Class methods

Stéphane Ducasse

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

In a nutshell

- There is no difference between instance and class methods.
  - public
  - virtual / late-bound
  - return self by default
- Classes are just objects too.
- Class methods are resolved *exactly* the same way than instances methods.

Some examples

```
Time now
> 10:44:16.10794 am
```

```
Date today
> 29 July 2015
```

Some more

```
(FileLocator workingDirectory filesMatching: '*.jpg')
```

```
DateAndTime fromUnixTime:
  ((ByteArray readHexFrom: 'CAFEBABE4422334400FF')
     copyFrom: 5 to: 8) asInteger
```

```
(ZnEasy getPng: 'http://pharo.org/web/files/pharo.png')
   asMorph openInWindow
```

```
(ZnServer startDefaultOn: 8080)
  onRequestRespond: [ :request |
    ZnResponse ok: (ZnEntity with: DateAndTime now printString) ]
```

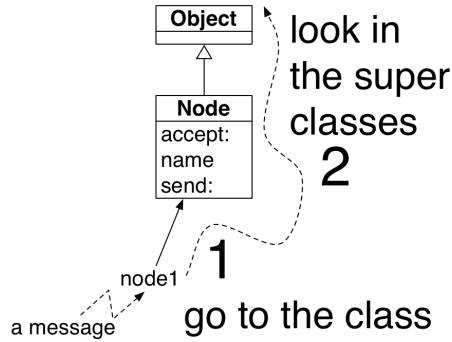
Isn’t simply natural?

- We just send messages to classes and they perform some actions
- Most of the time they create instances

Remember

Sending a message to an object

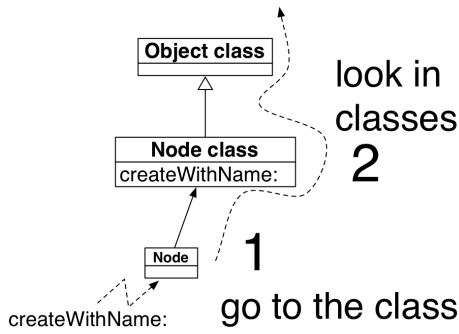
- go to the class of the object
- follow the inheritance chain
- apply the found method to the receiver



Sending a message to a class

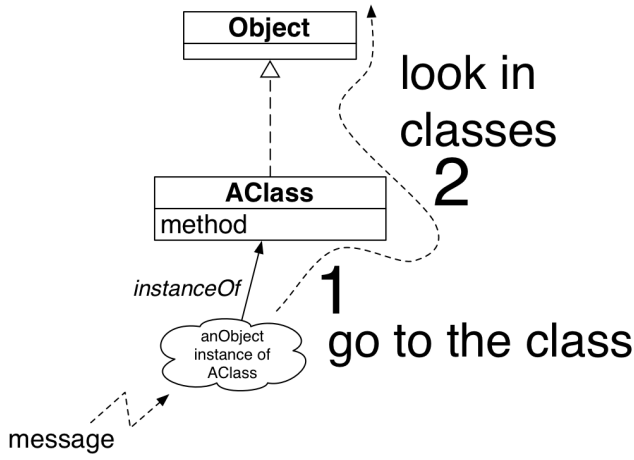
Sending a message to a class
(`Node`)

- go to the class of the class
  (`Node class`)
- follow the inheritance chain (up to `Object class` and upper)
- apply the found method to the receiver (the class `Node`)



Stepping back

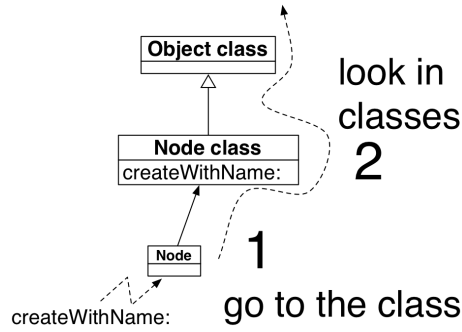
- There is only one way to execute messages!
- instance/class and class/superclass relations are systematically followed.



Class as objects

A class is an object instance of another class (called a metaclass)

- A metaclass is just one class whose instances are classes!
- `Point` is the unique instance of the class `Point class`
- THe `Point class` is automatically created when you create the class `Point`
- The class of class `XXX` is named `XXX class`



Common mistake

```
MyObject>>createWithName: aString
    self new name: aString
```

- `MyObject createWithName: 'pilou'` returns the class itself `MyObject`

Why?

```
MyObject>>createWithName: aString
    self new name: aString
```

Since there is no explicit return the method return `^ self`

```
MyObject>>createWithName: aString
    self new name: aString
    ^ self
```

- `self` here is the class `MyObject`

Correct way

We should just return the instance

```
MyObject>>createWithName: aString
    ^ self new name: aString
```

Conclusion

- Messages sent to classes are resolved the same way as messages sent to instances.
- Class methods are
  - public
  - late-bound
- Classes are instance of other classes (called metaclasses).