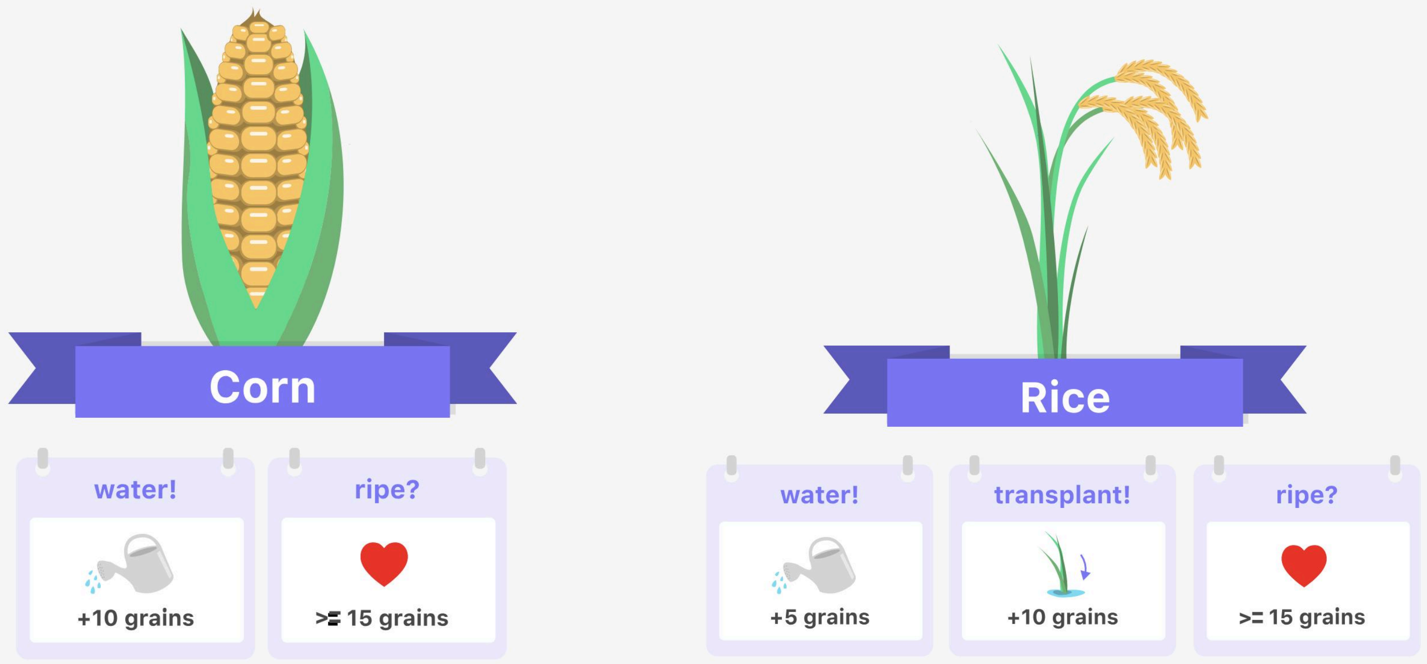

IMPORTANT: In this challenge, **do not** use rake to code your classes! Code the interface in **lib/farming_diary.rb** and let the program guide you into designing the classes! At the end of the challenge, when the interface prints the expected output, check your classes code with rake 👌

The Corn class
To start, code a Corn class in corn.rb with the following methods:

- initialize sets the instance variable @grains to zero.
- water!: adds 10 grains anytime it is called.
- ripe? returns true if there are at least 15 grains.

Open **farming_diary.rb** and complete the Day One section. Adapt the code to make it print the following output:

📝 Day One: Corn The corn crop produced 10 grains The corn crop is not ripe
Run your diary with:
ruby lib/farming_diary.rb

The Rice class
Create a Rice class in rice.rb and copy / paste all the methods from the Corn class.

- Adjust the grains production in water!: it adds only 5 grains.
- ripe? has the same behaviour than in Corn.
- Rice has a specific method called transplant! which produces 10 more grains.

Continue your farming diary by planting some rice on Day Two.

Refactoring

If you felt uncomfortable when copy / pasting code, you were right! Duplicating code is more maintenance and a source of errors. That's where inheritance comes to the rescue to keep the code DRY (Don't Repeat Yourself).
The crops share many similarities, refactor them:

- Introduce a parent class named Crop and move the shared methods into it.
- Make Corn and Rice classes inherit from Crop.
- Don't forget to require_relative.

Checks and takeaways

Now let's run rake! Take the time to make all the tests green, to validate your architecture and classes public interfaces. If you wonder why we restrict some setters in the specs, it is because the farming diary did not require to add them! Remember encapsulation?
Encapsulation is hiding by default the internal state or behaviour of an object, and exposing it with the right level of abstraction according to your programs needs
Congratulations on taking time to let the program guide your classes code before running the rake. It's an important path in your learning journey as a developer gaining autonomy.