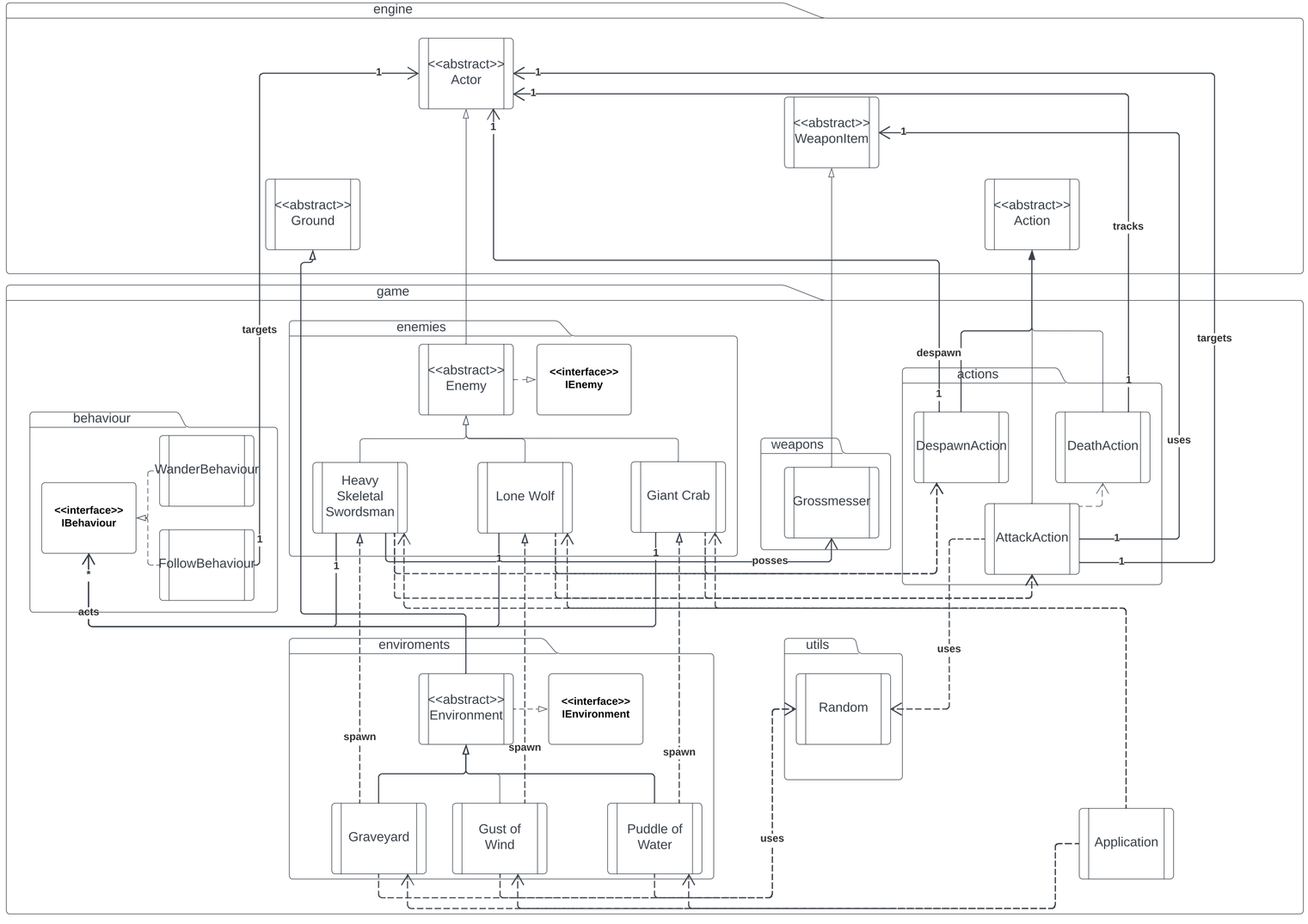
REQ1: Environments & Enemies



Design Rationale for REQ1

During the designing of req1, serveral concept and theorties is taken into account to ensure a good OOP design, and does not change anything in engine package.
Inside the game package of this uml, contains a total of 6 package and 17 concrete classes , 2 abstract class, 3 interface.

classes are designed to follow single resposibility princible, for example:
In enemy package, Enemy classe is reponsible for enemies general attributes. HeavySkeletalSwordsman, LoneWolf, and GiantCrab classes are responsible for their type of enemy attributes. same goes with other classes.

Also designed to follow interface segregation principle, for example:
Interface is implemented for enemy(HeavySkeletalSwordsman, LoneWolf, and GiantCrab classes), enviroment(Graveyard, GustOfWind, and PuddleOfWater classes), and behaviour, because those class have similarities. this improve modularity and maintainability, and easier for creating new child class by implementing interfaces.

designed to follow Liskov Substitution Principle, for example:
enemy(HeavySkeletalSwordsman, LoneWolf, and GiantCrab classes) and enviroment(Graveyard, GustOfWind, and PuddleOfWater classes), those child class can replace their parent class and still keep the program working.

designed to follow Dependency Inversion Principle, for example:
enemy(HeavySkeletalSwordsman, LoneWolf, and GiantCrab classes) is depending on IBehaviour interface instead of a concrete class, which reduces coupling between enemies and other concrete class and making it more flexible and adaptable to changes, extends. same goes with actions.

designed to follow open and close Principle, for example:
WeaponItem class attack a single actor, while Grossmesser are overrided to attack multiple target, which makes WeaponItem class open for extension but closed for modification. this way the code is easier to maintain and test.

Inheritance is used in enemies, enviroments, weapons and more, which reduce code duplication and improve reusablitity of the code base. Making it easier to add new enemies, enviroments, actions and weapons in the future.

Multiple package is created to improve cohesion and coupling, making it easier to manage, update and understand.

PileOfBones is not implemented as a class because I believe it is better to implemented within the HeavySkeletalSwordsman class since PileOfBones doesn't require much code and seems easier to implenment into the HeavySkeletalSwordsman class.

In conclusion, the design ensure the program to be well structured, easyy to maintain, scalable and a good OOP design.