Design Rationale

1. Tree and Bush are connected to Fruit even though Location has its own item attribute

Tree and bush will store fruit that it has growing on itself and the Location class will store items that are on the ground.

1. Dinosaur is an interface

Each type of dinosaur has different attributes and interacts differently with the rest of the world (e.g. how it eats fruit). It is easier to having an interface to ensure each dinosaur has the required functionality expected from each dinosaur but the exact details can be separated between each type of dinosaur.

1. Dependency between Player and Location

Our diagram includes that to show that Player will interact with the item ArrayList at Location.

1. Eat is a separate class to EatDino, FeedDino and EatFruit

This is to ensure code is not repeated between the different eat actions (DRY) and to make code easier to maintain in case the implementation of feeding and hunger levels change. Instead of changing all the classes, we can just change the Eat class.

1. Dinosaurs store an age tracker to determine whether it is a baby or not

This will minimise repeated code (DRY) as the baby dinosaur is very, very similar to the adult dinosaur.

1. Egg parent class for each type of egg

Each egg is very similar. This will minimise repeated code (DRY) and make it easier to maintain.

1. Location has an ArrayList of items at the location but Bush and Tree have their own attribute of fruit. Tree also has an attribute for dropped fruit

This is to make it easier to understand where the fruit is located as depending on where the fruit is located, different actors will interact with the fruit differently. For example, stegosaur cannot eat fruit from the tree but can eat it dropped fruit. Technically, location could store dropped fruit however, it is easier to understand the code if Tree has its own dropped fruit attribute. In order to check if a location has dropped fruit under a tree and dropped fruit was stored under Location, it would involve checking the location’s ArrayList of Items as well as checking if there is a tree at that location which is more clunky than if we were to have a dropped Fruit attribute in Tree.

1. Each item has its own location stored as an attribute

This is to make it easier to travel to an item such as to a corpse. Items do not travel as much as actors so the location can just be stored unlike for Actors.

9. Util class

A Util class is useful to store methods and constants that are shared by several classes since they can just be imported by the classes that use them. This follows the Do not Repeat Yourself (DRY) design philosophy, and if we wish to make changes to any of theses methods/constants, we only need to change them in one place rather than in many places if we were to not use a Util class.

10. copyItems in groundInterface

Since dirt can grow a bush, a ground object will be replaced by another ground object. Therefore, we need a method to copy the items over from the old ground object to the new ground object.

11.