Design Rationale

- This file explains all the classes we are about to add and all the reason for the existence of the classes and their role in our game, it is also mentioned a relationships and a few functions given that we will use to implement the roles for these classes.

1) Bushes Class – extends Ground and overrides the tick function to call the ripening() method every turn with a 10% chance of growing a fruit on the bush.

2) Dirt class – extends Ground, it will override the tick() method and for every turn it will call growBush() method and this method will check if the Dirt is next to atleast to squares of bushes or next to a tree, if it is next to two or more squares of bushes the possibility for the square of dirt to trun into a bush will be 10% and if next to a tree it will have 0% chance, if both these conditions are not met it will have a 1% chance to grow a bush. When a Bush is grown a Bush instance is created and added to the location of where the Dirt is on the Game map.

3) FoodItem Class- An abstract class that extends PortableItem class because all food items are portable and contains the basic common functionality of all the food items in the game.

4) Fruit Class – extends FoodItem and overrides the tick() function to call a function rot() which keeps track of whether the Fruit has been on the floor for over 15 turns and should rot.

5) Enum FruitStatus – has values {ON_FLOOR, ON_TREE, ON_BUSH, IN_INVENTORY} to add the capability to the Fruit class for it to know where the fruit is located.

6) Tree Class- will be updated to have a ripening() method just like bush to check if the tree has produced fruit (chance of producing ripe fruit for a tree is 50%) and this will update the static points variable of the Player and add 1 point.

7) Egg Class – extends the item class and overrides the tick function to find out how long it has been since the egg has been created and if the required number of turns for the egg is met then the Egg will run hatch() method which will create a dinosaur of the eggs species. The egg will also store what dinosaur the egg belongs to and will create a new instance of that specie of dinosaur depending on its hatch time.

8) Points Class – A class used to set and get the points value for the Actor using the Points class as a variable.

9) Player Class – will be updated to use a static variable of Points, so that this variable will be updated by the required Classes such as Tree, VendingMachine, Egg and etc... So that when the task of a ripe fruit is produced on a tree or a dinosaur hatches or a Item is bought from a vending machine, the required classes will update the Points on the Player accordingly, by reducing or increasing of the points.

10) Dinosaur Class – an abstract class for all the dinosaurs being created and the common functions and behaviours that all the dinosaurs will have when any dinosaur is created it will randomly be decided whether it is a male or female. It will also have a second constructor for an Admin to create Dinosaurs at the beginning and set the genders he/she wants.

11) Brachiosaur Class – extends Dinosaur and Overrides the required methods from the Dinosaur class,.

12) Allosaur Class – extends Dinosaur and Overrides the required methods from the Dinosaur class. Allosaurs will use the AttackAction with their weapon "claws" and deal 20 damage to the target stegosaurs Allosaurs will also have a list of stegosaurs they already attacked and cannot attack those stegosaurs until 20 turns pass by. They also cannot follow Brachiosaurs so they cannot have Brachiosaurs as targets Dinosaur's species for FollowBehaviour.

Note – All Behaviours of the dinosaur below are used to check whether the required conditions for the Actions to happen are met and if it does it will create and return the Action that the dinosaur will do.

13) BreedingBehaviour– extends Behaviour and Creates and returns the Action BreedingAction if the Dinosaur has more than the required foodlevel needed for breeding. If it does it will use the FollowBehaviour to follow a Dinosaur of the same species and opposite gender and the female is not pregnant and they will both do the BreedingAction(which extends Action).

14) HungryBehaviour– extends Behaviour and Creates and returns the EatingAction, It will first check if the Dinosaur has the less than the required food level to become hungry and if It does it will try to find its food and it will return the EatingAction(which extends Action) to eat the Food it has found, this means its heal() method will be called and will heal by required amount depending on the FoodItem it is consuming and whether or not the Dinosaur is a baby and the targets food level will decrease(if it is a dinosaur).

15) GrowingBehaviour – extends Behaviour and creates and returns the GrowingAction if the Dinosaur is a baby, the GrowingAction(which extends Action) will keep track of the dinosaurs

age(number of turns since it was born) and if it meets the required value, ex – stegosaur 30 turns, it will make the Dinosaur an adult.

16) LayEggBehaviour – extends Behaviour and creates and returns the LayEggAction if the dinosaur is a pregnant female, the LayEggAction(which extends Action) keeps track of how long the dinosaur has been pregnant and when the number of turns to lay an egg is reached it will create a new Egg item and the dinosaur will no longer be pregnant.

17) DeathBehaviour – extends Behaviour and creates and returns the DeathAction if the dinosaur has been unconscious for over the required amount of turns for its species and DeathAction(which extends Action) causes the dinosaur to die and creates a DinoCorpse object for the specie of dinosaur.

18) DinoCorpse Class– extends item and overrides the tick function to keep track of the number of turns that pass after it has died and calls the decay() function when the dead corpse has been on the map for the required amount of turns for the species.

19) LazerGun Class – extends WeaponItem and sets the damage that the lazer gun instance does (approximately a 1-2 hit to kill a stegosaur), we can use the AttackAction and Actor.getWeapon() to get this weapon and attack dinosaurs targets using it.

20) VendingAction Class – extends Action and overrides the menudescription() method to show options of the items the user can buy from the machine, then it takes the user input for the key of the item it has to enter to the VendingMachine, it validates if the entered key is correct and if a valid key is entered it sends this data to the VendingMachine instance and lets it process the points and transaction and if successful will add the item to Players inventory and if unsuccessful will stop the transaction.

21) VendingMachine Class – extends ground and holds a list of all the items it has within it for sale, It checks if the price of the item and the amount of points the users static point variable has and if he doesn't have enough acknowledges the Player he/she does not have enough points to purchase the item, other wise if they do then the VendingMachine will return the item to the vendingAction to add to the Players inventory.

22) FeedAction – will override the menuDescription() method to show the option to feed a target and will call the target Actors heal() method to heal the target by required amount depending on the FoodItem. The target can be fed an object of Fruit, MealKit or DinoCorpse depending on whether the species of the Dinosaur is carnivore or herbivore.

23) MealKit Class – An abstract class for the mealkits with all the common functions for both mealkits, MealKits will heal all Dinosaurs to max food level regardless of species.

24) Vege and Meat MealKit – extend MealKit class and will call the Actors heal() method to increase its food level. VegeMealKit only works on Vegetarian Dinosaurs and MeatMealKit works only on carnivorous Dinosaurs.