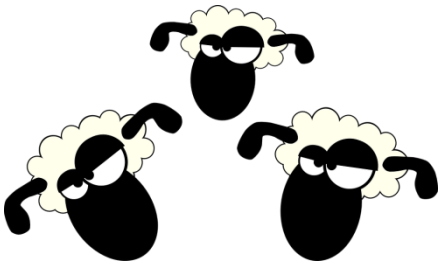

Unit 2 - OOP - Eco-System Simulator

This program will demonstrate the relationships between animals in an eco-system.
(From the point of view of computer programmer. #LimitedKnowledge)

You will be created a small environment with both predator and prey. Once the simulation begins we will be able to see what happens over time based on the initial conditions we provide.

Details

- Our simulated world is an N x M grid. (2D array, start with 25 x 25)
- Three different objects can exist on our grid: plants, sheep, and wolves.

Plant
- spawn at a random time in a random position in the grid
- contain a nutritional(health) value

Sheep
- are placed in the grid at the beginning of the simulation
- have a set health
- can eat plants

Wolf
- are placed in the grid at the beginning of the simulation
- have a set health
- can eat sheep

As time Progresses

- Sheep and wolfs move randomly throughout the grid (each turn they can move or not)

- Every turn sheep and wolves lose 1 health, (0 = death)

- If a sheep moves onto a plant it consumes the plant and the sheep's health increases by the nutritional value. (The plant is gone)

- If two animals (with health >20 each) collide they create a new animal that spawns randomly on the grid. The new animal has a health of 20 and both parents lose 10 health. (no space = no new animal)

- If a wolf moves onto a sheep it consumes the sheep. (The wolf nutritional value increases)

- If a wolf moves onto a wolf it damages the weaker wolf health by 10. (Wolves are *Comparable*)

Object Oriented Design Principle	Usage
Objects/Classes	Wolf, Sheep, Plant Objects are instantiated and used. Program structured with OOD in mind.
Inheritance	A hierarchy between classes exist as they share common characteristics
Encapsulation	All class variables are private and access methods exist
Polymorphism	All objects are placed in a grid of type *superclass*. Methods are called correctly thanks to dynamic dispatch
Abstract Classes/Interfaces	Abstraction is used to define superclass used to reference subclasses. All wolves should implement the comparable interface.
Exceptions	Optional usage somewhere in the program

Tips

* At the beginning of the program you should ask the user for the initial configuration: plant spawning rate, health, number of wolves, number of sheep, plant nutritional value, spawns health, etc…

* Try to experiment to see how the number of wolfs can be adjust to control the sheep population

* Your program should run continuously (after initial set has been input) and output the numbers of each species, and the number of turns that have occurred.

*Your program should end if any of the species become extinct

* You will likely have to add/remove/modify to make this simulation a success

* DO NOT FORGET TO USE OBJECT ORIENTED DESIGN PRINCIPLES *