ESS201 Programming II (Java)
Term 1, 2018-19
Assignment 7 (mini-project)
Simulation of flock behaviour

In this project we are trying to do a simple, yet interesting, simulation of a flock of birds. The intent is to understand how multiple developers working on a project can work independently and at the same time collaboratively to produce interesting programs. We will also look at how we can break up what appears to be a complex problem into simpler, manageable tasks.

To see some interesting flock behaviour, you can check out a video on "murmuration". Example:
https://www.youtube.com/watch?v=eakKfY5aHmY

To see samples of graphical simulations, try out examples like:
https://www.youtube.com/watch?v=zQhEGPrINJo
or
https://www.youtube.com/watch?v=dqB4Dcuo1v8

And if you would like to read one of the original papers on simulating flocks, here is a good start:
Reynolds, Craig, "Flocks, herds and schools: A distributed behavioral model". SIGGRAPH '87
http://www.cs.toronto.edu/~dt/siggraph97-course/cwr87/

We will try something simpler!

At any time, each bird is part of a flock. The birds within a flock try to maintain their relative position, and broadly following a "leader". The leader decides the direction the flock should fly. It turns out that the flock behaviour of birds can be modelled by some fairly simple autonomous decisions of individual birds. At each time instant, each bird adjusts its position to achieve two objectives:

1. Maintain its position relative to its nearby neighbours
2. Avoid colliding with other birds

So, we will break up the problem as follows:

- A flock which contains a set of birds, one of which is the leader
    - Flocks can split into two, with a new leader assigned for the new flock
    - Two flocks can merge into one
    - Every now and then, the leader changes - one of the other birds takes over as the leader, and the current leader swaps position with this bird.
- Birds, who know how to compute their next position, based on their nearest neighbours.
    - Different kinds of birds, but all of which follow the overall protocol
- The main (driver) which reads in user input and directs the leader, decides when to split or merge flocks etc.

You will be given the following:

- An abstract class Flock. You should implement a concrete extension of this that provides your way to implement the methods
- An abstract class Bird. You should implement at least one concrete extension of this that implements the flock behaviour of this bird.
- A set of classes that helps implement a simple graphical interface (using the Java classes Swift)
- The skeleton of a driver program (TestFlock) that you can extend as needed

Broadly, the program flow is as follows:

1. The main program sets up the graphical interface for the simulation
2. We create instances of different bird classes and add them to a flock
3. We identify a leader for the flock and start the main execution of the simulation. Each bird runs in its own thread.
4. The leader is provided with a target position - the position it should reach after some amount of time.
5. Each bird computes its new position by querying the position of its neighbours, and updates its current position. Its thread then “sleeps” for an interval of time and then recomputes its position.
6. The user chooses a new target by selecting a position on the screen (e.g with a mouse click).
7. The user can also choose to split or merge the flocks (e.g. by pressing buttons in the UI)

The evaluation will include a group demo: we choose a set of 6-10 students at random. Each of the groups runs a set of simulations. In each simulation we use the Flock instance (actually of the derived class) of one student and the Bird implementations of all the students in the group. We make one of them the leader and run the simulation. We try out different target positions, switch leaders, split and merge the flock, etc.

As part of the implementation, you will need to think through the following:

- What “formation” do the birds in a flock adopt. It might be easiest to assume they will fly in a random structure.. You could then tweak this for more interesting formations - straight lines, “V” formation, etc
- How does the leader compute the path from its current position to the target. You could use your creativity here
- How does a bird compute its next position - relative to neighbours
- Decide how you will decide the “neighbours” of a bird
- etc

Please note that you cannot change the abstract classes Flock and Bird. This will be common for everyone. If you have suggestions to improve these classes, send me a note and I can publish an updated version of the base classes and test class. Be ready for the fact that the classes will go through a few iterations till we are done. (The set of files of each version will be uploaded to LMS in directories labelled with the version numbers: v1, v2, ..