CS 2336 – PROJECT 1 – Beetle Battalion

Pseudocode Due: 2/7 by 11:59 PM

Core Implementation Due: 2/14 by 11:59 PM

Final Submission Due: 2/21 by 11:59 PM

KEY ITEMS: Key items are marked in red. Failure to include or complete key items will incur additional deductions as noted beside the item.

Submission and Grading:

- All project source code will be submitted in zyLabs.
    - Projects submitted after the due date are subject to the late penalties described in the syllabus.
- 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.
    - zyLabs will provide you with an opportunity to see how well your project works against the test cases. Although you cannot see the actual test cases, a description will be provided for each test case that should help you understand where your program fails.
- **Type your name and netID in the comments at the top of all files submitted. (-5 points)**

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

Problem: Game Circus, a developer located in Addison, has begun working on a new mobile game called Beetle Battalion. The game is a simple, casual game where players try to invade and defeat a colony of ants with an invading colony of beetles. As an intern for Game Circus, you have been given the task to help implement the novice level of AI for the program.

Pseudocode: Your pseudocode should describe the following items

- For each function, identify the following
    - Determine the parameters
    - Determine the return type
    - Detail the step-by-step logic that the function will perform
- Functions
    - Ant class
        - Breed
        - Move
    - Beetle class
        - Breed
        - Move
        - Starve
    - Main class
        - Main
        - Any additional functions that you plan to create

Core Implementation

- Read the input file and store it in the grid
- Reproduce the first 5 turns of the sample given
    - Basic beetle movement to the right and up
        - Moving toward closest ant
    - Basic ant movement to the right and down
        - Moving away from beetle
        - Not moving off grid or stepping on other ants
    - Ant breeding (turn 3)
- There are no ties
- The sample does not cover every possible use case for movement or breeding
- No starving check
- No beetle breeding check

Classes

- Base class
    - Name file **`Creature.java`**
    - Creature
        - Abstract class (-5 points if not)
        - Character to represent creature
        - Methods
            - Move (abstract) (-5 points if not)
            - Breed (abstract) (-5 points if not)
- Derived Classes
    - Ant
        - Name file **`Ant.java`**
        - Methods
            - Move
            - Breed
    - Beetle
        - Name file **`Beetle.java`**
        - Methods
            - Move
            - Breed
            - Starve
- The move, breed and starve functions will provide information to main to perform the operation
    - These functions do not modify the grid
    - It wouldn't make sense for an object in an array to pass in the array it's sitting in to a method
- You can add any member variables necessary for each class
- The main function must be in a file named Main.java

Details:

- The game will be played on a 10 x 10 grid
    - You decide how the grid will be stored in memory
    - This grid will be created in main
    - This grid will not be passed into any of the ant or beetle objects
- 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
    - Evaluate a column top to bottom before moving the next column
    - 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

Turn Order:

1. Beetles move
2. Ants move
3. Beetles starve
4. Ants breed
5. Beetles breed

Ant Details:

- **Move**
    - Each ant will move in the opposite direction of the nearest orthogonal beetle
    - If no orthogonal 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
        - If there are multiple possible directions to move (either multiple clear pathways or multiple beetles furthest away) use the following movement priority: N, E, S, W
    - Ants cannot move to an occupied space
        - If space moving to is occupied, no movement happens
    - Cannot move off grid
    - If an ant cannot move in the chosen direction, it does not move
- **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
        - If there is a tie for farthest edge, use the following priority: N, E, S W
- **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 user will be prompted for the following information in the order listed

- Initial grid filename
- Character to represent ant in output
- Character to represent beetle in output
- Number of turns

Input:

- The initial grid will be populated from file input.
- Prompt the user for the input filename
- 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 for each turn.
    - Display the turn header followed by a blank line
        - `TURN<space><turn number>`
    - Display each row of the grid followed by a newline
        - Each ant and beetle is represented by the user-defined character
        - An empty cell in the grid is represented with a space
    - Display a blank line after the last row of the grid