# FIT2099 Assignment 2 Design Rationale – Requirement 2

## Design Rationale

### Requirement 2

*Note: This Design Rationale is short in comparison to REQ1 as the underlying design for GoldenBeetle and GoldenEgg is discussed in REQ1 – its justifications, advantages, and disadvantages.*

**Implementation of GoldenBeetle and GoldenEgg**

GoldenBeetle and GoldenEgg are simply extensions of the NonPlayableCharacter and Egg classes respectively – adhering to OCP as no modification of the existing code was required. Both of these also implement the Edible interface as they can both be consumed by the Player, and implement their own eat methods for their unique behaviour - for example, in GoldenBeetle.eat, the actor will gain 15 health and 1000 runes through the Capability system in the engine. The GoldenBeetle also passes the amount of turns it takes to hatch (5) for its ReproduceBehaviour, to meet this requirement.

For the GoldenEgg, it uses the SurroundingStatusHatchingStrategy – and checks for nearby cursed entities – this HatchingStrategy accepts a status enum and checks surrounding entities (both ground and actors) – so it can be extended to any other status, such as grace – adhering to DRY and OCP.

**Implementation of Runes**

A new Enumeration – GameActorAttributes – is created, storing RUNE which can be added to the capabilities storage of the Player. This required minimal modification of the existing code and uses the already existing methods in the game engine to allow for interaction and storage of the runes – hence adhering to OCP.

**Implementation of Following**

The FollowBehaviour class, implementing Behaviour, integrates with the engine code to allow for an NPC to follow a FOLLOWABLE actor, and iterates over the entire map to find this actor – once found, it will store the actor and continue to follow by getting the closest path to its position until the actor dies – then finding a new actor to follow. This meets all the requirements and adheres to SRP (only implements logic for following), and OCP (no modification of engine code required to implement). If actors need to be marked as followable, they can simply add FOLLOWABLE from the ActorStatus enum.

**Disadvantages and Justification**

- The following behaviour has tight coupling to the FOLLOWABLE enum – a potential better implementation would be to have actors implement a Followable interface, however this could violate ISP as there is no current unique following behaviour, and hence could also be speculative generality – so the current implementation is preferred – also avoids downcasting and instanceof.