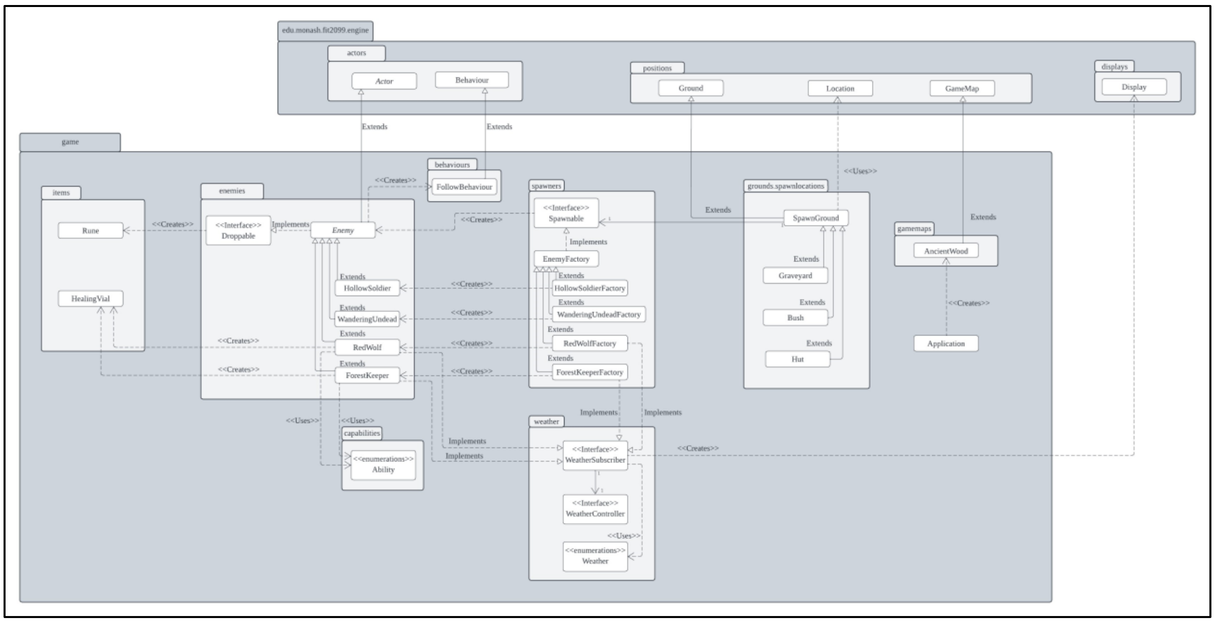
FIT2099 Assignment 2 Design Rationale

Requirement 1:



- In this requirement, we had changed the spawning mechanism which we kind of use instanceof method to spawn a valid enemy at a valid map in the last assignment. In this assignment, we introduce a spawnable interface which contains a spawn enemy method in it. Then, an abstract EnemyFactory class implements the Spawnable interface. This EnemyFactory class can be extended to form a concrete enemy factory class to form a particular type of enemy. This helps to avoid code repetition (DRY). The EnemyFactory class acts like a spawner which basically will manage the responsibility to spawn an enemy on the ground.

- We also introduced a SpawnGorund abstract class which extends from the ground. This SpawnGorund class is the base class for those ground that can spawn enemies. In the SpawnGorund, it will contain a Spawnable instance which will help to determine the type of enemy to be spawned. This follows SRP principle as now every class only focuses on their one and only responsibility and hence the code will be easier to manage and maintain in the future. This also follows the OCP principles as for a new enemy introduced, we can simply add a new enemyfactory class. Then, for different ground instances of the same type, we can also let them spawn different enemies by passing the respective enemyfactory to the constructor.

- The red wolf and forest keeper also implements droppable interface as they may drop some items after being defeated by the player. This helps to ensure that the drop implementation is done for every concrete class.

- Since the requirement stated that the enemies in ancient wood will have follow behaviour, all the enemies from the ancient wood is added with a follow ability in the constructor. The following behaviour will be added to the enemy when the enemy has a follow ability and it detects an actor nearby, which has the hostile to enemy status. By doing so, it follows OCP principles as we can simply let an enemy have the follow behaviour by adding the ability to them.

Pros:

- Follows SRP. A spawner is introduced which serves for the purpose of spawning an enemy. If compared to assignment 1, the code now will be much easier to maintain as cohesion is achieved.
- Easier to extend without needing to modify the existing code.

Cons:

- There will be an extra association between ground and spawnable instance, which may cause a tighter coupling.

Future extensions support:

- If a new enemy is introduced, simply create a new spawner class for the enemy.
- A spawning ground can now choose to spawn different enemies by passing the respective enemyfactory instances into the ground constructor.
- For a new enemy or existing enemies which decide to have a follow ability, simply add the follow ability to the enemy's capability set, and it will then start to follow a player when a nearby player is detected.