README

We used the default methods given to us and didn't create any new classes. The only thing new made were for some helper functions in Critter.java in order to make debugging the game engine a bit easier.

Critter List:

An important thing to note is that the Critter objects are stored in a List structure, called **population**. Likewise, the reproduced babies are also in a similar List structure, titled **babies.**

Critter Movement:

Notably, we used **private final void move(int direction, int steps)** as a helper function to the default **walk** and **run** methods. Essentially, we realized that there are only 7 directions (0-7) in which a Critter object can move, and so we used a switch/case structure to allow for readability when trying to understand what each individual direction requires and must do. In addition, we had two helper functions, **private int wrapX(int x)** and **private int wrapY(int y)** to the helper function, **move**. The **wrapX** and **wrapY** functions used the conjured formula to simulate a torus map, so that when a Critter object falls off the board, it wraps around to the opposite side.

Critter Fights:

Helper function, **private static ArrayList<List<Critter>> findEncounters()**, is used to find all "encounters" between Critter objects. An encounter is defined to be when two Critter objects share the same spot on the grid. We search the whole population of Critter objects and store their x-y coordinates to serve as a String key. We later use that key and the corresponding Critter object from the HashMap structure, **critter_positions**, to find all Critter objects that share the String key (i.e., the same x-y coordinates). The HashMap allows for readability when debugging and also lets us make use of the .containsKey method in order to search for same x-y coordinates. When an encounter is found, we put the two Critters into an ArrayList, **encounters_list**, which will later be used in helper function, **private static void battleEncounters(ArrayList<List<Critter>> encounters)** to simulate fighting.

Displaying the Board:

Rather simple, utilized a 2-d String array to represents rows and columns of the board.