ZooLife: A simulator of life in the zoo

Author: Tarek Oraby
Date: October 31, 2019

ZooLife is a java application that simulates life among a number of zoo animals. It is written in Java 8 and implemented as a java console. At the most general level, the design of ZooLife follows the Model-View-Controller (MVC) architecture. On one hand, the viewing and controlling parts in that architecture are implemented in two classes: the "Viewer" and the "Controller" classes, respectively. On the other hand, the implementation of the Model (which is the engine of the whole application) is distributed among a number of classes. Overall, the ZooLife app has the following packages and class hierarchy:

- com.zoolife.model
  - Animal
    - Bird
      - Chicken
      - Parrot
    - Dog
  - AnimalFriendship
  - Loader
  - Zoo
  - ZooKeeper
- com.zoolife.control
  - Controller
- com.zoolife.view
  - Viewer
- com.zoolife.main
  - Main
- com.zoolife.test
  - AnimalFriendshipTest
  - AnimalTests
  - ZooTests
  - ZooKeeperTests

Animals are represented in a number of inherited classes:

1. The "Animal" class is the super class of all animals, and it holds information about common animal features (i.e. their name and favorite food)
2. The "Bird" class is a subclass of the "Animal" class, and the superclass of the "Chicken" and "Parrot" classes.
3. The "Dog" class is a subclass of the "Animal" class

At the center of ZooLife's architecture is the "Zoo" class. This class holds information about:

1. the set of animals in the Zoo (represented as "Animal" objects)
2. the set of friendships among these animals (represented as "AnimalFriendship" objects)
3. the set of friendships that were established/broken up during the last simulated day (also represented as "AnimalFriendship" objects)

Other than maintaining and retrieving these records, the "Zoo" class main function is to ensure the structural integrity of the friendship network in the zoo. For example, it ensures that if animal A becomes friends with animal B, then B becomes also friend with A. Alternatively, to take another example, the "Zoo" class ensures that friendships are established only among animals that are already in the zoo.

In addition, another important class is the "ZooKeeper", which controls the methods of the "Zoo" class and initiates the daily simulations. The main purpose of the "ZooKeeper" class is to ensure that friendships are established or broken-up according to specific rules (which the "ZooKeeper" class takes as inputs). As such, while the "Zoo" class ensures the structural integrity of all edits to the zoo records, the "ZooKeeper" class is where the application ensures that the number of established or removed friendships per simulated day is within the desired limits.

Beyond this general overview, the details of the other classes in the ZooLife application are straightforward (and hopefully sufficiently documented). The one thing I wish to highlight is that I was not entirely sure about the sequence in which animals ought to establish and break friendships. I interpreted the rules as allowing animal A to befriend animal B in the same day in which they have earlier broken-up their friendship. Similarly, I interpreted the rules as allowing animal A to break-up with animal B in the same day in which they have earlier established their friendship. If this is not the correct interpretation, then few changes to the "ZooKeeper" class are needed.