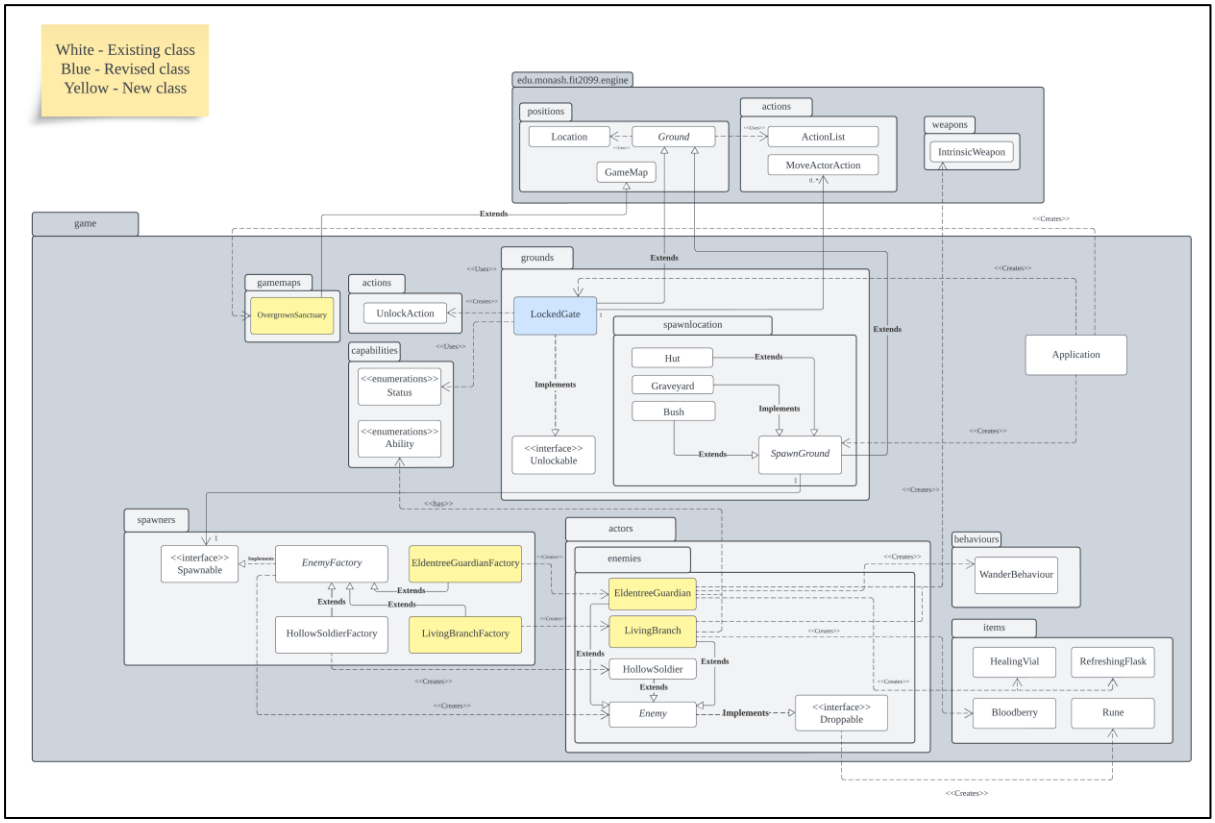
FIT2099 Assignment 3 Design Rationale

Requirement 1:



- For this requirement, A new map is created “The Overgrown Sanctuary”, the implementation was done with no problems since the functionality is exactly the same as the previous assignments.

- The implementation of locked gate is modified slightly. A second constructor is introduced in the LockedGate class. The process itself is pretty straightforward because most of the functionality of the locked gate has been done in the previous assignment. The second constructor allows us to have a locked gate with multiple destinations, this is done by accepting a HashMap/Map of type <String, MoveActorAction> rather than just one single Action. This still adheres to the SRP or Single-Responsibility Principle, since the gate’s functionality stays the same. It also strongly follows Open/Closed Principle or OCP where we can add/remove the gate’s destination without modifying the LockedGate class at all.

- Other than the LockedGate class, some brand new enemies are also introduced. These enemies can be spawned through the Huts and Bushes from assignment 2. Because the design of Assignment 2 strongly follows OCP, the implementation for the new enemies were straightforward, and seamlessly integrated into the existing game. The

utilization of EnemyFactory and Enemy abstract class were crucial in achieving this. The two new enemy classes: `EldentreeGuardian` and `LivingBranch` is extending to the Enemy abstract class , and by doing so, they inherited the methods of the Enemy class. This means that if there's a need to customize specific behaviors or attributes for these new enemies, adjustments can be made directly within their respective classes without having to modify the foundational Enemy abstract class. This approach not only promotes code reusability but also ensures that each enemy type can be individually tailored as needed. The two new factory classes: EldentreeGuardianFactory and LivingBranchFactory is extending to the EnemyFactory abstract class. Huts and Bushes simply relied on the new factory classes to generate the appropriate enemy. This level of abstraction and separation of concerns ensured that we didn't need to modify the original code of Huts and Bushes.

Pros:

- Adherence to Design Principles: The new introduced classes/implementations consistently follow SOLID principles, like OCP and SRP, making the code more maintainable and adaptable to future changes
- The second constructor introduced in the LockedGate class makes the game environment more dynamic, and flexible.

Cons:

- Redundant Constructor in the LockedGate class, by having two constructors, it can be seen as a little bit redundant, since even if we only have the brand new (second) constructor, a gate with one travel action is also possible.

Future extensions for support:

- The new enemies and factories were simply implemented by only extending the classes we had before, which means we have a strong foundation for some of the SOLID principles such as the OCP. In the future, if we have even more enemies, spawners, and factories, we can just simply extend without modifying any of the old class.