista que no puede curar enemigos.

UnidadAgregadaEnSectorEnemigoExcepcion: Excepción creada cuando se intenta agregar una unidad en un campo enemigo. Esta es atrapada por el controladorAgregarUnidad y la resuelve sin agregar la unidad en ese casillero e informando a la vista que no puede agregar la unidad en ese lugar.