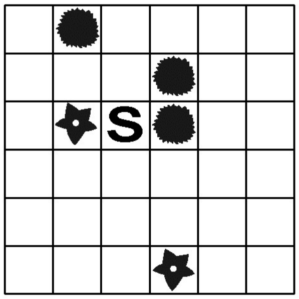
2. This question involves reasoning about the code from the GridWorld case study. A copy of the code is provided as part of this exam.

A `StockpileCritter` is a `Critter` that uses other actors as a source of energy. Each actor represents one unit of energy. The `StockpileCritter` behaves like a `Critter` except in the way that it interacts with other actors. Each time the `StockpileCritter` acts, it gathers all neighboring actors by removing them from the grid and keeps track of them in a stockpile. The `StockpileCritter` then attempts to reduce its stockpile by one unit of energy. If the stockpile is empty, the `StockpileCritter` runs out of energy and removes itself from the grid.

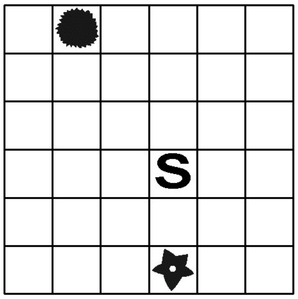
Consider the following scenario.

INITIAL WORLD



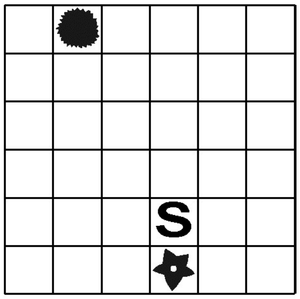
`StockpileCritter`
is in location (2, 2),
stockpile is empty

AFTER ONE ACT



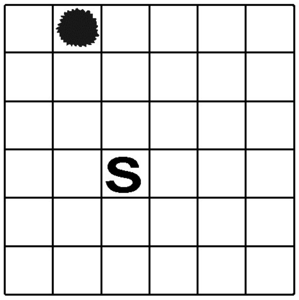
Gathered 3 actors,
used 1 energy unit,
2 remaining in stockpile,
moved to location (3, 3)

AFTER TWO ACTS



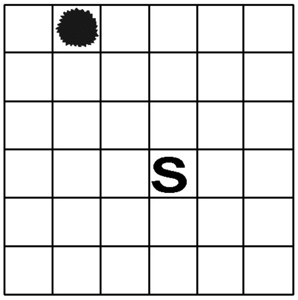
No actors gathered,
used 1 energy unit,
1 remaining in stockpile,
moved to location (4, 3)

AFTER THREE ACTS



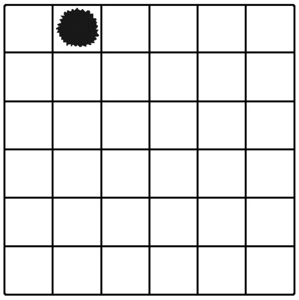
Gathered 1 actor,
used 1 energy unit,
1 remaining in stockpile,
moved to location (3, 2)

AFTER FOUR ACTS



No actors gathered,
used 1 energy unit,
0 remaining in stockpile,
moved to location (3, 3)

AFTER FIVE ACTS



Stockpile empty,
removed self from grid

Write the complete `StockpileCritter` class, including all instance variables and required methods. Do NOT override the `act` method. Remember that your design must not violate the postconditions of the methods of the `Critter` class and that updating an object's instance variable changes the state of that object.