Design Rationale for Requirement 5

The diagram represents an object-oriented system for enemies, weapons, and their related special skills in the game.

The three environments Puddle of Water, Graveyard, and Gust of Wind have been separated into West and East variants, each being their own class.

Each new enemy, Giant Dog, Skeletal Bandit, and Giant Crayfish have been implemented as the GiantDog, SkeletalBandit, and GiantCrayfish classes. They extend abstract classes of their respective base enemy type, CanineEnemy, SkeletalEnemy, and CrustaceanEnemy respectively, which themselves extend the Enemy abstract class.

The SkeletalEnemy abstract class implements the PileOfBonesAble interface, which enables the enemy to use the PileOfBones special ability. This allows all enemies which extend the SkeletalEnemy abstract class to use the ability.

The Giant Crab, Giant Crayfish, and Giant Dog enemies implement the SlamAttackAble interface, which allows them to use the slam special attack.

The Skeletal Bandit enemy wields the Scimitar weapon, which has been implemented as its own class Scimitar extending the WeaponItem abstract class. Scimitar and Grossmesser both implement the SpinAttackAble interface, allowing their wielders to use the spin special attack, implemented as the SpinAttack class, extending the AttackAction class.

The decision to implement abstract classes for enemy base types such as CanineEnemy was made keeping in mind that enemies do not attack other enemies of the same “type”. This also adheres to the Open-Closed Principle, by allowing new enemies of the same type to be created without modifying existing code. Subsequently, it adheres to the DRY principle, by reusing code through inheritance.

The decision to implement the ability to use these special attacks and abilities as interfaces was to allow enemies(classes) which were not necessarily similar to inherit their usage. For example, Giant Dog and Giant Crab both are able to use the slam special attack, but do not share the base enemy type (CanineEnemy vs CrustaceanEnemy). Therefore, the only way for both of them to inherit the slam attack functionality is through an interface (since multi-inheritance is not possible in Java). This adheres to the Dependency Inversion Principle, by not having concrete classes depend on each other, and using abstractions through an interface.

A slight disadvantage to the approach of creating and implementing the interfaces is that more code has to be written. They can also cause issues if implemented poorly, forcing subclasses to implement unneeded methods, if the Interface Segregation Principle is not followed. Additionally, the usage of abstraction in general means the code and logic is split across multiple classes rather than in one place, making it more difficult to understand.

Changes in Assignment 2

The functionality of spawning different enemies based on which side of the map the environment is in has been implemented in the class itself, through a method. This means both east and west versions of the environments are stored in the same class. It also allows future variants (such as north or south) of the environments to be easily implemented.

The Scimitar weapon extends the CurvedSword abstract class, allowing its user to use the AreaAttackAction similar to the Grossmesser weapon. This allows new weapons which can use AreaAttackAction to be easily implemented in the future. It also adheres to the DRY principle of reusing code, through inheritance. A disadvantage of this implementation is the code for the special actions are stored across multiple classes, making it harder to understand.

The enemies which turn into a Pile of Bones on death extend the SkeletalEnemy abstract class, which also prevents them from attacking each other. This allows future variants of skeletal enemies to be easily implemented. It also adheres to the DRY principle of reusing code, through inheritance. A disadvantage of this implementation is the code and logic for the special actions are stored across multiple classes, making it harder to understand.

The enemies which can use the slam area of attack action have the ENEMY_GIANT capability, created from the Status enum. The decision to store this as a capability rather than in the enemy types similarly to SkeletalEnemy, was because enemies with different types(Giant Crab/Crayfish vs Giant Dog) could use the slam attack, and attack each other. A disadvantage of this implementation is the code and logic for the slam attack is stored across multiple classes, making it harder to understand.