Program Breakdown:

Our program is broken down into a Model-view-controller(MVC) architecture. The Main method begins by creating a new Controller object, passing the keyboard as input.The controller class is responsible for handling user input. The commands “seed”, “make”, and “step” all directly affect the game state (AKA mode component of the MVC architecture). On the other hand, “show” and “stats” are commands that give the user a *view* of the game state.

The world consists of an ArrayList of Critters. This ArrayList is stored as a static field called population within the Critter class. The abstract Critter class is both responsible for defining “global” methods such as displayWorld, worldTimeStep, as well as providing the specific fields and methods that Critters inherit.

Other classes include Params and InvalidCritterException. The Params class is simply used to store constants that define the rules of the Critter world. Meanwhile, the InvalidCritterException class is just a subclass of the Exception class. InvalidCritterExceptions are thrown when commands have an error being processed. The rest of the classes found in the assignment4 package are subclasses of Critter, overriding the methods of Critter to have their own look and behavior.

New Classes Created:

Conflict Class: This class is a secondary static class within Critter. A new Conflict is created when two Critters are discovered occupying the same coordinates. These conflicts are then queued in a LinkedList and polled until no conflicts exist.

Fields:

- Critter m1 - First Critter involved in the conflict
- Critter m2 - Second Critter involved in the conflict

Methods

- void resolveConflict() - Calls both Critter’s fight methods to see how to proceed with the conflict. The purpose of this method is to guarantee that only one Critter remains at the coordinates after resolution of the Conflict.

Controller Class: A new Controller object is created in the Main method. Then, promptInput() is called repeatedly until quit is set to true.

Fields:

- boolean quit - Whether or not the program should be exited (Set to false by default)
- Scanner kb - The Scanner input passed from main

Methods:

- void promptInput() - Repeatedly prompts the user for input, prompts that input for commands, and then processes the commands.

Critter(1-4) Classes: These classes are subclasses of Critter. They redefine the toString, toTimeStep, runStats, and fight methods. This means that each Critter subclass has their own behavior and “appearance” when printed by Critter.displayWorld(). The behavior of each of these Critter subclasses is documented in the header of each Critter(x).java file. The java docs of the Critter methods are also provided in our docs folder.