Introduction to Programming

Inheritance and Abstract Classes in Java

This week the practical focuses on **inheritance**, **class hierarchy** and in particular **abstract classes** in the context of a more complex program the fox-rabbit simulator.

Level 1:

- Implementation: The Foxes-and-Rabbits simulation: using the code provided in the zip file `FoxesAndRabbits.zip`, create a class `Wolf` that hunts foxes and rabbits.

  Initially create the `Wolf` class by simply copying the class `Fox`, changing the name everywhere needed and keeping all its default values, add the ability to eat foxes, making foxes worth 14, i.e. twice the food value of rabbits. Note: you will have to make the following changes in the class `Simulator`.

  1. insert:
     `private static final double WOLF_CREATION_PROBABILITY = 0.02;`
     after
     `private static final double RABBIT_CREATION_PROBABILITY = 0.08;`

  2. In the constructor insert:
     `view.setColor(Wolf.class, Color.black);`
     after
     `view.setColor(Fox.class, Color.red);`.

  3. In the `populate` method add the following code

     ```
     else if(rand.nextDouble() <= WOLF_CREATION_PROBABILITY) {
        Location location = new Location(row, col);
        Wolf wolf = new Wolf(true, field, location);
        animals.add(wolf);
     }
     ```

  Run the simulation and observe what happens to the foxes and the rabbits. (note you may find the SimulatorView.isViable() method is being called and halts your simulation).

- Change the Wolves behaviour in the following way:

  ```
  // The age to which a wolf can live.
  private static final int MAX_AGE = 40;
  // The likelihood of a wolf breeding.
  private static final double BREEDING_PROBABILITY = 0.05;
  // The maximum number of births.
  private static final int MAX_LITTER_SIZE = 2;
  ```

  Run the simulation and observe what happens to the foxes and the rabbits.

- Additionally change the Wolves behaviour in the following way:

  ```
  private static final int FOX_FOOD_VALUE = 7;
  ```

  Run the simulation and observe what happens to the foxes and the rabbits.

Level 2:

- Implementation: The Foxes-and-Rabbits simulation: Implement a class `Hunter`, modify your design from level 1 accordingly. A hunter does not breed and becomes inactive after a random period of time. A hunter is a predator for rabbits, foxes and wolves.

  To do this use the concept of an abstract class introduced in the lecture to implement an `Actor` abstract class as shown in the lecture that is a suitable super class for all the actors (animal and human) in the simulation.

Level 3:

- Investigation: You can read more about emergent behaviour and swarming/flocking in agent based systems here:
  `http://en.wikipedia.org/wiki/Agent-based_model`