Project 1 Design Proposal

Document Author(s): John Ruisi

Date: 9/23/16

Design Rationale

For Project 1, I decided to use seven classes, and three interfaces in order to support the functionality dictated by the design requirements.

The main object being used in this design is that of type Animal. Everything in my design is based on adding functionality to the Animal object. Whether it be interfaces or subclasses, I will describe how they all help to bolster the functionality of the Animal class.

My abstract class, Animal, holds common code and declares abstract behavior in efficiently change behavior. There are three subclasses to the superclass "Animal", the first being java class "AnimalHigh". This class allows for the creation of an object that is an animal that is of the highest food chain. This class is needed due to the fact that animal high on the food chain have different characteristics than that of middle or low animals on the food chain. The other two classes, "AnimalMid" and "AnimalLow" were created for the same reasons. For example, we needed our own subclass for "AnimalLow" due to the fact that animals low on the food chain do not eat, and there for have different characteristics which are shown in the default values for each animal type. (Ex. The default value of the need to breed is different from high and low animals). The relationship between these classes are important since the creation of the subclasses allow for an inheritance relationship between the superclass and the subclass, thus allowing for better efficiency in my program.

There are three interface classes that check the states of the animals and assist in the process of deciding what the animal is going to do. The first interfacing class is "Eatable". This class examines the characteristics of two different animals, and decides whether or not one is able to eat the other. The second interfacing class is "Movable". This class examines an animal and its environment and decides if it is able to move to different units on the array. My third interfacing class is "Breedable". This interfacing examines two adjacent animals, examines the characteristics of both animals decides if they are able to breed based of the examination. All three interfacing classes allow for a quick examination that allows for other classes to be able to make decisions on the actions of a specific animal for each iteration. All three interfacing classes have an inheritance relationship with the class "Animal" which is necessary to examine the characteristics of the different animals.

I have one class that is dedicated to reading data from files that is crucial for the creation of the ecosystem simulation. That class is "EcosystemConfigIO". This class is responsible for reading input given by the user that details the configuration of the simulation. The class has two methods, "readEcosystemInfo" and "readEcosystemArray". The first method reads the first five lines of the input file This dictates how many animals are in the simulation, where they are located in the food chain, etc. The second method turns the data from the file into a two-dimensional array. Turning the data into the array is key for being able to run through each animal when we update the ecosystem after every iteration.

The view/controller portion of the MVC in my project is "EcosystemGUI" and the model portion of the program is every other class and interface in my project. The main model that interacts with the VC is my "EcosystemsArray"

class. This class keeps track of each animal and their state in a two dimensional array of animal objects. The method responsible for interacting the GUI is the method “getEcosystemArray” which allows for the displays of animal objects in a two dimensional array.

The method “EcosystemArray” is a composition relationship with the class “Animal” since it is unable to exist separate from the class “Animal” due to the fact that “EcosystemArray” is only interested in using objects of type Animal.

The main limitation that hinders the design of this program is that we are restrained to a twenty by twenty square box where different animals are able to make actions. This calls into question how realistic the simulation of this program will be since it is known that ecosystems do exist in small squares. Another limitation of the design are that animals have very specific actions that they make. The requirements to complete these actions are very specific and are not true to how animals interact in the real world. A constraint of my system is that some animals have favorable positions. Since the program does not make moves for every animal at the same time, it must go through one by one to make decisions for the animals. This causes an unfair advantage for those animals that get to make moves first.

Design patters are important to this project in order to complete all the required functionality. I have classes that use iteration patters, such as “EcosystemArray” that iterate through each point in a 2D-Array to gather data and update data. I have also made certain structure patterns that allocate certain responsibilities to certain classes. For example, I create the class “AnimalUpdater” that is dedicated to change certain characteristics of an animal’s state. Allowing to have many classes that do specific things allows us to use polymorphism to make our programs more efficient

Figure 1: Class Diagram for Project 1

Document Revision History

Date	Author	Change Description
9/22/16	John Ruisi	• Added skeleton structure of design rational
9/23/16	John Ruisi	• Added detailed information to skeleton of rational