Connor Byron (rcb2746)
Joel Guo (jg55475)

BASIC CODE STRUCTURE

1. CONTROLLER – MAIN.JAVA

The Main.java file contains the basic controller for our critter simulations. It has a main method that directs user input to commands that we have implemented (via a simple switch statement). Most commands are directed toward methods in the Critter class.

2. MODEL - CRITTER.JAVA

The Critter.java file stores our abstract Critter representation. All critters have energy, XY coordinates (represented as Points), and basic walk/run/reproduce methods. Subclasses of Critter are also required to implement unique behavior in doTimeStep and fight methods. More specifically, doTimeStep handles how they change each time step whereas fight determines whether or not the critter wants to fight in a given situation.

Here is the basic idea behind our worldTimeStep structure:

```
public static void worldTimeStep()
```

1. increment timestep count
2. loop through all critters in collection, call doTimeStep for each (walk/run, energy deduction, reproduce babies in crib, cheat)
3. doEncounters & fight
4. update rest energy
5. add algae
6. remove dead critters
7. add babies to populations

3. VIEW – CRITTER.JAVA

At the moment, our "view" is simply a text-based world view that is rendered by the displayWorld method inside Critter.java This should change in Part B of this assignment.

Connor Byron (rcb2746)
Joel Guo (jg55475)

CRITTER LAB – PART A

4. WORKING STRATEGY

We started by meeting up to create a GitHub repository to host our code. Next we began laying out the tasks that needed to be done. Then we assigned those tasks to each other, based on workload and personal strengths/weaknesses. Some parts, such as our 2 individual critter classes, were created separately. However, most of the other parts were programmed using "paired programming". We both rotated between driver and navigator intermittently.

5. CONTROLLER - MAIN.JAVA

Connor spent most of the time designing the command structure (prompting for input and catching exceptions). Both Joel and Connor worked to implement each of the text commands required.

6. MODEL - CRITTER.JAVA

Joel worked hard on handling encounters between different critters in the worldTimeStep method. He also helped with the critter reproduction code sections. Connor worked hard on many of the remaining methods such as worldTimeStep and displayWorld. Both Joel and Connor worked on the makeCritter method.

7. TESTING/DEBUGGING

Joel and Connor debugged most errors individually. Each of us ran our own simulations and tests to debug the core functionality. We also debugged our respective Critter subtypes.