Trabajo Práctico 2 — AlgoEmpires

[7507/9502] Algoritmos y Programación III
Curso 1
Segundo cuatrimestre de 2018

Bonin German:	99050
Copertini Felipe:	101651
Funes Pablo:	94894

Índice

1. Introducción

El presente informe reune la documentación de la solución del trabajo práctico final de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación similar al juego .ªge of Empiresçonceptos del paradigma de la orientación a objetos vistos en el curso.

2. Supuestos

Acciones del Jugador : Las acciones de un jugador son limitadas a un turno. Es decir, por cada accion que tome este, es lo que le va a costar el turno. Sea mover la unidad, sea atacar, construir, etc. Esta permitido nada mas realizar una por turno.

3. Modelo de dominio

Paquete modelo:

Clase Juego : Representa el flujo del juego.

Clase Jugador: Representa un modelo de cada jugador con sus atributos predefinidos (poblacion, oro), y ademas contiene sus respectivas acciones que puede realizar a lo largo del juego. Clase Mapa: Representa una abstraccion de lo que nos guiariamos como mapa del juego.

Clase Unidad: Es una representacion de los tipos de unidades con los que contamos en el juego. Clases como Arquero, Espadachin, ArmaDeAsedio, Aldeano, heredan de ella. Ya que todas cuentan con al menos caracteristicas similares que necesitan utilizar.

Clase Edificio: Representacion de los tipos de edificios con los que vamos a trabajar a lo largo del desarrollo del trabajo. Contamos con Castillo, Plaza Central, y Cuartel.

Clase Mapa: Es la representacion del tablero donde va a ocurrir todo, este consta de posiciones donde cada una contiene la informacion lleva al momento (Unidades, Edificios, o espacios vacios).

Paquete Vista:

RegistradorJugadores: Es la vista inicial del programa, en esta se almacena los nombres de ambos jugadores. Posee un Boton que nos permite iniciar el Juego.

Clase Boton: Es el accionable que controla el flujo del programa. Nos permite transitar de una escena a otra durante el juego.

VistaPrincipal: Es donde transcurre todo el juego. Cuenta con las acciones permitidas al jugador y la representacion del mapa.

Paquete Controladores:

Casillero: Representacion de la celda del tablero. Contiene la informacion tanto visual como la representacion del codigo de quien esta ocupando la celda del mapa.

EmpezarJuego: Es el flujo de transicion de una escena a otra.

InputUsuario: Encargado de capturar el texto ingresado por pantalla.

4. Diagramas de clase

Nunc molestie facilisis diam in auctor. Nulla sed porta nibh, eu elementum erat. Vestibulum in lectus ornare, sollicitudin ipsum eget, posuere risus. Duis ac ante sagittis, ornare urna a, scelerisque purus. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut commodo ultricies luctus. In a elit malesuada, semper felis at, varius lorem. Aenean hendrerit vitae lorem sit amet porttitor. Suspendisse vitae vulputate elit, a commodo lacus. Phasellus maximus arcu et eros sollicitudin, eu aliquet nulla efficitur. Aenean semper neque nec dignissim rutrum. Aliquam at purus vel tortor fringilla iaculis sit amet sit amet metus. Pellentesque faucibus a nulla eget molestie.

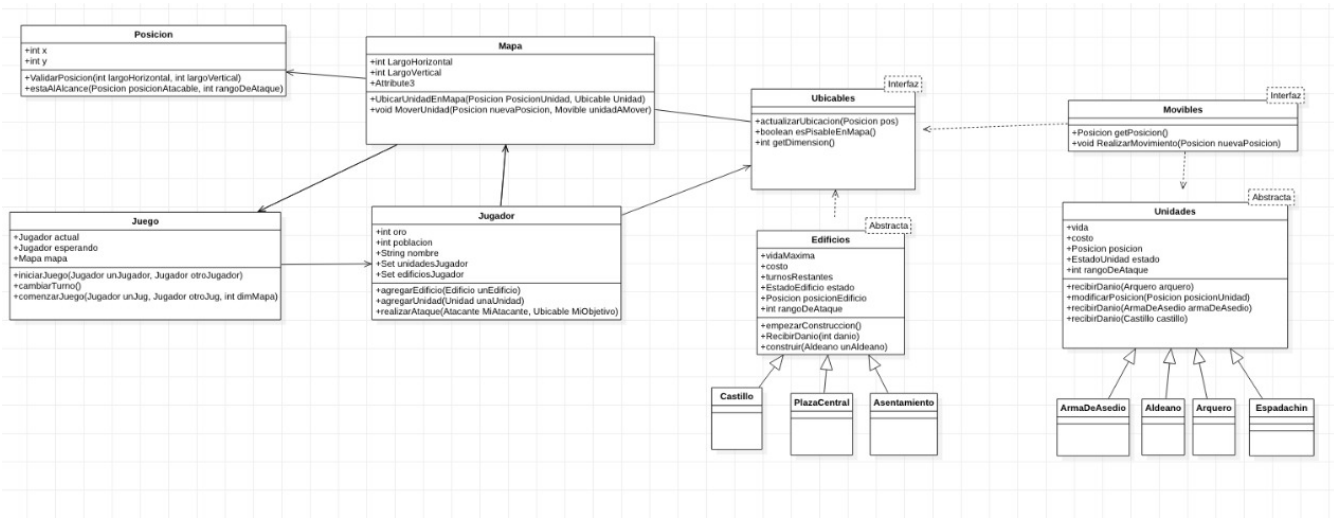

Figura 1: Diagrama de clase.

5. Detalles de implementación

5.1. Unidad y Edificio

El daño que reciben de parte de otras unidades es un atributo. Cada unidad que te ataca, afecta al mismo conjunto por igual. Ejemplo: si un arquero ataca a un edificio, le pega por igual a todos. El objeto que recibe daño es el encargado de quitarse los puntos de vida, ya que este sabe que "tipo de dato"lo ataca.

```
public abstract class Edificio implements Ubicable {
    protected int vidaMaxima;
    [....]
    protected int danioProducidoPorArquero=10;
    protected int danioProducidoPorCastillo=20;
    protected int danioProducidoPorEspadachin=15;
    protected int danioProducidoPorArmaDeAsedio=75;
    [....]

public abstract class Unidad{
    protected int vida;
    protected int costo;
    [...]
    protected int puntosDeAtaque=0;
    protected int danioProducidoPorArquero=15;
    protected int danioProducidoPorCastillo=20;
    protected int getDanioProducidoPorEspadachin=25;
    protected int getDanioProducidoPorArmaDeAsedio=0;
    [...]
```

5.2. Acciones del jugador

Como comentamos en el supuesto, la menos engorrosa de tener un flujo claro del juego para nosotros fue concluir de que cada turno del jugador iba a estar limitado a la accion que este realice. Sea atacar una unidad, construir un edificio, o crear un Aldeano. Optar por la otra implementacion

donde cada jugador podria realizar una cantidad x de acciones por turno, implicaba llevar un "trackeo"de todas las unidades hasta el momento y tambien considerar si estas realizaron alguna accion.

De esta manera nosotros pudimos condensar el flujo del juego en las acciones, lo cual consideramos que es un mas legible que la otra implementacion.

Un ejemplo para mostrar es mover una unidad en el mapa:

```
public void moverUnidad(Juego unJuego, Mapa unMapa, Movible unaUnidad, Direccion unaDireccion
    Posicion posicionMover = unaDireccion.ObtenerPosicion(unaUnidad);
    unMapa.MoverUnidad(posicionMover,unaUnidad);
    if (unJuego.getNombreActual() != this.nombre ){
        throw new JugadaInvalidaException();
    }
    unJuego.cambiarTurno();
}
```

6. Excepciones

AldeanoOcupadoException Sucede cuando se intenta cambiar el estado de un Aldeano y este ya esta realizando otra tarea en el momento. Por tarea puede entenderse construir un edificio, o repararlo.

AtaqueFueraDeRango Cuando una unidad intenta atacar fuera del rango que tiene permitido se levanta esta excepcion.

DimensionInvalidaMapa Cuando se intenta crear un mapa con dimensiones invalidas

TopePoblacionExcepcion Se lanza esta excepcion si se intenta sumar una unidad mas cuando ya se alcanzo el limite de poblacion.

MovimientoFueraDelMapa Se lanza cuando se intenta mover una unidad a rangos invalidos para el mapa.

JugadaInvalidaExcepcion Se lanza cuando se intenta jugar en un turno que no corresponde.

UbicacionFueraDelMapaExcepcion Se lanza cuando se intenta acceder a una Ubicacion invalida en el mapa.

UbicacionOcupadaPorOtraUnidad Se lanza cuando se intenta pisar una unidad en la ubicacion.

7. Diagramas de secuencia

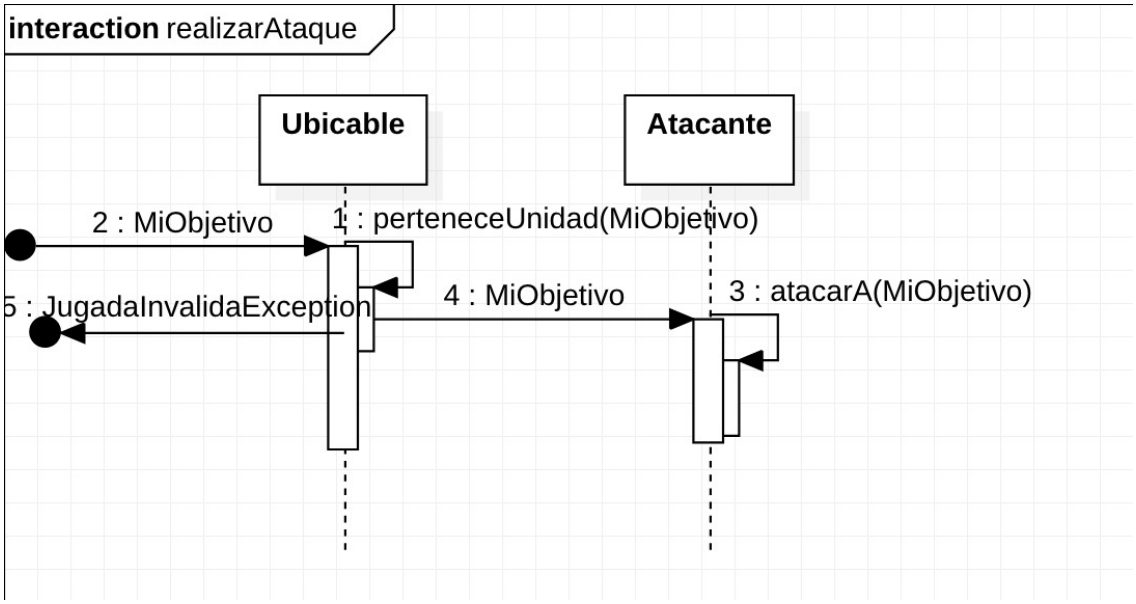

Figura 2: Realizar ataque.