Inheritance and Lookup

2: Lookup

Stéphane Ducasse and Damien Cassou

http://stephane.ducasse.free.fr/ stephane.ducasse@inria.fr damien.cassou@inria.fr

Goal

- The algorithm mapping a message to a method
- `self` represents the receiver

Single Inheritance

- Static for the instance variables
  - at class-definition time, the instance variables are collected from the superclasses and the class. No duplication of instance variables.
- Dynamic for the methods
  - late binding (all virtual) methods are looked up at runtime depending on the dynamic type of the receiver.

Message Sending

Sending a message means *looking up* the method to execute in the class of the receiver and *executing* it on the receiver with the arguments.
Sending a message is a two-step process:

1. look up the method whose name matches the message selector;
2. execute this method on the receiver with the arguments.

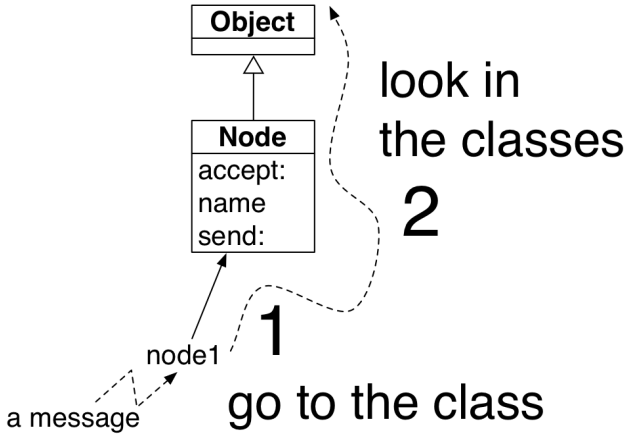
Let's present step 1 now.

Method Lookup

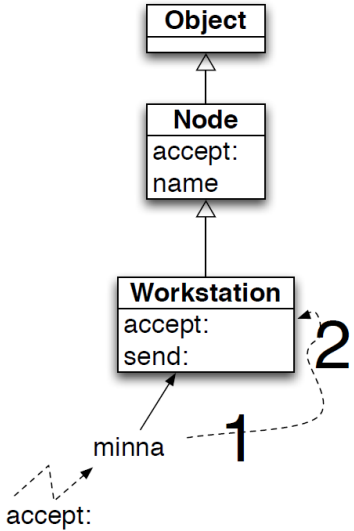
The lookup starts in the *class* of the *receiver* then:

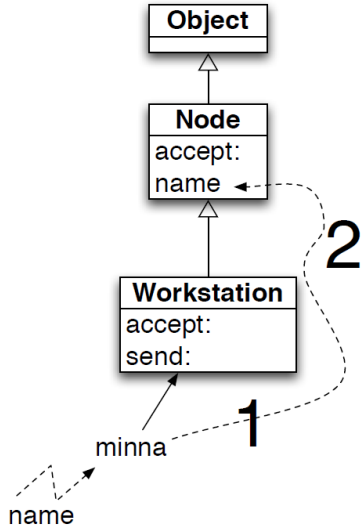
- if the method is defined in the class, it is returned;
- otherwise the search continues in the superclasses;
- when there is no more superclass... (explained later :-))

Method Lookup

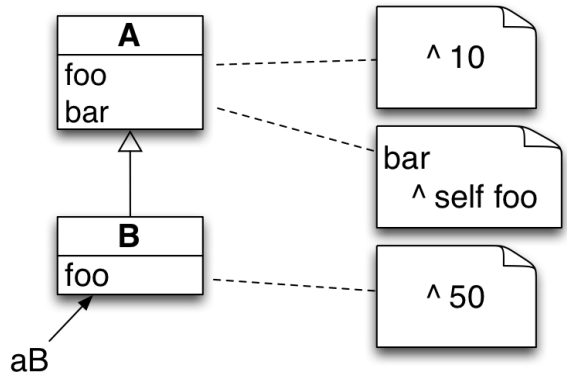


Some Lookup Cases



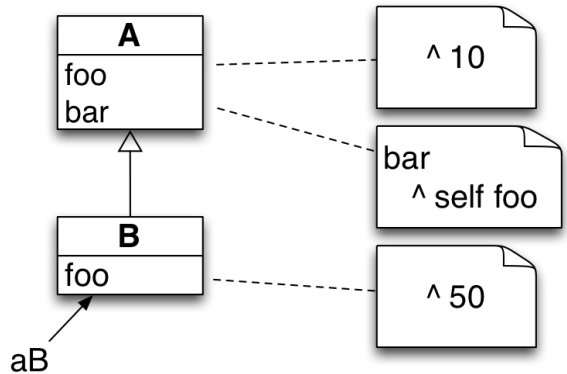


Method Lookup Starts in the Receiver Class




```
A new foo
> ...
B new foo
> ...
```

Method Lookup Starts in the Receiver Class




```
A new foo
> 10
B new foo
> 50
```

What is self/this?

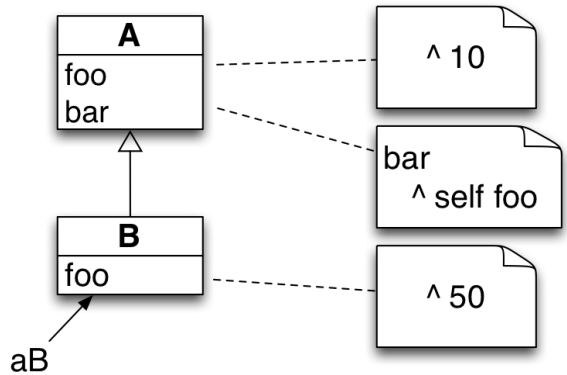
Take 5 min and write the definition of `self` (`this` in Java).

- Your definition should have two points:
  - What does `self` represent?
  - How are the methods looked up when a message is sent to `self`?

self/this

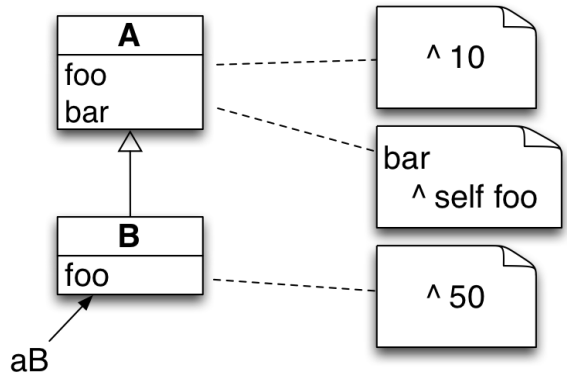
- `self` represents the receiver of the message
- `self` in Pharo, `this` in Java, C#
- the method lookup starts in the class of the receiver

self represents the receiver




```
A new bar
> ...
B new bar
> ...
```

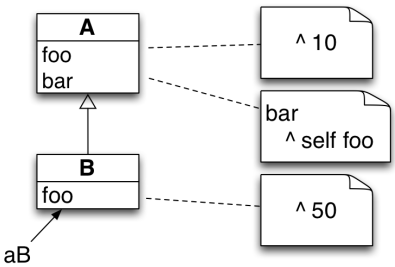
self represents the receiver




```
A new bar
> 10
B new bar
> 50          <-- discussed on the next slide
```

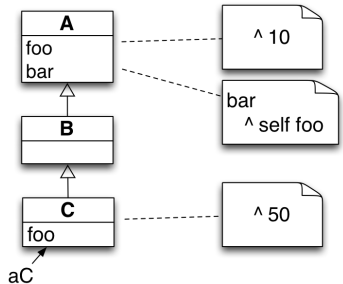
self represents the receiver

```
B new bar
> 50
```



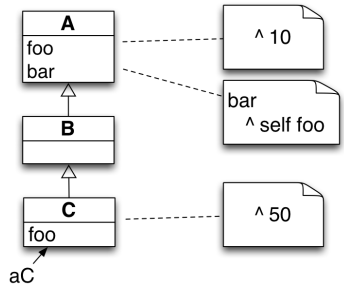
1. aB bar : no method bar defined in B
2. look up in A - the method bar is found
3. method bar is executed on the receiver (aB = B new)
4. self refers to aB
5. foo is sent to self
6. look up foo in the receiver's class: B !
7. foo is found there and executed on aB

self Always Represents the Receiver




```
A new bar
> ...
B new bar
> ...
C new bar
> ...
```

self Always Represents the Receiver




```
A new bar
> 10
B new bar
> 10
C new bar
> 50
```

What you should Know

- `self` represents the receiver
- Sending a message is a two-step process:
  1. look up the method whose name matches the message selector;
  2. execute this method on the receiver with the arguments.