Design Rationale

Name: Koe Rui En

ID: 32839677

REQ 1: The Player

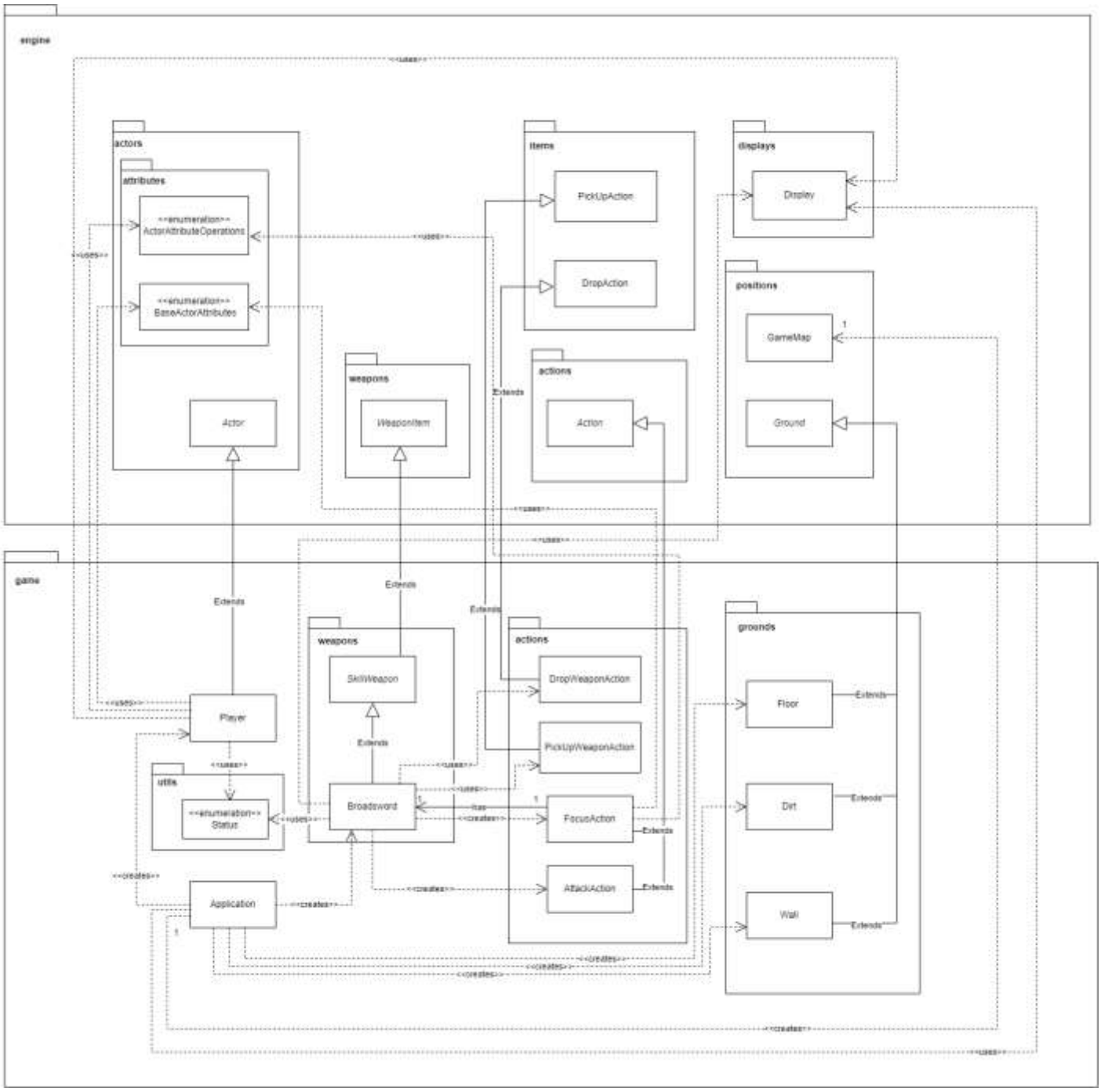

Figure 1: UML class diagram of REQ1 – The Player

The provided diagram shows the entire system implementation for Requirement 1, The Player. The primary goal of this design is that to create the player, “The Abandoned Village” game map, a weapon which have special skill and the building on the game up which is made up dirt, floor and surrounded by wall. In this requirement, several new classes are introduced to the existing system, which is Broadsword class, Skill Weapon class, DropWeaponAction, PickUpWeaponAction class and FocusAction class. New classes will interact with the existing classes in game package as well as classes from the engine.

The existing system includes a Player class, which extends the abstract Actor class from the engine. This Player class will be further extended to incorporate additional functionality. In this requirement, the Player class has dependency with Display class to display their details at every turn of the game. This Player class also uses BaseActorAttributes enum class and ActorAttributeOperations enum class to modify their attribute, which is stamina at every turn of game in the current implementation. The The Player class will be assigned a status to signify their status, thus resulting in dependency relationship towards this Status enum class. It is noted that status assigned to the Player class will be a generalised and flexible status. The advantage of having Status enum class is to differentiate the Actor, so classes are introduced in the later implementation that involves any status checking can use it before performing certain tasks or actions, and this may avoid violation of Open/Closed Principle (OCP).

Furthermore, we also decided to have SkillWeapon class as an abstract class and extends the WeaponItem abstract class from the engine. The reason we have this design decision is that not all the weapon items will have special skill. This approach can help us to differentiate the weapon item type. Besides, any new implementations made in this SkillWeapon abstract class will apply to other skilled weapon which means prevention of Don’t Repeat Yourself (DRY) as we avoid repeating codes in every weapon class. As we need to create a weapon for the player to attack the enemy, The Broadsword class is introduced. Since the Broadsword has special skill, Focus, so this Broadsword class extends the abstract SkillWeapon class. This Broadsword class also has a dependency relationship with Status enum class, so its status can also be used in the future implementation and if the Player who owns it will also have extra status, so we can identify the Actor more easily that have different types of status in any new implementation of the game. We also introduced the FocusAction class since this FocusAction class is a special skill of the Broadsword and extend the Action abstract class from the game engine. Thus, the Broadsword class has a dependency to the FocusAction class. The FocusAction class has an association relationship with the Broadsword class as it need to modify the damage multiplier and hit rate of instance of Broadsword class. Since the player also use the Broadsword to attack other actors, so this Broadsword class creates (dependency relationship) with the AttackAction class that extends from the abstract Action class.

We also introduced new 2 classes, PickUpWeaponAction class and DropWeaponAction class. PickUpWeaponAction class extend from PickUpAction class from the engine, while the DropWeaponAction extends from DropAction class from the engine. The reason we proposed these classes as mentioned in the requirement, the status of skilled weapon after being dropped and picked up again. By doing so, we can reuse these classes for the skilled weapons regardless skill types whenever those weapons are dropped or picked up again by the Actor. Besides, separating the implementations of the PickUpWeaponAction and DropWeaponAction of the skilled weapon to different classes, our design decision has aligned with the Single Responsibility Principle (SRP). Thus, the Broadsword class also has a dependency with these PickUpWeaponAction class and DropWeaponAction class since the Broadsword is a skilled weapon.

Moreover, the Application class creates (dependency relationship) Floor class, Dirt class and Wall class as well as GameMap class and Player class. This class also has dependency on Display class to display the message on the console menu.

On the other hand, our design may increase the complexity of the system as many new classes are introduced and increase the difficulty for the developers if any errors are occurring during development.