CS 2336 – PROJECT 2 – Ant Wars

Pseudocode Due: 2/7 by 11:59 PM
Project Due: 2/22 by 11:59 PM

Submission and Grading: All programs are to be submitted in eLearning. The pseudocode should be submitted as a Word or PDF document and is not accepted late. Projects submitted after the due date are subject to the late penalties described in the syllabus. Programs must compile and run in NetBeans 8.0.2. Each submitted program will be graded with the rubric provided in eLearning as well as a set of test cases. These test cases will be posted in eLearning after the due date. Each student is responsible for developing sample test cases to ensure the program works as expected. **Type your name and netID in the comments at the top of all files submitted.**

What to Submit: Save all project files in a directory named with the following format: <netID>_Project<number>. This directory should contain all files associated with the project including the project file itself. Zip up the directory into a standard Zip file. Please do not create a RAR or 7-Zip file, just a .zip file.

Objective: Implement object-oriented programming in Java with inheritance and polymorphism.

Problem: Mumbo Jumbo (a developer located in Dallas) has begun working on a new online game called Ant Wars. The game is a simple, casual game where players try to avoid their ant colony being destroyed by an invading colony of beetles. The player will have an opportunity to play against an AI or against another player. You have been assigned to help develop the AI part of the game.

Classes

- Base class
    - Creature
        - Methods
            - Move (virtual)
            - Breed (virtual)
- Derived Classes
    - Ant
        - Methods
            - Move
            - Breed
    - Beetle
        - Methods
            - Move
            - Breed
            - Starve
- You can add any member variables necessary for each class

Details:

- The game will be played on a 10 x 10 grid
- Beetles will move before the ants
- A file will be used to populate the grid
    - See sample file

    - a = ant
    - B = beetle
- Grid traversal should be by column first then row
    - Creatures to the left perform actions before creatures to the right
- User will specify number of turns to watch
    - There will not be any user interaction in the game
    - The goal is to test the AI, so the game will play against itself
- All movement is orthogonal (N, S, E, W)
    - Movement is limited to one space per turn
- Movement happens before breeding and starving
- A beetle will eat an ant if it moves into the same space as the ant

Ant Details:

- **Move**
    - Each ant will move in the opposite direction of the nearest orthogonal beetle
    - If no beetle, ant stands still
    - If multiple beetles are nearest, prioritize movement
        - Move ant in direction of no beetle if possible
        - If beetles in all directions, move toward farthest beetle
    - Cannot move to occupied space
        - If space moving to is occupied, no movement happens
    - Cannot move off grid
- **Breed**
    - If ant survives 3 turns, it breeds
    - Add ant in adjacent orthogonal space
        - Start with north space and check clockwise around space until empty orthogonal space found
        - If no empty spaces, no breeding
    - Ant may not breed again unless it survives another 3 turns

Beetle Details:

- **Move**
    - Move toward nearest orthogonal ant
    - If multiple ants are nearest prioritize movement
        - Move toward ant with most adjacent ant neighbors (orthogonal and diagonal)
        - If still tied, move toward ant with most ant neighbors using the following priority: N, E, S, W
    - If no ant, move toward farthest edge
- **Breed**
    - If beetle survives for 8 turns, it breeds
    - Use breeding algorithm for ants
    - Beetle cannot breed again unless it survives another 8 turns

- **Starve**
    - If a beetle does not eat an ant in 5 turns, it dies

User Interface: The only user interface will be the user entering the number of turns for the simulation.

Input: The initial grid will be populated from file input. The file will be named **`world.txt`**. There will be no need for input validation.

Output: All output will be sent to the console. Print the current state of the grid to the console window. The simulation should pause after each turn. Have the user press enter to start the next turn.