Requirement 5: More Enemies

The diagram represents an object-oriented design for requirement 5 and displays the necessary classes and their relationships with each other in order to fulfil the needs of the overall system. The features requested by requirement 5 are additional to what has been defined in requirement 1, hence both diagrams will be similar, and anything mentioned within the rationale of requirement 1 will also hold true here.

After receiving feedback from our initial Assignment 1 designs, we have made significant changes to the system design, which have helped us reduce the number of unnecessary classes and decrease the overall coupling and dependancy between classes. These changes have also improved the overall maintainability and extensibility of the system. We have thoroughly tested the updated design and are confident in its ability to meet the requirements of the project.

As mentioned prior in the revised requirement 1 rationale, both the mortal abstract class and skilled interface have been removed for violating open-closed and single responsibility principles as they involved multiple downcast operations in the codebase as well as resulting in high coupling. In requirement 5, we add more enemies and more weapons, which provide us the opportunity to create more abstractions in order to reduce repeated code. For example, the Heavy Skeletal Swordsman and Skeleton Bandit both need to spawn a pile of bones on death as well as perform a spinning attack; hence, we can create an abstract class called Skeletal Enemy that implements AoECapable with an overridden play turn method to spawn a pile of bones as well. Similar to giant dogs, crayfish, and crabs, they are all capable of performing AoE actions; hence, we can group them under an abstract class called GiantEnemy that implements AoECapable, allowing us to avoid repeating the same code repeatedly. Not only does this design enforce the DRY principle, but it also follows the open and closed principles, allowing our system to be easily extended with new giant and skeletal enemies. In the future, if we were to add a new skeletal enemy, we would only need to create a constructor in its main class, as the majority of what defines a skeletal enemy is already present in the SkeletalEnemy abstract class, allowing for much easier maintenance and extension.

Another major change that has been implemented is the addition of more weapons, specifically a new class of weapons known as giant weapons. Since we have simplified our design to ensure only weapons are responsible for the creation of skills, we needed to create an additional set of weapons to give giant enemies in order to enable them to use their slam skill. This has been done through the use of the GiantWeapon abstract class, which GiantDogHead, GiantCrabClaw, and GiantCrayfishPincer inherit from. These weapons have the exact same stats as their associated enemies intrinsic weapons; however, they are only used as factories for skills and are generated only when needed, such as during an attack area action. This approach satisfies the DRY, SRP, and open and closed principles, as adding more giant weapons is trivial since they can inherit directly from the provided abstract class. The downside of this approach is that we must maintain a specific giant weapon for each giant enemy, which creates more technical debt and increases coupling between the two classes slightly.