Lab 9

- Rabbits and sloths are animals. Animals and bushes are things.

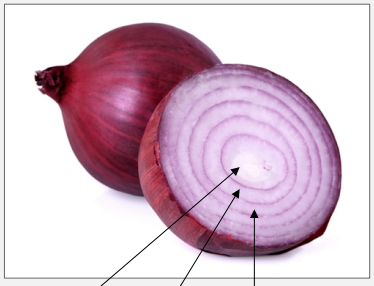
- "are" → `extends` or `implements`

Lab 9

- All things have a getter `getColor` which returns the thing's color (type `Color`). Bushes are green, rabbits are brown, sloths are gray. Since there's not a predefined `Color.BROWN` constant, you'll need `Color.rgb(165,125,0)` instead. (This getter can simply return the desired color — since a particular kind of thing is always the same color, there's no need for an instance variable for the color.)

- instance variables for `Thing`?
  - no
- method headers for `Thing`?
  - `getColor()`
- method bodies for `Thing`?
  - no
- `Thing` concrete, abstract, or interface?
  - no instance variables or method bodies, so could be abstract class or interface

`extends` vs `implements`



`extends` defines the core essence – can only extend one thing because there can only be one core essence

A, B extending C both share the same core

`implements` defines a hat being worn – can wear many hats at once

A, B implementing C both wear the same hat but are not necessarily otherwise related

https://itoldya420.getarchive.net/amp/media/santa-claus-dog-woman-animals-28e622

Lab 9

- All animals have a position (row and column) in the field, getters `getRow` and `getColumn`, and a setter `setPosition` which takes the row and column as parameters and sets the position accordingly.

- instance variables for `Animal`?
  - `row`, `column`
- method headers for `Animal`?
  - `getRow()`, `getColumn()`
  - `setPosition(row,col)`
- method bodies for `Animal`?
  - `getRow`, `getColumn`
  - `setPosition`
- `Animal` concrete, abstract, or interface?
  - concrete or abstract, depending on whether there are any unimplemented methods inherited or specified by interfaces

Lab 9

- Rabbits and sloths are animals. Animals and bushes are things.

Both animals and werebushes are creatures.

Werebushes are bushes

- "are" → `extends` or `implements`
- animals are both things and creatures
  → at least one of `Thing`, `Creature` must be an interface instead of a class
- werebushes are both bushes and creatures
  → at least one of `Bush`, `Creature` must be an interface instead of a class

Lab 9

Creatures are things that have a position in the field and can move.

- Creatures are things that have a position in the field and can move. Both animals and werebushes are creatures. Creatures have getters `getRow` and `getColumn()`, a setter `setPosition` which takes the row and column as parameters and sets the position accordingly, a `getNextMove` method which takes a `Field` object as a parameter and returns one of the constants defined in `Direction`, and a `reset` method which clears the creature's memory.

- ~~"thing" here isn't meant to refer to `Thing`~~
  - ~~intent is that creatures are defined by having a position in the field and being able to move~~
    - ~~which specifically means having `getRow`, `getColumn`, `setPosition`, `getNextMove`, `reset`~~
- "are" → `extends` or `implements`
  - intent is that creatures are things that have additional properties/functionality
  - note that interfaces can extend other interfaces and classes can extend other classes and implement interfaces but interfaces can't extend classes
    → `Thing` and `Creature` can both be interfaces or both be classes (abstract or not) or `Thing` can be an interface and `Creature` a class
    - cannot have `Thing` an abstract class and `Creature` an interface

Lab 9

Werebushes are bushes which sometimes turn into animals.

- this doesn't mean that an object changes type from `Werebush` to `Animal` and back
  - cannot change the type of an object in Java
  - (casting pulls one of the hats being worn to the top – it doesn't add a hat that wasn't there before)
- it also isn't intended to establish any kind of extends or implements relationship between `Werebush` and `Animal` or `Bush` and `Animal`

Lab 9

Werebushes are bushes which sometimes turn into animals.

- The position of a werebush is the position of its animal, even if it isn't currently transformed. The werebush's memory is its interval and duration counters plus whatever memory its animal has, so `reset` should set the counters back to 20 and 5, respectively, and reset the animal's memory. A werebush is drawn as its animal if it is transformed (the interval counter is 0) and as a bush otherwise. `getNextMove` should do the following: if the interval counter is greater than 0, decrement it and return `Direction.NONE` (no move when it is a bush), otherwise if the duration counter is greater than 0, decrement it and return the animal's next move (the werebush moves like its animal when transformed), and if both counters are 0, reset them to 20 and 5, respectively, and return `Direction.NONE` (the werebush is back to bush form).

- "turns into" is handled via composition
  - an `Animal` instance variable, which can be delegated to
    - "reset the animal's memory", "moves like its animal" → call those methods on the Animal