HW # 20

This homework is a larger project and will have multiple submissions. The goal is to create a simple 2D predator-prey simulation. In this simulation, the prey are ants and the predators are doodlebugs. These critters live in a world composed of a 20x20 grid of cells. Only one critter may occupy a cell at a time. The grid is enclosed, so a critter is not allowed to move off the edges of the grid. Time is simulated in time steps. Each critter performs some action every time step.

The ants behave according to the following model:

- Move. Every time step, randomly try to move up, down, left, or right (i.e., the attempted direction is random). If the cell in the selected direction is occupied or would move the ant off the grid, then the ant stays in the current cell.
- Breed. If an ant survives for three time steps, then at the end of the third time step (i.e. after moving) the ant will breed. This is simulated by creating a new ant in an adjacent (up, down, left, or right) cell that is empty. If there are multiple adjacent cells that are empty the choice of which one will be done at random. If no empty cell is available, no breeding occurs. Once an offspring is produced, the ant cannot produce another offspring until three more time steps have elapsed and the new ant must survive for three time steps until it can breed..

The doodlebugs behave according to the following model:

- Move. Every time step, if there is an adjacent cell (up, down, left, or right) occupied by an ant, then the doodlebug will move to that cell and eat the ant. If more than one adjacent ant exists the one chosen will be at random. Without an adjacent ant, the doodlebug moves according to the same rules as the ant. Note that a doodlebug cannot eat other doodlebugs.
- Breed. If a doodlebug survives for eight time steps, then at the end of the time step it will spawn off a new doodlebug in the same manner as the ant.
- Starve. If a doodle bug has not eaten an ant within the last three time steps, then at the end of the third time step it will starve and die. The doodlebug should then be removed from the grid of cells.

During one time step, all the doodlebugs should move before the ants.

The world will be represented by a Swing created grid with icons to represent a blank cell, an ant in a cell, and a doodle bug in a cell.

Design and code a program to implement this simulation.

Because object oriented programming is a major concern of this class, an object oriented approach is required. Critters will be objects (ant objects and doodlebug objects), the world will be an object, etc. There likely will be a class named Critter that encapsulates data and methods common to both ants and doodlebugs. Critter should have an overridden method named move that is defined in the derived classes of Ant and Doodlebug. World should have a method named toString() that will display an ASCII character based representation of the world that will be used to document a series of simulation times.

Initialize the world with 5 doodlebugs and 100 ants. After each time step, prompt the user to press enter to move to the next time step. You should see a cyclical pattern between the population of predators and prey, although random perturbations may lead to the elimination of one or both species.

The main class for your project, and hence the name of your .java file, must be named Doodle.

Additional requirements for this homework will be announced in class.

NOTES:
Don't try to do the whole thing all at once. The first step is to design your software. Then you might proceed to code and debug in this sequence:

0) Note that the size of the "world" can be established with parameters at the top of the program. I find it easier to initially work with a smaller world, say 10 x 10.

1) Get World's constructor(s) working and it's toString method that will display the current state of the world. Create a critter class and sub classes for Ants and Doodlebugs. Then create a world, call toString to display it, and count the ants and doodlebugs to make sure the right number are there.

2) Get the Ants moving properly. Have a method(s) to move one ant and then a method that will try to move all ants.

3) Get the Doodlebugs moving. First have them move only into empty spaces. Use the same strategy as for the Ants

4) Get the Ants breeding. Each Ant should breed if it has survived for three time steps (there will need to be variables in the critter class to keep track of life time).

5) Get the Doodlebugs breeding. Don't worry about eating or starving yet.

6) Get the Doodlebugs eating. They should move onto adjacent ants.

7) Implement and test doodlebug starvation.

   Run Doodle. Watch closely to see that unfed doodlebugs are starving, as appropriate. (You may find it helps your debugging to have a doodlebug be a single digit equal to the number of generations since it has eaten).