FIT2099 Assignment 3 (Updated)
Design Rationale
Tute 06 Team 47

Shaun Tan (31123996), Johan Azlan (31165001)

Dirt, Tree and Bushes

- **Dirt class (Ground Package)**
  - Create an instance of Dirt object
  - Have a tick method which for each turn it will check the first the adjacent square if it is a tree, if it is a tree, then it will have no chance to grow a Bush object, else if check the two adjacent squares is a bush, if it is then it will have a 10% chance to create a Bush object, else it will have a 1% chance to create a Bush object.
  - Need to remove the Dirt first, and then at the current location add in the Bush object

- **Bush class (Ground Package)**
  - Extends Ground
  - Bush depends on Dirt because in order for the game to have a Bush object, all depends on the probability of Dirt creating a Bush object.
  - Create an instance of Bush object
  - Have a list of fruits
  - Have a method tick which has a 10% chance to create a Fruits instance and add it into the list of fruits

- **Tree class (Ground Package)**
  - Create an instance of Tree Object
  - Have a list of fruits
  - Have a method tick which not only grow the tree but also for each turn, it has a 50% chance to create a Fruit object and add it into the list of Fruits
  - For each turn, it will also check if there's Fruit in the list of fruits, if there is, then it will have a 5% chance of being remove from the Tree list of fruits and add it to the ground of the current Tree

- **Fruit class (Item Package)**
    - Extends PortableItem
    - Extends PortableItem since it can be pick up by a player
    - Create an instance of Fruit Object
    - This class is also responsible for checking if the fruit is on the Ground or not, if it is then it will start to keep track of the turns that it is on the Ground, after 15 turns then remove this Fruit from the current location because it has rotten away. Else if not on the ground, it will not rot.

- **PickUpFromBushTree class (Action Package)**
    - Extends Action class
    - It will check the current location of the player and if the current location has a tree or bush, then it will check if the tree or bush has Fruit object, if it has then it will have a 40% chance of successfully picking up the fruit.
    - Implement pick up fruit from ground and bush.

Hungry dinosaurs

- **HungryBehaviour class (Behaviour Package)**
    - Implement the Behaviour interface
    - This class is responsible for checking if the hitPoints of the dinosaur is lower than their hungry level, if it is then first print out the message saying that the dinosaur at the location of the dinosaur is hungry, Then, check if the current location of the dinosaur has the correct food source for the dinosaur, if it has then consumed it and add the hitPoints of the dinosaur appropriately. Else, it will loop through the whole gamemap and check for the nearest food source and move one step closer to the nearest food source each turn until it reaches there and consume it. Else, it will just return null. If the dinosaur eats the food, the food will be removed from the location using the location.removeItem() method.
    - Stegosaur will check for Fruit that is laying on the Ground or Bush
    - Brachiosaur will find the nearest Tree
    - Allosaur will find the nearest Stegosaur, or Brachiosaur or even Egg
    - Pterodactyl will find the nearest Lake or corpse on the Map

- **UnconsciousBehaviour class (Behaviour Package)**
  - Implement the Behaviour interface
  - This class is responsible for checking if the dinosaur is conscious or not, if not conscious then it will print out a message on the console saying that the dinosaur is unconscious and the dinosaur will remain at the current location and create a new DoNothingAction(). Else it will just return null
  - This class is also responsible to check if the dinosaur is unconscious due to thirstiness or not, if it is then it will add this behaviour to the list of behaviour of the dinosaur

- **Stegosaur class (Dinosaur Package)**
  - Extends Dinosaur class
  - Each turn it will loop through the Behaviour list to execute all the Behaviour, and based on the return value of the Behaviour, it will do some appropriate action.

- **FeedAction class (Action Package)**
  - Extends Action class which is an action
  - it is responsible to feed the dinosaurs with the dinosaur's food
  - it will check if the player's inventory has the right item/food for the dinosaur and if it has then it will remove the item/food from the player's inventory and add the hitPoints of the dinosaur appropriately using heal method in the Actor class.

- **EatAction class (Action Package)**
  - Extends Action class
  - It is responsible to implement the eating action of the Dinosaur
  - It will check if what species of dinosaur it is then check for the approiate food and increase the food level of the Dinosaur. And also it will remove the food from the Ground after the Dinosaur eat it.

- **FollowFoodBehaviour class (Behaviour Package)**
  - Extends Behaviour class
  - This class is will have one Location object as the parameter, the location of the food. Then it will move the actor one step closer to the food.

Brachiosaur

- **Brachiosaur class (Dinosaur Package)**
    - Extends Dinosaur class
    - Create an instance of Brachiosaur with hitpoints 100 and max hitpoints is 160 if it is not a baby
    - Have a variable to indicate whether it is a baby or not. If it is then, it will not have the BreedingBehaviour(). Then will have a variable to keep track of the age of the Brachiosaur, and each turn add the variable. Then, if the age has exceeded 30 turns, then the variable of indicating baby or not will be false and BreedingBehaviour() will be added to the Brachiosaur.

    - Each turn it will check the current location of the Brachiosaur, if the current location has a Bush then it will have a 50% chance of removing the Bush and create a new Dirt object at the current ground using setGround(). Moreover, for each turn, it will also decrease the object hit points by 1 using hit method in Actor class and loop through each Behaviour and if the behaviour is not null then call the getAction of the Behaviour. And also, for each turn if the UnconsciousBehaviour is not returning null then it will also keep track of the total turns of the Brachiosaur being unconscious, if more than 15 turns then remove the Brachiosaur from the current location.

Breeding

- In order to implement the breeding, we have made a few changes to the previous class. For each dinosaur, we will initialize it as a male or female, with a 50% chance of each. And, it will have a variable that indicate whether the dinosaur is a baby or not.

- **BreedingBehaviour class (Behaviour Package)**
    - Implement the Behaviour interface
    - This class is responsible to check if the dinosaur is sufficiently well fed based on the species. If it is then it will first check if the adjacent have an actor or not, if it has then check if it is the same species and also the opposite sex, if it is then the Female dinosaur will add a capability of Enum pregnant. If the adjacent has no dinosaur, then it will loop through the whole map and check for the nearest same species and opposite sex. Then move one step closer to the location of that dinosaur.

- In order for the Pterodactyl to Breed on a tree only, we will create a new method which will check if the Pterodactyl is on a tree and if the adjacent has a Tree and if the Tree has a opposite sex Pterodactyl, if it has then it will breed by calling the Mating Action. If not, it will find the opposite sex Pterodactyl and go to it's nearest Tree. As for Pterodactyl breeding, the Female Pterodactyl will go to a Tree and wait for the Male Pterodactyl to come and mate with her.

- **MatingAction class (Action Package)**
  - Extends the Action class
  - This class will implement the mating process. It will add the PREGNANT capability to the female dinosaur if it successfully gets pregnant

- **Egg class (Item Package)**
  - Extends the Portable Item
  - It will have a name based on the Dinosaur species and the appropriate hatch time. And have a variable that keep tracks of the age of the egg.
  - It has a method of which for each turn it will keep track the age of the egg and also the hatch time. If the age of the egg reaches the hatch time, then it will remove the egg from the location and create a new Dinosaur based on the name of the egg and add the new Dinosaur at the current location using location.addActor() method.
  - When each baby Dinosaur is created, the variable that indicates the Dinosaur whether is a baby will be true and then it will not have the ability to Breed.

Eco Points and purchasing

- **LaserGun class (Item Package)**
  - Extends PortableItem since it can be pick up or add into the player's inventory

- **VegetarianMealKit class (Item Package)**
  - Extends PortableItem since it can be pick up or add into the player's inventory

- **CarniovoreMealKit class (Item Package)**
  - Extends PortableItem since it can be pick up or add into the player's inventory

- **VendingMachine class**
    - Extends Item
    - This class will be responsible to print out all the item that it is selling and the eco points of item
    - This class will check if the Player have enough eco points to buy the item, if doesn't have then it will print out the message. Otherwise, it will add the item to the inventory of the Player.

Allosaurs

- **Allosaurs class (Dinosaur Package)**
    - Extends Dinosaur class
    - Create an instance of Brachiosaur with hitpoints 20 and max hitpoints is 100.
    - Since Allosaurs do not appear on the map at the start of the game, they can only be grown from an egg, thus all the hitpoints starts with 20. And do not have the BreedBehaviour, after 50 turns, it will have the grown into a adult allosaur which will have the BreedBehaviour().

Death

- **Corpse class (Item Package)**
    - Extends PortableItem since it can be picked up by player
    - Create a corpse with variable that indicates the turn that the corpse should be in the map
    - This class is responsible to check if corpse have exceeded the time period, if it has exceeded, then it will remove the corpse

- **DeadAction class (Action Package)**
    - Extends the Action class
    - This class will implement all the appropriate steps when a Dinosaur is dead. It will check which species of Dinosaur it is then it will add a Corpse of the Dinosaur with the respective remain time and then it will remove the Dinosaur from the current location.

CheckBehaviour

- **CheckBehaviour class**
  - This class will contain all the static method to check all the Behaviours of the Dinosaur
  - If each Dinosaur have met the requirements for each Behaviour, then the respective Behaviour will be added into the list of Behaviour of the Dinosaur.
  - For each Dinosaur (Stegosaur, Brachiosaur, Allosaur), each turn they will use checkBehaviour() to check all of the respective behavior, then it will add the appropriate behavior to the list of behaviour then each turn the dinosaur will only get the last behavior of the list which is the behaviour that it should have at that turn. For example, food level > 50, Stegosaur will have HungryBehaviour and BreedingBehaviour, however, BreedingBehaviour will be executed first, if the food level drops below 50, then the BreedingBehaviour will be removed from the list and HungryBehaviour will be executed.
  - In the checkLayEgg function, since Pterodactyl can only lay egg on a Tree, we will implement a method to check if the Pterodactyl wanted to lay an egg then it will check if the current ground is a Tree or not, if it is then it will lay the egg, if not it will call add a WalkToTreeBehaviour which the Pterodactyl will walk to the nearest tree.
  -

Distance

- **Distance class**
  - This class is responsible to calculate the Manhattan distance between 2 locations.

Dinosaur Capability

- **DinosaurCapability class**
  - This class contain all the Enum representing the GENDER which will have MALE and FEMALE and PREGNANT.
  - Added a new Enum which is STATUS that have FLYING and WALKING to check the current status of the Dinosaur

Lake, water and rain

- **Lake (Ground Package)**
  - Extends Ground
  - This class has a tick method which adds fishes to the lake if the size of the array list of fish is lesser than the lakeCapacity. There is a 60% chance of adding fishes into the lake for every turn. The 60% chance

can be calculated by checking if rand.nextDouble() <= 0.6. If it meets the probability, it will add a fish into the lake for that turn. This class will also check if there is rain or not. This can be done by using gotRain. If gotRain is true then increment the lakeCapacity by the rainfall.
    - gotRain is a Boolean which indicates it is raining if it is true and not raining if it is false.
    - canActorEnter method is implemented to check if an actor can enter the lake. It checks the instanceof the actor and returns a Boolean to see if the actor can enter the lake or not. Stegosaur, Brachiosaur and Allosaur cannot enter the lake. Pterodactyl that is walking also cant enter the lake. But if the pterodactyl is flying it can enter the lake and the method will return true.

- **Fish (Item Package)**
    - Extend Item.
    - A fish class was implemented to represent the fish.

- **JurassicParkGameMap**
    - In order to implement rain, create a new class which extends the GameMap, and override the tick method. Each turn of the game, check if the game have already been played for 10 turns, if it has then implement the probability of 20% of raining, if it is true then get a random number from 0.1 to 0.6 and multiply by 20 and add the number to all the Lake's capacity.

Thirsty dinosaurs

- **ThirstyBehaviour (Behaviour Package)**
    - Extends Behaviour
    - This class will check if the adjacent square of the Dinosaur is a Lake, if it is then it will call the DrinkAction. If not, it will loop through the whole map to find the nearest Lake from the Dinosaur and move one step closer to the Lake.

- **DrinkAction (Action Package)**
    - Extends Action
    - This class will implement the drinking action of the dinosaur. It will reduce the lake total sips by one and add the appropriate thirst level to the respective dinosaur
- Each dinosaur will have a few new variables like a Boolean which check if the dinosaur is unconscious due to thirstiness, a int that keeps track of the unconscious turn due to thirst. This allow us to check if the Dinosaur

is unconscious due to thirstiness or not, if it is, then if the GameMap rain the Dinosaur will be able to regain consciousness and add 10 to it's thirsty level, if the unconscious turn due to thirstiness reaches 15 then the dinosaur will die

Dinosaur (Dinosaur Package)

- Extends Actor class
- This class is an abstract class which contains all the variable and method which a Dinosaur needs. For example, instantiating the Hungry Level of the Dinosaur, instantiating the Well Fed level for different Dinosaur, instantiating the character of the Dinosaur, and have a list of Behaviour which store all the Behaviour of the dinosaur instance. By having this class, we will be able to avoid many redundant codes and allow us to design a more cohesive game which will help us to maintain and update the code and also fix bugs if there's any error in the code more easily.

Pterodactyls

- **Pterodactyl class (Dinosaur Package)**
  - Extends Dinosaur class
  - This class will have an additional method called checkFlying which will check if the Pterodactyl have already fly 30 turns on the map, if it has already fly more than 30 turns on the map then it will have a WalkToTreeBehaviour which it will walk to the nearest unoccupied tree and refuel and then continue flying
- **WalkToTreeBehaviour (Behaviour Package)**
  - Extends Behaviour class
  - This class will check if the current ground is a Tree if it is then the Pterodactyl will be able to fly again by adding the flying capability to the Pterodactyl and call the Refuel Action, if not it will call the findUnoccupiedTree in Pterodactyl class to find the nearest unoccupied Tree.
- **findUnoccupiedTree method**
  - This method will loop through the whole map and find the nearest unoccupied Tree which means that a Tree which doesn't have an Actor on it.
- **Refuel Action (Action Package)**
  - Extends Action
  - This action will remove the walking capability from the dinosaur and allow the dinosaur to fly for another 30 turns

Second map

- In order to allow the player to go to a second map from the north of the existing map, in the Player class, create a new method which check the current position of the Player, if the current position of the Player is at y = 0 which means the most north of the existing map, then return a new MoveActorAction which will move the Player to the second map. And by having this function, each time the Player play turn, we will call the method that check the current position of the player.

Sophisticated game driver

- The sophisticated game driver is implemented in Application.
- There will be a method used to print the start menu for the player to choose the desired game mode which is Challenge or Sandbox or quit. This menu will be displayed at the start of the game. This method will keep track of the choice that the player has chosen.
- There will also be another method to display the menu and keep track of the player's choice.
- A method to reset the player's eco points for each turn will be implemented.
- A QuitAction class is needed to enable the game to have an action to quit. This will allow the player to quit the sandbox or challenge game if they do not want to continue.

Extra Information

Symbol representation

- Allosaur: A
- Brachiosaur: B
- Stegosaur: S
- Pterodactyl: P
- Lake: ~
- Corpse: %
- Bush: *
- Tree: t/T
- Dirt: .
- Fruit: f
- Egg: e
- Player: @
- Wall: #
- Floor: _
- Vending machine: $
- Vegetarian meal kit: v
- Carnivore meal kit: c