Design Rationale and Recommendations to game engine (Assignment 3)

Lakes,water and rain

The park must have a few lakes where dinosaurs can drink water. Each lake covers 1 square on the map. Use the tilde character (~) to represent water/lake and place some pools of water in the map

I have created a class named Lake extending Ground class so that I can use the ground class method (turn()) which will run for the instance of subclass of Ground inside the Application.Inside the constructor I can set the display character attribute in the Ground class(to ~) using the super() keyword

Each lake starts with a capacity of 25 sips. Every 10 turns, with a probability of 20%, the sky might rain which adds water to all lakes. The amount of water that is added to the lake is calculated by multiplying the rainfall by 20 sips where the rainfall is a random value between 0.1 and 0.6 inclusive.

Create an attribute inside the Lake class which accounts for the amount of water in the lake. Have a setwater() method that sets the amount of water inside the lake within the capacity(25). Create a Sky class with static boolean attribute(raining) (indicating if it is raining on specific turn) and static attribute(rainTurns) (storing number of turns passed before last rain) . Have a getter and setter for both attributes.Have a method(process()) inside the Sky class which will change the raining attribute if it has the required probability and sufficient amount of turns.In the Player class run() method, we can call Sky.process() for each and every turn.This allows me to use that raining attribute within the lower level classes instead of doing computation in each and every class to detect if sky is raining.In the turn() method, check if sky is raining using the isRaining() method inside the Sky class and then add the water(calculated using the formula)using the setwater() method and set the number of turns to 1 for water to be added everytime it reaches 10 inside Sky class

Each lake can also hold up to a maximum of 25 fish, each fish providing 5 food points. Lakes start the game with only 5 fish. Each turn, there is a probability of 60% for a new fish to be born (increasing the number of fish of the lake by 1).

Create an attribute inside the Lake class which accounts for the amount of fishes in lake and initialize it to 5.Have a setFishes() method that sets amount of fishes inside the lake within capacity(25).In the turn() method,using random number generator,decide on the probability and if it is equal to 60 percent, then increase the attribute value of amount of fishes by 1

Thirsty Dinosaurs

A dinosaur that is thirsty (water level below 40) should move towards a lake and drink water.
Modify the Following class by having an another boolean variable (toDrink) which indicates if dinosaur is thirsty set to true in dinosaur classes (Stegosaur,Brachiosaur,Allosaur,Pterodactyl) if water level is below 40 and add another condition to Following class using the boolean variable allowing it to move towards the closest lake.Following class already allows us to go the nearest intended location.Therefore, it is easier to extend

A dinosaur should start out with a "water level" of 60 out of a maximum of 100. Brachiosaurs have a maximum capacity of 200
In Dinosaur class,create two attributes (waterlevel and maxwaterlevel) and Intialise it within the constructor given as argument.In Dinosaur sub classes constructor ,use the super keyword to intialise the two attributes with their defined values.Only Brachiosaur will have maximum capacity of 200.

The water level should decrease by 1 on every turn. If the water level gets to zero, the dinosaur becomes unconscious and cannot move or act unless it is watered. After 15 turns of unconsciousness, if no rain occurs, the dinosaur dies. Each time the sky rains, unconscious dinosaurs due to thirsty must get a water level equals to 10 and return back to normal
Have a set and get method for water level inside Dinosaur class.Decrease the value for water level using set and get method for each dinosaur subclasses in their turn() method.Modify the inherited isConscious() method inside the Dinosaur class which returns true if both food level and water is greater than 0 .I am reusing the method from the Actor class so that I do not have to modify my subclasses of Dinosaur as they use the same method for checking consciousness. check if sky is raining using the getRaining() method inside the Sky class, add in their state of unconsciousness, add 10 to their water level.

Turning towards a lake should increase the dinosaur's water level by a maximum of 30 water-level points from each lake at a time (per turn). Brachiosaurs can increase their water level by 80 points in one turn. When a dinosaur becomes thirsty, a suitable message should be displayed (e.g. Stegosaur at (19, 6) is getting thirsty!).

Create a class Drink which takes the instance of the dinosaur and current location as an argument to its constructor.Have a method inside the drinking class (drink()) and in that method it check if in Exits to that current location is an instance of lake, then increase the water level points by 30 but do by 80 for Brachiosaur using the dinosaur instance.This class is created so that it can be reused by each and every dinosaur and so that implementation is easier to extend.Inside the dinosaur subclasses, Have an if condition for printing out if dinosaur is thirsty.Create a interface

DinosaurFunctions where we can have functions like (drink()).Create a class DinosurFunctions that implements this interface.Have a reference for DinosaurFunctions class inside each dinosaur class .Create a object of Drink class inside the main Application class and pass it to each constructor of Dinosaur

Land-based creatures cannot enter water, and trees/bushes cannot grow there. If you find that your dinosaurs need more water than they can drink from lakes and you don't want to (or don't have time to) experiment with animals' water capacity, you can add a "water bottle" to the vending machine, but you are not required to do this.

Inside Lake class, In the canActorEnter() overridden method , check if actor is instance of land dinosaurs , then return false otherwise return true.We have this method because it is used in Following class to determine if dinosaur will move

A more sophisticated game driver

1-Create a GameMode interface that has two methods

- checkGameFinished() that takes in one parameter which is Player class instance and returns true if game is finished
- getMessage() which returns the message in relevance to game mode

2-Create a Challenge class implementing GameMode

- In checkGameFinished() method, have three attributes
    1) maxNumberOfMoves(moves given by user)
    2) maxNumberOfEcoPoints(ecoPoints given by user)
    3) currentNumberOfMoves(moves that are passed in the current game)
    4) message(display)

    Using player instance ,we can get the number of eco points that can be matched with max number of eco points.If they are equal,check if the current number of moves are less than or equal to max number of moves ,then update the message variable indicating player has won otherwise update the message variable indicating player has lost.In either case method will return true .else if ecopoints are not equal, then increment current number of moves and return false

3-Create a Sandbox class implementing GameMode

It will have no attributes.The checkGameFinished() will return false as the normal game is based on player,s survival.the getMessage() method will also return null.We created this class as in the future we will might need for the game to become harder for player so we can extend this class

4-Player can quit as well so in order to allow that,Create a Quit class extending the Action class.Inside the execute() method ,we can remove the player from the location which will exit the game(map.removeActor(actor)).Inside the menu description method ,We can

return string asking player if we wants to quit the game.This action class can be added to Player list of actions inside Player,s playTurn() method at the end of every turn
5-Create a method inside the Application class which prints the menu asking player to choose between different game modes.(Sandbox and Challenge in this case) .call this method inside main method in Application class before starting game via world.run() method.Create a GameMode reference variable.Change the constructor of the Player so that it accepts GameMode type variable.Player inputs the chosen gameMode.If Challenge is chosen player is asked to input the number of eco-points and number of moves ,Then create an instance of Challenge class with these parameters and assign instance to GameMode reference.Otherwise if Sandbox is chosen, assign that instance to GameMode reference.After then pass that reference variable object to Player constructor while creating Player instance.Interface is created to induce abstraction so that Player only needs to know about the interface.It is also done to not allow dependencies in Player class by creating instance of Challenge and Sandbox outside the Player class and passing into its constructor.It also ensures that we can add more classes implementing the interface ,thus having more game modes and any extension that we make to the Challenge or Sandbox would not affect the Player class
6-Inside the Player class playTurn method ,we can use the reference that was passed to call the checkGameFinished() method and if it is true, we can call the getMessage() method ,print it out and remove the Player from map which will exit the game
7-Inside the Application class main method,regardless player wins ,loses or quits , we can insert a loop around the code inside the main which terminates if user do not wish to play another game which we can check after the world.run() method exits by asking for player input if they want to play another game

Second Map

Add a new map to your park. This map should be located to the north of the existing map. When you go north from the existing map, you should end up at the south edge of the new map. The map should be the same size as the existing map.

For the second map, we will first create another arraylist of Map that is similar to the original map. Then we add the portal to the north of the first map and south of the second map. The portal is a class that extends ground, it has the ability to "teleports" the player back and forth from the first map and the second map. In the Portal class, the allowableAction method will check that if the actor is a player, then the action that moves the actor to a different map will be added to the list of actions.

Pterodactyls

Pterodactyls are small flying dinosaurs. They can traverse water squares. In fact, they can eat fish which they would catch from the surface of the water by dipping their long beaks in as they fly over. Every time they fly over any lake square they can catch 0, 1 or 2 fish and increase 30 water points.

For pterodactyls we will have a capability of FLY, when pterodactyls have the capability to fly it can traverse water squares and catch fish/drink water. If they are walking then they can't catch fish since they are not traversing the water squares. To determine the amount of fish they can catch we set the default upper bound of random int to 3 so that it generates a number between 0 to 2. If the fish in the water square are less than the upper bound(3) then we make the upper bound to become the amount of fish in the lake.

For Feeding, when Pterodactyls find a corpse it will eat it and heal for 10hp, at the same time they reduce the corpse duration by 10. If the corpse duration reaches 0, it is removed from the map.

For breeding, an extra condition is added to Pterodactyls' breeding which is they will only lay their egg on a Tree. When their pregnant count reaches 15, they will start to move towards a tree.

For Pterodactyl there will be a getFlyingDuration() method to check how long they can fly, their flying duration decreases by 1 every turn. When it reaches 0, we will remove the FLY capability of it and set getFlyingStatus() to false. When getFlyingStatus is false, Allosaur can eat it as a whole and recover hp, in this case Pterodactyls will be removed from the map. If getFlyingStatus is true, Allosaur can't attack Pterodactyls because they are flying.

Recommendations to existing game engine

Potential Flaws

1- hit points attribute in the Actor class has privacy level set to protected which means it can be accessed by its subclasses and can lead to accidental changes as there is no guardian code such as setter around it which can assign an appropriate value(like if hitpoints become negative).Therefore such adjustments need to be done in subclasses which involves a lot more effort.Therefore,such attributes need layer of encapsulation by setting them to private and create a public setter,getter around it.It will also help future developers not to know of the bounds that are set to hitpoints and can use its setter and getter thus providing a layer of abstraction

2-The Actor class has an inventory attribute however only Player class uses it in our application and probably will do it in future as well.In our design ,we are intending to reduce responsibilities handle by a single class so that it is easier for us extend and as evident from the breakdown of classes as Player and Dinosaur class extends the Actor class.We can just move the inventory function and the attribute to the Player class

3-getIntrinisicWeapon() and isConcious() methods in Actor class should be in the subclasses as getIntrinisicWeapon specifies the damage which can be different for different species and isConcious() depends upon the biological characteristics of different species.Therefore,the Actor class should not handle methods that it is not concerned with rather these methods should be placed in an interface implemented by the subclasses.This way we would not be overriding the methods to and we can have separate implementations for them in each class.

4-ActorLocations class add method throws an IllegalArgumentException if an actor is already at the particular location.This exception should be handled by the Location inside addActor method and should display a message indicating the error as this method is frequently used in our game package.This way it will be easier for developer to understand and make changes accordingly
5-In The Ground class, displaychar is set to protected which as well does not adheres to the principle of encapsulation, we should set it to private and make use of efficient getter and setters in our subclasses

6-The showMenu method inside the Menu class puts the value against the action of Player in a HashMap and asks for input of the user corresponding to that action and and there is a do while which will only exit if their exists an action corresponding to the input however it is possible that there are no possible actions such as when the game is ending, (you lost or you won) but it will still ask for input no actions will be shown on the screen and user will be ask to enter input which will put program into an endless loop.Therefore there has to be assertion statement inside the function which can aware the programmer of the bug in the code in response to some unexpected situations adhering to the Fail fast principle.When we know this is a bug,we can insert an if condition outside of the while loop and only enter the loop if the (actions.size()>0) otherwise return an instance of DoNothingAction class

<u>New functionality</u>

1-Introduce random events

Although the game itself is already playable, it does not seem to be attractive from the gameplay wise, what we could do is to introduce random events that could happen throughout the game(e.g. Stepped on a trap, found a chest, lightning struck, food stolen etc). For example ground.allowRandomEvent(True), then an event is chosen at random from a list of possible events. The random events should either give benefit to the player or give drawback to the player. By doing so, we can generate multiple possibilities for the player to explore throughout the game.

2-Introduce experience system and ability system

In the original game, the player's status will always remain the same and the weapons that they use have fixed damage. So, we could introduce an experience system like other role playing games where the player can level up through various activities. At the same time, we can introduce an ability system that allows the players to customize their character based on their own preference(e.g. Hp, power, speed). By levelling, it gives player ability points that can improve the player's overall ability or unlock new abilities that allows the player to explore places that are previously inaccessible to them.