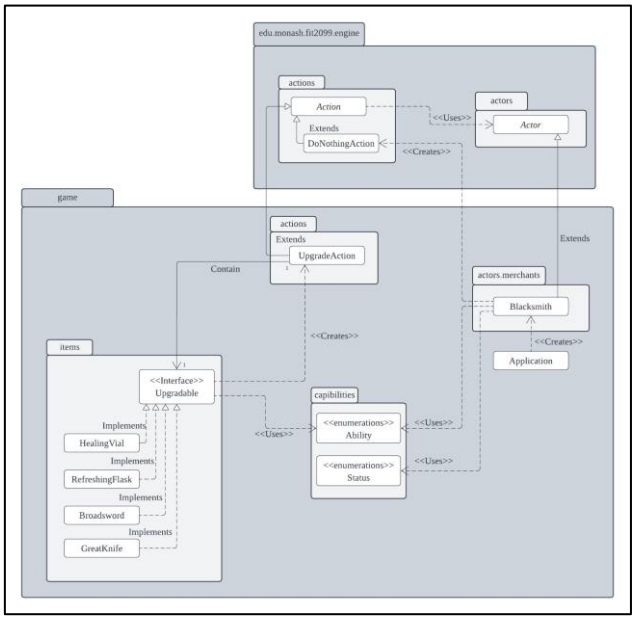
FIT2099 Assignment 3 Design Rationale

Requirement 2:



- In this requirement, a Blacksmith class is created to act as a blueprint for all blacksmiths in this game. The Blacksmith class is extended from the Actor class to inherit the methods from the Actor class (DRY). In this design, we don't have an abstract parent Blacksmith class, which is different with the merchant design in assignment 2. The reason is that the merchant is said to have different selling prices depending on which merchant is interacting with the player. However, for the blacksmith, we don't have this feature, and our assumption is that all the price offered by all the blacksmiths in the game is consistent and any weapons or items can be upgraded by all the blacksmiths in the game if the player have the weapons or items. This assumption follows the logic of most of the roguelike game. Hence, all the blacksmiths in this game should be instantiated from only this class. By following this design, this helps to reduce unnecessary complexity for the code as having multiple inheritance may be confusing and hard to understand for the developers. So, multiple inheritance should be avoided if we have enough reason to reject the use of it.

- In this requirement, we introduce a new interface, Upgradable. This interface is only implemented by those items or weapons that can be upgraded by the blacksmith. By introducing an interface, it helps to achieve the OCP as if there is a new items or weapons that can be upgraded, the new items or weapons can have this feature by implementing the Upgradable interface. The Upgradable interface only consists of two methods which are highly related to the upgrade feature, this follows ISP as every interface should only have methods that are applicable to all child class. Besides, by having these two methods in the interface, it enforces the child class to implement the complete code for the method with more specific details.

Pros:

- Follows OCP. Every new item or weapon that is upgradable can achieve this feature by simply implementing the Upgradable interface and complete the two methods in it.
- Easier to extend without needing to modify the existing code.
- Reduce code complexity by removing unnecessary multiple inheritance for blacksmith class (this is a concrete class compared to the merchant in assignment 2 which is an abstract class).

Cons:

- If every blacksmith in the game has different available upgradable item or have different upgrading cost, then the current blacksmith class should be modify to become an abstract class and the concrete blacksmith class should extends from this abstract class.

Future extensions support:

- All the blacksmiths in this game can be directly instantiated from the blacksmith class. All items or weapons that are upgradable can be upgraded by all the blacksmiths in the game.
- New items or weapons that can be upgraded by the blacksmith can achieve this feature by simply implementing the Upgradable interface. More specific details regarding the upgrade can be done in the upgrade method from the Upgradable interface.