Req 4 Rationale

Pros

- Each Growable tree only has access to its next stage, from the this.nextStage attribute, not all the stages. There is no single tree class that handles the growing of multiple trees. This thus follows the SRP as trees themselves are not handling the growing. Rather, the StageManager, whose responsibility is to handle the growing, does so
- The abstract classes ProduceGround and StageManager and the interface Growable help achieve OCP. If a new tree that can grow and/or produce fruit needs to be added, we can easily implement the Growable interface for them and/or extend from ProduceGround. Then, a new concrete implementation of the StageManager class can be created that accounts for new growing orders that include this new tree. Thus, no existing code needs to be changed.
- Abstraction of the ProduceGround class means only trees who produce fruit can do so, extending from the class. It also helps reduce repetition for trees that need to produce fruit, as code for finding an exit for the fruit is in its concrete method thus each child class doesn't need to implement it themselves.
- LSP is achieved with the Growable interface, as seen in the StageManager when iterating over the Growable stages and setting the next stage. This means, any object that is Growable can have its nextStage set thanks to this interface.
- ISP is also achieved because of the Growable interface. Only trees that need to grow do so, avoiding tree classes implementing methods they don't need. This is highlighted by InheritreeMature who doesn't grow and thus, doesn't implement the interface/have methods to do with growing.
- DIP is achieved with the Growable interface. The StageManager just depends on having a list of Growables, rather than specific trees. This also helps us complete a dependency injection as seen in StageManager as well; StageManager injects the nextStage for a given Growable thanks to the setNextStage method in Growable. Thus, the child implementations of StageManager help ensure that trees on different planets can have different growth cycles.

Cons

- The way the StageManager works is by iterating over the Growables and setting the next stage of the current Growable (i) to the Growable on the right (i+1). This thus assumes that the Growables are given in the correct order of growth, which means we have connascence of position. The way to get rid of this connascence of position is to abolish the StageManager classes and store the next stage of the tree within the tree class itself. However, then we can't have unique growth cycles on different planets and all trees would grow the same. Thus, to have a more interesting and flexible design, we kept the StageManager class and the connascence of position that comes with it.