# Design Rationale for Requirement 1: Let it Grow!

## Explanation of system

This system controls the trees and their actions. We have 3 types of trees: sprouts, saplings, and matures. Their core action is growing: the sprout is able to grow into a sapling after 10 turns, and the saplings are able to grow into matures after 10 turns. The matures also spread a new sprout every 5 turns, but also has a 20% chance to wither and die each turn, turning into dirt. Additionally, the sprouts are able to spawn goombas at a 10% chance each turn, the saplings can drop coins at 10% each turn, and the matures are able to spawn koopas at 15% chance each turn.

## Choice

Making Tree an abstract class

## Justification

This enables the efficient creation of multiple different sub-classes of tree, namely the sprout, sapling and mature trees. Their common/shared attributes and functionality can be inherited from the base Tree class, which supports the Don’t Repeat Yourself (DRY) principle and reduces redundancy.

## Choice

Addition of the Sprout, Sapling and Mature sub-classes of Tree

## Justification

These new sub-types of trees will have different functionality that distinguish them from each other. Their shared attributes and functionality will be inherited from the parent class Tree.

## Choice

Using an interface GrowCapable to provide the functionality for certain trees to grow.

## Justification

This implements the DRY principle, as by using an interface we are reducing redundancy of defining the similar method for each class. Instead, they can each have their own implementation of the single method under the same function name. It also supports the Open-Closed Principle from the SOLID principles of object-oriented design, as by using this interface to create the grow function, we are opening the system for the extension of adding new types of trees in the future that may use this method but closing the existing types of trees for modification.

## Choice

Actors will be stored in locations

## Justification

We decided that we will track the actors’ locations by moving the actor objects between different locations. This will include storing them in sub-types of locations, such as trees, dirt and floors. This will enable easy movement between locations, as well as helping provide the functionalities of stopping trees from spawning new actors when an actor is already standing on them.