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. This enables us to distinguish between the ground types Fruits are located in.

2. Dinosaur is an abstract class extended by all the dinosaurs

Having dinosaur as an abstract class allows all classes that extend it to share methods. This is applicable to the playTurn function since the dinosaurs all share a common priority when making a choice between finding food, breeding etc. Having dinosaur as an interface is not as effective since we don't only want dinosaurs to have the same methods/attributes – we want them to have the same methods/attributes definitions as well. This reduces the amount of repetition between the dinosaur classes (DRY) since all dinosaurs can reference their super methods.

3. 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.

4. Util Class incorporating many different functions.

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 these 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. Many of the functions used in the util class are functions that are generally very useful for multiple classes such as NearestItem and NextToGroundType. Another benefit of putting the functions in the util class is that it means any further classes or classes that we want to incorporate later on can use these functions very easily meaning expanding the project later on is much easier.

5. Vending Machine inherits from Ground

We believe Vending Machine should be treated like a Ground rather than an Item because it acts much more like a set object on the map (like Ground) rather than an individual item that can be moved and placed in the inventory. It is easier to make it a ground than an item.

6. There are classes for each type of Egg that inherit from the big Egg class rather than using the one Egg class and changing each instance's attributes to meet the requirements of each type of egg.

It is easier to create classes for each type of Egg and have predefined attributes for each type of egg. This makes it easier to understand what the properties each egg has and makes it very easy to instantiate the right type of egg in the code when dinosaurs breed.

7. Corpse is an Item rather than Ground

This makes it easier to interact with and easier for Allosaurs to eat it with having to make multiple edits to the Ground. Items sit on top of the Ground so it is easy to change with affecting other objects.

8. The argument for IncreaseEcoPoints is a String rather than an action or an Int

EcoPoints will store a hashMap to match certain actions represented by strings with an int value of how many points that action is worth. Storing it as a hashMap will make it much easier to edit the values later and also makes the function very easy to use as all it requires is a string input rather than having to pass certain objects which could make it messy.

9. Implementing MateBehaviour rather than using just MateAction and FollowBehaviour

Utilising the behaviours mean we can group together multiple different method calls that lead to Actions being returned. This makes the code easier to understand and reduces the dependencies that would have been spread to other classes that called the classes separately.

10. Buy item action takes in parameter the item to buy selected on the menu

Options on the menu typically show different action classes that are possible for players to take. By taking in a parameter to buy an item from a vending machine, this removes the need to create an action for returning each possible vending machine item.

11. addFruit method is not in Ground

This method is the same for both Bush and Tree but not for Dirt so it is put in Bush and Tree separately. This does repeat code but ensures that there is no chance that addFruit capability is added to Dirt.

12. All methods/attribtues in EcoPoints are static

Doing it this way reduces dependencies since we don't need to always pass the player through several classes when an action happens that increases/decreases the EcoPoints. Instead, we only need to reference a single class to update EcoPoints throughout the game.

13. Assertions throughout the code

Assertions were used throughout the code, especially in actions/behaviours that can only be called by certain actors. This uses the fail fast (FF) principle to indicate when our program has received bad input as early as possible, which reduces the difficulty when debugging.

14. MoveToLocationBehaviour

This behaviour returns an action that moves the actor towards a destination. This idea is repeatedly used throughout the program – when dinosaurs find food, find a mate or travel towards the player. To reduce repeated code, we decided to make a behaviour to handle it so that we didn't need to copy the code to move towards a destination repeatedly.

15. Using private attributes and setters and getters for dinosaurs

This design decision was made with the principle of command-query separation. This reduced the chances of accidentally changing attributes, as well as preventing any side-effects from occurring.