# Design Rationale - Edited

Name of the project – Assignment 2.

Team name: PJs.

Team members: Joleen Tang, Pasan Jayanetty.

A few amendments were made for the design for the project. The purpose of these amendments is to increase the quality of the overall design of the project. This document points out the most important design features implemented by the team.

The following superclasses were created:

* FoodItem
  + A superclass for Items that are edible.
* SearchBehaviour
  + A superclass containing different methods to search for surrounding locations that are more than one square away.
* Dinosaur
  + A superclass (extends Actor) which contains attributes that are important to individual Dinosaurs in the map. foodLevel etc.

Purpose of creating the aforementioned superclasses:

* Classes that have similar functions can significantly benefit from having a superclass. Inheriting from a superclass results in reduction of duplicating code written for the same function.
* Some classes share the same attributes and methods. In this case, having a superclass erases the need to re-code every similar class thereby adhering to the DRY principle.

Another hurdle the team was faced with was to choose between TreeMap and a HashMap to store prices for respective items that can be purchased from the VendingMachine.

The team decided to use a HashMap for VendingMachine prices.

* HashMap was implemented since HashMap does not attempt to store its elements in an order.
* The team decided that having an unsorted map would be better because the vending machine stores instances of items as keys which are not comparable to each other.

Getters and setters were used throughout the project.

Eg: private int getFoodLevel() in Dinosaur.

The purpose of using getters and setters is to ensure ‘Encapsulation’. This practice can protect sensitive data like foodLevel and hides it from other users. These data can only be accessed by getters and changed by setters from other classes. Further, this ensures that unnecessary modifications to important data does not occur.

UML class diagram link:

<https://app.lucidchart.com/documents/edit/56bdc45f-2b87-4e27-8ef1-0c36aa98714a/o8dxuE6i5s.Q?shared=true#?folder_id=home&browser=icon>

UML sequence diagram link:

<https://app.lucidchart.com/documents/edit/5c2cb2b5-957d-4fa7-a288-df11f3244dec/.b.x1VccSAc7?shared=true#?folder_id=home&browser=icon>

# Dinosaur

Dinosaur stores behaviours shared by both allosaurs and stegosaurs in a static TreeMap. This ensures that both species can access shared behaviours without duplication and removes the need to constantly construct new ones. The only behaviour not included is FollowBehaviour as that requires a different target every time it is constructed.

The playTurn method for both species is very similar. The dinosaur superclass contains the stegosaur playTurn method, which contains 3 different ‘components’. The ‘passiveAct’ component returns Actions that both species would do such as die, lay an egg, or do nothing while unconscious. Similarly, ‘breedAct’ covers breeding behaviours that are shared between the two. Finally, ‘foodAct’ is different between the two; unlike stegosaurs, allosaurs will hunt. To cater to this difference, allosaurs will override the getFoodAction method and apply that instead of the default dinosaur one. However, implementing playTurn in the superclass eliminates the need for extremely similar code in both species’ classes.

# PlaceVMAction

A new action, PlaceVMAction() was created to put a vending machine on the map. Previously, the vending machine existed in the system but had no way of being instantiated.