FIT2099 Design Rationale

Michael:

Having Dinosaur as the parent abstract class and then having abstract classes Carnivore and Herbivore inherit it reduces the need for repeated code (Don't Repeat Yourself) and data members for all dinosaurs. Doing so will also future proof the design as adding dinosaurs is now as easy as inheriting either the Carnivore or Herbivore class.

Classes are responsible for their own properties with Behaviour and Action classes explicitly being responsible for certain behaviours and actions that dinosaurs can take. Attributes only used in that class will only be declared in that class and variables used within methods only will only be declared within that method, declaring things within the tightest possible scope. This keeps classes very simple.

All behaviours also inherit from the Behaviour interface, once again removing the need to repeat code in each behaviour class. The same goes for action classes that make use of the abstract Action class code (HatchingAction being the exception, explained in the breeding section).

Some behaviours also make use of the same actions, such as UnconciousBehaviour and DyingBehaviour making use of the same DieAction class. This once again avoids the use of repeat code.

While Dinosaur don't use HatchingBehaviour, HatchingBehaviour still inherits the Behaviour interface as it still uses Behaviour's code. To combat the use of Dinosaur being able to use the HatchingBehaviour class, there will be exceptions to make sure Dinosaur will not be able to use the HatchingBehaviour class at initialisation, following the fail fast design principle. Despite being called HatchingAction, HatchingAction does not inherit from the Action class as it does not make use of the code within the Action class.

For dinosaurs to grow, a behaviour or action was not introduced. Instead there will be a boolean indicating if the dinosaur is an adult or not and after 30 turns the boolean will turn from false (for baby) to true (for adult). This reduces the need for extra classes, thus reducing dependencies.

The AttackBehaviour and AttackAction are part of Dinosaur rather than Allosaur since there are many other dinosaurs that can attack (including possibly the stegosaur). This reduces repeated code and supports easy extensibility and maintainability.

Stegosaurs possibility won't have all the behaviours and actions given to the Dinosaur class. This can once again be ensured with the use of exceptions to make sure non-Stegosaur behaviours and actions aren't included at instantiation, following the design principle of failing fast.

The abstract class EatAction is inherited by HerbivoreEatAction and CarnivoreEatAction. The two classes will use the code of EatAction except only take in HerbivoreFood and CarnivoreFood respectively. This eliminates repeated code while separating what herbivore and carnivore dinosaurs can eat.

MoveToFoodAction applies to both Herbivore and Carnivore dinosaurs so a parent class of CarnivoreFood and HerbivoreFood was made, called Food. MoveToFoodAction will take Food as a parameter meaning CarnivoreFood and HerbivoreFood can also be used as arguments, making use of polymorphism and eliminating repeated code.