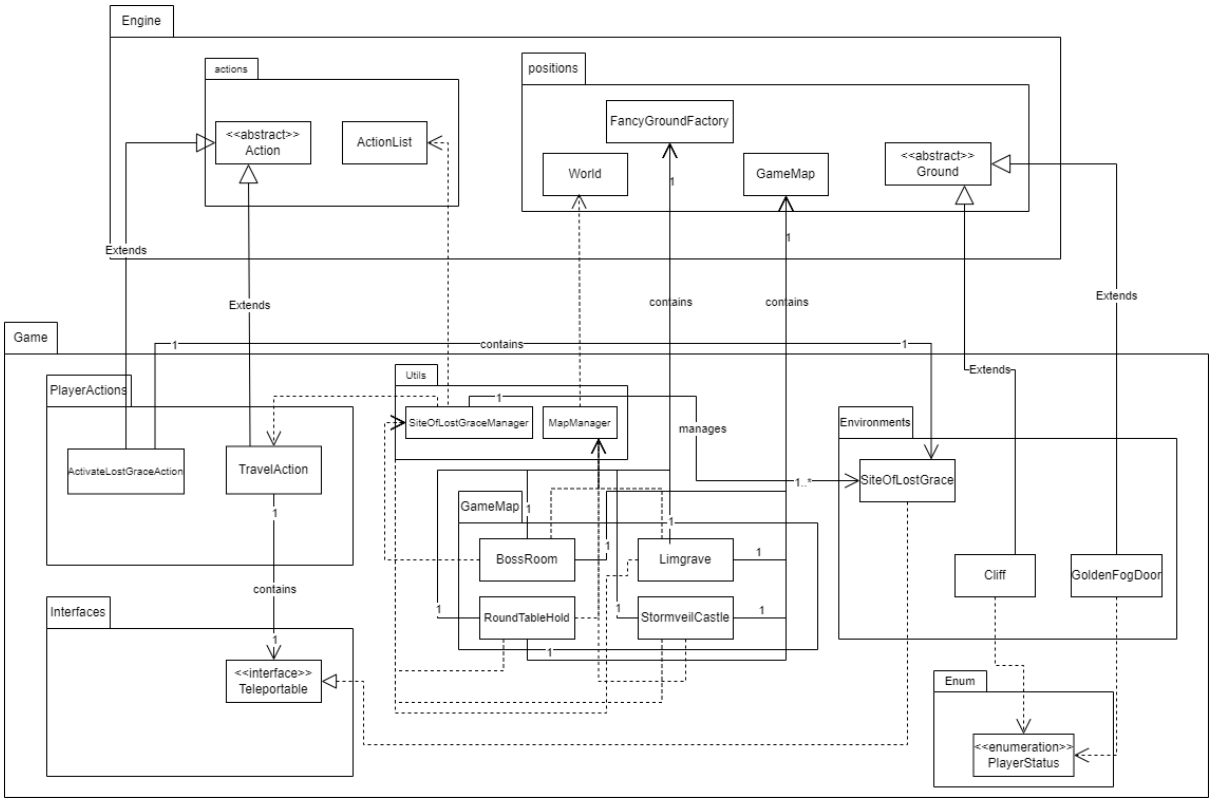
Assignment 3 Design Rationale

FIT2099 Lab 4 Group 2

Members:

Keith Ong Guo Er	**32287089**
Sam Zachery Chee	**32805195**
Yap Wing Joon	**31862527**

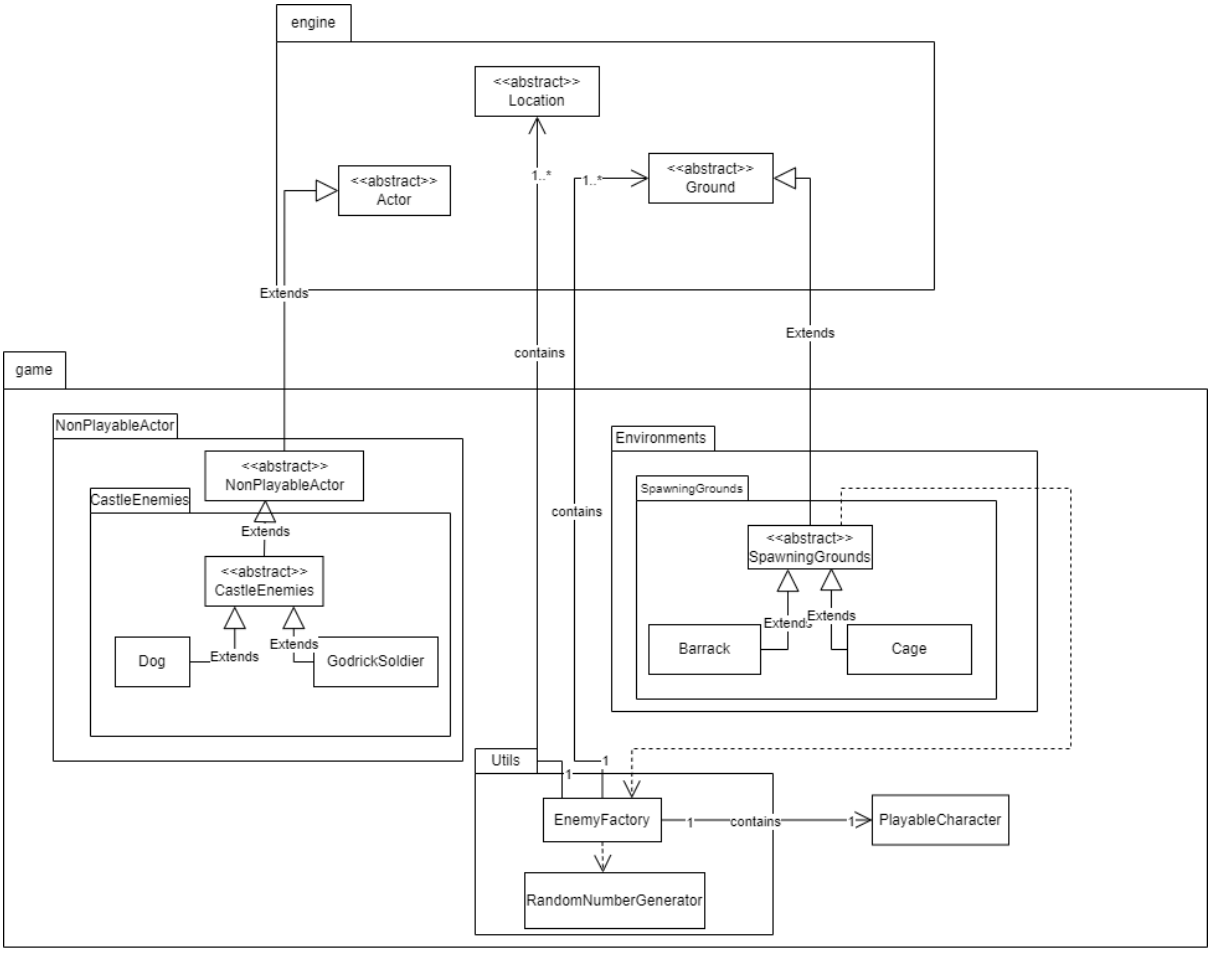
Requirement 1 – Travelling between Maps



In requirement 1, we first added two new classes which are Cliff and GoldenFogDoor classes which extend the abstract Ground class. Hence, it can inherit all attributes and methods of the Ground class in the engine code. These child classes of ground class depend on the PlayerStatus enum class as it contains constants of player statuses to determine if a player can interact with those two classes. Next, four new classes of GameMap are added. These classes are all associated to FancyGroundFactory and GameMap as they know about the two classes in the engine code and have attributes of their variable types. We then created a new MapManager class to manage every game map classes which in turn reduces multiple dependencies and adheres to the DRY principle as all four game map classes have dependencies on the World class which prevents redundancy. The same can be said for SiteOfLostGraceManager class which will manage all Site of Lost Grace which explains the association between that class with SiteOfLostGrace class which contains an arraylist of all Site of Lost Grace. The ActivateLostGraceAction is created to describe the action of activating the site of lost grace before the player rests on it. It has an

association with the SiteOfLostGrace class as it contains attributes of this class type. Lastly, we also created a TravelAction class which implements the interface Teleportable. This way, we can also prevent multi-level inheritance as we only need to implement the methods of the interface.

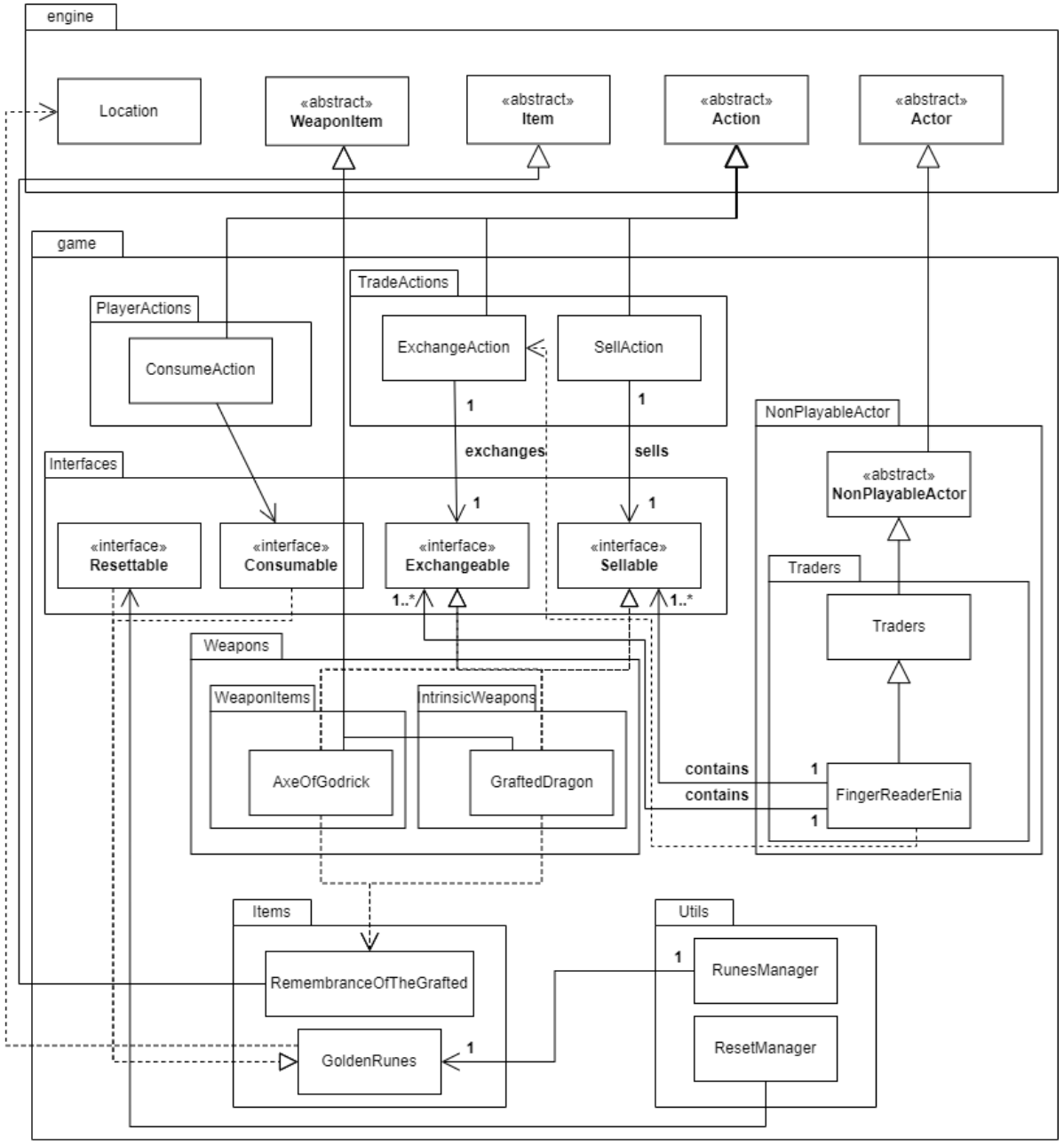
Requirement 2 – Inhabitant of the Stormveil Castle



In requirement 2, the design concept is the same as how we implemented it in requirement 5 of Assignment 2. The only changes that were made includes refactoring the class names to better suit our design. For instance, Enemies abstract class has been refactored to NonPlayableActor which is a better representation of the actors categorically related to that class. As this requirement only involves adding new grounds and enemies, we first created an abstract class SpawningGrounds to manage the tick of two new child ground classes Barrack and Cage as they both have the same tick method. Hence, it adheres to the DRY principle. We also added two new enemy classes Dog and GodrickSoldier which inherits a new abstract CastleEnemies class which has all attributes and methods relevant for castle enemies to inherit. This way, we would only need to add new methods and functions specific to the CastleEnemies class which corresponds to the Open-Closed principle. Enemies

outside of the CastleEnemies class would be unable to inherit their methods and attributes which improves code maintainability.

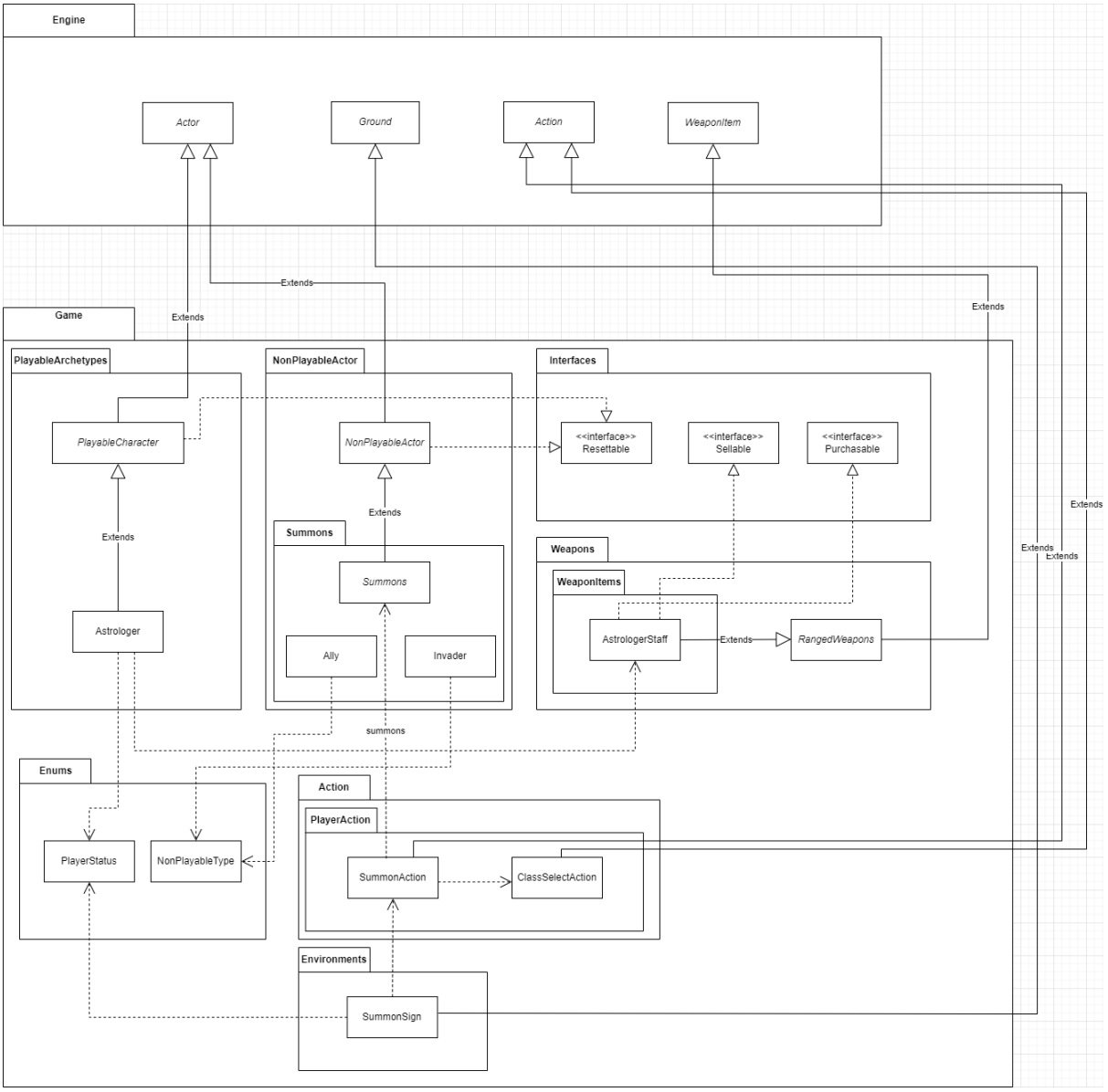
Requirement 3 – Godrick the Grafted



In requirement 3, two new weapons, **AxeOfGodrick** and **GraftedDragon,** are introduced which extend WeaponItem. These two weapons implement the **Exchangeable** and Sellable interfaces since they can only be sold or exchanged for **RemembranceOfTheGrafted** which extends the Item class and is used by both weapons for when the exchange occurs. **FingerReaderEnia** extends the Trader class and uses the **ExchangeAction** and contains both Exchangeable and Sellable interfaces. **GoldenRunes,** similar to Runes, extends the Item class, is associated with RunesManager and implements the Resettable interface. Since it can be consumed, it also implements the Consumable interface. Additionally since GoldenRunes is scattered across the map, it uses the Location class to be placed randomly.

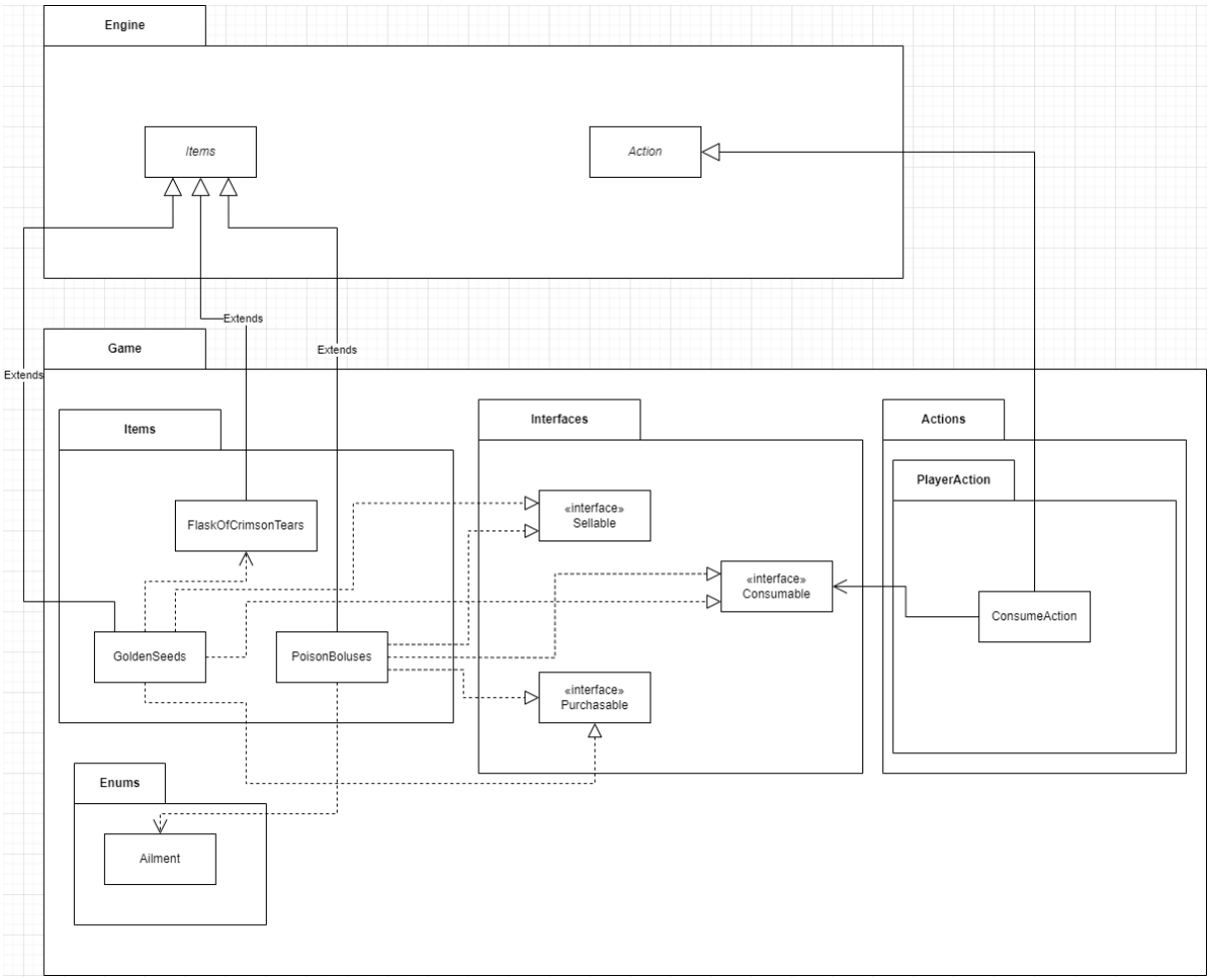
Lastly, GoldenRunes is set true for portable and therefore can be picked up and dropped by the player.

Requirement 4 – A Guest from Another Realm



In requirement 4, we added an Astrologer class that extends the PlayableCharacter abstract class. For the Ally and Invader classes, we created a Summon abstract class such that both Ally and Invader extend this class. This is to improve code extendibility as new guests that can be summoned can simply extend the Summon abstract class to inherit all the features of a summon. We also added a SummonSign class that extends the Ground abstract class in the engine. We added a SummonAction and ClassSelectAction class that extends the Action abstract class in the engine as well. The ClassSelectAction class facilitates the SummonAction class in summoning the guests Ally and Invade. The Summon Sign class allows the player to interact and summon guests.

Requirement 5 – Creative Requirement



In requirement 5, we added two new consumables that implements the consumable interface, namely Golden Seeds and Poison Boluses. Since they can be purchased and sold, they also implement the Sellable and Puchasable interfaces. This follows the Single Responsibility Principle as each interface only has a singular purpose. Poison Boluses depends on a newly created Enum class called Ailment, while Golden Seeds depends on the FlaskOfCrimsonTear class as it increases its use count.