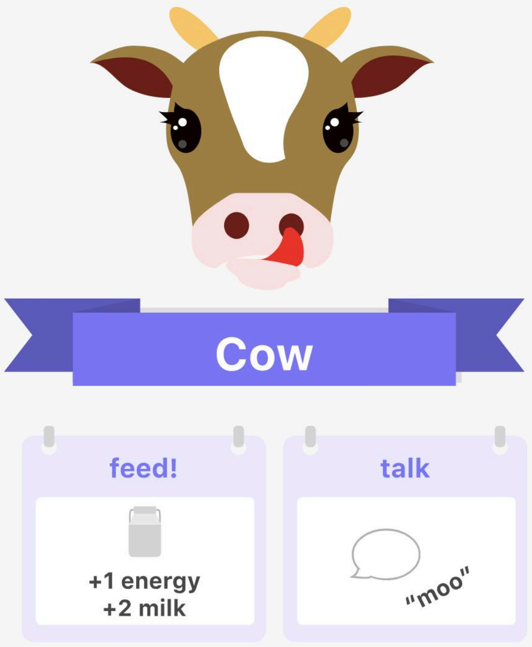

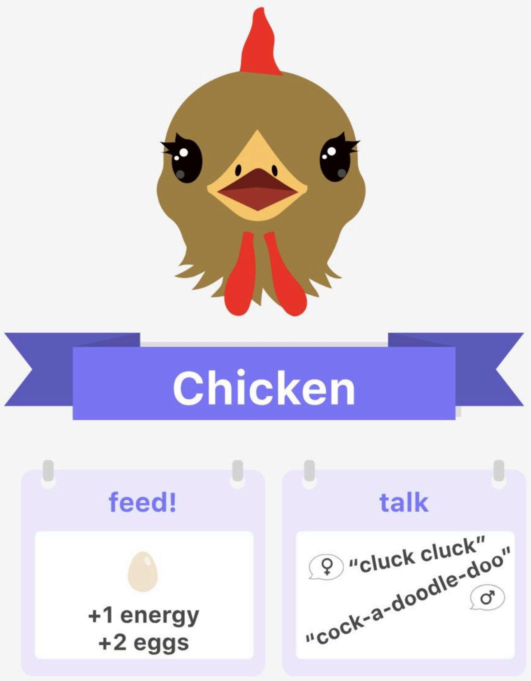

Background and Objectives

This challenge is an extension of the previous one: the farm welcomes its first animals!
Specs

Discuss with your buddy: how many classes do you think you need, and how would you structure them?
Don't rake! Wait until the very end of the challenge, follow the guidelines and let the farming diary guide you into coding the classes!

Parent and children

We are now familiar with the benefits of inheritance, let's go ahead and:

- Create the three empty classes
- Set the proper inheritance relationship between the children and the parent classes

Unlike in the previous challenge, let's start by coding the common behaviour in the parent class:

- An animal is initialized with zero energy
- You can feed an animal: it will increase its energy by 1

Animals Talk
To figure out the classes, let's start by the program we want to run:

- Open the lib/farming_diary.rb, read Day Three and gather information to code the classes.
- Run the file with ruby lib/farming_diary.rb. Solve one error at a time by coding the missing talk method in Cow and Chicken.

Expected output:
📝 Day Three: Animals Talk The cow says moo The female chicken says cluck cluck The male chicken says cock-a-doodle-doo

Feed The Animals
Let's move on to the Day Four and feed all the animals at once with an iteration. Remember your animals have a shared feed! method? You can call the same method on two objects of different types! This concept is called polymorphism 🤓

Here is what you need to know about **feed!**:

- Cow: beyond gaining energy, cows produce 2 liters of @milk
- Chicken: beyond gaining energy, females produce 2 @eggs (and males none 🤷‍♂️)

Hint: the children method extend the parent one. Don't forget to use super to call the parent's part!

Expected output:
📝 Day Four: Feed The Animals The cow produced 2 liters of milk The female chicken produced 2 eggs The male chicken produced 0 eggs

Take away
Congratulations! You can run the rake now to check that your code is properly organized.
In children class, there are 4 kinds of methods:

- methods that inherit from the parent class: the method is only defined in the parent class
- methods that extend the parent's method definition: the method is slightly different in the children classes
- methods that override the parent's method: definition is completely different than in the parent class
- methods that are specific to the child class: they are not defined in the parent class at all

Extending a method requires the super keyword: it acts as if you copied the body from the parent method and pasted it where super is invoked.