Assignment 6
Due: Aug 10, 2023 at 11:59PM

1 Overview

1.0.1 Goals

1. Practice writing C#'s object-oriented code
2. Practice constructing a simulator
3. Practice designing an ecosystem with approximate equilibrium

The purpose of this assignment is to implement a simulator of a pasture's ecosystem. There is a lot of supplied code already, and your task is to modify it extensively to achieve the specifications below. You have the liberty to change or add any class, as you see fit.

A pasture is simplified to be a rectangular area that is divided into a grid. The pasture will have various entities in it, including inanimate objects, plants, and animals.

Each entity in the pasture occupies exactly one grid square. Several entities may share the same square if the behavior of all of the entities in the square allow it.

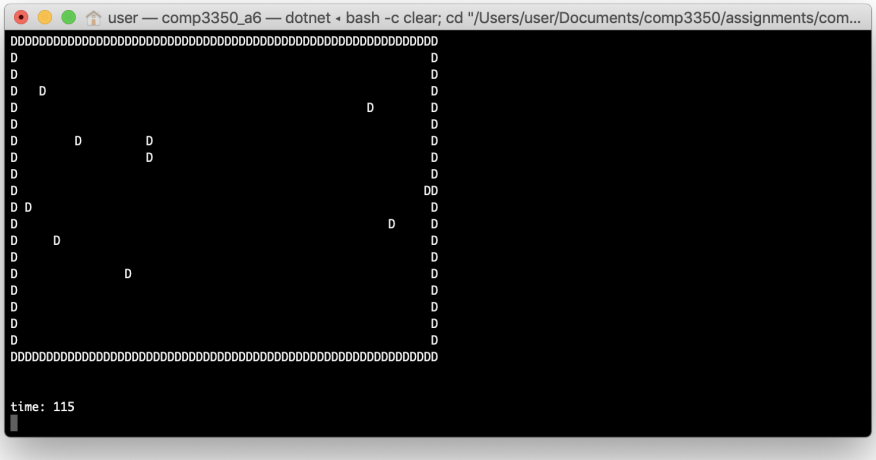
Inanimate objects reserve space by occupying a square and not allowing anything else to enter the same square. Fences are the only type of inanimate objects in this simulation. Fence objects are used to prevent other objects from entering a square.

Multiple plants/animals of the same kind can't occupy the same square, but different kinds can.

Plants are entities that reproduce at a fixed rate whenever there is room for them to grow. Grass is the only plant in this simulation, and it reproduces by division. It may grow into an adjacent grid square if the square has no fixed objects or grass. Grass only dies if it is eaten.

Animals are entities that reproduce (by division) at a rate depending on the type of animal. They can also move around in the pasture, at different speeds for different types of animals. They can move to an adjacent square – either vertically, horizontally, or diagonally. For this simulation, animals should move by picking a random legal direction. Animals must also eat to survive, which they should do automatically when they share a square with their food. Animals can die by being eaten or by starving.

For each entity, implement that class. If you find it helpful, you may implement a base class that the required classes inherit from. Work incrementally. Try adding one entity at a time, and testing its functionality before moving on. There is a **Dummy** class that has some useful features and some features that are not a very good object oriented style – for example, dummys have an 'alive' boolean value that would be better represented as two separate classes.



2 Assignment 6 Specification

The skeleton code at `https://classroom.github.com/a/nA6c3avW` is the start of this week's assignment.

2.1 Required Classes

2.1.1 Fence

Fences are inanimate objects. They do not move, eat, starve, or reproduce. Nothing else can occupy a square with a fence in it. The full perimeter of your pasture should have a fence. The skeleton code lines the perimeter with dummys, which you should replace with fences. Use the 'F' character as a fence's symbol.

2.1.2 Grass

Grass spreads after a certain time, provided that there is a free adjacent square. When grass spreads, a new grass should be placed in an adjacent square (vertical, horizontal or diagonal). Grass does not eat, starve, or move. Grass is a food so it may be eaten – it should be removed from the pasture if this happens. Use the 'g' character as the grass symbol.

2.1.3 Cow/Tiger

Animals – cows and tigers – move around the pasture. If they occupy the same position as their food, they eat the food. Eating should reset their starvation counter, and remove the eaten food from the pasture. It takes a certain amount of time for the animal to move one square and an animal can live a certain amount of time without eating before it dies of starvation.

An animal reproduces after a certain amount of time in a manner suspiciously similar to grass dividing. Animals can only reproduce if there is a free adjacent square. They live until they die of starvation or get eaten. When they die, they should be removed from the pasture.

The following parameters must be given for each animal species:

- The time required to move between two squares
- The time an animal can live without food
- The time before an animal reproduces

Tigers eat cows and cows eat grass. A tiger and grass can safely occupy the same square. Use the 'T'

character as a tiger's symbol, and 'C' as a cow's symbol. You can decide how to depict multiple entities that occupy the same grid square.

2.1.4 Other Modifications

You will have to modify either the **Pasture** or **Engine** class to implement starvation/eating/reproduction. You should also modify the starting count of the creatures in the pasture class' private fields. You may change any other parts of the code that help the pasture simulation work.

3 Testing

The **Pasture** class has the `Main` method - the entry point of the program. The engine has a variable for the simulator speed, which you can change for testing.

This assignment is more open-ended than previous assignments. The **primary goal** is to implement all the required behaviors for the entities. The **secondary goal** is to find rates of reproduction/starvation/movement that lead to a stable ecosystem. You don't want one entity which quickly dies out or overruns the pasture. Eventually one entity will probably dominate because balancing an ecosystem is tough, and our simplified simulation makes it tougher.

Experiment with different rates to find an approximate balance. (If you choose a slow reproduction rate, you will probably want a slow starvation rate too.) **All living species should exist in the simulation for at least 5 generations.**

If you want to add more creatures or simulator mechanics, feel free to experiment after the required sections are implemented.

4 Grading

Be professional. Make results easy to understand and grade. Include only those parts to be graded. Write comments in each class and for specific methods if they are confusing or complicated.

Your grade will be based on your implementation of correctly-behaving entities as well as the ecosystem balance.