

# 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
SUPUESTOS .....	3
DIAGRAMAS DE CLASES .....	4
DIAGRAMAS DE SECUENCIA .....	8
DIAGRAMA DE PAQUETES.....	10
DIAGRAMAS DE ESTADO.....	10
DETALLES DE IMPLEMENTACIÓN.....	11
ESTADO DE LOS UBICABLES - STATE .....	11
MAPA - COMPOSITE.....	11
EXCEPCIONES .....	11

# Supuestos

*En el siguiente trabajo, utilizamos los siguientes supuestos:*

- Suponemos que en el 1er turno de cada jugador no se cobre oro a sus 3 aldeanos desocupados*
- Suponemos que toda unidad y todo edificio solo pueden realizar una acción por turno*
- Suponemos que un ubicable no puede atacar a otro de su misma facción*

# Diagramas de clases

Para modelar el siguiente 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.

pkgDiagrama de clases - Mapa

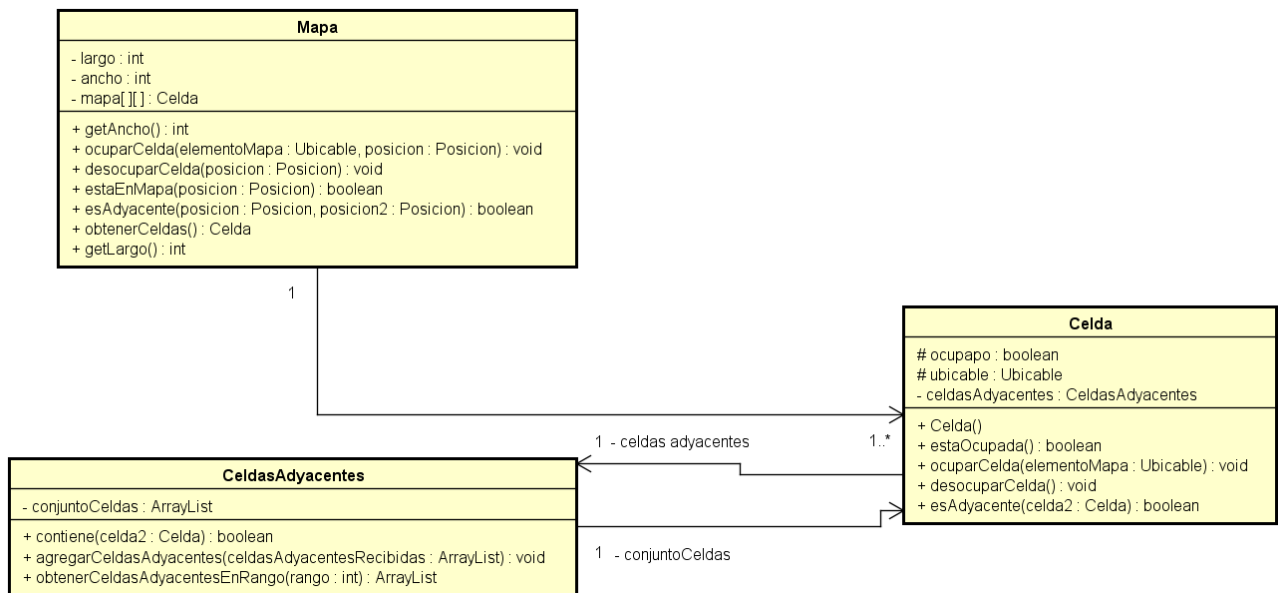


Diagrama de clases 1

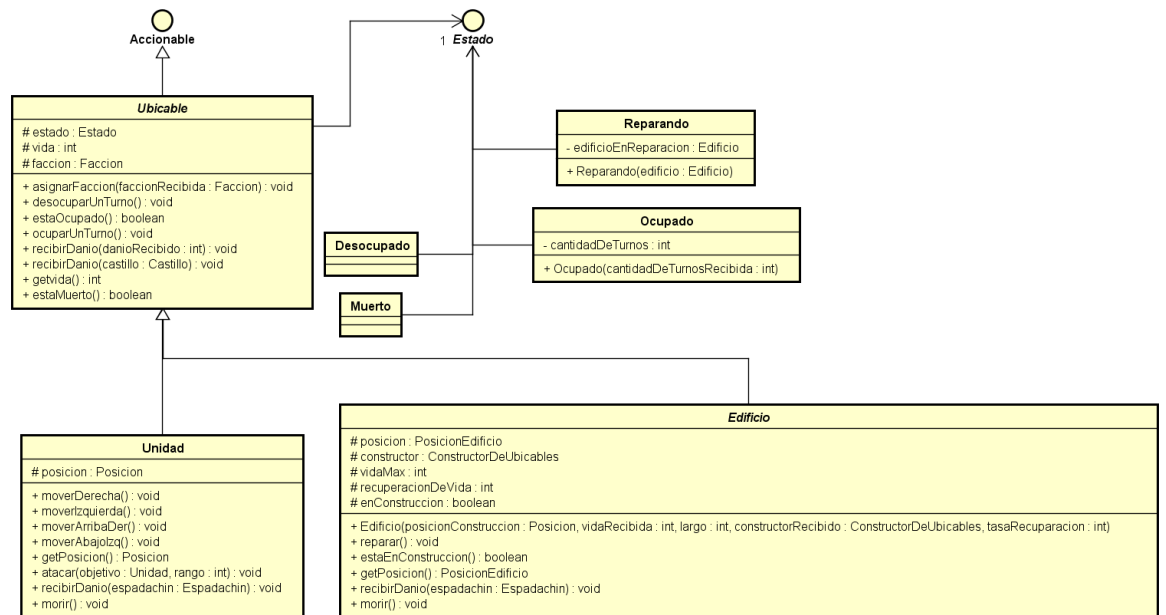


Diagrama de clases 2

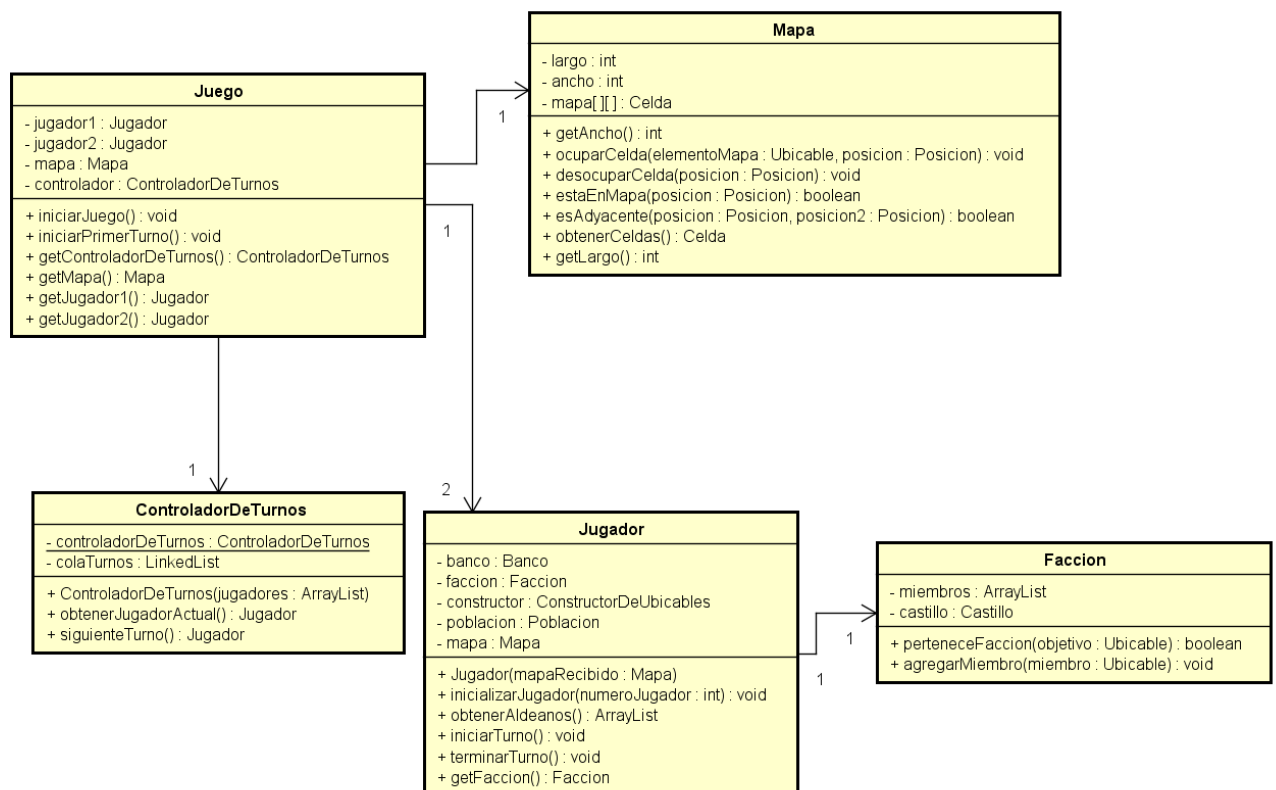


Diagrama de clases 3

pkg

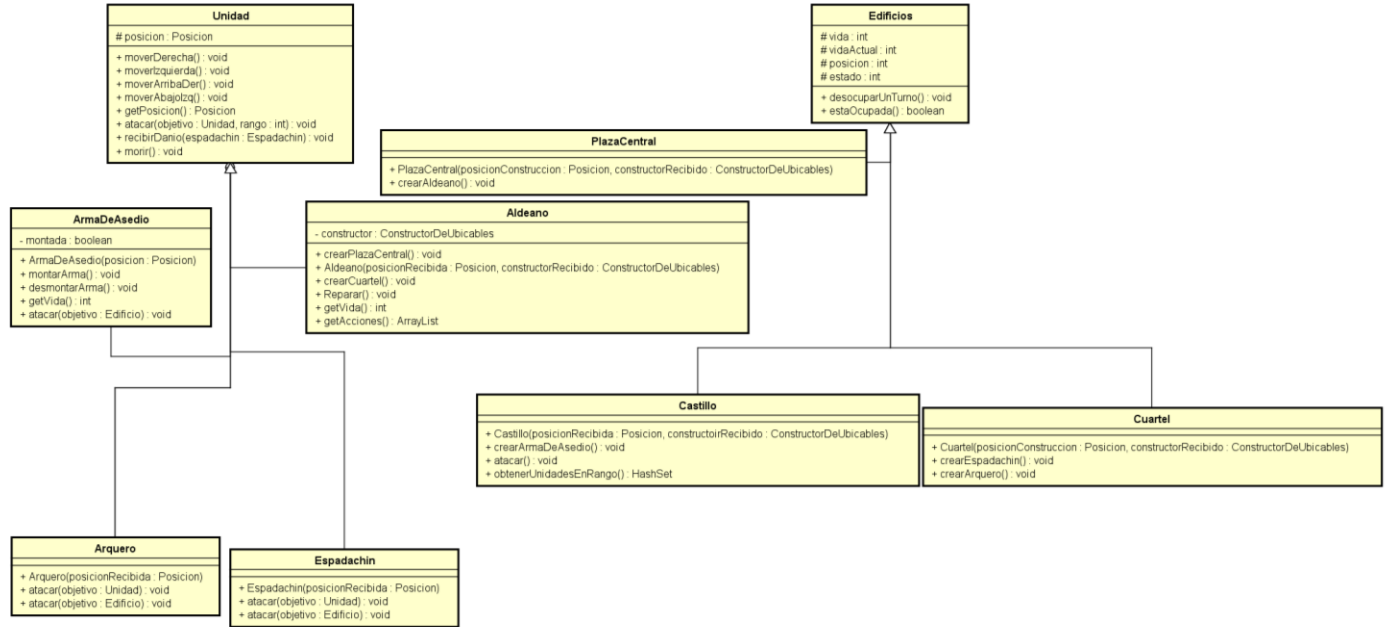
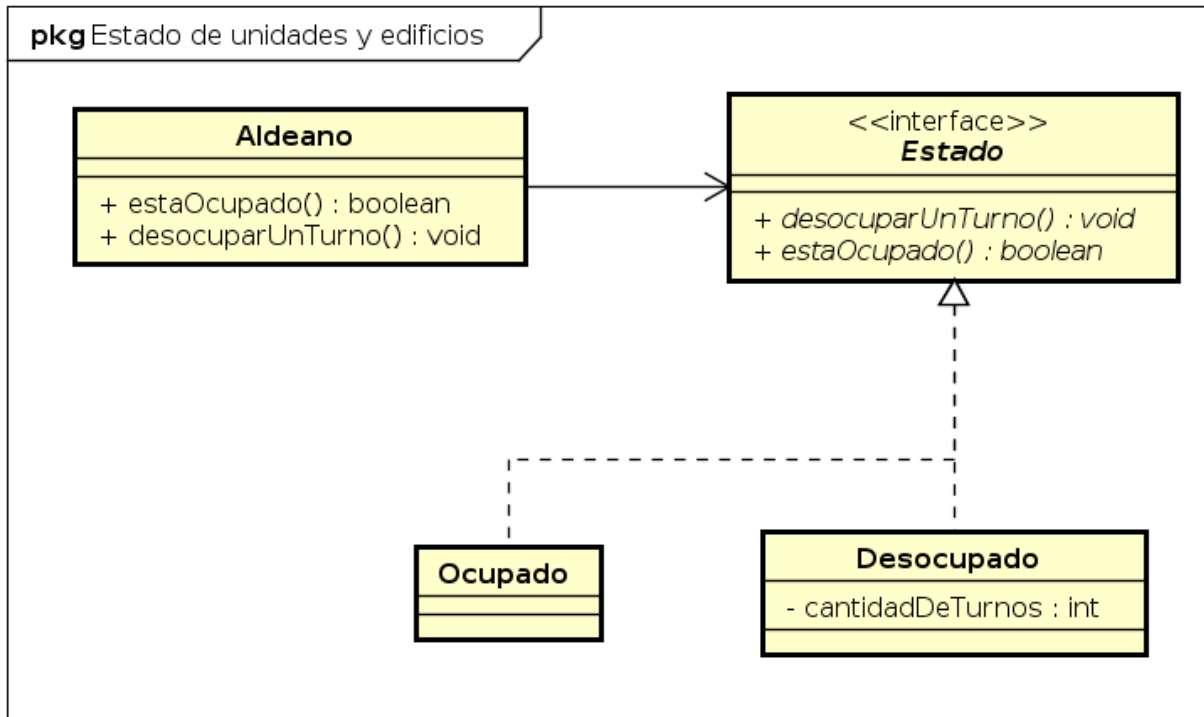
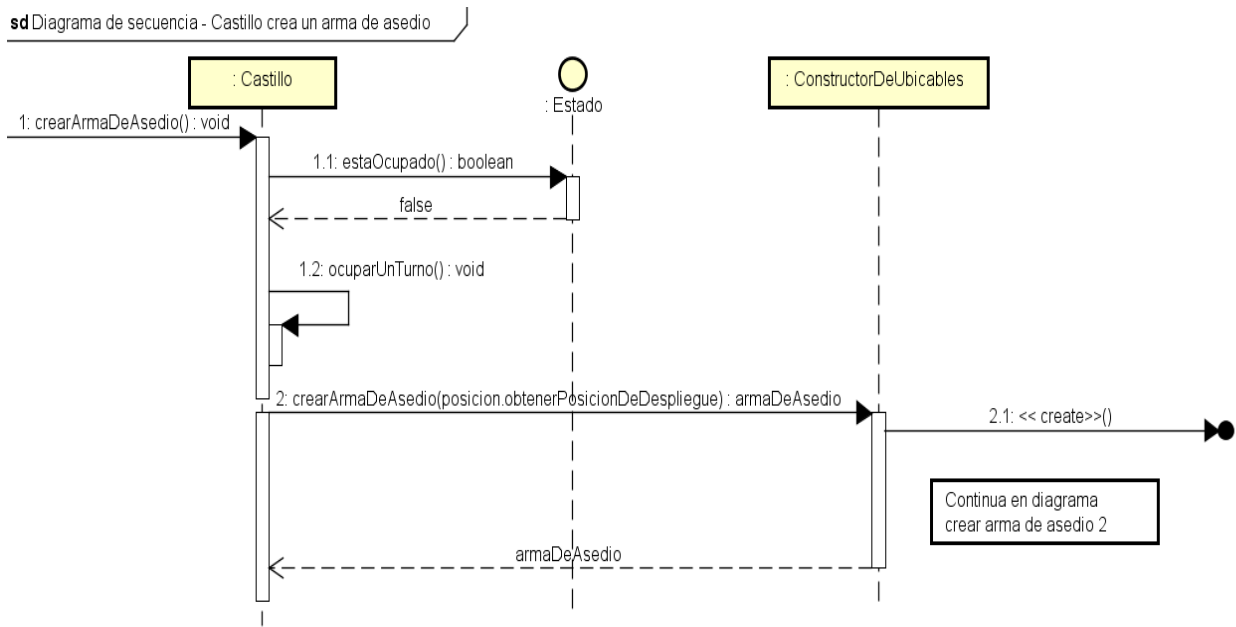
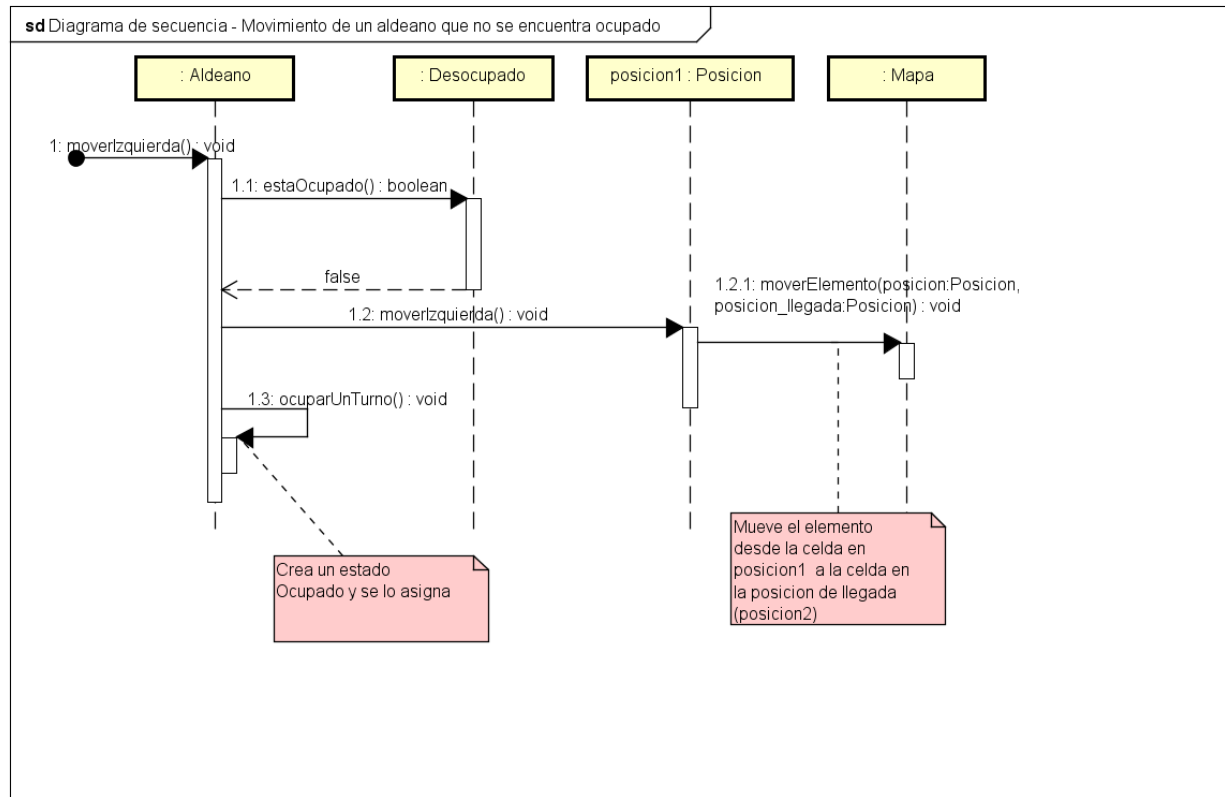


Diagrama de clases 4

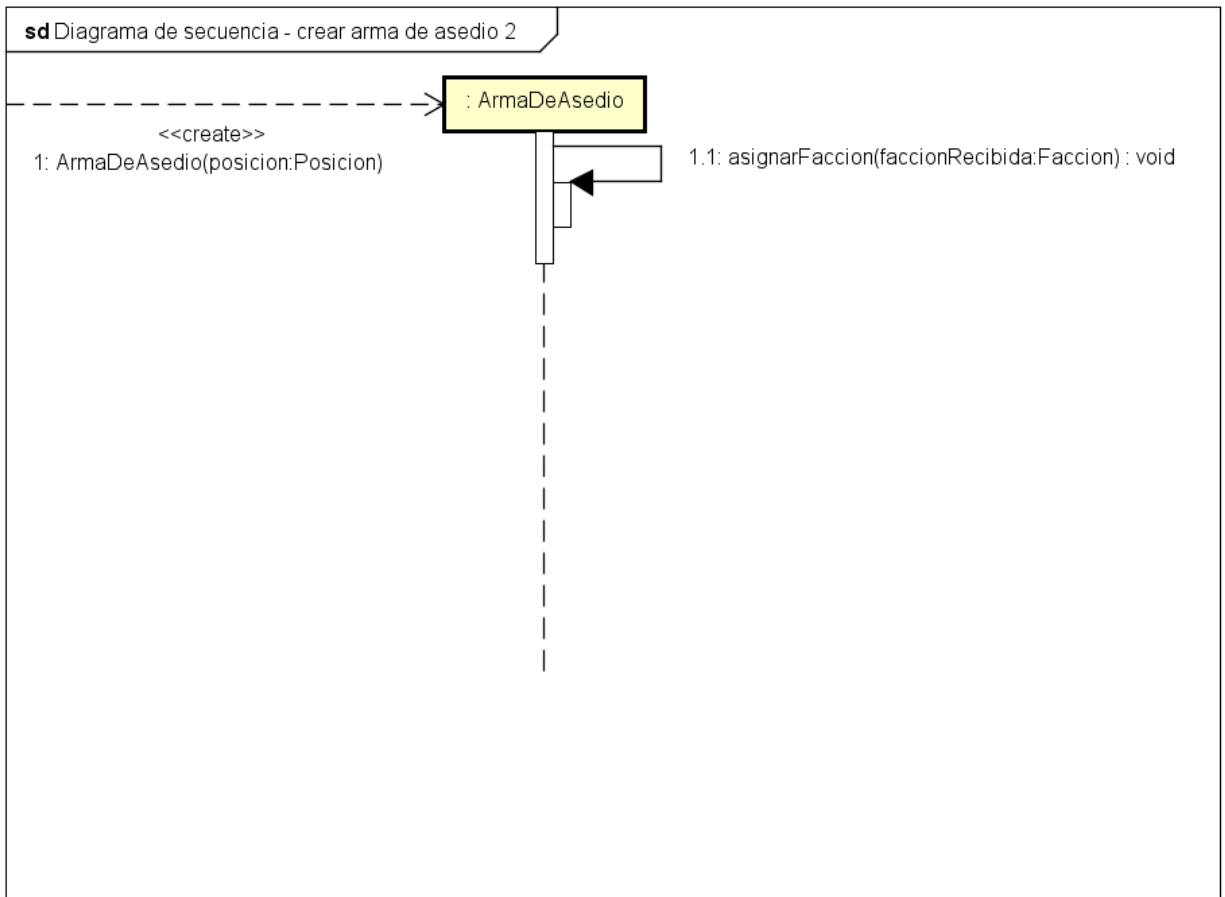


En este diagrama se muestra, usando como ejemplo a un Aldeano, cómo se modeló que las unidades y los edificios puedan aparecer como "ocupadas" cuando ya realizaron alguna acción en ese turno; o realizaron alguna acción que lleva varios turnos terminar.

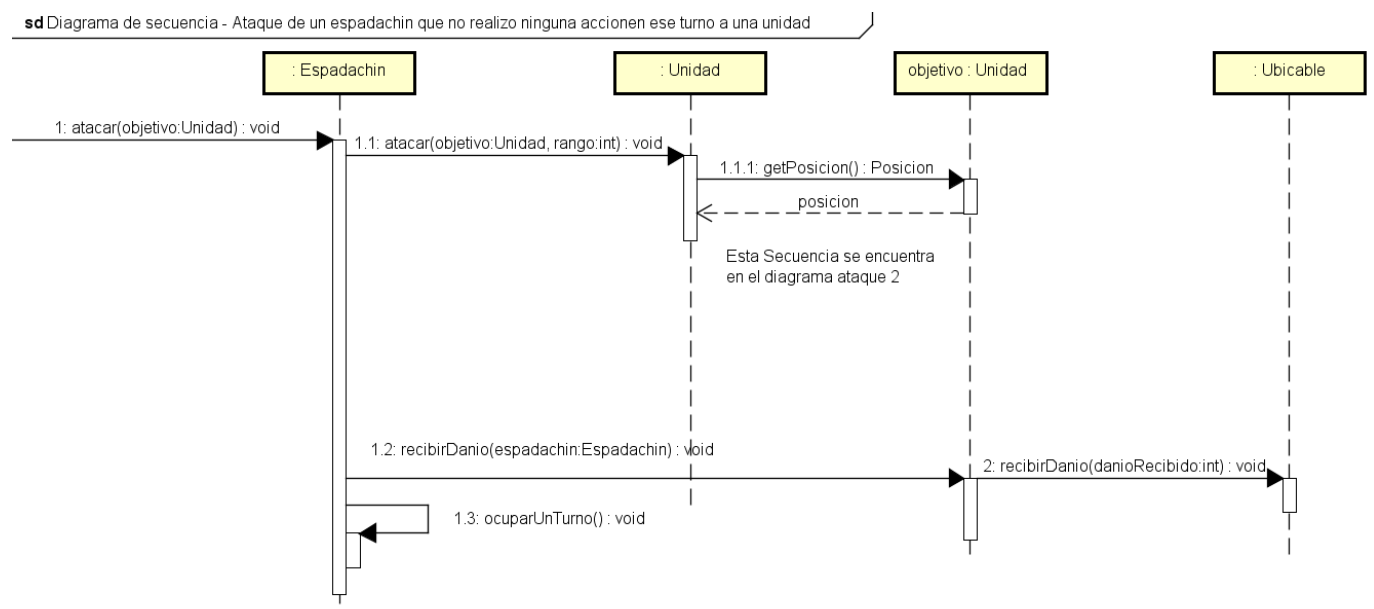
# Diagramas de secuencia

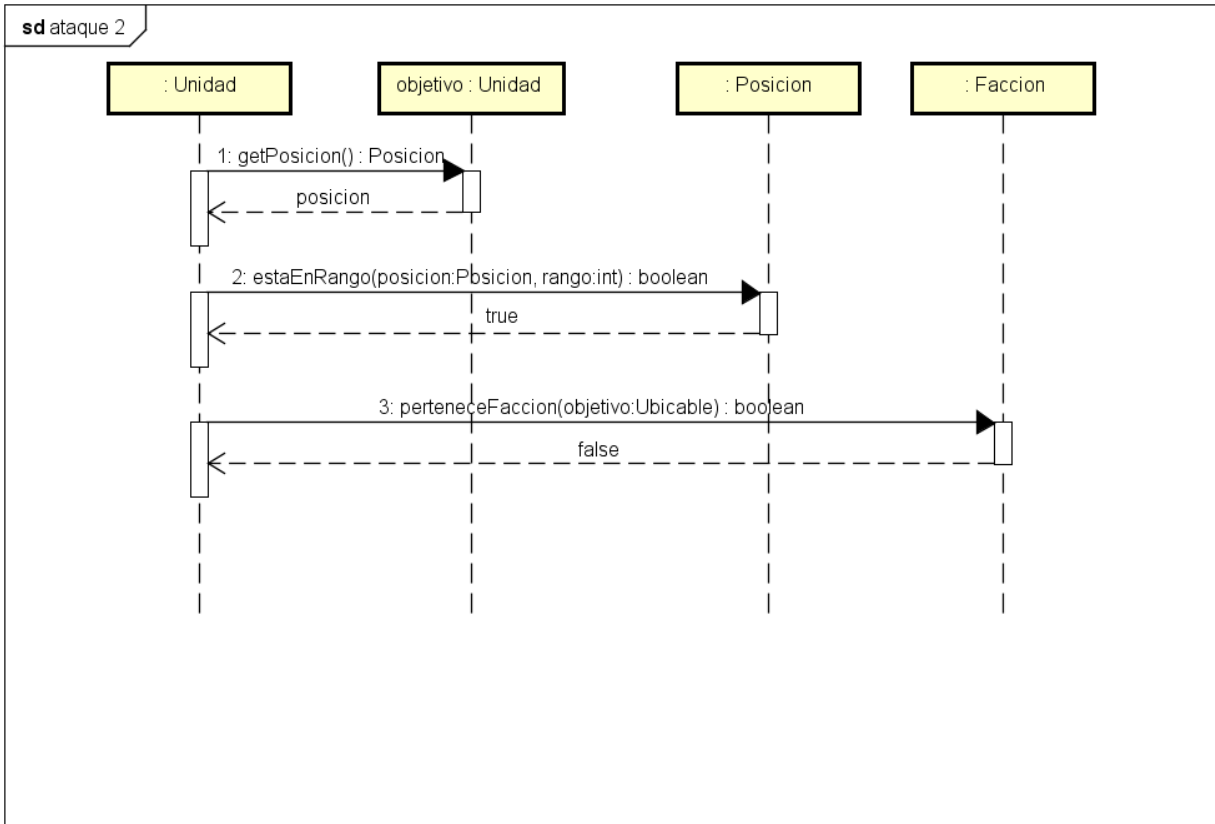






*crear arma de asedio 2*





Ataque 2

## 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

## 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.

En el diagrama de clases 2, se puede ver como los ubicables tienen 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 [diagrama de clases 1]. Este modelado de las celdas responde a un patrón Composite, ya que nos permite construir una estructura compleja como es el mapa, mediante la composición recursiva de instancias de clases más simples como son las celdas.

## Excepciones

- **PosicionInvalidaException:** Se utiliza para señalar que el intento de construir un edificio o una unidad fue fallido. Esto se puede dar porque las posiciones que se pensaban ocupar estaban ya ocupadas en el mapa o fuera de rango.

También 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
- **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.
- **OroInsuficienteException.** Se utiliza para señalar que no hay oro suficiente para crear un ubicable.
- **UnidadesMaximasException.** Se utiliza para señalar que el jugador ya tiene 50 unidades y no puede crear más.
- **UbicableFueraDeRangoException.** Se utiliza para señalar que el edificio o la unidad atacada no esta en el rango de ataque.
- **UbicableDeMismaFaccionException.** Se utiliza para señalar que el ubicable atacado pertenece a la misma facción que el atacante.
- **UbicableDeOtraFaccionException.** Se utiliza para señalar que el ubicable a reparar pertenece a una facción distinta que el aldeano.