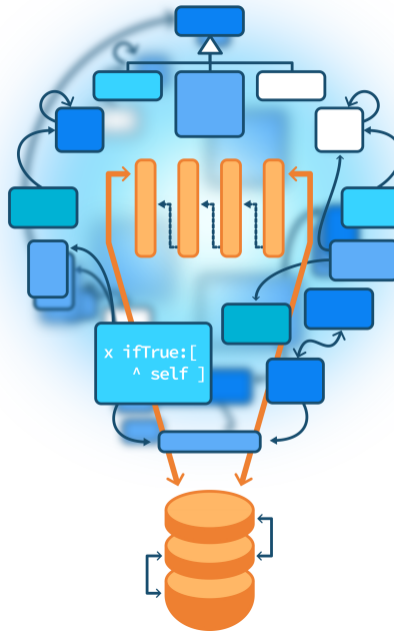


About super

S.Ducasse, L. Fabresse, G. Polito, and P. Tesone

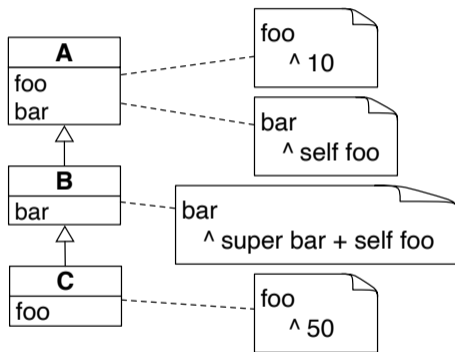


Goals

- Sending a message
- Method lookup
- super semantics and the differences with self



Define what super is!

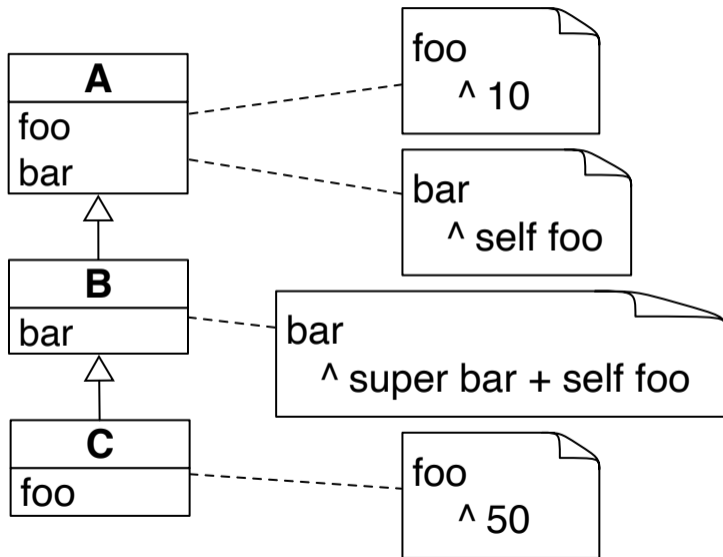


Take 5 min and write the definition of `super`

- your definition should have two points:
 - what does `super` represent?
 - how is a method looked up when a message is sent to `super`?

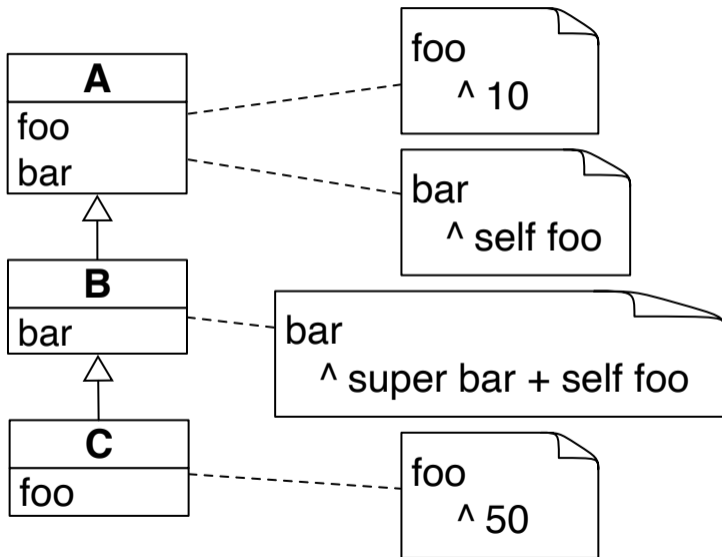


Challenge yourself with super!



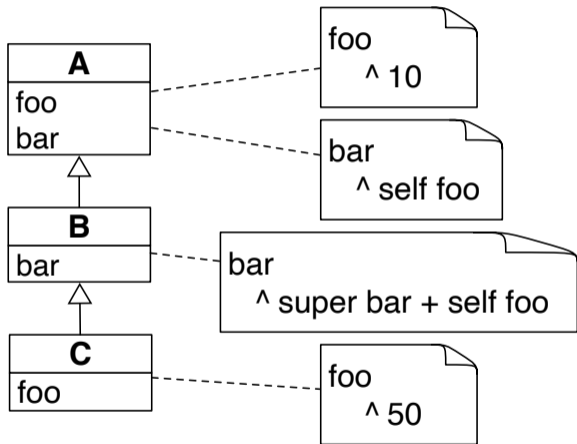
```
> aA bar
...
> aB bar
...
> aC bar
...
```

Challenge yourself with super!



```
> aA bar
10
> aB bar
20
> aC bar
100
```

super changes where the lookup starts



Evaluation of aC bar

1. aC's class is C
2. no method bar in C
3. look up in B - bar is found
4. method bar is executed
5. bar is sent to super
6. super is aC but lookup starts in A
7. bar is found in A and executed
8. foo is sent to aC
9. foo is found in C



super changes where the lookup starts

- super refers to the **receiver** of the message (just like self)
- The method lookup starts (Take 1 min to fill the dots)

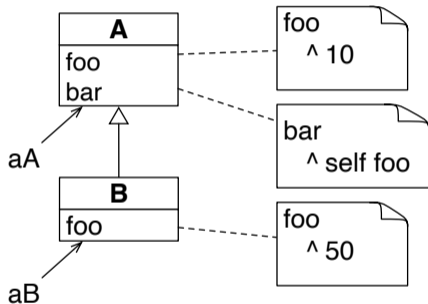


super in two sentences

- `super` refers to the receiver of the message (just like `self`)
- The method lookup starts in the superclass of **the class containing the `super` expression**



self is dynamic

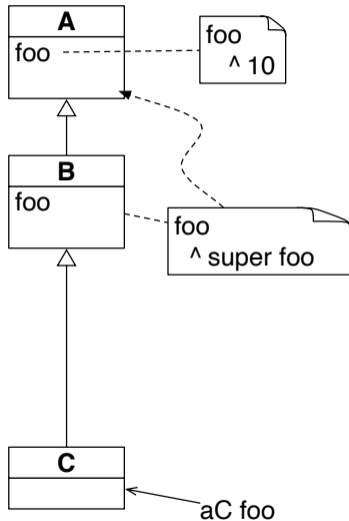


- At compilation time, we **don't** know
- to which object **self** points to
- to which **foo** method **bar** refers to

Imagine that we load a new subclass C of B and do `C new bar`, **self** will be pointing to such instance



super is static

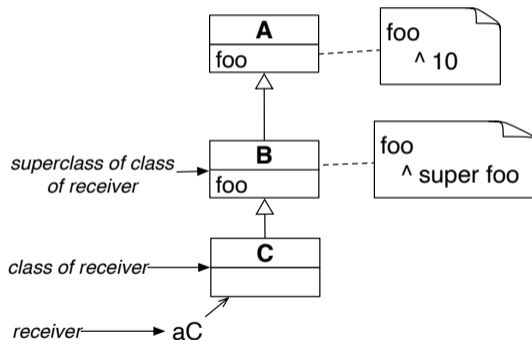


- At **compilation-time**, we know that $B \gg \text{foo}$ refers to $A \gg \text{foo}$
- we should look above the class containing the **method** using **super**



Even some books got it wrong

- **Wrong** definition: super *looks for the method in the superclass of the receiver's class*
- With this definition, this example would loop forever:



In reality it **does not** loop, the definition is wrong

What you should know

- self always represents the receiver
- super always represents the receiver
- super changes the lookup:
 - a super send starts the lookup in the class above it
- self sends act as a hook: code of subclasses may be invoked (see Lectures for more)



Produced as part of the course on <http://www.fun-mooc.fr>

Advanced Object-Oriented Design and Development with Pharo

A course by

S.Ducasse, L. Fabresse, G. Polito, and P. Tesone



Except where otherwise noted, this work is licensed under CC BY-NC-ND 3.0 France
<https://creativecommons.org/licenses/by-nc-nd/3.0/fr/>