# FIT 2099 Assignment 3

Design Rationale

Members: Tan Jin En, Vernise Ang Veng Yan

## Summary of the project

This project is a text-based game. The player is running a dinosaur park and many actions can be done throughout the game. Dirt, trees and bushes are plants which can produce fruits. Dinosaurs can move around, get hungry, breed, attack other dinosaurs and die. The player can collect fruits from the dirt, trees and bushes. Fruits can be fed to hungry dinosaurs. Players can also purchase items with eco points through the vending machine.

## Design Goals

The goal of this design is to ensure the program will run smoothly. The classes should be able to work with each other and the attributes can be passed between classes which are related.

## Ground, dirt, trees and bushes classes

Ground is an abstract class which implements the Ground Interface and it is under the engine package. This class contains and instance of capabilities. Capabilities of the ground can be added by using the addCapability method and can be removed by using the removeCapability method.

Both dirt and tree can be found on the ground, so dirt and tree inherit the properties of the ground. Bush has an association relationship with the Dirt class.

The Dirt class inherits the ground class. The representation of a dirt is a dot (.). A dirtToBush() method is implemented in this class. This method will get a random number using the random number generator. This value will be used to perform multiple conditional checks to if the current dirt can be replaced by a new bush object. The conditional check rules are as follow:

1. Initial chance to spawn a new bush is 0.01
2. If the dirt is next to two bushes, the chance is 0.1
3. If the dirt is next to a tree, the chance is 0

There is also a tick method use to check the condition around the dirt every turn. The growBush method is used to create a new bush.

The Tree class also inherits ground. It also has a tick method which increments the age of the tree every turn. Multiple conditional checks are carried out to check if the tree is fully grown. The representation of a small tree is ‘t’ (when the age reaches 10) and for a fully grown tree, its ‘T’ (when the age reaches 20). If the chances are fulfilled, a new fruit is produced at the same location of the tree.

**Lake class**

The representation of lake is '~'. The lake inherits the ground class. Every 10 turns, the chance for rain fall will be checked. If the chance is lesser than 0.2, then the capacity of the lake will increase by rainfall*20. The maximum number of fishes in the lake is 25. Every turn, there will be a 0.6 chance for a new fish to be created. If the lake has less than 25 fish, the number of fish will be incremented. The method pterodactylsFlewOver() will use the random number generator to decide whether it removes 0 or 1 or 2 fishes. Every time the Pterodactyls flies over, it's water level will increase by 30. The method canActorEnter() is used to prevent Allosaurs, Brachiosaurs and Stegosaurs to enter the lake, but Pterodactyls can.

**Fruit class**

The representation of fruit is 'f'. The eco point value of a fruit is 30 eco points.
A tick() method is implemented to check if the fruit will rot or not. If the number of turns reached 15 turns, the fruitRot() method will be called.
The fruitRot() method will be use the remove the fruit from the map.

**Food class**

The method getEnergyReplenish() will return the amount of energy replenished.
The method setEnergyReplenish() is use to set the amount of energy replenished.

**Lasergun class**

The representation of a lazer gun is "L". The Lazergun class inherits the WeaponItem class and it also implements The Purchasable interface.

**Fish class**
The representation of a fish is '<'. The energy replenished when it is eaten is 5.

**Dinosaur classes**

A dinosaur is an abstract class that extends Actor. It stores all the basic properties of a dinosaur in the game. The class defines attributes which are unique to each dinosaur types. For example, behaviour, food source, display character, gender etc. This abides by the DRY principle as the child classes will inherit all these attributes and methods inside the dinosaur class. There are three child classes: Stegosaur, Brachiosaur and Allosaur. The method playTurn is implemented. For every turn, the current age of the dinosaur will be incremented and the hitPoints will be decremented. The dinosaur's behaviour will be checked during every turn. If there is a behaviour, a corresponding action will be called.

**Brachiosaur class**

The Brachiosaur dinosaur is initialised. Brachiosaur’s representation is ‘b’. It’s max hit points is 160, the eco points earned when it’s egg hatches is 1000, it needs 30 turns during mating to get pregnant, it needs to wait for 50 turns for a baby brachiosaur to turn to an adult brachiosaur, the gender is either female or male and after being unconscious for 15 turns, the brachiosaur will die. Each Brachiosaur will have a wander behaviour.
The method setHitPoints() is use to set the current hit points.
The addFoodSource() method is use to add Brachiosaur’s food source which are vegetarian meal kit and fruits.
A killBush() and removeBush() method is implemented. In the killBush method, the random number generator is use to calculate the chance. If the Brachiosaur steps on the bush, it has 0.5 chance to kill it and the removeBush() method will be use to remove the bush.
The playturn() method is use to increment the brachiosaur’s age and decrement it’s hit points every turn. The behaviour of the brachiosaur is checked every time too to see if it has any behaviours.
The layEgg() method is use to place the egg which is laid by the brachiosaur on the map.

**Allosaur class**

The Allosaur dinosaur is initialised. Allosaur’s representation is ‘a’. It’s max hit points is 100, the eco points earned when it’s egg hatches is 1000, it needs 20 turns during mating to get pregnant, it needs to wait for 50 turns for a baby allosaur to turn to an adult allosaur and after being unconscious for 20 turns, the allosaur will die. Each allosaur will have a wander behaviour.
The method setHitPoints() is use to set the current hit points.
The addFoodSource() method is use to add Allosaur’s food source which are carnivore meal kit, stegosaur corpse, brachiosaur egg and stegosaur egg.
The Allosaur has a special behaviour call FierceBehaviour. The behaviour can call the AttackAction to attack stegosaurs and their food level will increase by 20.
The playturn() method is use to increment the brachiosaur’s age and decrement it’s hit points every turn. The behaviour of the brachiosaur is checked every time too to see if it has any behaviours.
The layEgg() method is use to place the egg which is laid by the allosaur on the map.

**Stegosaur class**

The Stegosaur dinosaur is initialised. Stegosaur’s representation is ‘d’. It’s max hit points is 100, the eco points earned when it’s egg hatches is 100, it needs 10 turns during mating to get pregnant, it needs to wait for 30 turns for a baby stegosaur to turn to an adult stegosaur and after being unconscious for 20 turns, the stegosaur will die. Each stegosaur will have a wander behaviour.
The method setHitPoints() is use to set the current hit points.
The addFoodSource() method is use to add Stegosaur’s food source which are vegetarian meal kit and fruit.
The playturn() method for the allosaur class is in the Dinosaur class. This method is use to increment the brachiosaur’s age and decrement it’s hit points every turn. The behaviour of the brachiosaur is checked every time too to see if it has any behaviours.
The layEgg() method is use to place the egg which is laid by the allosaur on the map.

## Ecopoints

Ecopoints is static and can be access by all the other classes in the game. Getter and setter methods are implemented to get and set current eco points. Ecopoints can be gain when:

- a ripe fruit is produced by a tree (1 point).
- a ripe fruit is harvested from a bush or a tree (10 points).
- fruit is fed to a dinosaur (10 points)
- a stegosaur hatches (100 points)
- a brachiosaur hatches (1000 points)
- an allosaur hatches (1000 points)

The method isSufficient() is use to check if the player's eco points are enough to purchase the item selected.

The method addEcoPoints() is use to increase the player's eco points when the player gains eco points when achieving the above goals.

The method subtractEcoPoints() is use to decrease the player's eco points when the player successfully purchased an item.

## Vending machine class

Vending machine extends ground. The representation of a vending machine is 'V'. Only one vending machine is allowed to be on the map. Players can call the PurchaseItemAction to purchase the items in the vending machine. Ecopoints will be deducted during the purchase item action. A conditional checking is perform to ensure that the item's cost is lesser than the ecopoints. The classes instantiated by vending machine are Fruit, DinosaurEgg, CarnivoreMealKit, VegetarianMealKit and LazerGun. LazerGun is a subclass of the WeaponItem in the Engine. The other items in vending machine are subclasses of Food. The method allowableActions() is use to buy items from the vending machine. The items will be added to an array list called 'vendingItem'. A for loop is use to check the cost of every item, if the player's eco points is sufficient to buy the item, the PurchaseItemAction is called and the item is either added to the map or to the player's inventory.

## Action classes

### Attack action

The attack action has a target to attack. The method execute() is use to undergo the attack process. A weapon will be obtained and the random Boolean value generator is use to check if the actor misses or successfully attack the target. The appropriate string will be displayed. After the attack, if the target is no longer consious, a corpse is created and the representation of the corpse is '%'. The corpse will be added to the same location of the target when it died.
The menuDescription() method is use to display the string: (Actor) attacks (target).

### Death Action

The method execute() is use to undergo the death process. DeathAction will remove the corpse out of the map.
The menuDescription() method is use to display the string: (Actor) is dead!!!

**Eat Action**

The method execute() is use to undergo the eating process. EatAction will find the location of the food that the dinosaur could eat. After eating, the food will be remove and the actor's food level will be replenished by using the heal() method.

**Feeding Action**

The method execute() is use to undergo the feeding process. FeedingAction can be called to feed the dinosaur. The item being fed will be removed from the player's inventory. The target dinosaur's food level will be replenished by using the heal() method. If the item fed to the dinosaur is a fruit, the method addEcopoints() will be called to add 10 eco points to the player.
The menuDescription() method is use to display the string: (actor) feeds (target)

**Mating Action**

The method execute() is use to undergo the mating process. If the dinosaur is not a male, it will call the pregnant behaviour. Otherwise, the target will call the pregnant behaviour.
The menuDescription() method is use to display the string: (actor) breeds with (target)

**PurchaseItemAction class**

Once the PurchaseableItemAction is called, the item will be stored in the instance variable 'item'. A method call 'execute() will add the item to the player's inventory and subtract the cost of the item from the player's eco points.
The method menuDescription() will be called to display the text "(Player) successfully bought: (item)".

**Starving Action**

StarvingAction will be called when the food level of the dinosaurs dropped to a certain level.
The menuDescription() method is use to display the string: (actor) is starving!!

**Drink Action**
If the location is of the lake is not null, check if the dinosaur is Brachiosaur, Allosaur or Stegosaur. If the dinosaur is a Brachiosaur, increase the water level by 80, if its an Allosaur and Stegosaur, increase the water level by 30

**Thirsty Action**
If the water level is below 40, the thirsty action will be called. The menuDescription() method is use to display the string: (actor) is starving!!

**TerminateGame Action**
If the game is terminated, the menuDescription() method is use to display the string: “Terminating game!!!”

**Win Action**
If the player won the game, the sentence “CONGRATULATIONS!!! YOU WON THE GAME!” will be displayed. If the player did not win, the sentence “TURN LIMIT REACHED!!! YOU LOSE! TRY HARDER NEXT TIME!!!” will be displayed.

**Behaviour classes**

**Dinosaur Behaviour**s

Dinosaur behaviour is an abstract class inherited from behaviour. It is the base class for all the dinosaur behaviours. The behaviours are pregnant behaviour, mating behaviour, fierce behaviour, follow behaviour and wander behaviour.

**Fierce behaviour**
The method getAttackable() is use to check if there is a stegosaur around the actor. If there is, the stegosaur will be the target.
The method getAction() is use to find the target. If no target is found, the actor will move around. If the target is a food, then the EatAction will be called.

**Follow behaviour**

The follow behaviour will have a target.
The method getAction() is use to find the target. If no target is found, the actor will move around.
The distance() method is use to calculate the distance between the actor’s current location and the destination.

**Mating Behaviour**

The method getAction() is use to find other dinosaurs which are eligible to breed with. If there is an eligible dinosaur, the follow behaviour will be called and the actor will move towards the nearest eligible dinosaur to mate with it. When the distance reaches to 1, mating action will be called. If not, the follow behaviour will continue. If there are no eligible dinosaurs to mate it, the actor will do nothing by using the DoNothingAction(). For Pterodactlys, the location of the tree needs to be checked first. If the two Pterodactyls dinosaurs are on top of two trees, then the mating action will be called, else, the actor will do nothing by using the DoNothingAction().

**Pregnant Behaviour**

The method getAction() will first check the type of the dinosaur. If it is a brachiosaur, allosaur or stegosaur, then it will check if dinosaur has reached the time to be pregnant. If the number of pregnancy turns are reached, the dinosaur will lay an egg. If the number of pregnancy turns have not been reached, the actor will do nothing by using the DoNothingAction(). If the dinosaur is a pterodactyls, it will check if dinosaur has reached the time to be pregnant and check if it is on the tree. If yes, then the Pterodactyls will lay an egg.

**Unconscious Behaviour**

The method getAction() is use to check if the number of turns after the dinosaur is dead reaches the unconscious limit, if it has reached the unconscious limit, the DeathAction() will be called and the dinosaur dies. Otherwise, the StarvingAction is called.

**Wander Behaviour**

The method getAction() is use to check if the actor can move to the next exit location. If yes, then the actor will move to the next exit. The method getFoodInTile() is used to check if the item is one of the food source of the dinosaur. The method getClosestFood() is to get the food which is located nearest to the dinosaur.