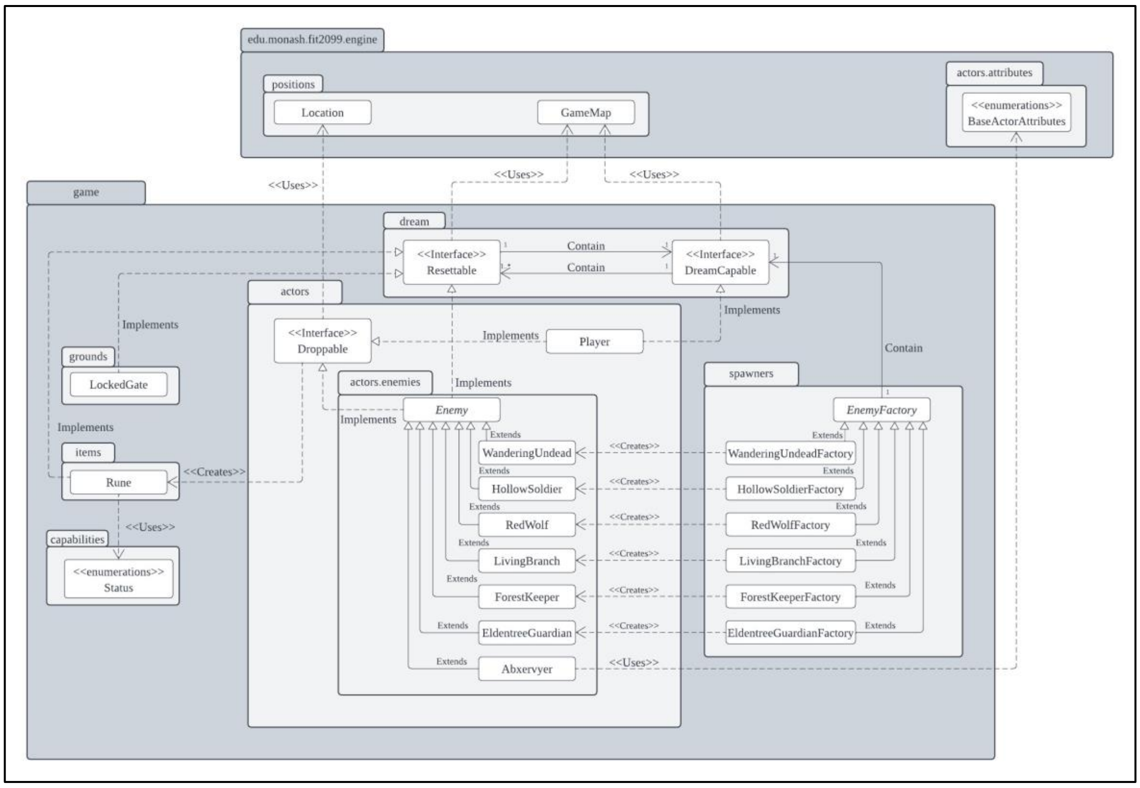
FIT2099 Assignment 3 Design Rationale

Requirement 5:



- Similar to requirement 5 in assignment 2, for this requirement, two new interfaces, Resettable and DreamCapable, are introduced. DreamCapable interface is implemented by the player. This interface mainly focuses on managing the respawn functionality of the player (i.e., how is respawn function works and what happens after the player is respawned). Meanwhile, Resettable interface is implemented by those game entity that are affected after the player is respawned. This interface only consists of one method which requires those that implement this interface to specify the actual details when the player is respawned. By having these two interfaces, it can help to follow ISP as each interface consists of methods that are applicable to all the classes that implement it. This also follows SRP as each of the interfaces only focuses on one responsibility. This helps the developers to maintain the code easily. Besides, having the two interfaces also allows us to follow OCP as the code can be easily extended without needing to modify the existing code. For example, if the weapons on the ground will also be removed when the player is respawned, we can simply let the weapon implements the Reettable interface to obtain this functionality.

- All the Resettable contain a DreamCapable as they need to subscribe to the DreamCapable. The spawners (enemy factories) also contain a DreamCapable so they can pass it to the enemy through the enemy's constructor. Meanwhile, the DreamCapable will consist of a list of Resettables so that it can inform them to reset when the player is respawned.

- The reset method in the Resettable interface need to use the GameMap in order to remove all the enemy from the map. Meanwhile, for the rune, the GameMap cannot be used as we need to track the actual location of the rune and the only way to do this is in the tick method. Hence, when the reset method is called for the rune (or all the item type), a reset status is added to the rune. The tick method will then check whether it has the reset status every turn, if the reset status is present, the rune will be removed from the ground.

- Since the player will drop runes when he is respawned, the player will also implement the Droppable interface. The developer will only need to specify the actual details of the drop method without needing to modify the existing code. This proof that using a Droppable interface is extensible and follows OCP.

Pros:

- Achieve OCP as it is easier to extend without needing to modify the existing code by having the DreamCapable and Resettable interface.
- Each interface only focuses on one responsibility (SRP) which helps to make the code easier to maintain.

Cons:

- Each of the resettables will contain a DreamCapable and the DreamCapable also consists of a list of resettables. This may cause tight coupling and cause the code to be harder to maintain.

Future extensions support:

- Implements Resettable for game entity that is affected when the player is respawned.
- For new enemy that implements Resettable, directly use the GameMap to remove the enemy from the map in the reset method.
- For new items that implement Resettable, add a reset status to the item in the reset method. Then, check for the status in the tick method for items located on the ground to decide whether to remove the item from the ground.
- If the player may drop other items in the future after respawning, implement the specific details in the drop method from the Droppable interface in the player class.