FIT2099 Assignment 1

By: Ong Jin Bin, Mohamed Nabhaan Ali

Design rationale

Design Goal:

- To implement Tree Class, Bush Class and Fruit class
- To implement attributes fruits for ground,trees and bushes
- To implement Hunger and Breeding
- To Implement Allosaur and Brachiosaur
- To implement vending machine

Vending Machine

The *Player* can access the *VendingMachine* class to buy *Product* to be stored in the *Inventory*. Hence, the dependency of *VendingMachine* to *Player*.

The *VendingMachine* has many *Product*. All items purchasable from the vending machine implements the *Product* interface, which has attributes **priceInEcoPoints : int** and methods *buy()*. The *Product* interface is implemented to ensure all the other classes implementing it have price as well as methods to define the transaction process (purchase, sell).

VegetarianMealKit and *CarnivoreMealKit* are classes implementing the *Product* interface, which allows them to be purchased by the user as well as allowing the flexibility of different methods/attributes for these 2 classes.

Weapon is an abstract class implementing the *Product* interface, which is the parent class for *LaserGun.* The *Weapon* class cannot stand on its own, but shares default methods (such as -foodLevel of *Dinosaur* after attacking them) as well as some undefined methods which are to be implemented by the children class, depending on use cases (such as only *LaserGun* is able to deal damage to *Stegosaur)*.

Dinosaur + Egg

Dinosaur is an abstract class, which is the parent class of *Stegosaur, Brachiosaur, Allosaur*. The *Dinosaur* class cannot stand on its own, but shares attributes (such as **status**, **turnsAfterHatched** etc.) and methods (such as **breeds()** and **eats()** etc.) with its children class.

The *Allosaur, Stegosaur, Brachiosaur* classes inherit the *Dinosaur* class, of which they have individual attributes and methods (such as **turnsUntilNextAttack : int** attribute and **attack()** method for *Allosaur*).

StegosaurEgg, BrachiosaurEgg, AllosaurEgg are classes implementing the *Product* interface, which allows them to be purchased by the user as well as allowing the flexibility of different methods/attributes for these 3 classes. *turnsTillHatched : int* counts the number of turns before the eggs hatch, and this value decrements every turn.

Fruit

Starting off with the main point of discussion which is the implementation of fruits into the game. The final decision to implement fruits as a separate class rather than as an integer attribute of ground,bushes and trees was because we needed to track where a fruit is such as whether it is on a tree,bush or ground and if it was on the ground needing to keep track of how long till it will rot. With keeping all this in mind we chose to create the Fruit Class with the attributes location(String) and turnsTillRotten(Integer) with the method rotting().The location would be the only variable to be instantiated in the constructor and turnsTillRotten will start of as 15 for every fruit. After which every turn if the location of the fruit is ground it will decrement turnsTillRotten using the method rotting().

ArrayLists of Fruit was added to the classes Tree, Bush and Ground to keep track of the number of fruits that the instance of that class contains. The choice to use ArrayLists rather than a normal Java Array was made solely due to the fact that the number of fruits in an instance of any of the 3 classes Tree,Bush and ground will never be fixed. ArrayLists gives an easy way to access the Fruits methods if need be in the case of it being on the ground.

Tree

The implementation of the Tree class was a pretty straightforward requirement of the assignment . Other than the already mentioned fact of the it having the attribute of an ArrayList of Fruit. The class will be implemented with three methods: produceFruit(), dropFruit() and removeFruit() . The produceFruit method will randomly generate a number which will be used to see if a fruit is added to the list or not. The dropFruit method will also generate a random number which will check if a fruit instance in the list will drop to the ground. When this occurs it will be removed from the fruits ArrayList in Tree and added to the ground ArrayList. The last method removeFruit is used if a player picks a fruit from Tree . It will generate a random number which will determine whether the pick attempt was successful or not and remove a fruit from the Arraylist if successful and add it to the players inventory. These three methods were implemented because an Instance of Tree needed to have a chance of producing Fruit, A Chance of the Fruit to be dropped as well as picked

Bush

The implementation of the Bush class was also a straightforward requirement of the assignment. It also contains an ArrayList of Fruits and has the methods: produceFruit() and removeFruit(). These methods have the same functionality of the methods described in the Tree class and were implemented as players needed to be allowed to pick fruits from the bush as well as the bush needed to produce fruit.

Ground

The *Ground* class was further implemented by adding methods: removeRottenFruits() and removeFruit(). The removeRottenFruits method will iterate through the fruits on the ground and if their turnsTillRotten is 0 then they will be removed from the list. The removeFruit method has the same functionality as that of the method stated in Tree Class and Bush Class which is to generate a random number to determine whether the player has picked the fruit and then remove it from the list if that is the case.