

Trabajo Práctico 2 — Java

[7507/9502] Algoritmos y Programación III

Curso 1

Segundo cuatrimestre de 2018

Grupo compuesto por:

Balestieri, Juan Diego- 101601 - diego.balestieri1998@gmail.com

Lampropulos, Santiago Nicolás - 101862 - Santiagolampropulos@gmail.com

Giampietri, Mauro Gabriel-101186 - flia\_giampietri@yahoo.com

Sturla, Manuel - 100912 - manuelsturla@gmail.com

# INDICE

[INDICE 2](#_Toc2)

[Supuestos 3](#_Toc3)

[Diagramas de clases 3](#_Toc4)

[Diagramas de secuencia 5](#_Toc5)

[Diagrama de paquetes 5](#_Toc6)

[Diagramas de estado 5](#_Toc7)

[Detalles de implementación 6](#_Toc8)

[Estado de los Ubicables - State 6](#_Toc9)

[Mapa - Composite 6](#_Toc10)

[Excepciones 6](#_Toc11)

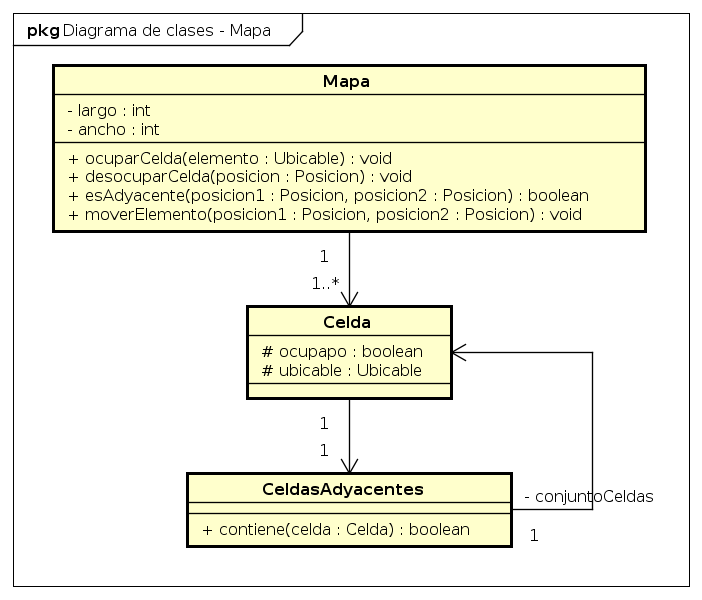
# Supuestos

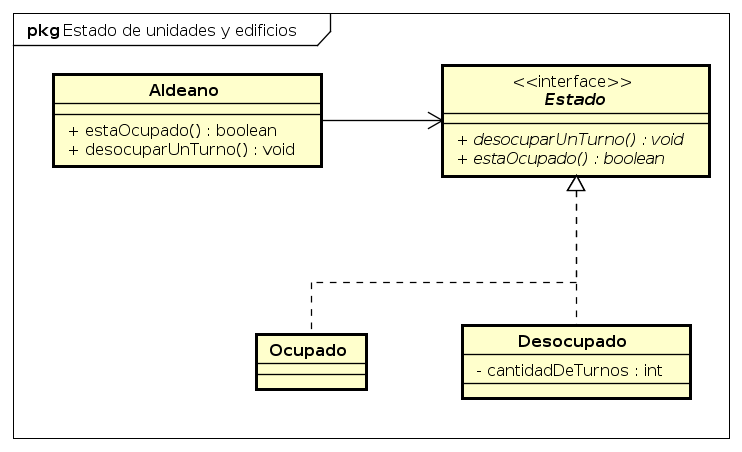
En el siguiente trabajo, utilizamos los siguientes supuestos:

# Diagramas de clases

*Varios diagramas de clases, mostrando la relación estática entre las clases. Pueden agregar todo el texto necesario para aclarar y explicar su diseño de manera tal que logre el modelo logre comunicarse de manera efectiva.*

Para modelar el siguente trabajo utilizamos un conjunto de clases de acuerdo a los siguientes diagramas. Estos mostrarán el modelo general de manera fragmentada y esquelética con el fin de condensar la información relevante de forma de que sea transmitida al lector más fácilmente.

**

**

En este diagrama se muestra, usando como ejemplo a un Aldeano, cómo se modeló que las unidades y los edificios puedan aparecer como "ocupadas" cuando ya realizaron alguna accion en ese turno; o realizaron alguna accion que lleva varios turnos terminar. Ver **Estado de los Ubicables.**

# Diagramas de secuencia

*Varios diagramas de secuencia, mostrando la relación dinámica entre distintos objetos planteando una gran cantidad de escenarios que contemplen las secuencias más interesantes del modelo.*

# Diagrama de paquetes

*Incluir un diagrama de paquetes UML para mostrar el acoplamiento de su trabajo.*

# Diagramas de estado

*Incluir diagramas de estados, mostrando tanto los estados como  las distintas transiciones para varias entidades del modelo*.

# Detalles de implementación

*Deben detallar/explicar qué estrategias utilizaron para resolver todos los puntos más conflictivos del trabajo práctico. Mencionar qué patrones de diseño fueron utilizados y por qué motivos.*

## Estado de los Ubicables - State

Con el fin de poder verificar que cada elemento pueda hacer solo una acción en un turno, y sabiendo que ese estado del elemento varía su comportamiento, decidimos usar el patrón State para representar este aspecto.

//Diagrama de clases chiquitito!

En el diagrama, se puede ver como el aldeano tiene un Estado, que puede ser Ocupado o Desocupado y dependiendo de qué estado tenga, su respuesta a métodos como estaOcupado() es distinta.

## Mapa - Composite

A la hora de modelar el mapa, decidimos que este sea un conjunto de celdas, las cuales tienen a su vez a su conjunto de celdas adyacentes como se puede ver en el diagrama **[inserte numero de diagrama de mapa aqui]**.Este modelado de las celdas responde a un patrón Composite, ya que nos permite construir una estructura compleja como es el mapa, mediante la composicion recursiva de instancias de clases mas simples como son las celdas.

# Excepciones

* **ElementoFueraDeRangoException**:
* **NoSePudoConstruirException:** Se utiliza para señalar que el intento de construir un edificio fue fallido. Esto se puede dar porque las posiciones que se pensaban ocupar estaban ya ocupadas en el mapa o fuera de rango.
* **MovimientoNoPermitidoException**: Esta excepción es la encargada de señalar cuando se intenta realizar un movimiento con un Ubicable que no está permitido y esto se puede dar por distintas razones:
  + La posición de destino está ocupada
  + La posición de destino esta está fuera del mapa
* **PosicionException**: engloba las excepciones: PosicionFueraDeRangoException y **PosicionNoDisponibleException**, para que luego sean capturadas por las unidades movibles cuando se las intenta mover y no se puede. Se arroja en cambio MovimientoNoPermitidoException.
* **PoscionFueraDeRangoException**: Señala que la posición a la cual se trató de acceder para mover un elemento (Ubicable) o posicionarlo se encuentra fuera del mapa.
* **PosicionNoDisponibleException**: Señala que la posición a la cual se trató acceder para mover un elemento (Ubicable) o posicionarlo, se encuentra ocupada.
* **UbicableEstaOcupadoException**: Se utiliza para señalar que el Ubicable en cuestión no está disponible para realizar alguna acción. Se puede activar por varias razones: al Ubicable ya se le indico que realice una acción en ese turno, o que ya estuviera ocupado realizando una acción que lleva varios turnos.
* **ArmaNoDesmontadaException**: Se utiliza para señalar que el Arma de Asedio no puede realizar una determinada acción (por ejemplo, moverse) por estar montada.