

# Ruby Beyond the Basics

Ruby is an Object Oriented Language



John Cinnamonond

@jcinnamonond | [panagile.com](http://panagile.com)

---

Rust

Javascript

Simula

io

Objective-C

C#

Python

C++

Java

Ruby

Scala

Smalltalk

Javascript

Rust

Objective-C

Simula

io

C++

Python

C#

"I can tell you that C++ wasn't

what I had in mind"

Ruby

Alan Kay, OOPSLA 1997

Smalltalk

Scala

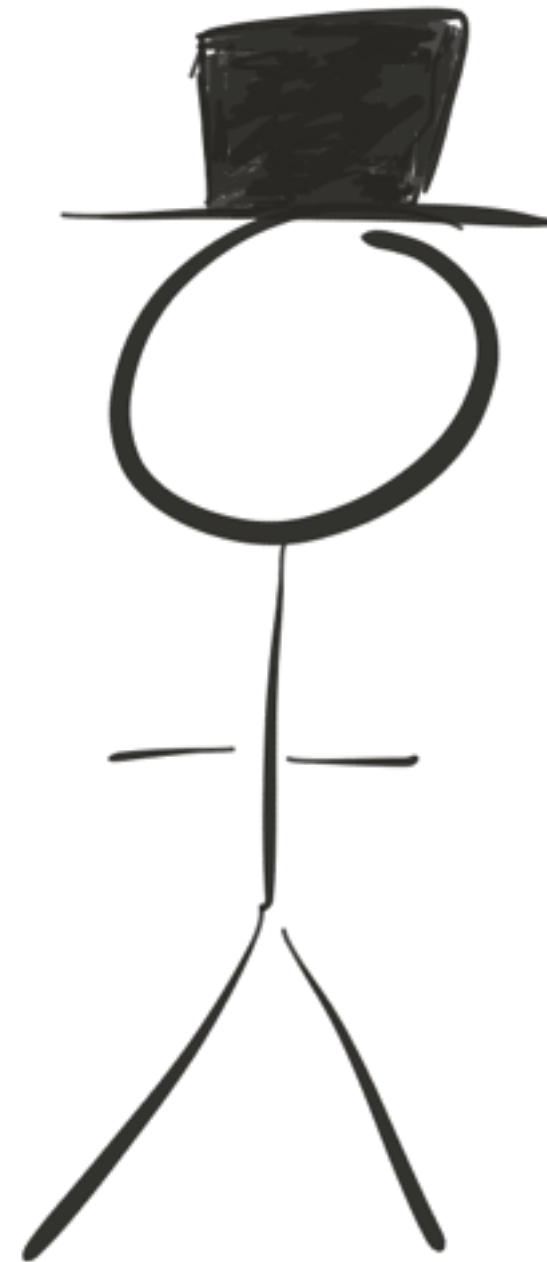
# Ruby

## Object Oriented

# Local State



Mary

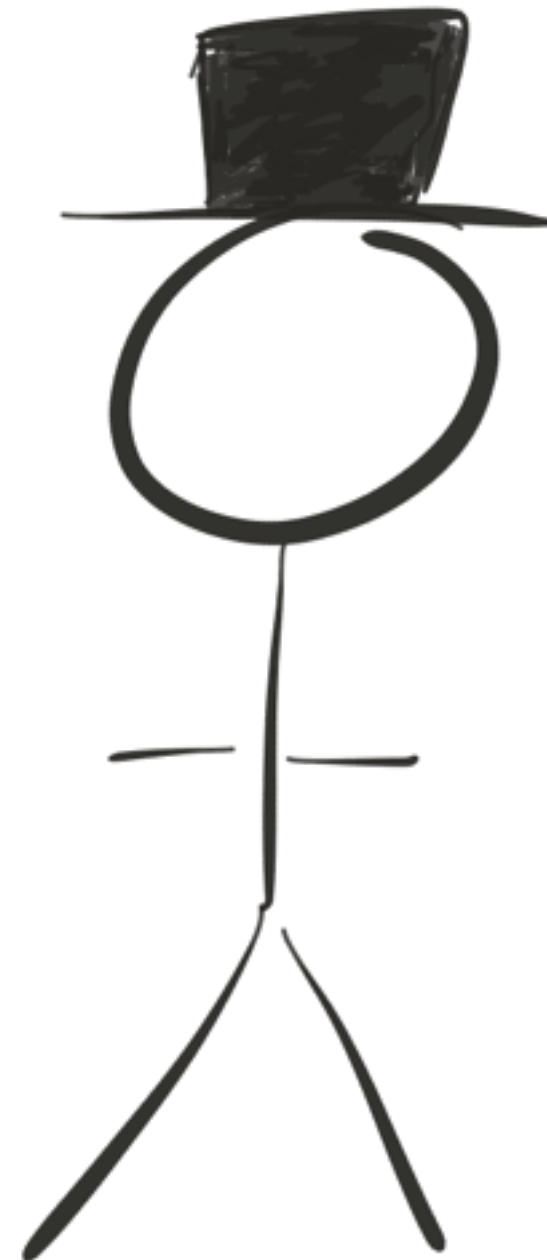


Bob

# Local State



Mary

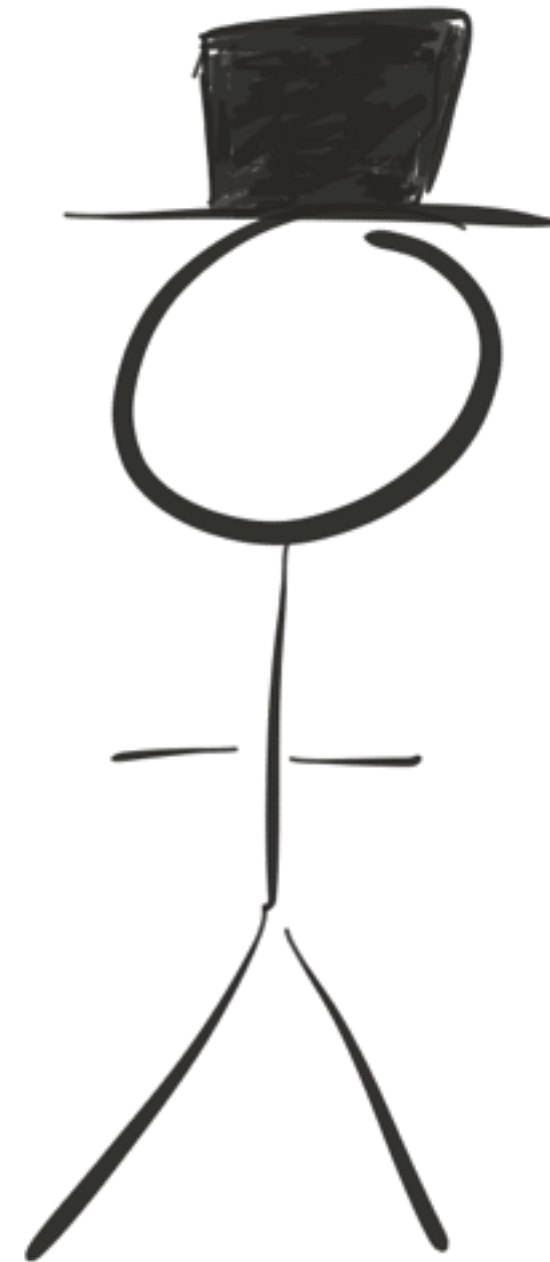


David

# Local State



Mary

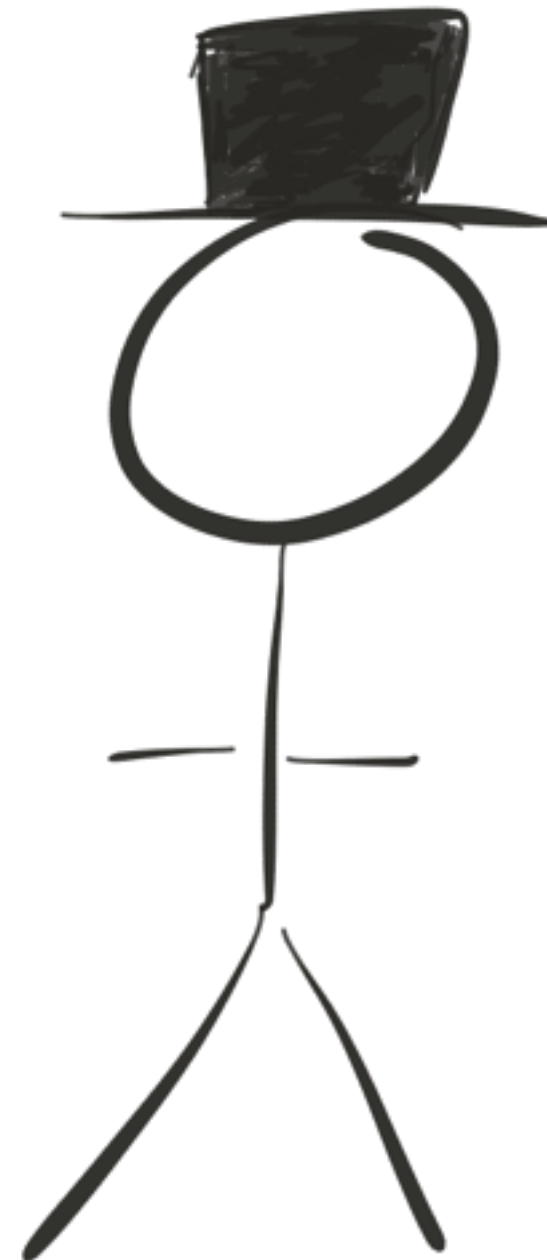


Mary

# Local State



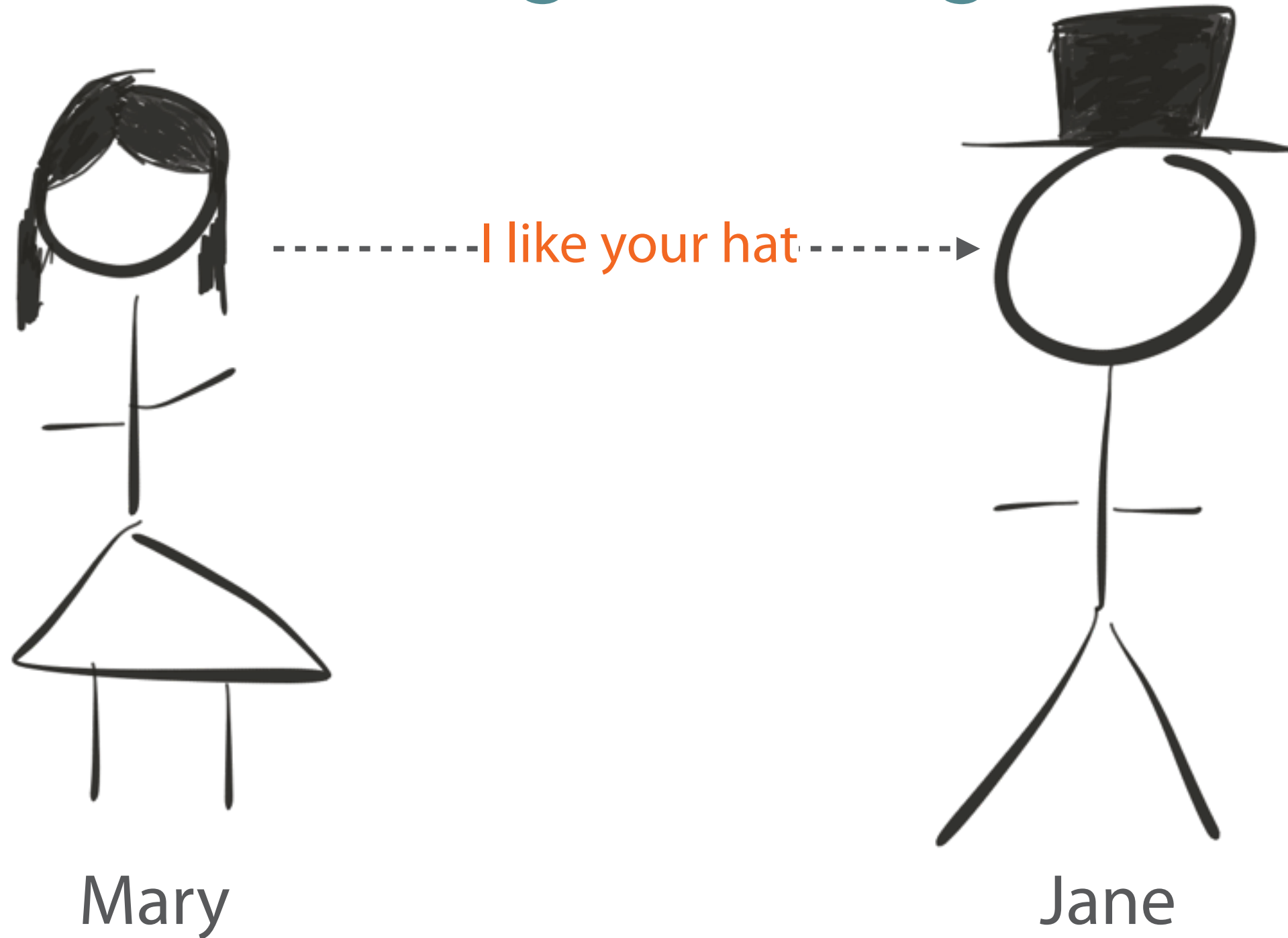
Mary



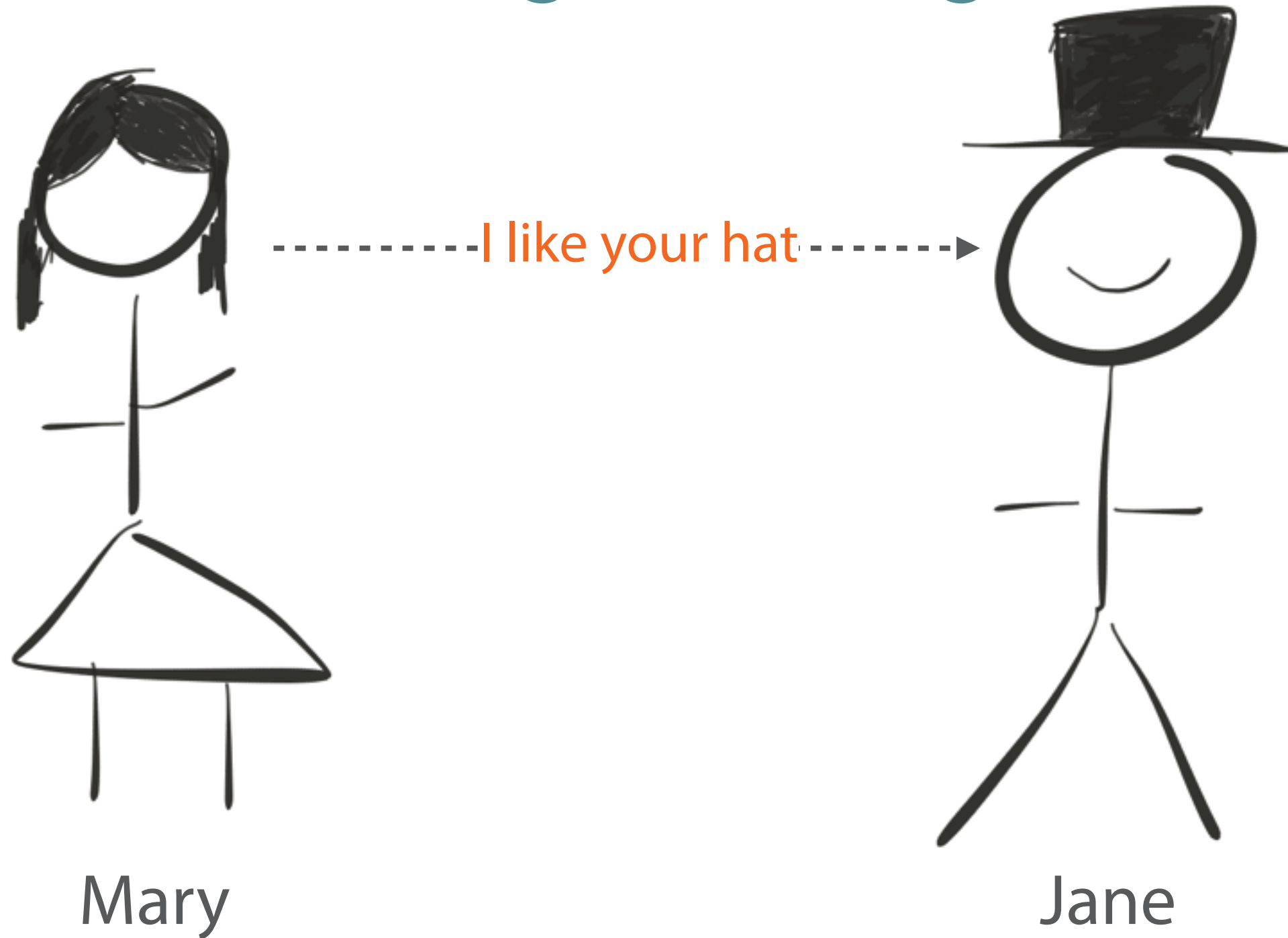
Mary



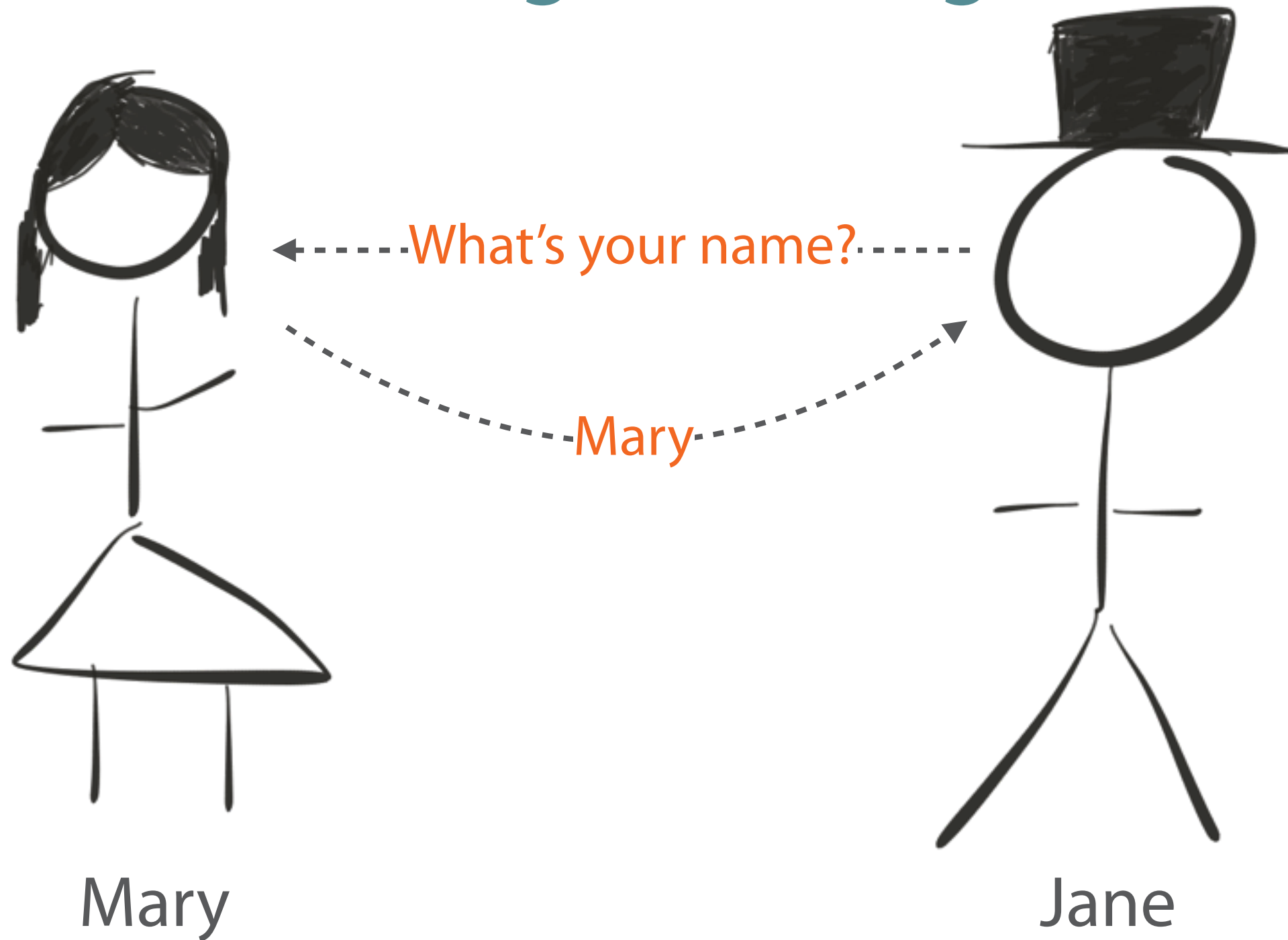
# Message Sending



# Message Sending



# Message Sending



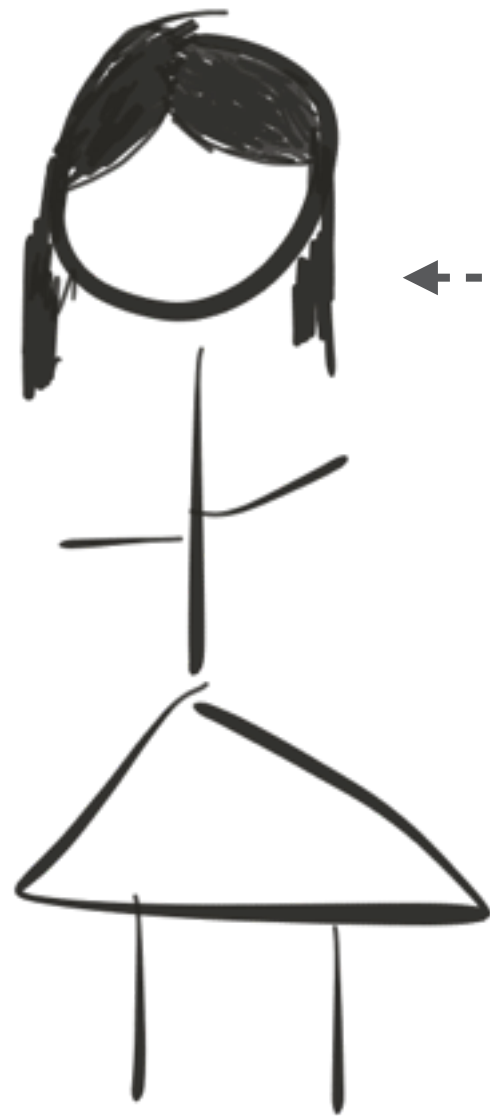
Messages == Calling Methods



Mary

Change your name  
to Jane





Mary

Change your name  
to Jane





Mary

Change your name  
to Jane





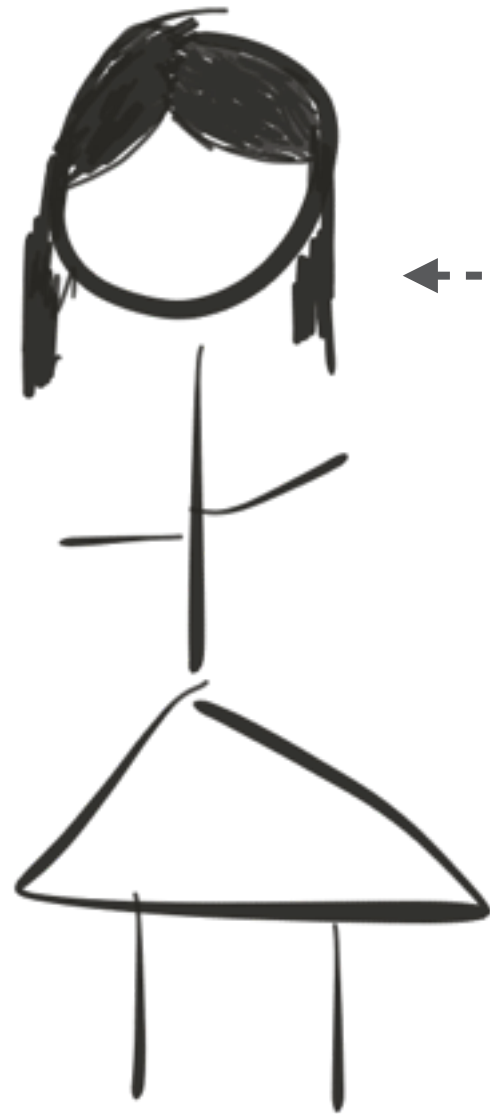
Mary

Change your name  
to Jane

You can't tell  
me what to do  
(Exception)



?



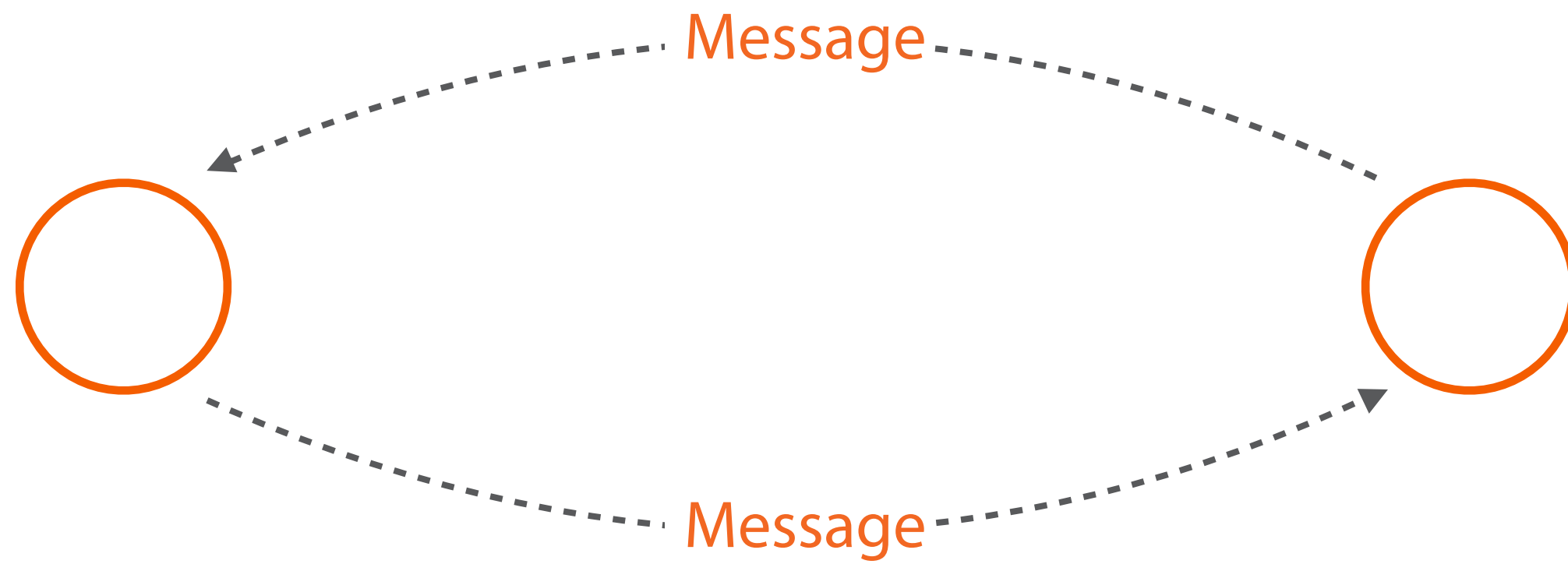
Mary

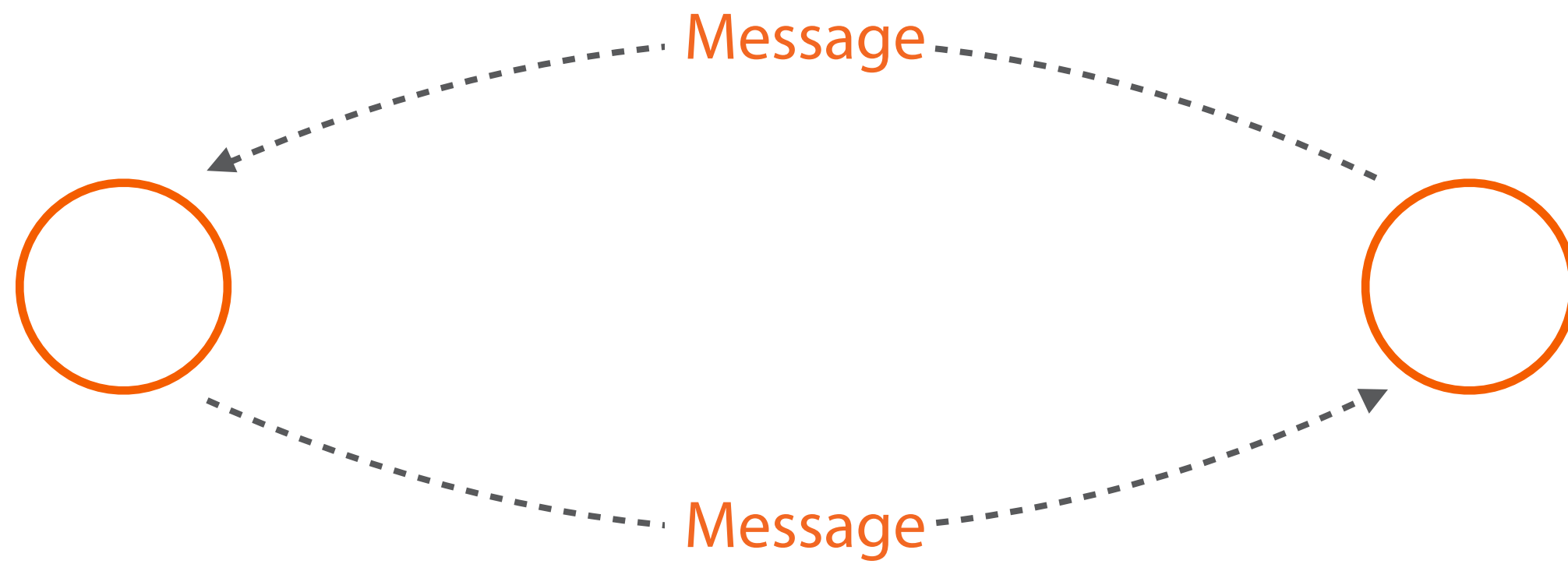
Change your name  
to Jane

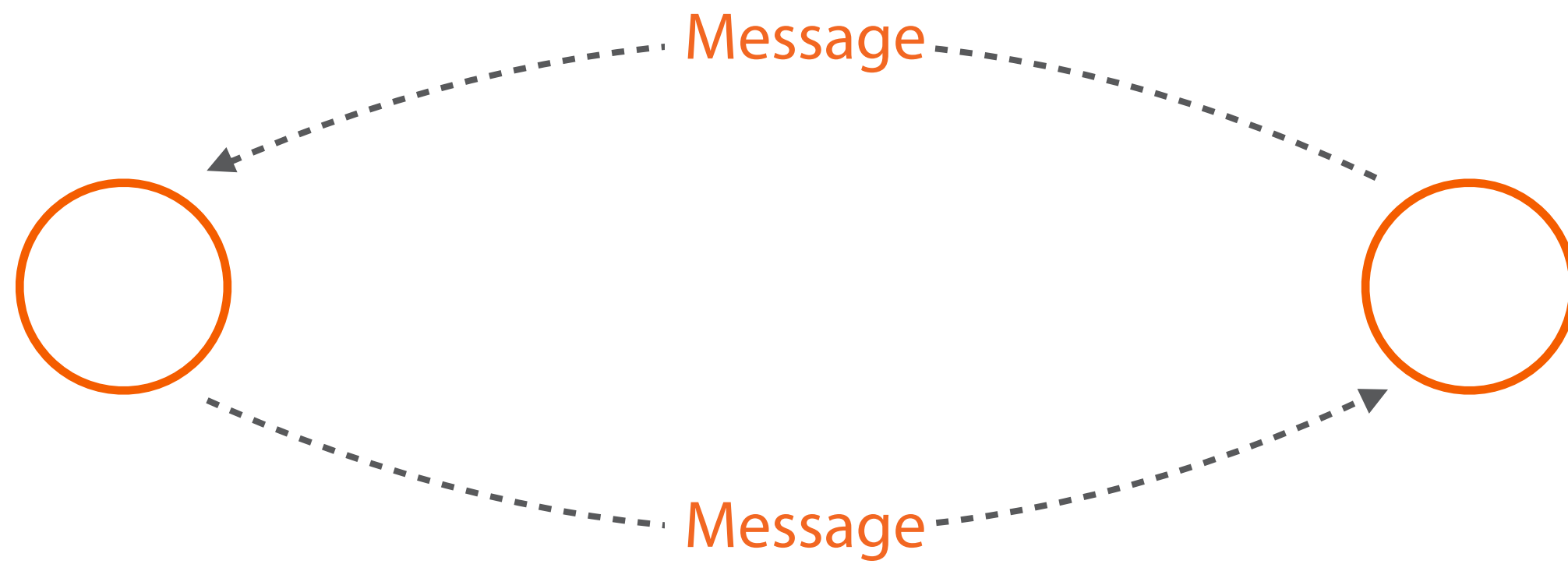




NoMethodError







---

# Everything is an Object

---

# Object

Local state

Collection of objects

Responds to messages



# Object

A collection of references to other objects  
(local state)  
that responds to certain messages

---

# Special Objects

---

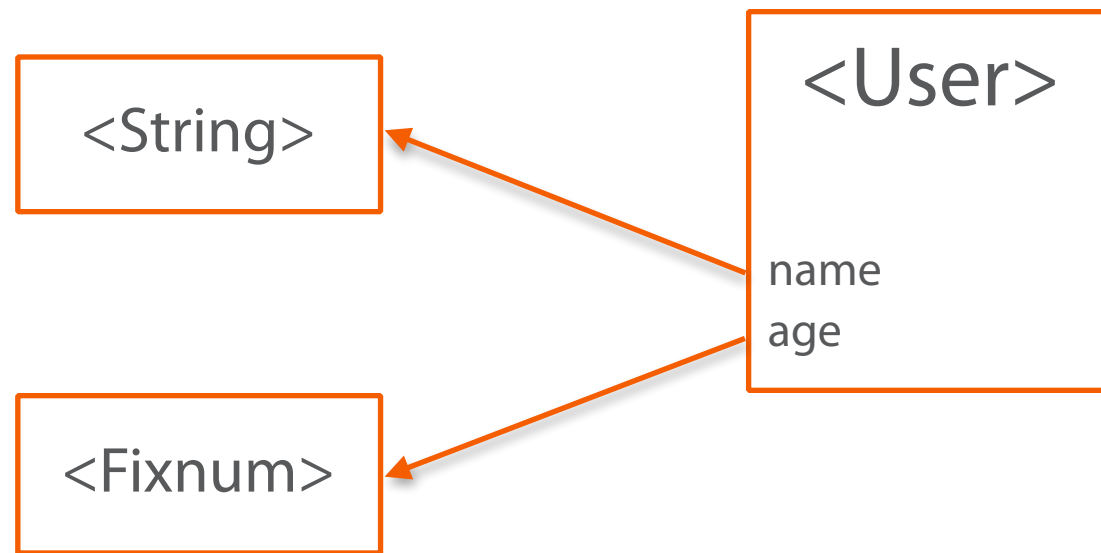
Local state

Messages

Object



```
graph TD; LS[Local state] --- O[Object]; M[Messages] --- O;
```



<Fixnum>



37

<Fixnum>



37

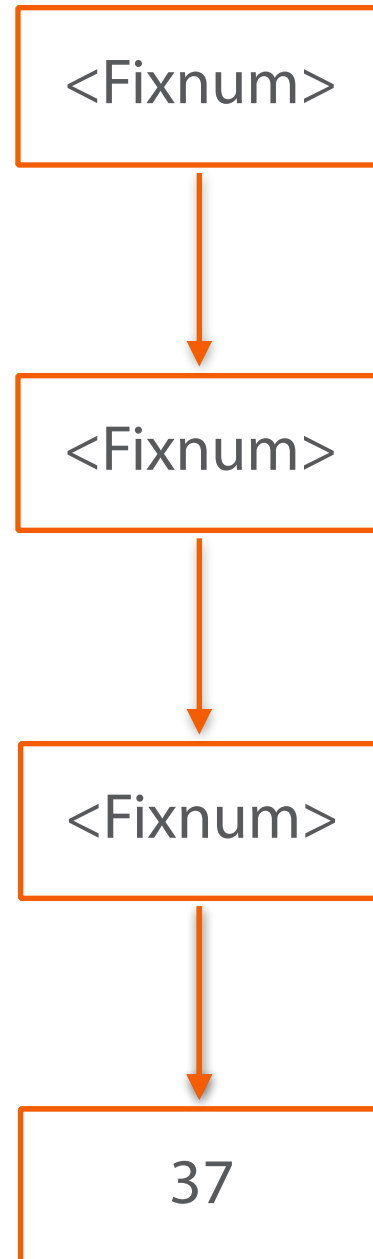
<Fixnum>



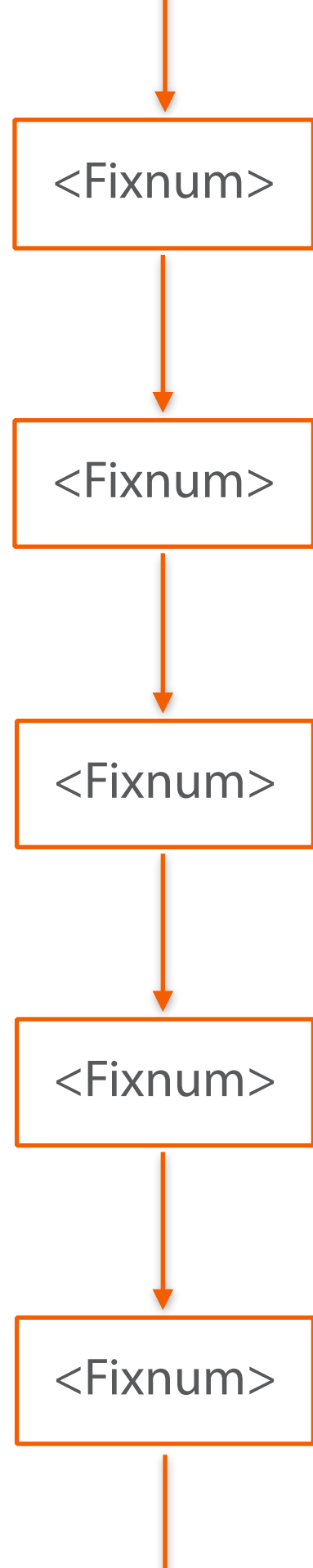
<Fixnum>

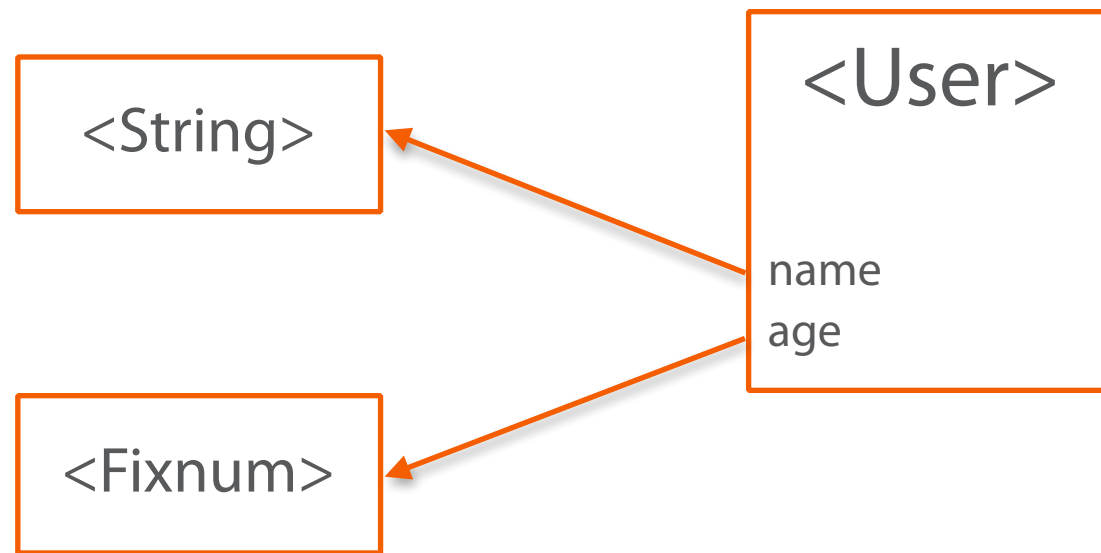


37





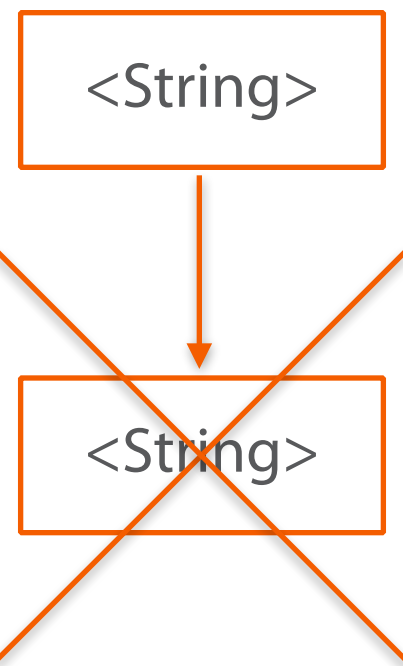




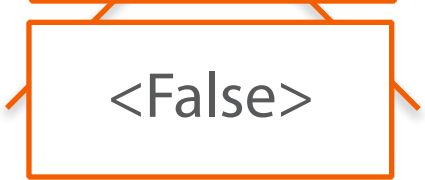
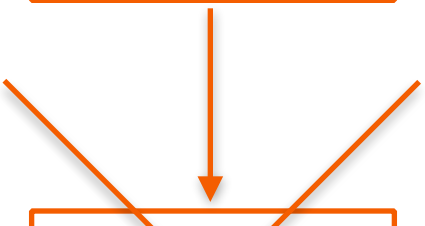
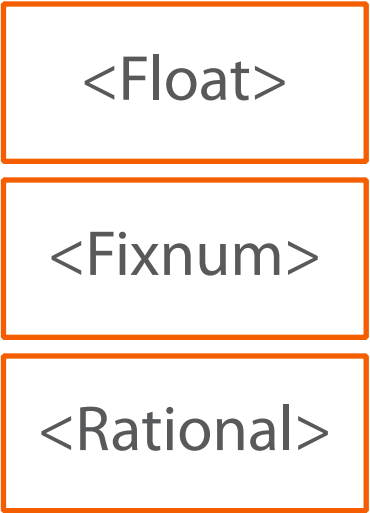
<String>



<String>

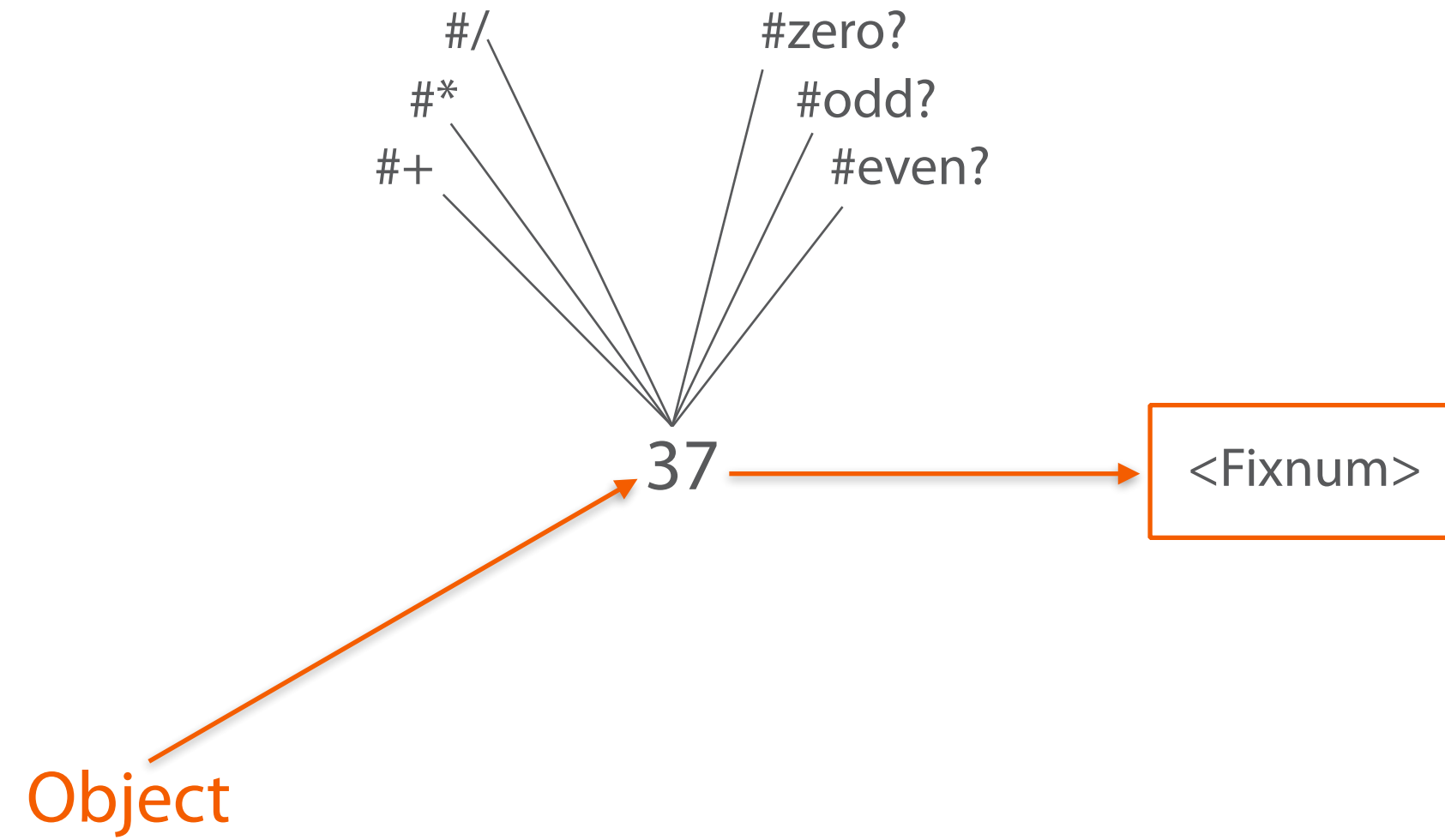


Stop it!



Stop it!

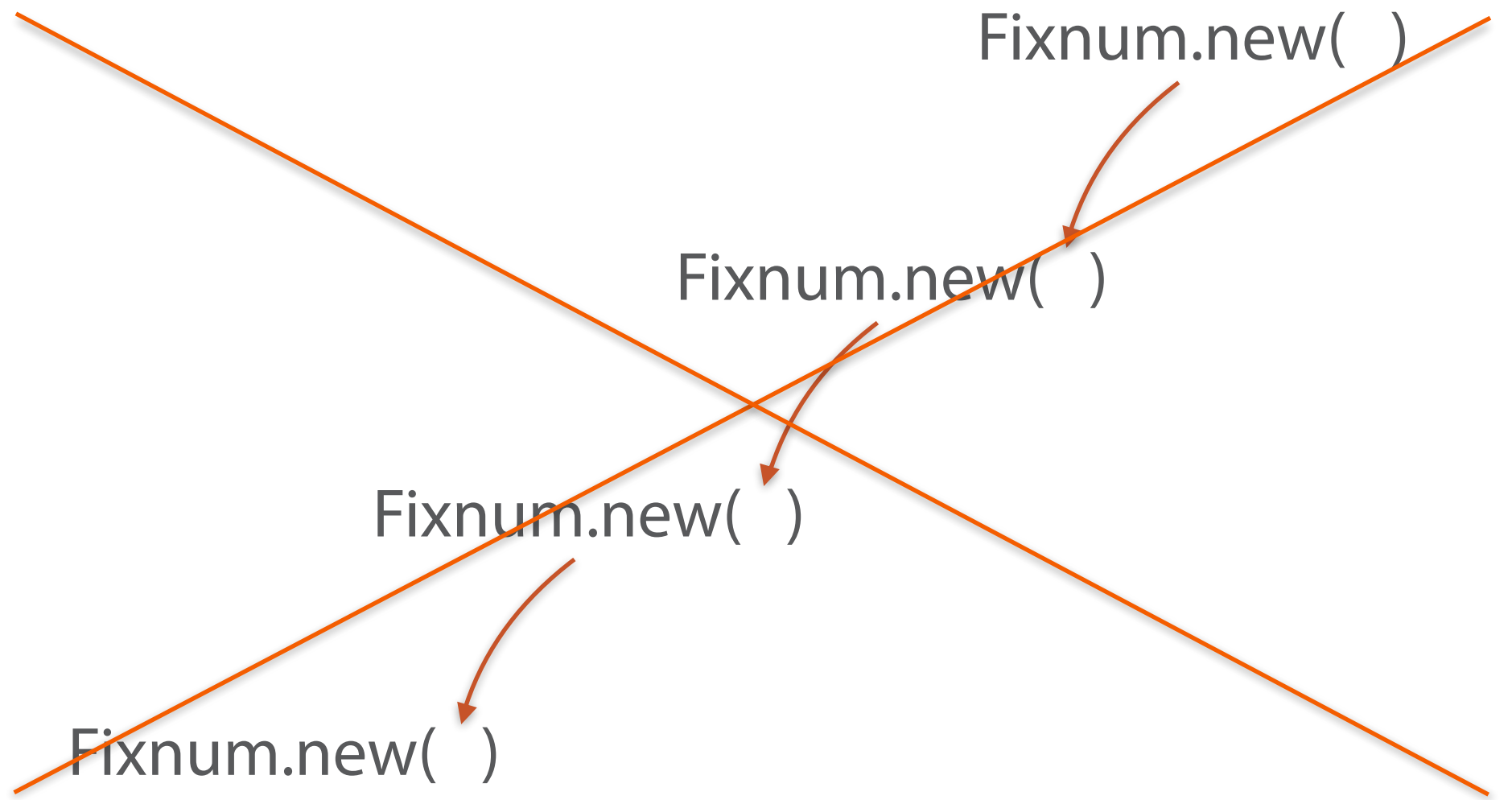




```
object = Class.new(...)  
num = 42
```



<Fixnum>



Stop it!



---

# Duck Typing

---



# Quack

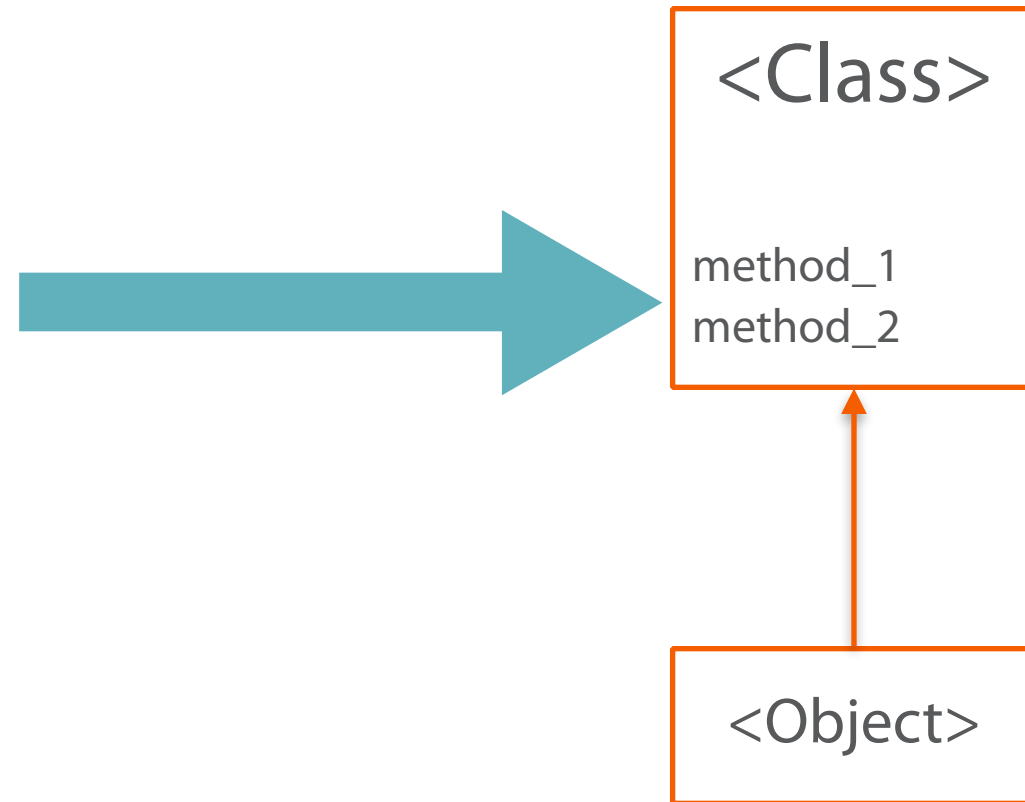
If it walks like a duck  
and quacks like a duck  
it probably is a duck

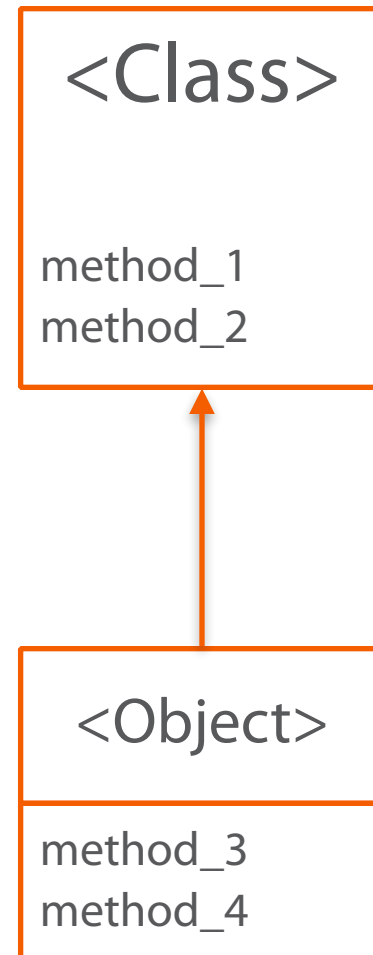
MacBook Air

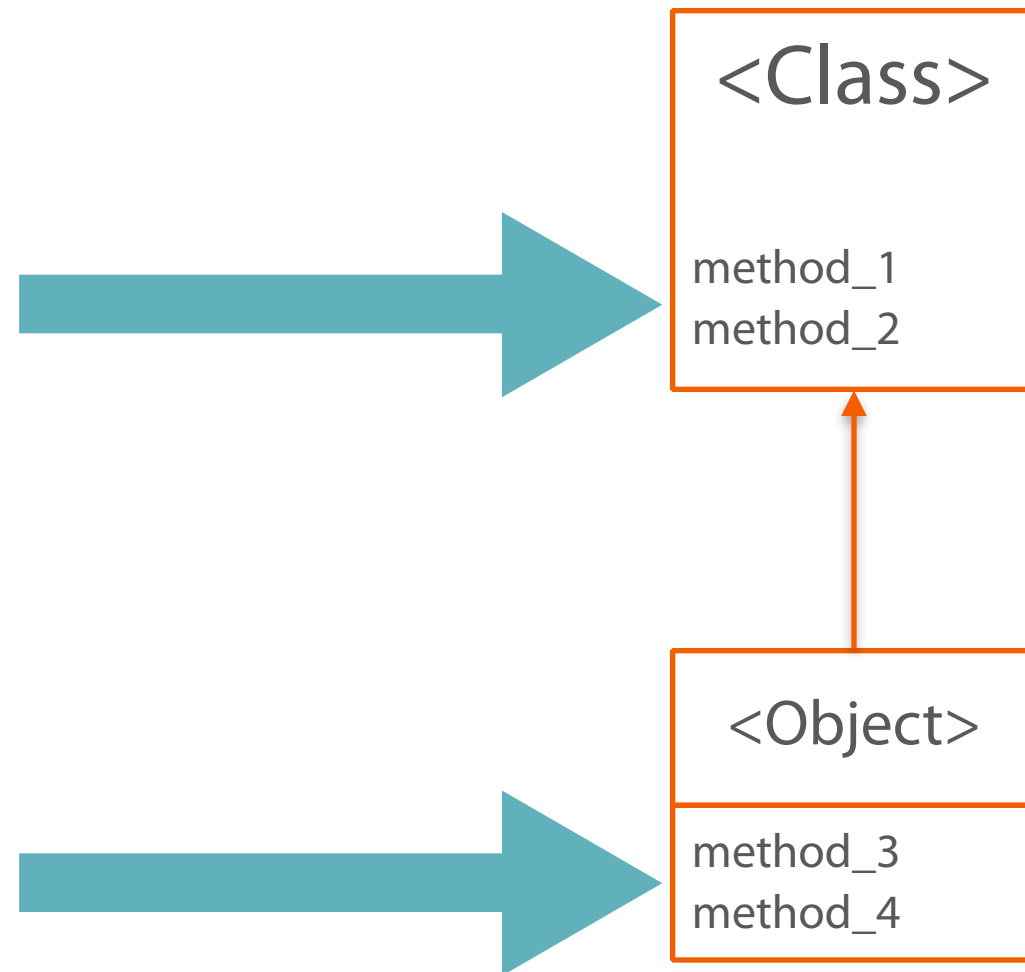
---

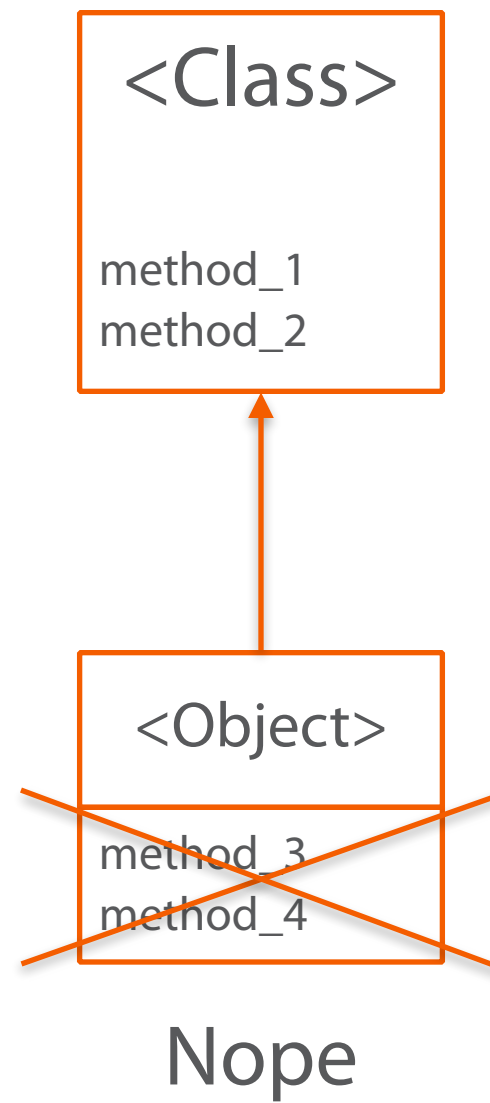
# Extend

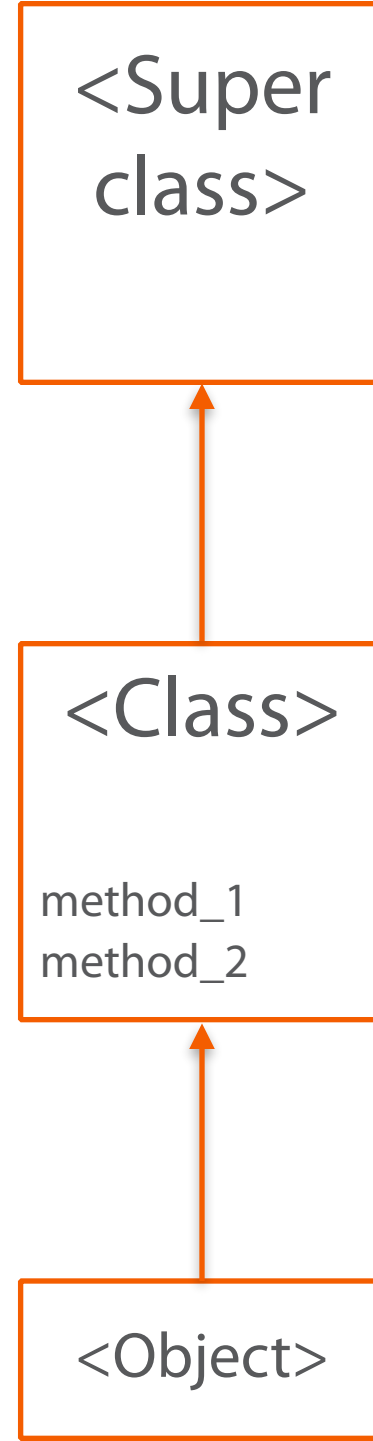
---



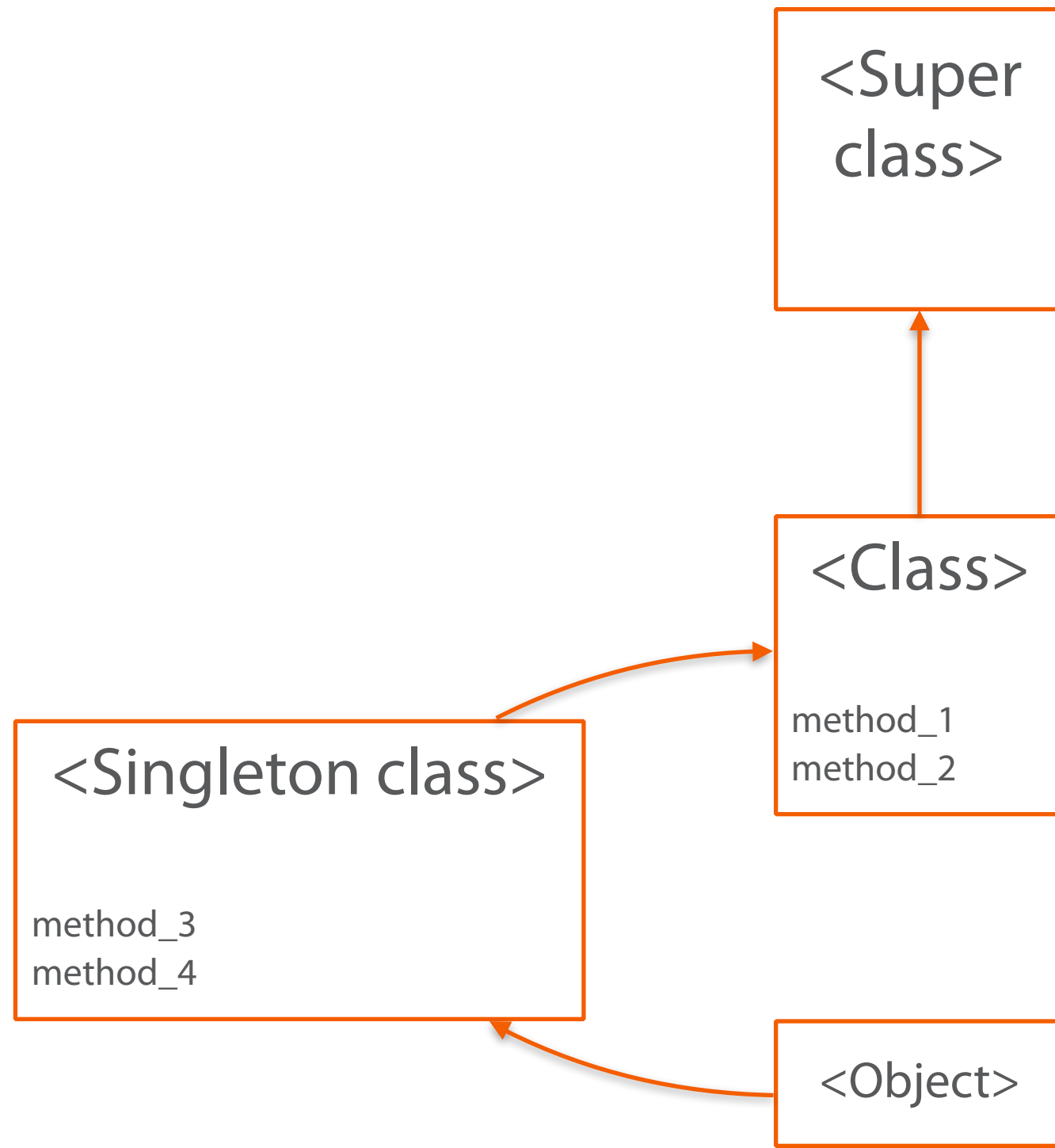


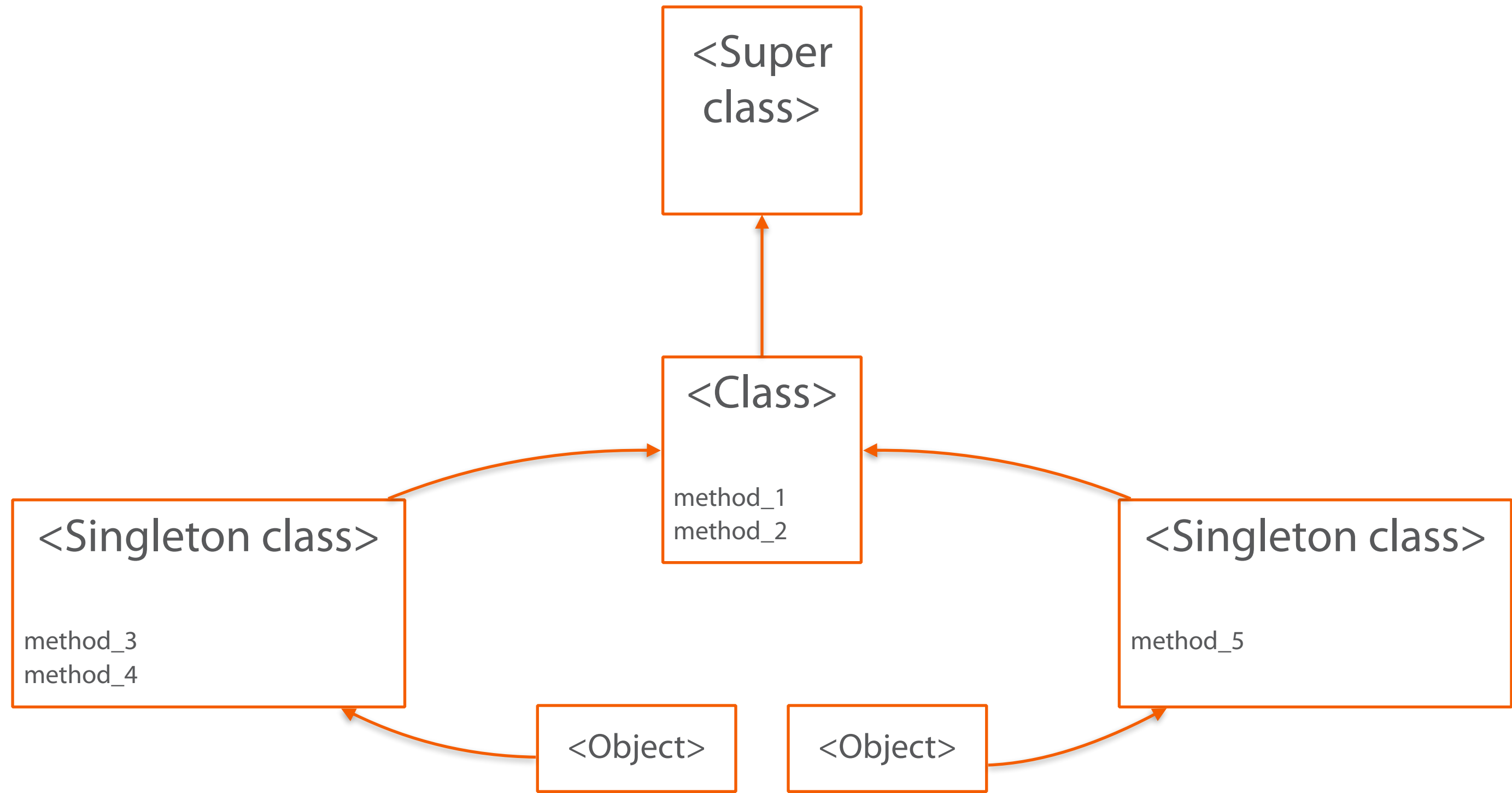










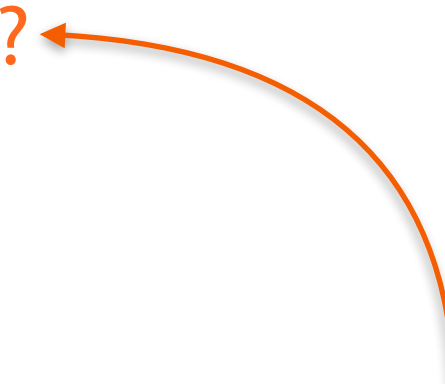


---

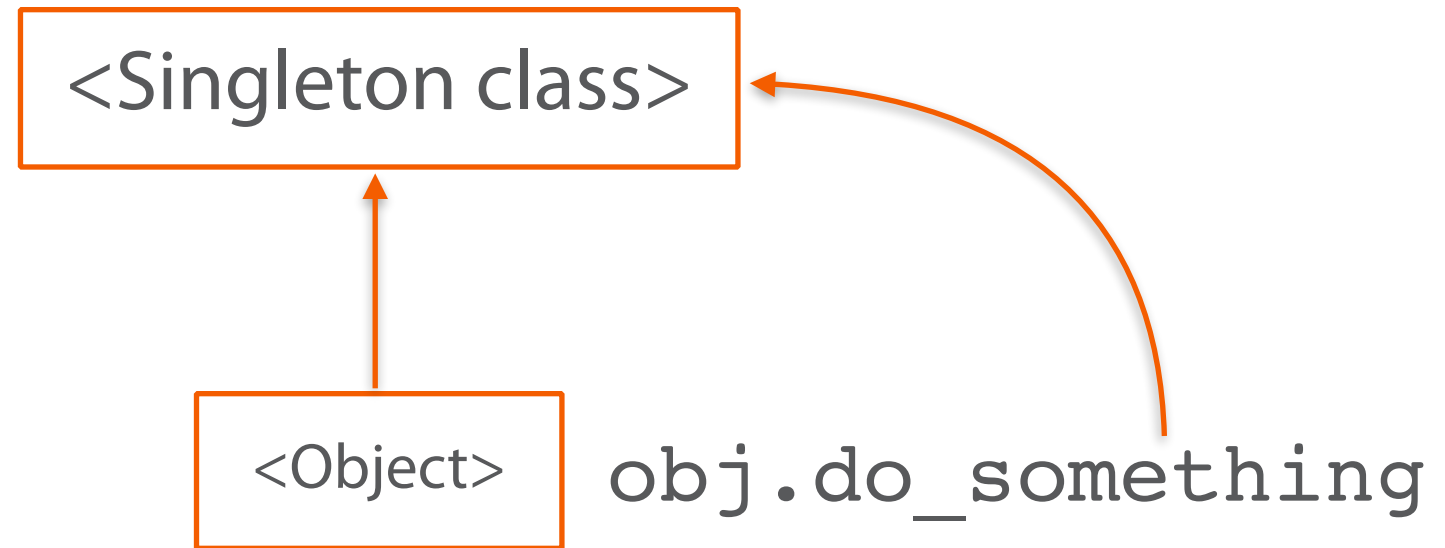
# Method Dispatch

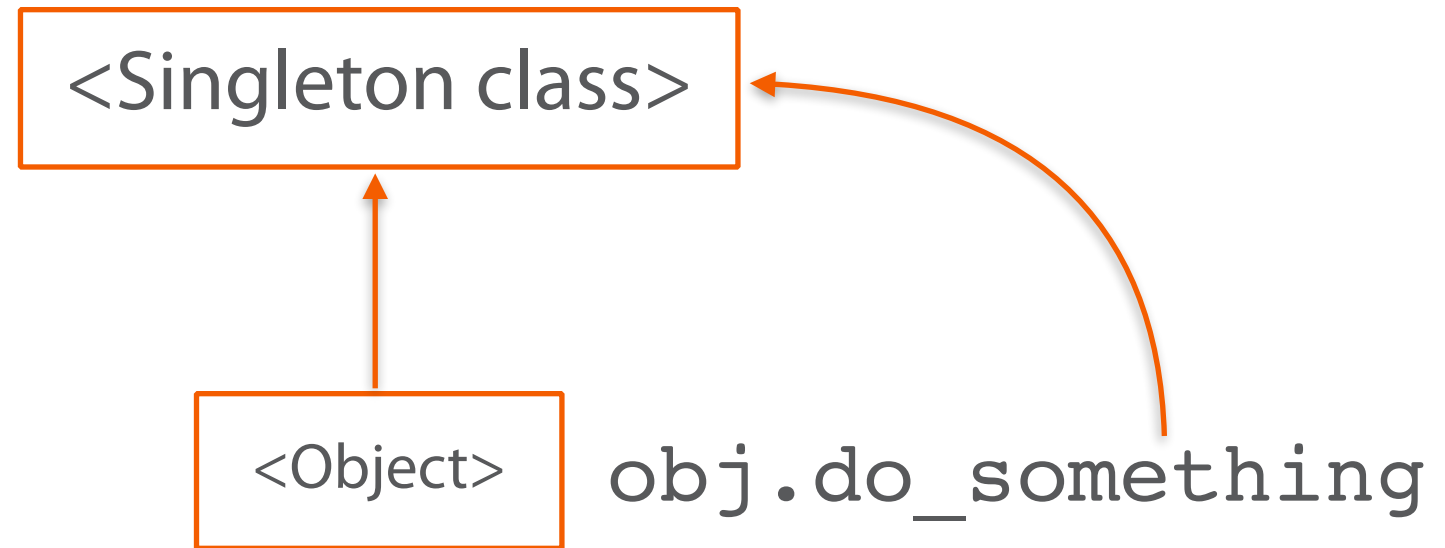
---

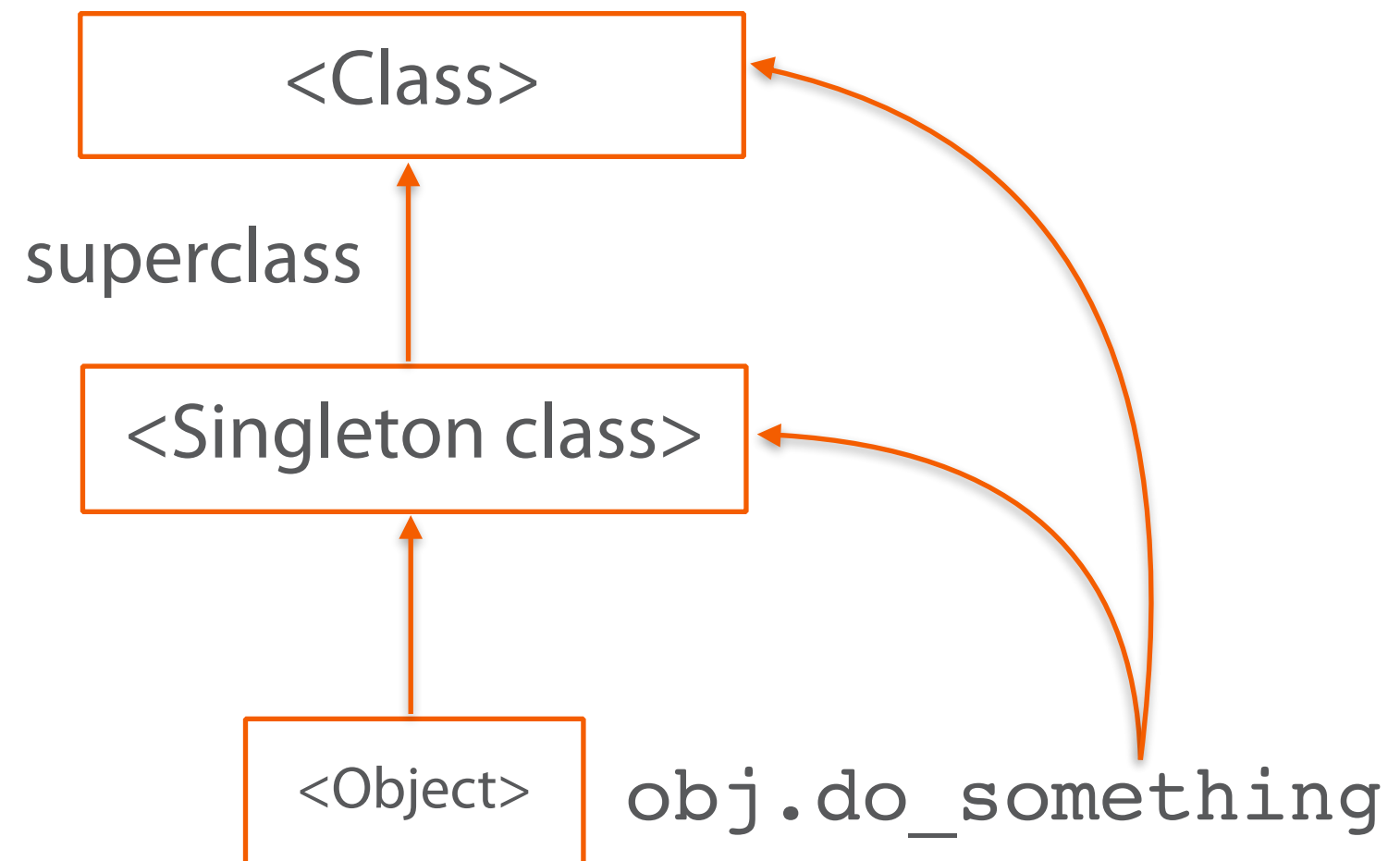
?

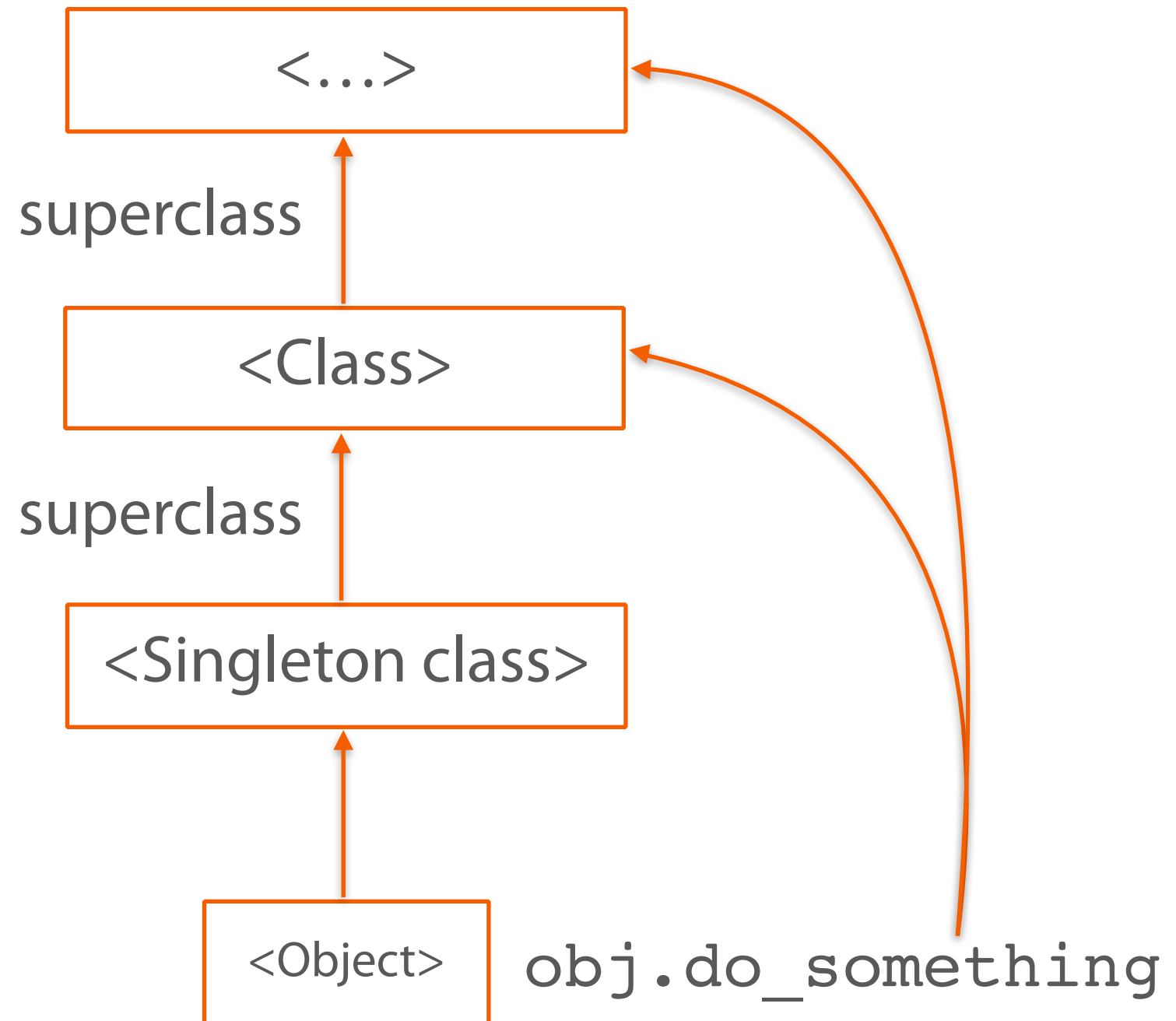


obj.do\_something

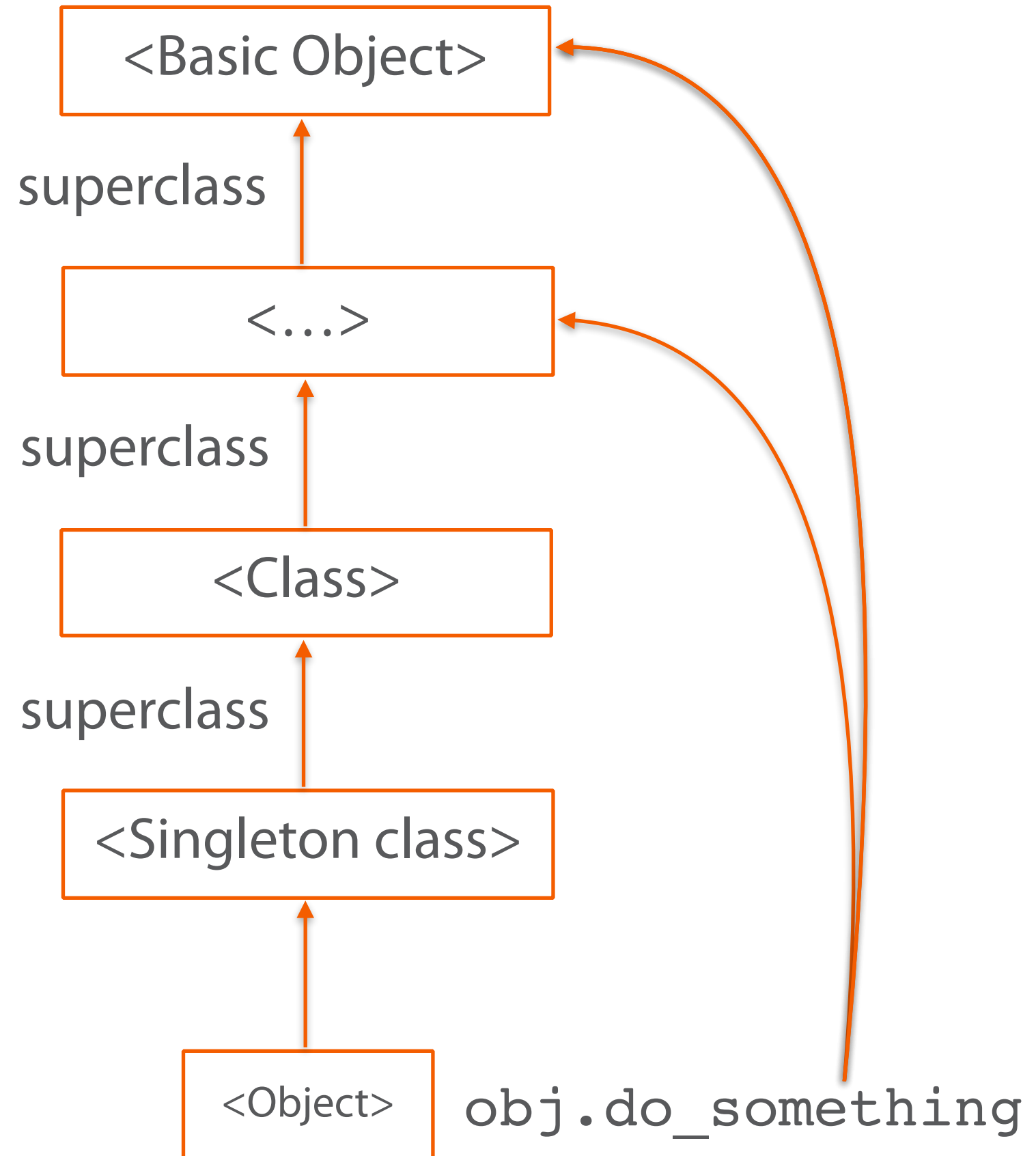


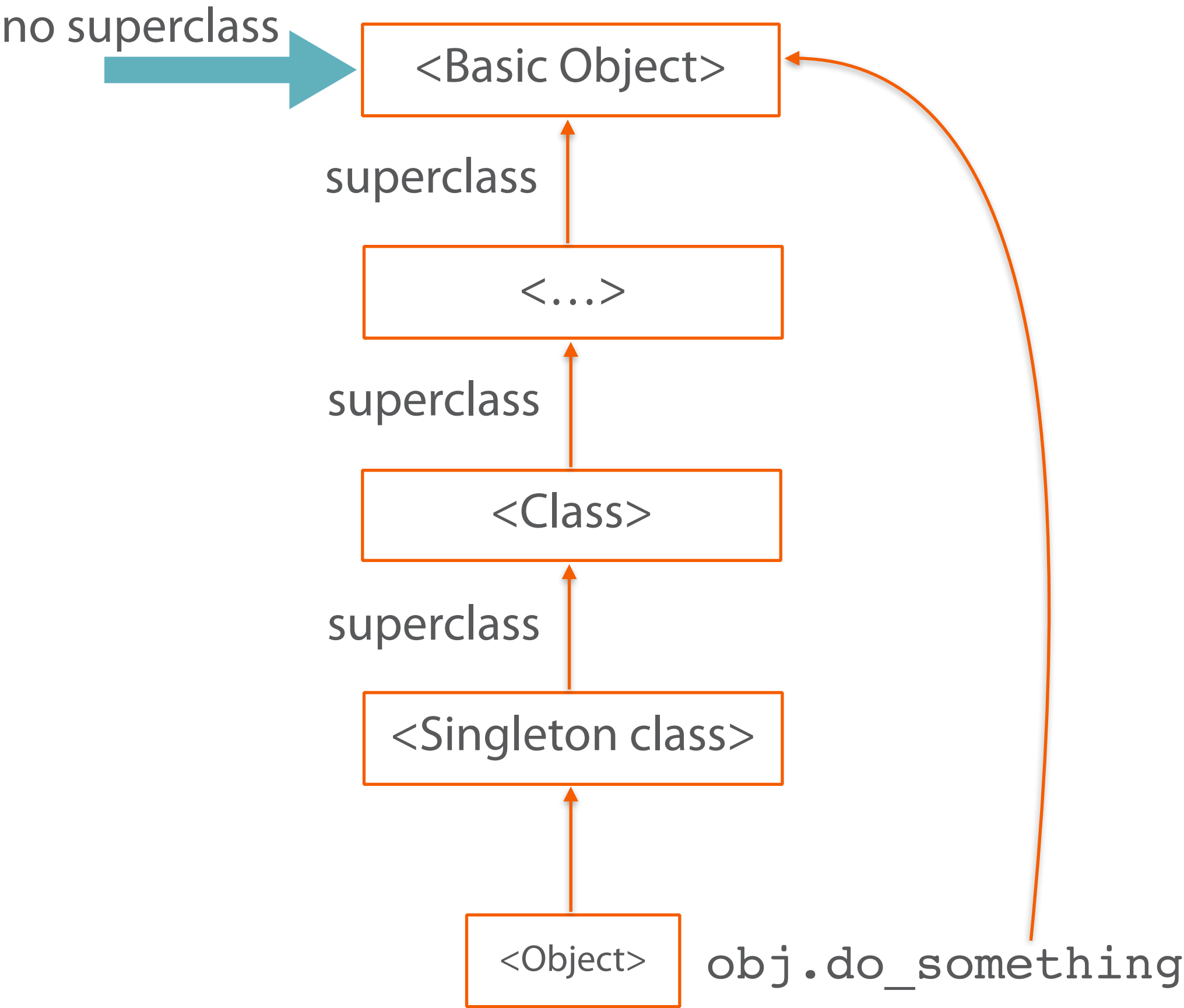


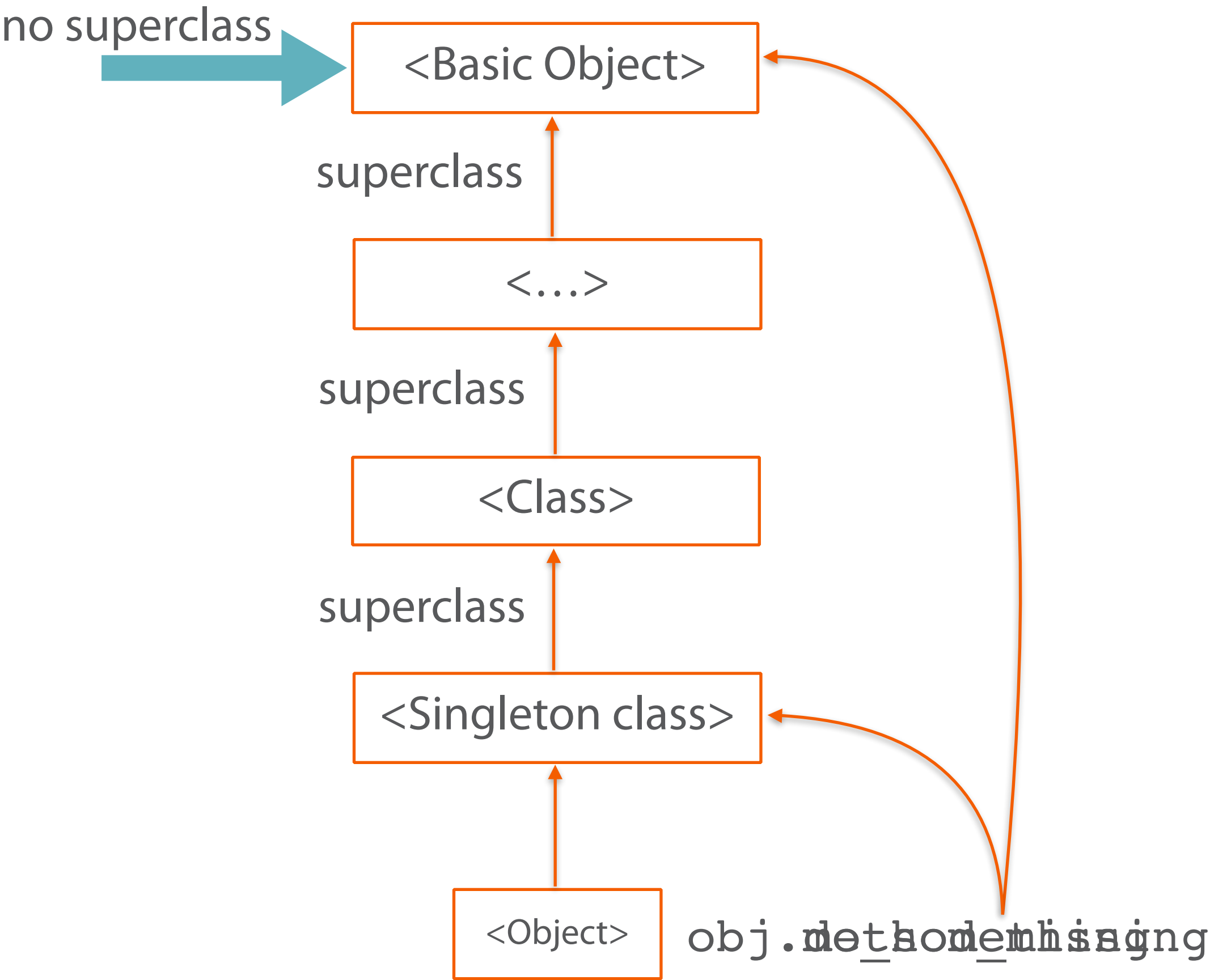












no superclass



<Basic Object>

superclass



<...>

superclass



<Class>

superclass

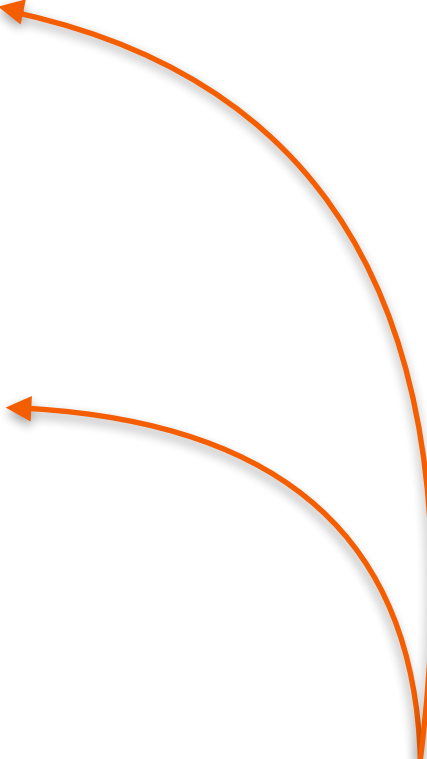


<Singleton class>



<Object>

`obj.method_missing`  
`do_something`



no superclass



<Basic Object>

superclass



<...>

superclass



<Class>

superclass

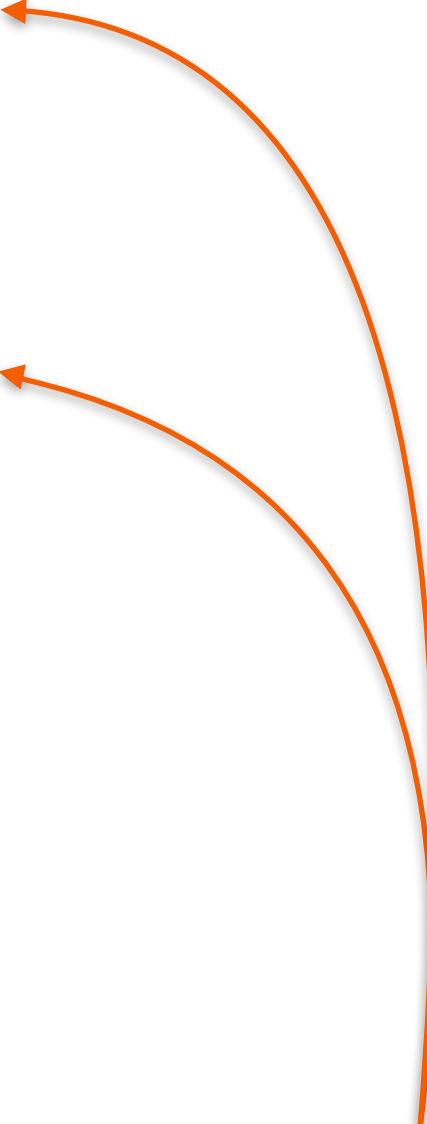


<Singleton class>



<Object>

`obj.method_missing`  
`do_something`



no superclass



<Basic Object>

superclass



<...>

superclass



<Class>

superclass



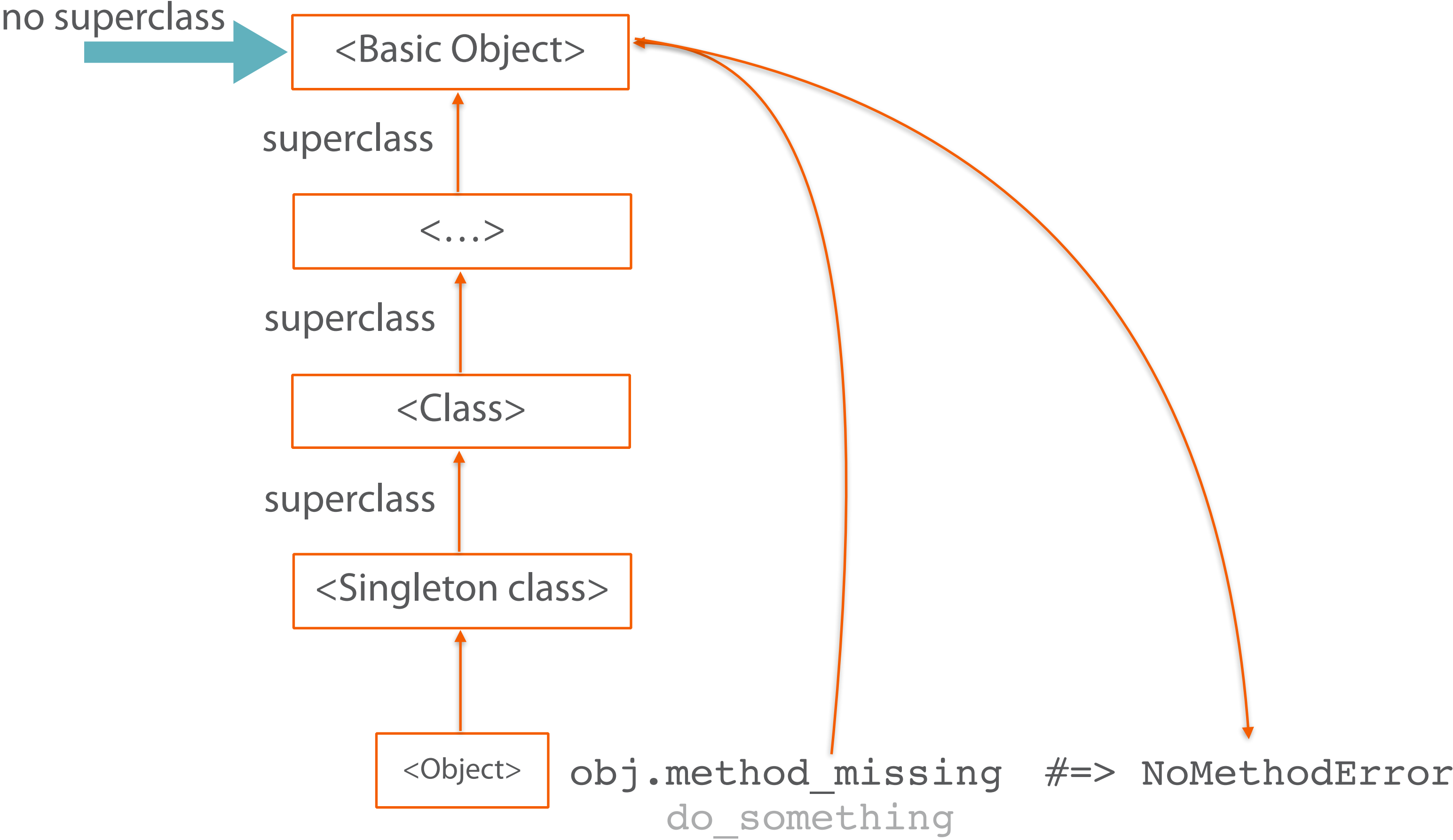
<Singleton class>

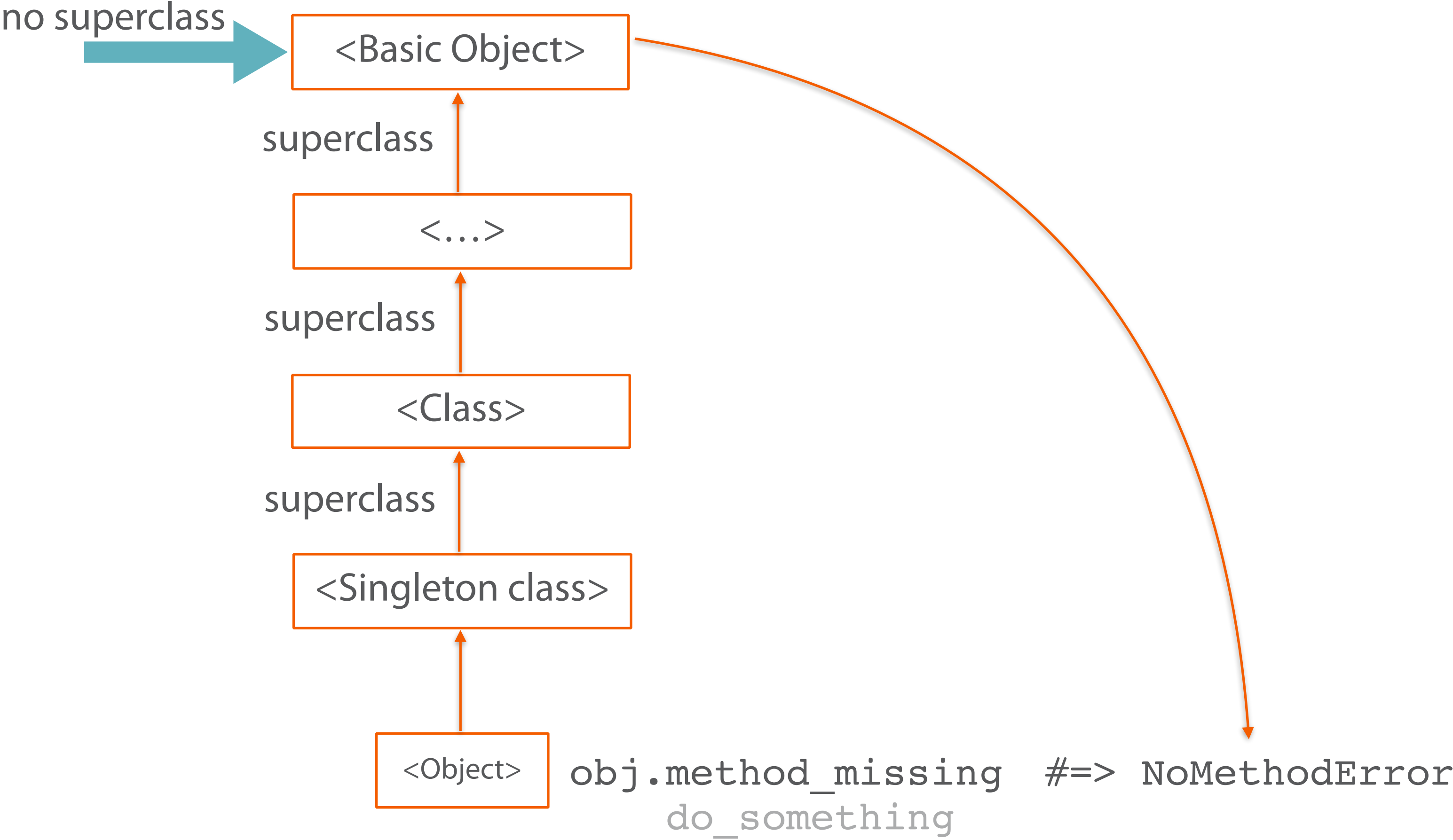


<Object>

`obj.method_missing`  
`do_something`









```
method_missing(name, *args, &block)
```

```
method_missing(name, *args, &block)
```

```
some_method(a, b, c)  some_method([a, b, c])
```