# Rogue

Projeto POO 2022/23

Vicente Laia nº 103961

## Estratégia

Antes da implementação foi necessário pensar na estratégia a utilizar.

Neste caso, cada objeto tem a sua própria classe. No entanto, cada objeto é um *GameElement* então seria útil criar uma classe abstrata *GameElement*, em que definimos atributos e funções comuns a todos os *GameElements* de modo a evitar a repetição de código.

Definimos também uma classe abstrata *HealthObject.* Todos os objetos que têm vida derivam desta classe.

Como existem vários objetos que se comportam da mesma forma é útil definirmos Interfaces.

Foram definidas cinco interfaces:

- Enemy
- Obstacle
- Item
- Consumable (HealingPotion)
- Collectable (Thief)

A classe Engine é um singleton de modo a poder comunicar com todas as outras classes. Esta classe vai conter as diversas listas de objetos, cada lista é do tipo Interface exceto a List<Door>.

Os inimigos implementam a interface Obstacle já que o herói não os consegue atravessar e implementam a interface Enemy já que atacam o herói.

Como o herói é a personagem principal e não pertence a nenhuma lista não faz sentido este implementar nenhuma interface.

## Implementação

O primeiro passo foi a criação uma classe estática MapReader.

Esta classe permite a leitura dos ficheiros “roomN.txt” e, através de uma função estática definida no GameElement, cria e adiciona os elementos lidos à respetiva lista.

O jogo começa pela função start() que desenha os objetos, o herói e cria o player. Cada vez que o herói muda de sala esta função é chamada.

Assim que o jogador prime uma tecla é tomada uma decisão na função update(). Se for uma tecla de movimento o Engine verifica se existe alguma colisão. Se for um número o herói larga ou usa algum item. Independentemente da decisão ainda é necessário mover ou remover os inimigos e dar update nos items ou vida do herói.

# Diagrama UML