

# Inheritance and Lookup: Self

Understand lookup once for all

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



# Goals

Understand:

- Sending a message
- Method lookup
- Semantics of `self/this`



# Remember inheritance

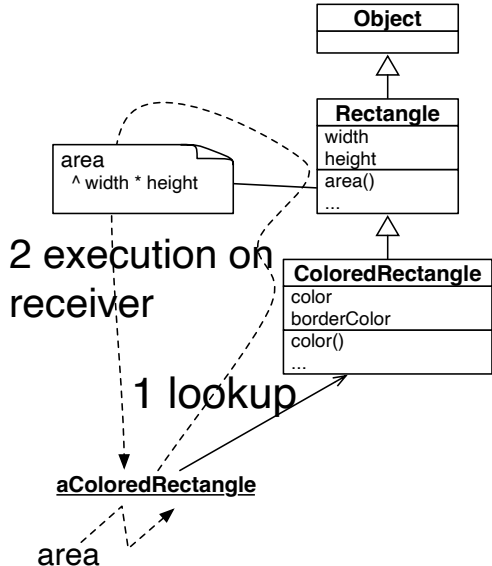
- Inheritance of **state** is **static** (done at compile time)
- Inheritance of **behavior** is **dynamic**
- In this lecture we focus on the behavior part



# Message sending

**Sending a message** is a two-step process:

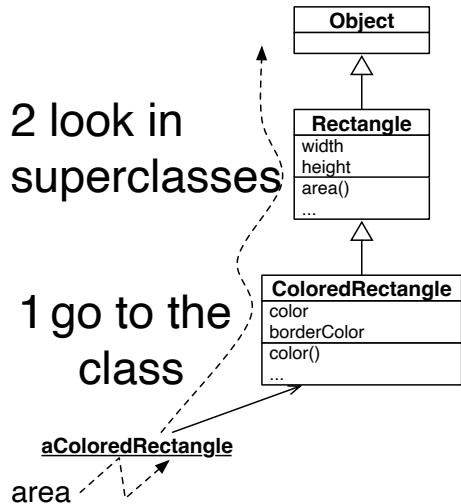
1. **look up** the **method** matching the message
2. **execute** this method on the **receiver**



# Method lookup

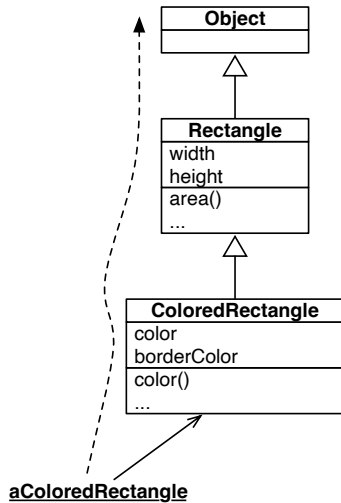
The lookup starts in the **class** of the **receiver** then:

- if the method is defined in the class, it is returned
- otherwise the search continues in the superclass



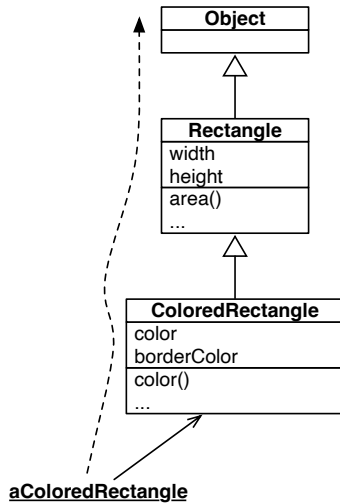
# Some lookup cases

Sending the message `color` to `aColoredRectangle`



# Some lookup cases

Sending the message `area` to `aColoredRectangle`



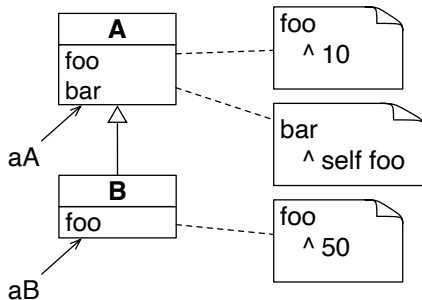
# About lookup implementation

- Most of the time, the result of a lookup is **cached** and a lookup happens only once
- In some languages, there are dispatch tables
- The point is that conceptually there is a lookup at execution





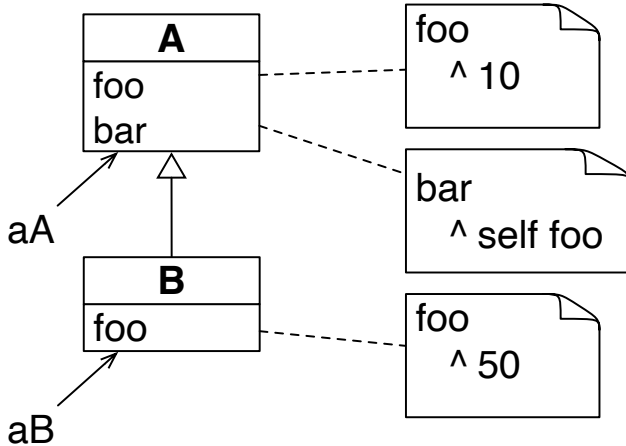
# What is self/this?



- Take 5 min and write the definition of `self` (this in Java)
- Your definition should have two points:
  - what does `self` represent?
  - how is a method looked up when a message is sent to `self`?

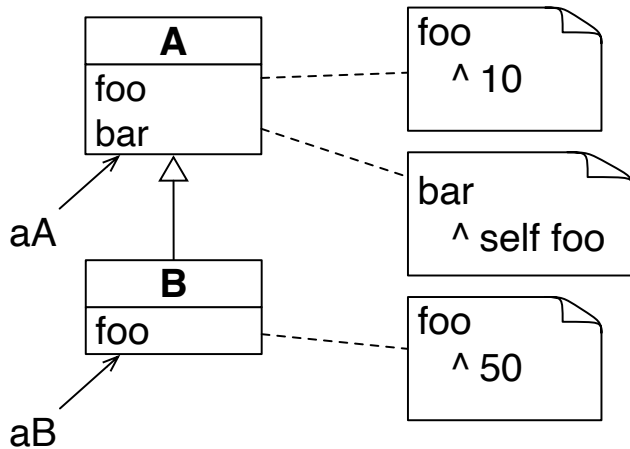


## Let us explore a bit



- aA is an instance of A  
(obtained executing A new)
- aB is an instance of B  
(obtained executing B new)

# Let us explore a bit



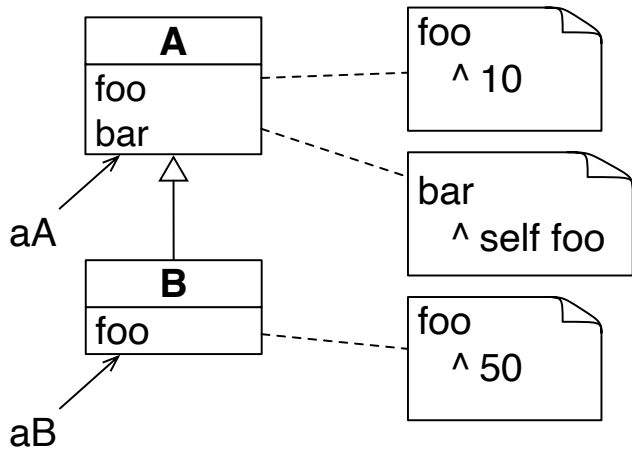
> aA foo

...

> aB foo

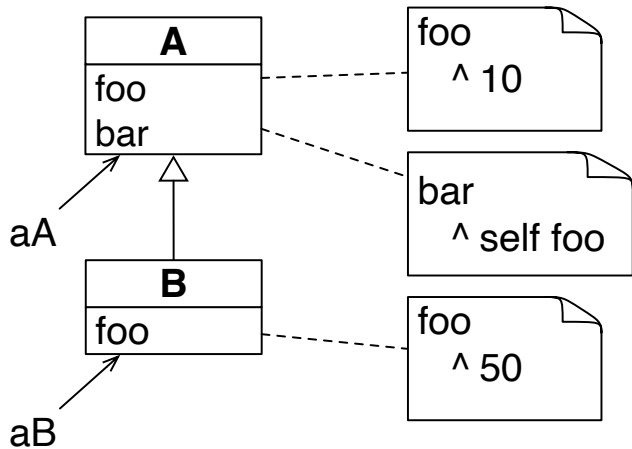
...

## self always represents the receiver



```
> aA foo
10
> aB foo
50
```

# self always represents the receiver



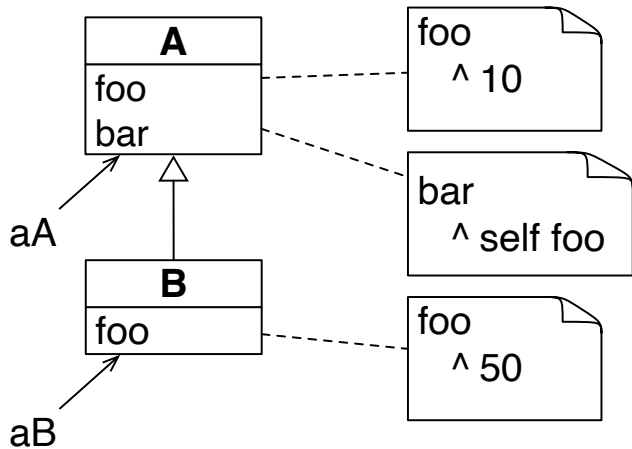
> aA bar

...

> aB bar

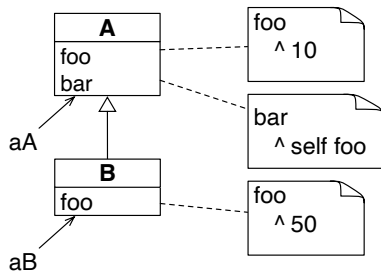
...

# self always represents the receiver



```
> aA bar
10
> aB bar
50
```

# Following message lookup and execution



Evaluation of `aB bar`

1. `aB`'s class is B
2. no method `bar` in B
3. look up in A - `bar` is found
4. method `bar` is executed
5. `self` refers to the receiver `aB`
6. `foo` is sent to `self`
7. look up `foo` in the `aB`'s class: B
8. `foo` is found there and executed

```
> aB bar  
50
```

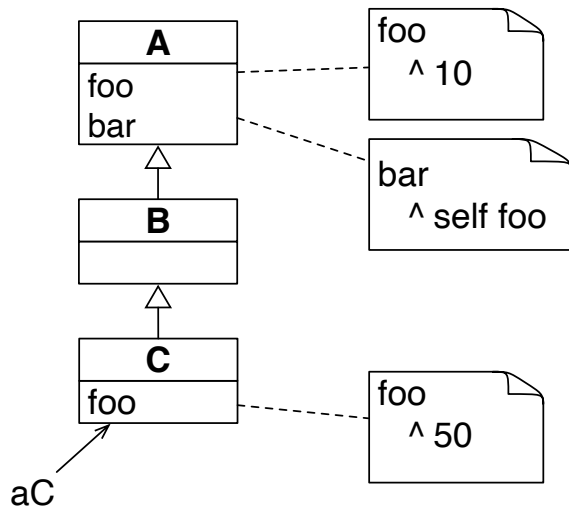
# self/this in two sentences

- self represents the **receiver** of the message
  - self in Pharo, this in Java
- The method lookup **starts in the class of the receiver**



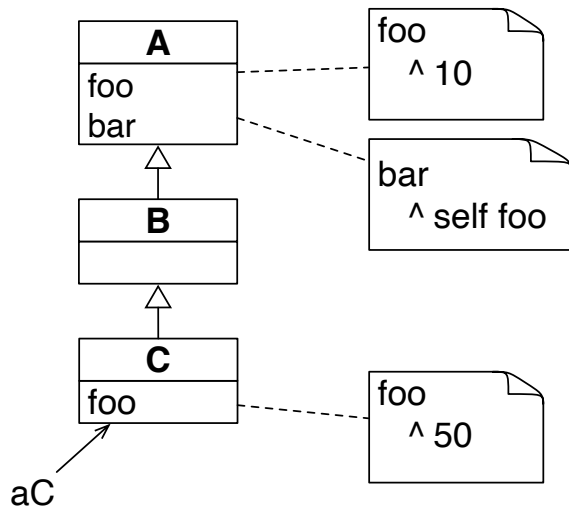


# self always represents the receiver



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

# self always represents the receiver



```
> aA bar
10
> aB bar
10
> aC bar
50
```

# What you should know

- self always represents the receiver
- Sending a message is a **two-step** process:
  1. **Look up** the method matching the message
  2. Execute this method **on the receiver**
- Method lookup maps a message to a method
- Method lookup starts in the **class of the receiver**
  - ...and goes up in the hierarchy



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/>