Assignment 3 Design Rationale: CL_AppliedSession10_Group1

[Survival Mode] REQ4: Refactorio, Connascence's largest moon

The goal of this requirement is to modify the existing Inheritree implementation to accommodate multiple phases of growth and fruit spawning behaviour while practising good design principles and best practises.

In the original Inheritree implementation, adding dormant Inheritree implementations required unnecessary methods since the spawn logic was hardcoded within each class. Hence, we decided to decouple the spawn and growth logic into their respective Behaviour class. Similar to the actorBehaviour class, GroundBehaviour was created such that new or old behaviours can be added and removed to various ground classes without any additional code duplication, which means the DRY principle is adhered. Moreover, the OCP principle is followed since we don't need to modify multiple classes when a behaviour is changed, making the behaviour system more reliable and extendable. For the SpawnFruitGroundBehaviour, fruit factories that were already implemented before was injected into the behaviour such that new types of fruit that is spawned can be easily changed or added for only one specific behaviour. For the GrowGroundBehaviour a Inheritree that represents the next growth stage is injected instead of a factory, since only one Inheritree is needed for each growth phase.

However, the cons of this implementation is that Inheritree's always follow a certain growth path, this is because the GrowthBehaviour for a Inheritree sets the next growth phase in the class itself. If this functionality is required in the future however, the Inheritree tree that represents the next growth stage can be injected either using setter or constructor injection instead into the different Inheritree classes.

In conclusion, The new GroundBehaviour system that is introduced into the Inheritree makes our code much more extendable for future growth phases and behaviours, and behaviour code can be reused for any types of grounds when required.