Assignment 1
Planning and design

Design Rationale

- To fulfil some missing parts of the game Jurassic World via creating and modifying classes.

Overview

Classes to be modified

- Dirt, Tree, Stegosaur

Classes to be created

- Bush, Egg, VendingMachine, Fruit, Dinosaur, Brachiosaur, Allosaur, HarvestAction, PurchaseAction, ShootAction, FeedAction, BreedBehaviour, LayEggBehaviour, SeekFoodBehaviour, HatchBehaviour, GroundLocation, FruitHarvested, CanivoreMeal, vegetarian, LaserGun, Type(Enum), EcoPoint(static)

We have designed most of the classes to inherit from the existing classes from the engine. Since engine provides the most of necessary parent classes, it should be able to reduce dependencies and fulfil given requirements. Stegosaur, Brachiosaur and Allosaur are going to have same attributes and methods, so we designed to create parent class of them, which is Dinosaur, and the parent class extends the Actor in the engine.

Hence, we look forward to reduce repetition(DRY) of codes. EcoPoint class will have static attributes and methods in order to access from any where it needs.

The reason for the EcoPoint being a static class is that in each turn, we do not know how many points will be obtained from each objects.

Hence, any classes need to interact with EcoPoint will access via static methods. We decided to create enum called Type to specify objects in tick() in GroundLocation in order for corresponding performance for different objects.

Design of each classes

Dirt (extends Ground)

- Responsible for produce fruits
- This overrides tick() method from Ground class.

- Modify the tick() to produce a fruit depending on a chance

Tree (extends Ground)
- Responsible for producing fruit and a bush and drop a fruit depending on a chance
- This overrides tick() method from Ground class
- Modify the tick() to produce a fruit depending on a chance

Bush (extends Ground)
- Responsible for producing a fruit by chance
- Bush extends Ground. When tick() is invoked in GameMap, check if it a Dirt, if so, replace to a Bush using setGround()
- Override tick() from Ground and then create Fruit instance if it can

GroundLocation (extends Location)
- Responsible for check capabilities of the object for corresponding performance. For instance, getGround() to check
  which object stays on the spot and then perform tick action.
- This class extends Location class to override tick() method to find objects nearby.
- If a instance of subclass of Ground at specific location override tick() that functions to find objects nearby
  and give it chance to grow a bush by different chance.

HarvestAction (extends Action)
- This action is to attempt to find a fruit from a tree or a bush. This action has a chance of failing.
- This class overrides the execute() and menuDescription() method from the Action class
- In every turn, user will be given a chance for choosing an action. If this is selected, execute().

Stegosaur (extends Dinosaur)
- Responsible for wander, eat fruits and breeding
- This class extends Dinosaur that provide shared methods with brachiosaur and allosaur.

- Overrides methods to if it is hungry, baby and conscious for WanderBehaviour, EatBehaviour and BreedBehaviour.

Brachiosaur (extends Dinosaur)
- Responsible for wander, eat fruits and breeding
- This class extends Dinosaur that provide shared methods with Stegosaur and Allosaur.
- Overrides methods to if it is hungry, baby and conscious for WanderBehaviour, EatBehaviour and BreedBehaviour.

Dinosaur (extends Actor)
- Responsible for providing shared methods for Stegosaur, Brachiosaur and Allosaur
- This extends Actor to let child classes to override from Actor
- This simply gives an ability to override.

Allosaur (extends Dinosaur)
- Responsible for attack Stegosaur or Brachiosaur
- This extends Dinosaur
- This also override getAllowableActions() to add ShootAction and may be dead if get a gun shot.

BreedBehaviour(implements Behaviour)
- Responsible for breeding when two same species with opposite gender are nearby in given condition
- This overrides getAction() from Behaviour class
- Create distance() method to check for adjacency with the target before performing an action.

LayEggBehaviour(implements Behaviour, Dinosaur)
- Responsible for laying an egg in a given condition
- This implements Behaviour to override methods.
- Override canLayEgg() methods from Dinosaur and getAction() from Behaviour before an action.

Egg(extends Ground)
- Responsible for wait for a few turns before creating an instance of corresponding dinosaur.
- This extends Ground class and override tick() method for counting the number of ticks. After that, GroundLocation

will be responsible for creating an instance of it on the same spot.
- Interacting through GroundLocation class after appropriate turns pass to create an instance.

EcoPoint(static class)
- Responsible for collecting point from different situations
- As this class is static, any classes need this will interact via static methods
- Any classes that need to check add or subtract will use this class using static methods and a variable.

VendingMachine(extends Ground)
- Responsible for selling items to the player
- Extends Ground and override allowableAction() for PurchaseAction from the player
- Override tick() methods to check if items are sold and notify to EcoPoint

Type (ENUM)
- Responsible for checking which object is in the spot
- Objects will be added capabilities using this Enum
- This enum will specify which object is in the specific location for corresponding to determine what needs to be done.

PurchaseAction (extends Action)
- Responsible for the player can purchase when VendingMachine is nearby
- This will extends Action. getAllowableActions() will be overridden in VendingMachine class and then PurchaseAction will be added.
- Since PurchaseAction is added to VendingMachine, if player approaches to the VendingMachine player may perform PurchaseAction

Fruit (association with Tree and Bush)
- Responsible for staying in the same spot for 15 turns.
- This class does not directly interact with the existing system, but once tick() in Tree and Bush will use Fruit class.

- Since tick() in Tree or Bush will be called every turn, rotCounter will be incremented if they have Fruits

CarnivoreMeal, vegeterianMeal and FruitHarvested (extends PortableItem)
- Responsible for carried by the player and disappear when fed
- This extends PortableItem class. Then this will override getAllowableActions() to add PurchaseAction class.
- When player selects this from VendingMachine via PurchaseAction, then addItemToInventory() will be used.

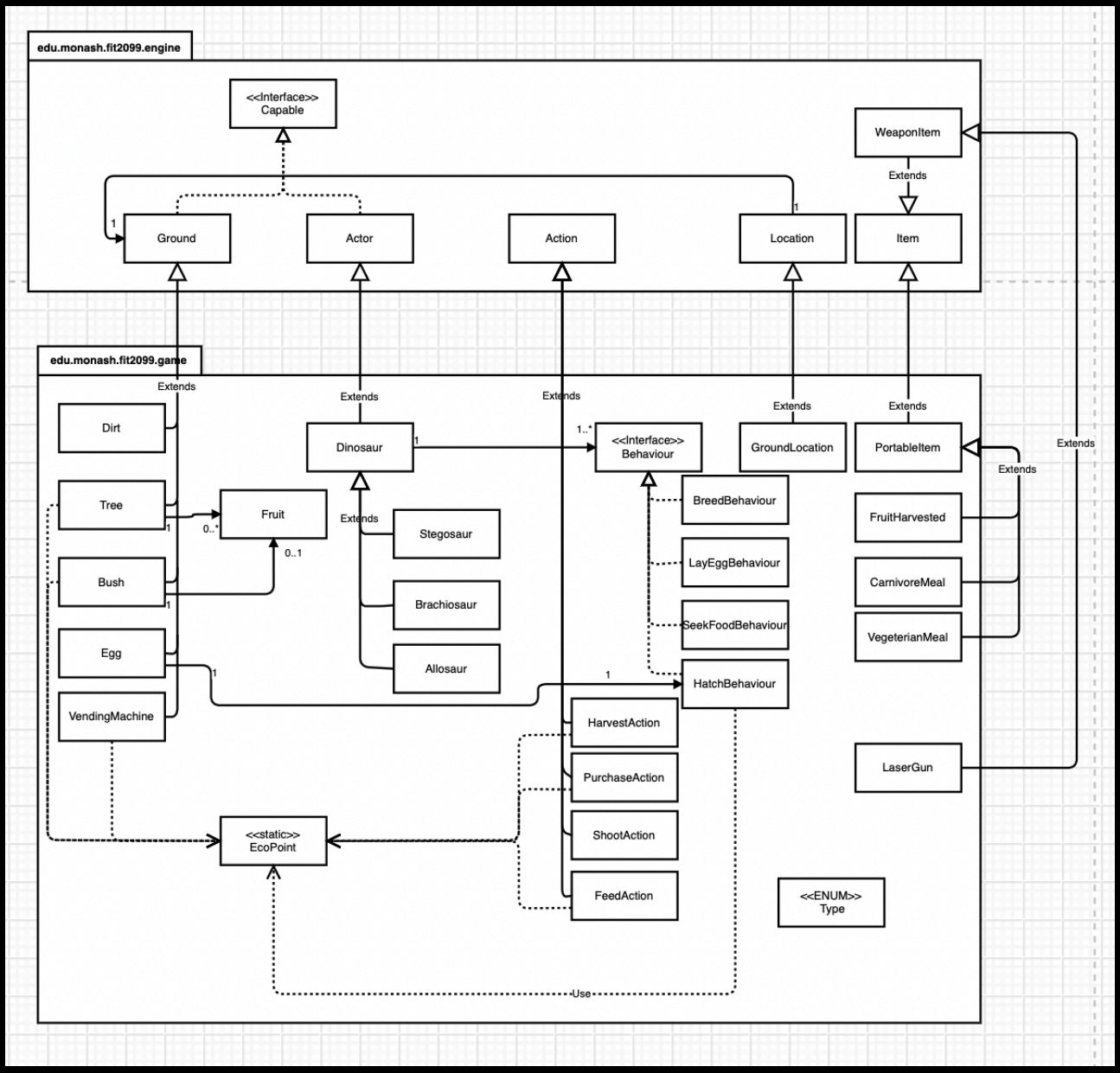
LaserGun (extends WeaponItem)
- Responsible for shoot the gun to remove Stegosaur if need.
- This extends WeaponItem and ShootAction will be added to getAllowableActions() in Allosaur.
- If LaserGun is in inventory, player always performs ShootAction

ShootAction (extends Action)
- Responsible shooting LaserGun if player holds it
- Overrides execute() from Action and getWeapon() to check if player holds LaserGun, if so, performs an action
- target will be assigned and if player wants to shoot then perform it

HatchBehaviour (implements Behaviour)
- Responsible for Egg to hatch in proper turns
- Overrides getAction() from Behaviour to count turns and create new instance
- Egg 'has a' HatchBehaviour and in every turn, invoke turnCounter method in HatchBehaviour

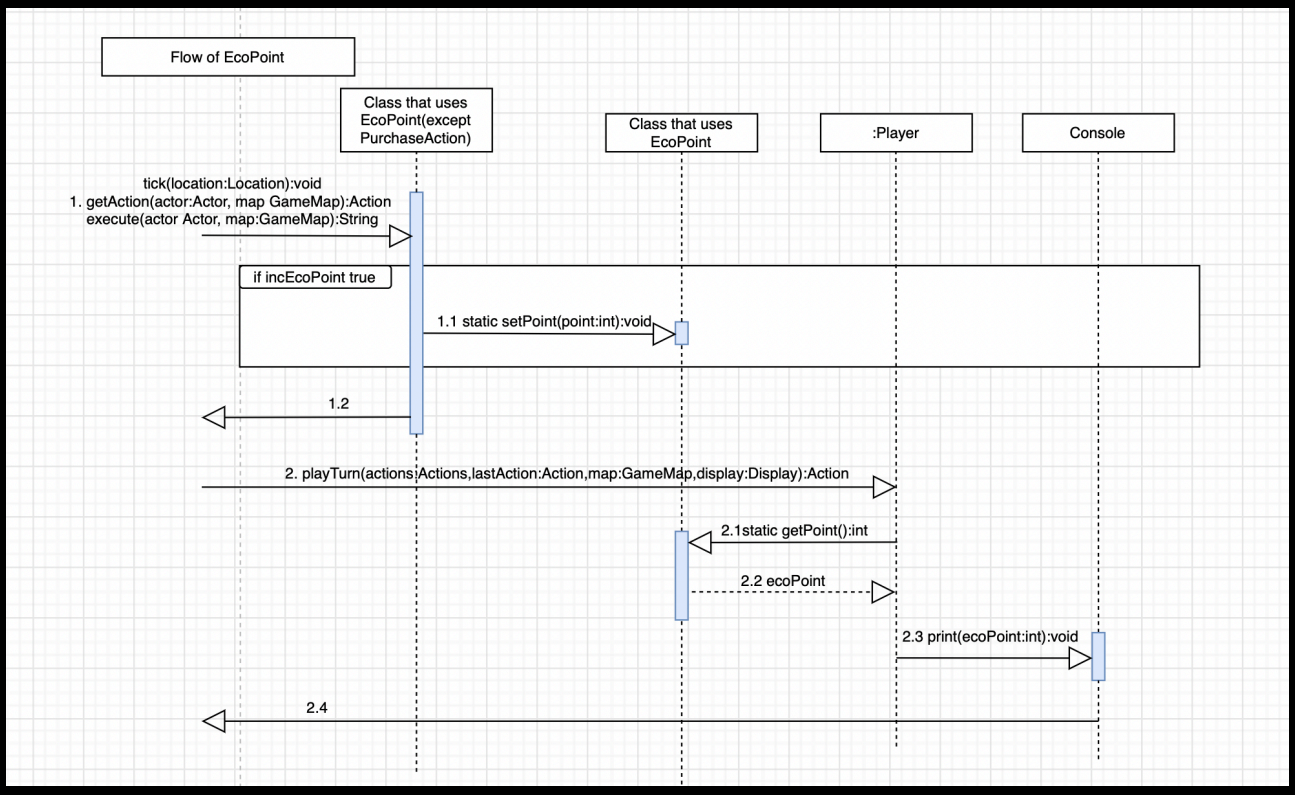
EcoPoint (static)
- Responsible for add or subtract eco points
- This does not interact with existing system
- Any class needs to add or subtract the point, static methods will be used.

UML Diagram

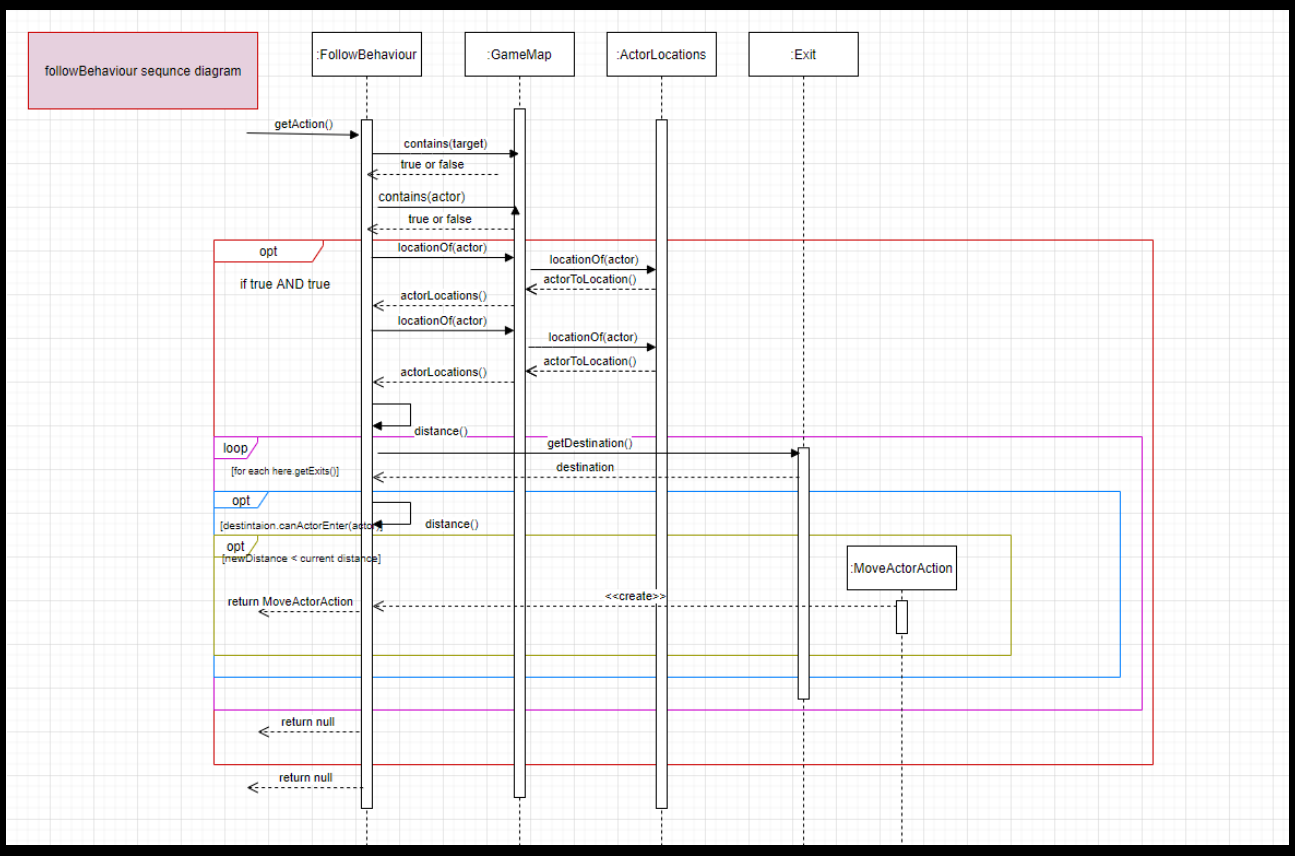

edu.monash.fit2099.engine
<<Interface>>
Capable
WeaponItem
Extends
1
Ground
Actor
Action
1
Location
Item
edu.monash.fit2099.game
Extends
Extends
Extends
Extends
Extends
Dirt
Dinosaur
1
1..*
<<Interface>>
Behaviour
GroundLocation
PortableItem
Extends
Tree
Fruit
Extends
Stegosaur
BreedBehaviour
FruitHarvested
1
0..*
0..1
Bush
Brachiosaur
LayEggBehaviour
CarnivoreMeal
1
Egg
Allosaur
SeekFoodBehaviour
VegeterianMeal
1
1
VendingMachine
HatchBehaviour
HarvestAction
LaserGun
PurchaseAction
<<static>>
EcoPoint
ShootAction
FeedAction
<<ENUM>>
Type
Use

Interaction Diagram(as in Sequential diagram)

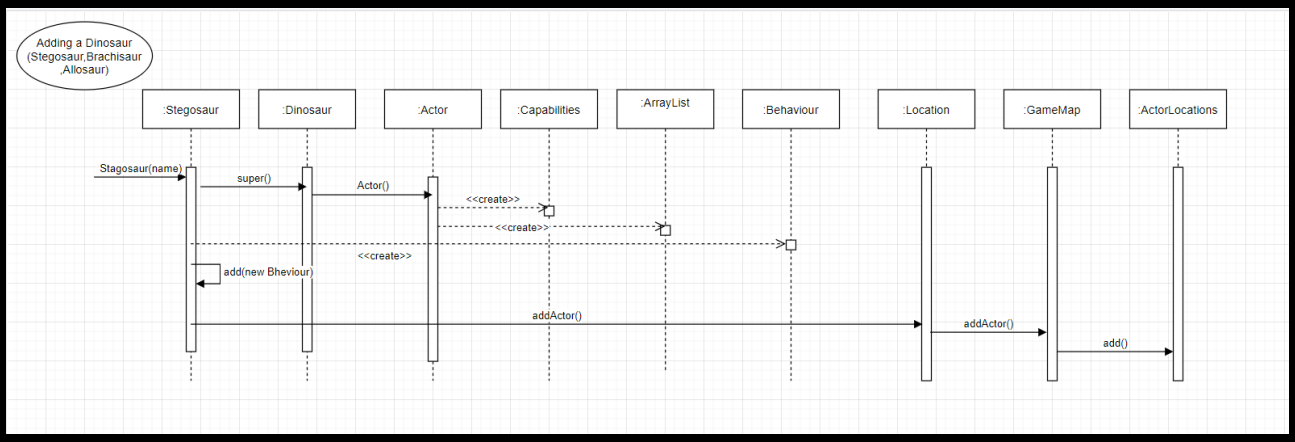
Sequential diagram for EcoPoint



Sequential diagram for FollowBehaviour



Sequential diagram for Adding a Dinosaur



Sequential diagram for subclasses of Ground