FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Java

[7507/9502] Algoritmos y Programación III
Curso 2
Segundo cuatrimestre de 2018

Alumno:	Gonzalez,Mauricio
Número de padrón:	100948
Email:	gonza.mauricioivan@gmail.com
Alumno:	Rodriguez,Jose
Número de padrón:	89786
Email:	fiubajose@fi.uba.ar

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 un juego similar al Age of Empires basado en un sistema de turnos en Java, utilizando los conceptos del paradigma de la orientación a objetos vistos en el curso.

2. Supuestos

Consideramos los siguientes supuestos para la realizacion del trabajo:

- Las piezas del juego que no estén en el mapa y no tengan jugador no podrán moverse.
- Intentar mover una unidad, a un lugar ocupado o fuera del mapa, no es posible hacerlo.
- El castillo ataca en sus turnos propios.
- El castillo si se está reparando ataca igual, pero no crea unidades.
- Edificio construyendo no puede crear unidades.
- Edificio reparandose no puede crear unidades.
- Edificios pueden crear en un mismo turno las unidades que quieran (salvo oro y población).
- Los edificios se construyen donde el usuario elige y sin limites.
- Las unidades se crean en la posición del mapa que elija el usuario.
- El aldeano puede reparar o construir a la distancia estando en cualquier lugar del mapa.
- Las unidades que ataquen, no podrán realizar ninguna accion hasta el proximo turno.
- Las unidades que se muevan, no podran realizar ninguna accion hasta el proximo turno.
- El aldeano si se esta moviendo, suma oro igual, pero no puede hacer mas nada hasta el proximo turno.
- Las piezas del juego que no esten en el mapa ni sean propias de un jugador no podran atacar.
- Una pieza de un jugador no puede atacar a otra pieza del mismo jugador.

3. Modelo de dominio

Se creao la clase mapa, que servira como contenedora de las unidades y edificios que los jugadores quieran crear en el mapa, y tiene las responsabildiades de colocar, remover y mover objetos ubicables en el, una vez que le indican como.
Para las unidades y edificios optamos por un diseño en el cual ambos implementan la interfaz Ubicable, es decir pueden ser ubicados en un mapa, y poseen una posicion especifica en el mapa en el momento que son colocados.
Esta posicion es elejida por el usuario siempre y cuando cumpla que es una posicion valida(dentro de los limites del mapa, no ocupada por otra unidad, no superpuesta con alguna otra unidad o edificio).
Las unidades, heredan de la clase UnidadMovil, ya que todas pueden moverse, y poseen entre otros el metodo mover, que sirve para ser desplazados desde una posicion del mapa a otra, cuya distancia es unitaria. Para poder realizar este movimiento se necesita que el usuario que quiere mover a la unidad, elija una posible direccion de movimiento valida(la posicion siguiente en esta direccion tiene que cumplir las caracteristicas nombradas anteriormente).

Por otro lado, los edificios y las unidades al pedirles que realicen una accion, se les debe enviar su jugador para que sea posible preguntarle al jugador si es su piezay asi la unidad que quiere ser movida, pertenece al jugador que quiere iniciar el movimiento y si el edificio que quiere ser utilizado para crear una unidad o para alguna otra accion puede ser utilizado por el jugador de turno.
Hay dos clases de edificios, EdificiosConstruibles y el Castillo que no es construible.
Tienen un estado dependiendo si esta en fase de reparacion, construccion o estado normal, para el caso del castillo no tiene estado de construccion.
Las unidad aldeano tienen un estado dependiendo si esta en fase reparacion, construccion o reparacion.
La clase turno, se encarga de informar quien es el jugador que debe comenzar su turno.
la clase juego posee una clase listaDePiezas que se encarga de almacenar las piezas para poder ser utilizadas por el controlador.
El juego es observador del castillo, de esta manera se entera cuando un castillo es destruido e finaliza el juego.
La clase jugador y juego son observables por la vista, asi ante una pieza que es agregada o eliminada y un juego finalizado, la vista se actualiza.

4. Diagramas de clase

Presentamos los diagramas de clases realizados para la resolucion del trabajo.

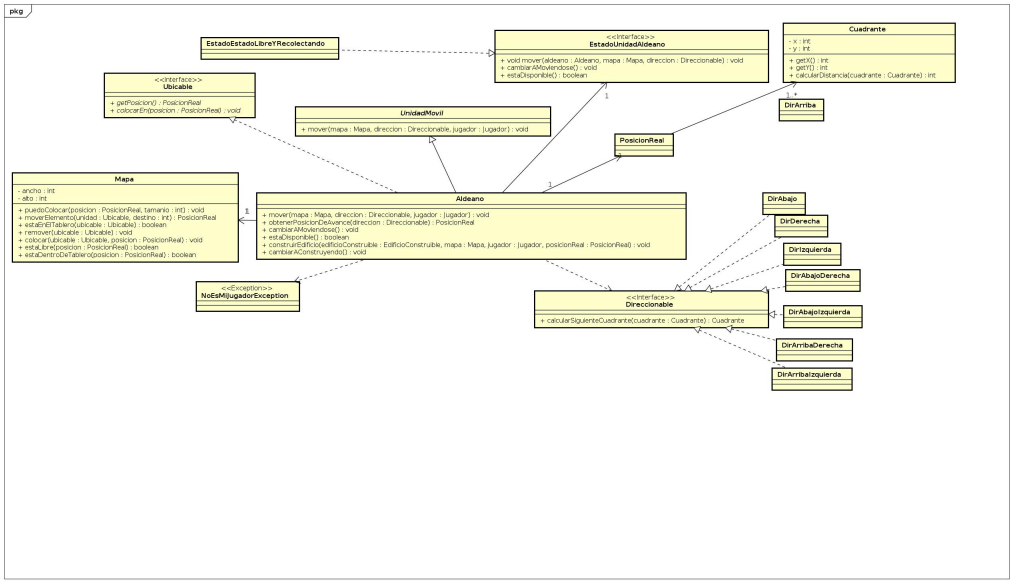

Figura 1: Diagrama de clases para la resolucion del movimiento.

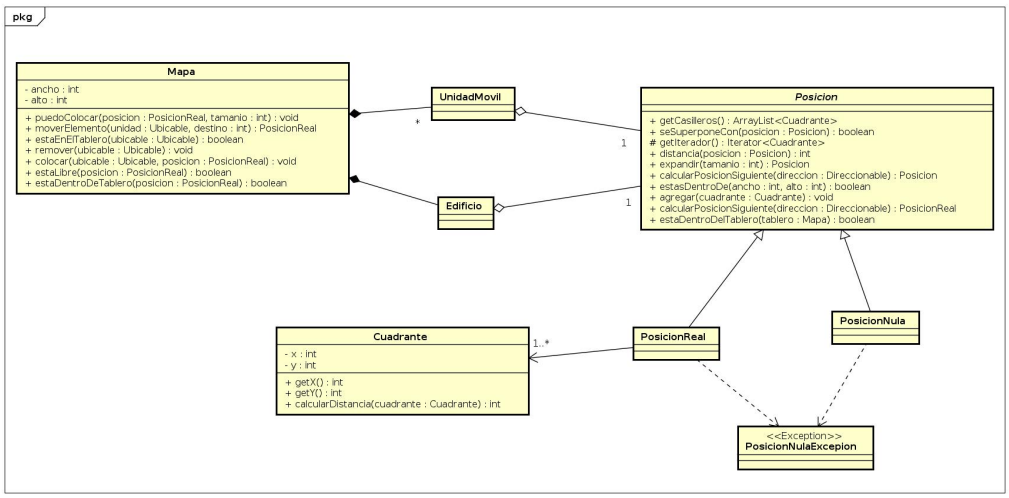

Figura 2: Diagrama de clases del Mapa y las Posiciones.

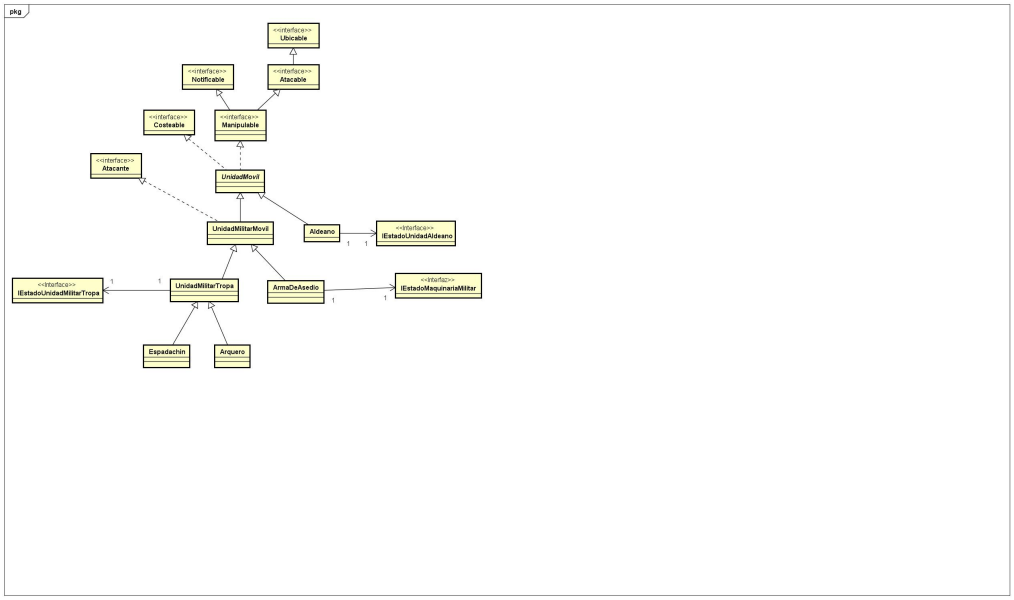

Figura 3: Diagrama de clases las unidades Moviles.

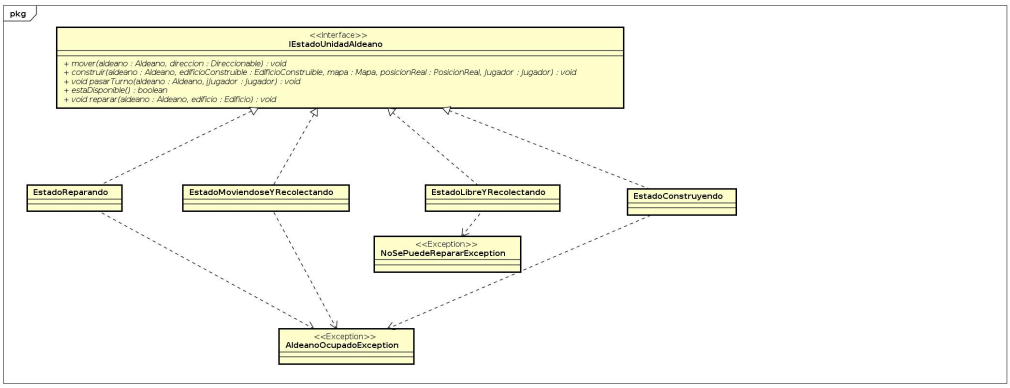

Figura 4: Diagrama de clases de los estados del Aldeano.

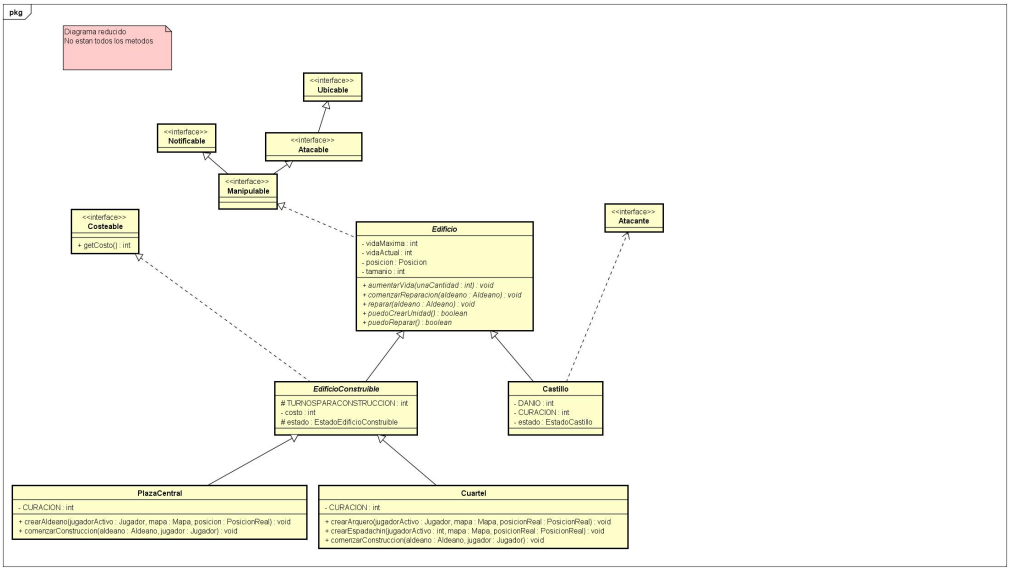

Figura 5: Diagrama de clases de los Edificios.

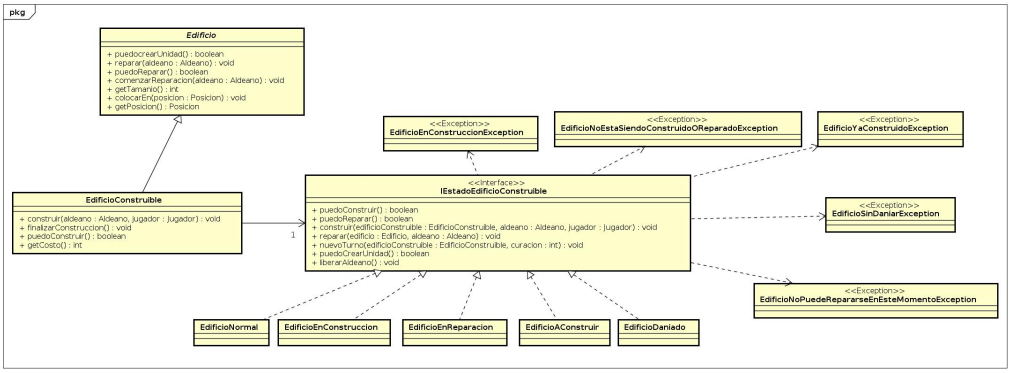

Figura 6: Diagrama de clases de los estados de los Edificios.

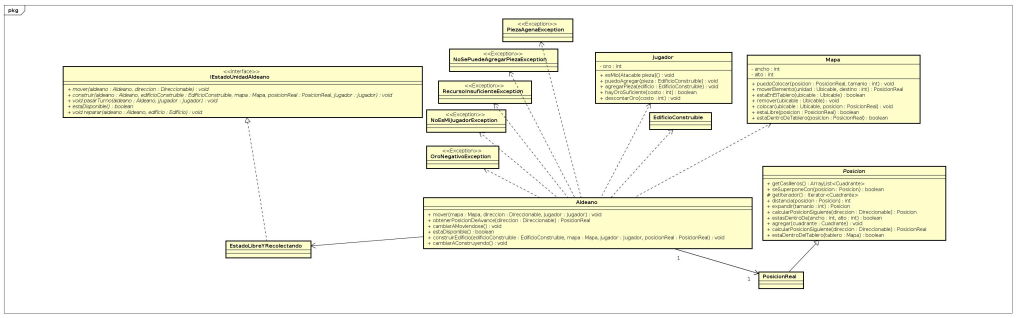

Figura 7: Diagrama de clases de la construccion de los edificios.

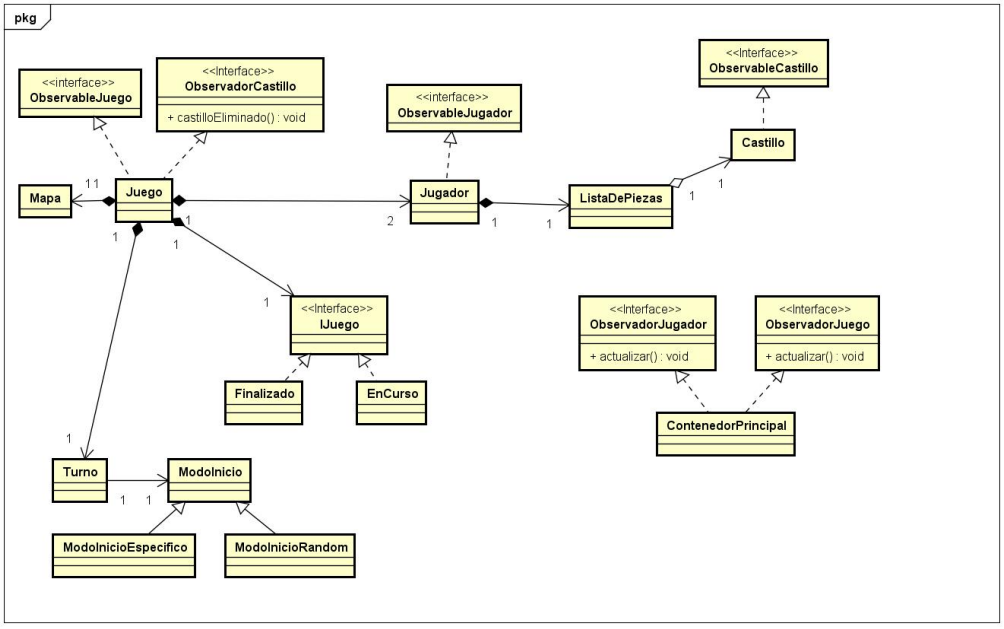

Figura 8: Diagrama de clases del juego.

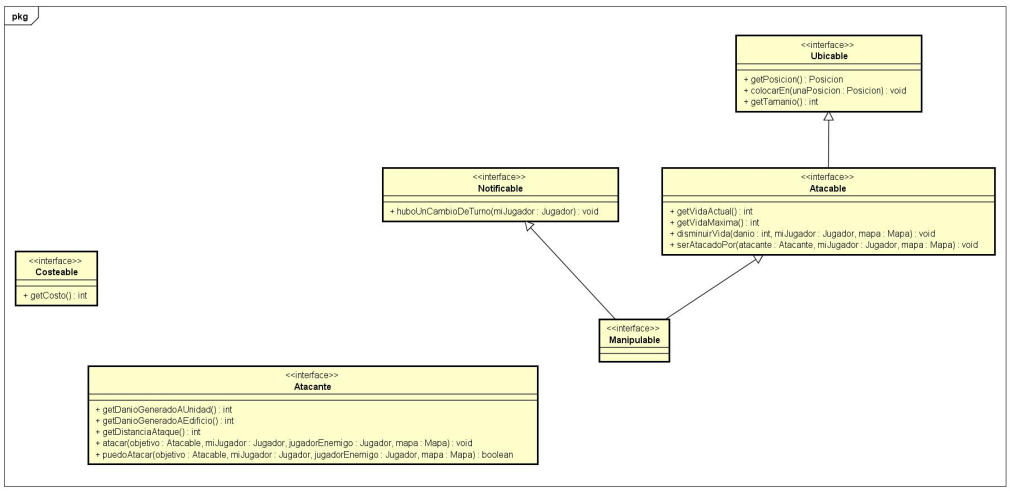

Figura 9: Diagrama de clases de las interfaces.

5. Detalles de implementación

5.1. Mapa

El mapa posee una coleccion de IUbicables que son piezas que implementan la interfaz.
Se les pide la posicion y de esta manera las posiciones se comparan para saber si estan superpuestas o no, para luego poder agregarse al mapa.

5.2. Posicion/Real Nula

Existen 2 posiciones. Real y Nula. Las unidades y edificios se crean con posición nula y al colocarse en el mapa tienen una posicion real (patron NullObjet).

5.3. Edificio

Se creo una clase Edificio, de la cual heredan Castillo y EdificioConstruible.
De Edificio Construible hereda PlazaCentral y Cuartel. La diferencia esta en que Castillo no es construible. y ademas Castillo implementa la interfaz Atacante.
Para el ataque del castillo como de las unidades se uso un patron Double Dispatch.

5.4. Unidades

Para las unidades se creo una clase UnidadMovil de la cual heredan UnidadMilitar y Aldeano.
La diferencia es que UnidadMilitar implementa la interfaz Atacante.
De UnidadMilitar heredan UnidadMilitarTropa y ArmaDeAsedio.
La diferencia radica en que El arma de Asedio tiene distinto comportamiento, debe montarse y desmontarse para realizar acciones. De UnidadMilitarTropa heredan Arquero y Espadachin

5.5. Estados de Unidades

El Aldeano tiene su propio estado implementado con patrón State
El Arma de Asedio también, tiene su propio estado implementado con patrón state.
Para el caso de la UnidadMilitarTropa, el estado esta implementado con un patron strategy

5.6. Construccion de edificios

En el aldeano para construir edificios se uso tambien un patron Double Dispatch,de esta forma se le puede pasar un edificio generico y luego el edificio sabiendo de que tipo es, se agrega al jugador.

5.7. Jugador

El jugador tiene una clase ListaDePiezas, que se implemento para delegar todo el manejo de las unidades y edificios. El jugador maneja el oro.

5.8. Lista de Piezas

La ListaDePiezas internamente guarda cada pieza por separado y es quien las entrega al jugador para que luego este se las pueda entregar al usuario de la clase y porder manipularlas.
Se utilizo una interfaz Manipulable, como interfaz comun a todas las piezas que puede recibir mensajes desde Atacable, NotificableDeTurno y Ubicable.
La ListaDePiezas ademas, maneja la cantidad de poblacion.

5.9. Juego y sus estados

El juego posee un turno, una lista de jugadores , un mapa y un estado de juego.
Se encarga de crear los objetos e inicializarlos.
Tiene un estado implementado con un Strategy que cambia al recibir el mensaje de castillo eliminado

5.10. Turno

El turno se inicializa con un strategy que puede ser Random o Especifico.

5.11. Observadores

Hay 3 tipos de observadores implementados con un patron Observer:
ObservadorJugador, ObservadorJuego,ObservadorCastillo, con sus respectivos Observables.
El Jugador cada vez que agrega o elimina una pieza notifica a su observador (la vista).
El Castillo al morir notifica a su observador que es el Juego.
El Juego luego de ser notificado por el castillo finaliza y notifica a su observador (la vista) que hay fin de juego.
Por su parte, el controlador se encarga de actualizar la vista, solamente en el caso de que una unidad se mueva.
Esto nos pareció que no hacía al modelo ya que un movimiento de una unidad no le interesa ni al jugador ni al juego. y por el tipo de implementación adoptado se dificultaba poder agregar a la vista como observador de las unidades.

5.12. Interfaces

Atacable, hereda de ubicable. Son todas las piezas. Y es usado para que puedan ser atacadas.

Atacante, la implementa Castillo y las unidades militares.

Costeable, es implementada por edificiosConstruibles y unidades.

Ubicable, es implementado por todas las piezas y lo utilliza el mapa y el ataque para calcular la distancia.

Notificable: es implementado por todas las piezas para notificarle que paso un turno.

Manipulable: lo utiliza La ListaDePiezas y el Jugador, para poder implementar algunos metodos genericamente, como por ejemplo EliminarPieza(Manipulable). y NotificarTurno(Manipulable). Esta interfaz hereda de Atacable y Notificable.

6. Excepciones

JuegoEnCursoException Se lanza cuando se intenta obtner el ganador del juego, cuando este aun no finalizo.Es lanzada por la clase EstadoJuegoEnCurso.

JuegoFinalizadoException se lanza si se intenta pasar un turno pero el juego ya finalizo.Es lanzada por la clase EstadoJuegoFinalizado

JugadoresInvalidosException Se lanza cuando la cantidad de jugadores ingresados en invalida, siendo invalido pasar menos de 2 jugadores y mas de dos jugadores. Es lanzada por la clase turno.

TamanioDeMapaInvalidoException Se lanza cuando se intenta crear un mapa con un ancho y un alto menor a los valores que tomamos como minimos(13 ancho y 12 alto).Es lanzada por la clase Juego.

LimiteDePoblacionAlcanzadoException Se lanza cuando se intenta agregar a algun jugador una unidad y este ya alcanzo el limite maximo de poblacion.Es lanzada por la clase Jugador.

NoSePuedeEliminarElCastilloException Se lanza cuando se intenta eliminar un castillo. Es lanzada por la clase lista de piezas.

PiezaAgenaException Se lanza cuando se intenta eliminar una pieza no propia. Es lanzada por la clase ListaDePiezas.

PiezaYaAgregadaException Se lanza cuando la posicion a la que se quiere agregar a la lista de piezas una pieza ya agregada anteriormente. Es lanzada por ListaDePiezas.

CantidadIncorrectaAldeanosException Se utiliza cuando se le pasa al jugador en el constructor una lista de aldeanos, con una cantidad menor que 3 o mayor que 3.Es lanzada por Jugador.

CostoNegativoExeption Se utiliza cuando se quiere descontar una cantidad de oro negativa del jugador.Es lanzada por la clase Jugador.

NoSePuedeAgregarPiezaException Se utiliza cuando se quiere agregar una pieza al jugador y no hay oro suficiente, cuando se quiere agregar una unidad o un edificio que ya esta agregado al jugador.Es lanzada por la clase Jugador.

OroNegativoException Se utiliza cuando se quiere sumar una cantidad de oro negativa del jugador.Es lanzada por la clase Jugador.

RecursoInsuficienteException Se utiliza cuando se quiere descontar al jugador una cantidad de oro mayor a la que actualmente tiene.Es lanzada por la clase Jugador.

ElElementoYaExisteException Se utiliza cuando se quiere agregar al mapa una pieza que ya esta en el. Es lanzada por la clase Mapa.

FueraDelMapaException Se utiliza cuando se quiere colocar una pieza en el mapa en una posicion que esta afuera de el. Es lanzada por Mapa.

NoExisteElementoException Se utiliza cuando se quiere rmover una pieza del mapa y la pieza nunca fue colocada en el. Es lanzada por Mapa.

NoSePuedeMoverElElementoException Se utiliza cuando se quiere descontar una cantidad de oro negativa del jugador.Es lanzada por la clase Jugador.

PosicionNulaException Se utiliza en la clase PosicionNula en la cual hicimos uso del patron Null Object.Se lanza en todos los metodos de esta clase cuando son invocados.

NoSePuedeRepararException Se utiliza cuando se quiere reparar con un aldeano,un edificio que no esta en estado daniado.Es lanzada por la clase Aldeano.

DebeDesmontarsePrimeroException Es lanzada por la clase ArmaDeAsedio cuando se quiere mover la unidad y esta en estado montada o montandose.

DebeMontarsePrimeroException Es lanzada por la clase ArmaDeAsedio cuando se quiere mover la unidad y esta en estado desmontada o desmontandose.

UnidadYaRealizoMovimientoEsteTurnoException Se lanza por todas las clases que heredan de unidad militar, es decir, Arquero,Espadachin,Arma de asedio cuando se intenta que realicen mas de una accion por turno.

AldeanoOcupadoException Se utiliza cuando se quiere construir o reparar algun edificio con un aldeano que esta reparando,construyendo o que ya realizo movimientos en el turno, Es lanzada por Aldeano.

EdificioYaAgregadoException Se utiliza cuando se intenta que un aldeano construya un edificio ya existente.Es lanzada por Aldeano.

NoSePuedeConstruir Se utiliza cuando se quiere construir un edificio en una posicion invalida, cuando el otro es insuficiente para la construccion e igualmente se intenta construir o cuando el edificio no esta en estado a construir. Es lanzada por Aldeano.

EdificioYaAgregadoException Se utiliza cuando se intenta que un aldeano construya un edificio ya existente.Es lanzada por Aldeano.

EdificioYaAgregadoException Se utiliza cuando se intenta que un aldeano construya un edificio ya existente.Es lanzada por Aldeano.

UnidadFueraDeRangoDeAtaqueException Se utiliza cuando se quiere atacar a una unidad o un edificio que se encuentra a una distancia mayor que el rango de la unidad atacante. Es lanzada por todas las clases que heredan de unidad militar(espadachin, arquero, armaDeAsedio).

EnemigoSinJugadorException Se lanza cuando se ataca a una unidad que no pertenece a ningun jugador. Es lanzada por cualquier unidadmilitar.

EstoyEnDosJugadoresException Se lanza cuando se ataca a una unidad que pertenece a ambos jugadores. Es lanzada por cualquier unidadmilitar.

FuegoAmigoException Se lanza cuando se ataca a una unidad perteneciente al mismo jugador que la unidad que ataca. Es lanzada por cualquier unidadmilitar.

NoEsMiJugadorException Se lanza cuando algun jugador intenta atacar con una unidad que no es de el. Es lanzada por cualquier unidadmilitar.

7. Diagramas de secuencia

Diagrama de secuencia de la creacion de un edificio por un aldeano. Se considera que la posicion tanto del aldeeano, como del edificio a construir son validas, alcanza el oro del jugador para construirlo y el aldeano esta en estado libre.

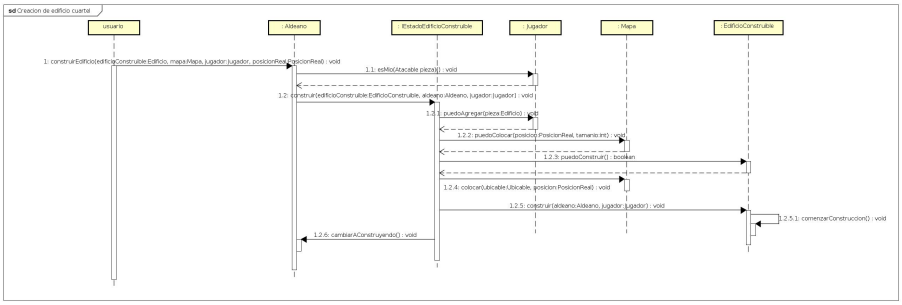

Figura 10: Diagrama de Secuencia de creacion de edificios

A continuacion diagrama de secuencia del ataque de un arquero, que se encuentra en posicion valida y puede atacar a la unidad enemiga.

de un arquero.jpg

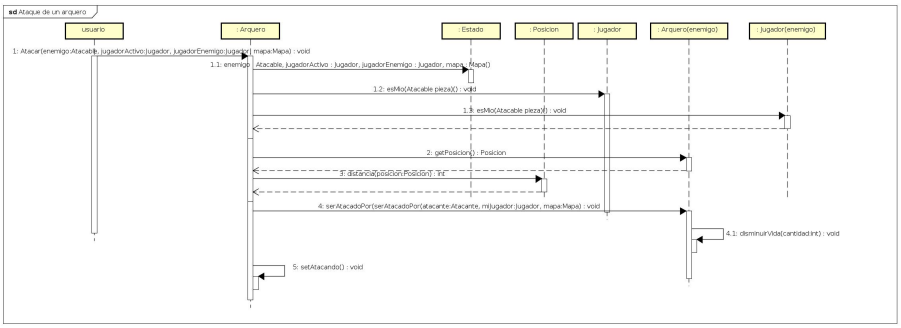

Figura 11: Diagrama de Secuencia ataque de un arquero

A continuacion diagrama de secuencia del movimiento de un aldeano que se encuentra en una posicion valida y se va a mover hacia otra posicion valida, dentro del mapa y a distancia 1, en una direccion valida.

de aldeano - Secuencia.jpg

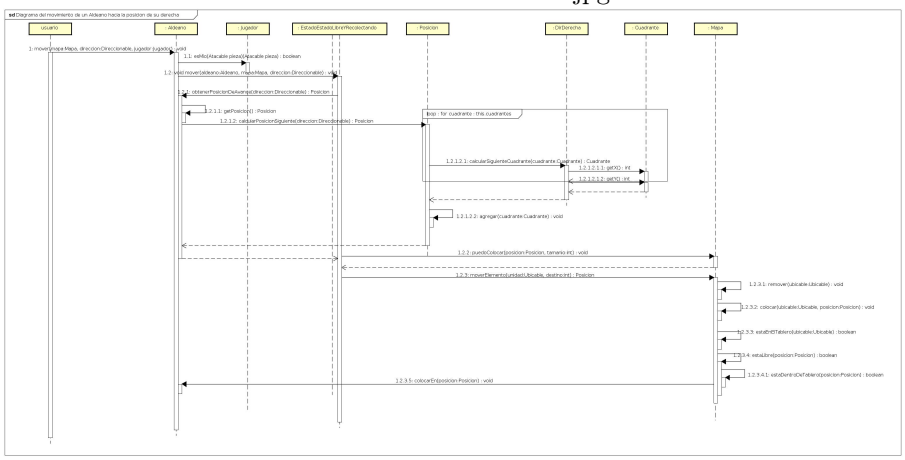

Figura 12: Diagrama de Secuencia del movimiento de un aldeano