FIT2099 - Assignment 1 Design Rationale

This document serves to cover all the newly implemented items, actors, behaviours and any additional classes which have been altered. It also explains why classes were created and/or altered.

*Any new classes are denoted by an underline. E.g. NewClass
**Any altered classes are denoted by italics. E.g. *AlteredClass*

Item Classes

*subclasses of PortableItem

1. Laser Gun
    a. Purchasable from vending machine
    b. SubClass of WeaponItem
    c. Allows player to shoot dinosaurs to control population

2. Eggs
    a. Subclass of Portable Item
    b. Purchasable from vending machine
    c. Can be spawned by female dinosaur after mating or purchased from vending machine
    d. Contains Attributes & Methods which all egg classes share to reduce repeated code
    e. Parent Class of 3 Eggs
        i. StegosaurEgg
        ii. AllosaurusEgg
        iii. BrachiosaurEgg
    f. Consumable food item for Allosaurus
    g. Replaces itself with a dinosaur of its type (e.g. stegosaur) after a set amount of time after its dropped

3. Fruit
    a. Subclass of Portable Item
    b. Spawns from Trees and Bushes using a Probability rate
    c. Purchasable from vending machine
    d. Consumable food item for Brachiosaur and Stegosaur

4. VegetarianMealKit
    a. Subclass of Portable Item
    b. Purchasable from vending machine
    c. Consumable food item for Brachiosaur and Stegosaur

5. CarnivoreMealKit
    a. Subclass of Portable Item
    b. Purchasable from vending machine
    c. Consumable food item for Allosaurus

6. Corpse
    a. Spawns when a dinosaur dies
    b. Consumable food item for Allosaurus

Actor Classes

*subclasses of Actor

Dinosaur

- Subclass of Actor
- Contains Attributes & Methods which all dinosaurs share to reduce repeated code
- Parent Class of:
    - *Stegosaur*
    - Brachiosaur
    - Allosaurus

Action Classes

*subclasses of Action

1. FeedingAction
    a. Allows the Player to remove a food item from their inventory and feed it to an adjacent Dinosaur, thereby increasing their health

2. PickFruitAction
    a. Action to pick fruit from a tree/bush
    b. Implements a probability of success when picking, which is dependent on
        i. the type of actor picking fruit
        ii. If the fruit is on a tree/bush
    c. E.g. Stegosaurs cannot eat from trees, only from a bush or on the ground
    d. Brachiosaur can only eat from trees
        i. Can eat as many fruits as are on the tree in 1 turn

3. EatingAction
    a. Contains Attributes & Methods which all Eat actions share to reduce repeated code
    b. Parent Class of:
        i. EatMeatAction
            1. Ability to eat any meat item
        ii. EatFruitAction
            1. Ability to eat any fruit item

4. BreedAction
    a. Action to breed 2 dinosaurs of the same subclass, opposite sex and have the mateable capacility
    b. Creates an egg object in the female Dinosaurs inventory
        i. of same type as the dinosaur (e.g. brachiosaurEgg from 2 Brachiosaurs)
    c. Egg is dropped after a certain amount of turns from the female dinosaur.
    d. Dinosaurs have a timeout period after breeding before they can breed again
    e.

5. DisplayHungerAction
    a. Implements Printable
    b. When a dinosaurs hunger falls below a certain amount, it executes this action to print its Dinosaur Type, position and a “is getting hungry” message on the console”
        i. E.g. Stegosaur at (19, 6) is getting hungry!

6. *AttackAction*
    a. The class function is modified to initially check if the actor is an instance of the player or if it is an instance of Allosaur.
    b. If the actor is an Allosaur, it will check if it is has the capability to currently attack
    c. If the actor is a player, then it will always be able to attack

Behaviour Classes

***** Behaviour is the parent class of all other behaviour child classes

1. *Behaviour*
    a. Altered the getaction() method to take in a Dinosaur Object instead of a Actor object
    b. getAction(Actor actor, GameMap map) → getAction(Dinosaur dinosaur, GameMap map)
    c. Allows us to directly access specific capabilities of Dinosaurs which would not be possible if we took in the parent Actor Class

2. *FollowBehaviour*
    a. Added another constructor and getaction() which takes in a Location instead of an Actor
    b. These new additions were made to allow us to pass through a Location for the Actor to follow instead of only following Actors
        i. This functionality is linked to FindNearestLocation class which can return a location on the map.

3. MatingBehaviour
    a. Checks if it can immediately mate in the actionList, if not:
    b. Looks for all Actor object on the GameMap and creates a potentialPartner ArrayList containing those which:
        i. Are the opposite Gender to the current Actor
        ii. Have the capability to Breed
    c. Sends the list of potentialPartners to FindNearestLocation which returns the closest potentialPartner to the current Actor
    d. Passes the closestPartner to FollowBehaviour which moves the current Actor closer to the partner each turn

4. HuntingBehaviour
    a. Checks if it can immediately eat an adjacent corpses/eggs or attack an adjacent dinosaur in the actionList, if not:
    b. Looks for all Stegosaurs and corpses/eggs and creates arrayLists, preys and ItemLocations.
    c. Passes both preys and ItemLocations into FindNearestLocation which will return the closest ItemLocation and the closest preyActor

d. Compares ItemLocation and preyActor and returns the one which is closest to the current Actor
e. Passes the closest object to FollowBehaviour which moves the current Actor closer to the food source each turn

5. ScavengingBehaviour
    a. Checks if it can immediately eat an adjacent fruit in the actionList, if not:
        i. Looks for all fruits in the GameMap and creates an arrayList of Item Locations.
        ii. Passes ItemLocations into FindNearestLocation static class method which will return the location of an item which is closest to the actor.
        iii. It then uses this location to create an instance of FollowBehaviour which will be assigned to the followBehaviour attribute
        iv. followBehaviour.getAction() will be returned at the end of the function

Ground Classes

1. *Tree*
    a. Stores an arraylist of fruits
    b. Has a 50% create an instance of a fruit with the capability “On tree” that signifies that the object is on the tree
    c. Has a 10% chance to drop a fruit on the ground, which changes the capability of a fruit in its arraylist from being “OnTree” to being “OnGround”
    d. The tree will grow old and die after 25 turns

2. *Ground*
    a. Added the functionality such that it has 1% probability to spawn a bush
    b. *ground adjacent to at least 2 pre-existing bushes has a 10% probability to spawn bush
    c. **ground CANNOT grow bush if it is adjacent to a tree

3. Bush
    a. Has an attribute that is an arraylist of fruits
    b. Has a 10% chance to create an instance of a fruit with the capability “OnGround”
    c. The bush will grow old and die after 15 turns

4. VendingItemFactory
    a. Is a child class of Ground
    b. It has certain capabilities which allows it to check Ecopoints and create an instance of certain items in the Player Inventory if Ecopoints have enough credits
    c. It is implemented as a child of ground such that the preexisting action check (checking the possible actions of the ground surrounding) can natively access the vending machine’s functions

Other Classes

1. Counter
    a. Is a class which counts down its int counting attribute every turn
    b. It is used by:
        i. Dinosaurs to timeout its ability to be attacked
        ii. Pregnant female dinosaurs for when to lay the egg on the ground
        iii. Egg to tell it when to hatch

iv. Mated Dinosaurs to timeout when it can mate again

2. Ecopoints
    a. We implement the ecopoints class as a class with static attributes and methods.
    b. This is because there is only 1 player (so only 1 ecopoints tracker is needed) and we do not have to make multiple instances
    c. Any class can simply call the static methods without having to interact with the player

3. FindNearestLocation
    a. The use of this class is to solely provide static methods which return the closest Location/Actor to the current actor
    b. We put it in its own class to meet Object Oriented Design Principles
    c. Takes in a currentActor along with a list of Actors or Locations
    d. Has an int attribute which records the current minDistance
    e. If the input is a list of Actors:
        i. Has a closestActor variable which stores the closest actor to the current Actor
        ii. Calls a distance() method between the currentActor and each of the Actors in the Actors list
        iii. Compares the actors distance to the current min and sets it as the closestActor if its distance < minDistance
        iv. Returns the closestActor after running through the list of Actors
    f. If the input is a list of Locations
        i. Has a closestLocation Variable which stores the closest Location to the current actor
        ii. Calls a distance() method between the currentActor and each of the Locations in the Locations list
        iii. Compares the Locations distance to the current min and sets it as the closestLocation if its distance < minDistance
        iv. Returns the closestLocation after running through the list of Locations