Question 2: Stockpile Critter (GridWorld)

+1 class header
+1/2 properly formed class header for StockpileCritter
+1/2 extends Critter class

+1 1/2 stockpile state
+1/2 declares instance variable capable of maintaining state
+1/2 private visibility
+1/2 initialization of state appropriate to usage of variable

+1 overrides methods and maintains all necessary postconditions
(No points awarded if overrides act method)

+1 processActors overridden (No points awarded if overrides act method)

+1 stockpile state maintenance
+1/2 accumulates based on number of actors passed to processActors
+1/2 decrements appropriately each act

+1 1/2 removes neighboring actors from grid
+1/2 removes at least one neighboring actor from grid
+1 removes all neighboring actors from grid

+2 self-removal
+1/2 checks status of stockpile by using state variable in a relational expression
+1/2 ever removes self from grid
+1 removes self from grid when and only when stockpile state indicates empty
public class StockpileCritter extends Critter
{
  private int stockpile;

  public StockpileCritter()
  {
    stockpile = 0;
  }

  public void processActors(ArrayList<Actor> actors)
  {
    for (Actor a : actors)
    {
      if (a instanceof Actor)
      {
        a.removeSelfFromGrid();
        stockpile++;
      }
    }
  }

  public Location selectMoveLocation(ArrayList<Location> locs)
  {
    stockpile -= 3;
    if (stockpile < 0)
    {
      return null;
    }
    return super.selectMoveLocation(locs);
  }
}
Public class StockpileCritter implements Critter {

private int energy = 0;

public void processActors(ArrayList<Actor> actors) {
    for (Actor a : getNeighbors(getLocation())) {
        a.removeSelfFromGrid();
        energy++;
    }
}

energy--;

if (energy == 0) {
    removeSelfFromGrid();
}
public class StockpileGriffon extends Griffon
{
    private int energy;

    public StockpileGriffon()
    {
        energy = 0;
    }
}
Question 2

Overview

This question involved reasoning about the code from the GridWorld case study, emphasizing object-oriented concepts. Students demonstrated their understanding of the case study and its interacting classes by extending the Critter class to derive a StockpileCritter class with modified behavior. This question tested numerous concepts: creating a class, inheriting from an existing class, overriding appropriate methods, and maintaining the overridden methods' postconditions. Students were specifically instructed not to override the act method, and they were explicitly cautioned to abide by the postconditions of all methods.

Sample: A2a
Score: 8

The student correctly declares a class StockpileCritter that extends Critter. There is a private instance variable, stockpile, that is correctly initialized. The student overrides selectMoveLocation but fails to maintain all postconditions because the decrement of the stockpile causes a change in the critter's state. The student overrides the method processActors correctly. Within processActors, the student correctly adds the number of neighboring actors to the stockpile and correctly removes all these neighbors from the grid. The check for “a instanceOf Actor” is redundant but causes no ill effects. The student checks the status of the stockpile in selectMoveLocation and returns null when the stockpile is empty. This allows removeSelfFromGrid to be called in all appropriate cases. There is also a correct decrement to the stockpile.

Sample: A2b
Score: 5

The student correctly declares a class StockpileCritter but uses implements Critter instead of extends Critter. There is a private instance variable, energy, that is correctly initialized. The student fails to override the method selectMoveLocation but overrides the method processActors correctly. Within processActors, the student incorrectly loops over getNeighbors(getLocation) instead of over actors. This resulted in a ½-point loss for failing to accumulate based on the parameter actors and also resulted in a full-point loss for not correctly removing all the grid neighbors. The student received ½ point for removing one neighbor from the grid. There is a correct decrement to the stockpile. The student checks the status of the stockpile and correctly calls removeSelfFromGrid in one case. The student incorrectly calls removeSelfFromGrid when energy == 0.

Sample: A2c
Score: 2

The student correctly declares a class StockpileCritter that extends Critter. There is a private instance variable, energy. The student did not get the ½ point for “initialization of state appropriate to usage” because there is no usage of this instance variable.