Project 4 - Build Me An Antfarm

Due Date: 5:00 p.m., December 9, 2015
You will not demo this assignment. Changes and clarifications to the assignment will be in red.

Project Description

You and a partner are going to build an antfarm, and you're going to use some common design patterns to do it. The program will create several anthills. Ants must leave the anthill and randomly wander the meadow in search of food. When they do, they may randomly run into other ants. If the other ant is from another hill and is of a certain type, they should fight. The loser of the battle should die. The program will run cycles in which it performs a series of steps each cycle until only a single anthill is left.

- Functional Requirements:
  - A meadow can have many ant farms in them. For this assignment, only ever allow one meadow to take place.
    - **Meadow requirements:**
      - The meadow will be a 25 x 25 grid
        - You may adjust the grid size if you feel a smaller or larger grid is more effective
        - Each grid element should contain a Cell object that has the following information
          - @hill = nil
          - @food = 0
          - @ants = Array.new
    - The meadow should randomly place 'food' in various cells after every cycle
  - You will randomly place Anthills in the meadow at various starting positions.
    - An anthill should take a single grid Cell
    - A meadow should spawn 10 anthills on initialization
      - Anthills are spawned by 'releasing a queen ant' who randomly selects a grid cell to build her anthill.
      - Anthills start with 5 pieces of food
    - **Anthill requirements**
      - Anthills can spawn any number of ants of the following type on each cycle
        - foragers (used to search for food)
        - warriors (used to kill other ants)
        - builders (used to build rooms in the anthill)
    - You cannot spawn more ants unless you have enough food and rooms
      - 1 piece of food and 1 vacant room allows you to create 1 ant

    - The food is consumed when the ant is created
  - When an ant is created, it will leave the anthill and randomly wander the meadow in search of food or battle.
    - Each ant moves 1 square in the meadow per turn
      - Movement should be random, but the ant should not move off the grid. For example, if the ant is in a corner cell, it should only have 2 directions it can travel in and not be able to leave the ‘meadow’.
    - There are 3 ant types:
      - The forager searches each square for food as it wanders the meadow
        - Food found in the meadow is immediately added to the ant hill (no need to hike it back)
          - Food can only be found by foragers
      - The warrior searches each square for other ants as it wanders the meadow
        - If the other ant is a forager and from a different anthill, the forager immediately dies.
        - If the other ant is a warrior and from a different anthill, a random winner is selected
      - The builder ant will build rooms that will in turn spawn your ants.
        - Rooms should be contained within the anthill.
          - Ants are spawned by rooms. A room can only spawn ants of a certain type (forager, builder, warrior), and is dependant on available food.
    - You will need a room for each type: forager, builder, warrior
      - Creating rooms costs 1 food and kills 1 builder ant
  - If a warrior of an ant colony wanders onto the anthill of another ant colony, it has a low chance to destroy the entire antill (something like ⅕, but experiment with what you think works best). If it succeeds in killing the hill of another colony, the colony (including all its ants) dies and is deleted from the meadow.
  - The simulation ends when there is 1 (or less) active colonies.
- Technical Requirements: (Design patterns bolded)
  - The Meadow class MUST be a **singleton** using the singleton module.
  - There should be a single Ant class. An antfarm should build each of the Ants the same, then add the warrior or forager functionality afterward using *runtime object modification*, and the return the ant. (**factory pattern**)
    - You will need to have some way to differentiate between ant types, i.e. a flag, storing them in separate arrays, etc.

        - On initialization, Queen ants should be created, then sent out into the meadow to pick a random spot for their ant hill using the **builder pattern**.
    - Your program should print a summary of the meadow after every 5 full turns
        - A full turn is defined as
            - each anthill produces all of the ants it can
                - You need to come up with an algorithm for how many foragers, builders, and warriors should be created

    - Extra Credit
        - Forager must bring the food back to the ant hill before it is counted. That means you must determine how the forager will find it's way back.
            - Use a print statement to show when food is found, and when the food is returned to the anthill
        - Ants gain experience after certain operations. For example, a forager ant gains experience when it finds food and a warrior ant gains experience when it kills another ant.
            - The experience can then be used to give the ant an advantage. You can increase the ant's speed (2 cells per cycle), or give them an edge in a fight.
        - Each anthill has its own formula for what should be created next, Forager, Builder, or Warrior. Some prefer creating builders, some prefer creating warriors, etc.
            - Each one balances the ratio of warrior, builder, and forager differently.

Changing the Assignment

You can change the assignment in any way you wish to make the assignment more interesting for extra credit. Any changes made to the requirements to make the program more sophisticated or interesting must be accompanied by an explanation in the Readme. The only rule is that the changes must not simplify the project.

Submission

Expected Interface and Test Output

- Required code organization:
    - /project4
        - Meadow.rb

            - Cell class can go in here since as a nested or external class
        - Anthill.rb
            - Contains the rooms, food info, existing ant references, and the logic for creating the ants
        - Ant.rb
            - Ant type is built on the fly
        - driver.rb
            - contains the main
        - Room.rb
            - Builds the ant according to the type of room it is (forager, builder, warrior)
        - Any additional classes you may need should be in separate files
        - readme
            - contains a description of your design

- Expected Output:
    - After every 5 rounds

        -
          ```
          Anthill Name: Killer
          Forager Ants: 14
          Warrior Ants: 5
          Builder Ants: 2
          Ant Kills: 18
          Colony kills: 2
          =============
          Anthill Name: Pansy
          Forager Ants: 14
          Warrior Ants: 5
          Builder Ants: 0
          Ant Kills: 2
          Colony kills: 0
          =============
          ....
          ```
- *All file inclusions must be case sensitive. If you are working on a Windows machine, you will need to double check this.*

- Create a zip archive of your project 4 folder with the following command
  - zip -r project4 <<foldername>>
    - This creates an archive of all file and folders in the current directory called project4.zip
- Upload the archive to Blackboard under Project 4.
- You may demo your lab by downloading your archive from Blackboard. Extract your archive, then run your code, show your source, and answer any questions the TA may have

Grading Guidelines and Checklist

Total: 75

- **Project Implementation**
  - Meadow (16 points)
    - Meadow class with a fixed sized grid using the singleton module *(3 points)*
    - 10 anthills are spawned during initialization using Builder pattern *(5 points)*
    - The simulation prints the status of each anthill after every 5 turns *(3 points)*
    - The simulation ends when only 1 or less anthills remains *(5 points)*
  - Anthills (17 Points)
    - Anthills contains Room objects that spawn an Ant object, consuming 1 piece of food *(5 points)*
    - Room creation kills the builder ant who created it *(2 points)*
    - Ant creation is dependant on room type *(2 points)*
    - Object Runtime Modification is used to add forager, builder, or warrior functionality to the ants (8 points)
  - Ants (35 Points)
    - A builder ants add rooms to the Anthill, consuming 1 food, then expiring *(4 points)*
    - An student determined algorithm determines which room type should be built next (4 points)
    - Rooms only create a certain kind of ants (3 points)
    - Each cycle, an anthill can only create a number of ants equalling the maximum number of rooms and total available food (5 points)
    - Only the Forager finds food, and only the warrior fights *(3 points)*
    - Foragers die when encountering Warriors *(3 points)*
    - When two warriors meet, the outcome is random, but one dies *(3 points)*
    - When a warrior happens on another Anthill, there is a **small** chance the Anthill will be obliterated *(5 points)*

        - All ants move randomly by 1 cell every cycle (3 points)
        - Ants do not leave the meadow when they move (2 points)
- *Extra Credit*
    - *Forager brings food back to the anthill before the food is used (7 points)*
    - *An experience mechanism is implemented that affects the outcome of rival ant interactions (7 points)*
    - *Each anthill has a unique formula for balancing ant type creation (7 points)*
- **Submission (7 points):**
    - Follows requested project structure and submission format (5 points)
    - Follows formatting guidelines (2 points)