Lab sheet 4:
More on Inheritance

Setter: Verena Rieser

Please use the code in **`lab7/foxesAndRabbits2`** for the following exercises.

Refactor the Code

Population Generator

Look at the previous version of the code in: **`lab7/foxesAndRabbits1`** and note the improvements made by introducing the Animal class.

Also, note the changes made to the **`simulateOneStep`** method in **`Simulator.java`**: By introducing the Animal superclass we have removed the dependencies (couplings) of the **`simulateOneStep`** method and the Fox and Rabbit class. The Simulator class, however, is still coupled to Fox and Rabbit in the populate method.

Your task is to further improve this code by introducing a new class **`PopulationGenerator`** and move the populate method into this class.

Now only this new class is coupled to the concrete animal classes, making it easier to find places where changes are necessary when the application is extended. The **`PopulationGenerator`** should be called in **`Simulator.java`**.

(2 marks)

Introduce New Animals

Define a new type of animal

- Define a completely new type of animal, as a subclass of **`Animal`**. You will need to decide what sort of impact it will have on the existing animal types. For instance, your animal might compete with foxes as a predator on the rabbit population, or your animal might prey on foxes but not on rabbits.

- You will probably need to experiment with the configuration settings in order re-establish the equilibrium in your ecosystem!

- Don't forget to modify the **`populate`** method.

- You should also define a new colour for you new animal class in **`SimulatorView`**. You can find a list of pre-defined colour names on the API page documenting the **`Color`** class in the **`java.awt`** package.

(1 mark)

Define a new Superclass

Introduce a new Abstract Class, which helps to reduce code duplication within other animals, for example, you could introduce a superclass **`Predator`**, which implements a hunt method.

(1 mark)