UNRN

Pensamiento en clases y objetos

Programación II

Gonzalo E. Aburto
12-4-2025

Para ponernos en contexto debemos pensar en el videojuego Minecraft, un juego programado en Java que servirá como ejemplo para explicar los conceptos de la Programación Orientada a Objetos.

Minecraft se basa en un mundo formado de entidades físicas. Estas entidades no son solo los bloques, objetos y las criaturas con los cuales el jugador puede interactuar, sino que los mismos jugadores cuentan como entidades. Por lo que podemos definir a las entidades como todo aquello con lo que se puede interactuar, ya sea de forma directa o indirecta.

En este escrito nos vamos a centrar en aquellas entidades que cuentan con la capacidad de vivir y morir, con comportamientos específicos, ya sea hostil, no hostil o las entidades neutras, que reaccionan en defensa propia; además de los jugadores, los cuales cuentan con un comportamiento variable.

Dentro del juego, podemos identificar distintos tipos de entidades "vivas" como lo son:

- Entidad: Clase base, de ella salen los demás tipos
- Jugador
- Espectador: Estado del jugador
- *Mob* que se diferencian por su nivel de hostilidad.

Dentro de cada tipo podemos identificar los siguientes atributos:

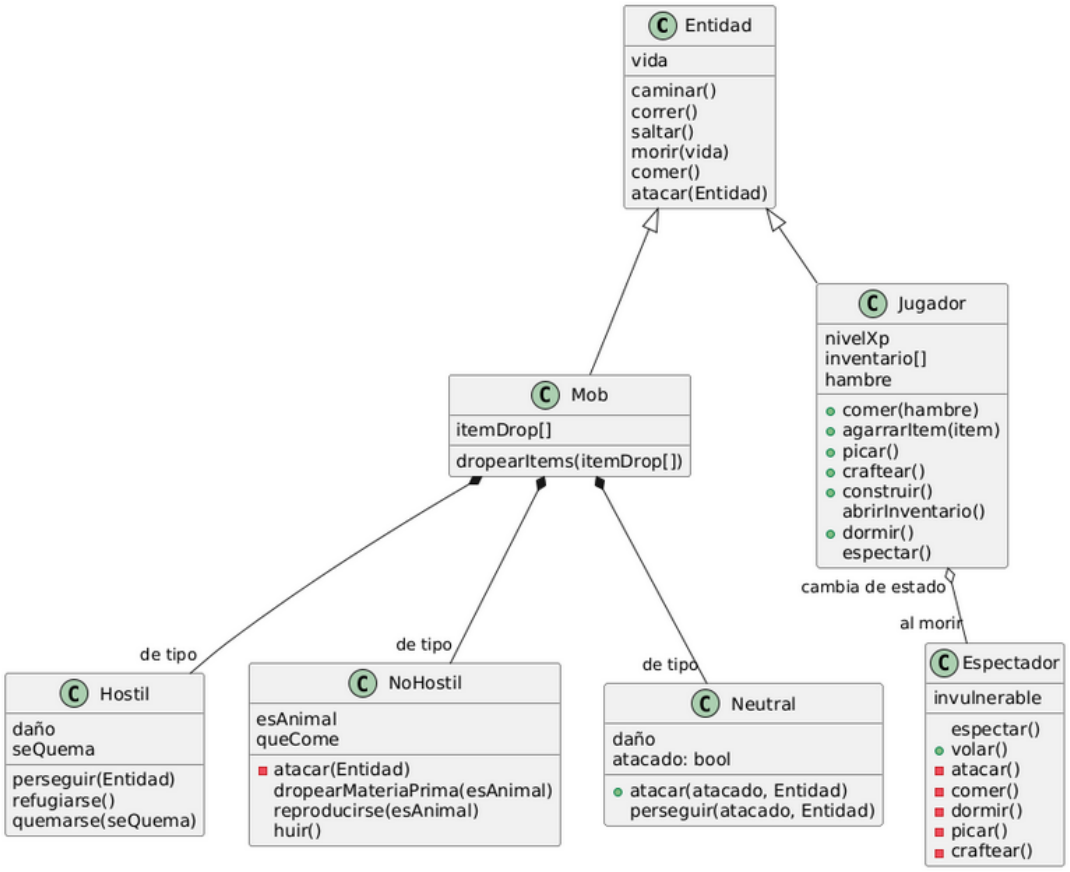
- Entidad: Vida
- Jugador: Vida, nivel de XP, inventario, hambre
- Espectador: Invulnerabilidad
- Mob: Vida, objeto que deja al morir.
    - Hostil: Cantidad de daño que realiza, se quema con el sol o no
    - No hostil: Es animal, que come
    - Neutral: Cantidad de daño que realiza, esta siendo atacado o no

Si descomponemos a las entidades e identificamos sus partes nos quedamos con que una entidad se caracteriza por tener comportamientos bien definidos.

- Una entidad a grandes rasgos puede moverse (caminar, correr, saltar), comer y morir. Por temas de reutilización, se agrega el comportamiento de atacar otras entidades.
- El jugador al ser hijo de la clase entidad hereda sus comportamientos y suma la capacidad de tener hambre, atacar otras entidades, picar bloques, crear objetos (***craftear***) y estructuras (construir), abrir el inventario, dormir y cambiar al estado espectador (cuando muere).

- Cuando el jugador cambia al estado espectador, pierde la capacidad de atacar, comer, dormir, picar y construir.
- Los *mobs* en general, comparten la característica de soltar objetos al morir (dropear ítems).
    - Los *mobs* hostiles pueden perseguir y atacar una entidad, quemarse con el sol (no todos) y si se queman con el sol, entonces también buscan refugio durante el día.
    - Los *mobs* no hostiles no pueden atacar otras entidades, en cambio pueden huir, soltar materia prima y reproducirse.
    - Por último, los *mobs* neutrales pueden atacar solo si los atacan primero y perseguir a una entidad.

Para complementar toda esta información, a continuación, se deja un diagrama de clases hecho en UML con su respectivo código:



```
@startuml

Entidad <|-- Mob
Entidad <|-- Jugador

Mob *-- "de tipo"Hostil
Mob *-- "de tipo"NoHostil
Mob *-- "de tipo"Neutral

Jugador"cambia de estado" o-- "al morir" Espectador

class Entidad {
    vida
    caminar()
    correr()
    saltar()
    morir(vida)
    comer()
    atacar(Entidad)
}

class Mob {
    itemDrop[]
    dropearItems(itemDrop[])
}
class Jugador {
    nivelXp
    inventario[]
    hambre

    +comer(hambre)
    +agarrarItem(item)
    +picar()
    +craftear()
    +construir()
    abrirInventario()
    +dormir()
    espectar()

}
class Espectador {
    invulnerable
    espectar()
    +volar()
    -atacar()
    -comer()
    -dormir()
    -picar()
    -craftear()
}

class Hostil {
    daño
    seQuema
    perseguir(Entidad)
    refugiarse()
    quemarse(seQuema)
}
class NoHostil {
    esAnimal
    queCome
    -atacar(Entidad)
    dropearMateriaPrima(esAnimal)
    reproducirse(esAnimal)
    huir()
}
class Neutral {
    daño
    atacado: bool
    +atacar(atacado, Entidad)
    perseguir(atacado, Entidad)
}

@enduml
```