Design Rationale

Assignment 3

Lakes, water and rain

Because lakes are on the ground, so I create a Lake class extends Ground class.

For water and fish, at first I create Water class and Fish class and create their objects, but the engine can’t support so many objects on the same square (the game will freeze), so I switch to integers, and lakes themselves are drinkable by all dinosaurs and eatable by Pterodactyls, just minus water and fish integers every time they are being eaten.

For rain, I create a rain boolean, with a random determinator, every time it rains, the boolean becomes true for that while then add random number sips of water to lakes and wake unconscious thirsty dinosaurs up.

Thirsty dinosaurs

I create a water level for dinosaurs, when water level reaches 0, dinosaurs will become unconscious, drinking water increases their water level.

Also, I create a DrinkAction class, all Behaviour classes need to refer this class. To let dinosaurs drink water when they stand next to a lake, I also remove previous restriction on eating only on the same square only for drinking water.

Pterodactyls

I create a Pterodactyls class extends Dinosaur class, a PterodactylsEgg class extends Egg class, a PterodactylsCorpse class extends DinosaurCorpse class and a BabyPterodactyls class extends BabyDinosaur class, also a PterodactylsBehaviours class extends Behavior class for their unique behaviour.

Second map

First I create a second map then addGameMap, but the tricky part is how to let Player move between the maps, so I create a Passage class for a passage item, add it on two maps, this passage item let Player transport between the maps on the specific location.

A more sophisticated game driver

For the player to choose the game mode, I create a Option method to receive player’s input. It is easy to create the sandbox mode because it does not change anything, but the challenge mode is something else. To create the challenge mode, I override processActorTurn(Actor actor) in World class to update player’s turns, also override stillRunning() in World class to check if the number of player’s turns exceed turn limt or player’s eco points reach target eco points, at last I need to override endGameMessage() in World class to display if the player wins or loses at the end of the game.

But the main method in Application class is static, it cannot use these overidden methods, so I move all contents in Application class and all these overidden methods to a new class called Mode, then I override run() in World class and super it. In the main method I create a Mode object then run it. Now the program can use all these overriden methods.