Introduction

Jon Minton. E63

This document addresses the task listed in A.D.2

Problem

The task is as described below:

Include in your screenshot:

- A class diagram

The class diagram must contain:

- At least two classes contained in rectangles with the title of the class at the top (capitalised)
- At least one attribute for each class with types indicated (e.g. “name: string”, “age: int”)
- Any methods that each class has
- Lines or "relationships" to indicate the structure of the classes and how they interact with each other. Types of relationships, such as many-to-many, do not need to be indicated at this level

Submit a comment explaining your understanding of:

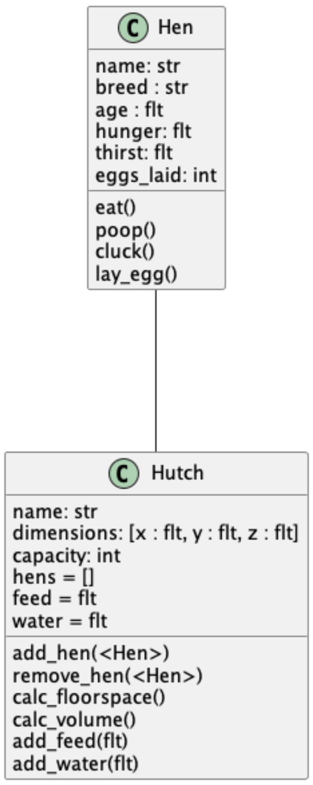
- The class diagram you’ve submitted and its use
- Design decisions made when producing the diagram

Remember:

- Attach examples as image files or screenshots
- Use original work that is your own (i.e. no examples handed out as course materials (unless PDA specific), copied from the internet, or submitted by another student)
- Reattach all files on any resubmission
- Do NOT reuse examples from another submission

Solution

The following image is a class diagram meeting the requirements of the problem:



Description of class diagram and design decisions

The class diagram has two classes, `Hen` and `Hutch`.

The Hen class has the following attributes and methods:

- Attributes:
  - name
  - breed
  - age
  - thirst
  - eggs_laid
- Methods:
  - eat()
  - poop()
  - cluck()
  - lay_egg()

The types of the attributes in the Hen class are indicated in the class diagram. The attributes name and breed are of type string, age and thirst are floats, and eggs_laid is an integer as only discrete whole numbers of eggs can be laid.

The methods do not take any arguments. When the lay_egg() method is called the eggs_laid field will be incremented by 1.

The Hutch class has the following attributes and methods:

- Attributes:
  - name
  - dimensions
  - capacity

    - hens
    - feed
    - water
- Methods:
    - add_hen()
    - remove_hen()
    - calc_floorspace()
    - calc_volume()

The types of the attributes in the `Hutch` class are indicated in the class diagram. The attribute `name` is a string. The `dimensions` attribute accept three float values to provide the width, depth and height of the hutch object created. Capacity is an integer and indicates the number of hens the hutch can include.

The methods `add_hen()` and `remove_hen()` add to and remove an element from the `hens` list. Each element in the `hens` list is intended to be an object of class `Hen`. The `add_hen()` method should not accept a new hen if the hutch is already at capacity (i.e. the length of the `hens` list is equal to the `capacity`), and the `remove_hen()` method should do nothing if the hen proposed as an argument is not in the `hens` list. The methods `calc_volume()` and `calc_floorspace()` both make use of the contents of the `dimensions` attribute, though the `calc_floorspace()` method only makes use of two of the three elements within dimensions (i.e. its depth and width, but not its height).

The two classes are related to the extent that the `hens` attribute of the Hutch class is intended to receive objects of class `Hen`. A line is shown between the two classes to indicate this relationship. n.b. the direction of the line and relative position of the two classes are **not** intended to indicate a hierarchical relationship between the two classes. (i.e. neither class inherits from the other class.)

Another way in which the classes could also be related would be to reduce the amount of feed in the hutch when a hen inside the hutch feeds. The `add_hen()` method of `Hutch` could also have additional conditions, such as only allowing hens of a single breed to enter. A method for updating the capacity based on the type of breed (with some expected to be larger than others) could also be added to the Hutch class.

Appendix

The following shows the PUML code used to generate the class diagram.

@startuml class Hen { name: str breed : str age : flt hunger: flt thirst: flt eggs_laid: int eat() poop() cluck() lay_egg() } class Hutch { name: str dimensions: [x : flt, y : flt, z : flt] capacity: int hens = [] add_hen(<Hen>) remove_hen(<Hen>) calc_floorspace() calc_volume() } Hen --- Hutch; @enduml